End-to-End Trust Fulfillment of Big Data Workflow Provisioning over Competing Clouds

Hadeel El-Kassabi

A Thesis

In

The Concordia Institute

for

Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Information and Systems Engineering) at

Concordia University

Montréal, Québec, Canada

May 2018

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Hadeel El-Kassabi**

Entitled: **End-to-End Trust Fulfillment of Big Data Workflow Provisioning over Competing Clouds**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Abdel R. Sebak

_____ External Examiner
Dr. Hossam S. Hassanein

_____ External to Program
Dr. Anjali Agarwal

_____ Examiner
Dr. Jamal Bentahar

_____ Examiner
Dr. Nizar Bouguila

_____ Co-Supervisor
Dr. Mohamed Adel Serhani

_____ Co-Supervisor
Dr. Rachida Dssouli

Approved by _____
Dr. Chadi Assi, Graduate Program Director

11^{th} of July, 2018

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

End-to-End Trust Fulfillment of Big Data Workflow Provisioning over Competing Clouds

Hadeel El-Kassabi, Ph.D.

Concordia University, 2018

Cloud Computing has emerged as a promising and powerful paradigm for delivering data-intensive, high performance computation, applications and services over the Internet. Cloud Computing has enabled the implementation and success of Big Data, a relatively recent phenomenon consisting of the generation and analysis of abundant data from various sources. Accordingly, to satisfy the growing demands of Big Data storage, processing, and analytics, a large market has emerged for Cloud Service Providers (CSPs), offering a myriad of resources, platforms, and infrastructures. The proliferation of these services often makes it difficult for consumers to select the most suitable and trustworthy provider to fulfill the requirements of building complex workflows and applications in a relatively short time.

In this thesis, we first propose a quality specification model to support dual pre- and post-cloud workflow provisioning, consisting of service provider selection and workflow quality enforcement and adaptation. This model captures key properties of the quality of work at different stages of the Big Data value chain, enabling standardized quality specification, monitoring, and adaptation.

Subsequently, we propose a two-dimensional trust-enabled framework to facilitate end-to-end Quality of Service (QoS) enforcement that: 1) automates CSP selection for Big Data workflow processing, and 2) maintains the required QoS levels of Big Data workflows during runtime through dynamic orchestration using multi-model architecture-driven workflow monitoring, prediction, and adaptation.

The trust-based automatic service provider selection scheme we propose in this thesis is comprehensive and adaptive, as it relies on a dynamic trust model to evaluate the QoS of a cloud provider

prior to taking any selection decisions. It is a multi-dimensional trust model for Big Data workflows over competing clouds that assesses the trustworthiness of cloud providers based on three trust levels: (1) presence of the most up-to-date cloud resource verified capabilities, (2) reputational evidence measured by neighboring users, and (3) a recorded personal history of experiences with the cloud provider.

The trust-based workflow orchestration scheme we propose aims to avoid performance degradation or cloud service interruption. Our workflow orchestration approach is not only based on automatic adaptation and reconfiguration supported by monitoring, but also on predicting cloud resource shortages, thus preventing performance degradation. We formalize the cloud resource orchestration process using a state machine that efficiently captures different dynamic properties of the cloud execution environment. In addition, we use a model checker to validate our monitoring model in terms of reachability, liveness, and safety properties.

We evaluate both our automated service provider selection scheme and cloud workflow orchestration, monitoring and adaptation schemes on a workflow-enabled Big Data application. A set of scenarios were carefully chosen to evaluate the performance of the service provider selection, workflow monitoring, and the adaptation schemes we have implemented. The results demonstrate that our service selection outperforms other selection strategies and ensures trustworthy service provider selection. The results of evaluating automated workflow orchestration further show that our model is self-adapting, self-configuring, reacts efficiently to changes, and adapts accordingly while enforcing QoS of workflows.

Acknowledgments

In the name of Allah, the Most Gracious and the Most Merciful.

Alhamdu li Allah, all praises to Allah for the strengths and His blessing in completing this thesis. First and foremost, I would like to thank Allah Almighty for giving me the power, knowledge, ability, and opportunity to undertake this research study and to persevere and complete it satisfactorily.

In my journey towards this degree, I have found a teacher, a friend, and a pillar of support in my supervisor, Dr. M. Adel Serhani. He has been there providing his heartfelt support, valuable guidance, inspiration, and suggestions in my quest for knowledge. He cared about my work, and responded to my questions and queries so promptly. I have been inspired by his meticulousness, his attention to details, and his energetic application to any problem. Without his guidance, this thesis would not have been possible and I shall eternally be grateful to him for his assistance.

I also pay my gratitude to my co-supervisor, Prof. Rachida Dssouli for her continuous support, motivation, and providing the necessary infrastructure and resources to accomplish my research work. This work would not have been possible without her help, caring, and encouragement. Under her guidance, I successfully overcame many difficulties and learned a lot.

My acknowledgement would be incomplete without thanking the biggest source of my strength, my family. I owe thanks to a very special person, my husband, my best friend, Waleed Khalil for his continued and unfailing love, friendship, encouragement, and understanding during my pursuit of Ph.D. degree that made the completion of this thesis possible. The one who always makes me feel safe and loved. Your support is what made me who I am today. Thank you for bearing up with me and coping up with me. For that and for so much more, I love you beyond words. I appreciate my beautiful children Khaled, Noor, and Omar for their patience they showed during my thesis writing.

Words would never express how grateful I am to all of you. I consider myself the luckiest in the world to have such a lovely and caring family, standing beside me with their love and unconditional support. I would also like to express my thanks to my sister, brothers, cousins, and their families for their support and encouragement.

I am highly indebted to my family-in-law for showing faith in me and supporting my decision to pursue what I desired. I salute you all for the love, care, help, and sacrifice you did to shape my life. I would never be able to pay back the love and affection showered upon by my parents in law.

I would like to express my gratitude to Dr. AbdulMutalib Wahaishi for his great support, encouragement and sincere advice. Words are not enough to thank all my colleagues especially Ikbal Taleb and Alramzana Navas for their continuous help and dedication.

I would like to thank all those people who made this thesis possible and an unforgettable experience for me. To those who have supported and helped me directly or indirectly through the success of this study and made it an unforgettable experience, thank you.

I would like to dedicate this work to my late parents Mrs. Rafif El-Saadi and Dr. Talaat El-Kassabi whose dreams for me have resulted in this achievement and without their loving upbringing and nurturing; I would not have been where I am today and what I am today. This one is for you Mom and Dad!

Contents

List of Figures

List of Tables

List of Acronyms

API	Application Programming Interface. 4, 140, 160
ARIMA	Autoregressive Integrated Moving Average. xiv, 174, 175, 178–181, 185, 187, 188, 191–193
ARMA	Autoregressive Moving Average. 178–181, 187
BCI	Bootstrap Confidence Interval. 127
BMRM	Benchmark Reputation Model. 124
BPaaS	Business Process-as-a-Service. 63
BPEL	Business Process Execution Language. 4, 59, 64
BPMN	Business Process Modeling Notation. 4, 59
CCRM	Category-based, Context-aware and Recommendation incentive-based reputation Mechanism. 48
CF	Collaborative Filtering. 56, 57
CSMIC	Cloud Services Measurement Initiative Consortium. 37
CSP	Cloud Service Provider. iii, vii, xi, xii, 3, 5, 7–9, 36, 43, 48, 50–53, 55, 56, 69, 73, 74, 84, 92, 96–106, 110, 112–116, 118, 119, 121–124, 126, 132, 195, 198
CTL	Computation Tree Logic. 158

CWTS Comprehensive Weighted Trust Score. xiii, 119, 129–136

DBA Deterministic Buchi Automaton. 158

DSL Domain-Specific Language. 64

EML Ecological Metadata Language. 78

FGDC BDP Federal Geographic Data Committee Biological Data Profile. 78

GRU Gated Recurrent Unit. 178

IaaS Infrastructure-as-a-Service. 13, 17, 29, 43, 46, 49, 53, 56, 72

IAH Iterative Adjustment Heuristic. 61

IoT Internet-of-Things. 1, 199

IOWA Induced Order Weighted Averaging. 48

JSON Java Script Object Notation. 78

KPI Key Performance Indicator. 37, 62

MADM Multiple Attribute Decision-Making. ix, 52, 93, 100, 105–107, 110, 121, 126, 136, 145, 148, 181, 195, 199

MDP Markovian Decision Processes. 183

MIMICIII Multi-parameter Intelligent Monitoring in Intensive Care III. 138, 162, 185

ML Machine Learning. 67

MLR Multiple Linear Regression. ix, 94, 106, 112–114, 126, 129, 136

MSE Mean Square Error. xiii, 186

NIST National Institute of Standards and Technology. 27

OIM Open Information Model. 78

OLS Ordinary Least Squares. 113

OWL Web Ontology Language. 54

PaaS Platform-as-a-Service. 13, 29, 43, 53, 72

PSO Particle Swarm Optimization. 54

QoCS Quality of Cloud Service. xii, xiii, 5, 8–10, 13–15, 17–19, 37, 40, 43–49, 53, 73, 82, 91, 93, 96–98, 100–105, 108, 111, 112, 114–120, 129, 134–136, 195

QoS Quality of Service. iii, iv, viii, ix, xi, xii, xv, 2–4, 6, 7, 10–14, 16–24, 35, 36, 40–44, 50–54, 56–63, 65–71, 73, 74, 83, 84, 87, 88, 90, 97, 100, 103, 104, 106, 108, 110, 113, 114, 121, 126, 135, 137–150, 152, 154–156, 158, 159, 162, 167, 171–174, 176, 178–181, 183–185, 188, 189, 191–194, 196, 198, 200

R^2	Absolute Fraction of Variance. 186
RDF	Resource Description Framework. 78
RMSE	Root Mean Square Error. xiii, 186
RNN	Recurrent Neural Networks. 178
SaaS	Software-as-a-Service. 13, 17, 29, 43, 46, 52–54, 56, 72
SAW	Simple Additive Weighting. 106, 107, 109, 110, 121, 123, 124, 129, 136
SBA	Service-Based Applications. 64
SLA	Service Level Agreement. 3, 27, 35, 46, 47, 51, 53, 60, 63, 65, 67, 145
SMART	Simple Multi-Attribute Rating Technique. 106, 107
SMI	Service Measurement Index. 37
SOA	Service Oriented Architecture. 32
SWA	Simple Weighted Average. 105
TIFNs	Triangular Intuitionistic Fuzzy Numbers. 52
TMC	Trust Management System. 47
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution. 106, 109, 121, 123, 124, 136
UML	Unified Modeling Language. 78
VM	Virtual Machine. 2, 49, 61–63, 67, 143, 173, 176–178, 183, 185, 199

WPM Weighted Product Method. 106, 121, 123, 124, 136

WSDL Web Services Description Language. 55, 57, 64

WSML Web Service Modeling Language. 54

XML Extensible Markup Language. 55, 57, 78

Chapter 1

Introduction

The evolution of information technology and the major paradigm shift of computation from the age of colossal machines to the omnipresent digital era have made information technology an essential aspect of daily human activities. Nanotechnology, quantum computing, cloud-based computing, mobile computing, and the new area of computation known as the Internet-of-Things (IoT) have generated massive volumes of structured and unstructured data. To glean valuable insights, Big Data require processing, storage, analysis, and visualization. Big Data is not only defined by its size, but is also known by the multi V-based characteristics of Volume, Variety, Velocity, Veracity, Validity, Volatility, and Value [1], [2], [3], [4]. These special characteristics of Big Data introduce several challenges, such as data collection and integration problems, due to the data being distributed across diverse geographical locations. Moreover, the management, processing, and storage of Big Data present significant challenges considering the enormous volume and heterogeneous nature of the datasets, in which traditional processing platforms are unable to handle such massively heterogeneous data volumes efficiently.

The special characteristics of Big Data necessitate new computational paradigms and capabilities to effectively process prodigious datasets in real time including modeling, visualization, prediction, and optimization. Today, Cloud Computing is considered a promising paradigm as it offers a suitable infrastructure for large-scale and complex computations. The advantages of Cloud Computing include parallel processing, data service integration, and scalable data storage, which make

Cloud Computing a perfect platform for Big Data processing and storage. Moreover, it satisfies organizations' economic goals by providing on-demand computing resources saving upfront costs especially for small and medium sized businesses. In addition, it fulfills the scientific and engineering community requirements of building sophisticated, complex applications with relatively minimal effort and time. Hence, more organizations and enterprises are increasingly adopting cloud-based solutions, such as storage, database and search services to develop and build business and research applications.

The correlation between Cloud Computing and Big Data is identified in [5] as provisioning good performance with respect to computer power, storage, and network communications. In addition, Amazon, eBay, Google, Microsoft, and other technology leading and e-commerce companies provide scalable Cloud Computing infrastructures and platforms that are suitable for Big Data processing, such as MapReduce, the Google File System, BigTable, and Dynamo [6].

Not only do complex Big Data applications require the aforementioned tools and resources, but also need to be decomposed workflows which comprise series of smaller components, such as tasks and services for better scalability and performance. Cloud workflow has been proven to be an appropriate model for many application domains, which features a set of tasks aggregated and executed either in sequence or in parallel to fulfill a particular goal. Workflows executed on a composed cloud services are distinguished by their ability to scale up or down according to the fluctuating nature of job or task requirements. This is achieved through orchestration functionalities which can result in adding more storage space, auxiliary memory, additional servers, or reinstating corresponding Virtual Machines (VMs) in accordance to the sequence events that might take place, such as usage increase, or task failures. These orchestration functionalities allow real-time automated reconfigurations of the appropriate resources.

Maintaining an end-to-end QoS of such complex cloud workflows is very important to end users and applications. However, achieving this requirement necessitates the following: The careful selection of the cloud provider capable of satisfying the requisite level of QoS, and guaranteeing the QoS during workflow execution, which cannot be archived throughout orchestration alone, but also through automated monitoring and control of multi-cloud services and resources. In the following, we identify and discuss some of the relevant research problems that need to be considered in

guaranteeing end-to-end QoS of cloud workflows.

Research problem # 1: Selection of a CSP

Cloud service providers selection is an important factor in supporting end-to-end QoS enforcement in Big Data lifecycle. Recently, a large number of CSPs emerged in the market offering similar functional properties to satisfy growing demand. Consequently, selecting the most suitable cloud provider becomes a challenging task. The selection of a CSP depends on many factors, and the following are some of the recommended criteria that are adopted by the community of researchers in this field: 1) certifications and standards, 2) technologies and service roadmap, 3) data security, data governance, and business policies, 4) service dependencies and partnerships, 5) contracts, commercials, and Service Level Agreements (SLAs), 6) reliability and performance, 7) migration support, vendor lock-in and exit planning, and 8) business and company profile. Despite the fact that the criteria above are commonly used for CSP selection, in this work, we consider selection criteria that can be measured, monitored, and enforced. Other selection criteria can be specified and included in a formal contract known as an SLA. In this research, we scope our use of 'suitable' cloud provider as being trustable in terms of satisfying properties required by the user, which include a certain number of functionalities with some level of quality of service, as well as reputation.

Providing quantitative approaches for evaluating CSP performance is essential to reassure users when moving Big Data applications to the cloud and further utilizing and exploiting its boundless capabilities and potential. According to [7], organizations are reluctant to use the cloud because of many reasons, such as performance. They revealed that 43.5% of enterprise IT managers fear losing profit because of bad cloud performance and about 80% fear hidden costs caused by losing their reputation due to downtime and poor performance. Another survey collected from 3000 cloud users shows that approximately 84% of cloud users do not fully trust cloud service providers primarily due to data control issues [8]. Therefore, trust, especially trust-based cloud service selection, has recently attracted the attention of academic researchers.

Research problem # 2: Workflow QoS Guarantee

Guaranteeing workflow runtime QoS is another well recognized research problem. According to [9], few research initiatives were proposed in the area of designing automated execution and monitoring complex workflow systems. Enabling easy-to-use systems that allow specification of QoS

requirements' levels and flexible deployments and resource allocation is highly required. This includes building models that describe algorithms and structures to empower these systems. Using state machine-based models to formulate the resource orchestration and auto-reconfiguration is recognized for its capability to represent the continuous and dynamic nature of cloud resources. Maintaining the timely state of each entity, such as resources, quality requirements, and tasks performance, allow for easy tracking, efficient monitoring, and automated reconfiguration of the cloud resources and workflow deployment. Existing resource orchestration systems focus on resource configuration, deployment or control. However, they do not provide full automation to support self-configuration and self-learning where failures and performance deficiencies are detected and resolved automatically to maintain the required QoS [9].

Providing runtime intelligence in a sophisticated orchestration system involves high processing capabilities and adds more overhead on the cloud resources to provide analysis of large amounts of real-time monitoring data. Also, some workflows are deployed on multiple clusters and cloud providers, which makes it even harder to support runtime intelligence across different cloud environments.

Federated cloud resource orchestration involves connecting multiple interacting cloud services to perform a composed service. Existing orchestration techniques depend on procedural programming that employ low-level scripting languages and heterogeneous Application Programming Interfaces (APIs), which are highly provider-configuration dependent [10]. This imposes more time and effort burden on the consumer. Hence, various research initiatives have proposed common interfaces and APIs over multiple clouds, such as Apache Deltacloud [11], Apache Libcloud [12], jclouds [13], and OpenStack [14]. However, dynamic orchestration using high-level policies specified by administrators rather than consumers is highly compulsory. The currently used service composition techniques such as the Web Service Business Process Execution Language (BPEL) and Business Process Modeling Notation (BPMN) do not support application resource requirements and constraints, and optimized resource scheduling which are essential for a comprehensive orchestration process [10]. Hence, trust enforcement is highly recommended to support the intelligent orchestration framework that handles the quality requirement of Big Data.

Research problem # 3: Trust Management in the Cloud

Trust is one of the important issues in Cloud Computing-based environments because the adoption of such technology will allow better application experiences with enhanced system resource consumption. Additionally, tremendous effort has been devoted towards addressing Big Data challenges and issues that are related to social, ethical, and technical perspectives. However, very few research initiatives addressed the issue of establishing trust for Big Data processing and storage over a single or federated clouds, which is considered a crucial challenge due to its special characteristics mentioned above. In the following, we describe how we tackle trust in a cloud environment to fulfill Big Data requirements through two phases: prior to cloud provider selection and during cloud service and resources provisioning.

First phase: Trust Assessment to Support CSP selection

Selecting the *best* cloud provider among a vast competing pool of options to store and process Big Data is a challenging process. It is difficult for service consumers to decide which cloud provider to deal with as they may lack knowledge about whether the available cloud resource capabilities can handle Big Data tasks while satisfying a set of QoCS requirements. In addition, published QoCSs might be inflated for marketing purposes, so they cannot always be trusted. Furthermore, current trust models lack the flexibility to accommodate fluctuating and ever-changing user QoCS requirements. These models tend to ignore the dynamic nature of trust, particularly in cloud environments. A QoCS is dynamically altered due to several factors such as changing demand levels (the number of service requests changes continuously over time) and the cloud provider's resource limitations. Thus, a trust model should adapt to the dynamic nature of cloud service usage. Therefore, a Big Data user (client or application) should typically perform a trust evaluation with measurements of a cloud provider before any decision on transferring critical data to the provider's cloud for processing capabilities or storage purposes.

Second phase: Trust Assessment to Support Workflow Execution

While cloud resource requirements need to be enforced within a dynamic orchestration, a trust evaluation must also be sustained. A trust model should consider all the workflow phases and evaluate trust for each composed service, and then aggregate the overall workflow trust evaluation across

multiple cloud providers. The model must carefully deal with all trust components, such as trust propagation, and trust aggregation in federated cloud services. The trust score evaluation consists of capturing and monitoring the workflow runtime environment data to provide and maintain required orchestration of QoS levels. Yet, the complexity of orchestrating cloud services for Big Data is emphasized by the growing number of cloud services in terms of quantity, quality, and diversity. Few research initiatives fulfill user requirements in a real-time and context-aware manner, especially with the overwhelming amount of data coming from various sources of high veracity and variety.

Therefore, trust evaluation schemes and models should cope with the nature of intelligent workflow orchestration and composition of cloud services, especially when dealing with scalable and adaptive composition solutions that handle large-scale, highly dynamic, and diverse Big Data services. Supporting trust enforcement on orchestration frameworks creates an additional challenge to assess the contribution of the component services towards the composite services. This is because each service component might have different functionalities, significance and impact within different compositions. Additionally, any proposed model must consider lightweight monitoring mechanisms with minimal overhead without affecting the overall service performance.

1.1 Motivation

Nowadays, data analytics is considered a cornerstone for decision-making as well as strategic planning. Hence, applications, such as real-time fraud detection, prevention of disease outbreaks, management of natural disasters or intelligent vehicle management, require processing of Big Data generated from an unlimited number of information sources for all decision-making processes.

Cloud Computing has emerged as a powerful paradigm for provisioning Big Data application storage, processing, and services supported by a variety of scalable virtual resources and services. These applications can be modeled and characterized as complex cloud workflows and automatically orchestrated to respond to the scalability and dynamicity requirements of such applications.

The cloud workflows exhibit special characteristics that require a high-level of quality and are time sensitive. Guaranteeing and maintaining the crucial quality of service levels for complex cloud

workflows requires the monitoring of quality perspectives across different phases of workflow deployment, execution, and adaptation. The first phase is the selection of the cloud service provider that is trustworthy and guarantees the required level of quality in its cloud workflow. Once the cloud workflow is deployed, another phase of maintaining the quality of service is required using monitoring and adaptation when needed. End-to-end quality guarantees impose the need for a workflow management process and procedure that defines a lifecycle model applicable for Big Data applications and cloud-based environments. This model comprises the following four phases:

(1) Cloud Service Provider Selection: choose a favored CSP for workflow execution.

(2) Resource configurations: define the composing tasks of a workflow and specify the workflow structure.

(3) Deployment: describe the execution environment under which the workflow tasks are to be executed.

(4) Monitoring and control: monitor the workflow during execution to guarantees the required QoS levels.

One of the critical issues a user might face is the selection of an appropriate and trustworthy cloud provider to process Big Data workflows while guaranteeing a convinced QoS level. Once the selection decision is reached and after the workflow is deployed, guaranteeing the performance quality of the workflow becomes another challenging issue. Furthermore, as the cloud workflow is a complex composition of multiple tasks, it is difficult to self-adapt, self-configure, and scale to react to runtime environment changes and maintain the required performance level. Applying the concept of trust and trust evaluation reinforces the end-to-end QoS guarantee throughout the phases above.

In this work, we first propose a workflow quality specification model that provides a multi-dimensional Big Data quality assessment specification while combining both data-driven and process-driven quality assessments. This quality model enables the quality evaluation performed in our proposed trust model for both CSP selection and workflow orchestration and adaptation. We next propose a generic trust enforcement model for cloud service provider selection and workflow orchestration. This model supports adaptation through monitoring and prediction, which engender a

vision of how trusted CSP are selected according to the desired end-to-end QoCS levels.

1.1.1 Workflow Quality Specification Model

Big Data-enabled workflows have gained momentum because of the major paradigm shift in computation from massive machines to the ever-present digital era generating vast volumes of data requiring high-levels of service quality. This issue motivates the adoption of an end-to-end quality and trust specification framework. This framework should support large-scale heterogeneous cloud environments. However, there are limited initiatives in the industry and the literature to provide a framework that captures related quality specifications at different granularity levels, including cloud resource configuration, usage, customization, and all aspects of workflow management. Hence, this facilitates examining, monitoring, and managing of the quality of complex and heterogeneous cloud resources, workflows, and cloud providers using a unique end-to-end quality specification framework.

1.1.2 First Phase, Trust-based CSP Selection for Workflows

As a result of the aforementioned limitations, automating the decision-making process of cloud provider selection with a special focus on Big Data processing requirements and user QoCS preferences is highly desirable. Likewise, a comprehensive trust model is required that does not rely on the potentially falsely advertised QoCSs of cloud providers nor on historical records that cannot deliver accurate trust scores due to dynamic changes in cloud resources. Accordingly, in this work, we propose a multi-dimensional trust model that evaluates the services of cloud providers based on 1) the client's QoCS requirements, 2) the provider's current resource availability, 3) the historical records of previous communications with the cloud service providers, and 4) the neighbors trust score evaluation based on their historical records of previous communication with the cloud service providers.

The fulfillment of user preferences is a highly valued component of our proposed trust model. The majority of current trust models do not consider users' QoCS preferences and how much (i.e., via weights) each quality attribute should contribute to the trust score evaluation.

A QoCS is dynamically altered due to several factors, such as changing demand levels (the

number of service requests changes continuously over time) and the cloud provider's resource limitations. For example, a cloud server might be fully loaded at a specific time of day and lightly loaded at another time, which might be due to a periodic rush hour (e.g., end of month transactions) or an unpredicted increase in the number of service requests. Furthermore, existing CSP selection solutions collect QoCS preferences from users through a sophisticated process that requires technical competence of the users, which makes the CSP selection a difficult task to accomplish [15]. A review of some of cloud service selection systems performed in [16] revealed a deficiency in advanced measurements of user preferences techniques.

Because the marketed QoCS information is often unreliable, trust can be more accurately evaluated using previously recorded QoCSs. However, the number of service providers has increased worldwide concomitant with an increase in the number of transactions between users and service providers. Thus, the process of gathering information about these numerous transactions for trust evaluation has become a sophisticated and keenly important topic attracting the attention of academic researchers.

Other existing proposed solutions have limitations related to the consideration of only partial context QoCS attributes. They lack consideration of a quantitative evaluation model for historical QoCS records. Therefore, the development of efficient and accurate trust models for Big Data service evaluation remains an increasingly challenging and open area of research [17]. Previous research initiatives focused on trust models that were primarily based on reputation, which is not dynamic and lacks real-time representation. Moreover, reputation can be a misleading property in the case of untrustworthy users (either service providers or consumers). Although several users might have different subjective opinions about a specific service, it is well observed that many of those users tend to have malicious and biased intentions. Feedback-based models consider to some extent the opinions of users and are assumed to be rational and meaningful measures of service reputation [18], [19]. In such models, consumers usually view the services in terms of the preferences that are important from their point-of-view, such as QoCS and cost. Also, they have a higher chance of pinpointing the strengths and weaknesses of the service from a neutral and unprejudiced view. However, a problem with feedback-based models is the lack of a qualitative measurement mechanism that can assign accurate initial trust values. A bootstrapping mechanism was used in a

few reputation systems to address this issue [20], [21].

The sincerity level granted to users' opinions might be biased in some cases. Users employ different ranking methods when writing their reviews; where some users choose numbers to rank services while others choose descriptive words like 'excellent', 'good' or 'bad'. Service providers can offer discounts on their service fees to encourage users to provide reviews, which might drive an unfair advantage to service providers who provide incentives over those who do not. Furthermore, service providers frequently afford an incomplete description of their service to obscure QoS shortcomings and flaws for commercial reasons [22].

Local trust and recommendation trust have been combined in systems that use weights and values for each type of trust. The weights are chosen in a way that it is not necessarily dynamic and might not fulfill the user QoCS requirements, which are typically subjective. Service trust based on previous work ignores information about the dynamic resources of the cloud service provider.

Diverse single-dimensional trust models are recommended in the literature, however, due ue to its limited perspective, important criteria can be missed in such homogeneous models. Other proposals ignore the trustworthiness of reviewers, which is vital to a reputation model and can result in inaccurate trust evaluation. Furthermore, several approaches give credibility to the majority of ratings, which is not always warranted. Consequently, the development of a relatively comprehensive trust model remains an open challenge.

Accordingly, we propose a formal multi-dimensional trust model for selecting a cloud provider that is applicable for Big Data distribution and processing. Our comprehensive trust model assigns a trust score value for each cloud provider according to the three weighted factors of current cloud advertised properties, evaluated reputation, and supported historical communications. Correspondingly, a user will be able to choose the cloud provider who accumulates the highest trust score.

It is noteworthy, that continuous monitoring and adaptation are two essential activities that need to be taken into consideration during the execution of Big Data workflows to achieve and guarantee the expected QoCS claimed by the selected cloud service provider.

1.1.3 Second Phase, Trust Enforcement through Workflow Orchestration, Reconfiguration, Adaptation, and Self-learning

Workflow management and execution frameworks are typically comprised of various components (e.g., Monitor, Resource Scheduler, and Adapter), which rely on cloud resources exhibiting special characteristics, such as scalability, availability, and flexibility. The management of such components involves supporting different functionalities including configuration, deployment, quality of service, and communication. Hence, cloud workflow management is considered a complicated process that consumes both time and resources, particularly if it is manually configured using predefined rules, which usually lack standardization, interoperability, and reusability.

Workflow frameworks should support separating functionalities into specialized layers such as workflow planning, deployment, and monitoring layers. For instance, the planning layer is concerned with handling data flow and errors. Whereas, handling deployment matters should be performed at a higher level of abstraction to decrease technical details for the operators [23].

Current orchestration frameworks usually support some level of QoS, but they do not guarantee the QoS from various user perspectives. Yet, the orchestration schemes are used to optimize the selection of the required cloud resources to satisfy the user's QoS needs. A comprehensive workflow management system should support capturing user requirements and quality enforcement issues, such as performance constraints and privacy rules so that workflow plans can be automated. Nevertheless, keeping the required level of QoS is even more important for the functionality of a workflow management framework, which is supported through workflow monitoring and event capturing and analysis. Collecting monitoring data logs of workflow execution environment parameters and analyzing them by taking some intelligent actions to prevent errors or QoS violations will cause performance degradation, but will also help predict workflow resource utilization and the reaction to resource shortages before it causes service performance degradation.

An ultimate workflow management solution should monitor, predict, and adapt workflows, in addition, to evaluating trust in a highly dynamic workflow environment. The continuous monitoring of resource utilization combined with workflow resource prediction will help in detecting QoS degradation and violations, and will eventually apply different adaptation strategies. Such strategies

are developed to capture and classify violations and accordingly respond with the appropriate actions to accommodate cloud resources as needed. Performing actions, such as adding resources, are not only intended to prevent performance degradation, but also to stabilize the required QoS levels. Workflow's trust evaluation that can be conducted at different granularity levels (e.g., composed tasks trust and resources performance trust) will support these adaptation schemes.

1.2 Problem Statement and Research Questions

The challenges that face applications dealing with Big Data in clouds such as data collection, storage, search, analysis, and sharing were discussed in many studies in the literature. The selection of a suitable and trustworthy cloud provider for Big Data workflows execution enforcing a certain QoS level of acceptance is the foremost concern to address. However, very few research initiatives focused on this issue from a comprehensive point of view which include the cloud provider advertised, reputation and self-experience dimensions while considering the user preference.

The limitations of existing cloud provider selection trust models include a non-dynamic nature and lack of real-time adaptability, which make them unsuitable for Big Data and cloud environments. Depending solely on reputation can be misleading if the users are untrustworthy or subjective, especially given that different users have diverse opinions about the provisioned services. Other researchers have combined local trust with recommendation trust using weights. However, the various methods used to determine the weights are not necessarily dynamic and amenable to the user's point-of-view. The majority of research initiatives on the service trust ignore information about the dynamic resource status of the cloud providing the service. Moreover, the existing trust models do not base their trust score evaluation on the QoS attributes related to Big Data special characteristics, and they produce unsatisfactory results with respect to Big Data application requirements. Considering these challenges, we propose a framework and formalize our proposed trust model in Chapter 5.

Following cloud provider selection and deployment of workflows, it is essential to sustain the required QoS levels through workflow orchestration. The orchestration of workflows is still in its infancy, and the more complicated a workflow, the harder the orchestration process becomes. Hence,

existing orchestration frameworks do not guarantee the prerequisite levels of QoS and emphasize monitoring few QoS attributes and handle limited corrective actions. Therefore, extra effort is required to guarantee workflow QoS and avoid service degradation or interruption. Trust enforced monitoring of workflow status at different decomposed services, cloud resources, and at different granularity levels is crucial to reach the targeted levels of QoS. Moreover, automatic system adaptation and reconfiguration based on monitoring as well as prediction evades degradation in QoS.

As described above, the intention here is to facilitate an end-to-end QoS through the support of trust model frameworks by first automating the cloud service provider selection for Big Data processing or storage, and second by maintaining the required QoS levels of cloud workflows during runtime through orchestration based on monitoring, prediction, and adaptation.

The following are the key research challenges and questions we address in this thesis, which are divided into three areas:

(1) ***Cloud workflow QoS-based trust: attributes, requirements, and evaluation.***

Q1.1. How can the trust-based quality of service be specified? What are the QoS attributes that need to be included in a trust-based QoCS model? Who is considered the trustee and who is the trustor and on what basis it is perceived? Trust can be subjective or objective in terms of how the trustor views the trustee. No standard defines trust to help users understand, measure, and compare cloud service providers with respect to the quality of their services both subjectively and objectively. Existing trust models neither satisfy nor capture all requirements linked with the multi-Vs Big Data Characteristics and do not consider the user QoCS preferences. Also, linking the quality attributes to the cloud service model is another challenging issue as different QoCS attributes should be provided if the target is Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) or Infrastructure-as-a-Service (IaaS), such as performance, usability, privacy, and price.

Q1.2. What mechanism can be developed to capture users' QoS requirements without prior knowledge of the selection criteria of QoCS? Collecting QoCS attributes from the user is not straightforward because they should match the user application requirements and allow the user to input their preferences. To accomplish this task, the user is expected to have a certain

level of knowledge. Existing trust models require a detailed QoCS definition as input, which usually involves technical knowledge the user might not have. Moreover, existing commercial and non-commercial service selection systems lack support for users with respect to defining and validating the QoCS requirements and lack the automation capability for collecting Big Data and cloud quality attributes.

Q1.3. What quality attributes do we monitor in service composition and orchestration frameworks? What complements the existing monitoring systems? Capturing and monitoring the workflow runtime environment data is of prime importance to support QoS during orchestration. Nevertheless, challenges of capturing, monitoring, and analyzing runtime data are associated with supporting intelligence in resource orchestration, which incorporates runtime resource description, requirements, and constraints.

(2) ***Trust in cloud service provider selection.***

Q2.1. What mechanisms and strategies must be developed to ensure a rational selection of clouds based on QoCS and Big Data requirements? The two strategies adopted in the literature to build trust are based on direct interactions and indirect methodology, which have advantages and limitations. With the direct interaction strategy, trust evaluation relies on evidenced recordings of previous interactions with the other entity. However, trust can only be evaluated after using the service and not before. In contrast, reputation-based trust evaluation collects the trust values from other parties who had previously interacted with the entity being evaluated. Although the indirect-based trust evaluation overcomes the missing initial trust value problem in the direct-based trust evaluation, it encounters two main issues: the false or fake reputation values, and the subjective judgment of the other entities which may have different preferences or requirements. Handling malicious and fake ratings in reputation-based systems is still an open issue in the literature. Though different methods were proposed to handle false ratings, there are still open research challenges related to the different degrees of complexity of these methods. The subjectivity nature of trust evaluation of reputation-based systems is another open issue in the literature. Research proposals combine both direct and indirect strategies. However, none of them incorporate user preference in the indirect

trust evaluation.

Q2.2. Do the distributed nature, heterogeneity, and autonomy of cloud environments affect the trust evaluation? Trust model evaluation for clouds must cope with the distributed nature and heterogeneity of this dynamic environment. This imposes many requirements that need to be met by the trust model, such as identity management of different applications and users, continuous trust assessment and enhancement, and handling opinions and experiences of different users.

Q2.3. Can a trust model satisfy different properties of trust? Most of the proposed trust models satisfy some of the following features: 1) the use of evidenced interaction experience, 2) consideration of user preference for selecting the different quality attributes for trust evaluation, and 3) control of the impact of fake ratings when using reputation-based strategy for trust evaluation. To the best of our knowledge, none of the existing cloud selection models satisfy a holistic set of requirements like being lightweight in terms of model complexity, being adaptive to the dynamic nature of existing clouds, consideration of the user preferences in choosing the QoCS attributes, and ranking their importance along with the support of detection of untrusted entities.

Q2.4. How can we prove that our trust model is lightweight and does not induce burden on the user, community members, and cloud providers? Formal evaluation is required to measure the complexity of trust evaluation algorithms, to ensure low complexity. In addition, appropriate communication overhead evaluation involved in managing, handling, measuring, and evaluating trust is also required to evaluate the performance of a trust model.

(3) ***Trust in cloud service composition and orchestration.***

Q3.1. How can trust be evaluated in cloud service composition and orchestration? Do the existing trust models apprehend all granularity levels of cloud workflow orchestration, such as the performance of composed services and allocated cloud resources? Different strategies can be used to extend a trust model to cope with cloud services composed of more than one cloud provider. A trust model should work across the workflow phases and evaluate

trust for each composed service, and then aggregate the overall workflow trust across multiple cloud providers. This opens up issues related to services composition and others related to evaluating trust of different composite cloud service types, such as sequence, parallel or loops. Trust evaluation techniques should cope with scalable and adaptive composition solutions associated with large-scale, dynamic, and diverse Big Data services. Other challenges include trust propagation, trust aggregation, decomposition, and trust sharing in composite cloud services.

Q3.2. Are existing prediction models suitable for predicting workflow performance and effective in guaranteeing the QoS? Is prediction and QoS value calculation performed dynamically at runtime? Existing prediction approaches do not combine monitoring QoS data with the prediction model, which limits the effectiveness of the model in guaranteeing the QoS. Thus, the adaptation strategies and actions should be issued based on real-time monitored performance information as well as prediction of the cloud resources behavior. Selection of the appropriate QoS attributes to be used for monitoring and prediction is essential for an effective QoS guarantee. These QoS attributes are directly related to the context of the application and the cloud resources.

Q3.3. How should workflow monitoring and event capturing be analyzed? The autonomic orchestration is usually supported through the collection and monitoring of the environmental parameters and the data analysis to perform some intelligent actions. It also requires the development of concepts and techniques to model, capture, and abstract the states of each component of the workflow and the deployed resources. For example, characterizing the states of an application or a specific application component into meaningful concepts improves cloud elastic resource orchestration purposes.

Q3.4. Do existing orchestration frameworks support self-reconfigurable and self-learning workflows at runtime? Most orchestration schemes are either configuration or deployment dependent. Fully automated workflow orchestration envisions an autonomic orchestration that is self-adapting, self-configuring, and self-learning. In response to any cloud services performance degradation, a workflow should dynamically implement high-level reasoning of

the runtime environment properties and autonomic orchestration tasks. Such adaptation tasks include, for example, quality degradation detection and restoration and resource shortage prediction and provisioning. Additionally, an orchestration framework should learn from past executions to build experience to use for self-protection in future executions from similar situations to avoid quality degradation.

1.3 Summary of Contributions

The objective of this dissertation is to develop a framework for an end-to-end trust-based framework for orchestrating Big Data workflows among competitive clouds while emphasizing the workflow monitoring, prediction, and adaptation activities. Such a framework allows for the automation of the decision making the process of selecting the most suitable cloud provider for Big Data processing that fulfills user's requirements and preferences while guaranteeing the required QoS levels during runtime and enabling automated workflow reconfiguration to avoid quality degradation. The model defines a formal trust model, which enables the enforcement of end-to-end QoS in Big Data workflows.

The contributions of this research are outlined in the following.

1.3.1 Big Data Workflow Quality Specification Model: Attributes, Requirements, and Evaluation

This contribution is detailed in Chapter 4.

(1) For trust quality attributes specifications, we:

- Propose a mapping scheme between Big Data properties and cloud quality metrics that result in a generated set of quality attributes to be used to evaluate the degree of trustworthiness of the cloud providers, and
- Consider both the QoCS of SaaS and IaaS, which are evaluated using historical records logged by the customer and neighbors, and by measuring cloud resources, i.e., memory and processing power, respectively.

This answers the research question Q1.1 in Section 1.3.

(2) For user quality requirement collection, we automate the process of collecting Big Data and cloud quality requirements from the user through a guided web-based application enabling the user to formulate the desired quality requirements without requiring expertise. This answers the research question Q1.2 in Section 1.3.

(3) For supporting users with limited technical cloud knowledge, we propose a comprehensive framework to consolidate quality specifications at different granularity levels. This answers the research question Q1.2 in Section 1.3.

(4) For guaranteeing user QoS in the cloud and Big Data workflow orchestration frameworks, we propose a profile-based description scheme to capture and monitor runtime environment quality attributes for the orchestration framework. Different application types involve different sets of metrics to maximize the user-defined QoS requirements towards effective monitoring. Therefore, we complement the existing monitoring systems by defining a runtime resource description, requirements, and constraints including specific properties and metrics (e.g., performance, throughput, response time, and utilization) that are appropriate for each task specification and profile. This answers the research question Q1.3 in Section 1.3.

1.3.2 Cloud Service Provider Selection for Big Data Workflows Based on Trust Evaluations

This contribution is detailed in Chapter 5.

(1) To ensure rational selection of clouds based on QoCS and Big Data requirements, we propose a multi-dimensional trust model that implements three strategies relying on the provider's advertised QoCS, neighboring assessments, and on the user's past personal experience with the cloud provider. The neighbors' assessments are also based on the user preference regarding the significance of each quality attribute. Our model automates the decision-making process of cloud provider selection with an eye towards Big Data processing requirements and user QoCS preferences. The formal trust model satisfies a holistic set of requirements as follows:

- Provides a trust score evaluation combining the user's experience, the reputation of the cloud service provider, and the cloud's resources. The model is designed to accommodate dynamic and continuously changing cloud environment resources with different load levels and time slots.
- Considers the user preferences in choosing the QoCS attributes and ranks their importance.
- Supports detection of untrusted entities by using community-based reputation management wherein a community management system enforces a set of engagement and participation rules.
- Copes with the distributed nature of cloud environments including independent users and applications.

This answers the research questions Q2.1, Q2.2 and Q2.3 in Section 1.3.

(2) To evaluate our multi-dimensional trust model we conduct a set of experiments that combine our developed simulation package and CloudSim simulator package [24]. We also evaluate the complexity of the trust evaluation algorithms from the perspectives of provider resource capabilities, self-evidenced provider service quality and reputation information collected from neighbors. We also evaluate the communication overhead involved in managing, handling, measuring, and evaluating trust. This answers the research question Q2.4 in Section 1.3.

1.3.3 Trust Enforcement on Cloud Workflow Service Orchestration

This contribution is detailed in Chapter 6 and Chapter 7.

(1) We propose a workflow orchestration, monitoring, prediction and adaptation model that relies on trust evaluation to detect QoS performance degradation and perform an automatic reconfiguration to guarantee QoS of the workflow.

- The trust model propagates across the workflow phases and evaluates trust for each composed service, then aggregates the overall workflow trust across multiple cloud

providers. Simultaneously, the QoS-based trust is assessed for allocated cloud resources. This answers the research question Q3.1 in Section 1.3.

- We propose monitoring, prediction, and adaptation schemes that detect and repair different types of real-time errors and trigger different adaptation actions, including workflow re-configuration, migration, and resource scaling.
- We formalize the cloud resource orchestration using a state machine that efficiently captures different dynamic properties of the cloud execution environment. Also, we use a validation model checker to validate our model in terms of reachability, liveness, and safety properties.

(2) We propose an improved orchestration framework to support self-reconfiguration, self-learning, and self-adaptation dynamically at runtime. We support QoS trust monitoring and automatic reconfiguration through the collection, analysis, and prediction of performance information to detect quality degradation or execution violation and automatically repair the problem by reconfiguring the workflow accordingly with proper real-time actions.

- Our orchestration framework includes a formal model to allow real adaptation for complex composition situations where composed tasks may undertake several dependency issues. Our workflow responds dynamically to any cloud service performance degradation by implementing corrective actions automatically during runtime.
- Our orchestration framework learns from the past execution behavior and uses it to avoid expected quality degradation.

This answers the research question Q3.4 in Section 1.3.

(3) We propose a monitoring system that supports monitoring at different granularity levels (e.g., task, application, and system resources) to satisfy an overall workflow, composed services, and cloud resources performance evaluation to guarantee a comprehensive perception of accepted quality of service levels. This answers the research question Q3.3 in Section 1.3.

(4) We propose a prediction model to anticipate workflow performance degradation or resources shortage, or execution interruption.

- Prediction of QoS values is performed dynamically at runtime combined with the monitoring of QoS data to calculate trust scores of the workflow.
- Each workflow has specified QoS attributes to be used for prediction, which is well defined in a profile knowledge base and are chosen based on the application contextual and field related properties.

(5) We implement three adaptation strategies to capture changes in environment resources, categorize various violations, and take the necessary actions to adapt resources according to workflow needs.

- The first strategy is to respond to agile resource performance degradation based on predicting eventual QoS degradation.
- The second strategy is to respond to severe and unexpected resource performance degradation or errors in real time based on monitoring information through continuous analysis.
- The third strategy is a hybrid model that reinforces monitoring data with prediction information to support both short- and long-term actions.

This answers the research question Q3.2 in Section 1.3.

We conduct a series of experiments to evaluate our workflow monitoring, prediction, and adaptation using various scenarios executed over a cloud cluster. This fulfills a set of real-world monitoring processes and datasets where resource shortage is contingent to workflow performance degradation.

Our trust model satisfies all functional and nonfunctional requirements throughout the two stages of end-to-end Big Data workflow QoS enforcement. Some of the functional requirements for the cloud service provider selection include: 1) guarantees dynamic trust score updates because it supports both periodic and event-driven update strategies, 2) the historical records are maintained because each transaction is logged in a specified database, 3) credibility validation is provided through our community management system, and 4) the collection of reputation information is performed dynamically with reputation request messages broadcast to community members. Additionally, the functional requirements through the second stage of guaranteeing QoS requirements

of workflow orchestration include: 1) supports the monitoring and adaptation of cloud workflows to guarantee the required level of QoS, 2) supports self-reconfiguration and self-adaption of cloud workflows, and 3) uses performance and resource usage prediction to avoid service degradation and performs corrective actions to maintain the required QoS levels. The complexity overhead of the trust evaluation algorithms is analyzed, which is lightweight and does not endure high processing overhead on the user, community members, and cloud providers. Moreover, our model handles the dynamic nature of cloud resources and services and copes with the complexity of workflow monitoring and adaptation. To capture different dynamic properties of the workflow and the cloud execution environment, we formalize the cloud resource orchestration using a state machine with validation using a model checker.

1.4 The Organization of this Thesis

The remaining chapters of this thesis are structured as follows:

Chapter 2: Background. Introduces the background knowledge of Big Data, Cloud Computing, trust, and trust models in the cloud and Big Data.

Chapter 3: Literature Review. Presents a systematic literature review of QoS-aware cloud service provider selection approaches classified as trust-based selection approaches, non-trust-based selection approaches, and other QoS-aware cloud service orchestration approaches.

Chapter 4: Big Data Workflow Quality Specification Model. Details our end-to-end multi-dimensional quality specification and trust assessment specification for Big Data workflows. The model combines both data-driven and process-driven quality evaluation for Big Data workflows. This chapter includes our paper published at BigData Congress 2016.

Chapter 5: Towards a Multi-Dimensional Trust Evaluation Architecture for Cloud Service Provider Selection. Details the design and implementation of our proposed multi-dimensional trust evaluation architecture for cloud service provider selection. It emphasizes the details of our cloud service provider selection trust model formalization. This chapter includes our paper submitted to IEEE Access as well as our papers published at AFRICATEK 2017 and BigData Congress 2017.

Chapter 6: Trust Enforcement Through Self-Adapting Cloud Workflow Orchestration. Describes an end-to-end trust model framework for orchestrating Big Data workflows. This model provides QoS enforcement on workflow orchestration through automatic monitoring, adaptation, and trust evaluation according to user's preferences. This chapter includes our paper submitted to Future Generation Computer Systems.

Chapter 7: Towards a New Model for Cloud Workflow Monitoring, Adaptation, and Prediction. Details a model that applies prediction of QoS performance and initiate necessary adaptation actions to avoid degradation of service performance. This chapter includes our papers published at IEEE BigDataSE 2018 and IEEE Cloud 2018.

Chapter 8: Conclusion. Discusses the conclusions about the research described throughout the dissertation, and recapitulates the contributions, limitations, and presents proposals and directions for future work.

Chapter 2

Background

The extensive data growth during the past few years and the emergence of the Big Data phenomena have urged the shift from traditional data management systems to cloud-based computational systems.

This chapter provides a brief overview of Big Data with a special focus on its characteristics, quality specifications process models, and the corresponding assessment models. It is organized as follows: Section 2.1 introduces Big Data definition and characteristics. Then, Section 2.2 introduces the Cloud Computing including the service models and the deployment models. After that, Section 2.3 provides Big Data processing research niches and the proposed solutions using Cloud Computing. Thus, it introduces cloud workflows, characteristics, orchestration techniques, and adaptation towards QoS guarantee. Moreover, Section 2.4 introduces the concept of Trust and its properties. In addition, it provides background related information about trust evaluation in Big Data and cloud workflows. Finally, Section 2.5 summarizes the contents of this chapter.

2.1 Big Data

Data is exploding at rates never previously experienced or perceived. The data gathered from different information sources across various application domains has exponentially grown in volume, velocity, variety, and veracity. These new trends characterize the phenomena known as “Big Data.” This extensive data growth has urged organizations and enterprises to shift from traditional data

management systems to cloud-based computational and storage systems. Big Data cloud-enabled systems offer on-demand, scalable, and flexible data management services that are proven to be efficient and cost-effective. In many application domains, the increase of electronic data advocated the use of sophisticated systems to support Big Data transfer, processing, storage, replication, analysis, and retrieval. Consequently, Big Data's inherent characteristics require a high computation processing power to handle the enormous volumes of data that can reach in the petabytes.

2.1.1 Definition

Various definitions of Big Data are proposed in the literature. It is noteworthy that Big Data should not be viewed as a technology, but a phenomenon resulting from the vast amount of raw information generated by society and subsequently collected and used by commercial and government organizations and enterprises [25].

In one of the accepted conventional definitions, Big Data was viewed in terms of the tools, processes and procedures that allow an organization to create, manipulate, and manage very large data sets and storage facilities [26]. Furthermore, Big Data can be defined as the collection of large and complex data sets that are difficult to process using conventional database management systems or traditional data processing application and tools.

2.1.2 Characteristics

The enormous amount of structured and unstructured data available today makes it difficult to process Big Data using traditional database and software techniques. Thus, Big Data requires new techniques for efficiently processing large, heterogeneous, dynamic, and high-speed data given certain time constraints and quality requirements. These techniques encompass several specialized areas including statistics, machine learning, data mining, neural networks, signal processing, pattern recognition, social network analysis, mathematical optimization, and visualization [27].

New technologies built around Big Data drives the need for developing new venues of designing infrastructure components, solutions, and effective processes to provide collection, storage, processing, classification, and indexing to ensure acceptable Quality of Service, scalability, reliability, and security [28].

The challenges facing Big Data systems and applications include, but are not limited to, data collection, storage, search, analysis, and resource sharing. The primary motivations for adopting Big Data systems and applications include the currently increasing number of sensing technologies and tools for capturing data, the cost reduction in collecting such data, and that an increase in technology-aware users multiplies scientific discoveries over time.

Big Data is not only defined by size, but is also characterized by the multiple "V's" of volume, variety, velocity, veracity, validity, volatility, and value [1], [2], [3], and [4].

- **Volume:** The massive amount of data generated by humans and machines, sensors, networks, human interactions, and social media.
- **Variety:** The data varies in type from being structured, semi-structured, or unstructured. Data today takes the form of not only databases and spreadsheets but also emails, audio, photos, videos, and monitoring devices, for example. The problem arises when starting the process of storing, processing, and analyzing data to have meaningful information and, hence, taking appropriate decisions accordingly. Being unstructured makes the issues above harder to implement and process.
- **Velocity:** The speed at which sources generate data, such as through sensors, machines, networks, business, and human interaction from social media and mobile devices. The stream of data is enormous and continuous, and the velocity of real-time data has many benefits like helping business and researchers make the right decisions at the right time to obtain competitive advantages and ROI.
- **Veracity:** The abnormality and noise existing in data, such that preventing noisy data from accumulating in the system, in addition to guaranteeing the purity and cleanliness of the stored and analyzed data, is very challenging.
- **Validity:** A characteristic of Big Data for evaluating the correctness and accuracy of data to be used, which assists in making the best decisions.
- **Volatility:** Deciding on the best time to invalidate the data is important for efficient data analysis. Volatility denotes the amount of time the data will remain valid and worth storing.

- **Value:** Describes how a company exploits data.

These special characteristics of Big Data introduce several challenges, such as data collection and integration problems, due to the data being distributed across diverse geographical locations. Moreover, the management, processing, and storage of Big Data also present significant challenges considering the enormous volume and heterogeneous nature of the datasets, and traditional processing techniques are unable to handle such massively heterogeneous and distributed data volumes efficiently.

In the subsequent section, we detail key information about Cloud Computing, which is the embracing environment for Big Data applications management.

2.2 Cloud Computing

Cloud Computing has emerged as a promising and dominant paradigm for managing and delivering computation, applications, and services over the Internet [5]. The rapid advent of such a compelling paradigm has already changed the milieu of information technology and started to realize the long-held sought for capabilities of utility computation. Various definitions of Cloud Computing are proposed in the literature. The National Institute of Standards and Technology (NIST) adopted the definition of Cloud Computing as "a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [29].

Vaquero et al. in [30] coined Clouds as "a large pool of easily usable and accessible virtualized resources such as hardware, development platforms, and/or services. These resources can be dynamically reconfigured to adjust to a variable load (scale) allowing also for optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized SLAs" [30].

The powerful processing in the cloud covers a wide landscape of information technology services, such as storage and application services. This powerful processing computation platform has enabled various computationally extensive and scientific applications to perform vast experiments

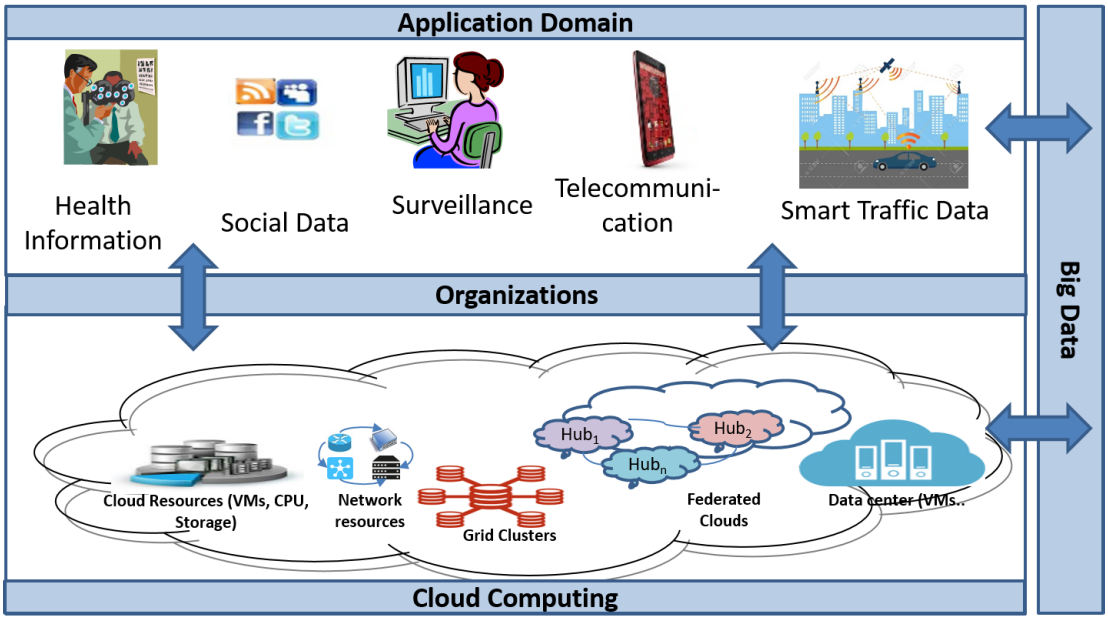

Figure 2.1: Cloud Computing and Big Data.

that were not possible by local servers which lacked sufficient computing facilities. Such a trend has significantly provided a means of lowering the total cost incurred by the pertinent software systems [5]. Figure 2.1 illustrates Big Data and Cloud Computing.

A major objectives of Cloud Computing is to make hardware and software in the data centers accessible to users via a "pay-as-you-go" approach. Public users can virtually have unlimited resources upon request at any time. Knowing this, scientists use the term "Utility Computing" to refer to the "*product*" that the Cloud Computing provider delivers [31]. This virtual infinite computing resource availability releases cloud users from the burden of resource provisioning. It has the advantage of releasing the upfront commitment for small businesses to buy hardware and software resources because these are available in the cloud as needed [6].

Users can find within cloud services some virtualization of computation, storage, and communication models that are available to any application in the cyberspace [32]. Clouds can be used for many applications such as mobile interactive applications, parallel batch processing, business analytics, and running mathematics software packages [6].

Amazon, eBay, Google, Microsoft and other leading technology companies provide scalable

Cloud Computing infrastructures suitable for Big Data processing, such as MapReduce, the Google File System, BigTable, and Dynamo [6]. BigTable is an example of Cloud Computing infrastructure that is a reliable distributed storage system for managing structured data at Google. It is capable of scaling to enormous data volumes (up to petabytes) and a large number of servers (up to thousands) while maintaining high performance and high availability. More than sixty Google products use BigTable, including Google Analytics, Google Finance, Personalized Search, and Google Earth [25].

2.2.1 Cloud Service Models

The three cloud service models are PaaS, SaaS, and IaaS [33] [5].

- **PaaS** offers on-demand platforms for application development including tools for application design, development, testing, integration, deployment as well as hosting and other development related tools [33]. Examples of PaaS are Google's Apps Engine, Salesforce.com, and Microsoft Azure [5].

- **SaaS** is a licensing and delivery model in which software applications are exposed to customers on different cloud servers. Service providers charge businesses for the time and number of users and not for hardware [33]. Examples of SaaS are Google Docs, Gmail, Salesforce.com, and Online Payroll [5].

- The **IaaS** service model provides virtualized computation resources over the Internet on a per-use basis [33]. Examples of IaaS are Flexiscale and Amazon's EC2 [5].

2.2.2 Cloud Deployment Models

The cloud deployment model represents a specific type of cloud environment, which is differentiated by the access, ownership, and size of the customer organization. Customers prefer to access the computing resources with respect to the scale, availability, and cost [29]. The following describes four cloud deployment models [29] [34]:

- **Private Cloud:** A single organization operates the cloud exclusively. The organization can maintain and operate the cloud or cooperate with a third party to perform such functionalities.

The infrastructure can be located physically on the organization premises or geographically distributed.

- **Public Cloud:** : It is used by the general public or a large industry group. The owner is an organization providing cloud services (a business, academic institute, government organization or a combination). The cloud infrastructure is typically located physically at the service provider. The resources are provided to customers as a service. Public cloud is the more commonly used deployment model.
- **Hybrid Cloud:** A mixture of the private and public cloud deployment models.
- **Community Cloud:** Used by a community or group of users with a common goal or interests, such as mission, security requirements, policy or compliance considerations. The cloud can be operated by a community or a third party.

2.2.3 Prospects of Cloud Computing

Cloud Computing has numerous desirable advantages and prospects including giving the opportunity to organizations to concentrate on the core business without distraction with matters concerning resource availability or infrastructure [5]. Moreover, in the fields of science and engineering, Cloud Computing open-source infrastructure and programming tools allow researchers and engineers to build sophisticated, complex applications in relatively shorter time. It was reported that an application, which took several years to build previously, was developed as an eight-week course project by Berkeley undergraduates using cloud resources [31]. Cloud Computing facilitates better and faster research, provides high-performance computing, and enables transfers of Big Data.

2.3 Cloud Computing and Big Data

Cloud Computing has gained much attention from the research community as an important application environment for Big Data [35]. It is also a promising design paradigm for Big Data processing as Big Data has such a huge volume, it makes good use of distributed storage provided by the cloud infrastructure. Moreover, Cloud Computing supports applications using visualized technologies,

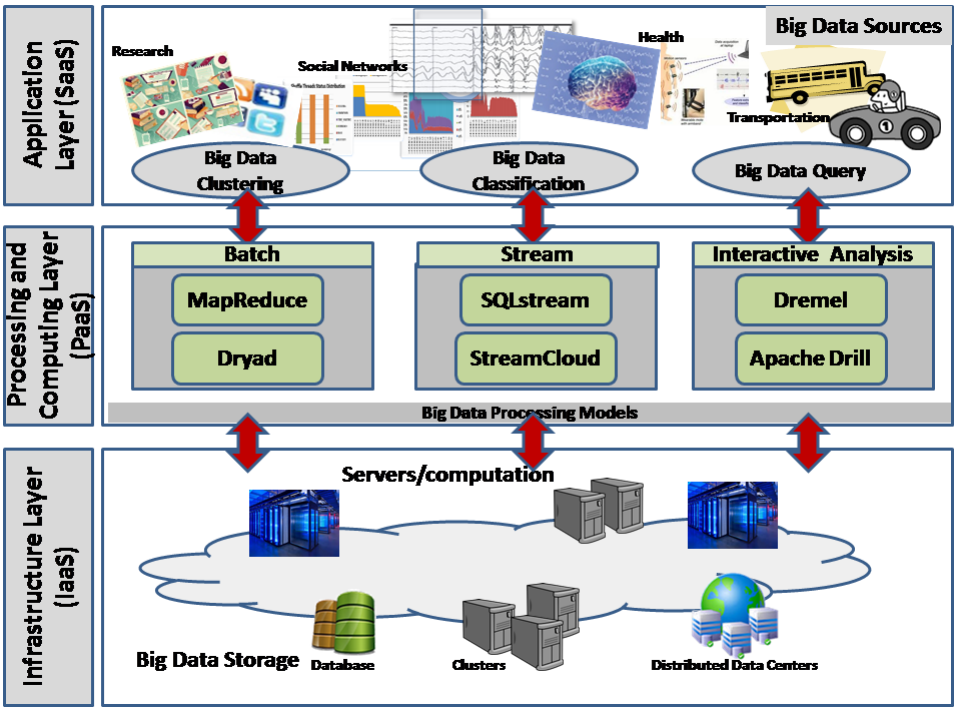

Figure 2.2: Big Data and Cloud Computing framework.

which helps efficiently evaluate Big Data. Consequently, Cloud Computing can be viewed as a service model to Big Data that offers prominent computation and processing capabilities. Big Data requires parallel processing that is usually expensive and unfavorable for adoption by medium-sized enterprises. However, cloud service providers can provide these facilities with an affordable budget [33]. The authors in [5] summarized the correlation between Cloud Computing and Big Data as "Cluster computing which exhibits good performance in distributed system environments, such as computer power, storage, and network communications."

Figure 2.2 shows the layered architecture of Big Data and Cloud Computing as an application layer, processing and computing layer, and infrastructure layer. The application layer consists of Big Data applications emerged from areas such as healthcare, transportation, scientific research, and social networks. The processing layer provides the processing platforms for Big Data, such as batch, stream, and interactive processing. The infrastructure is the lowest layer of the architecture providing storages facilities (e.g., servers, data centers, and clusters) for Big Data. Cloud-based

technologies and applications help build enterprise infrastructure to support workflow and operational procedures related to processes monitoring and data processing with increased ease and speed [28].

In the following sections, we introduce Big Data processing research niches and the proposed solutions using Cloud Computing. Our main focus will be on the research issues related to resource management, performance optimization, and cost minimization of Big Data on the cloud.

2.3.1 Research Trends in Big Data Processing in the Cloud

Big Data processing involves many challenges, among which is the difficulty of data collection and integration since it is distributed across geographical locations. It is challenging to manage and store this collected data with respect to the vast volume and the heterogeneous nature of the datasets. A higher degree of challenge is reached when facing the issues above while guaranteeing functional and performance assurance, especially in terms of fast retrieval, scalability, and privacy protection. Also, the cloud service providers are concerned with profit maximization, while application users are concerned with cost minimization to fall within allocated budgets [36]. Tremendous effort has been devoted towards addressing Big Data challenges and issues related to social, ethical, and technical perspectives. A major crucial challenge is Big Data processing in the cloud. Performance optimization is one of the classic and significant issues in a Cloud Computing-based environment because the adoption of suitable optimization techniques will allow better application experiences with enhanced system resource consumption [36].

2.3.2 Cloud Workflows

The evolution of service composition dates back to the early software engineering discipline, where pieces of software code (or programs) were developed and executed sequentially or in parallel over different application servers, which then might be composed and integrated into a different server. Following Service Oriented Architecture (SOA) principles, services composition techniques, languages, and tools provided abstractions, constructs, and runtime facilities to define and orchestrate composite services.

However, this type of composition lacks dynamicity in handling real-time errors and coping with

the changes that might take place within the runtime environment. With the emergence of Cloud Computing technologies and mechanisms, similar composition schemes were utilized through configuration files developed using deployment tools, such as Docker, to build resource deployment workflows. These workflows usually included a defined sequence of procedures of how data can be processed and stored in various execution environments. Some additional features encompassed resources monitoring, automated testing, and resources migration across cloud services. These composition techniques lacked real adaptation for complex situations where a task (or activity) cannot be executed due to violations in policy regulation or data privacy.

2.3.2.1 Definition and Characteristics

Cloud-based systems have been extensively used for web and business applications. However, managing and running massive data workflow applications in the cloud are not yet highly adopted. Currently, workflow management in the cloud is represented as batch scripts on a programming model, such as MapReduce, or other scripts that connect the output of a specific service to another.

We define a workflow to be the automation of a domain-specific application process composed of a set of tasks or services aggregated and executed either in sequence or parallel for collaborating and managing data flow to achieve a certain goal according to specified rules. When the workflow application domain follows the characteristics of Big Data (e.g., data-intensive), then it is named Big Data workflow. Furthermore, in this work, we use both task and service interchangeably to refer to composed elements of workflows.

The term "Cloud Workflow" can be defined as the "specification, execution, and provenance tracking of scientific workflows, as well as the management of data and computing resources to enable the running of scientific workflows on the cloud" [37]. Therefore, throughout this thesis, we use both cloud workflow and Big Data workflow interchangeably to refer to Big Data-enabled systems deployed and executed over the cloud.

Workflows display many characteristics that signify requirements. Classifying and detailing these characteristics enable improved engagement of solutions in terms of planning, management, and resource provisioning in the light of workflow requirements and constraints. One approach presented a detailed multi-dimensional workflow characterization model that classifies workflows

according to different aspects, such as size, resource usage, structural pattern, data pattern, and usage scenarios [38]. The following summarizes these aspects:

- **Size:** Measured with the number of tasks composed in a workflow, width and the length. The width is represented by the number of parallel paths signifying the concurrency level. Also, the length, which is expressed as the number of tasks in the longest path, determines the turnaround of the workflow.

- **Resource usage:** Measured in terms of the computational requirements, such as a number of concurrent processors, and computation time required for processing the workflow. The resource usage is also determined by the size of data required for input, output, and intermediate data generated.

- **Structural pattern:** Workflows exhibit different structural patterns, such as sequential, parallel, split, and merge patterns. A workflow is known to be sequential if its tasks are executed in sequential order. On the contrary, a workflow is considered parallel if its tasks can run simultaneously and concurrently. Other patterns include parallel-split in which multiple tasks depending on the output of one task and parallel-merge in which many tasks merge into one task or the combination of both. Additionally, a workflow can be structured as a mesh where the task dependencies are incorporated and defined.

- **Data pattern:** The data plays an important role in workflows during their lifecycle phases. Data varies in types, sizes, and association to the workflow as input, intermediate or output. The authors in [38] classified the data patterns of workflows as data reduction, when the input data size is larger than the output data size, data production, when the output data is greater than the input data of the workflow, and data processing, when the data is changed but the difference in size between the input and output is not significant.

- **Usage scenarios:** Workflows have different usage scenarios, which can be interactive, event-driven or user constrained. The workflow is considered interactive when the user is involved in the execution of a workflow. The workflow is event-driven when it has a dependency on external events, such as new input data patterns, and user constrained workflows to partake

constraints applied by users, such as time limit or budget.

Management of cloud workflows through workflow orchestration necessitates special attention to all the aforementioned characteristics to fulfill the goal of satisfying the required quality of service levels.

2.3.2.2 Orchestration

Over the lifecycle of the workflow, cloud resources are managed through many processes and services that are involved, including resource selection, configuration, deployment, monitoring, and control. These workflow management processes and services are referred to as cloud resource Orchestration [9]. Consumers perceive the functionality of orchestration as an abstraction layer that focuses on resource management and services, such as deploy, monitor or scale-up operations, rather than emphasizing the details of the resource infrastructure [39]. Cloud resource orchestration frameworks apply service-oriented models to allow users to utilize and consume available resources according to their requirements. In this view, the main objective of cloud resource orchestration is to guarantee successful hosting and execution of workflows by satisfying the user's QoS requirements [9].

Monitoring is generally conducted to confirm that the provided QoS satisfies the SLAs and triggers adaptation to respond to performance degradation. Most of the existing monitoring frameworks are not designed to accommodate the workflow detailed QoS-specific requirements that adapt to the dynamic processing requirement of Big Data workflows. In such workflows, a variation in one activity impacts the overall performance of the entire workflow. In addition, they do not comprehensively support monitoring and integration of workload input, performance quality characteristics, and SLA in different levels through all activities of Big Data workflows as well as identifying the SLA violations' root causes based on QoS performance logs collected from data flows [40].

2.3.2.3 Adaptation Based on Monitoring and Prediction

As previously mentioned, the ultimate composition goal is to respond dynamically to specific application needs with a declarative and automatic workflow that is self-configurable, adaptive, self-learning, captures event and status changes of runtime environment components, and automatically

recover using corrective actions.

- **Self-reconfigurable and adaptive workflows**, are continuously monitored in real time to detect QoS degradation or service failure and is then dynamically reconfigured. Reconfiguration tasks include adding new VM instances to respond to heavy loads on services that might exceed beyond a defined certain threshold. A workflow that dynamically adapts to runtime environment changes, and, accordingly, implements high-level reasoning of runtime environment properties and autonomic orchestration tasks. Such adaptation tasks include implementing security policies to protect integrity and privacy of sensitive data as well as real-time error detection and repair.

- **Self-learning workflows**, learn from past execution to build some experience, for example, in the form of rules (e.g., ***if*** *CPU utilization is greater than 90%* ***then*** *add a new node to the cluster*), and use these to self-protect during future executions of similar situations. This requires a mechanism to record and analyze the environment component states, capture event patterns, and abstract them into expressive models. Examples include characterizing states of an application or a service, state of a specific application component, and the behavior of users from specific geolocation.

2.4 Trust and Trust Models in Cloud and Big Data

Numerous models have been recently developed to build trust between consumers and cloud service providers. It is difficult to identify a precise definition of trust, which can be described in various ways. Trust is defined differently depending on the manner in which it is perceived. In other words, trust is defined by characterizing the trustee and trustor. In [8], trust is defined as "the expectation of a cloud consumer regarding the actions and behavior of a CSP that will affect the consumer's choice in the selection of a CSP." The authors in [41] described common trust definitions, including reliability trust, which is the subjective probability by which an individual expects that another individual will carry on a certain action on which its welfare depends.

According to [42], trust is modeled as the function of a trustor and trustee pair that results in a trust level value. In this context, three aspects of trustworthiness are proposed: the trust of web

services, trust of users, and trust of network transmission. Appropriate trust scores are given to satisfactory/unsatisfactory service and compliant/noncompliant customers. In this thesis, we refer to the service requestor as the trusting entity and the cloud service provider as the trusted entity.

Trust in [43] is evaluated using either a policy-based or reputation-based scheme. Other approaches classified trust evaluation as being either a direct or an indirect evaluation relationship [44]. Policy-based trust evaluation weighs recorded interactions between entities having direct contact with each other. In contrast, reputation-based trust evaluation elicits the referrals or recommendations of other entities while considering the history of interactions between those entities [43], [44]. Reputation is defined in the Concise Oxford dictionary as "what is generally said or believed about a person's or thing's character or standing." This type of referral can be used as a measure of trustworthiness [43]. In other words, the better the reputation, the higher the degree of trust.

Selecting the *best* cloud provider for the processing and storage of private data is challenging. A service requestor typically performs a trust evaluation of a cloud provider and its services before any interaction that might involve sharing or transferring sensitive and critical data to the provider's cloud either for processing computations or storage capabilities. The diverse methodologies, mechanisms, strategies, and conventions used to assign trust value for Cloud Computing services are known in the literature as "trust models" [45].

A credible trust model should have the following characteristics and prerequisites: 1) sufficient trust evidence for user ratings as an exposure for different levels of service quality, 2) the ability to support user's preferences for selecting different quality attributes for trust evaluation, and 3) a moderation capability to reduce the impact of dishonest users who falsely provide erroneous or biased ratings and, thus, reduce the model's impact [46].

Measuring QoCS attributes is not an easy practice where, in many cases, users lack the applicable technical knowledge that enables them to establish an effective and qualitative judgment. Within this context, the Cloud Services Measurement Initiative Consortium (CSMIC) proposed the Service Measurement Index (SMI) [47] as a measuring model. SMI is a set of business relevant Key Performance Indicators (KPIs) that is considered a universally adopted metric for Cloud Computing related quality attributes. Although the SMI is an advancement geared towards standardizing cloud QoCS, it falls short in addressing other important attributes that are desired by service consumers

and service providers, such as location information, feedback, and reputations [48].

Distinguishing between trust systems and reputation systems is worthwhile. Trust systems result in a score value based on a subjective view of trustworthiness. However, reputation systems generate a score based on public reputation from a specific community perspective. The work in [22] describes the difference between trust and reputation as the former being mainly "personal and subjective" and the latter being "public and combined." For example, a person might trust a system with a bad reputation based on a previously positive mutual interaction.

Trust can also be defined as personally expected behavior that might take place in the future. If this type of trust is evaluated numerically, it signifies the level of trustworthiness of a system to achieve a required task [22]. According to [41], there are many classes of trust including access trust, identity trust, and provision trust. In this work, we are interested in provision trust, which evaluates the service and resources of cloud service providers. For instance, the specification of quality requirements for the delivery of services is considered to be provision trust in the present study. If a user provides an evaluation based on past subjective experience, then it is described as a subjective measure. If an evaluation is done according to a formal assessment, then it is considered to be an objective measure. The main problem with subjective measures is their high probability of incorporating unfair evaluations into a model.

A system's reputation can be measured by the feedback and the associated ratings given by the users of the system and, thus, viewed as an indication of reliability. Reputation systems can be either centralized or distributed depending on the degree of interaction and coordination. Distributed systems allow users to submit their opinions and experiences with different stores. The participants are responsible for collecting ratings from different sites as well as from other participants. Thus, it is impossible and prohibitively expensive to collect the total ratings from all the available distributed sites. Instead, the collected ratings are considered to be a subset of the total ratings.

Intensive research efforts have focused on the reputation mechanism as a key factor for managing trust to enhance the Quality of Service of Big Data applications. Trust is an important consideration for workflow management to improve results, satisfactory performance and failure avoidance [49]. Many trust and reputation models have been proposed in the literature. Within the web arena, trust is needed to distinguish between similar services' functionalities offered in the market while

adhering to specific quality requirements. Since not all provided services perform as expected, the selection of a service provider should involve a careful review of the history of the claimed services. More often, a published service quality and functionality do not adhere to the claimed features, thereby unreliable and irrational selection may burden users with complications that can result in higher expenses with lower Quality of Service.

There are many issues with Big Data trust in its lifecycle from the data collection stage, processing, analysis, and through to the usage stages [50]. The literature on trust evaluation modeling for Big Data and Cloud Computing is still in its infancy, and extra effort is required to make it more comprehensive.

Trust in this work is defined differently compared to what have been commonly used in the literature. It combines multiple and various quality dimensions and attributes related to Big Data workflows that are specified by the user. Each quality dimension and its significance towards the trust score evaluation is selected by the user through a guided application. Measurements of contributing quality attributes relies on the aggregation of three main dimensions: self-experience based on recorded historical transactions, computed reputation from neighboring community, and provider advertised resource qualities. In the next section, we further study and describe trust.

2.4.1 Properties of Trust

Few surveys in the literature describe the many properties of trust, such as subjectivity, dynamicity, and context dependency [51]. Trust by nature is subjective because it depends on a user's opinion based on personal perspective and preference. However, the objective assessment of trust, which depends on real evidenced measurements, may be challenging to achieve due to incompleteness and uncertainty factors. Subjective assessment is usually studied using probability set theory and fuzzy set theory techniques [8].

Another property of trust is dynamicity where the trust is subject to elapsed time, amount of interaction, external factors like authority control and contract rules, and the decay of physical resource capabilities over time. This necessitates the periodic refreshment of trust evaluation.

Trust also is context dependent because an entity can be trusted in a service domain but not in another. This property is modeled in various works in literature such as [52], [8], and [53].

2.4.2 Functional and Nonfunctional Requirements of Trust Models

Trust models should support multiple functional and nonfunctional features to guarantee reliable trust evaluation. Authors in [8] provided some of these features that we use as a benchmark for our trust model, including a dynamic trust score update and up-to-date transaction history loggings. Moreover, especially for reputation systems, functional requirements include credibility validation and dynamic collection of reputation information.

The nonfunctional requirements refer to quality features such as performance and model complexity. In this thesis, we conform to the meaning of performance, as in [8], which is measured by two metrics: the accurate analysis of QoCS offered by the cloud service provider and the detection of malicious behavior in the cloud. The first metric is evaluated based on the direct measurement of QoCS attributes or collection of feedback. The second metric is achieved using credibility weights or the effects of majority agreement between community members. The trust model should not impose high complexity to not add extra processing overhead.

2.4.3 Trust Evaluation in Cloud Workflows

Cloud workflows are composed of many tasks that can run on one or multiple clouds delivered by different cloud service providers having diverse quality levels. Generally, users require different service types, for example, processing and storage, with different levels of QoS, such as minimum cost or total execution time. However, cloud environments exhibit high dynamicity and variety in terms of resources and services making it challenging for users to obtain their required quality levels. Thus, evaluating cloud service trust is necessary to support such requisites.

Trust evaluation of a single service can be achieved through the propagation of reputation evaluation conducted by users based on historical experience. However, trust evaluation for service composition becomes more sophisticated because of the complexity of evaluating the trust of each component service separately. Despite this complexity, trust evaluation supports intelligence, scalability, and adaptive composition solutions for large-scale, highly dynamic, and orchestration frameworks to guarantee the quality of service requirement.

Few initiatives were proposed in the literature which used trust to enhance workflow scheduling,

orchestration, and management [54]. Furthermore, limited research was conducted on workflow orchestration that integrates features such as real-time monitoring, workflow auto-configuration, and QoS guarantee during execution. This is because monitoring involves a high complexity to accommodate the context-aware changes in real-time environments.

Service trust evaluation in federated and interconnected cloud environments is more sophisticated [55]. Customers and different cloud providers need to trust each other to be able to collaborate. Thus, it is essential to evaluate the trustworthiness of cloud and cloud federations [56].

2.5 Conclusion

Over the past few years, Big Data has attracted the attention of both academia and industry. Big Data processing is still a challenging and time-consuming task that requires sizeable computational resources. Cloud Computing addresses these challenges by offering cost-effective, reliable, and efficient resources that can be consumed upon request. Moreover, it allows infrastructure scalability according to dynamically changing demand. Big Data cloud workflows are endowed with orchestration techniques, which provide elastic, quantifiable, and service management control geared towards guaranteeing the required QoS levels. Enforcing trust through a Big Data value chain and cloud workflows is effective in provisioning improved QoS performance.

In this chapter, we established key concepts and knowledge about Big Data, Cloud Computing, cloud workflows, and trust for Big Data and its enabling promise of the Cloud Computing paradigm. In the subsequent chapter, we further survey and analyze additional research initiatives related to these topics to identify open issues for actively improving the end-to-end QoS of Big Data workflows.

Chapter 3

Review of Related Work

Cloud Computing has enabled the implementation and success of Big Data workflows. The latter has special characteristics that require a high-level of performance in processing and storage. Ensuring and preserving the significant quality of service levels for large-scale cloud workflows derives the necessity of monitoring the quality through different workflow management phases. These major phases are the selection of a trustworthy cloud service provider for provisioning the required QoS, which is the first phase, and in the second phase maintaining the quality of service through monitoring and adaptation to detect any performance violation due to resource shortage or even cloud service interruption.

This chapter surveys and classifies the most relevant work in the literature for QoS-aware cloud service selection as well as QoS-aware cloud service orchestration schemes and adaptation approaches. Furthermore, it identifies the open research areas in this field. This chapter is organized as follows: Section 3.1 elaborates on the existing cloud service provider selection approaches. The related studies are classified and summarized according to strategies and models used. Then, Section 3.2 presents the existing approaches of cloud service and workflow orchestration and adaptation approaches. Finally, Section 3.3 summarizes the findings and draws the related conclusion.

3.1 QoS-aware Cloud Service Provider Selection Approaches

Diverse and competing Cloud Computing environments have made it challenging to provide an automatic and straightforward means to select an appropriate cloud provider that will support Big Data cloud workflows and, accordingly, guarantee a high level of QoCS. Cloud service selection differs from the selection of a web service as the former encompass distinct requirements and QoCS attributes. Cloudorado, RankCloudz, and Intel cloud finders are examples of existing cloud service selection tools available in the industry. These are evaluated in [15], which concluded that although these commercial tools facilitate the selection process for IaaS-based services, they still exhibit many limitations, such as the lack of support for other cloud deployment models, namely PaaS and SaaS. Another issue is that none of these tools offer an "easy-to-understand" explanation for the quality attributes. Additionally, none of these tools keep dynamically updated information about the service providers overlooking the dynamicity nature of the environment. In other words, further enhancements are required in the areas of dynamicity and usability to account for frequent changes of the cloud provider status and user level of expertise.

Existing work in the literature depicts different models for CSP selection. We classify these models into trust-based and non-trust-based. We first survey the trust-based strategies and models followed by the non-trust-based selections models.

3.1.1 Trust-based Selection Approaches Classification

Several models were recently developed for establishing trust between users and cloud service providers. In this section, we propose a detailed classification of trust-based selection approaches in the cloud as depicted in Figure 3.1 We first classify the trust enforcement strategies, including trust score computations and evaluation approaches. Next, we propose a new classification of different trust models existing in the literature.

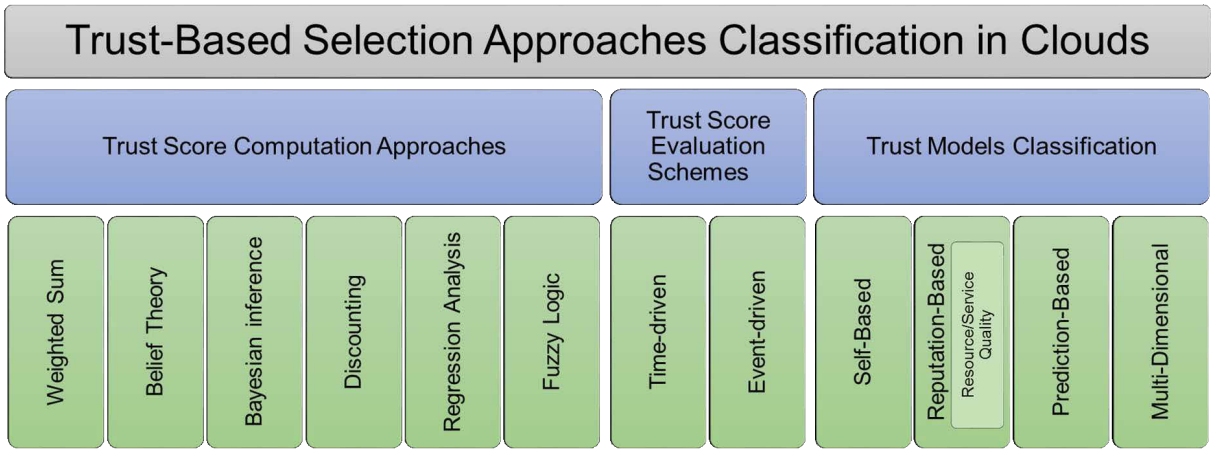

Figure 3.1: Classification of QoS trust in clouds.

3.1.1.1 Trust Enforcement Strategies

In our QoCS trust model classification, we describe the QoCS trust in clouds in terms of the trust-based quality computation methods and trust evaluation strategies used. For the computation methods, we classify trust models based on the computation algorithm used for measuring trust. Further classification is based on the trust score evaluation trigger, which can be time-driven or event-driven. Detailed descriptions of these models are provided in the following sub-sections.

3.1.1.1.1 Trust Score Computation Approaches

Different approaches are proposed in the literature for trust computation in different application domains. A simple way to evaluate reputation-based trust scores is to calculate the difference between the number of positive and negative ratings. This easy-to-understand approach was used in eBay's reputation forum [57]. However, it can lead to ineffective results due to its simplicity. A more sophisticated approach, used by many commercial websites such as Epinions and Amazon, calculates the average of all the ratings. A similar approach involves calculating a weighted average of all the ratings where the weights are based on the rater's credibility, age, and the distance between the new and existing ratings. A weighted sum trust calculation was also used in [58]. According to [59], other types of computational reputation models include Bayesian approaches [60], Regression Analysis [61], Belief Models [62], [63], Fuzzy Models [64], [65], [66] and Flow Models [67], [68].

However, not all the approaches above are used for cloud provider trust evaluation because of unsuitability or have simply not been tested. The different computation methods are also associated with how the trust scores are scaled, which include binary, discrete, nominal scale, and continuous values [51].

One problem with some of the trust score evaluation methods is that they are based on sophisticated and time-consuming mathematical models. Such time-consuming trust models, which are either service-oriented or resource-oriented, exhibit certain limitations, such as non-dynamicity and a lack of real-time adaptability. These limitations make them undesirable for Big Data and cloud environments which require coping with dynamicity and fast decision-making. Other trust models focus only on reputation aspects, which can be misleading if the users are untrustworthy. Other approaches have used assigned weight-value measures that, in most cases, are not necessarily dynamic and suitable to the user's choice.

3.1.1.1.2 Trust Score Evaluation Schemes

Trust score evaluation is related to the frequency of updating the trust score value. Studies in the literature either undergo trust evaluation periodically to revive the trust score, after a transaction or upon request [59]. This periodic update is needed in cases with no existence of events or transactions leading to obsolescence of the QoCS information. A fade factor is used to determine how new are the historical logs are, as some strategies give higher weight to newer records to reduce the emphases of the older records [17].

In this work, the proposed trust model adopts a hybrid approach that combines the two selection strategies of periodic and event-driven. Periodic strategy relies on the cloud provider's willingness to provide users with up-to-date information about the cloud resources. However, the event-driven strategy is executed upon receiving requests from users. The two strategies might be implemented concurrently to assure accuracy of data used to compute trust scores.

3.1.1.2 Trust Models Classification

Trust models are classified into four categories as described in [69]: self-managed case-based, SLA-based [30], [69], [70], broker-based [71] and reputation-based [72]. These approaches are all based

on continuously monitoring the SLA to maintain trust in a dynamic cloud environment.

A self-based trust model consults a recorded history of service provision to utilize the user's demonstrated experience with the service provider. However, a reputation-based trust model is based on the opinions and experiences of other users with respect to service providers. Reputation-based models can be further classified into service quality-based and resource quality-based models. A reputation service quality-based model evaluates the trust of the QoCS and is typically performed on the SaaS layer. In contrast, a reputation resource quality-based model uses the quality of the cloud resources to evaluate the trust of the cloud service and is typically performed on the IaaS layer. Other models may integrate self-based, reputation-based, and provider-advertised-based approaches commonly known as a multi-dimensional trust model.

Other classification initiatives have relied on the perception of either the user, provider, or both to define a trust model. For example, in [44], a trust model is proposed based on evaluating the functional and non-functional properties of cloud services (QoCS) from the perspective of both the provider and consumer. The authors in [73] classified the trust models into policy, reputation, recommendation, and prediction. In this section, we suggest a new classification of trust model strategies to include self-based, reputation-based, prediction-based, and multi-dimensional trust models were more than one strategy is adopted.

3.1.1.2.1 Self-based Trust Models

In [74], a trust model was proposed for scheduling service requests according to the SLA set of priorities. A trust monitor, a third party entity, obtains the SLA criteria from the user, then monitors the performance of the scheduled requests, alerts the user upon violations, and, accordingly, a rule enforcement is taken. Another model defines a mediation-based architecture that defines the various entities of the SLA agent, cloud consumer component, cloud service directory, and cloud provider. The entities collaborate to select a cloud provider and suggest it to the user from a list of trusted ones based on their SLA [70]. A trust framework was proposed in [75], based on a multi-layer monitoring scheme. The monitoring component tracks the communication that takes place between the user and the providers. When a violation is detected, the trust module at the provider side will handle it internally. A trust approach proposed in [17] adopted historical service usage records as

the basis of trust evaluation and employed the Last-K algorithm wherein only the newest K records were used to calculate the trust score. However, this approach can result in decreased accuracy due to the limited number of attributes used, such as the time of invocation, while ignoring other important attributes, such as user input and user location.

The authors in [76] proposed an approach based on game theory to evaluate trust combining both resources and users perception. Other approaches used game theory to model trust for data-intensive cloud federations as depicted in [21], [77], and [78].

3.1.1.2.2 Reputation-based Trust Models

We classify reputation trust models into service-oriented and resource-oriented according to the type of quality attributes used as a basis to evaluate the trust score.

Various research initiatives focused on service quality-based reputation trust models. In [79], the authors recommended a registry and discovery system that keeps track of service providers and feedback from credible service providers and users. The credibility of a service provider is measured as the period over which the service is provided divided by the number of times the service is offered. However, user credibility is measured by the duration of their engagement with the service. A trust score is then calculated using the standard deviation, which is inversely proportional to trust.

In [80], they evaluated the trust score of a cloud resource based on multiple QoCS attributes. However, the weights are manually and nearly uniformly assigned, so it was inflexible to user quality preferences for services. In [64], a fuzzy logic approach to calculating the trust score of a service provider based on user recommendations is proposed. Users collected the recommendation information and stored at a third-party repository. The collected information was combined with SLA monitoring information, and the trust value and probability of service failure were calculated. The authors in [81] introduced a Trust Management System (TMC) for mobile ad-hoc clouds that calculated the reputation trust values of cloud nodes based on availability, neighbor evaluation, response quality, and task completeness. In the approach proposed in [45], trust values were calculated based on QoCS attributes, such as accountability, skills, service reliability, cost, performance, security, privacy, and usability. Other researchers introduced algorithms to calculate trust values based on QoCS attributes by users' experience with QoCSs, rather than their opinions [82]. They recommended two

adaptive modeling algorithms, the rough set and an Induced Order Weighted Averaging (IOWA) operator, to calculate trust scores. The advantage of the rough set is that, unlike traditional models, the weights of the QoCS attributes are not assigned subjectively. The advantage of the IOWA operator is it uses time series for trust evaluation, thus adapting to the dynamic nature of the cloud.

In the context of Big Data and Cloud Computing, the approach proposed in [52] adopted a Category-based, Context-aware and Recommendation incentive-based reputation Mechanism (CCRM) for improving veracity and protecting data against internal attacks. A dynamic trust evaluation model with dual consideration of user preferences and false ratings is proposed in [83]. The authors in [84] proposed a trust evaluation methodology for grid and cloud resources using a resource broker wherein a suitable grid or cloud is chosen according to user requirements. However, only simple factors, which did not cover the complexity of the trust evaluation, were used for the trust score evaluation [82]. A trust model was developed in [85] to enhance file transfers between the nodes of a private cloud while the trust score was calculated based on node storage space, the operating system, network bandwidth and processing capacity. The authors in [86] proposed a trust framework for cloud service selection named TRUSS, which combines objective and subjective assessments based on QoCS monitoring and feedback ratings. Other research initiatives also combined objective and subjective models for evaluation of trust [58].

The authors in [87] apply fuzzy logic methodology to evaluate reputation-based trust scores for CSPs. The model includes subject quality attributes, such as security, that undergo three common fuzzy-logic stages: fuzzification, inference engine, and defuzzification to reach quantitative output. The security attributes included in the model are compliance, access control, auditability, and encryption. Availability and trustworthiness are the basis for reputation-based CSP selection proposed in [88].

Another issue that faces the reputation-based trust model is the malicious information that can be generated from different cloud users. Malicious feedback can cause additional problems for reputation models, which the authors in [18], [89], and [90] proposed solutions that focus on the *'majority of ratings'* concept in which a user is considered a trusted entity if their opinion agrees with the majority of the recorded feedbacks. However, malicious users can still impose their biased opinions by submitting a large number of fake reviews [22]. In [91], a trust model was introduced

to improve the QoCS provided by the cloud, IaaS specifically, based on certain parameters such as the processing capabilities of the VMs, i.e., processing speed, fault rate, bandwidth, and price. However, only IaaS was considered, and no benchmarking study was performed to compare the obtained trust results with other trust models.

3.1.1.2.3 Prediction-based Trust Models

Prediction-based trust models typically use statistical techniques for trustworthiness evaluation and prediction [73]. They are convenient for cases where there are no previous historical recorded interactions with the cloud service provider. The capabilities and the historical reputation of specific service providers are closely studied and, accordingly, the inherent algorithm predicts the service providers' corresponding behavior. These approaches use Fuzzy logic, Bayesian inference, or logistic regression models to estimate the trust of service providers as the probability of providing satisfactory QoCS to users [59]. It is noteworthy that these models are often used when there is no previous historical record of interactions with the cloud service provider. They are also resilient to false reputation attacks especially the logistic regression models that are known to detect outlier values [61]. The Bayesian inference approach is widely used since it considers trust as a probability distribution with a simple and strong statistical basis. However, the belief discounting technique is prone to false attacks [59]. The fuzzy logic based models consider the approximation for trust evaluation within a range between 0 and 1 rather than as binary sets. It is widely used despite incurring some high implementation complexity and low malicious behavior detection [8].

3.1.1.2.4 Multi-dimensional Trust Models

In this work, the trust score strategy is viewed as single-dimensional and multi-dimensional trust models. The single-dimensional trust models use a single strategy to evaluate the overall trust score, such as considering service quality or resources quality. On the contrary, the multi-dimensional trust models combine more than one strategy to evaluate trust, which is more comprehensive as it provides a higher coverage of trust criteria. Examples of multi-dimensional trust models were elaborated in [52], [83], and [58]. In [52], the authors proposed a category-based and context-aware

reputation-based trust model that integrates economic Vickrey-Clark-Groves recommendation incentive scheme for defending against internal attacks and bad mouthing reputation. The latter did not provide implementation and lacked a detailed system framework design. However, the authors in [83] proposed a trust model for web service selection framework to allow user QoS preferences in addition to false rating detection. Moreover, in [58], the authors defined two models for trustworthiness management in the Social Internet of Things. Each node is responsible for calculating trust subjectively or objectively.

Another approach proposed a taxonomy for trust evaluation in cloud service providers which identifies the main aspects, characteristics, and factors for developing a trust framework [92]. A multi-dimensional trust framework named SelCSP was proposed in [93]. It evaluates CSP trustworthiness according to the history of interaction with the CSP that is either self-based or reputation-based. The authors in [94] utilized SelCSP to check the trustworthiness of CSP in addition to producing encrypted identity keys and adopting a symmetric encryption algorithm to encrypt data to improve the security of the framework.

Unlike these proposed approaches, our model uses a triple-strategy by considering cloud resources quality, self-experience, and reputation strategies for trust evaluation. The resource information is required to evaluate the cloud provider objectively, and depends on the self-evaluation as an important factor to match the personal preference of self-context and environment. In addition, the reputation assessment is desirable especially if the user has no prior experience with the provider. We enforce the user preferences with emphasis on Big Data processing requirements for our reputation assessment, as detailed in Chapter 5.

3.1.2 Non-trust-based Selection Approaches

Not all cloud services selection models use trust as some use reputation information collected from the community or third party centralized intermediary entities (e.g., brokers and facilitators). Other models use description languages known as declarative-based models for selection decisions while some use prediction techniques for cloud service selection, and examples of these are described in the following. Our classification of non-trust-based selection approaches in the cloud is depicted in Figure 3.2.

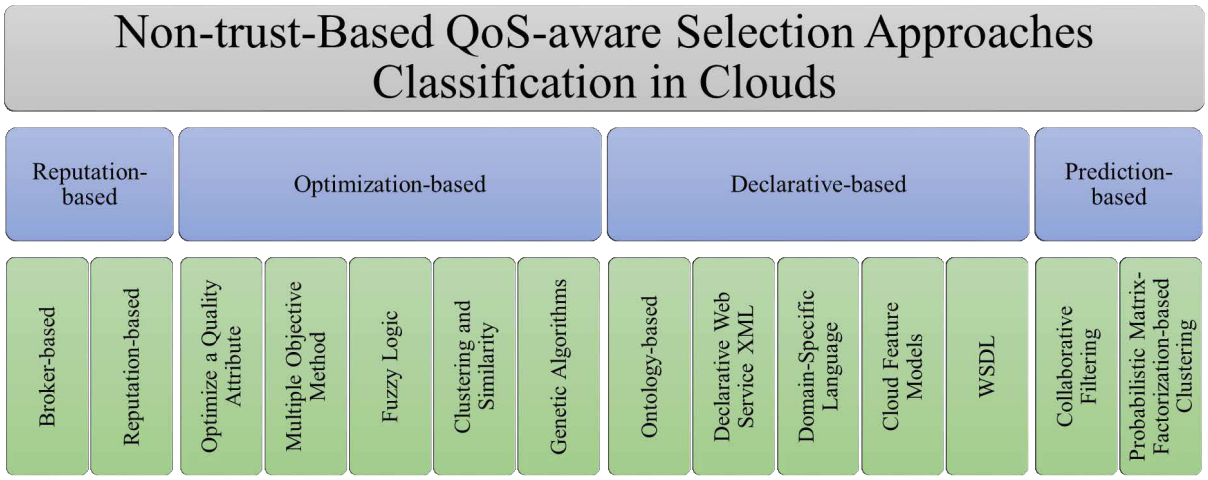

Figure 3.2: Classification of non-trust QoS in clouds.

3.1.2.1 Reputation-based and Broker-based Selection Approaches

A broker-based system is described in [95] where the authors proposed a multi-attribute negotiation to select services for the cloud consumer. The quality data is collected during predefined intervals and analyzed to detect any quality degradation, thus allowing the service provider to allocate additional resources if needed to satisfy the SLA requirements. Another broker-based framework was proposed for monitoring SLAs in a federated cloud environment [96] with monitored quality attributes measured periodically and checked against defined thresholds. Additionally, in [97] a cloud service broker system with a single portal for the cloud service broker, cloud service provider, and cloud service consumer was proposed.

Similarly in [98], a broker-based model was proposed to support the selection of desired cloud services. They presented a Dynamic Cloud Service selection strategy (DCS) based on an adaptive learning mechanism. Different layers are used, including the user layer, cloud service broker layer, and cloud service resource layer. The cloud brokers in the broker layer evaluate and update the performance of the cloud resources. The proposed model uses clusters of brokers to reduce the service selection computation time. Another broker-based CSP selection framework was proposed in [99], that allows a large number of CSP information indexed at the broker for faster retrieval. Clustering is adopted for CSPs with similar characteristics according to user preference and requirements.

Reputation systems are also adopted for cloud service provider selection. Authors in [100] proposed a reputation-based selection framework that applies Triangular Intuitionistic Fuzzy Numbers (TIFNs) with the MADM methodology for ranking different providers.

Depending solely on reputation information and third-party brokers for CSP selection does not holistically reflect the user requirements and preferences and, thus, it is further recommended to adopt other models that put forward more weight on user preferences and special quality attributes characterizing Big Data workflows for CSP selection.

3.1.2.2 Optimization-based Selection Approaches

Optimizing the performance is a significant issue in Cloud Computing environments. In other words, better resource consumption and enhanced application performance will be achieved when embracing the appropriate optimization techniques [36]. For example, minimizing the cost or maximizing one or more performance quality attributes.

Various optimization techniques were developed in the literature for cloud service provider selection. The authors in [101] and [102] used optimization techniques to select a Cloud Data Center for multimedia applications according to required Quality of Service levels as well as minimizing cost. They proposed a priority-based heuristic approach to select among multiple data centers.

The authors in [103] introduced another cloud service provider selection that maximizes the benefits in terms of provisioning data storage considering an accepted budget. The authors provided a mathematical formulation that defines the objective functions and the cost of maximizing the data availability and minimizing price and failure probability. They consider the cloud service provider selection as a knapsack problem and solve it using simple dynamic programming.

In [104], a formal model was proposed for cloud service selection where the objective is to not only the cost but also the risks (e.g., cost of coordination, and cost of maintenance). In this evaluation, the model studies different cost factors, such as coordination costs, IT service costs, maintenance costs, and the cost of taking risk. Furthermore, the risks are denoted in terms of integrity, confidentiality, and availability.

The authors in [105] proposed a QoS-aware cloud service selection to provide SaaS developers

with the optimized set of composed services to attend multiple users having different QoS level requirements. They used cost, response time, availability, and throughput as different QoS attributes. The ranking of services is evaluated using integer programming, skyline, and a greedy algorithm providing a near-optimal solution.

In previous work, automatic cloud infrastructure selection and virtual machine allocation based on proximity as well as cost were presented in [106]. The authors optimized the overall distance considering the data centers' and users' distribution and the relationships between application components. The problem is modeled as an integer linear programming problem that minimizes both the distance and cost of deployment.

In [107], the authors model cloud service selection as a multi-objective p-median problem according to pre-defined optimization objectives. Their objectives are to optimize the QoS, the number of provisioned services, the service costs, and network transmission costs simultaneously in the given continuous periods. The model also supports the dynamic changing users' requirements over time.

Location information was used in the QoCS evaluations in [108], [109], and [110], which based their service recommendations on location information without considering the different weights given to historical QoCS records. In [111], the authors proposed to use a multi-objective optimization approach to allow users to make accurate decisions based on the completion time and price QoCS attributes. Their approach lacked a complete framework that incorporated a quantitative weight model to emphasize recent historical QoCS records over older ones. The authors in [112] incorporated the IaaS, PaaS, and SaaS service subjective quality attributes based on user preference and applied fuzzy rules based on training samples for evaluation of cloud services quality.

A resource management framework is proposed in [113] using a feedback fuzzy logic controller for QoS-based resource management to dynamically adapt to workload needs and abide by SLA constraints. Also, fuzzy logic was adopted in [114] to allow for a qualitative specification of elasticity rules in cloud-based software for autonomic resource provisioning during application execution. A CSP ranking model was proposed in [115] based on user experience, and service quality using an intuitionistic fuzzy group decision making for both quantifiable and non-quantifiable quality attributes to help users select the best CSP conferring to their requirements.

Another cloud service recommendation system was presented in [116] with a selection based on similarity and clustering according to user QoS requirements for SaaS, including cost, response time, availability, and throughput. The users are clustered according to their QoS requirements and are ranked based on multiple aggregation QoS utility functions. Their approach is composed of different phases, starting with clustering the customers and identifying the QoS features, then mapping them onto the QoS space of services, clustering the services, ranking them, and finally finding the solution of service composition using Mixed Integer Programming technology. Li et al. [117] proposed a cloud service selection using a Particle Swarm Optimization (PSO)-based web service with functional and non-functional QoS constraints. They also used time, cost, availability, and reliability quality attributes.

Temporal constraints were addressed by [118] during service composition and runtime. The services are selected dynamically according to temporal constraints using a penalty-based genetic algorithm intended for large-scale and complex service composition. Checkpoints are used to detect violations which may result in process re-planning during runtime. The model involves time, cost, reputation, success rate, and availability in the quality aggregation utility functions. An improved genetic algorithm for service selection was proposed in [119], and in [120], the authors proposed a two-stage dynamic optimization. They first used a queuing network to validate the temporal constraints with respect to the operation time of services; then they design a temporal adjustment model having temporal compensation requirements and adjustment penalties. Finally, they solve the model as an optimization problem using linear programming.

3.1.2.3 Declarative Selection Approaches

Automatic service selection and composition languages are introduced to allow users to declaratively specify composition scripts. To support this objective, the service providers express services using languages and scripts such as Web Service Modeling Language (WSML) [121] and Web Ontology Language (OWL) [122]. Moreover, the users' services and QoS requirements are communicated using the same language. Service selection is performed using ontology-based algorithms, which match the service definitions to the user requirements, or autonomic agents [123].

The authors in [124] proposed a brokerbased cloud service selection framework which uses an

ontology for web service semantic descriptions named OWL-S [125]. In this framework, services are ranked based on a defined scoring methodology. First, the services are described using logic-based rules expressing complex constraints to be matched to a group of broker services. Another service selection system was proposed in [126] where the authors proposed a declarative ontology-based recommendation system called "CloudRecommender" that maps the user requirements and service configuration. The objective of the system is to automate the service selection process, and a prototype was tested with real-world cloud providers Amazon, Azure, and GoGrid, which demonstrated the feasibility of the system.

In [127], a declarative web service composition system using tools to build state charts, data conversion rules, and provider selection policies was proposed. The system also facilitates translation of specifications to Extensible Markup Language (XML) files to allow de-centralized service composition using peer-to-peer inter-connected software components. In addition, the authors in [128] proposed a storage service selection system based on an XML schema to describe the capabilities, such as features and performance.

Zabolotnyi et al. in [129] proposed SPEEDL, a declarative domain-specific language for cloud resource management event-driven policies creation, which helps in task mapping and allows for scaling policies. Nevertheless, the declarative automated monitoring of cloud services is still in its infancy [9]. The authors in [130] represented the features and capabilities of the cloud services using variability modeling to produce cloud feature models to facilitate the description of the requirements and filtering for service selection purposes. The model included both functional and non-functional features, and further decision-making techniques are applied after the first stage selection is performed. A web service framework proposed by Goscinski and Brock to provide facilities such as service providers publication, discovery, and selection based on dynamic cloud characteristics and attributes [131]. The attributes are defined using Web Services Description Language (WSDL) [132].

Although using declarative help to improve the standardization of the CSP selection process, it requires all stakeholders to learn sophisticated syntax and rules of the language, which imposes an extra burden on the non-technical users and hinders the adoption of such models.

3.1.2.4 Prediction-based Selection Approaches

The aforementioned approaches propose service selection based on QoS assessment using reputation information, self-judgments or measured quality attributes through monitoring or testing. QoS prediction techniques were also used in a couple of works in the literature. Using prediction is particularly useful when missing some quality performance measures. The following are some examples of using prediction techniques for service selection.

The authors in [133] used Collaborative Filtering (CF) prediction approach based on QoS historical records to evaluate the end-to-end performance of cloud applications. The framework takes into account the application's multiple layers of IaaS and SaaS components by first finding the similar properties in both layers, then predicting the combined end-to-end performance. CF was also used in [134] for QoS prediction of cloud services based on user location. The model first applies data smoothing to replace missing values; then a prediction is applied in two steps, one using user-based CF and the second service-based CF for finding similar services. Finally, the prediction results are integrated to increase the accuracy of results.

CloudRec is a cloud service selection framework using probabilistic matrix-factorization-based clustering that was proposed in [135]. Clustering is performed on functional requirements and QoS user requirements based on the community with similar cloud-related features and historical cloud service performance. The clustered information is used to predict the unknown cloud service QoS performance. Experiments showed good prediction accuracy and suitability for cloud environment natural characteristics.

Some of the prediction models used for cloud service selection depends on sophisticated algorithms requiring an enormous amount of processing power that consume long processing times, especially when the number of attributes and data records are significant. In this case, these models are inadequate for performing the CSP selection for large-scale cloud workflows, which have limited time constraints and a large number of quality characteristics. Table 3.1 summarizes the various QoS-aware cloud service provider selection approaches found in the literature.

Table 3.1: Summary of QoS-aware Cloud Service Provider selection approaches.

Trust-based Models		
Self-based	SLA-based	[70] [74] [75]
	Game theory	[21] [76] [77] [78]
	Last-K algorithm	[17]
Reputation-based	Miscellaneous	[45] [64] [71] [72] [79] [80] [81] [82] [84] [85] [86] [87] [88] [91]
Prediction-based	Fuzzy and other statistical methods	[8] [73]
Multi-dimensional	Reputation and context information	[52] [83]
	Self and reputation	[93] [94]
	Miscellaneous	[58] [92]
Non-trust-based Models		
Broker-based	Miscellaneous	[95] [96] [97] [98] [99]
Optimization-based	Minimization or maximization	[36] [101] [102] [103] [104] [105] [106]
	Multi-objective optimization	[107] [111]
	Fuzzy logic	[112] [113] [114] [115]
	Clustering and similarity	[116]
	Genetic algorithms	[118] [119] [120]
	Other	[108] [109] [110] [117]
Declarative-based	Ontology-based	[124] [126]
	Declarative web service XML	[127] [128]
	Domain-specific language	[129]
	Cloud feature models	[130]
	WSDL	[131]
Prediction-based	CF	[133] [134]
	Probabilistic matrix-factorization-based clustering	[135]

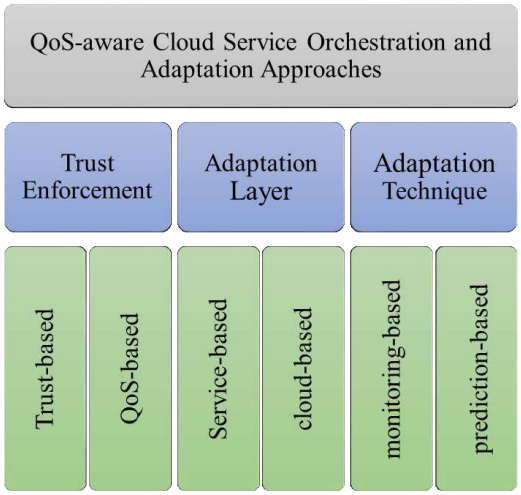

Figure 3.3: Classification of cloud workflow orchestration and adaptation approaches.

3.2 QoS-aware Cloud Service Orchestration Approaches

Cloud systems empower resource control through configuration actions to specify the resource types or quantities. Workflow configuration, re-configuration execution, monitoring, and adaptation over a cloud environment are considered very challenging activities. This is because such activities are resource-aware, require intensive processing, and should adapt to dynamic cloud changes.

In this section, we discuss the existing state of the art on service composition and workflow orchestration including orchestration frameworks with QoS guarantee and trust enforcement in cloud service composition and orchestration adaptation approaches based on QoS and trust monitoring and prediction. Our classification of cloud workflow orchestration and adaptation approaches is depicted in Figure 3.3.

3.2.1 Orchestration Frameworks

Cloud resources and services orchestration provide the runtime execution environment responsible for handling the composition execution of orchestrated cloud resources involved to fulfill the workflow QoS. This refers to languages and models used to represent the configuration, deployment,

monitoring, and control tasks of cloud resource orchestration including, for instance, resource description, orchestration rules, and policies enforcement. Current models and cloud resources description languages facilitate constructs allowing automatic and intelligent policy enforcement and support elasticity to respond to consumer's QoS requirements. State machines are considered a promising model for declarative resource orchestration as they provide a flexible representation of resource requirements while most of the existing low-level languages and scripting orchestration tools fail to do so [9]. Other initiatives include combining cloud orchestration and management standards, such as CAMP, along with adding extensions to CAMP using declarative policies to support end-to-end multi-cloud application orchestration [136]. Docker [137], Juju [138], DeepDive [139], Google Kubernetes [140], and DevOps [141] are examples of these tools and platforms that provide services to translate high-level workflow models into resource descriptions, management rules, and policies which can be interpreted by configuration and orchestration tools. In addition, OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) [23] is a standard for automation of management and deployment of workflows. It uses workflow languages such as the BPMN [142] or the BPEL [143], for developing workflow plans.

Other orchestration models were proposed in the literature, including an orchestration framework from [144] to support multi-tenant scientific workflow management including scheduling and intensive data flow management. Using a metadata-based architecture, it applies the two strategies of a policy-based strategy used by the scheduling engine and semantic-based strategy for describing the data semantics. To support multiple users, the framework defines different layers of metadata such as tenant-specific metadata, common metadata, and data.

Asterism is an open source framework for data-intensive workflow management [145], which supports stream-based dataflow scheduling, storage, and transfer. The authors in [146] proposed an approach to allow a description of cloud service composition using a pattern-based methodology to improve resource orchestration activities. In [147], a workflow management model is proposed that allow large-scale workflow partitioning to support scalability and enhance the performance. An auto-healing framework proposed in [148] includes a suite of customizable plugins for the cloud orchestration function based on OpenStack [14]. A dynamic resource orchestration using a centralized controller to formulate the orchestration as a multi-objective optimal problem using metrics

such as energy consumption, cost, and availability was proposed in [149]. The model uses a particle swarm algorithm to approximate the optimal solutions.

Guaranteeing the required QoS levels across workflow orchestration is challenging due to dynamic changes of resource statuses such as availability and reliability. However, it is one of the critical issues that is of high importance to handle. The following section reviews this open area of research.

3.2.1.1 Workflow QoS Guarantee

The objective of the cloud resource orchestration is to guarantee application execution while maintaining the user's QoS requirements. Usually the cloud resource orchestration methodologies use general-purpose or domain-specific scripting languages to define orchestration strategies [9].

The authors in [150] proposed a contribution-based distribution of reputation approach to propagate the reputation of a composed service to each component service according to the extent to which it contributes to the composed service. The importance or the amount of contribution of each component service towards the composed service is assigned based on its reputation. Typically, orchestration methodologies facilitate describing resources of one provider. Other orchestration techniques support cross-provider resources, such as Compute-Service in JCloud, and are used for configuration and management of federated clouds [151].

Web services frequently undergo dynamic changes in the environment such as overloaded resources. Hence, the authors in [152] proposed a multi-dimensional model, named AgFlow, for component services selection according to QoS requirements of price, availability, reliability, and reputation. The model optimizes the composite service QoS required by the user and revises the execution plan to adapt to the changes in the resource performance. The authors in [153] proposed an SLA renegotiation mechanism to support and maintain QoS requirements in cloud-based systems. They use historical monitoring information including service statuses such as availability, performance, and scalability to predict SLA violations.

3.2.1.2 Trust Enforcement in Cloud Service Orchestration Approaches

Trust models are developed to support monitoring, adaptation and prediction of cloud workflows provisions while guaranteeing the required workflow QoS. Few initiatives were proposed in the literature which used trust to enhance workflow scheduling, orchestration, and management. The following is a review of existing literature using trust to support QoS in cloud workflows.

Recently, the authors in [54] proposed a trust framework that includes an Iterative Adjustment Heuristic (IAH) model to assess trust in composed services. Trust in federated clouds was also addressed in the Sky Computing project [154], which is intended to enable several virtualized sites to increase resource availability. The project studied the trust, VM portability, and connectivity of geographically-spread resources. Bernstein et al. in [155], proposed a blueprint for interconnection of cloud data centers where they addressed issues about virtual machine mobility, storage, network addressing, security in terms of identity and trust, and messaging. However, no trust management was provided in this work.

Few existing cloud federation projects are based on brokering technologies for multi-cloud composed services. Hence, more research needs to be done towards a standardized methodology for handling interoperability and standard interfaces of interconnected clouds [156]. Trustworthiness evaluation models among different cloud providers were proposed and focus on a fully distributed reputation-based trust framework for federated Cloud Computing entities in cloud federation. In this model, trust values are distributed at each cloud allowing them to make service selection independently [56]. Trust modeling was also tackled in federated and interconnected cloud environments [55] where both consumers and different cloud providers need to trust each other to cooperate.

Executing scientific workflows exhibit many challenges including designing a framework architecture with different functionality layers, such as operational, task management, and workflow management. In addition, integration of workflows into cloud systems is a challenging process leading to computational challenges which involve resource provisioning and allocation based on user requirements, i.e., quality of service and error recovery. From a data management perspective, the massive data volumes require special techniques to handle data flow in and out of cloud and data storage, which must account for the location and processing nodes in the cloud. Further,

cloud service management is considered another workflow challenge in handling service discovery and monitoring [37]. Workflow management must consider security management issues that handle data and resource access control.

3.2.2 Cloud Resources and Services Adaptation Approaches: Dynamic and Automatic Workflow Orchestration

Guaranteeing the user required QoS of application execution is the key purpose of cloud resource orchestration. Existing platforms that support Big Data orchestration such as YARN [157], Mesos [158], and Amazon EMR [159], do not handle failure recovery or automatic scaling to correspond to the application changing requirements, such as the data flow changing volume, velocity or variety [40]. Some initiatives proposed automatic scaling of Big Data processing framework as in [160] for batch processing and in [161] for stream processing. Other orchestration frameworks provide online or interactive dynamic reconfiguration [162] [163].

Self-healing is referred to here as the capability of a workflow to recover its functionality when a problem occurs during execution while guaranteeing the QoS level requirements. Recent research approaches endorse automatic self-optimization workflow orchestration realized by dynamic resource re-configuration to fulfill QoS requirements [9]. An example of an autonomic cloud orchestration engine is CometCloud [164], which supports the integration of local and public cloud services and the distribution and scheduling of these services according to resource status and QoS requirements, including budget, deadline, and workload. The authors in [165] proposed a self-healing Web Service Composition algorithm using a QoS performance-aware prediction technique. Moreover, Schulte et al. in [166] proposed a fuzzy BPM-aware technique that scales according to VM KPIs.

Current resource allocation techniques and existing frameworks do not support the dynamic and heterogeneous nature of clouds and resource behaviors. Therefore, the need to provide autonomic Cloud Computing methodologies that allow better resource allocation based on user QoS requirements as well as failure recovery during runtime is becoming inevitable. Researchers use various key QoS parameters for QoS-aware clouds, such as price, time, and response time. Most optimization techniques rely on the evaluation of time and price while other important QoS attributes

(e.g., data privacy) are not considered. The authors in [167] pointed out some QoS parameters used in autonomic Cloud Computing, including scalability, availability, reliability, security, cost, time, energy, SLA violation, and resource utilization. Other research approaches focus on user requirements, such as unit cost per resource, the processing speed of VMs, SLA levels, geolocations, and device capabilities of end-users.

The authors in [168] proposed a dynamic service selection and execution due to irregular activities such as changing requirements or performance. They use a Markov decision process [169] to estimate the performance of tasks and recompose the workflow in case of degradation. However, they only consider the sequential composition.

A middleware architecture was proposed by Ferretti et al. in [170] to dynamically reconfigure cloud resources and services according to some QoS requirements specified in the SLA. Monitoring is used to support dynamic management, load balancing and reconfiguration of resources allocation features. Moreover, a quality-aware framework, named Q-Cloud, is suggested in [171] were resource allocation is performed at runtime. The key requirement is to guarantee QoS among multiple workload applications, and the framework uses QoS states to support different levels of application-specific QoS assignments. The authors in [148] proposed adding extra modules to support the auto-healing capability to a common cloud service orchestrator.

The authors in [172] propose a Business Process-as-a-Service (BPaaS) as a dynamic orchestration framework that supports rule-based adaptation to maintain required service levels. Additionally, the rules can be altered manually by experts or automatically by the framework evaluation environment. In this framework, the history records of adaptation actions are stored and analyzed for optimal reconfiguration actions.

3.2.2.1 Types of Adaptation

In this section, we classify the adaptation procedures according to the way the workflow is adapted or reconfigured. Some of the proposed works in the literature perform the adaptation at the service level where the services are replaced or recomposed when a violation is detected. Another way considers the cloud components by including the infrastructure and cloud-based application layers. Accordingly, we review the proposed works in the following sections.

3.2.2.1.1 Service-based Adaptation

In this section, we survey the adaptation frameworks where the emphasis is on the service or task level, and the adaptation actions target the services by performing service re-composition or service substitution.

The authors in [173] proposed a declarative framework for monitoring and adaptation of complex service-based applications in the cloud. These functionalities and rules are provisioned through the use of an introduced Domain-Specific Language (DSL) called MONINA. Deployment optimization is performed by using a binary integer quadratic programming problem and takes into consideration the dependency weights, runtime, and overhead. The authors use event processing queries to perform monitoring while using action rules for adaptation. However, the model was implemented as a prototype and not evaluated through experiments.

The work in [174] introduced a formal adaptation model across multi-layer applications. The methodology uses adaptation techniques through the use of templates, such as BPEL processes or services. The templates invoke interfaces, such as WSDL, to handle application mismatch types as well as taxonomies of mismatches for classifications of common layer-specific mismatches. However, this methodology requires high expertise for implementing application-specific requirements.

CLAM is a cross-layer adaptation framework for Service-Based Applications (SBA). Adaptation decisions are recommended through multiple SBA layers based on composition, execution time, and cost [175]. An integrated multi-layer monitoring and adaptation framework was presented in [176] to avoid discrepancies when monitoring and adapting single layer application independently. This framework supports holistic loop-back monitoring and adaptation through multi-layer service-based applications. The adaptation is applied to dynamically deployed BPEL processes, and the framework features four components. The first component monitors using sensors to collect runtime information about the data, software, and infrastructure elements of the system. The second component analyses the monitored data. The third component includes multi-layer adaptation action formations based on CLAM. The fourth component is responsible for applying the adaptation actions. Additionally, supporting predictive adaptation is a recommended feature of this model.

A cross-layer service monitoring and adaptation framework was presented in [177] where event-patterns are matched to adaptation actions according to specified rules when a problem is detected.

The framework support logic-based mining for history logs to avoid problems, such as higher service execution time or low memory. However, not all cloud-based layers are supported by the framework.

3.2.2.1.2 Cloud-based Applications Adaptation

Some cloud-based application adaptations were just simple monitoring based adaptations that are performed at the infrastructure level. Other sophisticated adaptation frameworks target multiple layers of the cloud-based applications, among which is a framework in [178] called Axe, which is an approach that enables the monitoring of virtual machines across multiple cloud providers based on the PaaSage project [179]. The collected monitoring data is analyzed according to user-defined rules to generate application adaptation actions.

The system proposed for cloud-based applications in [180] facilitates simple user elasticity requirement specifications, monitoring, and control of cloud services across multiple layers. The main module in this system is the elasticity control service responsible for collecting the monitoring information and generating elasticity plans, including adaptation actions that can be applied to the application, and service plans. However, the platform layers are not supported by the system, so no automation for adaptation processes is performed.

3.2.2.2 Adaptation Methodologies

Workflow adaptation is based on monitoring or prediction approaches. The following subsections survey the existing adaptation schemes presented in the literature.

3.2.2.2.1 Monitoring-based Service Composition and Workflow Adaptation Approaches

Monitoring is defined as gathering and analyzing events and performance logs and is necessary for supporting the management of unpredicted and undesired behaviors [9]. It is typically adopted to guarantee the required QoS by the SLAs and maintain stable performance by responding to quality degradation. Existing cloud resource monitoring tools, such as Nagios, CloudFielder, and Splunk are used by DevOps to describe SLAs, recognize glitches, and issue alarms when violations occur [181] [182]. Other Big Data monitoring frameworks like Ganglia [183], Apache Chukwa

[184], Sematex [185], and SequenceIQ [186] provision QoS metrics information, such as resource utilization (cluster, CPU, and memory) in addition to application types (disk, network, and CPU-bound) [187]. Alhamazani et al. proposed a multi-cloud application QoS monitoring framework capable of monitoring sub-application distributed components, such as databases and web servers [188]. The authors in [189] proposed a multi-layer monitoring framework that allows dynamic and self-configuration of the monitoring quality attributes and time intervals. The QoS attributes are collected from both the application layer and the infrastructure layer. Adaptation decisions, such as resource provisioning, are taken when violation events occur during runtime. Other cloud QoS monitoring frameworks were presented in [190], [191], [192], and [193].

Most of the monitoring frameworks do not support the Big Data workflow specific QoS requirements, such as time sensitivity or task dependency. They usually monitor the workflow as a black box without involving the details of activities as in Amazon CloudWatch used by Amazon Elastic Map Reduce [40]. Such requirements involve data flow behavior and sub-activity process monitoring. Activities in these workflows implicate continuous variations that affect other dependent activities and eventually affect the performance of the overall workflow. Present orchestration frameworks do not comprehensively support intelligent monitoring and automatic reconfiguration to respond to QoS violations. Such violations could occur in the context of a variety of inputs and performance quality characteristics throughout all the activities involved in the Big Data workflows. Additionally, intelligent monitoring should identify and handle the performance violations based on data flow collected logs. Since changes in Big Data workflow activities influence the performance of other dependent activities, they impact the overall performance of the workflow.

In summary, existing workflow frameworks do not provide holistic support for all aspects, i.e., monitoring, integration of workload input, and performance quality characteristics through several activities of Big Data workflows as well as detecting QoS violations along with the main causes [40]. Cloud resource orchestration platforms aim to execute applications while guaranteeing a user's QoS requirements. Techniques used in cloud resource orchestration are based on general-purpose or domain-specific scripting languages [9].

Another QoS-aware workflow adaptation framework was proposed in [194] to evaluate monitoring data collected from applications and resources followed by reaching a corrective adaptation

decision according to provisioning policies at runtime. Experimentation used deployments across different sites and proved the validity of the framework. Monitoring is also supported in MicroCloud, which is a container-based resource management tool to support automatic resource allocation [195]. It supports multi-level adaptation by coordinating the VMs, containers, and platform technologies running within the containers.

3.2.2.2.2 Prediction-based Service Composition and Workflow Adaptation Approaches

There are different research initiatives on workflow resource prediction. For instance, Ramakrishnan in [196] used a probabilistic model to allocate the resource that will meet the required QoS properties, such as availability and response time. The prediction is based on the previous behavior of the system. Statistical techniques, such as clustering, were suggested to categorize comparable task requests and make the appropriate resource quality predictions. Other work was proposed for cloud-based systems where the QoS requirements are maintained through an SLA renegotiation mechanism. Monitoring past service status, such as availability, performance, and scalability are used for SLA violation predictions [153].

"Maestro" is a proposed orchestration framework to support concurrent execution of mobile applications. It replicates critical workflow tasks to support self-healing and avoid failures of devices and services or results in corruption [197]. A comparative analysis of the appropriateness of different Machine Learning (ML) algorithms for predicting cloud resource utilization according to application and system quality attributes was proposed in [198].

Fewer workflow execution frameworks apply both monitoring and prediction techniques for self-adaption of dynamic environment changes to satisfy users' needs [199], which allow for auto-scaling [200]. Dutreilh et al. [201] proposed workflow automation using reinforcement learning for prediction of resource allocation.

The authors in [202] proposed an Adaptive Resource Management algorithm (ARM) for service workflows in cloud environments. This agent-based algorithm performs resource requirement prediction based on periodic monitoring of information, such as load levels. The algorithm also manages the allocation, distribution, and deallocation of resources dynamically to guarantee the workflow performance. The decision of choosing the appropriate prediction model is based on the

least calculated accumulated error value for each model, which can be different for each monitoring cycle.

A MONitoring and ADaptation framework called MONAD was proposed in [203] and supports self-adaptation based on the monitoring of system performance metrics and task allocation statuses as well as prediction of system performance. An updated allocation action is performed upon performance violation detection and uses a multilayer neural network algorithm to provide a powerful approximation of non-linearity and the dynamic workflow to predict the performance in the next time window. The monitoring and prediction are done at the task level to allow more flexibility and scalability of large-scale workflows. Table 3.2 summarizes the workflow adaptation approaches reviewed in this chapter.

Table 3.2: Summary of workflow adaptation approaches.

Adaptation Types		
Service-based	[174] [175] [176] [177]	
Cloud-based	[178] [180]	
Adaptation Methodologies		
Monitoring	[40] [152] [178] [188] [190] [191] [192] [193] [194] [195]	
Prediction	Clustering	[196]
	Machine Learning	[198]
	Auto Scaling	[199]
	Reinforcement Learning	[201]
	Neural Networks	[203]
	Other	[153] [197] [202]

3.3 Discussion

To guarantee end-to-end QoS for Big Data workflow orchestration over competing clouds, we studied the existing work in the literature in this chapter. The review demonstrated that existing work exhibit some limitations related to the cloud provider selection trust models, including non-dynamicity and deficiency of real-time adaptability, which do not fit the Big Data and the cloud environment

special characteristics. Most of the proposed CSP selection frameworks depend solely on one dimension of trust, such as reputation, which does not provide an accurate and optimum selection decision. The literature is missing a comprehensive selection model that covers all layers, dimensions, and components in a multi-dimensional model that satisfies CSP selection for such constrained Big Data and complex workflows. Moreover, among the several methods used to determine the user QoS preference, none exhibit the flexibility to make it responsive to the user's point-of-view as well as comprehends the specific characteristics related to Big Data.

Additionally, this review shows that orchestration of workflows is still in its infancy, which is not suitable for the growing complexity of Big Data workflows. Therefore, existing orchestration frameworks do not completely guarantee the prerequisite levels of QoS of Big Data workflows and have limited adaptation actions. Consequently, extra efforts are needed to establish QoS and avoid service degradation or interruption.

None of the surveyed studies comprehensively handle trust-enforced monitoring of workflow status at different decomposed services, cloud resources, and granularity levels. Hence, comprehensive modeling for all stakeholders of Big Data workflows is inevitable for the provisioning of automatic adaptation and reconfiguration systems to evade workflow QoS degradation or violation. This is emphasized through monitoring and the prediction of workflow performance to obtain the best corrective actions and sustain the required workflow QoS.

3.4 Conclusion

This chapter surveys the literature related to QoS-aware cloud provider selection approaches. It classifies the selection approaches according to trust adoption methodology. The trust-enforced model classifications describe the QoS trust in clouds in terms of trust-based quality computation methods and the trust score evaluation strategy used. A new classification scheme was proposed for different trust model strategies. In addition, it reviews the non-trust QoS-aware cloud provider selections available in existing work into reputation-based, optimization-based, declarative or prediction-based.

Following the review of cloud service provider selection techniques, a survey of related works

was presented for QoS-aware cloud service orchestration and adaptation approaches. We first reviewed the existing orchestration frameworks and approaches and emphasized our review on the trust enforcement strategies on workflow orchestration. Furthermore, we thoroughly reviewed the workflow adaptation techniques classified into service-based or cloud-based according to the reconfiguration layer considered. We also classified adaptation methodologies according to the information used to detect violation into monitoring-based or prediction-based approaches.

Throughout this chapter, the current research focus areas were identified as well as open issues related to cloud service provider selection for workflow processing and workflow orchestration and adaptation with an emphasis on end-to-end QoS enforcement and guarantee were emphasized.

Chapter 4

Big Data Workflow Quality Specification Model

While the potential benefits of Big Data workflow adoption are significant, and some initial successes have been realized, there remain many research and technical challenges that must be addressed to realize this potential fully. The Big Data workflow phases of processing, storage, and analytics provide major challenges and are the ones most easily recognized. However, Big Data workflow quality management is considered another key challenging dimension that is not thoroughly tackled in the literature.

Most of the recent works [204] [205] [206] have proposed few initiatives to incorporate data quality. However, these initiatives remain premature and do not provide comprehensive solutions that guarantee quality in all Big Data processes. Therefore, building end-to-end quality enforcement in Big Data workflows is of vital importance. Cloud infrastructure and services allow implementing QoS enforcement mechanisms for Big Data tasks including Big Data storage, distribution, replication, and retrieval. Such developments consider 1) data provenance and annotation schemes to track the effect of data transformation occurring in each phase, 2) cost optimization schemes for Big Data distribution, 3) QoS-aware Big Data resource allocation and scheduling, and 4) extending Big Data technologies to incorporate QoS enforcement and management of features.

The acceleration in using Big Data workflows that demand a high level of service quality

prompted the need for an end-to-end quality specification framework. Each phase of the Big Data workflows requires a different set of quality properties, attributes, and dimensions starting at the phase of selecting the cloud provider and ending at maintaining the required level of quality during runtime. This involves quality management of the underneath layer consisting of cloud resources and services including deployment, configuration, and adaptation, which require a different set of quality requirements. This also assures the necessity of having a comprehensive quality specification framework.

Forcefully, providing an end-to-end quality specification that serves Big Data workflows by coping with large-scale heterogeneous cloud environments remains a priority in the industry despite limited initiatives in the literature. We argue that having such a quality specification framework will facilitate:

- Leveraging cloud resource configurations for efficient workflow orchestration.
- Capturing, customizing, and reusing existing quality specifications.
- Satisfying expected customized workflow requirements for IaaS, PaaS, and SaaS or having public, private or federated deployments.
- Consolidating quality specifications at different granularity levels, which is mainly beneficial for users having limited technical cloud modeling and management skills. This can be achieved through combining existing quality specification knowledge in a unified, end-to-end model so that users do not go through different dimensions separately by studying, examining, monitoring and managing the quality of low-level, complex, and heterogeneous cloud resources, workflows, and cloud providers.

In summary, to address challenges related to Big Data quality management throughout Big Data workflows, we propose a quality specification model that incorporates and integrates both data-driven and process-driven quality specifications for Big Data workflows. These workflows include tasks such as pre-processing, processing, and analytics. Furthermore, building trust, based on a multi-dimensional quality specification model, allows for a wider, more comprehensive, and more efficient quality assessment as it aggregates multiple and various quality dimensions and attributes.

Hence, we propose a new model which incorporates both quality and trust assessment specifications to deal with quality of data from its inception through analytics while enforcing quality for all Big Data activities.

Moreover, providing help in solving this research problem is beneficial to cloud users and workflow administrators as it serves the following main goals that we identify and detail in the following chapters:

(1) CSP Selection (Chapter 5): providing workflow quality specifications inclusive of different granularity levels, such as data, task, and task compositions that contribute to the workflow trust assessment evaluation. Workflow trustworthiness contributes to trustworthiness assessment of the cloud provider service provisioning. Thus, this assists with CSP selection decisions for Big Data workflows processing.

(2) Workflow orchestration, adaptation, and reconfiguration (Chapter 6 and Chapter 7): quality specification and trust assessment modeling for tasks and workflows enable workflow orchestration and adaptation while guaranteeing required levels of QoS.

The following sections describe our proposed multi-dimensional, quality-based trust assessment specification for Big Data workflows. In Section 4.1, we propose an end-to-end quality specification model for Big Data workflows consisting of first assessing the quality of Big Data by defining a Big Data quality specification (data-driven). Next, we describe the quality specification of the process (process-driven) by assessing the quality of tasks handling the Big Data, which involves, for example, processing and analytics processes. Then, a multi-dimensional trust evaluation approach is introduced and proposes a model for evaluating trust at each composed task across the Big Data workflow. We propose a mapping scheme between QoCS and the Big Data characteristics of volume, velocity, veracity, and variety. We match a set of Cloud Computing quality attributes to those related to Big Data properties to allow users to determine QoCS preferences for CSP selection easily. We also model the quality specification for cloud workflows as an aggregation of each task quality. An illustrative workflow is depicted in Section 4.2 to detail a epilepsy monitoring workflow along with its composed tasks. We detail the workflow quality formal model in Section 4.3 which describes the related QoS dimensions used to evaluate the composite quality of the workflow. In

Section 4.4, we describe our application developed to collect Big Data workflow QoS preferences from the user. Finally, Section 4.5 concludes this chapter.

4.1 End-to-End Quality Specification Model

In this section, we propose an end-to-end quality specification model depicted in Figure 4.1. In our model, we start quality assessment at lower task granularity level to which we evaluate the data input and output quality as well as the process quality aggregated to be used at the trust assessment level. Then, we move up in the hierarchy to evaluate the quality of the Big Data workflow as an aggregation of all composed tasks trust assessments. The trust assessment for each workflow is aggregated to assign trust assessment for the cloud provider hosting the workflow. These trust assessments from the different granularity levels are used according to the required goal of the task. Our first objective of enforcing end-to-end quality assessment is to help users select the suitable cloud provider for processing Big Data workflows performed by the CSP selection module. The selection decision can be realized by ranking the trust assessed for each cloud provider shown in our model presented in Figure 4.1. We detail the CSP selection module in Chapter 5. Then, we continuously monitor the trust assessments for tasks and workflows to satisfy the purpose of maintaining the required quality levels during workflow orchestration, adaptation, and reconfiguration in the case when quality degrade. This is performed by the orchestration and adaptation modules detailed in Chapters 6 and 7.

To implement the quality assessment and the trust assessment, it is necessary to specify the details of quality and trust throughout the proposed specification model. The following subsections detail the quality specification at each level, including metrics and attributes used for quality and trust assessment.

4.1.1 Big Data Quality Assessment Specification (Data-driven)

In this section, we describe the Big Data quality specification, including a description of the quality properties, attributes, and the corresponding metrics. In addition, we explain the quality of metadata, which is considered an essential element for quality assessment of Big Data.

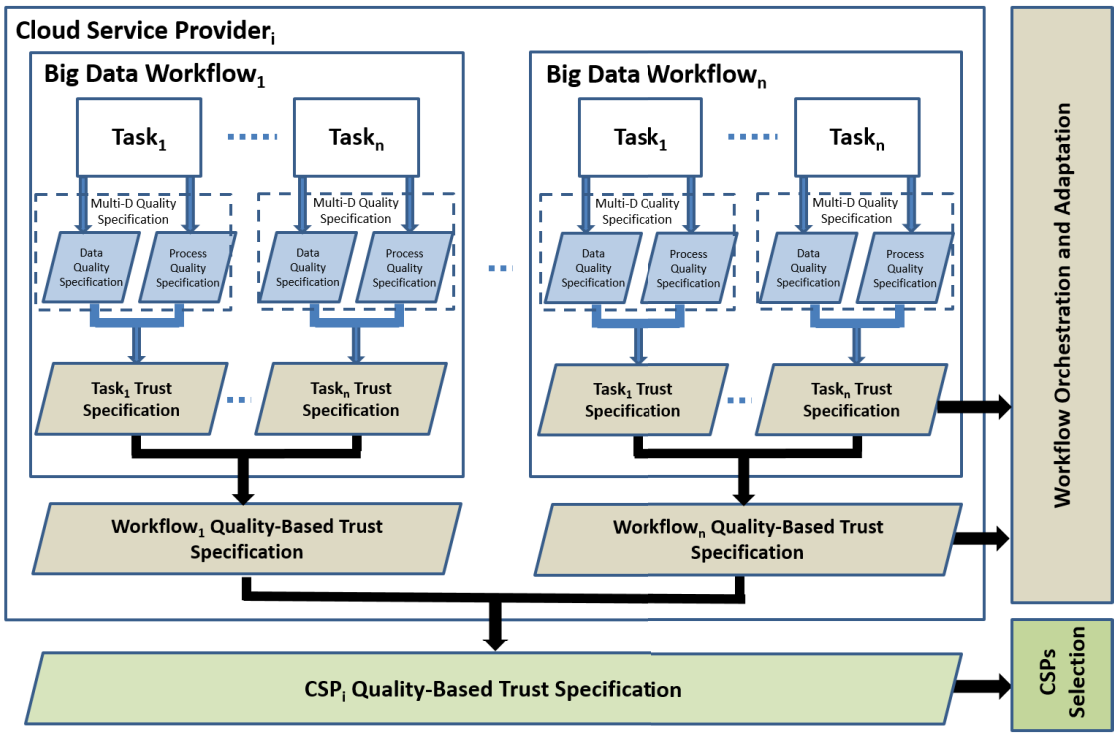

Figure 4.1: End-to-end Big Data workflow quality specification.

4.1.1.1 Big Data Dimensions and Metrics

Data quality dimensions play an important role in data quality assessment. There are multiple definitions of data quality dimensions in the literature. However, they are commonly classified into the two categories of contextual and intrinsic [207]. Contextual dimensions are related to the data values while intrinsic deals with the intention of the data, which is a model-based ontology used to define conceptualization and the associated relationships [208] [209]. Standard quality dimensions discussed in the literature involve timeliness, accuracy, completeness, and consistency [208]. The following are the agreed upon definitions of four well-reputed quality dimensions accepted in the literature:

- **Timeliness:** also referred to as currency and volatility, and is usually related to the age of the data and the degree of its validity in the system or the real world. In other words, it describes how much the data is up-to-date (currency dimension). On the other hand, the frequency of the data value change occurrence defines the volatility dimension.

- **Accuracy:** measures how much the recorded data is correct and resembles real-world values and, hence, is reliable.
- **Completeness:** related to the ratio of missing or null values to the size of the universal relation.
- **Consistency:** the structure and semantics of the data follow a set of rules and constraints [208].

Each quality dimension is characterized by one or more quality metrics as shown in Table 4.1 where some quality dimensions are adopted from [208], such as timeliness, currency, and consistency. Others are newly introduced or altered to integrate the quality specification model proposed in Figure 4.2, such as volatility (VMb) and completeness metrics (CMPMc). In this work, we use the four quality dimensions defined above, as they are very relevant for the application domain example we consider for this thesis of continuous health monitoring. Data acquisition from sensors is very sensitive to such quality dimensions including precision, accuracy, and timeliness. For example, any timely EEG episode may reveal crucial information to disease monitoring while inaccurately collected data may lead to a wrong diagnosis and leading to incorrect clinical decisions.

4.1.1.2 Quality Metadata

Metadata describes relevant information about the data such as provenance, quality, and other details. In other words metadata is the "data about data" [210] that makes it easier and faster to process and extract data features. It is defined as "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource" [211]. Usually, it includes extra information about the quality to help evaluate the data [208], and these quality attributes are referred to as the "quality metadata." Accuracy, timeliness, and consistency are attributes related to the data quality that comprises the quality metadata. Many initiatives studied metadata, as in [210] where the authors provided a comprehensive classification of multiple quality metrics along with their description, purpose, target, evaluation technique, value range, constraints, and applicability.

Table 4.1: Data quality dimensions and metrics.

	Formula	**Description**
Timeliness metrics:		
TMa	= 1 - CMa / VMb	1 - Currency/Volatility
TMb	= numOfProcessedRecs / totalRecs / timePeriod	Percentage of the completed processed records within a time limit
Currency metrics:		
CMa	= currentTime - updateTime	Time of update
CMb	= updateTime - storageTime	Difference between time of update and time of storage
Volatility metrics:		
VMa	= ConstantTimePeriodValue	Time length for which data remains valid
VMb	= (storageTime - updateTime) / totalTime	Volatility: (time of data – time of update)/total time
Accuracy metrics:		
Ama	= numOfCorrectValues / totalValues	The ratio between the number of correct values stored and the total number of values.
Amb	= AvgUsrResponse	User questionnaire
Completeness metrics:		
CMPMa	= numOfEmptyValues / totalValues	The ratio of the number of empty or null values over the total number of values.
CMPMb	= AvgUsrResponse	User questionnaire
CMPMc	= actualTotalSize / expectedTotalSize	The total size of the stored records over the expected size of the data
Consistency metrics:		
CNSMa	= numOfInconsistentValues/totalValues.	The ratio of the total number of inconsistent values over the total number of values
CNSMb	= numOfViolations	The total number of values violating constraints and rules.

Creating metadata needs a domain expert knowledge to define quality policies and rules [210]. During the data extraction stage, quality policies define the acceptable quality attributes and metrics of the data related to given quality dimensions. These quality attributes are evaluated and the results of the evaluation are stored in the specified quality metadata knowledge base. The extracted data is saved in designated data storage while the corresponding metadata can be stored in a different storage space (external) or together with the data (internal) [212]. Another classification can describe metadata as static or dynamic, while static metadata is fixed and has to do with information that does not change with the data and dynamic metadata is continuously changing during runtime [213].

Metadata descriptions are represented using vocabularies that follow well-defined standards and models. The syntax of metadata is defined as the set of rules that govern the structure of its basic elements [214]. Each metadata scheme can be represented using any markup or programming languages having different syntax notations. One commonly used standard is the Dublin Core, which can be written using HTML, XML, and Resource Description Framework (RDF) [215]. Other domain related metadata languages were proposed in the literature, like Ecological Metadata Language (EML) or the Federal Geographic Data Committee Biological Data Profile (FGDC BDP), which are languages that provide a formal description to information that describes ecological data [216]. A more general metadata model is Open Information Model (OIM), which is a specialization of the Unified Modeling Language (UML) related to a particular domain based on the UML, XML, and SQL [217]. Also, Java Script Object Notation (JSON) [218] is a metadata standard used to represent massive data into a format based on property graph models and is a lightweight standard for the Big Data-interchange format.

4.1.2 Quality of Service (Process-driven)

In this section, we introduce the quality assessment specification of the Big Data process and its metrics. To evaluate a workflow quality, we need to evaluate the quality of the data and the quality of service (process) of data handling at each stage as depicted in Figure 4.1. Therefore, we describe the quality metrics related to data processing in Big Data workflows including pre-processing, processing, and analytics. However, the quality evaluation of the visualization process is out of the

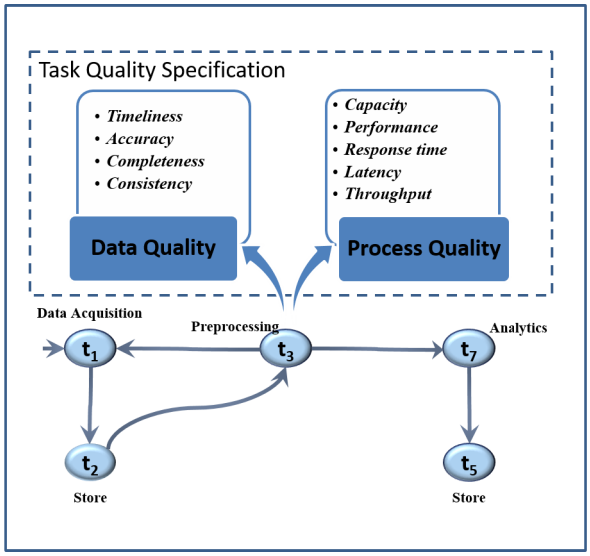

Figure 4.2: Big Data task quality specification model.

scope of this work. The common processing quality dimensions discussed in the literature include:

- **Capacity:** the maximum number of concurrent connections or processes.
- **Performance:** the speed of data processing.
- **Response time:** the maximum or average time to complete the processing of each record (or the total records)
- **Latency:** the total time to receive the processed data (delay).
- **Throughput:** represented in terms of the number of processed records over a time period.
- **Accuracy:** measured by the number of errors resulting from processing the data. Additional quality attributes not considered in this chapter are availability, robustness, and scalability, as they are more specific attributes related to the quality of the hardware and the infrastructure used [219].

4.1.3 Multi-dimensional Task Quality Specification Model

Figure 4.2 describes a conceptual view of the main components that constitute to the Big Data task quality specification model. To illustrate this model, we use a simple Big Data workflow example,

which includes tasks such as data acquisition, pre-processing, storage, and analytics. We model the quality as multi-dimensional where we combine both a data quality specification (data-driven) and a tasks quality specification (process-driven) to model the quality of the composed tasks in the workflow. For example, considering the preprocessing task t_3, we perform pre- and post-Big Data quality evaluations which are the quality of input and output data, then we perform pre-processing task quality evaluation. These quality assessment processes communicate and integrate seamlessly into all tasks to achieve a complete quality assessment of the Big Data workflow. The detail of the proposed quality assessment approach is elaborated in Chapters 6 and 7.

The Big Data workflow incorporates a set of tasks and processes including Big Data pre-processing, processing, analytics, and visualization. These processes generate data, process data, analyze and visualize data within the complete workflow phases. In the following, we describe each of these phases and how they handle quality assessment.

Data acquisition phase: handles data collection from its source, and relays it to the storage and processing location. Quality evaluation in this phase is important for the next phases of the Big Data workflow as it is the starting phase and the input for the rest of the phases. However, it is highly linked to the design of how data is generated from sources, the utilized devices used, the data sampling technique used, the underplaying network, and the communication protocol used. All these might affect the quality of the data collection, including accuracy, timeless, and latency. We do not handle quality of data collection, and we consider it for future work although it might influence the quality evaluation of the remaining phases of Big Data pipeline.

Pre-processing phase:

(1) **Pre- and post-Big Data quality evaluation:** a data-driven quality evaluation conducted before and after a pre-processing task. It aims to measure the degree to which the quality of data improved after pre-processing. In the pre-Big Data quality evaluation, we identify the percentage of incomplete data, inconsistent data, and incorrect data to decide which pre-processing scheme (e.g., cleansing, transformation or approximation) should be applied. Many data quality metrics can be measured and considered to be very important to access the overall data quality. Examples of these metrics include data accuracy, correctness, completeness, and consistency.

(2) **Pre-processing task quality evaluation:** a process-driven quality evaluation that consists of many activities related to data preparation for the next phase in the Big Data workflow (processing and analytics tasks). Due to the diversity of sources, the collected datasets may have different quality attributes including noise, redundancy, and consistency. Additionally, other processes, such as transferring and storing raw data, would have necessary costs that should be considered by the quality specification model. Other sets of quality metrics can be measured to evaluate the quality of pre-processing including accuracy, throughput, and response time.

Processing and analytics phase: consists of immediate exploitation of data after which a supervised pre-processing is complete. Processing may involve the application of data mining methods and machine-learning procedures to lead to a set of target data results. Processing can be centralized or distributed over a cluster or a data center, as it needs highly powerful processing nodes. However, analytics consists of mining large amounts of long-term periods, heterogeneous data, and data from different sources to extract data knowledge, hidden patterns, and unknown correlations, to other useful information to get insights for further decision-making. This phase can apply processing on data through multiple iterations to achieve data refinement. The analytical outcomes can lead to more effective decisions, faster interventions, improved processes efficiency, and competitive advantages over traditional data analytics techniques. The same pre-processing quality metrics (both data-driven and process-driven) can be measured to evaluate the quality of the processing and analytics phase, which may include accuracy, throughput, and response time.

Visualization phase: while this process quality is not evaluated in this work and left for future consideration, the visualization process consists of viewing data resulting from the Big Data workflow execution to support formulating decisions and reporting on continuous updates about the Big Data status. Visualization serves to validate the collected data, to support formulating decisions, and to report on continuous updates of data collected. Data can be presented using different views including a summary of monitoring results, graphs, the pattern of readings, and even reports on discrepancies of measures from which can be generated automatic preventive actions. A set of quality metrics can be used to evaluate the quality of this last process and are mostly linked to user satisfaction and quality of data representation.

4.1.4 Task Trust Specification

In this section, we describe the trust assessment specification based on the multi-dimensional task quality specification depicted in the previous section. To comprehend how trust is assessed for Big Data workflows in the cloud, we first need to detail the relationship between Big Data properties and cloud quality characteristics. Table 4.2 describes our proposed mapping scheme between Big Data properties and cloud quality metrics. These quality attributes are aggregated into trust assessment specifications for each composed task along the workflow and eventually are used to evaluate the trustworthiness of workflow orchestration during runtime.

To this end, we aggregate all quality attributes including data-driven and process-driven to assess the quality-based trustworthiness of workflows. The performance level of all quality properties, whether objective or subjective, are normalized then aggregated using different algorithms, detailed in the subsequent chapters, to generate a trust assessment.

We propose a mapping of some key Big Data characteristics to their related cloud quality attributes in Table 4.2. To incorporate the aforementioned QoCS attributes, we categorize them into the four classes of low, medium, high, and very high, with 1 being low and 4 being very high. Table 4.2 presents the following attributes:

- **Volume:** the size of the data to be processed determines the class of this attribute.
- **Variety:** relates to the type of data to be processed with class 1 comprising structured data, class 2 for unstructured data, and class 3 for mixed structured and unstructured data types.
- **Velocity:** relates to the speed of the Big Data application with class 1 indicating an offline data application and class 4 for the streaming of high-speed data. Classes 2 and 3 represent intermediate speed levels.

The remaining QoCS attributes are measured according to common cloud characteristics used in the literature [45], [82], [220], and behavior observed during communications as follows:

- **Reliability:** the task success ratio equals the total number of task requests less the number of illegal connections and the number of denial of service incidents divided by the total number of task requests.

- **Response time:** the actual execution time equals the time spent between sending a request and receiving the last byte of the response, in milliseconds.
- **Availability:** the ratio of the number of received responses to the number of sent requests.
- **Throughput:** The number of requests handled per second
  $= totalnumberofrequests/(endtime - starttime) \times 1000$,
  where end time = last request response time and start time = first request start time.
- **Confidence:** The degree of confidence in the response time considers the delays caused by external factors on the client side. It is calculated as 1 - σ (response time) / μ(response time), where σ and μ are the standard deviation and mean of the response time, respectively.

Table 4.2: Big Data QoS attributes.

Big Data Property	**Metric (cloud)**	**Description**
Volume	Data size trend	The size of the data to be processed determines the class of this attribute (1: low, 2: medium, 3: high, 4: very high).
	Disk space	Available disk space in the cloud.
Velocity	Throughput	Total number of requests / (end time − start time).
	Response time	Actual execution time.
	Availability	Number of received responses / number of sent requests.
	Resources (memory, processing power)	Available memory and processing power in the cloud.
Veracity	Reliability	Task success ratio, the total number of task requests - the number of illegal connections and the number of denial of service incidents / total number of task requests.
Variety	Data type	Categorized into four classes according to data type (1: structured, 2: unstructured, 3: mixed).

4.1.5 Cloud Workflow Quality-Based Trust Specification

Various QoS properties are used in the literature to evaluate the trust of cloud workflows. Among these attributes are performance, including network and Cloud services [221], privacy, scalability, and extensibility. Other key metrics suggested in [40] involve the following: 1) delay of event discovery and decision making, 2) throughput, response time, and latency of results generation in workflow, 3) distributed file read and write latency, 4) cloud resource utilization and energy-efficiency, and 5) quality of the network, such as stability, routing delays, and bandwidth. In this context, the monitoring system is required to be comprehensive to have a full picture of the problem. In other words, monitoring application parameters measures the high-level health of the system and will help in detecting the most serious issues. Whereas, monitoring the resource parameters allows finding and resolving the root cause of these issues. These quality parameters are monitored through a collection of cloud resources, such as CPU, memory, file system, and network usage statistics including utilization, saturation, availability, and errors. Also, monitoring is applied to some application-specific quality parameters like throughput, success rate (number of errors), and performance. Existing tools used for monitoring cloud resources like processing, storage, and network include cAdvisor, Heapster, InfluxDB, Google Cloud Monitoring, and many others [140]. Table 4.3 summarizes some key metrics for different application types.

Not only are these quality attributes used to evaluate the quality of workflow orchestration during runtime, but they are also used to evaluate the degree of trustworthiness of the cloud providers. Hence, trust management and assessment at different granularity levels, such as task, workflow, and Cloud Service Providers, serve two goals of this thesis:

(1) **CSP Selection:** several quality attributes contribute to the trust score evaluation. To select a cloud provider for Big Data workflows processing, we consider the ability of the cloud to process Big Data with respect to its key Big Data characteristics such as volume, velocity, and variety. Hence, it is essential to consider Big Data quality attributes that are essential factors in selecting a suitable cloud provider.

Table 4.3: Key metrics for popular technologies.

App type	example	Metrix Name	Description	Type
HTTP server and reverse proxy server.	NGINX	Accepts, handled, active	Number of connections requested, successful and active	Utilization
		Requests per second	Number of requests	Throughput
		Dropped connections	Calculated accepts – handled	Error
		Server error rate	Count 4xx and 5xx codes	Error
		Request processing time	Time to process each request (s)	Performance
data store	Redis (in-memory key/value)	latency	Average time (ms) for Redis server response	Performance
		Number of commands processed	Total number of commands processed per second	Throughput
		Memory usage	Total number of bytes allocated by Redis	Utilization
		Fragmentation Ratio	Ratio of memory allocated to the instance by OS to actual memory used by the Redis	Saturation
		Evictions	Number of keys removed by Redis when reaching the maxmemory limit	Saturation
		Rejected connections	Number of rejected connections due to reaching maxclient limit	Error
	MongoDB (NoSQL database)	Number of read requests		Throughput
		Number of write requests		Throughput
		Connections - current	Number of current connections	Utilization
		Connections - available	Number of available new connections	Utilization
		Storage size	Data, index, and total extents storage size	Utilization
		Virtual memory	Virtual memory usage (MB)	Utilization
		Cache memory	storage engine's cache and the filesystem cache size	Utilization
		Number of assertions	Number of assertions on message, warning, regular, and user	Error
	MySQL (relational database server)	Query throughput	Number of executed statements, number of writes	Throughput
		Query run time	Run time per schema	Performance
		Query errors	Numbers of statements with errors	Error
		Threads_connected	Number of current connections	Utilization
		Threads_running	Number of available new connections	Utilization
		Connection_errors_internal	Count of connections refused due to server error	Error
		Connection_errors_max_connections	Count of connections refused due to max_connections limit	Error
		Buffer pool usage	Memory (buffer pool) used to cache data for tables and indexes	Utilization
distributed document store engine	Elasticsearch	Search query total	Total number of queries	Throughput
		Search query time	Total time spent on queries	Performance
		Current queries	Number of queries currently in progress	Throughput
		Indexing performance	Updating index (refresh and flush)	Performance
		Memory usage	Utilization of RAM(JVM heap and the file system cache)	Utilization
		queueing	Number of queued threads in a thread pool	Saturatrion
		Rejection	Number of rejected threads a thread pool	Error

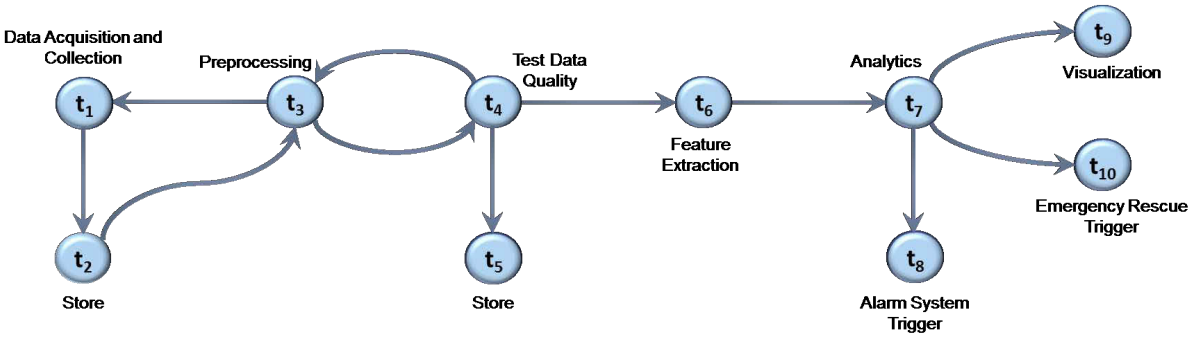

Figure 4.3: Epilepsy monitoring workflow.

(2) **Workflow orchestration, adaptation and reconfiguration:** as illustrated in our specification model depicted in Figure 4.1, trust assessment for tasks and workflows depend on evaluating both data-driven and process-driven quality performance. Hence, studying the relationship between Big Data characteristics and the quality attributes will allow choosing the best quality attributes that fit the characteristics of the Big Data and processing tasks. In addition, it provides a more accurate quality assessment model.

The following section depicts an example of a Big Data workflow to highlight the details of each composed task and how our quality specification model is applied.

4.2 Illustrative Workflow

Our illustrative workflow is about an epileptic patient who needs to be continuously monitored to predict seizures before they occur to take an immediate intervention. Monitoring process should not restrain the mobility of patient both indoors and outdoors. Therefore, multi-channel wireless sensors are placed on the patient's scalp to record EEG signals and send these to a smartphone that allows a patient to move while being monitored. Since recorded data is continuous from different channels, it can result in a Big Data (e.g., 128 EEG channels using a sensing frequency rate of 128 HZ generate approximately 1 GB of data during every hour of monitoring). Nevertheless, smartphones still lack full capabilities to handle Big Data, so Cloud Computing technologies can efficiently enable acquiring, processing, analyzing, and visualization data generated form monitoring. Figure 4.3

describes the epilepsy monitoring workflow.

Brain sensors collect the EEG signals, which are transferred to a smartphone or a back-end server to be processed, analyzed, and visualized to serve the seizure prediction and prevention. The workflow is composed of ten tasks as follows:

Task 1	Data acquisition and transmission is the process where the EEG data is acquired from the scalp by sensor electrodes that measure the electrical activity of the brain and then transfers the signals to a computing environment for preprocessing or to temporary storage.
Task 2	Raw data storage is the process of storing the raw EEG signals.
Task 3	Data preprocessing conducts some data cleansing and filtering activities to remove unwanted and noisy signals.
Task 4	Test data quality conducts an assessment of activities of a set of data quality attributes including data accuracy, completeness, and consistency.
Task 5	Storing the preprocessed data.
Task 6	Feature extraction applies selection techniques to extract relevant features from the EEG signal to support the analytics.
Task 7	Data analysis where techniques are applied to the EEG data to extract meaningful information and insights that will support diagnosis and decision-making.
Task 8	In case a seizure is predicted, an alarm is triggered.
Task 9	Visualization task generates graphical reports to be viewed by different stakeholders.
Task 10	Upon diagnoses of a seizure event, the emergency rescue task is triggered.

4.3 Workflow Quality Formal Model

Big Data workflow aggregates different tasks that exhibit certain requirements such as optimized execution time, and efficient processing power. A workflow instance can be executed by one or more cloud providers [222]. Hence, the quality of a cloud workflow instance needs to be collected from different cloud providers and a variety of resources that creates a complex combinatorial problem. Accordingly, guaranteeing high-quality workflow output from different quality dimension perspectives becomes very challenging. Quality models were designed in the literature to support the lifecycle of cloud workflow instances comprehensively. The main components of existing quality models in the literature are time, cost, and reliability [223]. These QoS models use formal mathematical techniques to estimate the overall QoS for a workflow process by determining QoS for each

task and transition belonging or included to a given workflow.

The following are some of the related QoS dimensions we used to evaluate the composite quality of the workflow made up of the quality of its composing tasks:

- **Time:** the total time needed by a workflow instance to complete a Big Data job. Reducing the total execution time for a set of tasks is the ultimate objective of the user.
- **Cost:** the cost incurred when a workflow instance is executed. The cost is measured by the amount of money paid for executing the job.
- **Reliability:** the probability the tasks will perform as per user expectation and is measured based on success and failure rates.

We model a workflow as a directed acyclic graph DAG $w = (T,\ R)$, where *T* is the set of *N* tasks $\{t_1,\ t_2,\ \cdots,\ t_n\}$ and *R* is a set of *M* transitions between two tasks t_i and t_j so that t_j will not be executed unless t_i is completed. Tasks are represented using circles, and transitions are represented using arrows. We define *P* as a set of *S* possible paths in the workflow. Each path represents a different sequence of tasks performed from the start to the end of the workflow instance and is represented by $P = \{p_1,\ p_2,\ \cdots,\ p_s\}$ where p_i is a sequence of tasks, $t \in T$. Tasks t_i in a workflow Path p_i can follow a simple sequence, parallel sequence or contain loops. The task can be one of two types of processing tasks or storage tasks. Processing task is a task that performs a computational operation on the input data, while storage task is the task of storing the data. Every executed task, $t \in T$, uses data with different sizes. In other words, a task can take different time depending on the data size processed by this task. Hence, each task is then represented in terms of task type and data size *dsz*: *t = (type, dsz).*

The first quality property to model is the execution time, which is measured from the start to the end of the workflow and it is the aggregation of the time of each task in each path *P* sequence. Hence, the time of the workflow can be measured as the maximum path time among the set of paths. The time taken to execute an atomic task is a function of data size *dsz* used by this task t_i and the average time taken by t_i to process one byte of data.

$$Time\,(t_i) = tAvg_i * dsz \tag{1}$$

$$Time\ (w_j) = \max_{0 \le j \le s} \left(\sum_{i=1}^{N} Time(t_i) \right) \tag{2}$$

The second important quality property is the cost, which is measured by adding the cost of each task executed along the workflow path. For the data storage task, the cost is a function of data size and time needed for storage. Hence, the cost is the amount of money paid to store the data. Conversely, the processing task is the amount of money paid for executing that task.

$$Cost\,(t_i)_{processing} = \sum_{i=1}^{n} cp_{t_i},\ \forall\, t\ \in T_{processing} \tag{3}$$

$$Cost\,(t_i)_{storage} = \sum_{i=1}^{n} cs_{t_i} * \ dsz,\ \forall\, t\ \in T_{storage} \tag{4}$$

$$Cost(w) = Cost\,(t_i)_p rocessing + Cost\,(t_i)_s torage \tag{5}$$

In this work, we use Amazon services' price as our reference of $0.15 per Gigabyte per month for the storage resources. $0.10 per CPU hour for the computation resources [224].

The third quality dimension is reliability, which is defined as the probability that the task can be completed successfully. Our proposed workflow model has been designed to cope with the situation where tasks can be handled by different cloud providers. However, in our trust model we only consider that tasks of the workflow will be handled by one cloud provider. The reliability of a workflow w is the product of the reliabilities of the cloud provider executing each task, $t\ \in\ T$. It is the difference between the number of task requests and the failed tasks divided by the total tasks requests.

$$Reliability\,(w) = \prod_{i=1}^{n} R(t_i) \tag{6}$$

In this section, we provided a mathematical formulation model for our illustrative workflow. We also studied the quality of the data used by each task in the workflow and the quality of the process as it is necessary for building the workflow quality specification. Although the above mathematical model is commonly used in the literature and can be used to evaluate the overall performance of the workflow, it does not incorporate the differences between tasks and their quality properties, nor it considers the user preferences. Each task exhibits different quality characteristics that should be

involved in the task quality assessment process. Indeed, there are common workflow properties, which we have modeled above, however, other task specific properties should be considered as well. For example, the quality property 'saturation' is crucial for assessing a task implementing a queueing functionality but is not required for assessing a storage task quality. Hence, we cannot aggregate this property among all tasks in a workflow as it is not significant for some of them. In addition, the quality properties differ in significance for each task, thus we should consider a weight for each quality property when assessing the overall task quality. In our model, we give the user the chance to specify the quality attributes that properly describe a task as well as the quality importance level preference for each task. In the next section, we propose a user application enabling a guided collection of user quality preferences for a Big Data workflow.

As being said, to allow an overall quality assessment for a workflow, we aggregate the quality of all its composed tasks. However, because each task has its own set of quality properties, not all tasks will consider each workflow quality property. To overcome this issue and be able to accurately and comprehensively assess workflow quality, we propose using trust to model each task with its own quality properties and weight preference. Hence, we aggregate the trust as a unified assessment approach among all tasks to reach a workflow trust-based quality assessment. The details of modeling Big Data workflow trust is depicted in Chapter 6.

4.4 User Interface for the Collection of Workflow QoS Requirements

In this section, we describe a web-based application we developed for collecting Big Data workflow QoS preferences from the user and generating a quality specification profile, which will be used as basis for task and workflow quality-based trust assessment as shown in Figure 4.4. This GUI application, collects the quality specification that captures the key requirements a Big data workflow and its composed tasks. Some of the workflow quality requirements are application domain, data type, data operation and data location. Furthermore, for each composed task in the workflow, the application collects the required quality information such as quality dimension, quality attributes and the weight values required for the overall trust score calculation. In addition, output data quality is specified for each task along with the weights preferred by the user. Finally, a complete workflow

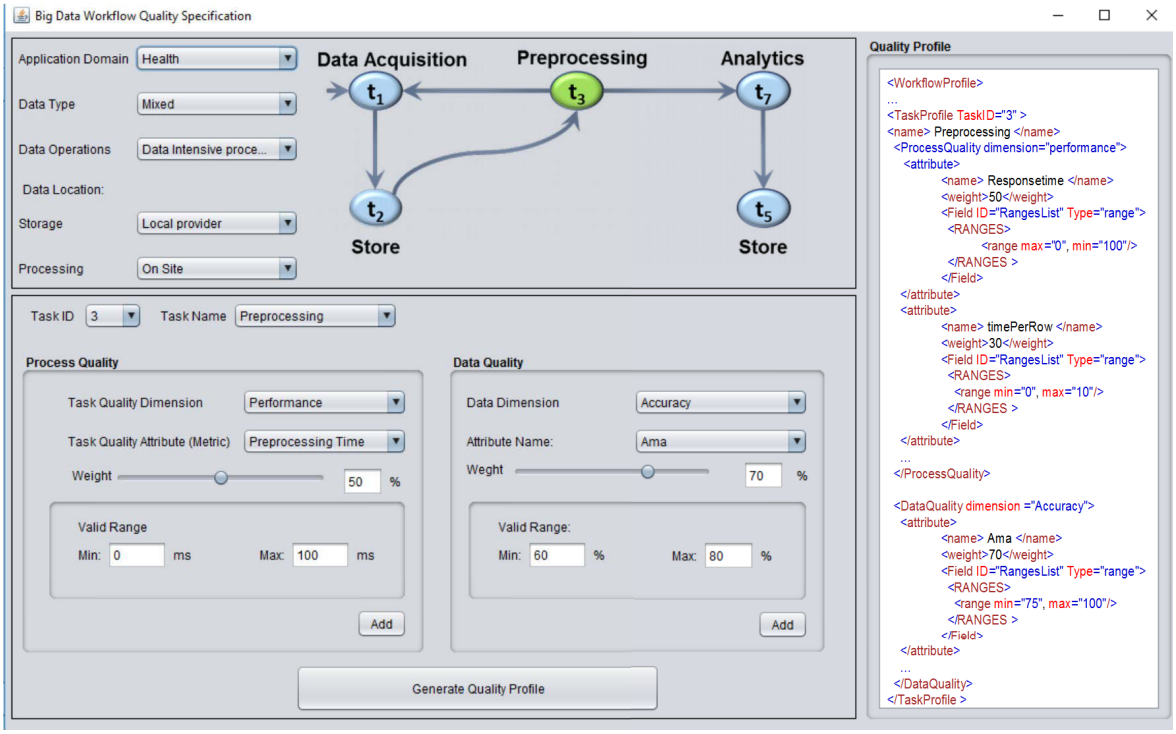

Figure 4.4: User interface for the collection of Big Data QoCS requirements.

quality profile is generated that enumerates the most, and best requirements and specifications that fit each Big Data task (e.g. Big Data Preprocessing task).

4.5 Conclusion

Big Data workflow quality evaluation has become an urgent concern for researchers in both academia and industry. There are very limited initiatives so far that tackle this important aspect in the Big Data research area. Therefore, in this chapter, we addressed end-to-end data quality specifications for Big Data workflows. We offered a multi-dimensional Big Data quality specification model that combined both data-driven and process-driven quality evaluations. The model specified the quality of Big Data, including a description of quality properties, attributes, and the corresponding metrics. In addition, we described the quality metrics related to data processing in Big Data workflow including pre-processing, processing, and analytics, then used the combined data quality specification (data-driven) and tasks quality specification (process-driven) to model the quality of the composed

tasks in the workflow. We further provided a quality specification of successive tasks for the overall workflow quality specification. We finally developed a trust assessment specification model based on the aforementioned multi-dimensional task quality specification.

In summary, based on the quality specification model we proposed, the following are some conclusions that should be highlighted:

- The earlier we address the quality of Big Data, the more we enforce quality in the remaining phases of the Big Data value chain.
- Quality evaluation is a continuous process that involves both data-driven and process-driven quality evaluation.

Providing comprehensive end-to-end quality specifications of Big Data workflows is essential to enable CSP quality evaluation and facilitate CSP ranking to help users in CSP selection decisions. Additionally, maintaining the quality of workflow orchestration at runtime can be determined by specifying the quality at different granularity levels starting at the task and aggregating the quality for the complete workflow. Accordingly, we proposed data and tasks quality specifications, aggregation, and trust assessment specifications for Big Data workflows in addition to cloud providers.

The subsequent chapters detail the CSP selection framework and workflow orchestration quality enforcement framework based on the quality specifications provided in this chapter.

Chapter 5

Towards a Multi-Dimensional Trust Evaluation Architecture for Cloud Service Provider Selection

In this chapter, we propose a multi-dimensional trust model for Big Data workflow over competing clouds. Our model evaluates the trustworthiness of cloud providers based on three levels of trust evaluation using the most up-to-date cloud resource capabilities, the reputation evidence as measured by neighboring users, and a recorded personal history of experiences with the cloud provider. The ultimate goal is to ensure an efficient selection of trustworthiness in a cloud provider who eventually will guarantee high QoCS and fulfills key Big Data requirements

In Section 5.1, we describe the dimensions we use in our trust evaluation model. Initially, we measure the resource capabilities of each cloud by collecting information from the cloud provider. We then collect the personal service history QoCS records followed by the provider's reputation from other users' historical QoCS records. Section 5.2 details the proposed trust model framework along with the description of all its modules. In Section 5.3, we describe the algorithms performed by each dimension of our model, including the resource-based trust, self-based historical records trust, and the community-based trust. Following the sub-trust evaluation algorithms, we describe the final trust score aggregation algorithm. Next, we formalize the model using a MADM technique

and further propose MLR for trust score prediction in Section 5.4. Our experiments and evaluation are depicted in Section 5.5. Finally, we conclude the chapter in Section 5.6.

5.1 Dimensions of Trust Evaluation

Throughout the literature, trust evaluation is based on one or more different quality dimensions. Among these are self-based, reputation-based, provider advertised characteristics, broker-based, SLA-based, and others. For all dimensions, the quality attributes used for trust evaluation are classified based on subjectivity or objectivity. The details of our proposed Big Data quality, and quality-based trust specifications are previously depicted in Chapter 4. To this end, we recommend a three-dimensional trust assessment and prediction model, which includes self-based, reputation-based, and provider-advertised dimensions.

The trust evaluation in our model is based on subjective and objective quality attributes. The objective quality attributes are based on the statistics and measurements recorded through monitoring the personal experience during previous communication with the cloud provider. In our model, we incorporate the subjective quality attributes by collecting the quality reputation information from the neighboring community. In order to reduce the subjectivity when collecting the reputation information, we calculate the trust score locally based on the user preference and not based on neighbors' evaluations.

Some of the quality attributes are qualitative, i.e., they cannot be measured quantitatively. Hence, for these attributes, users can provide a score value from 0 to 1 where 0 is the worst and 1 is the best. In this way, it can be quantified and treated just like the rest of measurable attributes.

5.1.1 Self-based Trust

In the self-based trust evaluation dimension, the trust is evaluated based on history logs recorded locally at the user side. These logs are engendered through monitoring the communication between the user and each cloud provider during past experiences.

The user maintains a database for each cloud provider containing historical performance records

for each transaction. Among the tracked quality attributes are the data size, response time, availability, request status (successful or failed), the distance between the user and cloud provider's data center, cost, and others. Each record is associated with a time stamp of the transaction. The most recent records have more significance than the older ones as they represent the quality of the most recent performance of the cloud provider. Thus, the records are treated as weighted moving average time series, where the newer records have more weight compared to the older ones. Accordingly, a trust score is generated for each cloud provider using the aforementioned logs.

5.1.2 Reputation-based Trust

In reputation-based trust, the trust score is calculated based on performance records collected from neighboring community members. As explained in the previous section, each user keeps the logs of self-experience towards each cloud provider. On one hand, exchanging the entire database between users would cause too much traffic and unnecessary communication overhead. On the other hand, having each user evaluate his own cloud provider trust score does not reflect the perspective of other users having other quality preferences. Accordingly, the reputation trust score is not necessarily appropriate for all users. Hence, in this model we let each community member evaluate trust based on the set of preferences sent by the requesting user to reduce the communication overhead and satisfy the users requirements. Hence, a different trust score will be generated for the same cloud provider according to the requesting user. Likewise, the community member calculates the trust based on a time-weighted average giving higher priority to more recent records as explained in the previous section.

Once the trust score is evaluated for each cloud provider, a list of tuples containing the cloud provider id number and trust score is sent to the requesting user. The user collects these lists from all community members and incorporates this information into the reputation database by averaging the trust scores for each cloud provider.

5.1.3 Provider Advertised Trust

Evaluating trust for cloud providers must not ignore the providers' own perceived performance quality. This dimension of trust is based on the infrastructure and cloud providers' resources properties.

The properties of allocated resources, such as the nodes handling the processing include but are not limited to the price, disk space, delay, free memory size, total processing power, and communication bandwidth. The user defines the priority of each quality attribute according to the nature of the workflow application requirements, some of which require more space, but not high processing power, while others have strict time constraints.

Not only this trust dimension important for determining the physical characteristics of the cloud providers, but it can also compensate other missing dimensions. For example, when the cloud provider is new to the market, or there is a limited number of community members having prior experience with it, then there will be a lack of the self-based and reputation-based information dimensions.

The user sends a request to each of the available cloud providers to obtain their characteristics. Upon receiving the response, the user will evaluate the trust score according to his own preferences giving a different weight to each quality attribute.

The following sections detail the proposed CSP selection framework and the main modules at the user's, neighbors' and the cloud providers' sides.

5.2 Framework and Main Components Description

This section describes our multi-dimensional trust evaluation Framework along with the key components of the user side modules, the CSP modules, and the community members' modules. Figure 5.1 depicts the proposed framework and modules.

5.2.1 User Side Modules

This section describes modules that reside on the user side and are operated and managed by a user with the ultimate goal of determining which cloud service provider fits his/her QoCS preferences.

User QoCS Preference Management: This module is responsible for managing the user's preferences in terms of the QoCS attributes and values that are required by the user and their acceptance levels (e.g. service availability not less than 95%). We developed a GUI to enable the user to input his/her specific requirements, which are collected and sent to the trust module which its role detailed

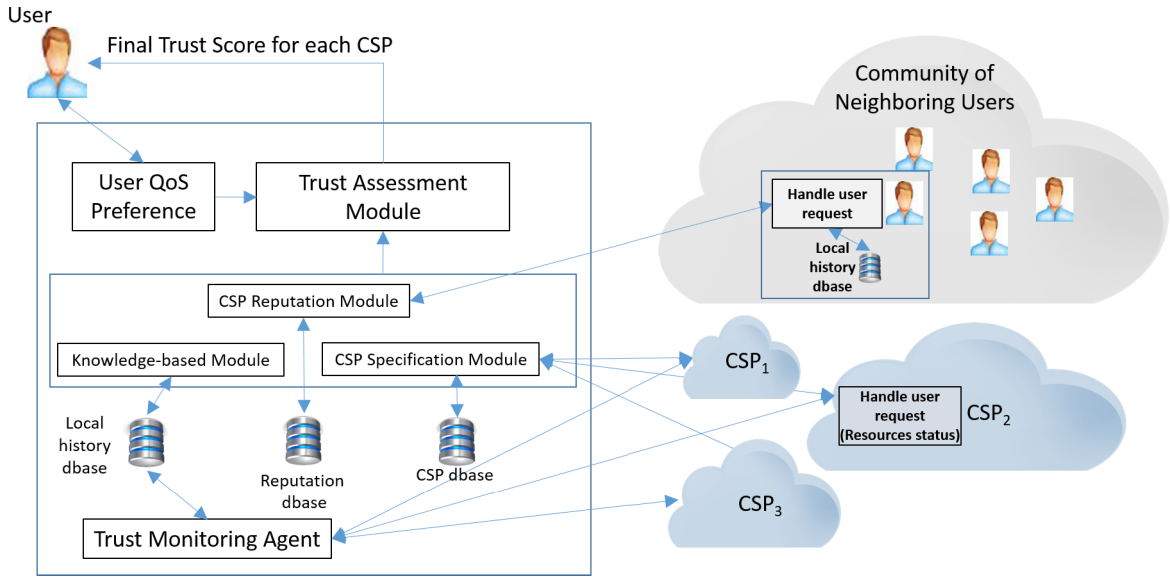

Figure 5.1: Multi-dimensional trust evaluation architecture.

hereafter.

Trust module: This is the core manager module responsible for collecting data from the user QoS preference management module. It also analyzes all databases, *Local History, IndirectReputation, and Cloud_Spec*, to evaluate the trust score for each cloud provider and provides a cloud selection decision to the user using the trust evaluation algorithm explained in Section 5.3. This module produces a trust value for each CSP and provides the user with the decision that yields the highest trust score. In order to reach this decision, the module runs the proposed algorithms on: 1) the *Cloud_Spec* database to generate a *Cloud_Spec* trust score for each cloud, 2) the *Local History* database to generate a direct reputation trust score for each CSP and 3) the *IndirectReputation* database to calculate an indirect reputation trust score for each CP. Subsequently, it applies the weights provided by the user to determine a final trust score for each CSP and finally selects the one with the highest trust score.

Trust Monitoring Module: This module monitors communications with other clouds and collects the cloud's direct reputation information. A record is logged for each communication transaction exchanged between the user and the cloud provider. The log record contains QoCS information that can help to evaluate a cloud's trust score. This information is stored in a local database called the

Local History database. For each cloud, the log information includes multiple transaction logs, each of which contains the start time (invocation) of the transaction, data size, response time, cost and the distance between the user and cloud as well as success status (success or fail).

Local History database: This is the local history database containing the log information for each transaction invoked between the user and each cloud. It includes information about each cloud utilized by the user, including the QoCS attribute values for each service, the time stamp of each task executed, information about the data being exchanged and the distance of the user to the cloud.

CSP Reputation Module: This module is responsible for collecting the cloud's reputation information from neighboring users, i.e., indirect reputation information. It sends an information request message to neighboring users in the community and handles the reply messages received. The request message contains the QoCS attributes to be evaluated and the preferred weight of each attribute. Each reply message contains a list of cloud providers and their corresponding trust scores calculated by the neighboring user according to the original QoCS user preference information parsed from the request message. This module also analyzes all the reply messages received and generates an average trust score for each cloud called the *avgIndirectScore* (more details will be explained in Section 5.3). This generated information is stored in the *IndirectReputation* database and is eventually communicated to the Trust module for the final trust evaluation.

CSP Specification Module:This module is responsible for collecting the quality specifications and characteristics of each cloud, the details of which were explained in Chapter 4. It sends a message to all known CSPs requesting the specifications information including but not limited to the cost information such as the cost per second to use this resource, the cost to use memory of this resource, the cost to use storage of this resource, the cost per bandwidth, available memory, storage space and CPU processing power. Then, this module analyzes the reply messages and stores all the parsed information into the *Cloud_Spec* database.

Knowledge-based Module: This module is responsible for analyzing the data in the *Local History* database and generating a trust score for each CSP called *directCPscore*, which is then communicated to the trust module for final trust evaluation.

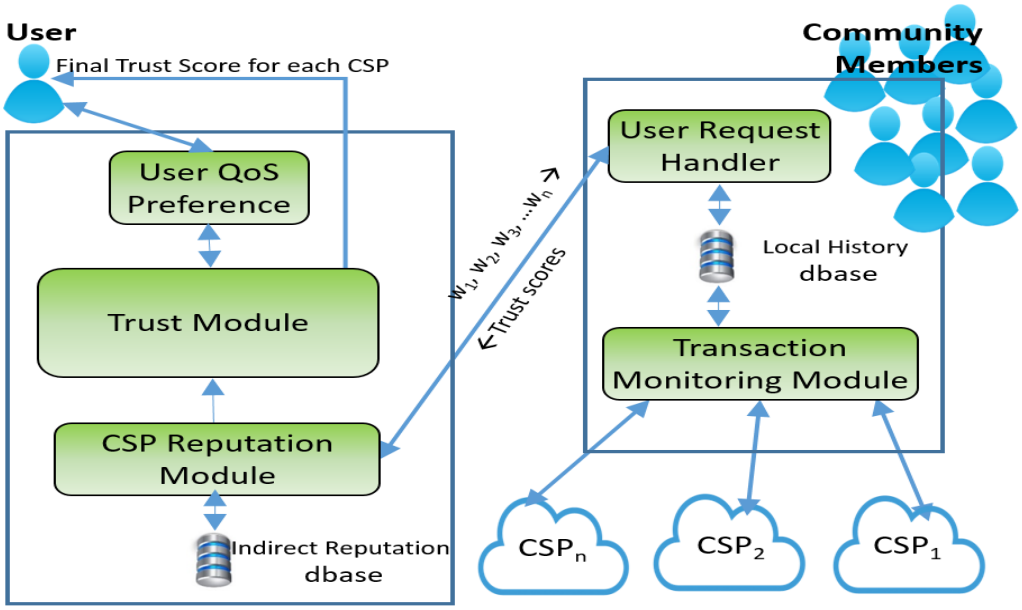

Figure 5.2: Reputation trust framework.

5.2.2 Cloud Service Provider Module

One module is needed on each cloud provider's side to handle resource information request messages and is called the User Request Handler. It generates a reply message containing resource information such as available memory and CPU power. The reply message is received and handled on the users side by the CSP Specification Module.

5.2.3 Neighbors (Community Members) Module

In the current study, we view the reputation from a community perspective, which we will be detailed in this section. We propose a trust model that uses reputation information within a neighborhood community to evaluate trust of cloud service providers as shown in Figure 5.2.

Each community member has two main modules: 1) *Tansaction Monitoring* Module responsible for monitoring the self-transactions with other CSPs and keeping a history log in a database called Local History database, and 2) *User Request Handler* Module responsible for handling the user reputation requests by receiving reputation request messages from other users, generating a trust score for each CSP upon receiving a request message and analysing the Local History database, and

sending reply messages containing trust scores to each requesting user.

This module receives request messages from the user, which contain the QoCS attributes and their weights. Upon receiving a request message, the module analyzes the information stored in the local copy of the Local History database and generates a trust score for each CSP, called the indirectCPscore, which is then populated in a reply message that is sent back to the requesting user, specifically, the CSP Reputation Module. The Local History database consists of a local history log of communication with CPs similar to the user side history log.

The trust score for each CSP is generated by applying one of the MADM methods, which is detailed in Sections 5.3 and 5.4. In this model, the weights, are the QoS preferences sent by the requesting user, whereas the attribute values are the CSP performance in each metric which is stored in the history log. This model will be detailed in the next section.

Community Management: Part of our trust evaluation depends on the CSP's reputation within the community neighborhood. Because we are requesting reputation information from the user's neighbors, we must establish a degree of trust towards the neighbors. However, the neighbors require incentives to provide reputation information, hence, we propose a community management scheme to enforce this requirement. Community is defined in the Oxford dictionary as "the condition of sharing or having certain attitudes and interests in common". From this perspective, the user's community members have used mutual services or interacted with the same set of CSPs. In addition, they have a similar interest in obtaining CSP reputation information from other community members. Community management is discussed in the literature in [225], [226], [227]. The community should be dynamic and adapt to the nature of the cloud environment. To protect the trust score against malicious reputation evaluations, we employ the following rules of engagement:

A third party agent maintains a database of community members' information. To join the community, a user sends a join request, which includes user authentication information, to the third party agent. Upon acceptance, a new community member is given an identification number and an initial reputation score. Community members have the following rights and responsibilities:

(1) Provide honest information.

(2) Other community members are responsible for providing CSP reputation information when requested to do so.

(3) If malicious reputation information is provided by one of community members, a penalty is applied, i.e., reducing the community member's reputation score. This type of behavior is detected if the reputation score provided is considerably higher or lower than the average reputation score of the majority of the community members.

(4) The reputation score is increased slightly each time a member provides CSP reputation information to other members.

(5) It is difficult for a member to regain a favorable reputation score caused by a false accusation. A false accusation causes the reputation score to decrease dramatically, whereas any increase in the score is gradual.

(6) Members with low reputation scores do not receive reputation information from other members until they raise their reputation scores.

5.3 Trust Evaluation Algorithms

In this section, we present our formal trust model for processing Big Data over a cloud platform. We formalize a Big Data service evaluation using the cloud's resource capabilities, its reputation among neighboring users and personal history of user experience. Figure 5.1 above describes the proposed framework.

Initially, we measure the resource capabilities of each cloud by collecting information from the cloud provider. We then collect the personal service history QoCS records followed by the provider's reputation from other users' historical QoCS records. The following sub-section details the three levels of our trust evaluation scheme.

5.3.1 Cloud Resource-based Trust

The first stage of our trust evaluation scheme involves collecting the current resource characteristics from the potential cloud providers, i.e., memory, processing power, cost and the distance of the user from the provider. In addition, the user enters his/her personal preferences regarding the required QoCS. A weight is then assigned to each QoCS attribute according to the personal preference information. Next, a partial score is calculated for each cloud provider, which is called *CPcharacteristics* score. The pseudo code shown in Algorithm 1 describes the proposed algorithm for collecting cloud service provider characteristics. The attributes used to calculate the trust for these CSP characteristics include:

(1) Memory size

(2) Processing power

(3) Response time

(4) Data center parameters (processing speed, failure rate and bandwidth)

(5) Cost: It is generally known as a fixed attribute published by cloud providers and most of trust models do not incorporate the component cost in the trust assessment. However, we decide to include the cost in the trust assessment to allow some flexibility in renegotiating the cloud service price, which can be initiated either by the user or the cloud provider, thus, might affect the overall trust assessment.

5.3.2 Local Service History-based Trust

The second stage of our trust evaluation scheme involves calculating the history-based trust score. First, the user saves his/her service history records with each cloud provider in a database. We then calculate the cloud service providers' trust scores for each Quality of Service attribute. This is shown in Algorithm 2. The two factors that can affect the history-based trust score are: 1) the number of times the user has interacted with the cloud provider, and 2) the timeliness of the service history records expressed in terms of how recently they were recorded. The user experience factor

Algorithm 1 Resource quality-driven CSP trust score calculation algorithm

Input:
$CSPList$: List ofCSPs,
$attributesList$: List of QoS attributes of Cloud resources,
$weights$: Weight of each QoS attribute
Output: Trust Score updated for each CSP
procedure EVALUATECPCHARACTERISTICS($CSPList$, $attributesList$, $weights$)
    for all $csp \in CSPList$ **do**
        $sendRequest(csp, attributeList)$
    end for
    while $true$ **do**
        $recvResponse(csp, attributeVal)$
        $score \leftarrow 0$
        for $i \leftarrow 1, n$ **do**
            $score \leftarrow score + attributeVal[i] \times weights[i]$
        end for
        $update(csp) \leftarrow score$
    end while
end procedure

is represented as a weighted score, the experience weight value for each CSP, which is calculated as follows:

$$E_{i,j} = 1 - e^{(-0.5 \times N_{i,j})} \tag{7}$$

where $E_{i,j}$ is the experience of user *i* with the cloud service provider CSP*j*, and $N_{i,j}$ is the number of history records the user *i* stored on cloud service provider *j*, which may be outdated. Hence, we incorporate a time factor to calibrate the final score of each cloud provider using the following equation:

$$TF_{i,j} = 1 - e^{(-0.5 \times 1/\Delta t_{i,j})} \tag{8}$$

$TF_{i,j}$ is called the time fade factor with respect to user *i* to CSP_j, and Δt is the difference between the current and last interaction time between user *i* and CSP*j*.

The details of how we calculate a score for each cloud provider from their history of previous direct interactions with the user is shown in Algorithm 2. This score aggregates the scores of each QoCS attribute.

We calculate a final score using the user's personal preference weights for each QoCS attribute. We also assign higher weights to recent records over older ones using a weighted moving average algorithm.

Algorithm 2 Self-historical interaction-based CSP trust scores algorithm

Input:
$CSPList$: List of CSPs,
$CSPServiceLog$: Service Log of all CSPs,
$attributesList$: List of QoS attributes,
WA: Weight of each QoS attribute,
WT: Weight of each time slot
Output: $CSPListScore$: Trust Score for each CSP
procedure EVALUATECPSERVICE($CSPList$, $attributesList$, WA, WT)
    for all $csp \in CSPList$ **do**
        for $attLabel \leftarrow 1, nAttributes$ **do** //in attributeList
            $attScore \leftarrow 0$
            for all $timeSlot \in CSPServiceLog$ **do**
                $attScore \leftarrow attScore + attributeVal[timeSlot] \times WT[timeSlot]$
            end for
            $CSPListScore[csp][attlabel]$ += $attScore \times WA[attlabel]$
        end for
    end for
    return $CSPListScore$
end procedure

The CSP score is then calculated for each CP, followed by the final *Max CPscore*. We use the following equation to calculate the *directCPscore* for each CP_i:

$$directCPscore_i = E_i \times TF_i \times CPscore_i \tag{9}$$

5.3.3 Community-driven Reputation-based Trust

The third stage of our trust evaluation scheme addresses reputation-based trust. Several researchers have extensively studied user trust. In the current study, we view reputation from a community perspective, which we detail in this section.

Algorithm 3 Indirect trust score (reputation) algorithm performed by neighbors

procedure EVALUATECPSERVICEBYNEIGHBOR()
    while $true$ **do**
        $recvRequest(src, cspList, attributesList, WA, WT)$
        $cspListScore \leftarrow EvaluateCPService(cspList, attributesList, WA, WT)$
        $sendReply(src, cspListScore) \leftarrow score$
    end while
end procedure

Reputation based trust evaluation: The user requests the CSP scores from all the neighbors in the community according to: 1) the QoCS attributes chosen by the user and 2) the service history of

the neighbors. To encourage users to provide rating information, incentives should be provided to the neighbors [41], such as receiving CSP reputation scores from the community to aid in their own decision-making.

The reputation request message issued by the user contains a list of selected QoCS attributes and their user-assigned weights. Upon receiving a request from a user, the neighboring users perform the following steps as detailed in Algorithm 3:

(1) Calculate the experience weight value for each CSP (E_i).

(2) Calculate the time fade weight value for each CSP (TF_i).

(3) Calculate the CSP score for each CSP using the MADM method ($CPscore_i$) with the weights embedded in the request message.

(4) Return a reputation reply message containing a list of final CSP scores to the local user who originated the reputation request message.

The local user receiving the reputation reply message parses it to extract the scores for each CP, as shown in Algorithm 4. After all the replies are received by the user, the average score of each CSP among the *n* users who replied can be calculated with:

$$avgIndirectScore = (\sum_{i=1}^{n} CPscore_i)/n \tag{10}$$

5.3.4 Final Trust Score Calculation Conclusion

After collecting and compiling all the trust scores described in the previous cycle, i.e., the CSP resource-based trust score, the local history-based trust score and the reputation-based trust score, a final trust score is calculated for each CSP, and the max score becomes the selected CSP. The final score is calculated using the Simple Weighted Average (SWA) method with user-assigned weights [228]. The SWA is used to compute the average of a group of numbers with asymmetrical importance. The following formula depicts the calculation of the final trust score:

$$FinalScore = ds \times directScore + is \times avgIndirectScore + cs \times CPcharacteristic \tag{11}$$

where *ds*, *is*, and cs are weights given by the user, and

$$ds + is + cs = 1 \tag{12}$$

Algorithm 4 Indirect trust score (reputation) algorithm performed by neighbors

Input:
$neighborList$: List of neighboring users,
$cspList$: List of CSPs,
$attributesList$: List of QoS attributes,
WA: Weight of each QoS attribute,
WT: Weight of each time slot
Output: $localCSPScoreList$
procedure EVALUATEINDIRECTTRUST($neighborList, cspList, attributesList, WA, WT$)
    for all $nc \in neighborList$ **do**
        $sendRequest(nc, cspList, attributesList, WA, WT)$
    end for
    while $true$ **do** //with timeout limit
        $recvResponse(nc, cspListScore)$ //each record (csp, score)
        // update the local CSPListScore with the neighbor score
        for all $csp \in cspListScore$ **do**
            localCSPScoreList[csp].totalScore += cspListScore[csp]
            localCSPScoreList[csp].nReplies += 1
            localCSPScoreList[csp].score = totalScore/nReplies
        end for
    end while
    return $localCSPScoreList$
end procedure

5.4 Cloud Service Provider Selection Trust Model Formalization

In this section, we describe our formal trust evaluation model. We first formulate the cloud selection model as a MADM. In addition, we propose using a MLR method for trust score prediction, as detailed in the following sections.

5.4.1 Different MADM Techniques for Selecting a Cloud Provider

For each CP, we evaluate the trust score by formulating our model as a MADM. We propose three different multi-attribute scoring methods, which are the Simple Additive Weighting (SAW) method, the Weighted Product Method (WPM), and the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS). These methods follow a similar version of the Simple Multi-Attribute

Rating Technique (SMART) described in [229], which follow these steps:

Step 1: Determine the goal that is selecting the most suitable cloud service provider for Big Data processing.

Step 2: Identify the alternatives for evaluation that are the available cloud service providers in the market.

Step 3: Determine the attributes used as the basis for evaluating the alternatives. For example, throughput, reliability, and resource quality.

Step 4: Choose the weights for each attribute. In other words, determine the importance of each attribute.

Step 5: Evaluate the score of each alternative using one of the three MADM scoring methods.

Step 6: Analyze the results and decide on the best alternative.

We follow the SMART technique because it is considered the most common method actually used in real, decision-guiding multi-attribute utility measurements [230]. The SMART model does not depend on the alternatives and is not affected if more alternatives are added. In the following subsections, we describe the details of each method:

5.4.1.1 Simple Additive Weighting Model (SAW)

This method is also named a weighted sum model (WSM)) and is the most straightforward and most widely used methods [231]. The rank, or trust score, is calculated using SAW with weights assigned for each alternative. The values of the alternatives ranks are used to choose the best alternative [232] [233].

We formulate the score for each cloud provider as a MADM problem [234] wherein the model is expressed in the following steps:

Step 1: Model construction and initialization with

$$CP = \{cp_i \mid i = 1, 2, 3, \ldots n\} \tag{13}$$

$$A = \{a_j \mid j = 1, 2, 3, \ldots m\} \tag{14}$$

$$W = \{w_1,\ w_2,\ w_3 ... \ w_m\} \tag{15}$$

$$X = \begin{bmatrix} x_{11} & \cdots & x_{n1} \\ \vdots & \ddots & \vdots \\ x_{1m} & \cdots & x_{\mathrm{nm}} \end{bmatrix} \tag{16}$$

where cp_1, $cp_2 \ldots$ cp_n are the possible n alternative cloud service providers available to the user, a_1, a_2,..., a_m represent QoCS attributes (criteria), for example, response time, availability, and reliability. w_j is the weight (significance) of the j^{th} attribute and x_{ij} is the performance rating of the i^{th} alternative (CP) with respect to the j^{th} attribute. In this model, a higher score is assigned to the cloud provider with the highest performance rating, which preferably maximizes the j^{th} attribute.

Step 2: Construct the normalized decision matrix. This step allows for comparing attribute values with different scale units. The values here are normalized on a scale from 0 to 1. Some attributes, such as reliability, have preferably high values, whereas others, such as cost, have preferably low values. Thus, to normalize these values easily and fairly, we use Eq. 17 when a high value is preferred (beneficial attribute) and Eq. 18 when a low value is preferred (non-beneficial attribute).

$$\mathrm{r_{ij}} = \mathrm{x_{ij}} / \mathrm{x_{ij}^{max}}\ (row) \tag{17}$$

Or

$$\mathrm{r_{ij}} = \mathrm{x_{ij}^{min}} / \mathrm{x_{ij}}\ (row) \tag{18}$$

Step 3: Construct the weighted normalized decision matrix. We assign a different weight value for each attribute to give different preference of an attributes over other attributes. The user selects the weights based on QoS preferences and the type of Big Data application. We use the following equation to calculate the values of the weighted normalized decision matrix:

$$v_{ij} = w_j \times\ r_{ij}, s.t. \sum_{\mathrm{j=1}}^{\mathrm{m}} \mathrm{w_j} = 1 \tag{19}$$

Step 4: Calculate the score of each alternative (CP).

$$score_i = \sum_{\mathrm{j}=1}^{\mathrm{m}} v_{ij}, i = 1, 2, 3, .., n \tag{20}$$

Step 5: Select the best alternative (CP):

$$CPbestscore = \max_{0 \leq i \leq n} score_i \tag{21}$$

5.4.1.2 Weighted Product Method (WPM)

Another approach, is to use the weighted product method as a scoring technique. This method was first introduced by Bridgeman [235], which is a simple method that is not widely used despite its sound logic [233]. It is introduced as a modification to the SAW method [231], but does not require normalization because the attributes are multiplied to each other and raised to the weights as an exponent. For beneficial attributes, i.e., the attributes that are better with higher values, the weight exponent should be positive. Negative weights are given for non-beneficial attributes [233]. The score is then calculated using the following formula [236]:

$$score\left(cp_j\right) = \frac{\prod_{i=1}^{m} x_{ij}^{w_i}}{\prod_{i=1}^{m} x_i^{'w_i}} \tag{22}$$

where the $x_i^{'}$ value is the highest score of the attribute i among all alternatives CPs. Using $x^{'}$ allows to limit the resultant score value to be between 0 and 1 instead of using numbers that are greater than 1 because of the exponent property.

5.4.1.3 Technique for Order Preference by Similarity to Ideal Solution (TOPSIS)

The third scoring approach used in this work is the TOPSIS technique proposed by Hwang and Yoon [233]. It is based on the idea of choosing the alternative with the shortest distance from the positive-ideal solution and the longest distance from the negative-ideal solution. Here, the ideal solution is the assembled ideal scores of all attributes. The following are the steps required for this method:

Step 1 and **Step 2** are the same as in the SAW model.

Step 3: Identify Positive-Ideal and Negative-Ideal Solutions. The Positive-Ideal solution is the highest performance value for attribute *i* among all alternatives and is represented as follows:

$$X^* = \left\{ x_j^* \mid j = 1, 2, 3, \ldots m \right\} \tag{23}$$

The Negative-Ideal solution is the lowest performance values for attribute *i* among all alternatives and is denoted as follows:

$$X^- = \left\{ x_j^- \;\middle|\; j = 1, 2, 3, \ldots m \right\} \tag{24}$$

Step 4: Calculate Separation Measures by calculating the distance of each alternative cp_i from the positive-ideal solution *X** using the n-dimensional Euclidean distance:

$$D_i^* = \sqrt{\sum_{j=1}^{m} \left(x_{ij} - x_j^* \right)^2} \tag{25}$$

where *i* is the alternative index, and *j* is the attributes index. Also, the separation from the negative-ideal solution X^- is given by:

$$D_i^- = \sqrt{\sum_{j=1}^{m} \left(x_{ij} - x_j^- \right)^2} \tag{26}$$

Step 5: Calculate Similarity Indexes. We calculate the similarity index for alternative cp_i using:

$$score_i = \frac{D_i^-}{D_i^* - D_i^-} \tag{27}$$

where the 0 <= $score_i$ <= 1. The cp_i with the highest score is selected to process Big Data.

5.4.2 Formalizing the QoS Performance Scores Using Historical Records

Here, the x_{ij} in the matrix Eq. 16 are the CSP partial scores for each quality attribute calculated from the historical records. Following the same model used in a SAW in MADM problem, these partial scores are calculated following these steps:

Step 1: Model construction for calculating the x_{ij} in the above matrix Eq. 16:

$$A = \{a_j \mid j = 1, 2, 3, \ldots m\} \tag{28}$$

$$T = \{t_z \mid z = 1, 2, 3, \ldots nt\} \tag{29}$$

$$WT = \{wt_z \mid z = 1, 2, 3, \ldots nt\} \tag{30}$$

$$Y = \begin{bmatrix} y_{11} & \cdots & y_{\mathrm{m1}} \\ \vdots & \ddots & \vdots \\ y_{1\mathrm{nt}} & \cdots & \mathrm{y_{mnt}} \end{bmatrix} \tag{31}$$

where a_1, a_2 … a_m, are the m selected QoCS attributes, t_1, t_2, ..., t_{nt} are the different times at which the attributes were measured and nt is the number of time slots. We assume that $t_z < t_{z+1}$, for all $\{ z = 1, 2, 3 \ldots nt\}$. Y_{jz} are the performance rating values of attribute j at time z.

Step 2: Construct the normalized decision matrix as explained previously. For simplicity, we only describe the beneficial attribute as:

$$\mathrm{rx_{jz}} = \mathrm{y_{jz}} / \mathrm{y_{jz}^{max}}(col) \tag{32}$$

Step 3: Construct the weighted normalized decision matrix wherein higher weight is given to relatively recent attribute values, and lower weight is given to older values. A higher value of z gives a higher weight.

$$vx_{jz} = wtz \times rx_{jz}, \sum_{\mathrm{z=1}}^{\mathrm{nt}} \mathrm{wt_z} = 1 \tag{33}$$

Step 4: Calculate the score of each alternative (attribute).

$$x_j = \sum_{\mathrm{z=1}}^{\mathrm{nt}} vx_{\mathrm{jz}}, z = 1, 2, 3, \ldots, nt \tag{34}$$

5.4.3 Trust Score Prediction Formalization using MLR

In this section, we describe the trust evaluation problem in competing cloud environment as follows: a user wants to select a CSP to execute some Big Data processing task. Given a history of previous service interactions received from members of the community, the user will predict whether CSP_i is trustworthy or not. We define a trustworthy CSP as being able to satisfy a set of QoCSs. The goal is to reach a high prediction accuracy.

For each service interaction with CP_i at time *t*, a record containing the observed quality level of this service y_k^t by user *k* with respect to a set of quality attributes y_{ki}^t that is a real value [0,1]; such that:

$$CP = \{cp_i \mid i = 1, 2, 3, \ldots n\} \tag{35}$$

$$A = \{a_j \mid j = 1, 2, 3, \ldots m\} \tag{36}$$

$$P^t = \{p_1,\ p_2,\ p_3 \ldots\ p_m\} \tag{37}$$

where *t* is the time stamp of the observed service transaction, cp_1, cp_2 ... cp_n are the possible *n* alternative cloud service providers CPs available to the user *k*, and a_1, a_2,..., a_m represent QoCS attributes (criteria) such as reliability, availability, and throughput. p_1, p_2,..., p_m represent the performance level of a_1, a_2,..., a_m respectively.

Then, *trust* is the score that CP_i will achieve according to the set of QoCS at time *t* described by the p^t vector.

Let $y_i^t = y_{ki}^t \cup \{y_{ui}^t,\ k \neq u\}$ where y_{ui}^t is an observation of neighbor *u* about a prior service experience with CP_i provided to user *k*. The observation record is in the form of $\{P^t, y^t\}$ specifying the performance of each quality attribute at time *t*.

Let $y_i = \{y_i^t,\ \mathrm{t} = 1, ...,\ \mathrm{N}\}$ represent the set of observations gathered by a user *k* which includes both self-experience and collected observations from neighbors in [0, N]. And, let $\boldsymbol{p} = \{\mathrm{P}^t,\ \mathrm{t} = 1, ...,\ \mathrm{N}\}$ be the corresponding performance level of the quality attributes in [0, N].

We suggest using MLR to solve this problem and model the relationship between the trust score which we consider the dependent variable y and some explanatory (also named independent) variables *p* using a linear function of the independent variables [237].

$$E\left[y_i^t \mid p_i\right] = \beta_0 + \sum_{i=1}^{m} \beta_i P_i + \varepsilon \tag{38}$$

where βi = [β_i, $i = 1, 2, 3, \ldots m$] is a column vector of coefficients that are estimated values from the available data, and ε is the 'noise' which is a random variable having an independent normal distribution with mean equals to zero and unknown constant standard deviation σ.

We estimate the values for the β_i coefficients by minimizing the sum of squares of differences between the predicted values and the observed values in the data given by:

$$\sum_{i=1}^{N} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_m x_{im})^2 \tag{39}$$

Let the Ordinary Least Squares (OLS) $\widehat{\beta}_0, \widehat{\beta}_1, \ldots, \widehat{\beta}_m$ be the optimized coefficients that minimize Eq. 39. Then, we substitute the computed values in the linear regression model in Eq. 38 to predict the trust score for one CSP according to the following:

$$\hat{y} = \widehat{\beta}_0 + \sum_{i=1}^{m} \widehat{\beta}_i P_i \tag{40}$$

To summarize, historical experience {p, y_i} is a collection of self-experience QoS performance of CP_i and reputation provided by neighbors upon their experience dealing with CP_i. We perform the multiple linear regression processing for each CSP calculating the expected $\hat{y}$. The selected CSP would be the one with the highest $\hat{y}_i$ value, i.e., the one with highest predicted trust score, which means the highest probability of providing satisfactory QoS performance. The pseudocode shown in Algorithm 5 describes the CSP selection process according to trust score prediction using an MLR algorithm. A trust score is predicted for each CP_i. The algorithm then recommends a CP_i having the highest score.

Algorithm 5 MLR algorithm for CSP) selection

Input:
$CSPList$: List of CSPs,
$CSPServiceLog$: Service Log of all CSPs,
$ReqAttrVals$: List of Required QoS attributes
Output: CSP with Highest Predicted Trust Score
procedure PREDICTCPTRUST($CSPList$, $CSPServiceLog$,$ReqAttrVals$)
    for all $csp \in CSPList$ **do**
        $attScore \leftarrow 0$
        Evaluate B's coefficients according to Eq. 39 and Eq. 40
        for $attLabel \leftarrow 1, nAttributes$ **do** //in ReqAttrVals
            $attScore \leftarrow attScore + ReqAttrVals$[attLabel] $\times$ B[attLabel]
        end for
        $CSPListScore[csp] \leftarrow attScore$
    end for
    return max($CSPListScore$)
end procedure

5.5 Experimental Results and Discussions

In this section, we present both a formal evaluation and simulation experiments as follows: 1) we conduct a formal evaluation of our algorithm's complexity, 2) we formally evaluate the communication overhead generated from the execution of our multi-dimensional reputation scheme and 3) we simulate the environment of cloud selection-based reputation models and conduct various experiments to validate our trust model.

5.5.1 Algorithm Performance and Complexity Evaluation

5.5.1.1 Algorithm Time Complexity Evaluation

It is worthwhile to evaluate the time complexity of each of our proposed algorithms as it measures the algorithmic efficiency, which has an impact on execution time. To evaluate Algorithm 1, the execution time depends on the parameter *N*, the number of CSPs, and parameter *K,* the number of QoCS attributes. Therefore, the time complexity is on the order of O (N.K). Though, Algorithm 2 depends on the number of CSPs (*N*), the number of QoCS attributes (*K*) and the number of time slot records stored (*L*). Accordingly, the time complexity of this algorithm is on the order of O (N.K.L) because there is no relationship between N, L, and K. Moreover, in Algorithm 3, the time complexity depends solely on the number of requests *R* received by neighboring users in the

community, so, the time complexity is on the order of O (R). Similarly, for a Algorithm 4, the time complexity depends on the number of neighbors who replied to the requests, the maximum number of community members (M) and the number of CSPs (N). Hence, the time complexity is on the order of O (N.M). Overall, all proposed algorithms exhibit high efficiency and low execution time, which will not affect the performance of our multi-dimension trust-based Cloud selection approach.

5.5.1.2 Communication Overhead Evaluation

We evaluate the communication overhead for each evaluation strategy as the number of messages exchanged for the purpose of trust evaluation. Our comprehensive trust model requires messages to be exchanged between both 1) the customer and cloud providers, as well as 2) the customer and neighboring users in the community. Using this strategy, we have two partial trust score evaluations requiring message exchanges. For the cloud resource-based trust score evaluation, the client must send a QoCS information request message to each cloud provider and receive a response message that includes the requested information. For the reputation-based trust score evaluation, a request message is sent to the neighboring users, and a response message is sent back carrying the required trust score values for each CSP. The total number of messages can be calculated using the following formula:

$$total\ number\ Msgs = \ 2 \times \ nCSPs + 2 \times \ nNeighbors \tag{41}$$

Evaluating the size of each message is performed using the following calculations. First, for the cloud resource-based trust evaluation, the communication is expressed by the amount of data being transmitted between the CSP Specification module and the set of cloud providers. We measure the total size of the exchanged messages using the following formulas:

$$sizeReqMsg = \ nQoSAttr \times sizeQosAttrName \tag{42}$$

The size of an attribute name is one byte (assuming that the maximum number of QoCS attributes is 2 to the power of 8, or 265 attributes).

$$sizeRspMsg = \ nQoSAttr \times (sizeQosAttrName + sizeQosAttrPerformanceVal) \tag{43}$$

The size of an attribute value is 4 bytes, which is a double number.

$$sizeAllMsgs = nCSPs \times (sizeReqMsg + sizeRspMsg) \tag{44}$$

Figure 5.3 depicts the communication overhead generated in Kbytes per number of QoCS properties used in the trust evaluation. Second, for reputation-based trust evaluation, we evaluate the communication overhead by calculating the amount of data exchanged between the customer and neighboring users in the community, i.e., between the CSP reputation module hosted at the customer side and each CSP reputation module hosted at each neighbor. The following formulas express this strategy:

$$sizeReqMsg = nQoSAttr \times (sizeQosAttrName + sizeAttrWeight) \tag{45}$$

$$sizeRspMsg = nCSPs \times trustScorePerCSP \tag{46}$$

$$sizeAllMsgs = nNeighbrs \times (sizeReqMsg + sizeRspMsg) \tag{47}$$

Figure 5.3 shows that communication overhead is proportional to both the number of neighboring users in the community and the number of selected QoCS attributes. With 20 cloud providers, 100 active community members and 15 selected QoCS attributes, the calculated overall communication overhead was nearly negligible (15 Kbytes). This proves that our trust model is lightweight because it does not incur a heavy load in providing cloud providers trust scores prior to the selection of the best CSP and guarantees optimal adherence to QoCS user requirements without affecting the performance of the overall solution.

5.5.2 Experimentation

In this section, we describe the experiments we conducted to evaluate our proposed trust model framework. We explain the experimental setup and describe the simulator system including all

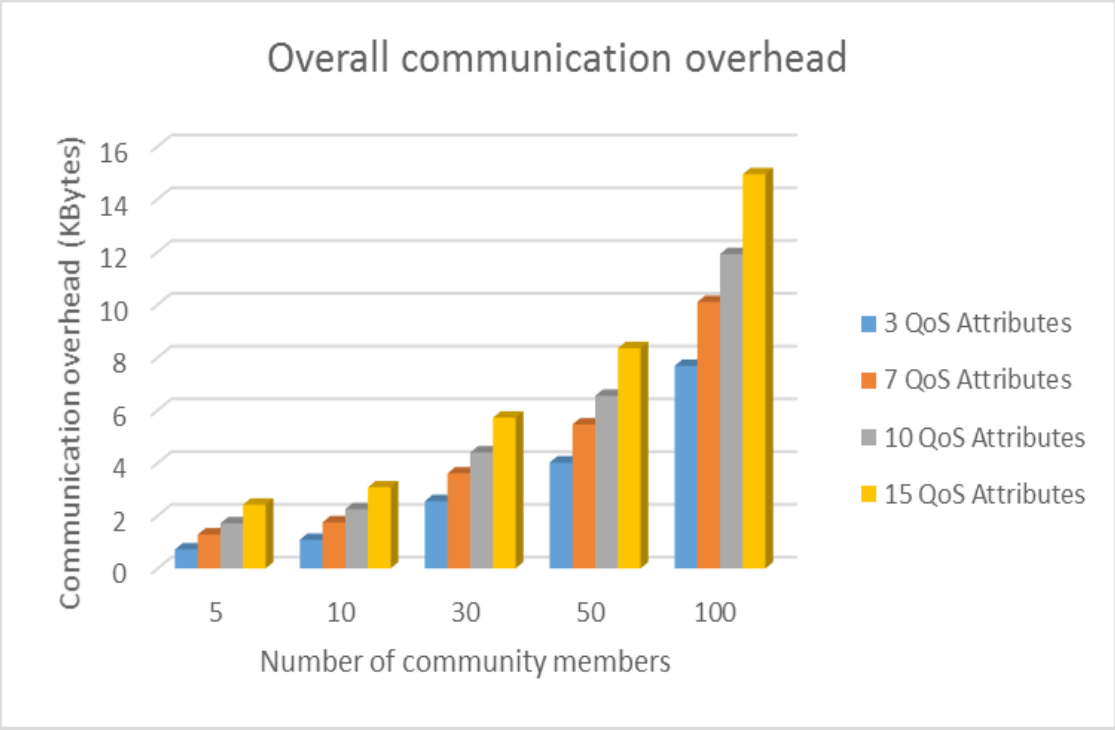

Figure 5.3: Overall communication overhead.

modules. We then propose to extend CloudSim to consider multi-clouds and implement properties of our trust model. After that, we explain the scenarios of our experiments along with results interpretations. Finally, we provide a discussion of our results.

5.5.2.1 Environmental Setup

We considered the following default simulation parameters:

(1) Number of clouds: 1 to 50 clouds

(2) Number of nodes within each cloud: 1 to 100 nodes.

(3) QoCS specification file: budget, availability, Big Data application type, file size and priority level.

(4) Cloud properties: proximity, average node performance, and unit storage price.

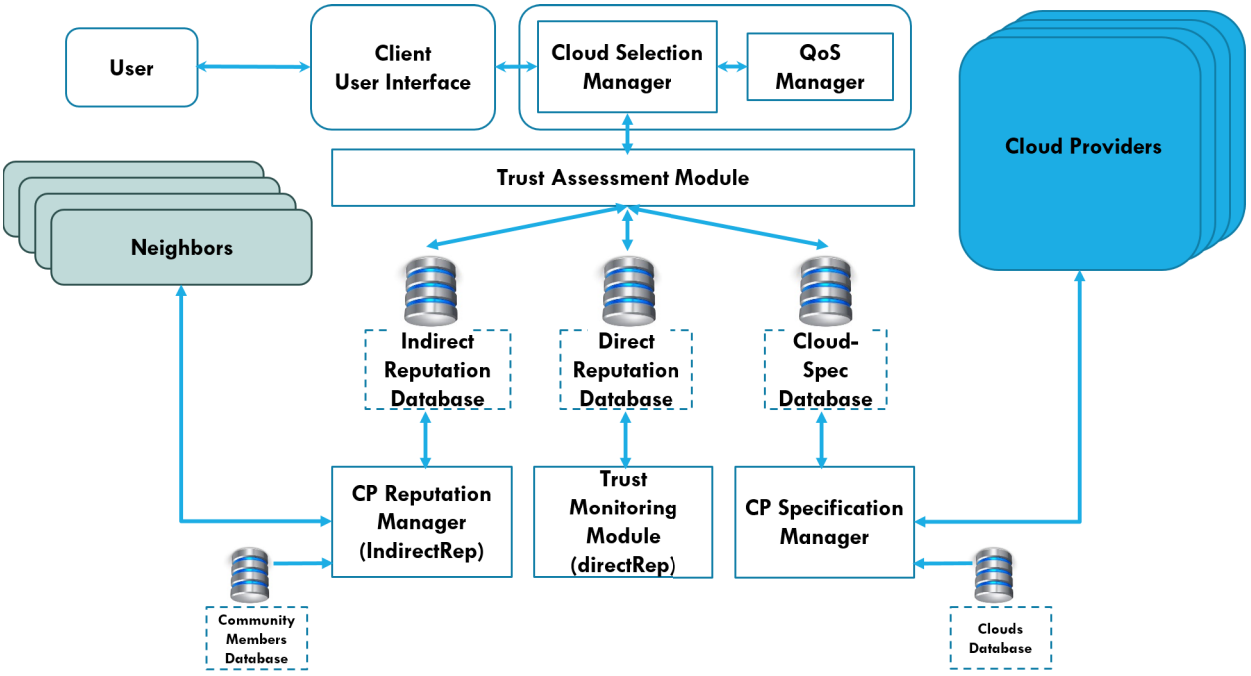

Figure 5.4: Simulation system components.

(5) Node properties: available resources, memory, disk space, processing power, round trip delay (RT) and bandwidth.

(6) QoCS attributes: data size, distance, cost, response time, availability, and confidence.

(7) Number of community members: 3 to 100 neighbors.

(8) Reputation database: 20 timely historical records for each CSP local to each community member.

5.5.2.2 Simulator System Description and Modules

We developed a simulator in Java to test our proposed trust model to implement all four of the trust evaluation algorithms described in Section 5.3. Figure 5.4 describes the main components of our simulation, including the client user interface, QoCS Manager, cloud selection manager and cloud providers, as well as neighbor components (e.g., other users).

The following evaluation is intended to prove the applicability of our proposed trust model for processing Big Data over competing clouds by: 1) evaluating whether the Big Data application

QoCS is significantly considered when selecting the appropriate cloud from the existing clouds, 2) showing the effect of each strategy that our CWTS evaluation depends on while varying the weight of each strategy and monitoring the selection results, 3) evaluating whether our trust model can detect malicious trust ratings and react accordingly, 4) evaluating the effect of deleting one of the strategies that compose our trust evaluation model, and 5) evaluating the scalability of our model towards supporting a large number of selection requests. Before we detail the evaluation scheme, we first describe our simulation implementation and component structure, our set of scenarios and the results that were obtained from executing the experiments. In some of the experiments, we used CloudSim to generate data to populate the cloud reputation databases, including direct and indirect reputation information. We also used the cloud characteristics data of CloudSim to populate the cloud spec database.

Each component is described as follows:

(1) CSP Reputation Manager: implemented according to the description in the framework section. It is responsible for collecting the clouds' reputation information from neighboring users.

(2) Clouds database: the database that keeps track of the candidate cloud service providers.

(3) Neighbor users' community database: the database that keeps track of the neighbors of each community member.

(4) Cloud Provider module: this component simulates a cloud provider. It uses static log files collected by multiple runs of CloudSim, each with different cloud configurations and multiple users. We collect the log information to populate our databases. CloudSim provides a generalized and extensible simulation framework that enables modeling, simulation and experimentation with emerging Cloud Computing infrastructures and application services, allowing its users to focus on specific system design issues without handling the low-level details related to cloud-based infrastructures and services [24].

(5) Neighbor Component: simulates a neighboring user that uses the log generated by CloudSim

runs to populate the *Local History* database. Each neighbor object is responsible for responding to reputation requests from other neighboring users by analyzing its own *Local History* database.

(6) Indirect Reputation database, Direct Reputation (Local History) database, and Cloud Spec database: databases containing the QoCS information explained earlier in the framework section.

5.5.2.3 CloudSim Extension

CloudSim was developed to simulate a cloud environment and evaluate different cloud properties using features of the simulator. It has the capability of simulating large-scale Cloud Computing infrastructure with multiple data centers on one physical computing machine. The user can describe details of data centers, such as virtual machines, applications, users, computational resources, and policies [238]. However, the existing CloudSim simulation environment does not support multiple cloud simulation. We built a CloudSim extension to simulate multiple clouds and provide an application layer to run more extensive experiments with our full framework. Figure 5.5 depicts the main modules and components developed to extend the CloudSim framework.

The extension includes proxies that connect the multiple cloud instances to the community members and the user. The user and all community members log information about each transaction occurrence between them and each cloud instance in a database called *Local History*. The transaction takes the form of Big Data processing, distribution, or analysis requests. When the user decides to start the process of trust score evaluation, it first sends *ResourceCharacteristicsRequest* message to collect each cloud resource capability and log this information into a database called *CloudSpec*. The user also sends a *ReputationRequest* to each of the community members and collects their responses in the *IndirectReputation* database. Simulating this complete scenario allows accurate measurements of the communication overhead, and trust score evaluation overhead. In addition, it allows the evaluation of post cloud selection decision-making by simulating processing requests to the selected cloud and measure the QoCS after the responses.

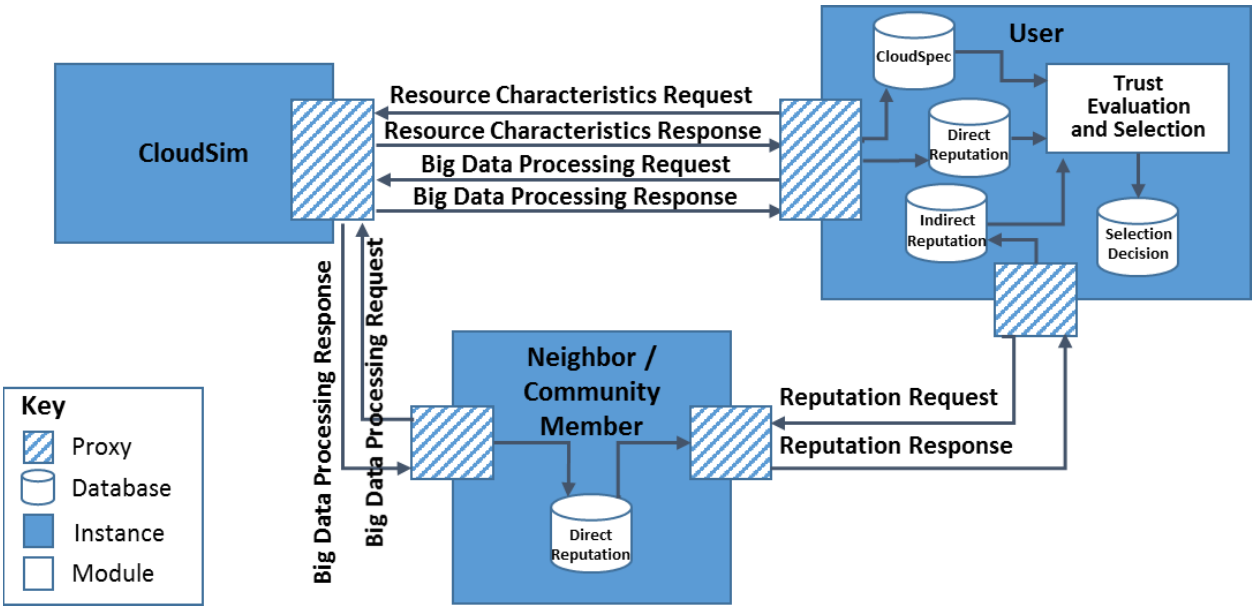

Figure 5.5: CloudSim extension framework.

5.5.2.4 Scenarios and Interpretation of Results

5.5.2.4.1 Evaluating Different MADM Techniques

In this section, we describe the main scenarios we developed to evaluate our reputation-based trust model.

Scenario 1: we evaluated the proposed MADM methods by comparing the different generated ranking results. Figure 5.6 shows that SAW and TOPSIS algorithms produce closer ranking results. However, the WPM algorithm shows a different ranking trend.

Scenario 2: we evaluated the scalability of our trust model where we increase the number of CSPs and measure the QoS of the selected CSP by each of the three MADM algorithms. We measure response time and cost quality attributes for this experiment. Figure 5.7 shows that response time and cost of the chosen CSP decrease as the number of CSPs increases for all three algorithms. This is because the more CSPs we have, the more options we have and, eventually, more chance to get a CSP offering better performance.

***Scenario 3*:** we evaluated the different weights assigned to different QoS attributes, and we analyzed the effect of changing the attribute weight on trust scores generated by each of the three MADM

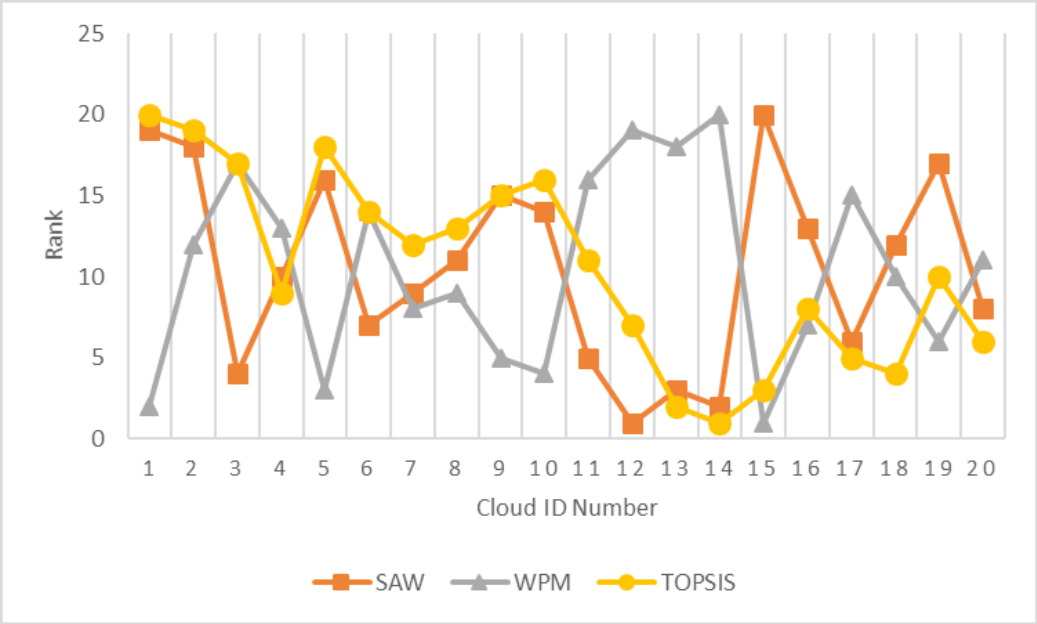

Figure 5.6: CSP rank.

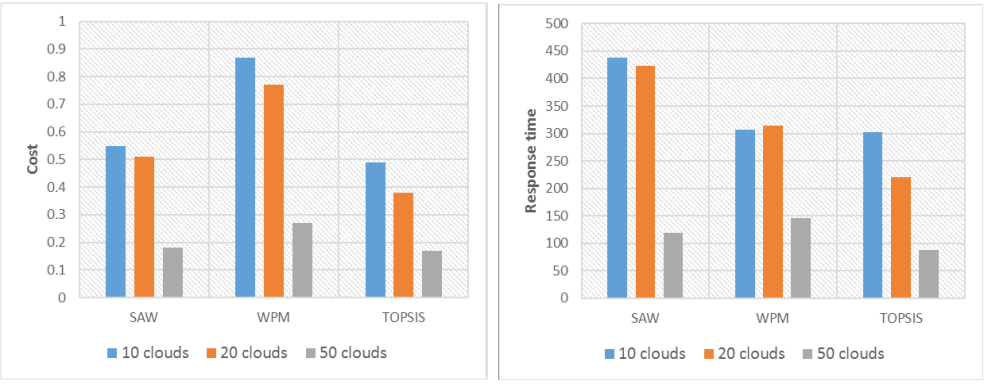

Figure 5.7: Scalability test of reputation trust model.

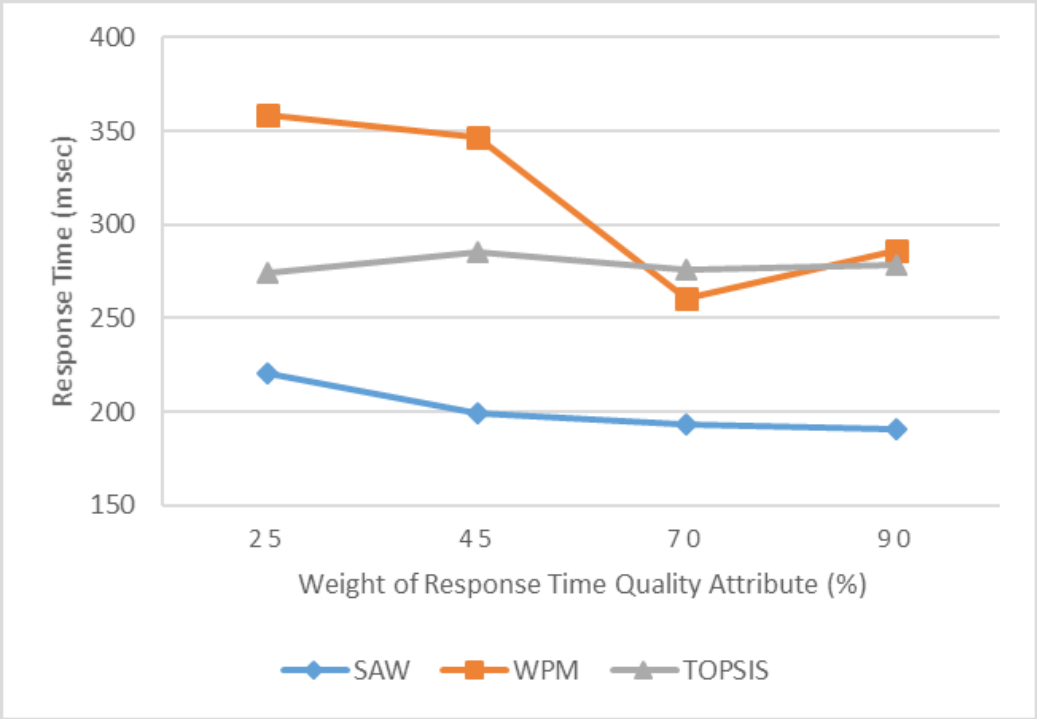

Figure 5.8: Response time for the reputation trust model.

algorithms. We chose response time, availability, and cost quality attributes for this experiment. Figure 5.8 shows that response time of the chosen CSP decreased as its weight value increased for all three algorithms. As the response time was made relatively more important, i.e., was favored over the other attributes for cloud selection, the selected cloud accordingly shows a better response time. We can also observe that the selection using SAW algorithm results in a better response time followed by the TOPSIS, and then the WPM.

Figure 5.9 shows that the availability of selected CSPs increases as its weight value increases for all three algorithms. This is because availability was also favored over the other attributes for cloud selection, so the selected cloud accordingly shows higher availability. Moreover, we can conclude that the selection based on the TOPSIS algorithm results in a better availability followed by the WPM and SAW algorithms.

In addition to response time and availability, we also tested the cost quality attribute. As shown in Figure 5.10, when the cost was favored over the other quality attributes, the selected CSP has a low cost for all three algorithms. The TOPSIS again, gives a better selection results as it shows lower

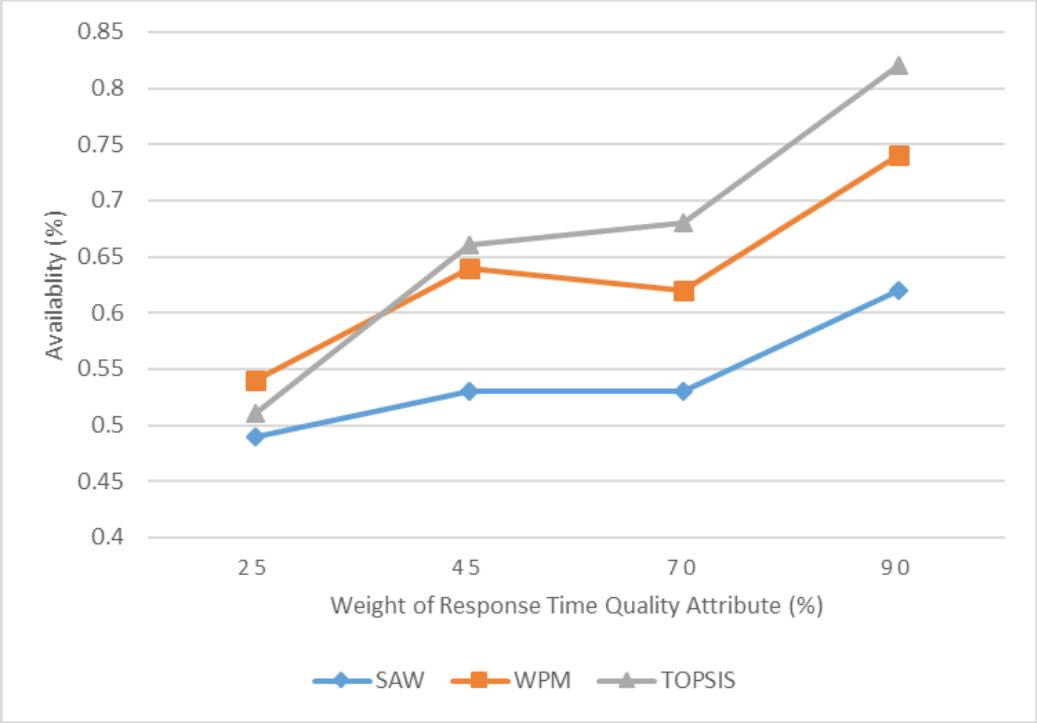

Figure 5.9: Availability for the reputation trust model.

cost values than the other two algorithms.

Scenario 4: we benchmark the three scoring algorithms with reputation-based trust scores calculated without user preferences. We compare our proposed user preference-based reputation model to a reputation model that does not involve the user's preference. The Benchmark Reputation Model (BMRM) hence, does not use weights for the attributes, but provides a local trust score to the user. We chose response time and cost quality attributes for this scenario. First, we give higher weight in favor of the cost quality attribute. The test shows that BMRM gives CSP selection with higher cost than the CSPs selected using our proposed model with the TOPSIS giving the best performance as in Figure 5.11. Second, we performed the test giving higher weight for the response time. Again, the results show that our proposed model using SAW, WPM, and TOPSIS algorithms perform better than the BMRM as shown in Figure 5.11.

Discussion:

To summarize this section, we evaluated a de-centralized reputation-based trust model to support Big Data processing over various cloud providers offering similar services. The model took into

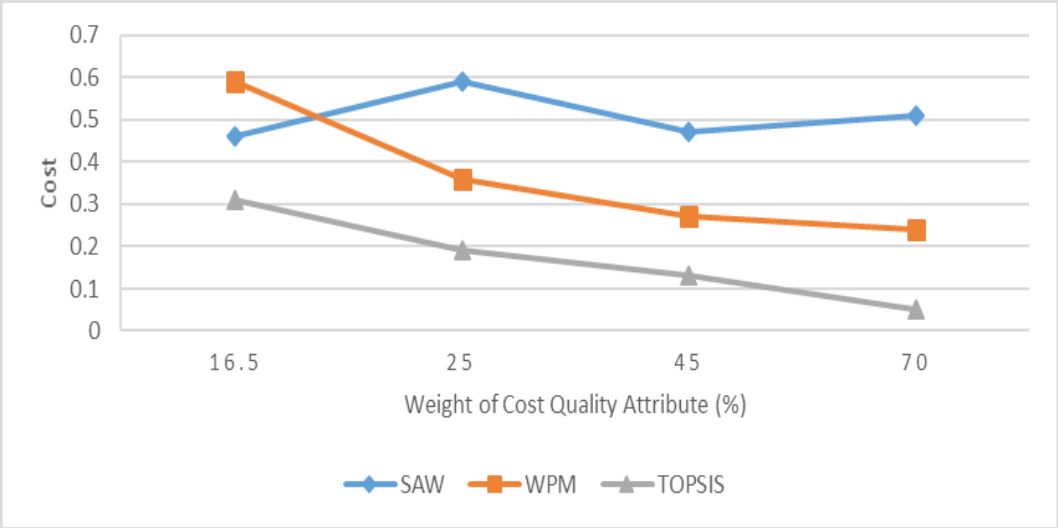

Figure 5.10: Cost for the reputation trust model.

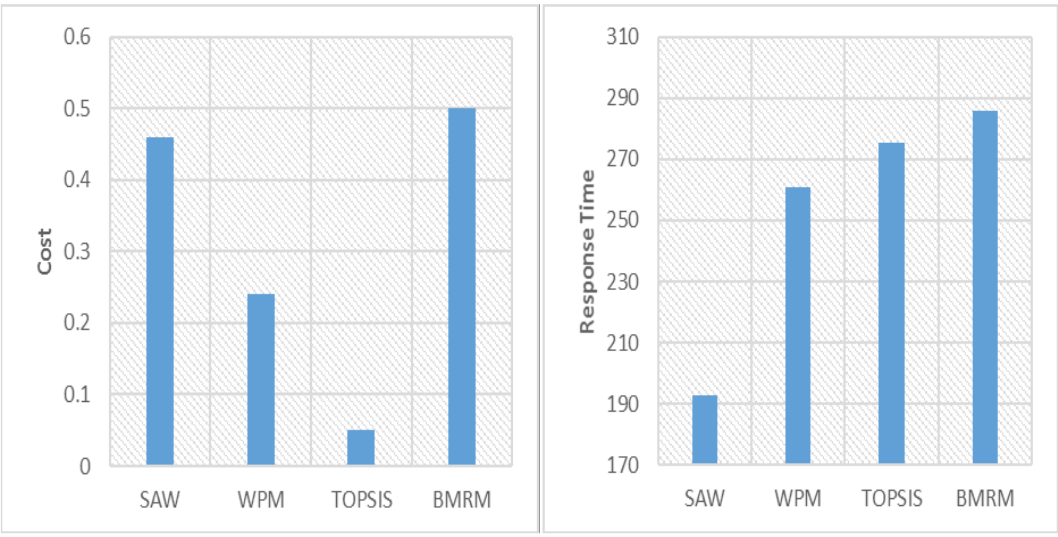

Figure 5.11: Cost and response time using different algorithms.

consideration the user QoS preferences in calculating trust scores. We also modeled the cloud selection problem as MADM relying on three trust scoring schemes. In addition, the model captured Big Data key characteristics and coped with some key features including flexibility, heterogeneity, and scalability of the studied environment. We conducted a set of experiments using a simulated cloud environment we developed to validate our trust model and assessing the three MADM methods. The results demonstrated that our proposed model captures users requirements and efficiently evaluates trust of cloud providers.

5.5.2.4.2 Evaluating MLR Prediction-based Trust Model

In this section, we describe the experiments we conducted to evaluate our proposed prediction-based trust model in which we apply MLR. The model relies on MLR to predict trust score of different cloud service providers.

The following is the implementation details of the main components involved in our trust prediction model, which we developed in Java. Our simulator implemented all modules described in Section 5.2 including user modules, which are the trust module, CSP reputation manager, transaction monitoring module, cloud providers' components, as well as neighbor components (e.g., other users). The simulation generates database logs that are analyzed using Weka MLR to predict the trust scores for each CP. All statistical results were obtained using R language and the packages MASS, DAAG and RELIMPO.

In this experiment, we generate 50 observations from one provider of the dependent variable trust denoted by *Y* and *six* explanatory variables data size (X_1), distance (X_2), availability (X_3), response time (X_4), confidence (X_5) and cost (X_6). First, the variable cost cannot be included in the model generated by one provider as it can only be used to compare different providers. We tested the correlation between the explanatory variables and the response variable, and we can clearly conclude that the correlations are significant with all independent variables except the confidence variable (X_5). Also, we note that the data size and the response time are highly correlated ($r = 1$). Therefore, the estimated regression equation is expressed by:

$\hat{y} = 0.00631 + 0.0243X_1 + 0.0165X_2 + 0.0194X_3$

The three variables have a significant positive effect (all p-values are close to zero) meaning that the

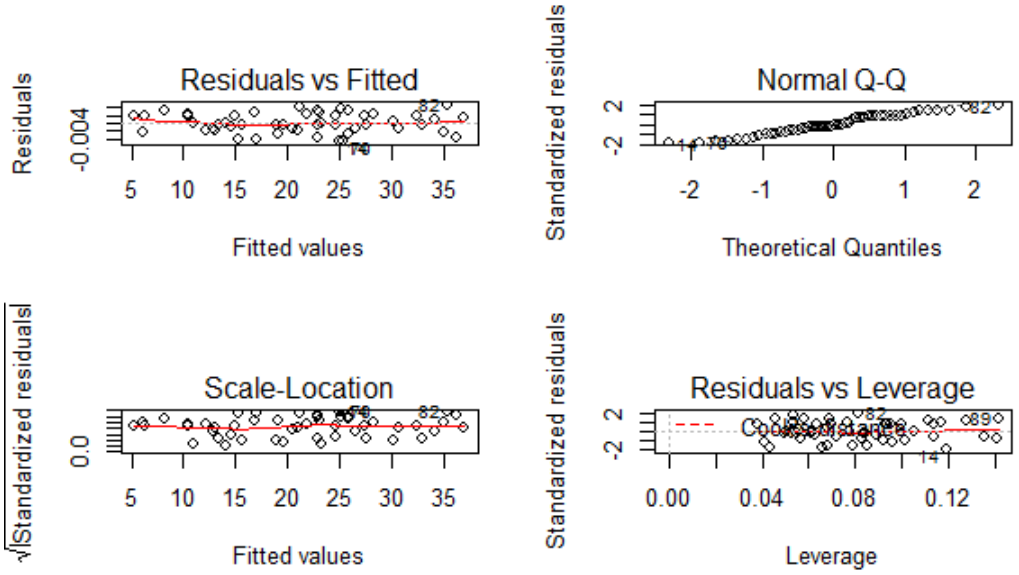

Figure 5.12: The regression residuals plot.

trust will increase with the increase of each of these explanatory variables. As depicted in Figure 5.12, the residuals satisfy the assumptions of normality (p-value for Shapiro-Wilk normality test is 0.1689), constant variance and independence. Using the cross-validation procedure to evaluate the consistency of the estimated regression equation, using three folds, we found that the model can perfectly be used to predict the response variable *trust* as depicted in Figure 5.13.

By calculating the relative importance for each explanatory variable, we found that the *data size* has the most relative importance for explaining the *trust* variable, roughly more than 62% followed by the distance variable, which has 25% of importance. The three variables explained 100% of the variability of the *trust* variable. To evaluate if the difference between the relative importance for trust is significant, we used the bootstrap procedure to calculate the confidence intervals of the difference between the relative importance of each pair of variables, as seen in Figure 5.14. Using the LMG metric, the 95% Bootstrap Confidence Interval (BCI) of the relative importance of data size variable is (51.43%, 71.56%) while using the LAST metric. We also note that the coefficient of determination is explained only by the data size and distance variables. In this case, the 95% BCI of the relative importance of data size variable is (78.29%, 89.32%).

In summary, we proposed a trust model for processing Big Data over different clouds. The model

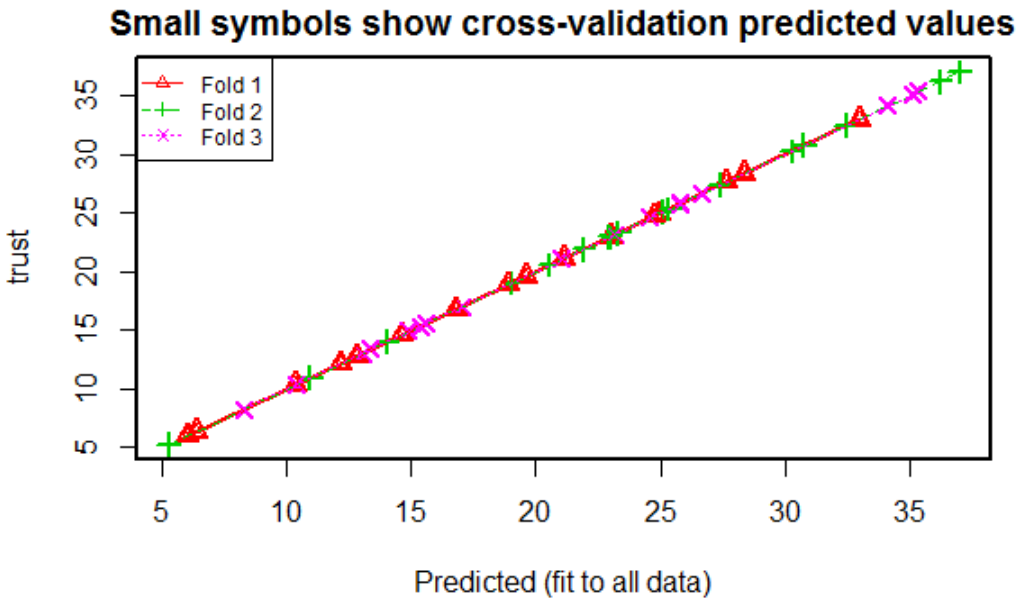

Figure 5.13: Cross-validation for predicted values.

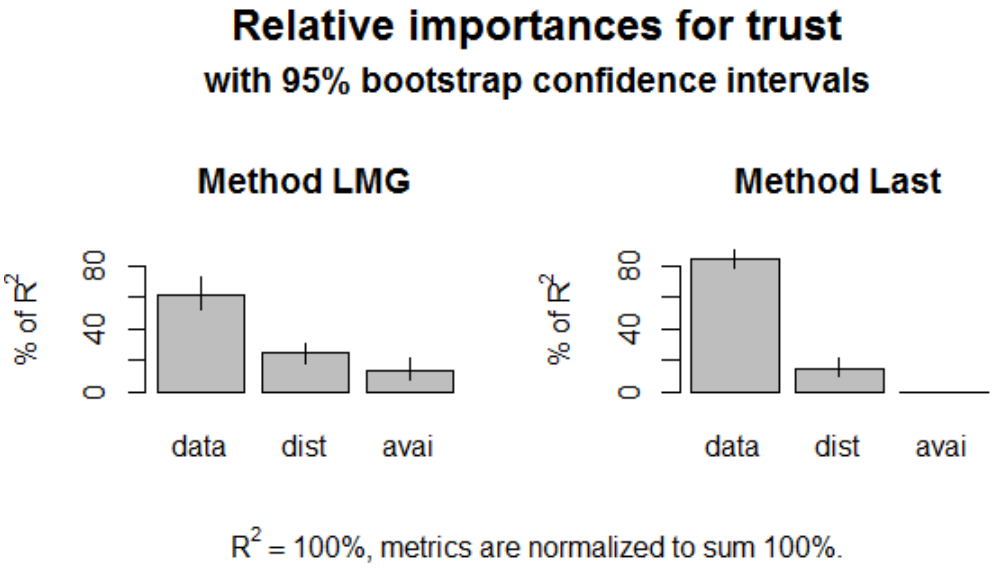

Figure 5.14: 95% Bootstrap confidence interval of relative importance for the trust.

applies MLR to predict trust scores for different cloud providers where trust is evaluated based on evidenced information collected about cloud resources availability,past experiences with the cloud provider, and the reputation collected from other user experiences with the same cloud services. The trust model we developed supports dynamic trust score calculations and updates, provides credibility validation through a community management system, and retrieves dynamical reputation scores. The model has been evaluated with few experiments and the results we achieved prove that our trust model exhibits high prediction accuracy. To evaluate the prediction accuracy, the consistency of the estimated regression equation, and the trust significance, we used the cross-validation method. As a result, we found that the model can perfectly be used to predict the response variable trust. Finally, we estimated and compared the relative importance of each explanatory variable in the model using the bootstrap confidence intervals for the difference between the relative importance of each pair of variables. We found that the data size variable explains the largest relative importance in the proposed trust model followed by the distance variable.

5.5.2.4.3 Evaluating Overall Framework Using SAW Model

We built a simulator in Java and implemented the trust evaluation algorithm depicted in Figure 5.4. We generated a series of ad-hoc client queries on Big Data sets that require different QoCS properties, values, and prominence. The client provides this QoCS information, including the weights of each QoCS attribute, via the application interface.

We evaluated whether the cloud selection manager module performed the appropriate selection of clouds that satisfy the client QoCS requirements. We also evaluated our CWTS evaluation algorithm against three other single-strategy algorithms. The first algorithm generates a trust score based on physical cloud characteristics and qualifications in terms of capacity, memory and processing power (Cloud_Spec). The second algorithm generates a trust score based on the calculated direct reputation, which consists of the logs of the past interactions between the Big Data application client and the cloud provider (Direct_Rep). The third algorithm generates a trust score based on calculated indirect reputation, which is the average trust score generated by the neighboring users in the community.

The neighbors collect reputation information during their communications with the cloud provider

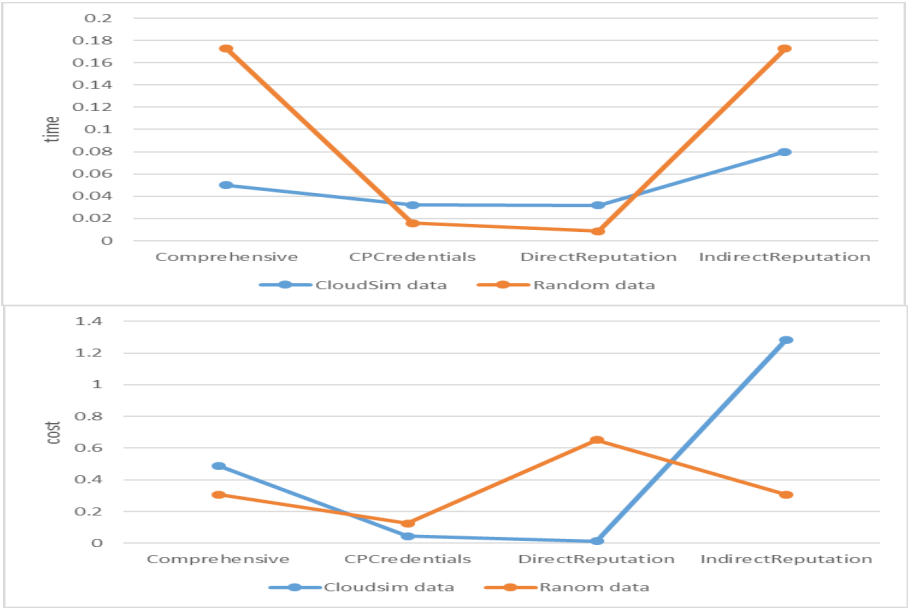

Figure 5.15: Response time and cost of the CWTS strategy versus other strategies.

(*Indirect_Rep*). We first ran the CWTS algorithm giving equal weight to all three of the aforementioned strategies. We then ran the CWTS algorithm giving 100% weight values to the *Cloud_Spec, Direct_Rep* and *Indirect_Rep* strategies, respectively. We also considered the FIFO strategy in some of our experiments. In the FIFO strategy, the first available cloud in the clouds database is selected. We compared the performance of each cloud's response time and cost values using each selection strategy. In the remaining part of this section, we describe each scenario in greater detail.

Scenario 1: We evaluated the CWTS algorithm against three other single-strategy trust score evaluation algorithms. We ran our simulation using the CWTS algorithm with equal weight values for each of the three strategies, and ran each strategy separately, as explained in the previous section. We measured the total data processing cost and response time of the selected cloud using each algorithm. We first ran this test using data populated by our simulation framework, followed by CloudSim-generated data. The results are shown in Figure 5.15 by two graphs indicating that the CWTS algorithm yields an average time response and cost compared with the other algorithms, which is due to using equivalent weights for each algorithm for evaluating the trust score of the CWTS.

Scenario 2: We changed the weight assigned to each strategy for the trust score evaluation using

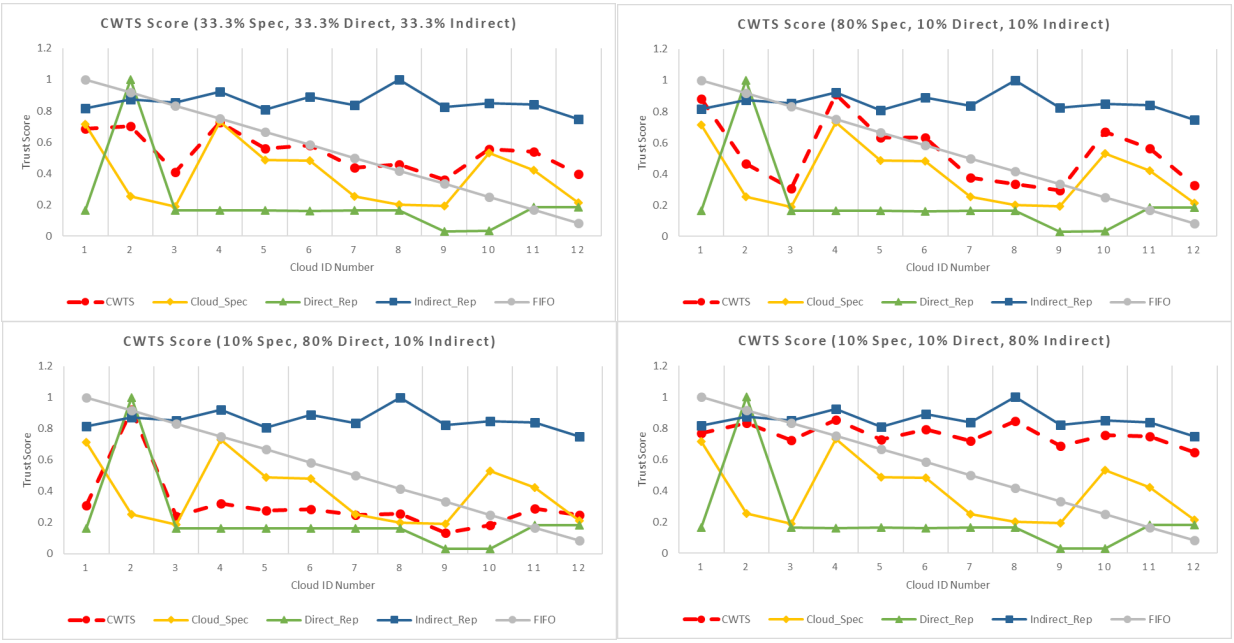

Figure 5.16: Behavior of CWTS under various assigned weight values for each strategy.

the CWTS algorithm and compared the resulting trust score values with those calculated via the *Cloud_Spec*, *Direct_Rep*, *Indirect_Rep* and FIFO strategies. Figure 5.16 shows the trust score of the selected cloud for each selection strategy. In this graph, we notice a trend in the CWTS algorithm: it adapts to the changes in weights assigned to each strategy, and the CWTS graph line behaves similarly to the strategy with the highest assigned weight. In addition, the FIFO algorithm graph line tends to decrease because it does not consider reputation, cloud characteristics or quality attributes. It assigns the highest weight to the first available cloud provider followed by gradually decreasing weights for the remaining cloud providers.

Scenario 3: We evaluated the performance of our CWTS algorithm in the presence of malicious or false reputation information from neighboring users. We measured the trust score of the selected cloud using the *CWTS*, *Cloud_Spec, Direct_Rep* and *Indirect_Rep* algorithms against the percentage of false ratings. In Figure 5.17, the trust score value for the *Indirect_Rep* strategy decreases dramatically as the percentage of false rating increases, which proves that depending solely on indirect reputation is unconstructive for selecting a suitable cloud. It also shows that the trust score generated by the CWTS algorithm is resistant to false ratings as it is not significantly affected by

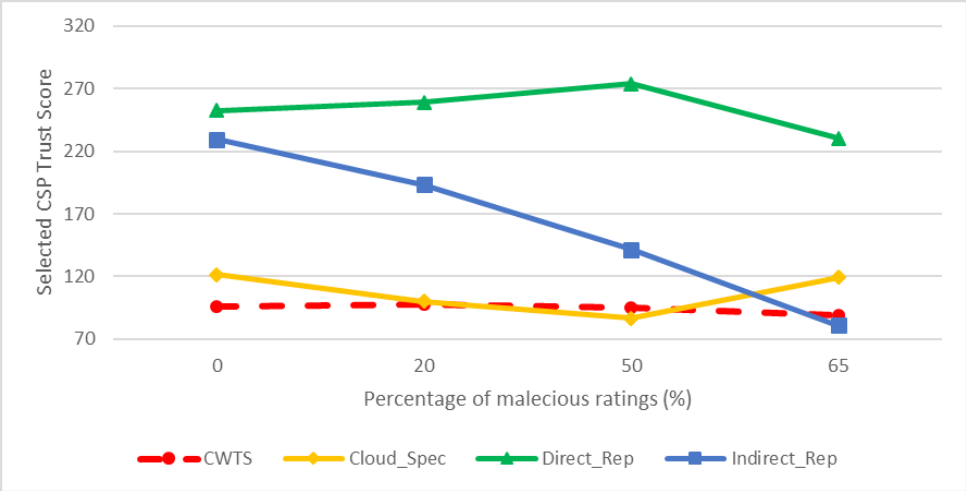

Figure 5.17: CWTS response to false ratings.

increases in the number of false ratings.

We also used the CWTS algorithm with different weight values for each strategy, giving more weight to indirect reputation, specifically. However, Figure 5.17 shows that the overall trust score of the CWTS was not affected by an increase in the false rating percentage, even when the indirect reputation strategy was given greater importance. Here, the weights are 60%, 20%, and 20% for the *Indirect_Rep*, *Direct_Rep*, and *Cloud_Spec* strategies, respectively. On another note, the graph lines in Figure 5.17 are based on pre-normalized trust scores, except for the CWTS.

Scenario 4: We simulated a new user trying to choose a suitable CSP and assumed that he/she had no prior experience with any of the cloud service providers (the direct reputation database contained no data). This user relied mainly on the indirect reputation database populated by reputation information from neighbors in the community as well as the cloud specification database populated by information collected from the existing clouds registered with the community. Figure 5.18 shows that the response time and cost of querying the selected cloud were satisfactory and unaffected by the lack of direct reputation. In addition, trust scores could not have been generated for any of the cloud providers if the user had relied solely on direct reputation.

Scenario 5: We changed the number of clouds and tested whether our CWTS algorithm had better performance using a greater number of cloud providers. Figure 5.19 and Figure 5.20 show that our CWTS algorithm led to improved response times with an in an increase in the number of clouds,

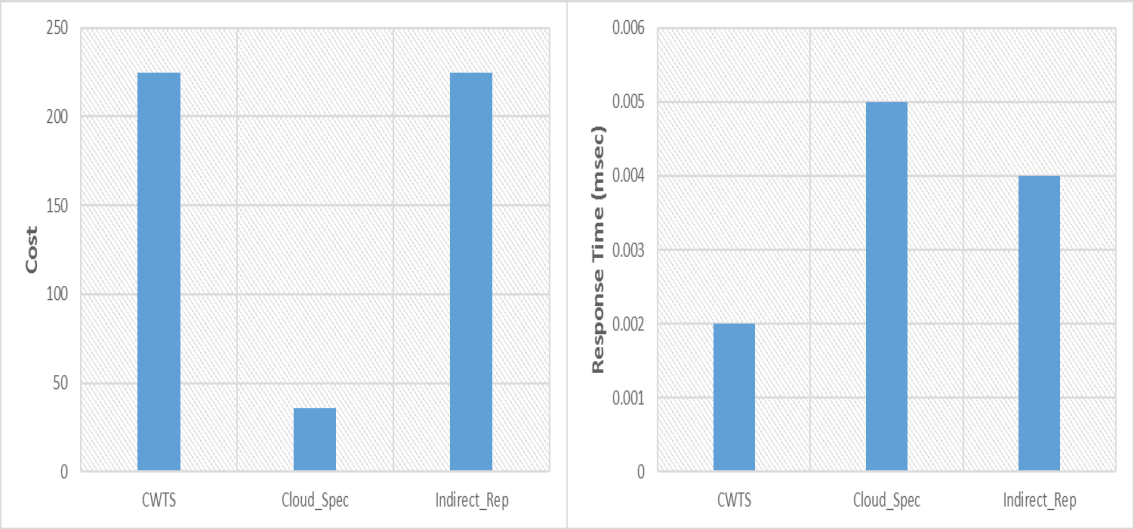

Figure 5.18: Behavior of CWTS when no user previous experience with the Cloud Provider.

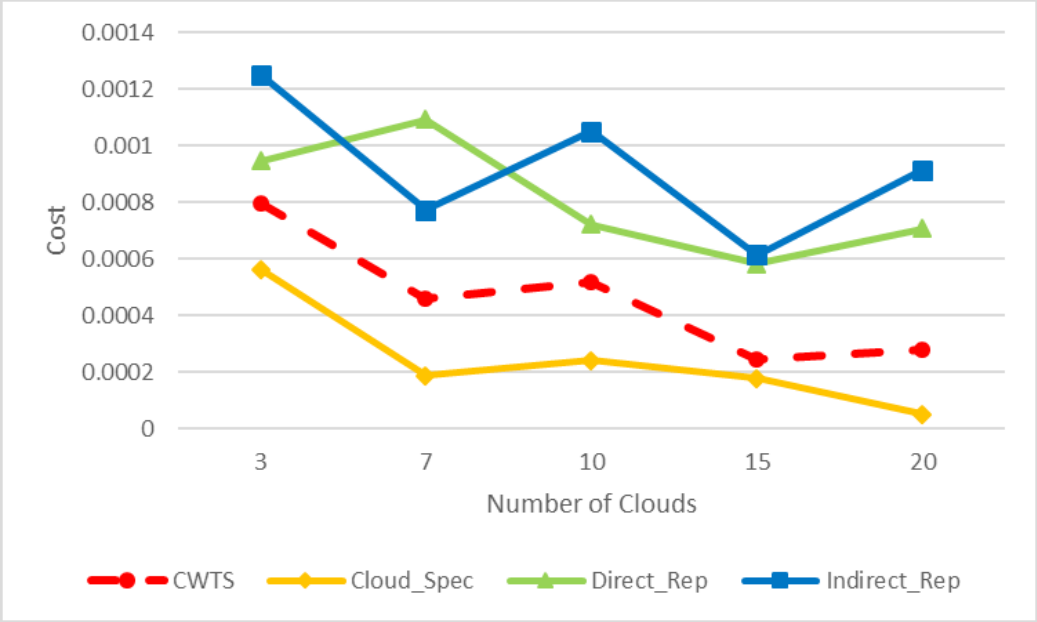

Figure 5.19: Cost measurements with an increasing number of Cloud Providers.

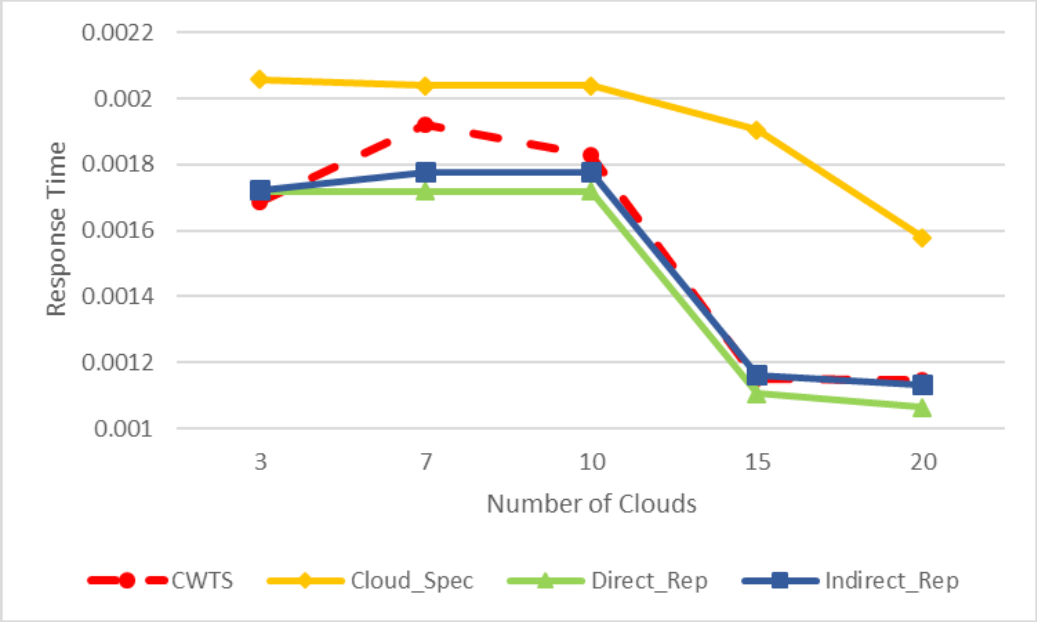

Figure 5.20: Response time measurements with an increasing number of Cloud Providers.

i.e., increased options. In addition, the figure shows that a lower cost is associated with an increase in the number of clouds. In this scenario, we populated our databases with randomly generated data from our simulator rather than CloudSim-generated data.

Scenario 6: We changed the weight value for a single QoCS attribute and tested the effect on the selection of the CWTS algorithm and the remaining strategies. We chose the response time as our test QoCS attribute. In this scenario, we used a random data generator to populate our databases. Figure 5.21 shows that response time decreased as its weight value increased. This can be explained by the fact that the response time was made relatively more important, so, the response time was favored over the other QoCS attributes during cloud selection.

5.5.2.5 Overall Discussion and Future Work

In this section, we discuss and evaluate our overall experimental results. Based on experimental and formal evaluations, in addition to the overhead evaluation described in the previous sections, we found that our trust model is sufficiently comprehensive and exhibits the following characteristics:

(1) It scales very well with increasingly high numbers of cloud providers, users, neighbors and

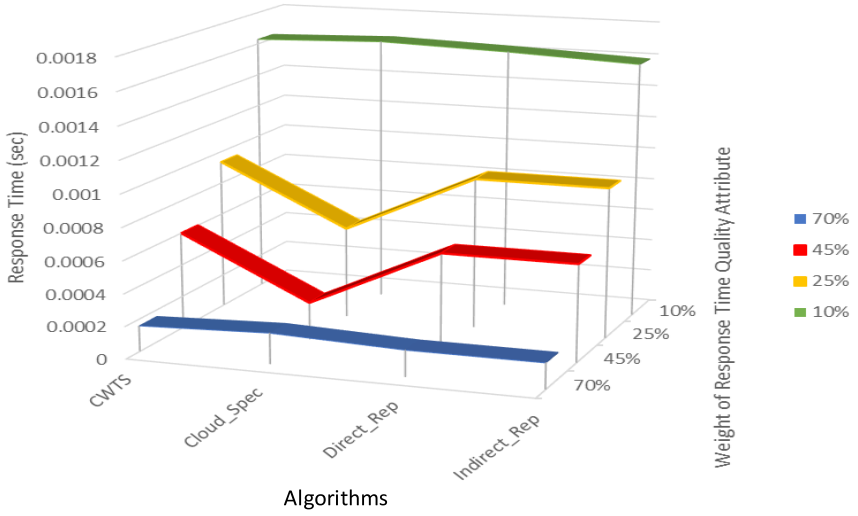

Figure 5.21: Behavior of the CWTS algorithm with various QoCS attribute weight values.

QoS properties.

(2) It balances the trust ratings of different stakeholders (cloud providers, customers and community members), objectively resulting in an accurate trust evaluation.

(3) It reacts efficiently to false ratings provided by malicious neighbors.

(4) It considers the user's QoS requirements and Big Data characteristics when evaluating the trust of cloud providers.

(5) It regulates and controls user behavior within a community of users wherein a set of rules, obligations, and penalties are enforced. This guarantees the accuracy of the ratings that contribute to the trust calculation.

(6) It exhibits negligible communication overhead during trust evaluation.

(7) It provides an application to accurately and easily retrieve the client's QoS requirements and Big Data quality properties prior to trust evaluation.

5.6 Conclusion

In this chapter, we proposed a multi-dimensional trust model that implements three strategies relying on the provider's advertised QoCS, neighboring assessments and on the user's past personal experience with the cloud provider. The neighbors' assessments are also evaluated based on the user preference regarding the significance of each quality attribute. Our model automates the decision-making process of cloud provider selection with an eye towards Big Data processing requirements and user QoCS preferences. We also proposed a community-based reputation provision wherein a community management system enforces a set of engagement and participation rules.

We also implemented a complete framework that calculates trust scores for given cloud providers and generates a recommendation for the cloud provider with the highest trust score. We conducted a series of experiments, and the results prove that our trust evaluation algorithms scale well with the number of requests with varying QoCS preferences.

We also proved that our trust model appropriately handles malicious trust scores from neighbors. The communication overhead of our solution was found to exhibit a small overhead. We have evaluated our CWTS algorithm against other strategies, and the results have convincingly shown that our CWTS algorithm selects the cloud that best matches the customer's QoCS priority requirements. We also built a CloudSim extension to simulate multiple clouds and provide an application layer to run more extensive experiments with our full framework.

Furthermore, we extended our trust model to include different MADM methods including SAW, WPM, and TOPSIS. We conducted a set of experiments using a simulated cloud environment we developed to validate our trust model and assessing the three MADM methods. The results demonstrated that our proposed model capture users' requirements and efficiently evaluate the trust of cloud providers. As future work, we plan to extend the model to cope with malicious reputation information.

Moreover, we formulated the trust evaluation problem in competing cloud environment using the MLR method for trust score prediction. Experiments conducted showed that the model could perfectly be used to predict the response variable trust.

Chapter 6

Trust Enforcement Through Self-Adapting Cloud Workflow Orchestration

In this chapter, we propose an end-to-end trust-based framework for orchestrating Big Data workflows in a competitive environments of a colossal cloud service providers. Such a framework allows the automation of the decision making process of selecting the most suitable cloud provider for Big Data processing that fulfills user's preferences which we proposed in Chapters 4 and 5. Furthermore, guaranteeing the required QoS levels during runtime and enabling automated reconfiguration of workflow orchestration to avoid quality degradation will be defined throughout this chapter as well as in Chapter 7.

We propose a workflow orchestration, monitoring, and adaptation model. This model relies on trust evaluation to detect QoS performance degradation and perform an automatic reconfiguration to guarantee QoS of the workflow. The monitoring and adaptation schemes can detect and repair different types of real-time errors and trigger different adaptation actions including workflow reconfiguration, migration, and resource scaling. We formalize the cloud resource orchestration using a state machine that efficiently captures different dynamic properties of the cloud execution environment. In addition, we use a model checker to validate our model in terms of reachability, liveness,

and safety properties. Extensive experimentation is performed using a health monitoring workflow we developed to handle datasets from Multi-parameter Intelligent Monitoring in Intensive Care III (MIMICIII) and deployed over a Docker swarm cluster. A set of scenarios were carefully chosen to evaluate workflow monitoring and the different adaptation schemes we implemented. The results prove that our automated workflow orchestration model is self-adapting, self-configuring, reacts efficiently to changes, and adapts accordingly while supporting a high-level of workflow QoS.

In this chapter, we first depict our proposed trust formalization and evaluation, and in Section 6.2 we detail our proposed self-adapting workflow orchestration model, including the architecture components and key features, formalization, and algorithms. Section 6.3 describes the cloud workflow monitoring state machine model and validation using model checker. In Section 6.4, we detail the experimentations conducted to evaluate our proposed monitoring and adaptation trust evaluation schemes. Finally, we conclude the chapter, and we draw some research directions for the next chapter.

6.1 Trust Formalization and Evaluation

6.1.1 Trust Evaluation of Cloud Workflow (Pre-deployment)

In this section, we explain the automatic evaluation of trust through a workflow that will be executed over a composition of cloud services. The selection of cloud services is based on the trust scores automatically evaluated before execution and during execution if re-allocation of cloud services or resources is needed. Trust should be based on a set of evaluation criteria with weights assigned to each of these criteria and decided by the user. The first criterion is the reputation of service components which generally relies on the user's past experience [150] [239]. This is called objective reputation and is done using monitoring, either by the users or third parties [240]. Other form of trust-based reputation relies on the opinion of users about the service which is known as subjective reputation. Both objective and subjective reputation can be combined to evaluate the trust and is referred to as a hybrid reputation scheme. Trust evaluation based on advertised QoS by service providers and self-experience can also be used. Each component service participates in the calculation of the overall trust of the composite service based on their contribution towards the composite

service. Each QoS attribute participates towards the overall trust evaluation with weights assigned by the user, commonly known as user preference based trust. The contribution of each component service should be automatically assigned and calculated. We detailed how QoS attributes are used for workflow trust evaluation previously in Chapter 4.

In Chapter 5, we evaluated the reputation of a single service, and reputation of composed services, which can be achieved using multi-attribute optimization techniques to measure and assess the reputation of every single service based on its contribution towards the overall trust of the composed service. The contribution ratio is determined by the user.

6.1.2 Trust Monitoring of Cloud Workflow Orchestration (Post-deployment)

After deployment, monitoring QoS of the workflow and all the allocated cloud resources, will guarantee the satisfaction of customer requirements. Monitoring the CPU utilization for example, will indicate that the application is performing as expected or experience delays when CPU is overloaded or might crash.

However, the complexity of monitoring Big Data workflows is characterized by the number of different QoS metrics that evaluate different activities and resources of the workflow. Such QoS metrics could be throughput, delay, event detection, response time, read/write latency, CPU utilization, energy efficiency, network delays, and bandwidth. Hence, it is rather challenging to combine all these different metrics into a holistic view across the workflow of different activities, the Big Data framework, and the utilized cloud resources. We depend on the cloud workflow quality specifications detailed in Chapter 4 in our orchestration, monitoring and self-adapting model, throughout the following sections.

6.2 Self-Adapting Workflow Orchestration Model

In this section, we describe the architecture we propose to monitor trust and QoS of the workflow orchestration to guarantee self-reconfiguring workflow upon the occurrence of abnormalities. Figure 6.1 depicts the main architecture components.

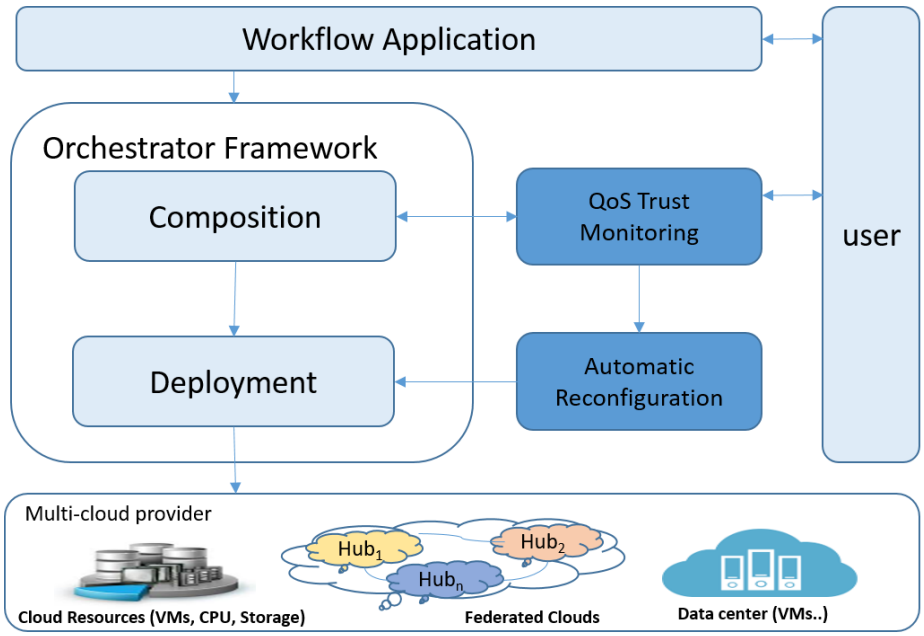

Figure 6.1: Workflow orchestration framework.

6.2.1 Architecture Components and Key Features

6.2.1.1 Cloud Workflow Composition

Big Data workflows are composed of various services some of which are dependent on another. In other words, changes in one service affect other dependent services. These services handle workloads with high volume and velocity data and have complex characteristics. Different application domains exhibit different modeling requirements that involve specific domain expertise to specify, understand and manage the entire pipeline of activities, data flow inter-dependencies, and the QoS properties and their levels and ranges. Once the workflow is designed, it is mapped onto an existing orchestration framework for deployment.

6.2.1.2 Cloud Workflow Deployment

Big Data workflow is mapped to orchestration frameworks that include Big Data programming APIs and cloud resources. The selection of suitable deployment configuration is challenging due to the complexity of the workflows and the abundance of selection possibilities. Choosing optimal workflow configuration is one of the open challenges that recently attracted researchers. For example,

stream processing requires an optimal combination of various worker instances to minimize the latency of processing activities and to optimize the cloud resources configuration. Such resource configuration includes the location of the data center, node hardware configuration, pricing plan, network latency, and bandwidth availability [40].

6.2.1.3 Trust-based QoS Monitoring

Workflows monitoring is required to guarantee that the run-time QoS is satisfied and that the deployed cloud resources are optimized. Monitoring means collecting performance status logs of all resources and running workflows. The importance of monitoring lies in detecting and handling problems, in addition to empowering flexibility of deployment. For example, monitoring the CPU utilization and data transfer activity will help to determine if containers are overloaded, underloaded, or operating as required [9].

We describe hereafter the main module of our architecture. After deployment, the monitoring module is responsible for monitoring the QoS of the workflow. It is first configured to set the QoS attributes that are required by the user along with their thresholds and acceptable values or range of values. Also, the user will assign trust evaluation preferences (weights) for each quality metric. Our monitoring system is responsible for monitoring each application including each composed service in the workflow application. Moreover, it is responsible for monitoring each data cluster of the service provider. The monitoring consists of three activities including monitoring the application, the cloud resources, and the QoS logs analysis.

Monitoring the application: a monitoring agent is placed on the master node of each cluster. This agent will continuously check logs generated by the application tasks. The logs contain different measurements collected on executed tasks such as throughput, latency, and errors (I/O error) resulting for example from invalid input or delay due to slow response from other dependencies. However, each task has its specific properties and metrics that should be tracked. Table 4.3 described earlier in Chapter 4 depicts some key metrics for different application types. Each task in the workflow is instrumented to generate the required measurement saved in the log files.

Monitoring the cloud resources: this module is responsible for monitoring the cloud resources orchestration and management. The main metrics to be considered include resource utilization such

as CPU usage, node CPU capacity, memory usage, node memory capacity, file system usage, and disk I/O. In addition, the monitoring observes the performance of the container such as container deployments, execution, and performance of required quality attributes.

QoS logs analyzer: part of the monitoring module that is composed of a set of processes distributed among each node. These processes collaborate to diagnose any problems, failures or abnormalities that occur in any application or happen in one of the clusters and evaluate a trust score for each node and task running on each node.

The design of process distribution works as follows: the node worker processes to monitor the node-specific quality metrics, the required metrics are passed through the main monitoring module along with their accepted values and ranges. The diagnose worker processes the watch of the streaming logs, checks the metrics values, and detects any out of range or failure values. The checked metrics values are interpreted, and a trust score, and is generated for each task and each node. These trust values are sent to the master node periodically after a specified time interval. Moreover, upon problem detection, a worker process sends a notification message to the master node analyzer process. The later analyses the notification messages coming from all worker processes and identifies the cause of the problem then sends a general notification message to the main monitoring and analyzer agent which resides at the user's side. Sending only the trust scores and the notifications upon failures reduces the communication overhead so that the monitoring activities will not affect the performance of the applications and the host clusters. The main monitoring and analyzer agent is responsible for generating a trust score for each application and cluster and sending the compiled problem notifications to the automatic reconfiguration module.

6.2.1.4 Cloud Workflow Automatic Reconfiguration and Self-Adaptation

Automatic reconfiguration is the mechanism of taking necessary actions when the monitoring process reports performance degradations. These violations might be with the running workflows, the underlying frameworks or the resources to allow automatic self-reconfiguration and maintain the required level of QoS. For example, if the monitoring process detects a dramatic performance degradation, then the automatic reconfiguration module will trigger operations such as scale up or migrate to preserve the required QoS. Other problems could be produced due to errors or unexpected

system behavior that might require restarting the container/VM which requires self-adaptation. The responsibility of the automatic reconfiguration module could be simple or sophisticated reconfigurations depending on the nature and the urgency of the occurred problem.

The complexity of dynamic and automatic reconfiguration of Big Data workflows arises because of its special characteristics are known by its multi-Vs. Hence, the first challenging issue is to model the QoS and estimate the data flow behavior with respect to volume, velocity, and variety and assessing the processing time and workflow I/O. Second, it is challenging to detect the cause of QoS abnormalities in heterogeneous frameworks as it can be originated, for instance, because of resource failure or congestion of network links. Another challenge is to model the runtime QoS changes of the workflow and construct orchestration so that the target QoS is upheld across the different layers of the orchestration framework.

Our automatic reconfiguration module detects the main cause of the problem upon receiving all the error occurrences in all applications and clusters from the primary monitoring module, then issues reconfiguration instructions to the corresponding application or cluster. Afterward, the reconfiguration instructions are sent back to the application or cluster to be reflected and deployed. The algorithms of each of the modules are detailed in the following section.

Automatic reconfiguration module: this module evaluates the status of each workflow and generates reconfiguration decisions to improve the performance of each workflow. This module receives and keeps the trust score for each workflow, the trust score for each cloud provider, and the error messages or abnormality notifications. Accordingly, it compares the latest trust score with the previous trust score, and if higher, then nothing will be done. However, if lower, then reconfiguration decisions should be made. Also, upon receiving error messages, reconfiguration decisions are made.

6.2.2 Automatic Cloud Workflow Trust Evaluation Model

Typically, tasks run independently or are tied together in an ad hoc manner. An orchestration environment, like Kubernetes, link these tasks together in a loosely coupled fashion. The following detail our monitoring model and Table 6.1 describes the symbols used.

Let ***Monitor*** (WF, Q) denotes a Monitor request to the global monitor GM to initiate workflow

Table 6.1: Symbols used.

P	number of tasks in the workflow
m	number of clusters allocated for a workflow
r	number of nodes in a cluster
s	number of containers allocated for a task
j	number of QoS attributes requested by the user
n	number of violation at time *t*

monitoring based on a given list of QoS attributes. The Monitor request starts the collection of the deployed workflow QoS logs. The workflow is modeled as a directed acyclic graph $\boldsymbol{WF}(\boldsymbol{T}, \boldsymbol{E})$ where $\boldsymbol{T} = \{tk_1,\ tk_2,\ \ldots,\ tk_p\}$ denote tasks to be monitored along with the deployment configuration which may include one or more clusters. The number of tasks in the workflow is denoted by p. Each task contributes with a different weight to the overall workflow. We denote the level of importance of a task towards a workflow by *il*. This value is given by the data analyst who constructed the workflow composition as $\boldsymbol{IL} = \{il_1,\ il_2,\ \ldots,\ il_p\}$, where p is the number of tasks in the workflow. $\boldsymbol{E} = \{(tk_i,\ tk_j) \mid tk_i,\ tk_j \in \boldsymbol{T}\}$, is the set of arcs representing a partial constraint relationship between tasks so that, $\forall\ (tk_i,\ tk_j) \in \boldsymbol{WF}\ (i \neq j)$, and tk_j cannot start until tk_i completes. Let $\boldsymbol{Clusters} = \{cl_1,\ cl_2,\ \ldots,\ cl_m\}$, where m is the number of clusters allocated for a workflow.

A $\boldsymbol{Container}$ is represented as $C\langle\ cn,\ tk_i,\ n_j,\ cl_k\rangle$, where:

- cn is a container *id* number, $tk_i \epsilon\ WF$, a node hosting cn, $n_j \epsilon\ Nodes$, and $cl_k \epsilon\ Clusters$ is the cluster that owns the node n_j.

- Each task tk is mapped to one or more node(s) in one or more cluster(s) and is represented as a tuple $tk\langle\ tn,\ \{c_1,\ c_2,\ \ldots,\ c_s\}\ ,\ st,\ in,\ out\rangle$, tn is the task name or id, and the second parameter is the list of destination containers allocated for that task. We assume that a task will run in one container per node. Multiple containers will be destined to multiple nodes. st is the state of the task (waiting, active, or completed) and in and out are the input and the output data set respectively.

- The $\boldsymbol{node}\ n_k\ \langle\ specs,\ lm\rangle$ is a tuple which represents the specification of the node, including

cpu, *memory*, and a local monitor lm which is responsible for calculating the trust score of the task and detect QoS violations.

- A $\boldsymbol{Cluster}$ $cl_j \epsilon Clusters$ is modeled as a list of nodes $cl_j = \{n_0, n_1, \ldots, n_r\}$, where n_0 is the master node and n_i is a worker node such that $i \in [1, r]$.

$\boldsymbol{Q} = \{q_1, q_2, \ldots, q_j\}$ where *j* is the number of QoS attributes requested by the user and the weights for each attribute are $\boldsymbol{W} = \{w_1, w_2, \ldots, w_j\}$.

We also refer to a list of QoS violations as $\boldsymbol{VList}(\Delta t) = \{v_1, v_2, \ldots, v_n\}$, at a time range/window Δt. We model the violation by a tuple $\boldsymbol{V}\ \langle C, Vtype, value, t\rangle$, where here the violation occurred at time t, is associated to a container tuple, the type of violation, and the value of violation (the abnormal value).

The Local Trust Score $\boldsymbol{LTS}$ is a score representing the level of satisfaction of all requested QoS attributes in $\boldsymbol{Q}$ according to the respective weights $\boldsymbol{W}$. The $\boldsymbol{LTS}$ is specific to each task running on a specific node. If the task is replicated on multiple nodes, then the $\boldsymbol{LTS}$ is aggregated as the average of all $\boldsymbol{LTSs}$ for that task among all containers.

$\boldsymbol{LTS^t_{ijk}}\ \langle tk_i, n_j, cl_k, qp, Q, W\rangle$, is calculated using a MADM method [234] while $\boldsymbol{Q}$ and $\boldsymbol{W}$ are the required quality performance values collected from worker node $\boldsymbol{n_j}$ in cluster $\boldsymbol{cl_k}$ for task $\boldsymbol{tk_i}$ at time $\boldsymbol{t}$ (where $\boldsymbol{t} > 0$), their weight, and its contribution towards the trust score respectively. The $\boldsymbol{qp_i^{'}}$ are the normalized task performance according to the QoS required value $\boldsymbol{qp_{target}}$. This guarantees that the trust score will be evaluated based on its proximity of the value to the required QoS value specified by the user and SLA which we describe as the target value (i.e., objective value). Alternatively, the target value could be the arithmetic mean of the maximum and minimum values in an accepted quality range $\boldsymbol{qp_{target}} = (qp_{min} + qp_{max})/2$.

$$\boldsymbol{qp_i^{'}} = \begin{cases} qp_i/qp_{target}, & qp_{target} > qp_i \\ qp_{taregt}/qp_i, & qp_{target} < qp_i \end{cases} \tag{48}$$

The calculation is performed by a local monitor $\boldsymbol{LM_j}$ residing in each node as a continuous function on the closed time interval [0, c]. If we consider an arbitrary constant $\boldsymbol{c} > 0$, then the average

local trust score $\boldsymbol{LTS^t_{ijk}}$ is represented by the following formula:

$$\boldsymbol{LTS_{ijk}} = \frac{1}{c} \int_0^c LTS^t_{ijk}\, dt \tag{49}$$

$\boldsymbol{ALTS_{ik}}$ is the aggregated $\boldsymbol{LTS}$ calculated at the master node $\boldsymbol{n_0}$ as the arithmetic mean of all trust scores collected from all worker nodes in cluster $\boldsymbol{cl_k}$ for a task $\boldsymbol{tk_i}$ at time $\boldsymbol{t}$ as $\boldsymbol{ALTS_{ik}}(t) = 1/r \sum_{i=1}^{r} LTS_{ijk}(t)$, where $\boldsymbol{r}$ is the number of worker nodes for one task $\boldsymbol{tk_i}$ deployed in $\boldsymbol{cl_k}$. The $\boldsymbol{ALTS_{ik}}$ is sent from the master node $\boldsymbol{n_0}$ in each cluster $\boldsymbol{cl_k}$ to the global monitor $\boldsymbol{GM}$. The following two scores $\boldsymbol{GTS_i}$ and $\boldsymbol{WFTS}$ are calculated at the $\boldsymbol{GM}$ as follows:

$\boldsymbol{GTS_i}$ the global trust score, is the average of all trust scores for task $\boldsymbol{tk_i}$ across all clusters at time $\boldsymbol{t}$. $\boldsymbol{GTS_i}(t) = \sum_{k=1}^{m} ALTS_{ik}/m$, where $\boldsymbol{m}$ is the number of clusters, and $\boldsymbol{t}$ is the time at which the trust scores were collected. The workflow trust score at time $\boldsymbol{t}$ is the weighted sum of all $\boldsymbol{GTS_i}$ for all composed tasks according to their importance level il_i towards the workflow $\boldsymbol{WF}$.

$\boldsymbol{WFTS(t)} = \sum_{i=1}^{p} GTS_i(t) \times\ il_i$, where $\boldsymbol{p}$ is the number tasks in a workflow.

A $\boldsymbol{Report}$ is a message that contains: 1) a workflow trust score, 2) list of trust scores of all composed tasks and 3) a list of QoS violations periodically sent from $\boldsymbol{GM}$ to the $\boldsymbol{ReconfigMgr}$. We model the Report as a tuple:

$\boldsymbol{Report}\, \langle WFTS\,(t)\,,\ \{GTS_1\,(t)\,,\ GTS_2\,(t)\ \ldots\ GTS_m\,(t)\}\,, \{v_1,\ v_2,\ \ldots,\ v_n\}\,\rangle\,.$

The $\boldsymbol{Handle}\ (Report)$ is the process called by the Global Monitor $\boldsymbol{GM}$ to the $\boldsymbol{ReconfigMgr}$ when a QoS violation is detected during runtime or periodically as explained earlier.

The $\boldsymbol{ReconfigMgr}$ processes the $\boldsymbol{Report}$ and reaches an automatic reconfiguration decision. The decision function $\boldsymbol{D}$ $At\ time = t$, is modeled as follows:

$$D(WFTS_t, VList_t) = \begin{cases} 1, & \text{if } V\ != null \\ -1, & \text{if } V = null\ \&\&\ WFTS_t < WFTS_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{50}$$

A $\boldsymbol{Decision}\ (NewConfigList\ \{\ \langle tk_i, c_j,\ configFile\rangle\})$ message is sent about each workflow to the concerned party to change the configuration. The $\boldsymbol{NewConfigList}$ includes a list of suggested configurations for one or more tasks in the workflow. Each tuple in $\boldsymbol{NewConfigList}$ contains

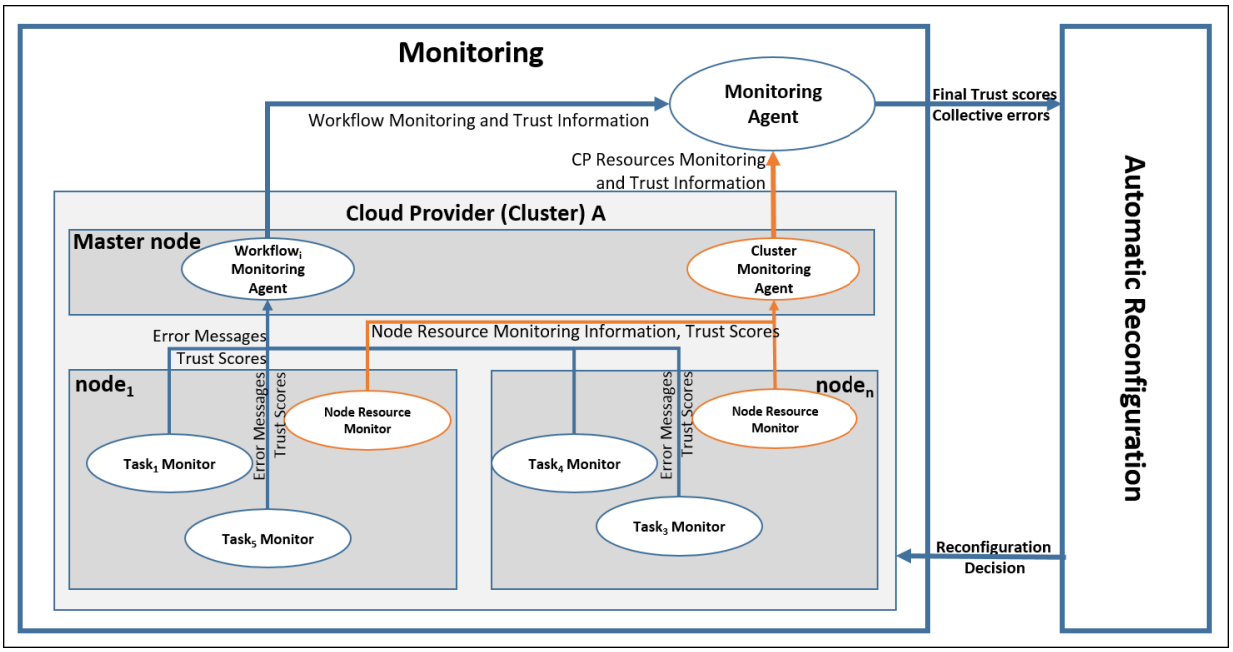

Figure 6.2: System architecture.

the task tk_i, destination container c_j, configuration file $configFile$, which is a script containing the new configuration suggested by the $ReconfigMgr$ usually specified in yaml format, which is a simple commonly-used language for application configurations that is compatible with many other languages and frameworks [241]. It is enhanced for data serialization, configuration settings, log files, and messaging, which fits our framework requirements. The destination of this message is the master node of each cluster hosting the container specified in the $NewConfigList$.

6.2.3 Automatic Cloud Workflow Trust Evaluation Algorithms

In this section, we propose automatic workflow trust evaluation algorithms during the pre-deployment, post-deployment, and self-adaptation in case of QoS requirements violation. The system architecture of our model is shown in Figure 6.2.

6.2.3.1 Pre-deployment Workflow Trust Evaluation

The services are composed of an optimal set based on trust scores according to QoS constraints. The trust scores of each service are generated based on historical QoS logs. Then, we compute the QoS

aggregation value of each workflow path and select the best path that meets the QoS requirements. We use the MADM method for trust evaluation of each task. Accordingly, the workflow tasks are mapped to a specific resource that responds to its QoS requirement. Mapping the services to the resources can be achieved using similarity matching as an initial deployment. For example, if the task needs storage, we match it to a resource with high capacity storage resource, and if it requires high processing, we match it to a high processing power server.

6.2.3.2 Post-deployment Trust Monitoring

Trust monitoring consists of measuring trust values that support the two modes of monitoring operations of periodic or continuous monitoring. The continuous operation mode requires running the monitoring process as a daemon that logs the status of the monitored tasks and system. The trust scores are evaluated by our monitor module which is comprised of two submodules: the local monitor (at master node, or worker node) and global monitor. The following describes the key activities supported by both local and global monitor for the sake of monitoring:

At the local monitor:

(1) Collect the performance values according to QoS required list for a task in the ***WF***

(2) Evaluate a trust score for a task

(3) Produce the output of a trust score for a task at node ***i***

At the local monitor in master node:

(1) Collect trust scores from all local monitors in other nodes for a task.

(2) Calculate the average trust scores to get *ATS* for a task at cluster *k*.

(3) Output is the *ATS* for a task at cluster *k*

At the global monitor:

(1) Collect *ATS* aggregated trust scores from all clusters for a task

Algorithm 6 Trust score calculation algorithm

Input:
$Tasks$: List of Tasks,
$QoSList$: List of QoS attributes,
$weights$: Weights of each QoS attribute
Output: $LTSList$: Local Trust Score updated for each Task
procedure EVALUATELOCALTRUSTATWORKERNODE($Tasks$, $QoSList$, $weights$)
    for $t \leftarrow 1, c$ **do**
        $scoresList_t \leftarrow empty$
        for all $tk \in Tasks$ **do**
            $score \leftarrow 0$
            for all $q \in QoSList$ **do**
                $score \leftarrow score \ + \ measuredQVal_q \times weights_q$
            end for
            $scoresList_t[tk] \leftarrow score$
        end for
    end for
    for all $tk \in Tasks$ **do**
        $LTSList[tk] \leftarrow \frac{1}{c} \int_0^c scoresList_t[tk]\ dt$
    end for
    return $LTSList$
end procedure
Output: $ALTSList$: Aggregated Trust Score (across nodes) for each Task
procedure EVALUATEAGREGATEDLOCALTRUSTATMASTERNODE
    for all $nodes \in Cluster$ **do**
        $getLTSList_{node}$
        for all $tk \in LTSList_{node}$ **do**
            $ALTSList[tk] \leftarrow ALTSList[tk] + LTSList_{node}[tk]$
        end for
    end for
    for all $tk \in Tasks$ **do**
        $ALTSList[tk] \leftarrow ALTS[tk]/nNodes$
    end for
    return $ALTSList$
end procedure
Output: $GTSList$: Global Trust Score (across clusters) for each Task
procedure EVALUATEGLOBALTRUSTATGLOBALMONITOR
    for all $cluster \in Clusters$ **do**
        $ALTSList_{cluster}$
        for all $tk \in ALTSList_{cluster}$ **do**
            $GTSList[tk] \leftarrow GTSList[tk] + ALTSList_{cluster}[tk]$
        end for
    end for
    for all $tk \in Tasks$ **do**
        $ALTSList[tk] \leftarrow ALTS[tk]/nClusters$
    end for
    return $GTSList$
end procedure

(2) Calculate the average trust scores to get *GTS* for a task among all clusters and calculates the *WFTS* for all tasks in a *WF* according to the task importance (weight) towards *WF*.

Algorithm 6 depicts this trust score calculation algorithm.

Algorithm 7 Automatic reconfiguration of workflow orchestration algorithm

Input:
$taskViolations$: QoS task violation List,
$sysViolations$: QoS system violation List,
$GTSTable$: GTS for each task in WF
Output: NewConfig
procedure AUTORECONFIGALGORITHM($taskViolations$, $sysViolations$, $GTSTable$)
  for all $tk \in taskViolations$ **do**
    $sv \leftarrow findNode(sysViolations)$
    if ($sv \neq \emptyset$)
      $svType \leftarrow violationType(sv)$
      $if(svType =$ "$sysOverload''$)
        $newConfig[tk] \leftarrow addNode(getCluster(sv))$
      $else\ if(svType =$ "$sysOverloadNoExtend''$)
        $newConfig[tk] \leftarrow migrate(tk)$
      $endif$
    $else$ //problem in task
      $newConfig[tk] \leftarrow scaleUp(tk)$
    $endif$
  end for
  for all $tk \in GTSTable$ **do**
    $avgT \leftarrow avg(historyTrust[tk])$
    $if(trust(tk) \leq avgT)$
      $newConfig[tk] \leftarrow findNewDeployment(tk)$
    $else$ //problem in task
      $newConfig[tk] \leftarrow \emptyset$
      $update(historyTrust[tk], trust(tk))$
    $endif$
  end for
  return $newConfig$
end procedure

6.2.3.3 Automatic Reconfiguration of Workflow Orchestration

Algorithm 7 depicts the automatic workflow orchestration reconfiguration algorithm. This algorithm analyzes each task violation by checking the root cause of the violation. For example, it checks if a resource limitation is the cause of the violation such as an overloaded node, then a message is triggered to add a new node to the cluster. However, if the cluster cannot be extended, then a migration message is issued, and the task is allocated to a new cluster (see Table 6.2). The

algorithm also analyzes the new trust scores for all the tasks in the workflow, and if it detects trust score degradation, then it generates a new configuration decision.

6.3 Cloud Workflow Monitoring Model

6.3.1 Characterizing System Elements and State Description

In this section, we model the parameters characterizing the state of each system component. We need to model the workflow and its constraints so that the monitoring system actions take into consideration the workflow status including task choreography, dataflow, recovery, and task dependencies. For example, if we have two tasks, *T1* and *T2*. We call *T2* dependent on *T1* when *T2* is invoked after the *T1* response is received or completed.

We also consider the data flow where the task input and output states are tracked. For each task *T1*, we retain information about the parameters, the data type and format of parameters, and the time expiry and validity of parameters. Additionally, recovery actions should be triggered when an error or delay receiving a response occurs such as *T1* terminate, *T1* reconfiguration (assign to the different cluster), or Ignore error.

6.3.1.1 Tasks

As described above, a task is modeled as a tuple $tk \langle\, tn,\ \{c_1,\ c_2,\ \ldots,\ c_s\},\ st,\ in,\ out \rangle$ previously detailed in Section 6.2.2 and task dependency is modeled in $\boldsymbol{E} = \{(tk_i,\ tk_j) \mid tk_i,\ tk_j \in \boldsymbol{T}\}$. In this section, we detail the state, input, and output. Figure 6.3 shows the states of each task and the related transitions. The state *st* of a task can be idle, running, and completed. Idle state is the state of the task before it starts running, a task is in running state when the previous task is completed, and the input is ready. However, a task is completed when the output set is ready.

6.3.1.2 Events

The event is usually a violation that occurs in a node or to a specific task such as CPU overload, disk full, increasing task errors, and task overload. We construct an event as a message sent to the master node with the format:

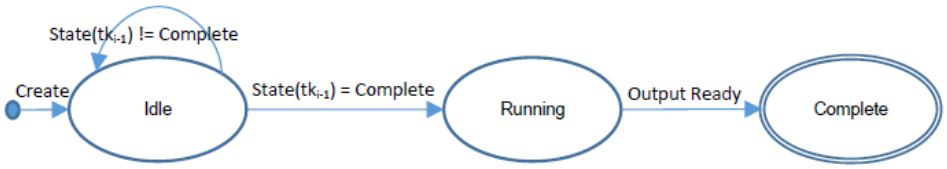

Figure 6.3: Task state machine automata.

sendNodeViolationMsg (source: (node, cloud), dest: master, <Type, value, category>, t).

Accordingly, the master node compiles a list of all received messages to be sent to the General Monitor with the format:

sendClusterViolationMsg (source: (cloud), dest: GM, list {<Type, value, category>}, t)

6.3.1.3 Monitoring Messages Specification

All the messages used in our workflow monitoring system and their details including source, destination, parameters and description are shown in Table 6.2.

6.3.2 Cloud Workflow Monitoring and Adaptation State Machine

Figure 6.4 depicts the state machine automata of our monitoring and reconfiguration framework. The following sections describe in detail this system state machine.

6.3.2.1 Workflow Monitoring

As mentioned above, monitoring consists of collecting the logs and QoS information regarding all the entities of interest, such as tasks and resources. It is also responsible for updating the trust scores of each task using the collected logs analysis results. Upon violation detection, a violation message is sent to the reconfiguration manager. During monitoring, the states of each entity are updated and kept in the system for further use during the reconfiguration state.

6.3.2.2 Workflow Reconfiguration

Upon a reconfiguration decision, the *AR* module decides what new configuration is suitable for the situation. The following is the description of the possible changes and the implication of each

Table 6.2: Workflow monitoring messages.

Message	Source	Destination	Parameters	Description
getLTSMsg$_t$	*Master node*	*Worker node*	*Q,W, list*{*taskid*}	The master node sends this message to all worker nodes in the cluster to collect the task trust values according to the required quality attributes and their weights passed in the message parameters.
replyLTSMsg$_t$	*Worker node*	*Master node*	*List* {<*taskid, LTS*>}, *List*{*sysViolations*}	This message contains a list of all task trust scores from each worker node to the master node as a response to getLTSMsg$_t$ message. This message also contains a list of system violations, such as CPU overload.
sendALTSMsg$_t$	*Master node*	*GM*	*List* {<*taskid, ALTS*>}, *List* {<*node, sysViolations*>}	This message contains the list of aggregated trust scores for each task running on this cluster. Also, it contains a list of system violations for each problematic node.
sendFTSMsg	*GM*	*AR*	*WF, list* {<*taskid, GTS*>}, *list*{*sysViolations*}, *list*{*taskViolations*}	This message is sent from the GM to AR for each WF and contains the list of tasks composed in the WF along with their GTS. Also, it contains the list of system violations and list of task violations.
autoReconfig	*AR*	*taskid, node, cluster*	*Reconfig File*	This message contains all reconfiguration commands issued by the AR and regarding each task ids in a certain node and certain cluster.

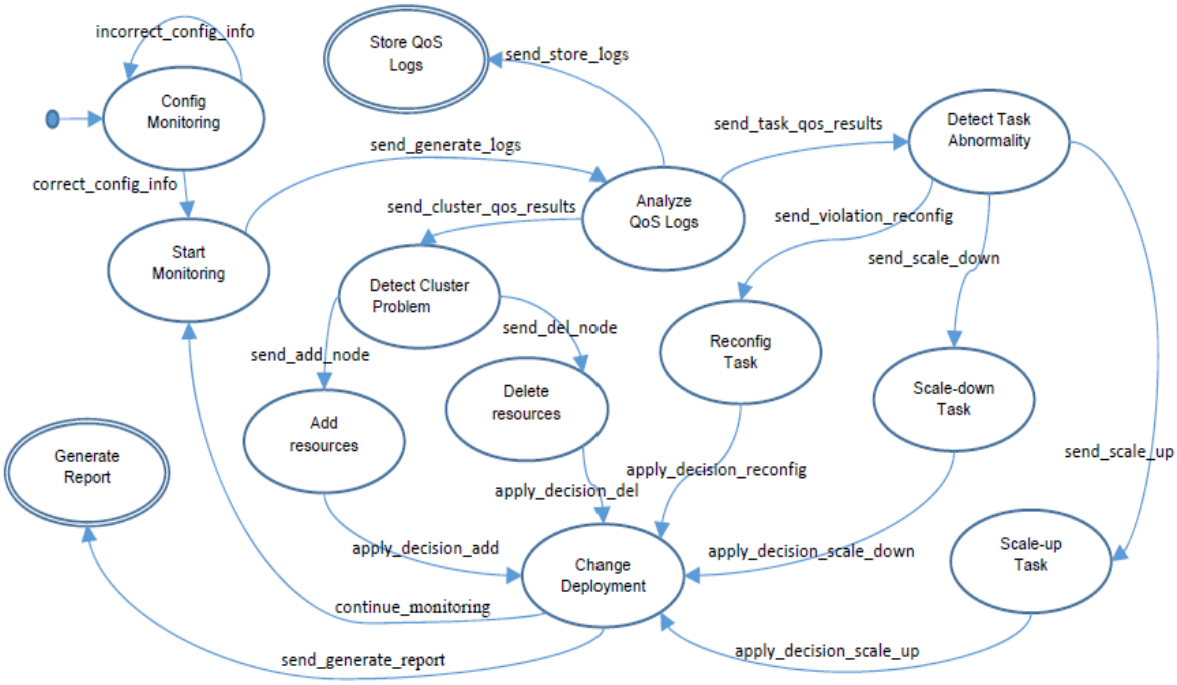

Figure 6.4: Workflow monitoring and adaptation state machine.

change regarding the state of the *WF*, task, and resources. The *AR* module first checks the state of the task, according to the task type (if the task allows scaling during the running state). If the task type is scalable, then scale up or down (by applying the change in the configuration file and deploy) and update the state of the task accordingly.

Scale up: run additional replications of the task on more nodes to handle the heavier traffic input, then update the state with the new number of replicas.

Scale down: when unused replicas are detected, then the replicas are deleted, then update the state with the new number of replicas.

Reconfigure: change the deployment configuration for the task by changing the node or cluster assignment according to considerations such as task type, task state, and task dependency. The task type can be scalable or non-scalable, and the task state can be waiting, running or complete, and the task dependency can be dependent on other tasks or other tasks dependent on this task.

Usually, the type of reconfiguration decision is taken following a QoS violation. For example, a migration decision is only taken depending on the severity level of the violation and the state of the

task. If there is an issue within the cluster (e.g., CPU overloaded) and the processing performance is degrading over time, then the decision is to migrate the task to another cluster having the best QoS trust score recently measured. In order to satisfy the self-adaptation feature during reconfiguration, specifically the migration decision, the state of the task plays a significant role. In other words, migration should consider the task and its dependent tasks including all the dependent task list. for simplicity, we do not need to migrate the predecessor tasks. Moreover, all the dependency input data should also migrate.

In case the cluster performance is degraded with a rate higher than a certain threshold, migrating the whole workflow is considered. If the task state is *'waiting,'* then the migration is straightforward, and the task along with its input dataset is migrated to the new destination (e.g., node). However, if the task state is *'completed,'* then migration is performed for the remaining dependent tasks in the workflow along with their input dataset. Nevertheless, when the task state is *'running,'*, many issues should be handled so the workflow required QoS is not affected. On the one hand, if the violation type is causing a service interruption, then we restart the task from the beginning at the new destination by resetting its state to *'waiting.'* On the other hand, deciding whether to move the task immediately or wait until it completes depends on the task completion status. The task completion status can be measured by calculating the percentage of generated output data against the expected output data. If the percentage of completion of a task is higher than a certain threshold, then we wait until the task is *'completed'* and migrate the remaining dependent tasks in the workflow. Otherwise, the task is considered at the beginning stage, and it is reset to *'waiting'* state, then migrated to the new destination.

6.3.3 Quality Metrics

The following in Table 6.3 are the common metrics and thresholds used to help in adaptation decision making and reconfiguration actions. Such threshold values are based on the application domain, workflow type, and user requirements. These values are reevaluated for every workflow according to its application domain and nature. The details of the suggested quality attributes are depicted in Chapter 4.

Table 6.3: Quality violations.

Quality Violation	**Threshold**
$abnormalCPUUtilization\ (x)$	80%
$abnormalHighMemUtilization\ (x)$	80%
$abnormalLowMemUtilization\ \ (x)$	15%
$abnormalNetworkAvailability\ (x)$	10%
$abnormalDiskAvail(x)$	80%

The priority of each of the above metrics varies according to the task QoS requirements. We define two classes of priority, $highPriority$ and $lowPriority$. Furthermore, we define two violation alert types, severe and moderate as:

$$severeViolationAlert(x) \leftarrow (lowPriority(x)\ \wedge\ EX\ lowPriority(x))\ \vee\ highPriority(x)$$

$$moderateViolationAlert(x) \leftarrow lowPriority(x)\ \wedge\ \neg\ highViolationAlert\ (x)$$

The reconfiguration decision is issued when a violation alert is received and includes either a high or low violation:

$$reconfig(x) \leftarrow highViolationAlert(x)\ \vee\ lowViolationAlert(x)$$

6.3.4 Validation-based Model Checker

The following describes our monitoring system where an administrator configures and initiates the monitoring process after workflow deployment. Once the system initializes the monitoring process, the QoS logs are generated, and the following actions are sequentially triggered when task abnormality is detected: *Analyze QoS Info, Store QoS Logs, Detect Task problem, Reconfigure Task, Change Deployment, and Generate Report*. Figure 6.4 describes the finite state machine of the workflow monitoring and adaptation system where a unique name identifies each state and connected to other states through applicable transactions. The transactions are labeled with names corresponding to the actions.

According to the type of detected problem, the system takes an appropriate action to maintain the required workflow QoS level. In the case of detecting an issue with task execution, such as low task response time is encountered then a scale up state is initiated where more containers are allocated for that task.

To formalize our monitoring system, we assume that our system is composed of a set $M = \{1, 2, \ldots, n\}$, of *n* services interacting together. Each service *i* ϵ *M* is defined by:

(1) A set of LS_i finite local states as shown in Figure 6.4 where *start_monitoring*, *analyze_QoS_info*, *store_QoS_info*, and *detect_task_problem* are some of the system local states.

(2) A set of LA_i of finite local actions as shown in Figure 6.4, for instance, *send_generate_logs*, *send_task_qos_results*, and *send_generate_report* are some of the system local actions.

(3) A local protocol $Pr_i : LS_i \rightarrow 2^{LA_i}$ is a function that describes the set of allowable actions at a given local state. For example, the following is one protocol depicted from Figure 6.4. $Pr_n\,(analyzeQoSInfo)$ ={*send_cluster_qos_results*, *send_task_qos_results*}.

At a given time, the configuration of all services in the system is characterized as a global state *S* of *n* elements represented as $gs = \{e_1, e_2, \ldots, e_n\}$, where each element $e_i \epsilon LS_i$ denotes a local state of the service *i*. Hence, the set of all global states $GS = \{LS_1 \text{ X } LS_2 \text{ X } \ldots \text{X } LS_n\}$ is the Cartesian product of all the local states of *n* services. The global transition function is defined as $T \;\; GS \text{ X } LA \rightarrow GS$, here $LA = \{LA_1 \text{ X } LA_2 \text{ X } \ldots \text{X } LA_n\}$. The local transition function is defined as $T_i \;\; LS_i \text{ X } LA_i \rightarrow LS_i$.

Definition (Model) Our model is represented as a *non-deterministic Buchi automaton* as a quintuple $MDL = (G, TR, I, F, v)$ where:

(1) $G \subseteq LS_1 \text{ X } LS_2 \text{ X } \ldots \text{X } LS_n$ is a finite set of global states of the system.

(2) $TR \subseteq G \text{ X } G$ is a transition relation defined by $(g, g) \epsilon TR$ if there exists a joint action $(a_1, a_2, \ldots, a_n) \epsilon LA$ such that $TR\,(g, a_1, \ldots, a_n) = g'$. a_i is called a joint action and is defined as a tuple of actions.

(3) *I* $\subseteq$ *G* is a set of initial global states of the system.

(4) *F* $\subseteq$ *G* is a set of final global states of the system.

(5) $V : AP \rightarrow 2^G$ is *the* valuation function where *AP* is a finite set of atomic propositions.

Then *MDL*, is a Deterministic Buchi Automaton (DBA) if and only if $\forall\ q \in GS\ \ and\ a \in i$ it holds that $|TR\ (q,\ a)| = 1$.

Having this formal representation of the system, allows easy implementation using the symbolic model checker, MCMAS [242]. The MCMAS tool is used for automatic verification of the correctness of the system expressed in Computation Tree Logic (CTL) [243].

Definition (Syntax). The CTL syntax is represented using the following grammar rules:

$\Phi ::= p \mid \neg\ \Phi \mid \Phi\ \vee\ \Phi \mid EX\ \Phi \mid EG\ \Phi \mid E\ (\Phi\ U\ \Phi)$ where the atomic proposition p ϵ AP; E is the existential quantifier on paths, and X, G, and U are path modal connective standing for "next", "globally", and "until", respectively. The Boolean connectives $\neg$ and $\vee$ are defined and read as "not", and "or" respectively.

Temporal properties:

The correctness of our system model can be checked using CTL by demonstrating the following significant properties:

(1) **Reachability property:** given a certain state, is there a computation sequences to reach that state from the initial state? The used reachability properties are defined as:

$$\Phi 1\ =\ EFDetect\_Task\_Abnormality \tag{51}$$

$$\Phi 2\ =\ EFChange\_Deployment \tag{52}$$

$$\Phi 3\ =\ EFStore\_QoS\_Logs \tag{53}$$

The formulas $\phi 1$, $\phi 2$, and $\phi 3$ check whether or not there exists a path to reach the *Detect_App_Abnormality* state, *Change_Deployment* state, and *Save_QoS_Logs* state respectively.

$$\Phi 4\ =\ E(\neg Analyze\_QoS\ U\ (Analyze\_QoS\ \wedge\ EF(Store\_QoS)) \tag{54}$$

The formula $\phi 4$ represents that there exists a path where the *Analyze_QoS* process will not start analyzing QoS data until the QoS data is collected.

(2) **Liveness property:** this property reflects that "*something good will eventually happen.*" For

example, in all paths globally if the System Analyze QoS detects an abnormality, then there is a path in its future through which the system will deploy the change for automatic reconfiguration thereby enhancing the quality of the orchestration.

$$\Phi 5 \; = \; AG(Detect\_Task\_Abnormality \rightarrow EF \; Change\_Deployment) \tag{55}$$

(3) **Safety property:** this property ensures that "*something bad never happens.*" An example of a bad situation is when the user does not correctly enter the required information to configure the system, but the latter initializes the monitoring cycle.

$$\Phi 6 \; = \; AG \; (\neg Config\_Monitoring \; (Correct\_Info) \; \rightarrow \; EF \; \neg Start\_Monitoring) \tag{56}$$

6.4 Experiments and Evaluation

In addition, to the above monitoring system validation using model checker, we describe in this section the experimental evaluation we conducted to assess our workflow monitoring model. Therefore, we evaluate the three adaptions schemes we proposed to dynamically reconfigure the workflow during its execution to respond to any cloud services performance degradation. We first, describe the environment set-up we configured and the key modules implemented to support monitoring and adaptation. We then depict the workflow we developed for evaluation purposes and the dataset we chose to execute our workflow. A set of scenarios were carefully chosen to evaluate workflow monitoring and the different adaptation schemes we implemented. Finally, we report and discuss the results we have obtained from the experimentations.

6.4.1 Environment Setup

Figure 6.5 describes the environment we established to execute, monitor, and dynamically adapt our workflow to respond to different performance degradation situations. In the following, we briefly describe each component of our experimentation configuration:

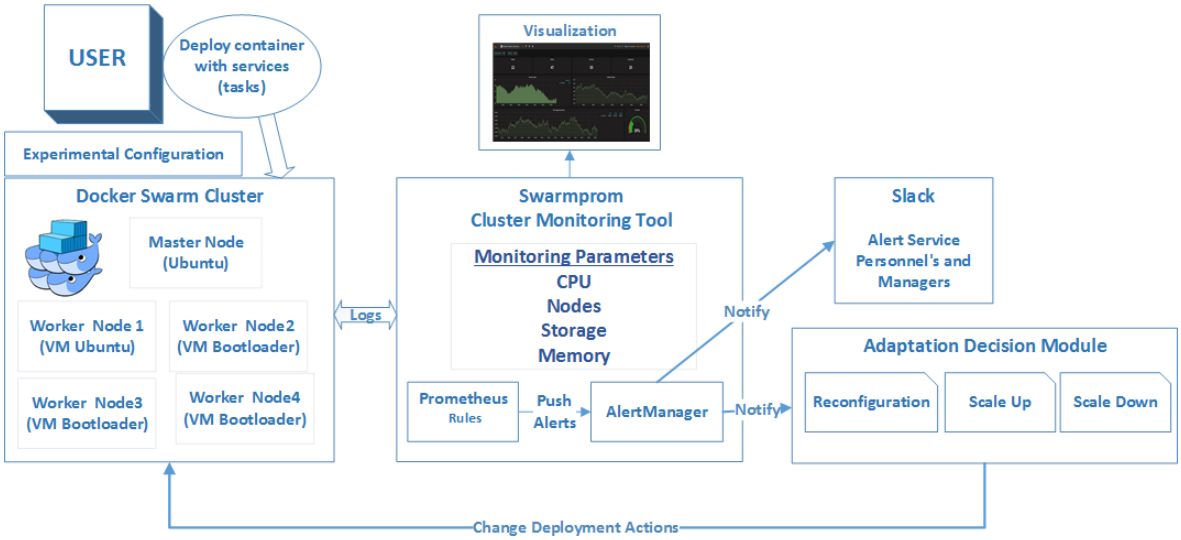

Figure 6.5: System implementation architecture.

Docker Swam Cluster. The Docker swarm cluster consisted of one master node and four worker nodes. We used Oracle Virtual Box driver to create the Docker nodes. These Swarm nodes can run any operating system and be managed on any cloud infrastructure. The workflow shown in Figure 6.6 is deployed on the Swarm cluster, and a Master node performs the orchestration and cluster management required to maintain the desired state of the swarm. Worker nodes receive and execute tasks dispatched from the manager/master node. To deploy an application to a swarm, a service definition is submitted to a manager node, and the manager node dispatches units of work, called tasks, to the worker nodes [244].

Swarmprom Cluster monitoring tool. This is a monitoring starter toolkit for Docker swarm services [245] equipped with *Prometheus*, *Grafana*, *cAdvisor*, *Node Exporter*, *Alert Manager*, and *Unsee*. These tools serve in providing continuous system performance measurements that are collected and analyzed by our monitoring system. Swarmprom Grafana [246] is configured with two dashboards and Prometheus [247] as the default data source. Monitoring parameters include CPU, memory, storage, and nodes, and Prometheus rules were used to monitor these parameters. Alert manager uses Slack, which is a cloud-based team collaboration tools and services. It brings team's communication together where conversations are organized and made accessible [248]. The Swarmprom Alert Manager can direct alerts through the Slack webhook APIs that is posted to the specific

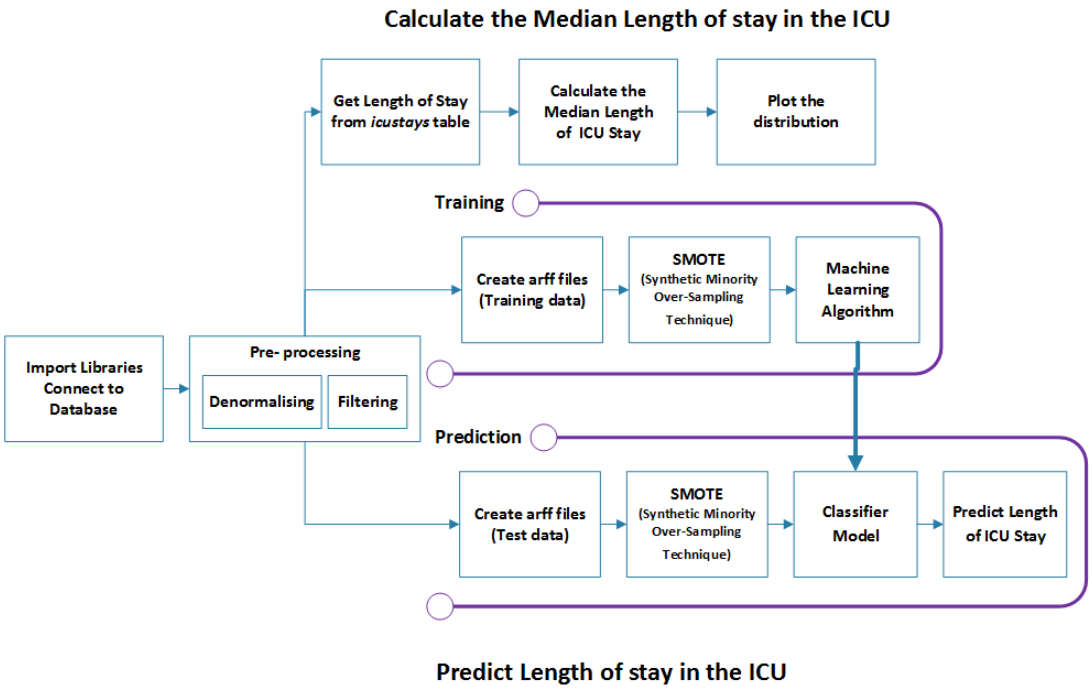

Figure 6.6: Health monitoring workflow description.

channels and alerts the concerned Managers and Service personnel who are on the move.

Adaptation Decision Module: This implements different reconfiguration decisions and is developed in the Perl language. An agent runs as a background process, which constantly monitors the CPU and memory status of the Docker services. Based on rules, the adaptation decision module inspects the Docker services and performs the necessary automatic reconfiguration of nodes in the cluster, such as scale up or scale down the services.

Visualization Module. This implements a dashboard to visualize in real-time monitoring information, including resource usage of both Swarm nodes and the services running on these nodes. It also integrates some visualization features, such as Zoom-in and out, and filtering. Graffana is an open source monitoring dashboard implemented with Docker.

6.4.2 Workflow and Dataset Description

In this section, we describe the dataset we used in our workflow as well as the workflow implementation and its composing tasks.

6.4.2.1 Dataset

The dataset we used to implement our workflow was retrieved from the MIMICIII database [249]. The dataset incorporates sixty thousand admissions of patients who stayed in critical care units Medical Center between 2001 and 2012. The database is available via PhysioNet, a web-based data resource that contains various physiological records. The available clinical information includes patient demographics, vital sign measurements, hospital admissions, laboratory tests, medications, fluid intake records, and out-of-hospital mortality.

6.4.2.2 Workflow Description

Figure 6.6 describes a health monitoring workflow we developed using the MIMICIII dataset to evaluate different aspects of an automatic reconfiguration workflow scheme we proposed in this chapter. The workflow is deployed on the Swarm cluster with PostgreSQL installed and the MIMICIII database tables loaded automatically [250] to perform the service tasks as outlined in the workflow. It consists of a set of tasks some of which are sequential and others parallel. The sequential tasks include retrieving data from the MIMICIII database and conducting data processing, while the parallel tasks include training and prediction tasks.

6.4.3 Cloud Workflow Adaptation Scenarios

We use the same workflow with different data sizes and processing complexity. Our baseline for comparison is workflow without adaptation or reconfiguration, measuring throughput, response time, CPU utilization, memory utilization, and execution time.

6.4.3.1 Scale-up (Client Gain)

In this scenario, we overload some nodes with extra processing tasks to affect the QoS of our workflow under investigation. We check the effect of our proposed framework including the monitoring and the automatic reconfigure modules on the QoS performance of the workflow. First the monitoring module will detect that the currently running tasks have lower performance due to overloading

of assigned nodes. Then, it forwards a message to the AR modules which in turn will issue a scale-up command message to the specific task at the assigned cluster (node). Scale-up will add more nodes to process the task, which will result in improving task performance.

6.4.3.2 Scale-down (Provider Gain)

Scaling down is performed when resources are not utilized in an optimized manner. This is done when the monitoring module detects low utilized nodes' CPU, which requires deletion of under loaded nodes from the cluster. In this scenario, we add an unnecessary number of nodes in the cluster handling the task and check the performance of the cluster before and after the scale-down.

6.4.3.3 Migration (Client and Provider Gain)

Workflow migration is usually needed if the cluster is overloaded with no extra resources available to be added to the cluster. In this scenario, we overload all the nodes of a cluster until they become slow in processing workflows as required, this will necessitate a migration of the workflow to a new data cluster. We observe the performance of the workflow and the cluster before and after the migration is performed.

6.4.4 Results and Discussion

In our experiments, we run the aforementioned workflow several times through which we use different dataset sizes and processing resource capacity. We apply our adaptation strategies to the workflow execution and compare the performance against a baseline scenario with no adaptation scheme, such as CPU utilization, memory usage, and trust scores. We run our monitoring system throughout the workflow execution.

Scenario 1: In this scenario, we evaluate the CPU utilization of a workflow among the nodes in the cluster. Figure 6.7 shows that CPU utilization increases as the workflow services are executed. However, the CPU utilization reaches significantly high values when the number of services increases. Thus, our monitoring system detects this issue and alerts the reconfiguration system which decides to add a new node and, accordingly, the load on the existing nodes is relaxed.

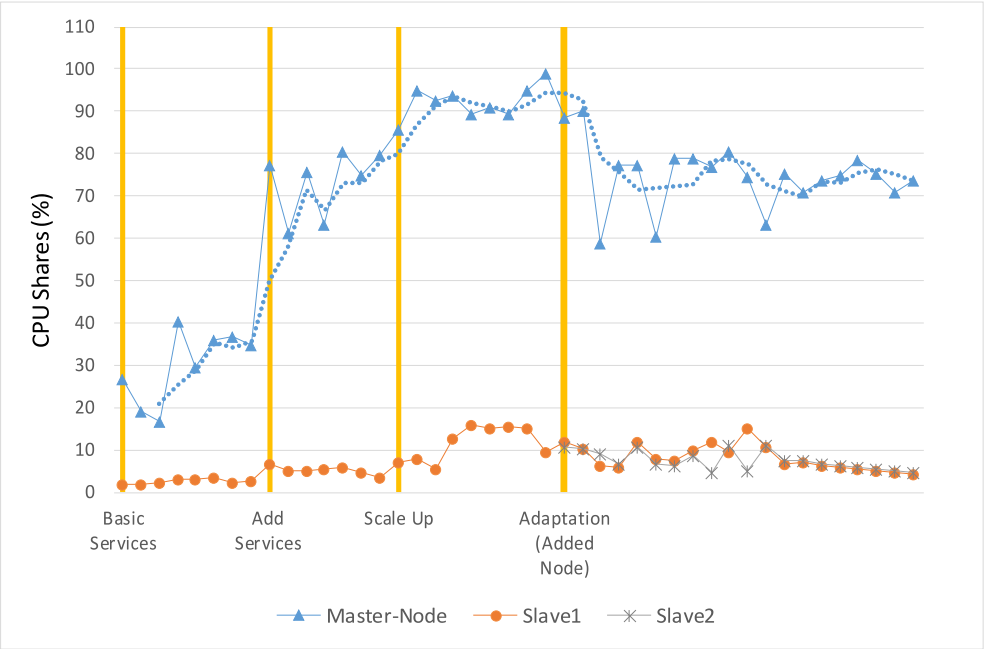

Figure 6.7: CPU utilization shares.

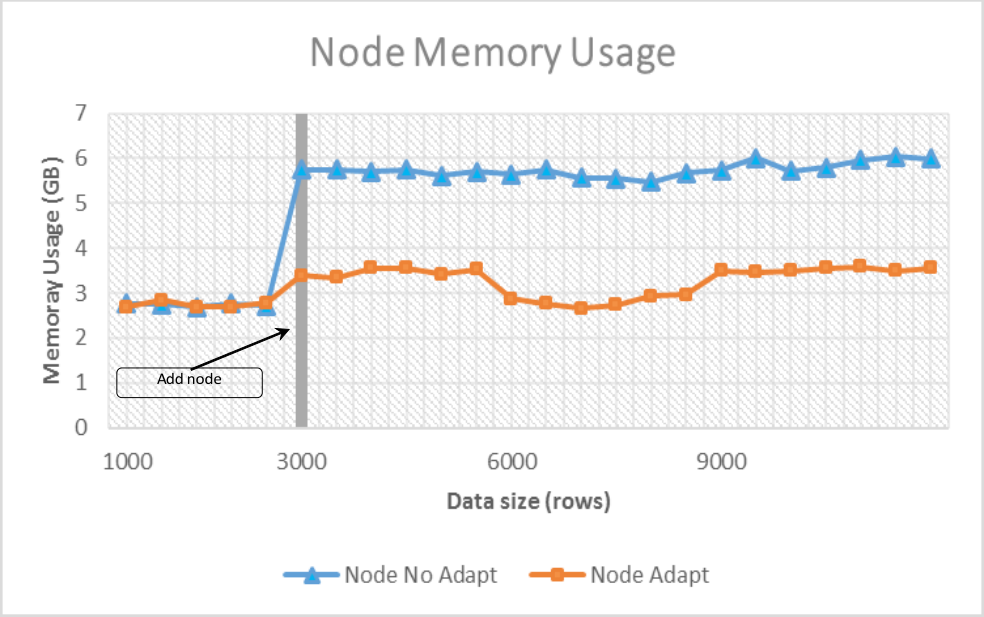

Figure 6.8: Node memory usage.

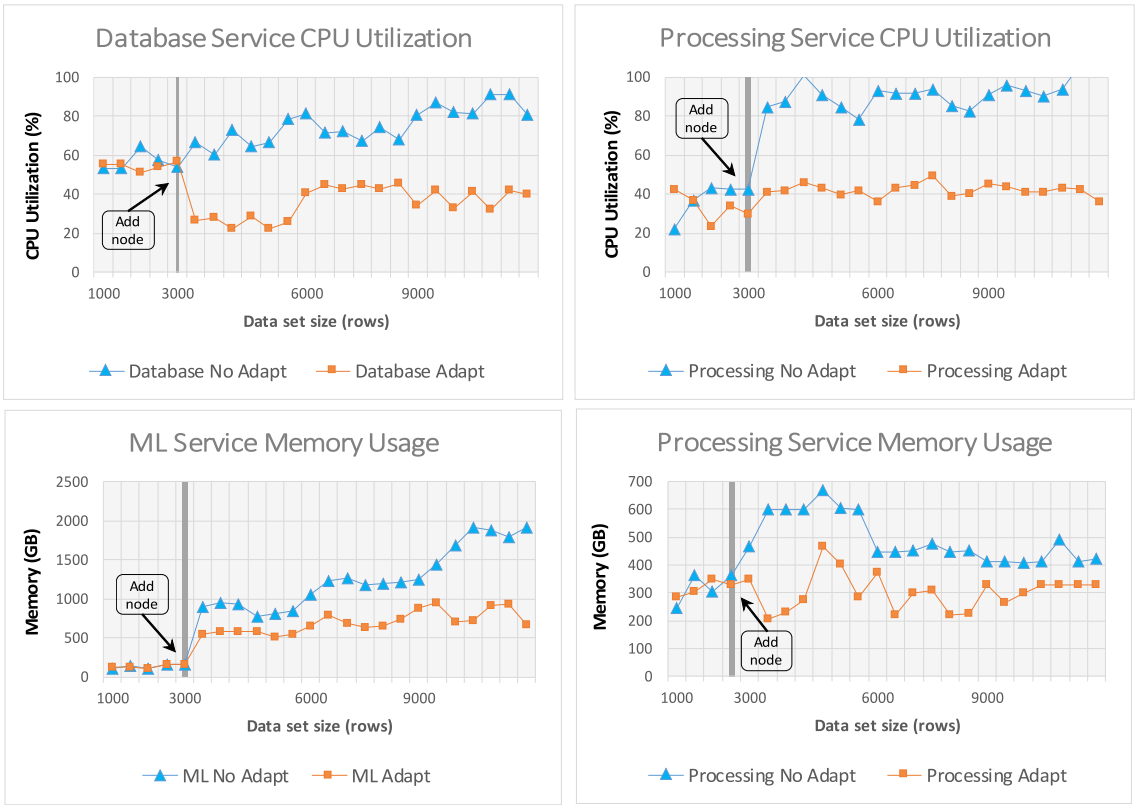

Figure 6.9: Service CPU utilization and memory usage.

Scenario 2: In this scenario, we evaluate the workflow memory usage for one of the nodes in the cluster. After adding a new node to the cluster resulting from an adaptation decision, the overall memory usage is significantly lower when compared to the usage in the case of no adaptation applied despite the increase in the size of the dataset as depicted in Figure 6.8.

Scenario 3: In this scenario, we monitor the CPU utilization and the memory usage of each task in the workflow. Whenever the CPU and memory performance is degraded, the reconfiguration system suggests adding resources to the cluster such as a new node in order to enhance the overall performance. Figure 6.9 shows some examples of tasks' memory usage and CPU utilization before and after adding a new node during which the dataset size increase overtime. The figure clearly shows the enhanced performance after adding an extra node.

Scenario 4: In this scenario, we compute different service trust scores for processing and database services. Figure 6.10 shows examples of service trust scores evaluated over time during which

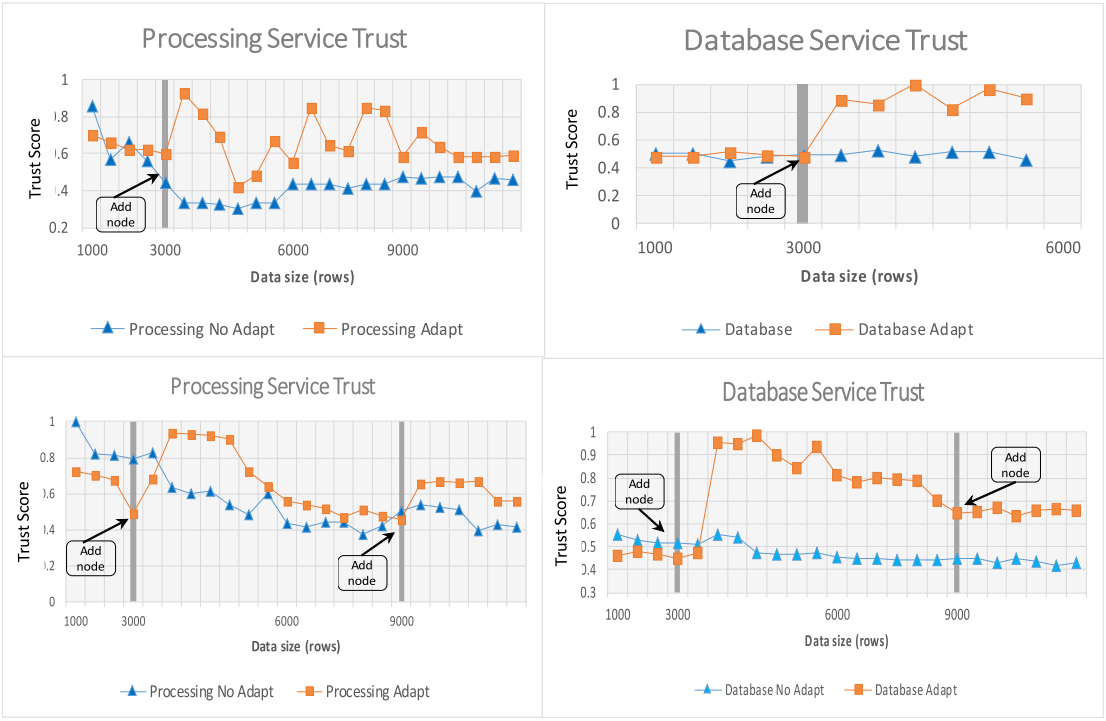

Figure 6.10: Service trust (1-step and 2-step adaptation).

the dataset size is increased. The trust score decreases as the data size increases till a threshold is reached and a new node is added to the cluster. The two upper figures of Figure 6.10 shows one step adaptation, and the lower two figures depict two-step adaptation. The more the data increases, the more nodes are required to process this data, and the trust scores increase after adaptation (i.e., adding extra nodes).

Scenario 5: In this scenario, we use scaled-down adaptation were we delete selected under loaded nodes when the CPU or memory utilization degrades. Figure 6.11 shows an example of a service resource utilization versus the number of nodes. We start at six nodes, at which we detect a low memory usage and CPU utilization per service. The system decides to delete two nodes which increases the utilization to an accepted level of about 25%. The figure also shows low Trust scores for some services and the overall workflow when we use an unnecessarily large number of nodes. The trust score increases when the utilization improves after adaptation (i.e., node deletion).

Scenario 6: In this scenario, we reduce the data size to reach low resource utilization. The monitoring system detects the low utilization quality violation and issues a node deletion adaptation

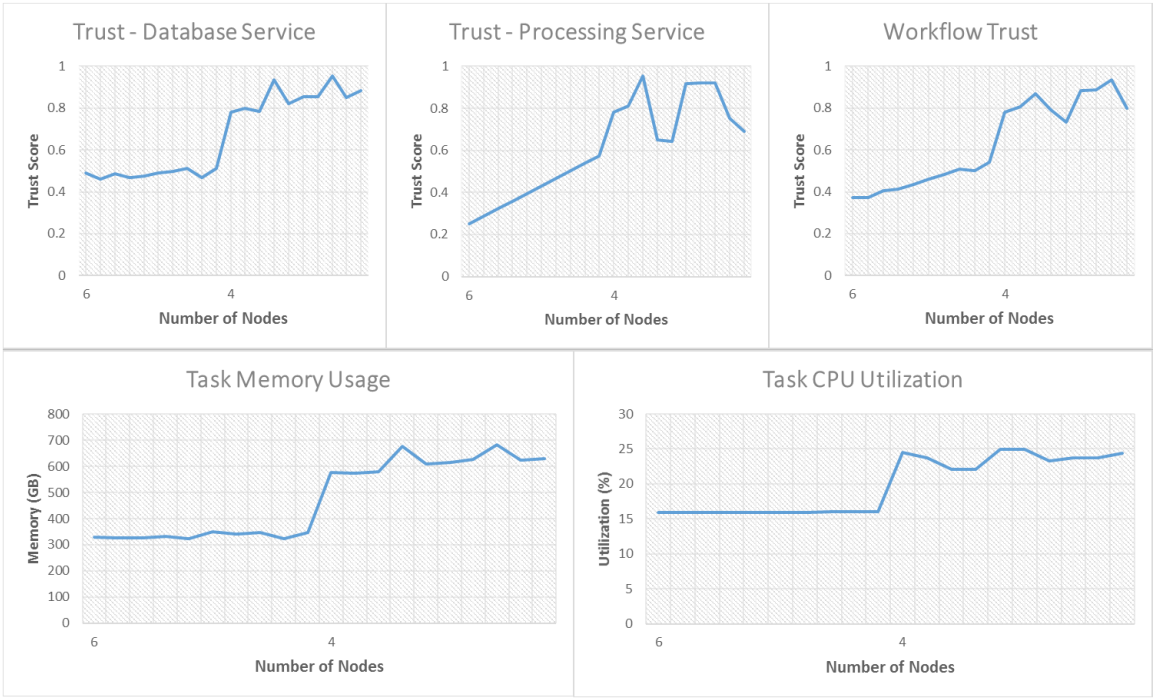

Figure 6.11: Scale down resources due to low utilization.

decision. Figure 6.12 shows that after a reduction of data size, memory usage and CPU utilization degrade and eventually the trust score decreases. After deleting the node, the trust increases again as the resource utilization improves.

Scenario 7: In this scenario, we perform a two-stage up-scale by adding a node at each stage. In the first stage, we use smaller dataset sizes, and we incremented it gradually. When the task CPU utilization and memory usage increase above a threshold, a new node is added to the cluster. In the second stage, we further gradually increase the dataset size until the monitored QoS attributes increase beyond the required threshold, and then another node is added. The results show an improvement of the performance after adding a node as shown in Figure 6.13. For some of the monitored services, the second stage adaptation does not reduce the CPU utilization but maintains a good performance level to compensate for the dataset size increase and prevents the service performance degradation. The figure also shows that our adaptation mechanism displays better QoS performance levels in comparison to the baseline of no adaptation service performance.

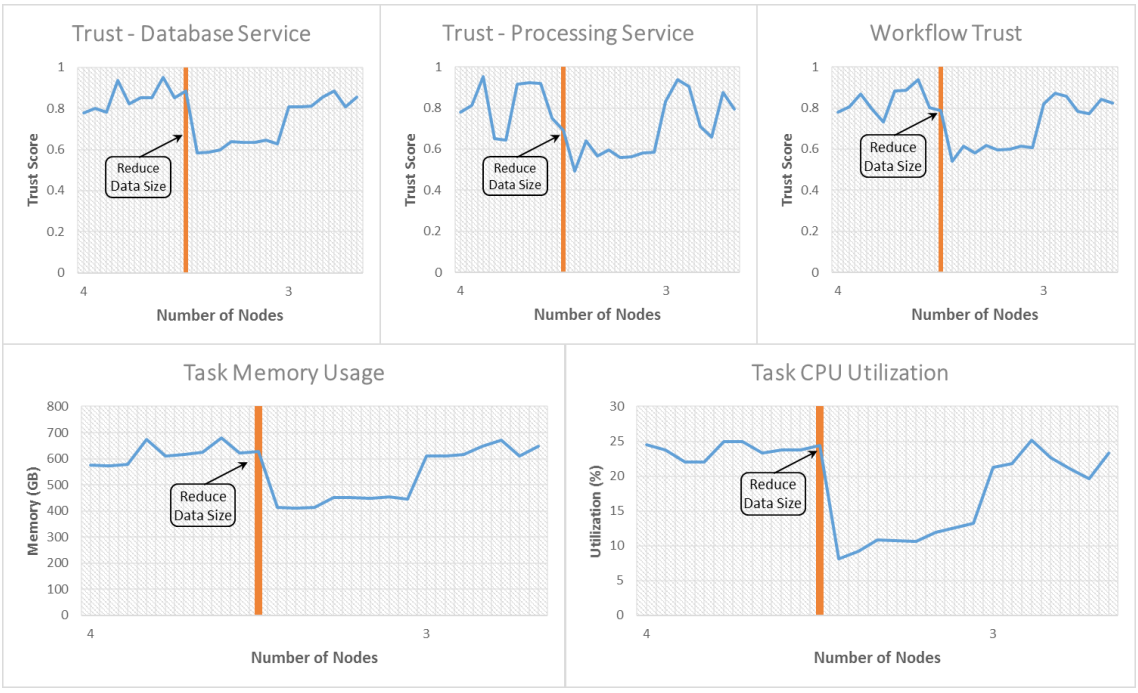

Figure 6.12: Scale down resources due to data size reduction.

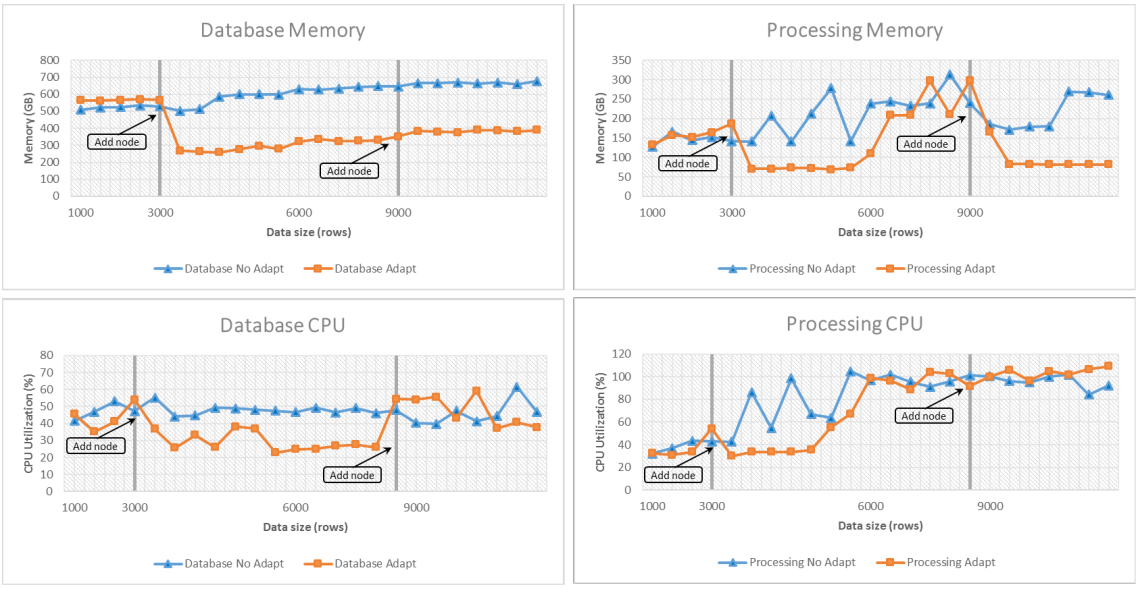

Figure 6.13: Two-stage resource upscale (node addition).

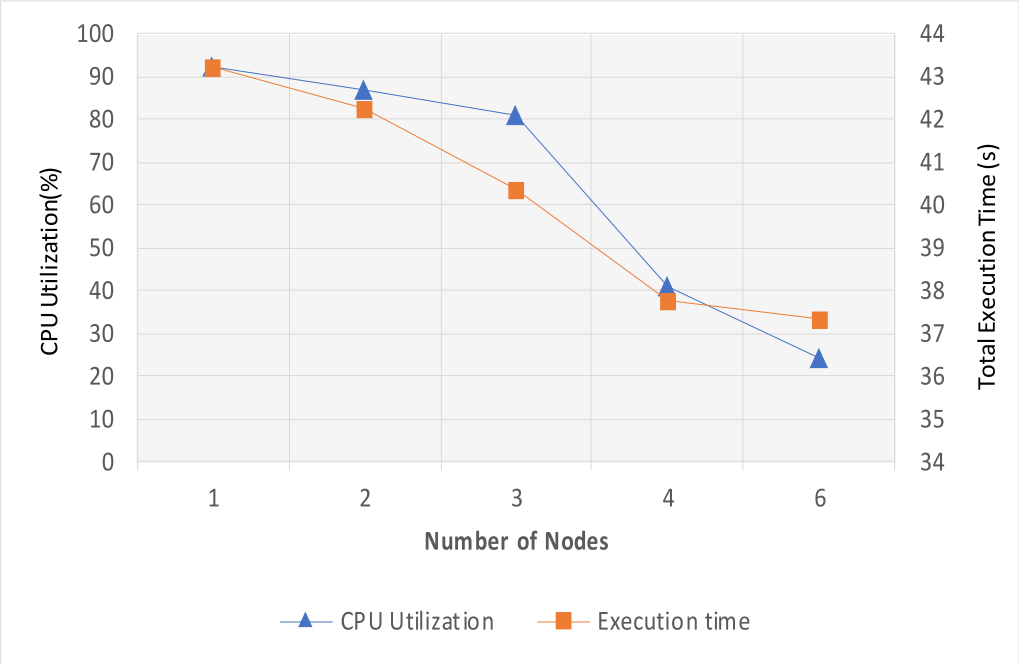

Figure 6.14: Total execution time.

Scenario 8: In this scenario, we perform multi-fold adaptation to optimize the total workflow execution time and CPU utilization. We monitor the aforementioned quality attributes and perform multiple node additions and adaptation actions until we reach the required quality level. Figure 6.14 shows a high CPU utilization level which triggers an adaptation action of adding a new node. However, the second monitoring cycle detected a quality violation and thus more nodes are added until we reach an adequate CPU Utilization. Adding nodes revealed an improvement of the total execution time as shown in Figure 6.14.

Scenario 9: In this scenario, we evaluate the migration adaptation decision. The currently used cloud cluster has limited resources and shows no possibility of further resource addition. Upon a quality degradation detection, in this case, CPU utilization, the reconfiguration manager reacts with a decision to migrate the workflow to another selected cluster offering more resources that can fulfill the requirements of the workflow under investigation. The results show an average of 11.5% improvement of the total workflow execution time and a significant enhancement of CPU utilization after migration for different sizes of the dataset as shown in Figure 6.15.

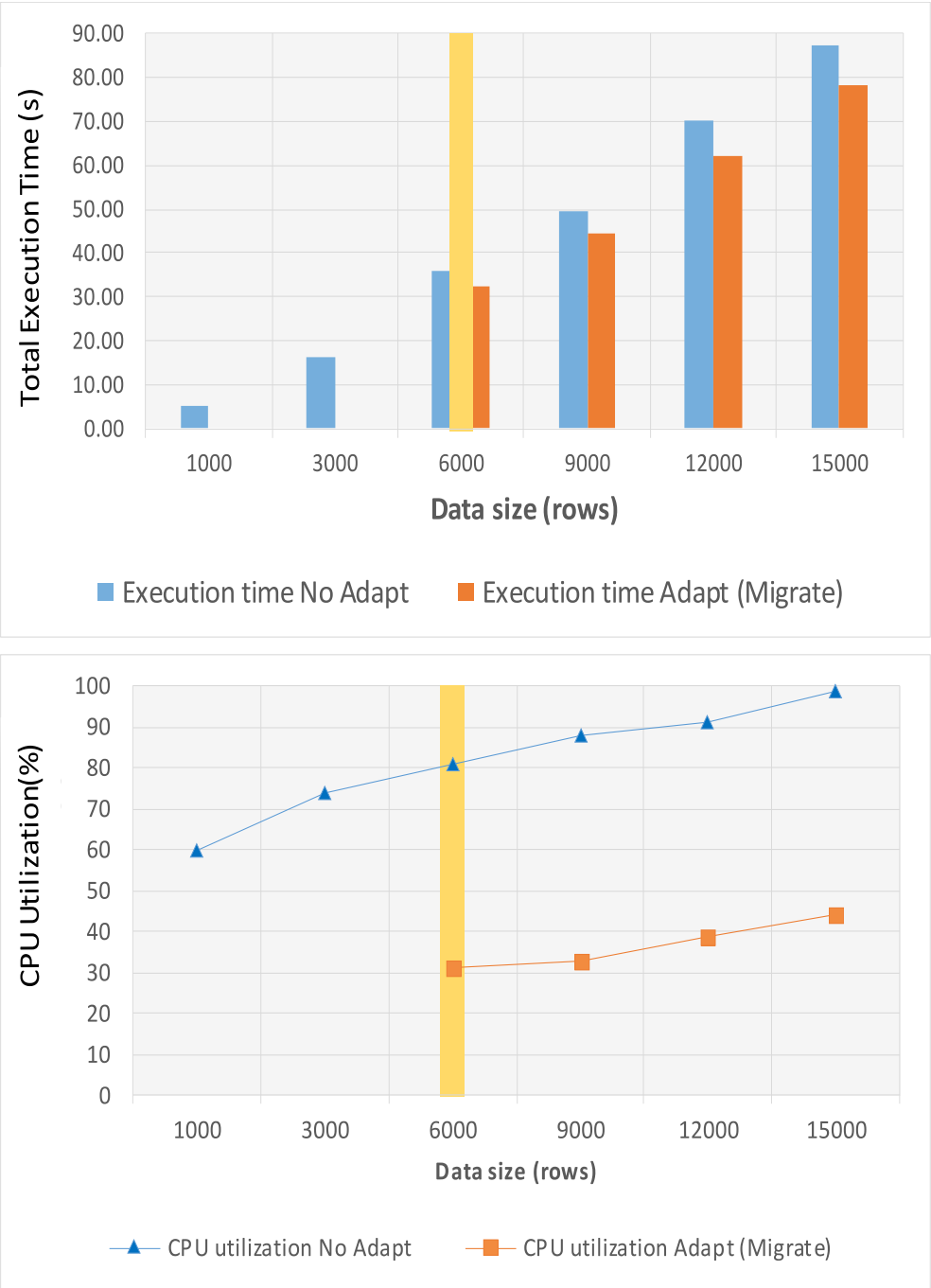

Figure 6.15: Total execution time and CPU utilization after migration.

6.4.5 Discussion

In this section, we discuss and evaluate our experimental results, which validated our monitoring and reconfiguration model by adopting the following strategies: 1) overload the system and monitor the workflow and cloud resources, and 2) underload the system and monitor the workflow and cloud resources. After that we test the reaction of the system and its effect on quality. Our objective is to keep the quality performance within the user's required ranges and the accepted trust scores.

Results show that our monitoring system detects the violation triggered when the quality attribute performance goes out the accepted or required range. This is reported to the automatic reconfiguration system which in turn issues the appropriate action to keep the required quality level.

In scenarios 1 through 4, we overload the system, monitored the CPU utilization, memory usage, and trust scores, and detected the quality violation. In all scenarios, the possible reconfiguration actions, such as adding new nodes at different stages, confirmed the improvement of the overall performance. In scenarios 5 through 6, we underload the system to detect lower resource utilization; then the reconfiguration manager would deallocate nodes as expected and accordingly improve the resource utilization.

We also tested the workflow migration and its effect on total time execution, and the results showed a significant improvement.

6.5 Conclusion

Provision of Cloud workflows QoS during execution necessitates monitoring and adaptation. The complexity of this process arises because of the dynamic nature of cloud resources and services, the variety of resource provisioning, and the variation of the workflow contexts and requirements. In this chapter, we proposed a trust-based model to support monitoring and adaptation of cloud workflows to guarantee a required level of QoS. This model handled the dynamic nature of cloud resources and services and coped with the complexity of workflow monitoring and adaptation. The proposed model supported workflow self-reconfiguration and self-adaption. Workflow reconfiguration is triggered to respond to performance violation detection after real-time monitoring of cloud

resources. To capture different dynamic properties of the workflow and the cloud execution environment, we formalized the cloud resource orchestration using a state machine and we validated it using model checker.

We conducted a series of experiments to evaluate our workflow monitoring, and adaptation using various monitoring and adaptation scenarios executed over a cloud cluster. The workflow is implemented and deployed over a Docker cluster. It fulfills a set of health monitoring processes and datasets where resource shortage is contingent to workflow performance degradation. The results we obtained from these experiments proved that our automated workflow orchestration model is self-adapting, self-configuring and reacts efficiently to various cloud environment changes and adapt accordingly while supporting a high level of workflow QoS.

In the next chapter, we will use the prediction of resource shortage to guarantee QoS prior to violation. This will strengthen our model to benefit from both real monitoring and prediction to proactively react efficiently to performance degradations and resource shortage.

Chapter 7

Towards a New Model for Cloud Workflow Monitoring, Adaptation, and Prediction

Intensive computing has enabled scientific applications such as genome analysis and weather prediction, which are usually composed of different tasks through a workflow system. Such workflows adhere to specific QoS requirements and other constraints such as time and cost. Moreover, these workflows share a number of cloud resources (e.g., VMs) that must be effectively allocated to fulfill user needs and requirements. There are ample tools developed for workflow planning that are based on different performance models. Others, developed workflow execution engines to manage runtime environment of workflows and record performance degradations and resource utilization and scarcity.

The workflow orchestration techniques are used to optimize the selection of the required cloud resources to satisfy the user's QoS needs. However, most current orchestration frameworks do not guarantee QoS during execution as a comprehensive workflow management system, which should support capturing user requirements and quality enforcement issues to enable automation of error prone workflow plans. In other words, maintaining the required level of QoS for a workflow is also an important feature of a workflow management framework, and this is supported by monitoring

and event capturing and analysis. Gathering monitoring logs and analyzing these logs will allow taking various actions to prevent errors or QoS violations that will cause performance degradation. Collected data will also help in predicting workflow resource utilization and react to a resource shortage before it causes workflow performance degradation.

In the previous chapter, we proposed a workflow orchestration model based on monitoring, adaptation and trust evaluation to detect QoS performance degradation and perform an automatic reconfiguration to guarantee QoS of the workflow. However, in this chapter, we extend this model to include the prediction of QoS performance and initiate necessary actions to avoid degradation of service performance accordingly. We propose a multi-model for workflow resource monitoring, resource prediction, and resource adaptations. Three adaptation strategies are proposed to capture changes in environment resources, categorize various violations and take the necessary actions to adapt resources according to workflow needs. This model relies on continuous monitoring of resource utilization combined with workflow resource prediction to apply different adaptation strategies. Such a strategy will capture and classify violations and respond with the appropriate actions to accommodate resources as needed according to user's preferences and application type. Performing actions, such as adding resources, will prevent performance degradation and help in maintaining and stabilizing the required QoS levels. The model also evaluates trust of workflow to support these adaptation schemes. Extreme adaptation is supported by continuously monitoring various workflow environment entities. Furthermore, we adopt the Autoregressive Integrated Moving Average (ARIMA) model to predict resource shortage and support adequate adaptation. Then, we use predicted values to generate trust scores for the workflow and its allocated cloud resources. Using trust helps customize and aggregate the different quality attribute measurements into one combined trust score. The ultimate objective is to achieve trustworthy workflow results and optimize task composition and allocated cloud resources with respect to the required QoS.

We implemented our model in a cloud environment and experimented with different adaptation scenarios. The results validated the effectiveness of our monitoring, prediction, and adaptation schemes in detecting violations and predicting cloud resource shortages accurately by taking the appropriate actions to deal with these violations.

The next section describes the development of the workflow monitoring and adaptation model

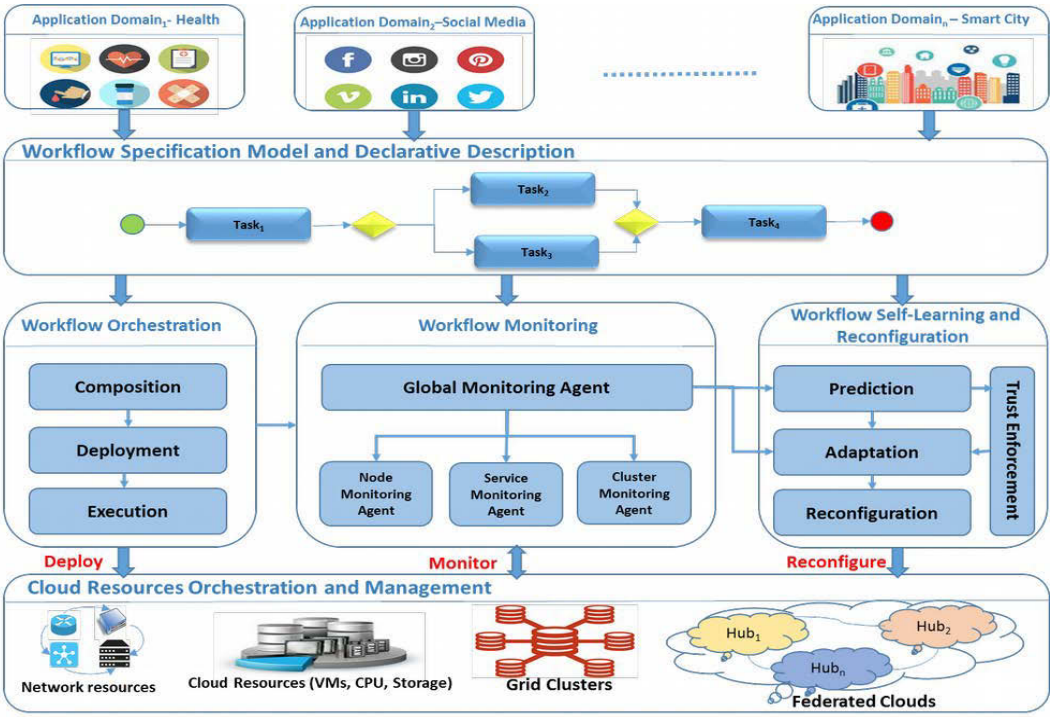

Figure 7.1: Monitoring, prediction, and adaptation system architecture.

to provide declarative workflow specification, monitoring model, and a self-learning and adaptation model. Section 7.2 depicts the development of the workflow prediction formulation based on ARIMA to model workflow resource usage. It also explains the trust evaluation formulation and trust score calculation. Section 7.3 defines the adaptation schemes and policies we developed in response to diverse types of violations and resource performance degradations. Section 7.4 details the implementations and experimentations conducted to evaluate the proposed monitoring, adaptation, and trust evaluation schemes. Finally, a conclusion section is introduced which points to interesting future research areas.

7.1 Cloud Workflow Monitoring and Adaptation Model

In this section, we describe our proposed workflow monitoring, prediction, adaptation, and trust evaluation model. Figure 7.1 depicts the main components of the model and their integration to fulfill key monitoring, prediction, trust enforcement, and adaptation features.

7.1.1 Declarative Cloud Workflow Representation

This component offers a generic abstractions and declarative techniques to simplify the design of complex workflow and enable a high-level and declarative analysis and manipulation of workflows. It supports constraints specification and verification (e.g., dead tasks), semantics, and patterns. In addition, it supports monitoring constraints states, instances states, and enforces correct instance execution and completion. Declarative languages for describing cloud resources and their relationship provide users and developers with the necessary constructs to describe elementary resources and their relationships (e.g., VMs, load balancer services, configuration parameters, and resource constraints). Examples of these languages include DSOL, COPE, and SPEEDL [251] [252] [129].

7.1.2 Cloud Workflow Monitoring Model

The automated monitoring and control of cloud services are still in the early stages [253] [254]. The monitoring model we propose relies on a multi-agent component that is responsible for conducting different measurements and collecting monitoring logs. These agents perform monitoring at different levels of abstractions within the cloud environment. For instance, from low granularity levels, such as at the task and service level, to higher abstraction levels, such as a node or even at the cluster level, agents collect various data types, such as CPU and memory utilization, storage occupation, and resource constraints violations. This data is analyzed in real-time modes and stored in a database for further analysis. In the following, we describe the common QoS properties of cloud services. We also depict the set of metrics used to measure these QoS properties. Table 7.1 describes a set of standard QoS properties used in cloud services as well the corresponding metrics to measure them. A more detailed metrics description is provided in Chapter 4.

7.1.3 Cloud Workflow Self-Learning and Adaptation Model

We are convinced that a monitoring and adaptation model must accommodate for flexible representation and planning of resource needs over time and the various phases of the workflow execution cycle. The adaption model we propose here comprises the three entities of prediction module, trust

Table 7.1: Key metrics for different application types.

Metric Type	Example Metric Description	App type
Throughput	Number of requests	Proxy server
	Number of read, write requests	Data storage
	Total number of queries	Processing
	Number of queries in progress	Processing
Utilization	Number of successful connection requests	Proxy server
	Number of active connections	Data storage
	Number of available connections	Data storage
	Available data storage size	Data storage
	Memory usage	Data storage
	RAM Utilization (e.g., file system cache)	Processing
Performance	Time to process each request (s)	Proxy server
	Scheme Runtime	Data storage
	Total time of queries processing	Processing
Error	Calculated accepts – handled	Proxy server
	Numbers of statements with errors	Data storage
	Count of connections refused	Data storage
	Number of rejected threads	Processing

score evaluation and enforcement, and adaptation modules. The prediction module uses machine learning algorithms to mine and analyze monitoring logs to predict, for instance, cloud resource shortages, VM performance degradation, unexpected spikes in demand, and detection of bottlenecks. Bottleneck detection is important for improving the quality of workflow, specifically the throughput quality attribute. Bottlenecks are the tasks that influence the overall performance of workflows. There are many bottleneck detection methods proposed in the literature which are long-term average based, such as total waiting time, average waiting time, and the average length of the queue. However, these methods are better for long-term bottleneck prediction rather than momentary bottlenecks [255]. In our model, bottleneck prediction is determined based on monitoring and predicting each task giving higher weight for the throughput quality attribute incorporated in the trust score. Accordingly, the trust score will indicate any predicted bottlenecks and the adaptation module can handle this situation using the appropriate actions.

The trust evaluation and enforcement module is in charge of calculating the trust scores for each task or service based on predictions and assumes the right enforcement decision. However, the adaptation module triggers adaptation events in response to trust deterioration and performance

degradation. Such adaption decisions include, for instance, node replacement, restart VM instances, task migration, and cluster reconfiguration.

7.2 Cloud Workflow Prediction Formulation

The amount of cloud resources used by workflows, such as CPU, memory, disk space, network, and the usage duration, can be predicted and modeled as a machine learning problem. The resource usage can be predicted based on historical data collected via system monitoring mechanisms. The previous observations can be used as training data to predict the expected future resource usage. Different machine learning algorithms including multi-parametric algorithms such as linear regression, polynomial regression, and non-parametric algorithms (e.g., KNN, and decision trees) can be used for predicting resource usage [256]. Other techniques use deep learning for sequence prediction such as Recurrent Neural Networks (RNN) and Gated Recurrent Unit (GRU) [257]. However, these techniques are expensive, exhibit high complexity, and require a lot of tuning, which is not suitable for our application type requiring lightweight algorithms that speed the evaluation process for earlier adaptation decisions and execution. However, the collected resource consumption parameters by our monitoring module are better modeled as a time series [258]. This is because the monitored values constitute a sequence of data points listed in time order as consecutive equally spaced points in time. We selected different time series techniques such as moving average, Autoregressive Moving Average (ARMA) and ARIMA. Consequently, we have conducted a series of experiments to compare them to select the best matching model to be used for predicting cloud resource QoS performance. The testing results showed that ARIMA performed better than the other tested techniques. The detailed measurements and results are reported in section 7.4.2. Therefore, we propose to use the ARIMA to model our collected resource usage data as detailed in the following sub-section.

7.2.1 Model Formulation Using ARIMA

Tasks composed in a workflow run independently and associated together in an ad hoc fashion. The orchestration environment manages these tasks and coordinates them to fulfill the workflow

execution objectives.

We model each workflow as a set of tasks $WF = \{t_1, t_2, \ldots, t_n\}$, where the number of tasks in the workflow is denoted by *n*. Each task is modeled as a tuple ***T*** (*type*, *input*, *profile*), where *type* is the type of task, for example, storage, network, or processing. *Input*$_i$ $\{I_1, I_2, \ldots, I_m\}$ is the set of *m* input parameters sizes for task T_i. The task is also mapped to a *profile* which describes the specification of the task including the quality attributes *Q* and their significance *W* with respect to the task. We denote $\boldsymbol{Q} = \{q_1, q_2, \ldots, q_j\}$, where *j* is the number of QoS attributes that describe the task performance and the weights for each quality attribute $\boldsymbol{W} = \{w_1, w_2, \ldots, w_j\}$ included in the *profile* which is generally assigned by the workflow administrator. At a time range/window Δt we store the QoS historical record monitored, which will be used for QoS prediction as depicted in the model we described in the previous section. Each quality attribute, such as CPU utilization, disk size utilization, and execution time, are timely and sequentially recorded form time series variables. Statistically, time series analysis is commonly handled using the ARIMA model, which is a generalization of an ARMA model. This model is appropriate for predicting and forecasting future values in time series related problem [259].

The ARIMA model is partially AR, which applies the automatic regression model on variables, and with the MA portion, which regresses the error terms occurred on previous time slots. However, 'I' stand for Integrated when the data shows non-stationary behavior. ARIMA models are represented as *ARIMA (p, d, q)*, where *p* is the number of time lags used for the auto regression part (number of parameters of AR), *d* is the differencing degree, and *q* is the moving-average order (number of parameters of MA). This model follows the Box-Jenkins methodology often used for the univariate time series [260]. This model is popularly used for time series analysis and forecasting [261].

In our monitoring system, performance history is stored for each task in a workflow. The related QoS attributes are chosen according to the task profile specification. Logs are collected for each environmental parameter as single observations recorded at regular time intervals. Past values of each parameter are collected in a database. The ARIMA model is then developed around each parameter separately to describe their values and underlying relationships, and predict their future values as univariate time series. Predicting environment parameters, such as CPU utilization, help

anticipates resource shortage and degradations in performance. In such a case, performing some precocious actions, such as adding resources, will prevent the performance degradation and help maintain and stabilize the required QoS levels.

ARIMA models the observed values of the series in a period as a linear element of the preceding time *t* and error terms as is thoroughly described in the following.

7.2.1.1 Autoregressive Component

In the stored monitoring logs, an observed value p_t depends on its own past values p_{t-1}, p_{t-2}, ..p_{t-n}. AR(m):

$$p_t = \beta_0 + \sum_{i=1}^{m} \beta_i P_{t-i} + \varepsilon, \tag{57}$$

where *m* is the number of previous observations, βi = [β_i, $i = 1, 2, 3, \ldots m$] is a vector of coefficients whose values are estimated from the previous observations data, and ε is the 'noise' which is a random variable having an independent normal distribution with a mean equals to zero and unknown constant standard deviation σ.

7.2.1.2 Moving Average Component

Here, p_t depends on the random error terms following a white noise process. For k past values:

$$p_t = \Phi_0 + \sum_{i=1}^{k} \Phi_i \varepsilon_{t-i}, \tag{58}$$

where *k* is the number of previous observations, Φi = [Φ_i, $i = 1, 2, 3, \ldots k$]

7.2.1.3 ARMA Component

For a time series of data p_t, and t is an integer number, ARMA(m,k):

$$\left(1 - \sum_{i=1}^{m} \beta_i L^i\right) p_t = \left(1 + \sum_{i=1}^{k} \Phi_i L^i\right) \varepsilon_t, \tag{59}$$

where *L* is the lag operator, the β_i are the AR, the Φ_i are the MA parameters and the ε_t are the independent error terms, which are identically distributed variables sampled from a normal distribution

with zero mean.

7.2.1.4 ARIMA (p, d, q)

This is an ARMA process for the differenced time series:

$$\left(1 - \sum_{i=1}^{m} \beta_i \, L^i\right) (1 - L)^d p_t - \beta_0 \;\; = \;\; \left(1 + \sum_{i=1}^{k} \Phi_i L^i\right) \, \varepsilon_t \tag{60}$$

$$y_t = \; (1 - L)^d p_t \tag{61}$$

where:

(1) p_t is the original nonstationary observation at time t.

(2) y_t is the observed differenced stationary output at time t.

(3) d is the integration order of the time series.

(4) ε_t is the error term at time t.

(5) m is the order of the last lagged variables.

(6) k is the order of the last lagged error.

(7) $\{\varepsilon_t\}$ is the time-series observations that are independent and identically distributed and follow a Gaussian distribution.

7.2.2 Prediction-based Trust Formulation

The ARIMA model is applied to each observed performance variable for each task to predict their expected performance values. Furthermore, we evaluate trust scores of a task modeled as a MADM. Each predicted performance value contributes towards a task trust score with a weighted percentage according to the task profile specification and the task input size. In other words, trust is measured as a weighted average of QoS performance according, for example, to input size, CPU utilization, storage, and execution time. We need to consider the task of input size variations and how it affects

the number of needed resources to satisfy the new circumstances. The trust is considered as an indicator that provides, in addition to monitoring, the task and prediction of its performance supporting the different adaptation actions due to trust scores regression. The trust score is evaluated according to the following formula:

$$TrustScore = \sum_{i=1}^{q} w_i \times \ p_i \tag{62}$$

where q is the number of quality attributes that contribute to the task trust score, p_i is the predicted quality attribute, and w_i is the weight of the quality attribute p_i towards the trust score.
Remedial action is taken according to the calculated trust when reaching a trust score under a threshold. The task profile contains a set of actions if the trust is decreased. Hence, if trust is under a certain threshold, then actions are taken according to the rules indicated in the profile. Subsequently, monitoring is resumed, and the cycle is repeated. Example of these rules: *if* the trust score of task *t* is below a threshold (e.g., 50%), then the CPU and memory resources needing to process this task must be increased.

7.3 Cloud Workflow Adaptation Schemes and Algorithms

In this section, we describe our adaptation scheme along with the adaptation algorithm of the adaptation module described above.

7.3.1 Adaptations Policies

Adaptation is usually triggered through invocation of actions, such as adding a new node, migrating services to another cluster, and adding storage resources. These actions run in response to events such as service usage increases beyond an upper bound, which initiate a cloud adapter to dynamically re-configure cloud resources.

Adaptation policies are currently classified in the literature into threshold-based policies and sequential decision policies [201]. The threshold-based policies use upper and lower constraints on certain required performance levels. Accordingly, specified resources are allocated or deallocated when the related quality performance measurements reach values outside of the adaptation specified threshold ranges. Writing scripts to describe these policies is fairly simple and, thus, are commonly

used. Although, it might be challenging to select the appropriate threshold values for complex workflows to find the best adaptive actions to improve the performance. Alternatively, the sequential decision policies models are based on Markovian Decision Processes (MDP) models and may be computed using, for example, reinforcement learning. Modeling adaptation schemes like MDP are widely used as it takes into consideration the inertia of the system. This is preferred when the costs for VM allocation and deallocation are fixed and, hence, changes to the number of VMs are not recommended unless the variation in the workload lasts enough time to necessitate making the change. Such situations are handled well using sequential decision making [201]. Nevertheless, in our model, we are handling Big Data workflow variation, which is characterized by its volume, variety, and velocity, and hence, worth taking immediate adaptation actions such as adding instant resources. Hence, we recommend using the threshold-based policies for adaptation.

7.3.2 Adaptation Algorithms

Existing research does not detail the quality attributes nor do they specify the input characteristics, which can be described as highly dynamic. The data-input-size values change all the time, which affects the performance of the task dramatically and eventually the overall workflow, especially if we face a resource shortage. This can be avoided using our prediction model.

Therefore, we implemented three adaptation approaches to cope with different levels of cloud resource limitations and service failures. The first adaptation scheme is executed to respond to nimble resource performance degradation. This adaptation is supported by predicting eventual resource limitations, node saturations to take the appropriate adaptation action that might aim, for instance, to maintain service provisioning at the required QoS level. The second adaptation, responds to severe and unexpected resource performance degradation or service failure. This adaptation relies on real-time monitoring data to detect severe performance violations or service failure which necessitate taking an immediate adaptation action. However, the third adaptation scheme adopts a hybrid model that combines data from monitoring reinforced with data from prediction to take a combination of short and long-term actions. This is done by checking the difference between the predicted value and the new monitoring logs. We can tolerate up to $\epsilon = 20\%$ difference which can be changed based on various factors, such as application domain and adaptation sensitivity. Such situations

Algorithm 8 Multi-strategy adaptation algorithm

Input:
$qosList$: List of QoS required attributes,
$validRanges$: QoS required value ranges,
$monitoringWindow$: Monitoring time window,
$hybridStrategy$: Flag
Output: $actionsList$: List of adaptation actions
procedure GENERATEADAPTATIONACTION($qosList$, $validRanges$, $monitoringWindow$, $hybridStrategy$)
    while $true$ **do**
        // Monitoring-based Strategy
        $stTime \leftarrow getTime() - monitoringWindow$
        $endTime \leftarrow getTime()$
        $monitoringLogs \leftarrow retreiveLogs(stTime, endTime, qosList)$
        for all $m \in monitoringLogs$ **do**
            if $(m \notin validRanges[m])$ **then**
                $detectedSevereDegradation \leftarrow true$
                $actionsList \leftarrow getActionsList(m, severe)$
            end if
        end for
        // Prediction-based Strategy
        $predictionResults \leftarrow applyARIMA(monitoringLogs)$
        for all $p \in predictionResults$ **do**
            if $(p \notin validRanges[p])$ **then**
                $detectedGracefulDegradation \leftarrow true$
                $actionsList \leftarrow getActionsList(p, graceful)$
            end if
        end for
        // Hybrid-based Strategy
        if $hybridStrategy$ = true **then**
            $stTime \leftarrow getTime() - monitoringWindow$ // new logs
            $endTime \leftarrow getTime()$
            $newLogs \leftarrow retreiveLogs(stTime, endTime, qosList)$
            for all $n \in newLogs$ **do**
                for all $p \in predictionResults$ **do**
                    if $\mid p - n \mid \geq \epsilon$ **then**
                        $adjustAdaptationAction \leftarrow true$
                        $actionsList \leftarrow getActionsList(actionsList, adjust)$
                    end if
                end for
            end for
        end if
        return $actionsList$
    end while
end procedure

involve reverting a short-term adaptation decision (e.g., task replication) or long-term adaptation decision (e.g., cluster reconfiguration) based on most recent monitored data.

The adaptation algorithm implements the three adaptation schemes as described above. Algorithm 8 depicts the adaptation algorithm components. This algorithm starts by retrieving the monitoring logs that include, for instance, CPU and memory usage and node saturation, along with the projected QoS ranges. Then, it decides on the adaptation strategy to be adopted, which remains contingent upon how severe are the detected violations. Finally, it performs the necessary adaptation actions to secure enough processing resources (e.g., scale up by adding new VM). The algorithm adopts the ARIMA prediction model that captures environment resource status and predicts, for instance, resources shortage, node saturation, and future demand. If the hybrid mode is enabled, then the algorithm will compare the new retrieved logs with the predicted values to account for prediction inaccuracies and revert any unnecessary adaptation actions. Finally, the algorithm loops back to a continuous monitoring and prediction state, after adaptation actions have taken place.

7.4 Experiments and Evaluation

In this section, we describe the dataset and workflow used in our experiments. We next describe the set of scenarios we conducted to evaluate the different adaptation schemes we proposed. Finally, we report and discuss the results and findings.

7.4.1 Dataset and Workflow Description

We used the same MIMICIII dataset adopted in the previous chapter's experiments [249] as well as the same health monitoring workflow developed in previous experiments to evaluate different adaptation schemes that we have proposed in this research. The workflow is deployed on the Swarm cluster with PostgreSQL installed and the MIMICIII database tables loaded automatically. It consists of a set of sequential as well as parallel tasks. Sequential tasks included, for instance, are data retrieved from the MIMICIII database and processing. However, parallel tasks included training and prediction tasks.

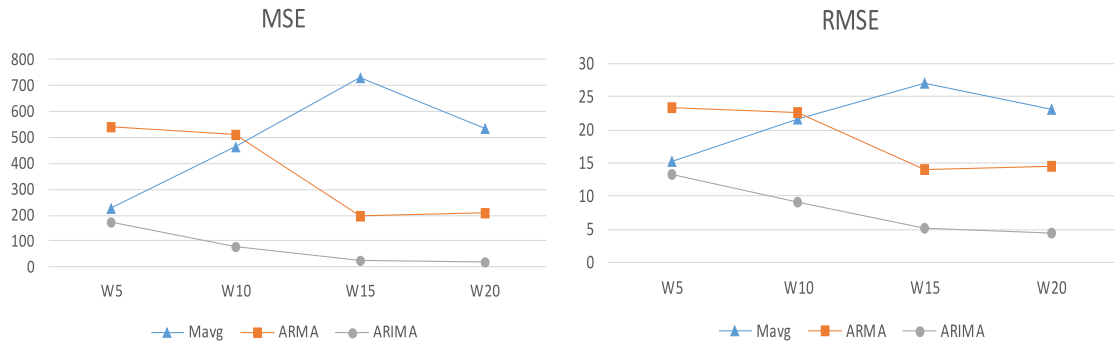

Figure 7.2: MSE and RMSE tests.

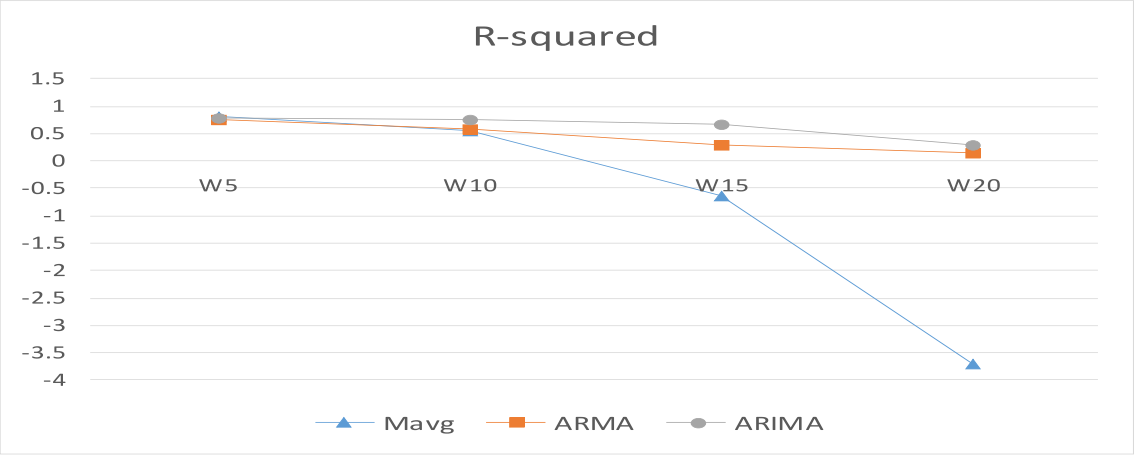

Figure 7.3: R-squared test.

7.4.2 Time Series Prediction Models Comparison

Prediction performance of the time series models is commonly measured using different methods such as MSE, RMSE, an Absolute Fraction of Variance (R^2), and error percentage. The following are the definitions and formulas for these measurements, respectively:

$$MSE = \frac{\sum_{t=1}^{n} (y_{pt} - y_t)^2}{n}, \tag{63}$$

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (y_{pt} - y_t)^2}{n}}, \tag{64}$$

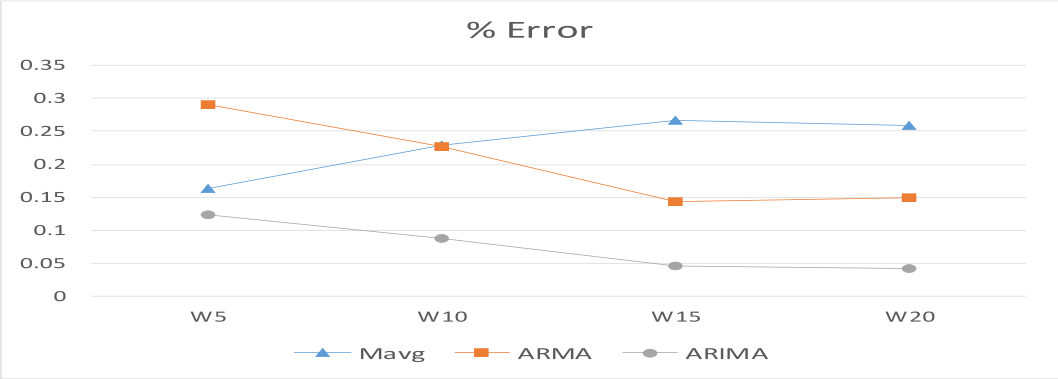

Figure 7.4: Percentage of error test.

$$R^2 = 1 - \frac{VAR_{t=1}^{n}\left(y_{pt} - y_t\right)}{VAR_{t=1}^{n}(y_{pt})}, \tag{65}$$

$$\%Error = \frac{\sum_{t=1}^{n}\left(y_{pt} - y_t\right) \ / \ y_t}{n}, \tag{66}$$

where y_{pt} is the predicted value at time t, y_t is the observed value at time t, and n is the number of observations. Clearly, the best score for the R^2 measure is 1 and for other measures is zero [262]. We performed the test for prediction windows sizes of 5, 10, 15, and 20 respectively. The results depicted in Figure 7.2, Figure 7.3, and Figure 7.4 show that the ARIMA model performs better than the other prediction models of, the Moving Average and ARMA.

7.4.3 Precision and Recall Measurement to Evaluate Our Prediction Model

In some cases, accuracy is not enough and does not provide a good measure for assessing the model performance. Hence, we used precision and recall to further qualify our prediction model. Precision can be called positive predictive value which is the fraction of the correct predicted values with respect to the total number of predicted values, while recall is the sensitivity which is calculated as the fraction of correctly predicted values the number of results that should have been correctly predicted [263]. These measures emphasis on the positive predictions examples.

In our model we define recall as the number of violations correctly identifies over the total number of correctly identified violations and violations incorrectly labeled as non-violations. We also define precision as the number of correctly identified violations over the total number of the

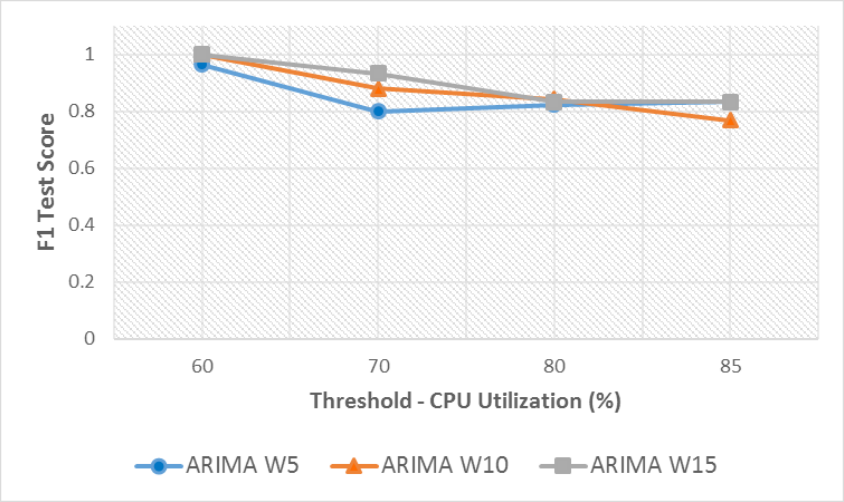

Figure 7.5: F1 test.

correctly identified violations and the non-violations incorrectly identified as violations. This is shown in the following formulas.

$$\begin{aligned} recall &= \frac{true\ positives}{true\ positives + false\ negatives} \\ &= \frac{violations\ correctly\ identified}{violations\ correctly\ identified + violations\ incorrectly\ labeled\ as\ nonviolations} \end{aligned} \tag{67}$$

$$\begin{aligned} percision &= \frac{true\ positives}{true\ positives + false\ positives} \\ &= \frac{violations\ correctly\ identified}{violations\ correctly\ identified + non{-}violations\ incorrectly\ identified\ as\ violations} \end{aligned} \tag{68}$$

We have applied these measurements on our dataset to further validate our model. We applied ARIMA model using different prediction window sizes and tested the predicted values against the actual measured values. After that we applied a threshold value defined in the profile database to identify QoS violations on both the predicted and the actual QoS measured values of one of the quality attributes. This procedure is performed to identify the number of true positives, true negatives, false positives, and false negatives, so that we can calculate the recall and precision measurements. However, in order to find an optimal combination of precision and recall we use the

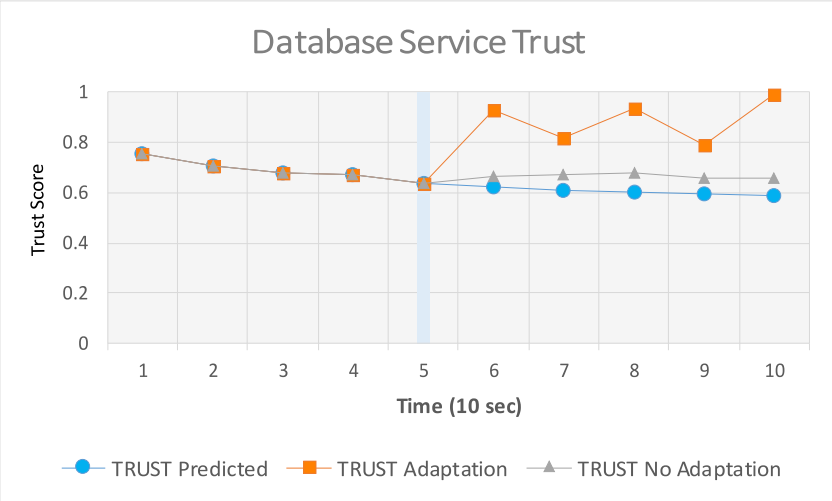

Figure 7.6: Task trust evaluation with and without adaptation.

F1 score. F1 score is defined as the harmonic mean of precision and recall. The aforementioned metrics are combined in the following equation:

$$F1 = 2 \times \frac{percision \times recall}{percision + recall} \tag{69}$$

The harmonic mean is useful to reduce the effect of the extreme values. Other metrics can be used to combine precision and recall such as geometric mean. However, F1 metric is the most commonly used, being maximized satisfies optimal balanced model for precision and recall. We applied the test on different threshold values determining QoS violations. Figure 7.5 depicts our test results which shows that our algorithm yields a good recall, precision scores, as well as F1 score with an average of 86.5% for all different threshold values and different prediction window sizes.

7.4.4 Scenario Description

In our experiments, we executed the same workflow but applied different data sizes and processing complexity. We compare our adaptation strategies to a baseline execution, which is without adaptation or reconfiguration in terms of CPU utilization, memory utilization, and trust score evaluation. In each experiment, we executed our workflow with no adaption actions taken, with adaptation based

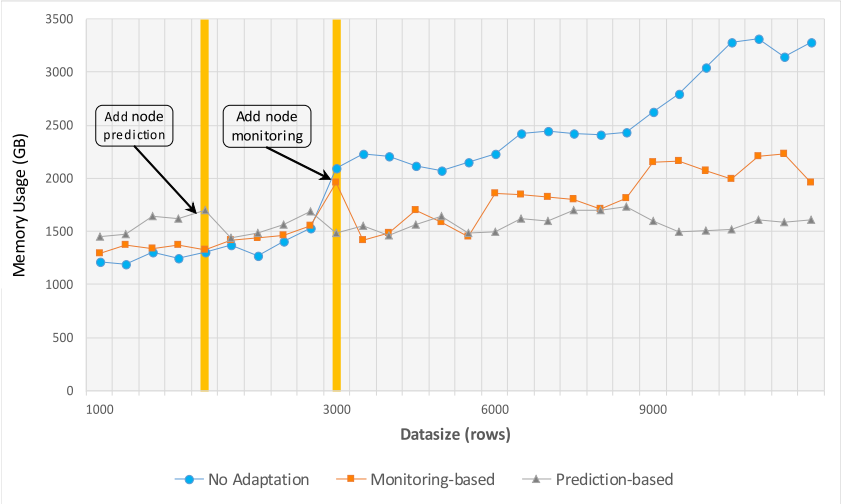

Figure 7.7: Memory usage over time.

on performance and resource monitoring logs, and adaptation based on a prediction of resource utilization.

Scenario 1: In this scenario, we evaluated the trust score for processing and database services over a period of time. Figure 7.6 shows that service trust scores significantly increased after performing an adaptation action of adding a new node to the cluster, whereas the trust decreases when no adaptation actions are taken. Moreover, the trust scores were evaluated based on the predicted quality attribute values are close to the actual trust proves that our prediction algorithm performed well and indeed, led to higher accuracy.

Scenario 2: In this scenario, we evaluate the total memory usage based on the three strategies: without adaptation, adaptation based on monitoring, and adaptation based on prediction. As shown in Figure 7.7, memory usage per node increases when data size increases. However, after adding a new node following an adaption decision based on prediction or monitoring, the memory usage per node decreases. This could be explained on the basis that workflow execution is balanced among all the available nodes, which helps avoid memory saturation.

Scenario 3: We evaluated the CPU utilization for the database service throughout three adaptation experiments: without adaptation, adaptation based on monitoring, and adaptation based on prediction. As shown in Figure 7.8, the CPU utilization increases while the data size increases until a

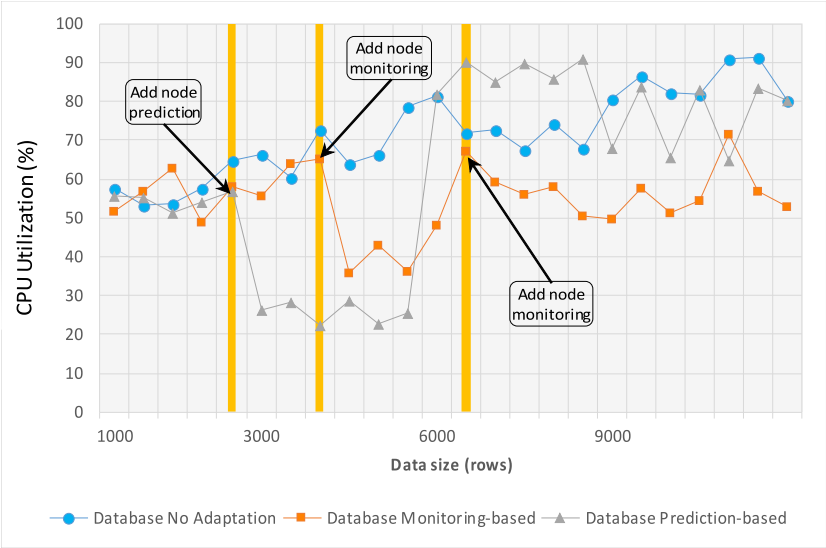

Figure 7.8: CPU utilization per service (e.g. Database).

certain threshold is reached. Based on predicting the violation of the CPU utilization-threshold, adding a node to respond to an adaptation action triggers the decrease in the CPU utilization to an acceptable range that satisfies QoS requirements. Alternatively, the adaptation action is performed only when the monitoring agent detects a CPU utilization violation when a certain threshold is reached. Hence, the CPU utilization drops down within the specified valid range, avoiding CPU overload, and eventually causing performance degradation.

Scenario 4: In this scenario, we evaluated the ARIMA model and assessed its prediction accuracy. Figure 7.9 shows the results of applying the stationary test and ARIMA model on two datasets, namely, CPU utilization and memory usage. The goodness-of-fit test proved the validity of the model for both datasets. After examining the model's residuals time series, we conclude that the residuals' *distribution* has a mean of zero, as the population mean (AVG) test is not significantly different from the target of zero. In addition, the population standard deviation (STDEV) test shown in Figure 7.9 confirms that our sample dataset standard deviation is close to the target. The population skew value indicates that the residuals distribution is symmetrical. We can also conclude from the population excess kurtosis test that the residuals distribution tails are normal. Hence, the residuals distribution follows a Gaussian distribution. Thus, the ARIMA model assumption is met, and the model can be considered fair.

Stationary Test					
Test	Score	P-Value	C.V.	Stationary?	5.0%
ADF					
No Const	1.0	90.4%	-2.1	FALSE	
Const-Only	-2.7	14.1%	-3.4	FALSE	
Const + Trend	-0.5	32.2%	-1.6	FALSE	
Const+Trend+Trend^2	-4.0	0.0%	-1.6	TRUE	

Stationary Test					
Test	Stat	P-Value	C.V.	Stationary?	5.0%
ADF					
No Const	1.4	95.3%	-2.0	FALSE	
Const-Only	-1.4	59.5%	-3.4	FALSE	
Const + Trend	0.4	99.6%	-4.1	FALSE	
Const+Trend+Trend^2	-3.2	0.1%	-1.6	TRUE	

ARIMA(1,1,1)			Goodness-of-fit		
	Param	Value	LLF	AIC	CHECK
	μ	1.90	-84.30	178.60	1
	ϕ_1	0.00			
	θ_1	0.00			
	σ	13.40			
	d	1			

ARIMA(1,1,1)			Goodness-of-fit		
	Param	Value	LLF	AIC	CHECK
	μ	0.00	-101.22	212.44	1
	ϕ_1	0.19			
	θ_1	-0.16			
	σ	19.00			
	d	2			

	Residuals (standardized) Analysis						
	AVG	STDEV	Skew	Kurtosis	Noise?	Normal?	ARCH?
	0.00	1.02	-0.12	-0.44	TRUE	TRUE	FALSE
Target	0.00	1.00	0.00	0.00			
SIG?	FALSE	FALSE	FALSE	FALSE			

	Residuals (standardized) Analysis						
	AVG	STDEV	Skew	Kurtosis	Noise?	Normal?	ARCH?
	0.05	1.24	-0.59	0.95	FALSE	TRUE	FALSE
Target	0.00	1.00	0.00	0.00			
SIG?	FALSE	FALSE	FALSE	FALSE			

Figure 7.9: Memory usage and CPU utilization prediction model (ARIMA).

7.4.5 Discussion

Our experiments satisfied two main objectives. First, evaluating the performance of our framework when applying different adaptation strategies (the monitoring-based and prediction-based adaptation strategies) as opposed to a non-adaptive strategy, and second, evaluating the ARIMA prediction model and how suitable it is to our dataset.

For the first objective, we monitored the service trust score and the performance of two important quality properties, memory usage, and CPU utilization. Service trust scores and both observed quality properties showed better values when using prediction based adaptation strategy in comparison to the monitoring-based adaptation strategy, which in turn, provides better results over the non-adaptive strategy. This is because perdition-based strategy performs the adaptive actions before the actual occurrence of QoS violation and, thus, maintains better quality performance. Monitoring-based adaptation strategy acts upon quality degradation events and, thus, the corrective actions are performed to return the performance to the required levels, and the overall performance will still be better than the non-adaptive strategy.

Our second objective is to test if the ARIMA model is appropriately used for prediction using

our dataset. The results show that the model has high prediction accuracy and can be applied for prediction of future quality performance values.

7.5 Conclusion

Supporting execution, monitoring, and adaptation of complex workflows over cloud environments is complex. In this research, we propose a multi-model approach that copes with these challenges and capture the dynamic aspect of cloud resources and services as well as the complexity of workflow monitoring and adaptation. The proposed model supports declarative workflow specification, monitoring, self-learning, and adaptation. The workflow monitoring model supported severe workflow adaptation and relied on mobile agents to continuously monitor different entities, i.e., from low-level task execution and node execution to high-level cluster monitoring. However, the workflow performance prediction based ARIMA was used to predict resource shortage and support agile workflow adaptation. We also evaluated workflow trust based on QoS to support the different adaptations schemes.

This research implemented workflow monitoring, prediction, and adaptation using different scenarios in a cloud environment. The generated results validated the effectiveness of our monitoring, prediction, and adaptation schemes in monitoring various cloud services and resources and in detecting violations, predicting accurately the cloud resource shortage, and in taking the appropriate adaptation actions in the face of various resource violations. The research results are of great importance to researchers and professionals interested in this area. In the following phase of this project, we are planning to experiment with other prediction techniques, such as reinforcement learning, and test our adaptation schemes on a larger cloud setup.

Chapter 8

Conclusion

The growing demand for Cloud Computing led to the rise of many research and development issues. One of the most important issues is the enforcement of the required end-to-end QoS levels since it is necessary for cloud users to guarantee the quality of Big Data workflows throughout their lifecycle. Achieving this goal requires handling many challenges, involving:

- The diversity of resources and services, which make the selection of the most suitable cloud provider a tricky process.
- The dynamic nature of the Cloud Computing environment and the high complexity of the workflows to be processed make the task of guaranteeing the QoS during execution very challenging.

8.1 Summary of Research Contributions

This dissertation focuses on these above-mentioned challenges. To this end, we studied four facets to achieve the goal of maintaining the end-to-end quality of service for workflows deployed and executed over federated clouds. We summarize the most significant contributions of this dissertation:

- We proposed an end-to-end quality specification for Big Data workflows. Our model provided a multi-dimensional Big Data quality assessment specification that combined both data-driven and process-driven quality implemented at different granularity levels including data, task,

successive tasks trust, and overall workflow trust. The model specified data quality and processing quality for each task in the workflow and used the resulting quality specifications to describe overall quality-based trust of all successive tasks and the workflow.

- The end-to-end quality and trust specification model are used to, first, enable CSP quality evaluation and facilitating CSP ranking, and, thus, helping users in CSP selection decisions. Second, maintain the quality of workflows orchestration at runtime through specifying quality at different granularity levels starting at the task and aggregating successive tasks quality to build a complete workflow quality model.

- We proposed a new cloud service provider selection model for Big Data workflows based on trust valuations. It is a comprehensive, de-centralized, and multi-dimensional trust evaluation model to guarantee QoCS by implementing three strategies that relying on 1) the provider's advertised QoCS, 2) the user's past experience with the cloud provider, and 3) the neighboring assessments evaluated based on user preferences including the significance of each quality attribute. Additionally, to appropriately handle malicious trust scores from neighbors, we provided a community management system that enforced a set of member's engagements and participation rules. Our solution uses a weighted average of three cloud selection strategies to guarantee reliable trust evaluation.

    - We modeled the cloud selection problem as MADM relying on three trust scoring schemes, SAW, WPM, and TOPSIS. In addition, the model captured Big Data key characteristics and coped with key features including flexibility, heterogeneity, and scalability of the cloud environment. Our proposed model captured users' requirements and efficiently evaluated trust of cloud providers.

    - Experimental results proved that our CWTS algorithm selects the cloud that best matches the user's QoCS preferred requirements. It also exhibited a low communication overhead generated for the sake of conducting cloud service selection. Moreover, a CloudSim extension was implemented to simulate multiple clouds and enable more extensive experiments with our full framework.

- Our model adopted Multiple Linear Regression (MLR) to formally model the trust evaluation problem in competing cloud environment to support trust score prediction. Conducted experiments demonstrated that the model can perfectly be used to predict the response variable trust.

- We proposed a model of trust enforcement on cloud workflow service composition and orchestration for QoS guarantee. Our model depicted a workflow orchestration, monitoring, and adaptation scheme that relied on trust evaluation to identify QoS performance degradation and performed an automatic reconfiguration to guarantee QoS of the workflow. The proposed model supported self-reconfiguration and self-adaption of cloud workflow. Workflow reconfiguration is triggered to respond to performance violation detection after real-time monitoring of cloud resources. We formalize the cloud resource orchestration using a state machine that efficiently captures different dynamic properties of the cloud execution environment. We also validated our model in terms of reachability, liveness, and safety properties using a model checker. We adopted three adaptation strategies to capture changes in environment resources, categorize various violations, and take the necessary actions to adapt resources according to workflow needs.

- We executed several experiments over cloud clusters to evaluate our workflow monitoring and adaptation using different scenarios. We used a scientific Big Data workflow implemented and deployed over a Docker cluster. The results proved that our automated workflow orchestration model is self-adapting and self-configuring. It adapts efficiently to changes while supporting the required level of workflow QoS.

- To predict cloud resources shortage and consequently enhance the Quality of Service of the cloud workflow, we proposed a prediction based workflow orchestration model. The workflow performance prediction used ARIMA to predict cloud resource shortages and supported agile workflow adaptation. We also evaluated workflow trust based on QoS to support the different adaptations schemes. The results we have obtained validated the effectiveness of our monitoring, prediction, and adaptation schemes. It evaluated monitoring various cloud services and resources and detected violations, predicted the cloud resource shortage accurately,

and took the appropriate adaptation actions in response to various resource violations.

8.2 Recommendations

The results of this study will be beneficial to researchers in the areas where workflow execution, deployment, adaptation, self-learning and self-reconfiguration are applicable. In the following, we depict some recommendations:

- Specifying end-to-end quality of Big Data workflows serves the advantage of capturing, customizing, and reusing existing quality specifications, and satisfying customized workflow requirements for IaaS, PaaS, and SaaS. Furthermore, reflecting different quality dimensions in a consolidated quality specification as an end-to-end model leads to accurate quality evaluation and is beneficial for users having limited technical cloud skills.

- CSP selection is one of the major challenges users face due to lack of technical knowledge or high number of available providers offering similar services. Our multi-dimensional selection model includes a set of algorithms and strategies that are proposed to help researchers consider different aspects of trust including self-experience, community assessment, and cloud providers advertised resource characteristics. Additionally, our model facilitates the automation of the CSP selection process with special emphasis on user quality preferences and Big Data workflows special characteristics. The developed application provides a user guided tool to effectively express their preferences while capturing all technical key aspects of the cloud service selection.

- Complex and data intensive workflows are currently implemented in different application domains including health, data curation, finance, transportation, energy, and smart cities. Researchers dealing with complex workflows are recommended to adopt automatic adaptation and reconfiguration mechanisms that features self-learning and self-reconfiguring to guarantee required QoS levels during runtime. Our monitoring and prediction based orchestration model satisfies the aforementioned highly valued features.

8.3 Future Directions

Cloud Computing offers scalable resources and multitude services to support workflow orchestration, monitoring, and adaptation. However, current cloud service orchestration approaches do not easily scale with the evolving complexity of workflows. Also, they are mostly adopting a procedural approach to build the workflow and manage its execution, which makes it hard to self-adapt, self-configure, and react to dynamic environment changes.

8.3.1 QoS Requirement Specification and Assessment

Most existing orchestration strategies do not focus on user QoS requirements. However, it is important that orchestration schemes select the required cloud resources that dynamically maximize the user-defined QoS (e.g., guarantee 24/7 service availability and supporting 100% data privacy). Such QoS enforcement may require extending the cloud service description with metadata to describe privacy rules and policies. Therefore, as future work, we suggest extending our specification model with metadata description to allow higher levels of user-defined QoS in Big Data workflow orchestration frameworks.

We also recommend assessing the data quality during earlier stages of Big Data pipeline, which will improve considerably the quality at final stages including analytics, and visualization of Big Data.

8.3.2 CSP Selection

Effective CSP selection is a major contributor to maintain high workflow QoS. We suggest to extend the CSP selection approach and perform a second phase of CSP evaluation post selection to ensure the appropriateness of the selection decision. This can be achieved through implementing a loop back process to allow any adjustment and re-evaluation of the quality metrics and processes during further provisioning phases. Additionally, we propose to exploit the monitoring and prediction techniques to continuously update the CSP ranking.

Recently, the number of cloud providers emerging in the market has increased as have the Big Data workflows grown in complexity. Hence, to handle the CSP selection problem among a very

large number of cloud providers for such different application types that have a high number of various quality attributes to describe requirements requires more sophisticated MADM algorithms. Accordingly, the quality attribute MADM constructed matrices will become very large and eventually require very high processing power. In other words, we suggest they can be considered as a Big Data workload problem. Thus, these matrices can be partitioned and distributed over multiple parallel VMs to scale up and handle extremely largescale data using MapReduce for promising selection results.

8.3.3 Runtime Intelligent Declarative Workflow Orchestration

Very few works have adopted a declarative workflow orchestration and incorporated features including real-time monitoring, workflow auto-configuration, and adaptation. However, these initiatives remain limited and do not handle the changes in its running environment and cope with the increasing complexity of real-time and context-aware workflow monitoring.

Therefore, it is important that future directions should focus on developing declarative cloud services orchestration that provision self-adaptation, self-healing, and self-configurable workflow featuring the following: 1) implementation of advanced reasoning of workflow runtime environment properties and autonomic orchestration tasks, 2) development of techniques to detect events patterns (e.g., search behavior in social media) and transform them into expressive models that are suitable for real-time cloud resource orchestration purposes, and 3) development of a kind of meta-model centered around planning, monitoring, and learning activities to support different levels of dynamicity and intelligences across the workflow value-chain.

In the context of Big Data workflow where, for example, dynamic IoT data is streamed from different devices and sensors, and a multitude of events can be executed (e.g., retrieve, regulate, automate, predict, and monitor) the above-proposed features can be entirely beneficial. For instance, real-time IoT stream processing will rapidly need to detect abnormalities in data collection and accordingly dynamically adapt the workflow. It should also support real-time responsiveness and deployment to fluctuating conditions and requirements. In addition, it should support performance and scalability as data volumes increase in size and complexity.

Challenges related to capturing, monitoring, and analyzing runtime data can be classified into

intelligence-aware resource orchestration and intelligence-aware data processing. Intelligence-aware resource orchestration deals with runtime resource description, requirements, and constraints. However, intelligence-aware data processing seamlessly integrates platforms, such as Hadoop, to support data processing distribution.

8.3.4 Cloud Workflow Orchestration Visualization Dashboard

Orchestrating Big Data workflows is challenging due to complex compositions and resource configurations. Workflow orchestration demands a deep knowledge of the resources and task dependencies and quality attributes to orchestrate the workflow while maintaining the required QoS efficiently. Nevertheless, most of the existing cloud orchestration tools use text-based resource description, deployment plans, and monitoring, which makes it hard to read, reconfigure, and comprehend with such large workflows. Hence, considering visual representations will enable seamless and efficient orchestration and adaptation. Using graphical notations to add resource components and develop workflow deployment plans simplifies orchestration, monitoring, adaptation, and general control of services and resources.

8.3.5 Cognitive Computing for Next Generation Workflow Systems

Cognitive computing is a new area of research, which is empowered by advances in machine learning, natural language processing, and broadly artificial intelligence, and has the potential to transform the next generation workflow systems. It will enable a scalable knowledge acquisition, learning at different stages of workflows including specification, deployment, and execution, which will open prospects for new levels of workflow automation and self-learning.

8.3.6 Distributed Trust Enforcement Using Blockchain

Blockchain is a new technology that enables secure transactions between unlimited numbers of users. It is an open and distributed ledger which supports transaction recordings permanently, safely, and efficiently to all users [264]. The transactions are recorded with complex encryption algorithms. Thus, it empowers and guarantee the trust in complex transactions. Its main power is adopting decentralization and replication throughout the nodes in the Web. As is being said, the

blockchain supports trust in information and algorithms. To this end, we foresee great potential in using blockchain technology to enforce trust over workflow orchestration. It has the capability of handling complex protocols and automation which would offer lower transaction costs when reducing human power and third-party trust evaluators. Nevertheless, it is recommended to use the blockchain to enable trust as a service model. It is worth investigating if the current cloud infrastructures will be able to handle the increasing demands and opportunities.

Bibliography

[1] M. D. Assuno, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *Journal of Parallel and Distributed Computing*, vol. 79-80, pp. 3 – 15, 2015, special Issue on Scalable Systems for Big Data Management and Analytics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731514001452

[2] R. Saranya, V. Muthukumar, and N. Mary, "Big data in cloud computing," *International Journal of Current Research in Computer Science and Technology (IJCRCST)*, vol. 1, pp. 44 – 49, 2014.

[3] SAS, "Big data meets big data analytics," Tech. Rep., 2012.

[4] K. Normandeau, "Beyond volume, variety and velocity is the issue of big data veracity," *Inside Big Data*, 2013.

[5] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, pp. 98 – 115, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306437914001288

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica *et al.*, "Above the clouds: A berkeley view of cloud computing," Tech. Rep., 2009.

[7] R. Quick, "5 reasons enterprises are frightened of the cloud," http://thenextweb.com/insider/2013/09/11/5-reasons-enterprises-are-frightened-of-the-cloud/, accessed: 2018-05-01.

[8] A. Kanwal, R. Masood, M. A. Shibli, and R. Mumtaz, "Taxonomy for trust models in cloud computing," *The Computer Journal*, vol. 58, no. 4, pp. 601–626, 2015. [Online]. Available: http://dx.doi.org/10.1093/comjnl/bxu138

[9] D. Weerasiri, "Configuration and orchestration techniques for federated cloud resources," Ph.D. dissertation, The University of New South Wales, 2016.

[10] A. L. Lemos, F. Daniel, and B. Benatallah, "Web service composition: a survey of techniques and tools," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 33, 2016.

[11] "deltacloud," http://deltacloud.apache.org, accessed: 2018-05-01.

[12] "Apache libcloud," http://libcloud.apache.org, accessed: 2018-05-01.

[13] "jclouds," http://www.jclouds.org, accessed: 2018-05-01.

[14] "openstack," http://www.openstack.org, accessed: 2018-05-01.

[15] M. Eisa, M. Younas, K. Basu, and H. Zhu, "Trends and directions in cloud service selection," in *Service-Oriented System Engineering (SOSE), 2016 IEEE Symposium on*. IEEE, 2016, pp. 423–432.

[16] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, vol. 45, pp. 134 – 150, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S108480451400160X

[17] L. Qi, W. Dou, Y. Zhou, J. Yu, and C. Hu, "A context-aware service evaluation approach over big data for cloud applications," *IEEE Transactions on Cloud Computing*, 2015.

[18] Z. Malik and A. Bouguettaya, "Rater credibility assessment in web services interactions," *World Wide Web*, vol. 12, no. 1, pp. 3–25, 2009.

[19] E. M. Maximilien and M. P. Singh, "Multiagent system for dynamic web services selection."

[20] Z. Malik and A. Bouguettaya, "Reputation bootstrapping for trust establishment among web services," *IEEE Internet Computing*, vol. 13, no. 1, pp. 40–47, 2009.

[21] H. Yahyaoui, "A trust-based game theoretical model for web services collaboration," *Knowledge-Based Systems*, vol. 27, pp. 162–169, 2012.

[22] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "A survey on trust and reputation models for web services: Single, composite, and communities," *Decision Support Systems*, vol. 74, pp. 121–134, 2015.

[23] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, "Tosca: portable automated deployment and management of cloud applications," in *Advanced Web Services*. Springer, 2014, pp. 527–549.

[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[25] N. Arora and R. K. Bawa, "A review on cloud to handle and process big data," *International Journal of Innovations & Advancement in Computer Science IJIACS ISSN*, pp. 2347–8616, 2014.

[26] J. Sun and C. K. Reddy, "Big data analytics for healthcare," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1525–1525.

[27] C. P. Chen and C.-Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on big data," *Information Sciences*, vol. 275, pp. 314–347, 2014.

[28] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," in *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE, 2013, pp. 48–55.

[29] S. M. Radack, "Cloud computing: a review of features, benefits, and risks, and recommendations for secure, efficient implementations," Tech. Rep., 2012.

[30] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[31] A. Fox, "Opportunities and challenges of cloud computing," in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2010 Symposium*. National Academies Press, 2011, pp. 3–14.

[32] A. D. JoSEP, R. KAtz, A. KonWinSKi, L. Gunho, D. PAttERSon, and A. RABKin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, 2010.

[33] B. M. Purcell, "Big data using cloud computing," *Journal of Technology Research*, vol. 5, p. 1, 2014.

[34] R. Branch, H. Tjeerdsma, C. Wilson, R. Hurley, and S. McConnell, "Cloud computing and big data: a review of current service models and hardware perspectives," *Journal of Software Engineering and Applications*, vol. 7, no. 08, p. 686, 2014.

[35] B. Li and R. Jain, "Survey of recent research progress and issues in big data," *Washington University in St. Louis, USA*, 2013.

[36] F. Pop and V. Cristea, "The art of scheduling for big data science." 2015.

[37] Y. Zhao, X. Fei, I. Raicu, and S. Lu, "Opportunities and challenges in running scientific workflows on the cloud," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2011 International Conference on*. IEEE, 2011, pp. 455–462.

[38] L. Ramakrishnan and B. Plale, "A multi-dimensional classification model for scientific workflow characteristics," in *Proceedings of the 1st International Workshop on Workflow Approaches to New Data-centric Science*. ACM, 2010, p. 4.

[39] R. Ranjan, L. Wang, J. Chen, and B. Benatallah, *Cloud computing: methodology, systems, and applications*. CRC Press, 2011.

[40] R. Ranjan, S. Garg, A. R. Khoskbar, E. Solaiman, P. James, and D. Georgakopoulos, "Orchestrating bigdata analysis workflows," *IEEE Cloud Computing*, vol. 4, no. 3, pp. 20–28, 2017.

[41] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision support systems*, vol. 43, no. 2, pp. 618–644, 2007.

[42] R. J. Manoj and A. Chandrasekar, "A literature review on trust management in web services access control," *International Journal on Web Service Computing*, vol. 4, no. 3, p. 1, 2013.

[43] J. Sänger, C. Richthammer, S. Hassan, and G. Pernul, "Trust and big data: a roadmap for research," in *Database and Expert Systems Applications (DEXA), 2014 25th International Workshop on*. IEEE, 2014, pp. 278–282.

[44] J.-b. Wu, "A trust evaluation model for web service with domain distinction," *International Journal of Granular Computing, Rough Sets and Intelligent Systems*, vol. 2, no. 4, pp. 273–280, 2012.

[45] A. Gholami and M. G. Arani, "A trust model based on quality of service in cloud computing environment," *International Journal of Database Theory and Application*, vol. 8, no. 5, pp. 161–170, 2015.

[46] B. Li, R. Song, L. Liao, and C. Liu, "A user-oriented trust model for web services," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. IEEE, 2013, pp. 224–232.

[47] *Cloud Service Measurement Index Consortium: SMI framework*.

[48] V. Jadhav, *Electing cloud service provider based on Quality of Service & Ranked method*.

[49] W. Viriyasitavat and A. Martin, "A survey of trust in workflows and relevant contexts," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 3, pp. 911–940, 2012.

[50] S. Yu and S. Guo, *Big data concepts, theories, and applications*. Springer, 2016.

[51] J.-H. Cho, K. Chan, and S. Adali, "A survey on trust modeling," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, p. 28, 2015.

[52] H. Lin, J. Hu, J. Liu, L. Xu, and Y. Wu, "A context aware reputation mechanism for enhancing big data veracity in mobile cloud computing," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2049–2054.

[53] O. Malačka, J. Samek, and F. Zbořil, "Event driven multi-context trust model," in *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*. IEEE, 2010, pp. 912–917.

[54] X. Li, W. Hu, T. Ding, and R. Ruiz, "Trust constrained workflow scheduling in cloud computing," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 164–169.

[55] D. Bernstein and D. Vij, "Intercloud security considerations," in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. IEEE, 2010, pp. 537–544.

[56] J. Abawajy, "Determining service trustworthiness in intercloud computing environments," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*. IEEE, 2009, pp. 784–788.

[57] P. Resnick and R. Zeckhauser, "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," in *The Economics of the Internet and E-commerce*. Emerald Group Publishing Limited, 2002, pp. 127–157.

[58] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on knowledge and data engineering*, vol. 26, no. 5, pp. 1253–1266, 2014.

[59] J. Guo and R. Chen, "A classification of trust computation models for service-oriented internet of things systems," in *Services Computing (SCC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 324–331.

[60] A. Josang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th bled electronic commerce conference*, vol. 5, 2002, pp. 2502–2511.

[61] Y. Wang, Y.-C. Lu, I.-R. Chen, J.-H. Cho, A. Swami, and C.-T. Lu, "Logittrust: A logit regression-based trust model for mobile ad hoc networks," in *6th ASE International Conference on Privacy, Security, Risk and Trust, Boston, MA*, 2014, pp. 1–10.

[62] A. Jøsang, "A logic for uncertain probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 03, pp. 279–311, 2001.

[63] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *Proceedings of the first international joint conference on Autonomous Agents and Multiagent Systems: Part 1*. ACM, 2002, pp. 294–301.

[64] A. M. Hammadi and O. Hussain, "A framework for sla assurance in cloud computing," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012, pp. 393–398.

[65] D. Chen, G. Chang, D. Sun, J. Li, J. Jia, and X. Wang, "Trm-iot: A trust management model based on fuzzy reputation for internet of things," *Computer Science and Information Systems*, vol. 8, no. 4, pp. 1207–1228, 2011.

[66] J. Sabater and C. Sierra, "Reputation and social network analysis in multi-agent systems," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: Part 1*. ACM, 2002, pp. 475–482.

[67] C.-N. Ziegler and G. Lausen, "Spreading activation models for trust propagation," in *e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on*. IEEE, 2004, pp. 83–97.

[68] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[69] W. Hussain, F. K. Hussain, and O. K. Hussain, "Maintaining trust in cloud computing through sla monitoring," in *International Conference on Neural Information Processing*. Springer, 2014, pp. 690–697.

[70] M. Alhamad, T. Dillon, and E. Chang, "Sla-based trust model for cloud computing," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*. Ieee, 2010, pp. 321–324.

[71] F. Jrad, J. Tao, and A. Streit, "Sla based service brokering in intercloud environments." *CLOSER*, vol. 2012, pp. 76–81, 2012.

[72] R. Manikandan and G. Kousalya, "A framework for an intelligent broker model of cloud service selection," *Asian Journal of Information Technology*, vol. 15, no. 11, pp. 1776–1784, 2016.

[73] T. H. Noor, Q. Z. Sheng, S. Zeadally, and J. Yu, "Trust management of services in cloud environments: Obstacles and solutions," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 12, 2013.

[74] D. Daniel and S. J. Lovesum, "A novel approach for scheduling service request in cloud with trust monitor," in *Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011 International Conference on*. IEEE, 2011, pp. 509–513.

[75] T. B. Quillinan, K. P. Clark, M. Warnier, F. M. Brazier, and O. Rana, "Negotiation and monitoring of service level agreements," in *Grids and Service-Oriented Architectures for Service Level Agreements*. Springer, 2010, pp. 167–176.

[76] K. Gokulnath and R. Uthariaraj, "Game theory based trust model for cloud environment," *The Scientific World Journal*, vol. 2015, 2015.

[77] M. M. Hassan, M. Abdullah-Al-Wadud, A. Almogren, S. Rahman, A. Alelaiwi, A. Alamri,

M. Hamid *et al.*, "Qos and trust-aware coalition formation game in data-intensive cloud federations," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 10, pp. 2889–2905, 2016.

[78] B. Khosravifar, M. Alishahi, J. Bentahar, and P. Thiran, "A game theoretic approach for analyzing the efficiency of web services in collaborative networks," in *Services Computing (SCC), 2011 IEEE International Conference on*. IEEE, 2011, pp. 168–175.

[79] M. K. Muchahari and S. K. Sinha, "A new trust management architecture for cloud computing environment," in *Cloud and Services Computing (ISCOS), 2012 International Symposium on*. IEEE, 2012, pp. 136–140.

[80] H. Kim, H. Lee, W. Kim, and Y. Kim, "A trust evaluation model for qos guarantee in cloud systems," *International Journal of Grid and Distributed Computing*, vol. 3, no. 1, pp. 1–10, 2010.

[81] A. Hammam and S. Senbel, "A trust management system for ad-hoc mobile clouds," in *Computer Engineering & Systems (ICCES), 2013 8th International Conference on*. IEEE, 2013, pp. 31–38.

[82] X. Li and J. Du, "Adaptive and attribute-based trust model for service-level agreement guarantee in cloud computing," *IET Information Security*, vol. 7, no. 1, pp. 39–50, 2013.

[83] B. Li, L. Liao, H. Leung, and R. Song, "Phat: A preference and honesty aware trust model for web services," *IEEE Transactions on network and service management*, vol. 11, no. 3, pp. 363–375, 2014.

[84] P. D. Manuel, S. T. Selvi, and M. I. Abd-El Barr, "Trust management system for grid and cloud resources," in *Advanced Computing, 2009. ICAC 2009. First International Conference on*. IEEE, 2009, pp. 176–181.

[85] E. D. Canedo, R. T. de_Sousa Junior, R. de_Oliveira Albuquerque, and F. L. L. de Mendonça, "File exchange in a private cloud supported by a trust model," in *Cyber-Enabled Distributed*

Computing and Knowledge Discovery (CyberC), 2012 International Conference on. IEEE, 2012, pp. 89–96.

[86] M. Tang, X. Dai, J. Liu, and J. Chen, "Towards a trust evaluation middleware for cloud service selection," *Future Generation Computer Systems*, vol. 74, pp. 302–312, 2017.

[87] J. Mitchell, S. Rizvi, and J. Ryoo, "A fuzzy-logic approach for evaluating a cloud service provider," in *Software Security and Assurance (ICSSA), International Conference on*. IEEE, 2015, pp. 19–24.

[88] S. Ristov and M. Gusev, "A methodology to evaluate the trustworthiness of cloud service providers' availability," in *EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE*. IEEE, 2015, pp. 1–6.

[89] J. H. Abawajy and A. M. Goscinski, "A reputation-based grid information service," in *International Conference on Computational Science*. Springer, 2006, pp. 1015–1022.

[90] Z. Malik and A. Bouguettaya, "Rateweb: Reputation assessment for trust establishment among web services," *The VLDB JournalThe International Journal on Very Large Data Bases*, vol. 18, no. 4, pp. 885–911, 2009.

[91] M. K. Goyal, A. Aggarwal, P. Gupta, and P. Kumar, "Qos based trust management model for cloud iaas," in *Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on*. IEEE, 2012, pp. 843–847.

[92] M. Alhanahnah, P. Bertok, and Z. Tari, "Trusting cloud service providers: trust phases and a taxonomy of trust factors," *IEEE Cloud Computing*, vol. 4, no. 1, pp. 44–54, 2017.

[93] N. Ghosh, S. K. Ghosh, and S. K. Das, "Selcsp: A framework to facilitate selection of cloud service providers," *IEEE transactions on cloud computing*, vol. 3, no. 1, pp. 66–79, 2015.

[94] S. R. Ahemad and L. Dole, "An efficient approach based on identity for distributed data possession in multicloud using selcsp framework," in *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017 International Conference on*. IEEE, 2017, pp. 1–4.

[95] E. Badidi, "A cloud service broker for sla-based saas provisioning," in *Information Society (i-Society), 2013 International Conference on*. IEEE, 2013, pp. 61–66.

[96] A. Al Falasi, M. A. Serhani, and R. Dssouli, "A model for multi-levels sla monitoring in federated cloud environment," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. IEEE, 2013, pp. 363–370.

[97] J. Lee, J. Kim, D.-J. Kang, N. Kim, and S. Jung, "Cloud service broker portal: Main entry point for multi-cloud service providers and consumers," in *Advanced Communication Technology (ICACT), 2014 16th International Conference on*. IEEE, 2014, pp. 1108–1112.

[98] X. Wang, J. Cao, and Y. Xiang, "Dynamic cloud service selection using an adaptive learning mechanism in multi-cloud computing," *Journal of Systems and Software*, vol. 100, pp. 195–210, 2015.

[99] S. Sundareswaran, A. Squicciarini, and D. Lin, "A brokerage-based approach for cloud service selection," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 558–565.

[100] Y.-f. Zheng and J. Xu, "Multiple attribute decision making with triangular intuitionistic fuzzy numbers and application to cloud service provider selection," in *Information Technology and Electronic Commerce (ICITEC), 2014 2nd International Conference on*. IEEE, 2014, pp. 311–315.

[101] R. Hans, U. Lampe, and R. Steinmetz, "Qos-aware, cost-efficient selection of cloud data centers," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 946–947.

[102] R. Hans, D. Steffen, U. Lampe, B. Richerzhagen, and R. Steinmetz, "Setting priorities-a heuristic approach for cloud data center selection." in *CLOSER*, 2015, pp. 221–228.

[103] C.-W. Chang, P. Liu, and J.-J. Wu, "Probability-based cloud storage providers selection algorithms with maximum availability," in *Parallel Processing (ICPP), 2012 41st International Conference on*. IEEE, 2012, pp. 199–208.

[104] B. Martens and F. Teuteberg, "Decision-making in cloud computing environments: A cost and risk based approach," *Information Systems Frontiers*, vol. 14, no. 4, pp. 871–893, 2012.

[105] Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin, "Qos-driven service selection for multi-tenant saas," in *Cloud computing (cloud), 2012 ieee 5th international conference on*. IEEE, 2012, pp. 566–573.

[106] H. Qian and Q. Lv, "Proximity-aware cloud selection and virtual machine allocation in iaas cloud platforms," in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. IEEE, 2013, pp. 403–408.

[107] H.-K. Chen, C.-Y. Lin, and J.-H. Chen, "A multi-objective evolutionary approach for cloud service provider selection problems with dynamic demands," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2014, pp. 841–852.

[108] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *Web Services (ICWS), 2010 IEEE International Conference on*. IEEE, 2010, pp. 9–16.

[109] Z. Luo, Y. Li, and J. Yin, "Location: a feature for service selection in the era of big data," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 515–522.

[110] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective web service recommendation method based on personalized collaborative filtering," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 211–218.

[111] F. Wagner, A. Klein, B. Klöpper, F. Ishikawa, and S. Honiden, "Multi-objective service composition with time-and input-dependent qos," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 234–241.

[112] S. Telenyk, P. Bidyuk, E. Zharikov, and M. Yasochka, "Assessment of cloud service provider quality metrics," in *Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), 2017 International Conference on*. IEEE, 2017, pp. 1–5.

[113] C. Anglano, M. Canonico, and M. Guazzone, "Fc2q: exploiting fuzzy control in server consolidation for cloud applications with sla constraints," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 17, pp. 4491–4514, 2015.

[114] P. Jamshidi, A. Ahmad, and C. Pahl, "Autonomic resource provisioning for cloud-based software," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2014, pp. 95–104.

[115] S. S. Wagle, M. Guzek, and P. Bouvry, "Cloud service providers ranking based on service delivery and consumer experience," in *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*. IEEE, 2015, pp. 209–212.

[116] Y. Wang, Q. He, and Y. Yang, "Qos-aware service recommendation for multi-tenant saas on the cloud," in *Services Computing (SCC), 2015 IEEE International Conference on*. IEEE, 2015, pp. 178–185.

[117] W. Li, Y. Zhong, X. Wang, and Y. Cao, "Resource virtualization and service selection in cloud logistics," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1696–1704, 2013.

[118] Y. Du, X. Wang, L. Ai, and X. Li, "Dynamic selection of services under temporal constraints in cloud computing," in *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*. IEEE, 2012, pp. 252–259.

[119] H. Liang, Y. Du, and S. Li, "An improved genetic algorithm for service selection under temporal constraints in cloud computing," in *International Conference on Web Information Systems Engineering*. Springer, 2013, pp. 309–318.

[120] H. Liang and Y. Du, "Two-stage dynamic optimisation of service processes with temporal

constraints," *International Journal of High Performance Computing and Networking*, vol. 9, no. 1-2, pp. 116–126, 2016.

[121] A. Polleres and D. Fensel, "Wsml-a language framework for semantic web services," 2005.

[122] T. Segaran, C. Evans, and J. Taylor, *Programming the Semantic Web: Build Flexible Applications with Graph Data*. " O'Reilly Media, Inc.", 2009.

[123] L. Qu *et al.*, "Credible service selection in cloud environments," 2016.

[124] L. D. Ngan and R. Kanagasabai, "Owl-s based semantic cloud service broker," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*. IEEE, 2012, pp. 560–567.

[125] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "Owl 2: The next step for owl," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, no. 4, pp. 309–322, 2008.

[126] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, "A declarative recommender system for cloud infrastructure services selection," in *International Conference on Grid Economics and Business Models*. Springer, 2012, pp. 102–113.

[127] B. Benatallah, M. Dumas, Q. Z. Sheng, and A. H. Ngu, "Declarative composition and peer-to-peer provisioning of dynamic web services," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 297–308.

[128] A. Ruiz-Alvarez and M. Humphrey, "An automated approach to cloud storage service selection," in *Proceedings of the 2nd international workshop on Scientific cloud computing*. ACM, 2011, pp. 39–48.

[129] R. Zabolotnyi, P. Leitner, S. Schulte, and S. Dustdar, "Speedl-a declarative event-based language to define the scaling behavior of cloud applications," in *Services (SERVICES), 2015 IEEE World Congress on*. IEEE, 2015, pp. 71–78.

[130] E. Wittern, J. Kuhlenkamp, and M. Menzel, "Cloud service selection based on variability modeling," in *International Conference on Service-Oriented Computing*. Springer, 2012, pp. 127–141.

[131] A. E. Walsh, *Uddi, Soap, and WSDL: the web services specification reference book*. Prentice Hall Professional Technical Reference, 2002.

[132] A. Goscinski and M. Brock, "Toward dynamic and attribute based publication, discovery and selection for cloud computing," *Future generation computer systems*, vol. 26, no. 7, pp. 947–970, 2010.

[133] R. Karim, C. Ding, and A. Miri, "End-to-end performance prediction for selecting cloud services solutions," in *Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on*. IEEE, 2015, pp. 69–77.

[134] M. Tang, T. Zhang, J. Liu, and J. Chen, "Cloud service qos prediction via exploiting collaborative filtering and location-based data smoothing," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 18, pp. 5826–5839, 2015.

[135] Q. Yu, "Cloudrec: a framework for personalized service recommendation in the cloud," *Knowledge and Information Systems*, vol. 43, no. 2, pp. 417–443, 2015.

[136] K. Alexander, C. Lee, E. Kim, and S. Helal, "Enabling end-to-end orchestration of multi-cloud applications," *IEEE Access*, vol. 5, pp. 18 862–18 875, 2017.

[137] "Docker," https://www.docker.com/, accessed: 2018-05-01.

[138] "Operate big software at scale on any cloud," https://jujucharms.com/, accessed: 2018-05-01.

[139] http://deepdive.stanford.edu/, accessed: 2018-05-01.

[140] "Tools for monitoring compute, storage, and network resources," https://kubernetes.io/docs/tasks/debug-application-cluster/resource-usage-monitoring/, accessed: 2018-05-01.

[141] "Devops," http://deepdive.stanford.edu/, accessed: 2018-05-01.

[142] "Object management group business process model and notation," http://www.bpmn.org/., accessed: 2018-04-01.

[143] M. B. Juric, B. Mathew, and P. Sarang, *Business Process Execution Language for Web Services : An Architect and Developer's Guide to Orchestrating Web Services Using BPEL4WS*. Packt Publishing, 2004.

[144] B. P. Rimal and M. A. El-Refaey, "A framework of scientific workflow management systems for multi-tenant cloud orchestration environment," in *Enabling technologies: Infrastructures for collaborative enterprises (wetice), 2010 19th ieee international workshop on*. IEEE, 2010, pp. 88–93.

[145] R. Filgueira, R. F. da Silva, A. Krause, E. Deelman, and M. Atkinson, "Asterism: Pegasus and dispel4py hybrid workflows for data-intensive science," in *Data-Intensive Computing in the Clouds (DataCloud), 2016 Seventh International Workshop on*. IEEE, 2016, pp. 1–8.

[146] F. Amato and F. Moscato, "Exploiting cloud and workflow patterns for the analysis of composite cloud services," *Future Generation Computer Systems*, vol. 67, pp. 255–265, 2017.

[147] W. Jaradat, A. Dearle, and A. Barker, "Workflow partitioning and deployment on the cloud using orchestra," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*. IEEE Computer Society, 2014, pp. 251–260.

[148] X. Li, K. Li, X. Pang, and Y. Wang, "An orchestration based cloud auto-healing service framework," in *Edge Computing (EDGE), 2017 IEEE International Conference on*. IEEE, 2017, pp. 190–193.

[149] Q. Qi, J. Liao, J. Wang, Q. Li, and Y. Cao, "Dynamic resource orchestration for multi-task application in heterogeneous mobile cloud computing," in *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*. IEEE, 2016, pp. 221–226.

[150] S. Nepal, Z. Malik, and A. Bouguettaya, "Reputation management for composite services in service-oriented systems," *International Journal of Web Services Research (IJWSR)*, vol. 8, no. 2, pp. 29–52, 2011.

[151] D. Weerasiri, M. C. Barukh, B. Benatallah, Q. Z. Sheng, and R. Ranjan, "A taxonomy and survey of cloud resource orchestration techniques," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 26, 2017.

[152] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on software engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[153] A. F. M. Hani, I. V. Paputungan, and M. F. Hassan, "Renegotiation in service level agreement management for a cloud-based system," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 51, 2015.

[154] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky computing," *IEEE Internet Computing*, vol. 13, no. 5, pp. 43–51, 2009.

[155] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud–protocols and formats for cloud computing. internet and web applications and services, 2009. iciw'09," in *Fourth International Conference*, 2009, pp. 1–3.

[156] A. N. Toosi, R. N. Calheiros, and R. Buyya, "Interconnected cloud computing environments: Challenges, taxonomy, and survey," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, p. 7, 2014.

[157] "Yarn," https://yarnpkg.com/en/, accessed: 2018-04-01.

[158] "Apache mesos," http://mesos.apache.org/, accessed: 2018-04-01.

[159] "Amazon emr," https://aws.amazon.com/emr/, accessed: 2018-04-01.

[160] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, and A. Rasin, "Hadoopdb: an architectural hybrid of mapreduce and dbms technologies for analytical workloads," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 922–933, 2009.

[161] R. Castro Fernandez, M. Migliavacca, E. Kalyvianaki, and P. Pietzuch, "Integrating scale out and fault tolerance in stream processing using operator state management," in *Proceedings*

of the 2013 ACM SIGMOD international conference on Management of data. ACM, 2013, pp. 725–736.

[162] A. Castiglione, M. Gribaudo, M. Iacono, and F. Palmieri, "Exploiting mean field analysis to model performances of big data architectures," *Future Generation Computer Systems*, vol. 37, pp. 203–211, 2014.

[163] D. Bruneo, F. Longo, R. Ghosh, M. Scarpa, A. Puliafito, and K. S. Trivedi, "Analytical modeling of reactive autonomic management techniques in iaas clouds," in *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*. IEEE, 2015, pp. 797–804.

[164] H. Kim and M. Parashar, "Cometcloud: An autonomic cloud engine," *Cloud Computing: Principles and Paradigms*, pp. 275–297, 2011.

[165] A. Nasridinov, J.-Y. Byun, and Y.-H. Park, "A qos-aware performance prediction for self-healing web service composition," in *Cloud and Green Computing (CGC), 2012 Second International Conference on*. IEEE, 2012, pp. 799–803.

[166] S. Schulte, C. Janiesch, S. Venugopal, I. Weber, and P. Hoenisch, "Elastic business process management: State of the art and open challenges for bpm in the cloud," *Future Generation Computer Systems*, vol. 46, pp. 36–50, 2015.

[167] S. Singh and I. Chana, "Qos-aware autonomic resource management in cloud computing: a systematic review," *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 42, 2016.

[168] J. Yang, W. Lin, and W. Dou, "An adaptive service selection method for cross-cloud service composition," *Concurrency and Computation: Practice and Experience*, vol. 25, no. 18, pp. 2435–2454, 2013.

[169] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[170] S. Ferretti, V. Ghini, F. Panzieri, M. Pellegrini, and E. Turrini, "Qos-aware clouds," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 321–328.

[171] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: managing performance interference effects for qos-aware clouds," in *Proceedings of the 5th European conference on Computer systems*. ACM, 2010, pp. 237–250.

[172] K. Kritikos, C. Zeginis, F. Griesinger, D. Seybold, and J. Domaschka, "A cross-layer bpaas adaptation framework," in *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on*. IEEE, 2017, pp. 241–248.

[173] C. Inzinger, W. Hummer, B. Satzger, P. Leitner, and S. Dustdar, "Generic event-based monitoring and adaptation methodology for heterogeneous distributed systems," *Software: Practice and Experience*, vol. 44, no. 7, pp. 805–822, 2014.

[174] R. Popescu, A. Staikopoulos, A. Brogi, P. Liu, and S. Clarke, "A formalized, taxonomy-driven approach to cross-layer application adaptation," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 7, no. 1, p. 7, 2012.

[175] A. Zengin, R. Kazhamiakin, and M. Pistore, "Clam: Cross-layer management of adaptation decisions for service-based applications," in *Web Services (ICWS), 2011 IEEE International Conference on*. IEEE, 2011, pp. 698–699.

[176] S. Guinea, G. Kecskemeti, A. Marconi, and B. Wetzstein, "Multi-layered monitoring and adaptation," in *International Conference on Service-Oriented Computing*. Springer, 2011, pp. 359–373.

[177] C. Zeginis, K. Kritikos, and D. Plexousakis, "Event pattern discovery for cross-layer adaptation of multi-cloud applications," in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2014, pp. 138–147.

[178] J. Domaschka, D. Seybold, F. Griesinger, and D. Baur, "Axe: a novel approach for generic, flexible, and comprehensive monitoring and adaptation of cross-cloud applications," in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2015, pp. 184–196.

[179] L. Schubert, J. Domaschka, and P. Guisset, "Paasage-making cloud usage easy," *CloudScape*, 2016.

[180] G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "Sybl+ mela: Specifying, monitoring, and controlling elasticity of cloud services," in *International Conference on Service-Oriented Computing*. Springer, 2013, pp. 679–682.

[181] W. Barth, *Nagios: System and network monitoring*. No Starch Press, 2008.

[182] P. Zadrozny and R. Kodali, *Big Data Analytics Using Splunk: Deriving Operational Intelligence from Social Media, Machine Data, Existing Data Warehouses, and Other Real-Time Streaming Sources*. Apress, 2013.

[183] "Ganglia monitoring system," http://ganglia.sourceforge.net/, accessed: 2018-04-01.

[184] "Apache chukwa," http://chukwa.apache.org/, accessed: 2018-04-01.

[185] "Sematext," https://sematext.com/, accessed: 2018-04-01.

[186] "Sequenceiq," http://sequenceiq.com/, accessed: 2018-04-01.

[187] K. Alhamazani, R. Ranjan, K. Mitra, F. Rabhi, P. P. Jayaraman, S. U. Khan, A. Guabtni, and V. Bhatnagar, "An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art," *Computing*, vol. 97, no. 4, pp. 357–377, 2015.

[188] K. Alhamazani, R. Ranjan, P. P. Jayaraman, K. Mitra, F. Rabhi, D. Georgakopoulos, and L. Wang, "Cross-layer multi-cloud real-time application qos monitoring and benchmarking as-a-service framework," *IEEE Transactions on Cloud Computing*, 2015.

[189] G. Katsaros, G. Kousiouris, S. V. Gogouvitis, D. Kyriazis, A. Menychtas, and T. Varvarigou, "A self-adaptive hierarchical monitoring mechanism for clouds," *Journal of Systems and Software*, vol. 85, no. 5, pp. 1029–1041, 2012.

[190] S. Clayman, A. Galis, C. Chapman, G. Toffetti, L. Rodero-Merino, L. M. Vaquero, K. Nagin, and B. Rochwerger, "Monitoring service clouds in the future internet." in *Future Internet Assembly*. Valencia, Spain, 2010, pp. 115–126.

[191] L. Romano, D. De Mari, Z. Jerzak, and C. Fetzer, "A novel approach to qos monitoring in the cloud," in *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*. IEEE, 2011, pp. 45–51.

[192] S. A. De Chaves, R. B. Uriarte, and C. B. Westphall, "Toward an architecture for monitoring private clouds," *IEEE Communications Magazine*, vol. 49, no. 12, pp. 130–137, 2011.

[193] Y. Sun, W. Tan, L. Li, G. Lu, and A. Tang, "Sla detective control model for workflow composition of cloud services," in *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on*. IEEE, 2013, pp. 165–171.

[194] D. Kyriazis, K. Kostantos, A. Kapsalis, S. Gogouvitis, and T. Varvarigou, "Qos-oriented service management in large scale federated clouds," in *Computers and Communications (ISCC), 2013 IEEE Symposium on*. IEEE, 2013, pp. 000 022–000 027.

[195] L. Baresi, S. Guinea, G. Quattrocchi, and D. A. Tamburri, "Microcloud: A container-based solution for efficient resource management in the cloud," in *Smart Cloud (SmartCloud), IEEE International Conference on*. IEEE, 2016, pp. 218–223.

[196] L. Ramakrishnan, J. S. Chase, D. Gannon, D. Nurmi, and R. Wolski, "Deadline-sensitive workflow orchestration without explicit resource control," *Journal of Parallel and Distributed Computing*, vol. 71, no. 3, pp. 343–353, 2011.

[197] H. Viswanathan, P. Pandey, and D. Pompili, "Maestro: orchestrating concurrent application workflows in mobile device clouds," in *Autonomic Computing (ICAC), 2016 IEEE International Conference on*. IEEE, 2016, pp. 257–262.

[198] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE Computer Society, 2010, pp. 495–504.

[199] X. Zhang, B. Li, and J. Zhu, "A monitoring and prediction model of workflow based self-adaptive software system," in *Advanced Cloud and Big Data (CBD), 2014 Second International Conference on*. IEEE, 2014, pp. 115–121.

[200] H. Arabnejad, C. Pahl, P. Jamshidi, and G. Estrada, "A comparison of reinforcement learning techniques for fuzzy cloud auto-scaling," in *Cluster, Cloud and Grid Computing (CCGRID), 2017 17th IEEE/ACM International Symposium on*. IEEE, 2017, pp. 64–73.

[201] X. Dutreilh, S. Kirgizov, O. Melekhova, J. Malenfant, N. Rivierre, and I. Truck, "Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow," in *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, 2011, pp. 67–74.

[202] Y. Wei, M. B. Blake, and I. Saleh, "Adaptive resource management for service workflows in cloud environments," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 2147–2156.

[203] P. Nguyen and K. Nahrstedt, "Monad: Self-adaptive micro-service infrastructure for heterogeneous scientific workflows," in *Autonomic Computing (ICAC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 187–196.

[204] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access*, vol. 2, pp. 652–687, 2014.

[205] F. Sidi, P. H. S. Panahy, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha, "Data quality: A survey of data quality dimensions," in *Information Retrieval & Knowledge Management (CAMP), 2012 International Conference on*. IEEE, 2012, pp. 300–304.

[206] P. Glowalla, P. Balazy, D. Basten, and A. Sunyaev, "Process-driven data quality management–an application of the combined conceptual life cycle model," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 4700–4709.

[207] B. T. Hazen, C. A. Boone, J. D. Ezell, and L. A. Jones-Farmer, "Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the

problem and suggestions for research and applications," *International Journal of Production Economics*, vol. 154, pp. 72–80, 2014.

[208] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 16, 2009.

[209] I. Taleb, R. Dssouli, and M. A. Serhani, "Big data pre-processing: a quality framework," in *Big Data (BigData Congress), 2015 IEEE International Congress on*. IEEE, 2015, pp. 191–198.

[210] A. Immonen, P. Pääkkönen, and E. Ovaska, "Evaluating the quality of social media data in big data architecture," *IEEE Access*, vol. 3, pp. 2028–2043, 2015.

[211] "Understanding metadata," Tech. Rep., 2004.

[212] D. O'Neill, "Id3.org," http://ID3.org/, accessed: 2018-05-01.

[213] T. N. Huynh, O. Mangisengi, and A. M. Tjoa, "Metadata for object-relational data warehouse." in *DMDW*, 2000, p. 3.

[214] W. Cathro, "Metadata: an overview," *National Library of Australia Staff Papers*, 2009.

[215] "Semantic recommendation," http://dublincore.org/specifications/, accessed: 2018-05-01.

[216] E. H. Fegraus, S. Andelman, M. B. Jones, and M. Schildhauer, "Maximizing the value of ecological data with structured metadata: An introduction to ecological metadata language (eml) and principles for metadata creation." 2005.

[217] T. Vetterli, A. Vaduva, and M. Staudt, "Metadata standards for data warehousing: open information model vs. common warehouse metadata," *ACM Sigmod Record*, vol. 29, no. 3, pp. 68–75, 2000.

[218] "Introducing json," http://www.json.org/, accessed: 2018-05-01.

[219] S. Ran, "A model for web services discovery with qos," *ACM Sigecom exchanges*, vol. 4, no. 1, pp. 1–10, 2003.

[220] M. Mehdi, N. Bouguila, and J. Bentahar, "A qos-based trust approach for service selection and composition via bayesian networks," in *Web Services (ICWS), 2013 IEEE 20th International Conference on*. IEEE, 2013, pp. 211–218.

[221] J. Huang, G. Liu, Q. Duan, and Y. Yan, "Qos-aware service composition for converged network-cloud service provisioning," in *Services Computing (SCC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 67–74.

[222] X. Liu, Y. Yang, D. Yuan, G. Zhang, W. Li, and D. Cao, "A generic qos framework for cloud workflow systems," in *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*. IEEE, 2011, pp. 713–720.

[223] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 281–308, 2004.

[224] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A cost-effective strategy for intermediate data storage in scientific cloud workflow systems," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–12.

[225] D. He, Z. Peng, L. Hong, and Y. Zhang, "A social reputation management for web communities," in *International Conference on Web-Age Information Management*. Springer, 2011, pp. 167–174.

[226] A. Gutowska and A. Sloane, "Evaluation of reputation metric for the b2c e-commerce reputation system." in *WEBIST*, 2009, pp. 489–498.

[227] B. Yu and M. P. Singh, "Distributed reputation management for electronic commerce," *Computational Intelligence*, vol. 18, no. 4, pp. 535–549, 2002.

[228] "Calculating a weighted average," http://www.blacksdomain.com/files/Notes/Calculating_WA.php, 2016, accessed: 2018-05-01.

[229] W. Edwards, "How to use multiattribute utility measurement for social decisionmaking," *IEEE transactions on systems, man, and cybernetics*, vol. 7, no. 5, pp. 326–340, 1977.

[230] W. Edwards and F. H. Barron, "Smarts and smarter: Improved simple methods for multiattribute utility measurement," *Organizational behavior and human decision processes*, vol. 60, no. 3, pp. 306–325, 1994.

[231] U. Kumar, A. Ahmadi, A. K. Verma, and P. Varde, *Current Trends in Reliability, Availability, Maintainability and Safety: An Industry Perspective*. Springer, 2015.

[232] P. Saripalli and G. Pingali, "Madmac: multiple attribute decision methodology for adoption of clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 316–323.

[233] K. P. Yoon and C.-L. Hwang, *Multiple attribute decision making: an introduction*. Sage publications, 1995, vol. 104.

[234] A. Adriyendi, "Multi-attribute decision making using simple additive weighting and weighted product in food choice," *International Journal of Information Engineering and Electronic Business*, vol. 6, pp. 8–14, 2015.

[235] P. W. Bridgman, *Dimensional analysis*. Yale University Press, 1922.

[236] E. Triantaphyllou, "Multi-criteria decision making methods," in *Multi-criteria decision making methods: A comparative study*. Springer, 2000, pp. 5–21.

[237] D. A. Freedman, *Statistical models: theory and practice*. cambridge university press, 2009.

[238] R. Kumar and G. Sahoo, "Cloud computing simulation using cloudsim," *arXiv preprint arXiv:1403.3253*, 2014.

[239] S. Nepal, Z. Malik, and A. Bouguettaya, "Reputation propagation in composite services," in *Web Services, 2009. ICWS 2009. IEEE International Conference on*. IEEE, 2009, pp. 295–302.

[240] L. Qu, Y. Wang, M. A. Orgun, L. Liu, H. Liu, and A. Bouguettaya, "Cccloud: Context-aware and credible cloud service selection based on subjective assessment and objective assessment," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 369–383, 2015.

[241] "Yet another markup language (yaml) 1.0," http://yaml.org/spec/history/2001-12-10.html, 2001, accessed: 2018-05-01.

[242] A. Lomuscio, H. Qu, and F. Raimondi, "Mcmas: A model checker for the verification of multi-agent systems," in *International conference on computer aided verification*. Springer, 2009, pp. 682–688.

[243] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT press, 1999.

[244] "Swarm mode key concepts, docker doc," https://docs.docker.com/engine/swarm/key-concepts/, 2017, accessed: 2018-05-01.

[245] S. Prodan, "Docker swarm instrumentation with prometheus," https://stefanprodan.com/2017/docker-swarm-instrumentation-with-prometheus/, accessed: 2018-05-01.

[246] "Grafana - the open platform for analytics and monitoring," https://grafana.com/, 2017, accessed: 2018-05-01.

[247] "Prometheus - monitoring system & time series database," https://prometheus.io/, 2017, accessed: 2018-05-01.

[248] "Slack features," https://slack.com/features, 2017, accessed: 2018-05-01.

[249] "The mimic-iii clinical database," https://www.physionet.org/physiobank/database/mimic3cdb/, 2017, accessed: 2018-05-01.

[250] "Mit-lcp/mimic-code," https://github.com/MIT-LCP/mimic-code/tree/master/buildmimic/docker, 2017, accessed: 2018-05-01.

[251] G. Cugola, C. Ghezzi, and L. S. Pinto, "Dsol: a declarative approach to self-adaptive service orchestrations," *Computing*, vol. 94, no. 7, pp. 579–617, 2012.

[252] C. Liu, B. T. Loo, and Y. Mao, "Declarative automated cloud resource orchestration," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 26.

[253] R. Ranjan and B. Benatallah, "Programming cloud resource orchestration framework: operations and research challenges," *arXiv preprint arXiv:1204.2204*, 2012.

[254] H. Lu, M. Shtern, B. Simmons, M. Smit, and M. Litoiu, "Pattern-based deployment service for next generation clouds," in *Services (SERVICES), 2013 IEEE Ninth World Congress on*. IEEE, 2013, pp. 464–471.

[255] C. Roser, K. Lorentzen, D. Lenze, J. Deuse, F. Klenner, R. Richter, J. Schmitt, and P. Willats, "Bottleneck prediction using the active period method in combination with buffer inventories," in *IFIP International Conference on Advances in Production Management Systems*. Springer, 2017, pp. 374–381.

[256] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[257] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[258] J. D. Hamilton, *Time series analysis*. Princeton university press Princeton, 1994, vol. 2.

[259] R. Nau, "Statistical forecasting: notes on regression and time series analysis," http://people.duke.edu/~rnau/411home.htm, accessed: 2018-05-01.

[260] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[261] Y. Hu, Q. Peng, X. Hu, and R. Yang, "Web service recommendation based on time series forecasting and collaborative filtering," in *Web Services (ICWS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 233–240.

[262] "Choosing the right metric for evaluating ml models - Łpart 1," https://towardsdatascience.com/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4, accessed: 2018-05-01.

[263] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," 2011.

[264] M. Iansiti and K. R. Lakhani, "The truth about blockchain," *Harvard Business Review*, vol. 95, no. 1, pp. 118–127, 2017.

Appendix A

Publications

This thesis is based on the research work I have performed with the help of my supervisors and other colleagues during my PhD program at the Department of Concordia Institute for Information Systems Engineering, Concordia University between 2016 and 2018. Some parts of my research have been published or are under review as follows:

(1) H. El-Kassabi, M. A. Serhani, R. Dssouli and B. Benatallah, "A Multi-Dimensional Trust Model for Processing Big Data Over Competing Clouds," IEEE Access, 2018. (Impact Factor: 3.55)

(2) M. A. Serhani, H. El-Kassabi, and Ikbal Taleb, "Towards an Efficient Federated Cloud Service Selection to Support Workflow Big Data Requirements," Advances in Science, Technology and Engineering Systems Journal (ASTESJ), 2018. (Scopus Indexed)

(3) H. El-Kassabi, M. A. Serhani, R. Dssouli and A. N. Navaz, "Trust Enforcement Through Self-adapting Cloud Workflow Orchestration," Under review in Future Generation Computer Systems Journal, Elsevier, 2018. (Impact Factor: 4.639)

(4) H. El-Kassabi, M. A. Serhani, R. Dssouli, N. Al-Qirim and I. Taleb, "Cloud Workflow Resource Shortage Prediction and Fulfillment Using Multiple Adaptation Strategies," the 11^{th} IEEE International Conference on Cloud Computing (IEEE CLOUD 2018), San Francisco, 2018.

(5) M. A. Serhani, H. El-Kassabi, N. Al-Qirim and A. N. Navaz, "Towards a Multi-Model Cloud Workflow Resource Monitoring, Adaptation, and Prediction," the 12^{th} IEEE International Conference On Big Data Science And Engineering (IEEE BigDataSE-18), New York, 2018.

(6) H. T. El Kassabi and M. A. Serhani, "De-centralized Reputation-based Trust Model to Discriminate Between Cloud Providers Capable of Processing Big Data," the 6^{th} IEEE International Congress on Big Data (BigDatavCongress), IEEE, Honolulu, 2017, pp. 266 - 273.

(7) H. El-Kassabi, M. A. Serhani, C. Bouhaddioui, and R. Dssouli, "Trust Assessment-based Multiple Linear Regression for Processing Big Data over Diverse Clouds," the International Conference on Emerging Technologies for Developing Countries, Springer, Marrakech, 2017, pp. 99 - 109.

(8) M. A. Serhani, H. T. El Kassabi, and I. Taleb, "Quality Profile-based Cloud Service Selection for Fulfilling Big Data Processing Requirements," the 7^{th} IEEE International Symposium on Cloud and Service Computing (SC2), IEEE, Kanazawa, 2017, pp. 149 - 156.

(9) M. A. Serhani, H. T. El Kassabi, I. Taleb, and A. Nujum, "An Hybrid Approach to Quality Evaluation Across Big Data Value Chain," the 5^{th} IEEE International Congress on Big Data (BigData Congress), IEEE, San Francisco, 2016, pp. 418 - 425.

(10) H. T. El Kassabi, I. Taleb, M. A. Serhani, and R. Dssouli, "Policy-based QoS Enforcement for Adaptive Big Data Distribution on the Cloud," the Second IEEE International Conference on Big Data Computing Service and Applications (BigDataService), IEEE, Oxford, 2016, pp. 225 - 233.