Long-term IaaS Cloud Service Selection

SHEIK MOHAMMAD MOSTAKIM FATTAH

Supervisor: Prof. Athman Bouguettaya
Auxiliary supervisor: Dr. Sajib Mistry

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

5 November 2021

Declaration

I certify that except where due acknowledgment has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; and, any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Sheik Mohammad Mostakim Fattah
School of Computer Science, Faculty of Engineering
University of Sydney
November 5, 2021

Acknowledgements

I want to take this opportunity to express my sincere gratitude to my supervisor Prof. Athman Bouguettaya for his extraordinary supervision. Prof. Bouguettaya has provided me with unprecedented support, guidance, and encouragement. I am deeply thankful for his patience, kindness, and thoughtfulness during his supervision. His contributions to the development of my critical thinking, writing and presentation skills are exceptional. Prof. Bouguettaya is one of the most inspiring mentors I have ever had.

I am exceedingly grateful to my co-supervisor Dr. Sajib Mistry for his outstanding support, guidance, and assistance. His unparallel supervision has made this research possible. In addition, he helped me immensely in developing my analytical and writing skills. I am very grateful to Dr. Mistry for each achievement during my Ph.D.

I am very thankful to each member of the Sensors, Clouds, and Services Lab. They helped me immensely during my Ph.D. by providing continuous feedback and advice. My research skill significantly developed because of their constant support. I found outstanding moral support from them during this challenging journey. It was an honor to work with them.

I am incredibly grateful to every teacher who taught me during any stage of my academic life. They are the source of my continuous inspiration, guidance, and support.

I want to acknowledge and thank Janelle Cruickshank for her editorial assistance. She provided excellent proofreading support during the preparation of this thesis.

I want to thank my wife Chadni Islam for her patience and sacrifices during my Ph.D. She constantly supported me to reach the final stage of this journey. I want to thank my sister Fatema Begum for her unconditional love and support. Furthermore, I want to thank all my relatives who supported me at any stage of my life. Finally, I would like to dedicate my thesis to my mother. Her sacrifices and contributions to every step of my life are invaluable.

Authorship Attribution Statement

The main results presented in this thesis were published or submitted as follows:

- S. M. M. Fattah, A. Bouguettaya and S. Mistry. Long-term IaaS Selection using Performance Discovery. IEEE Transactions on Services Computing (TSC), 2020. (**Preprint**) **(CORE 2020 Ranking: A*)** Presented in Chapter 3. I designed the main techniques, conducted the experiment and wrote the paper.
- S. M. M. Fattah and A. Bouguettaya. Event-based Detection of Changes in IaaS Performance Signatures. IEEE International Conference on Services Computing (SCC), 2020, pp. 210-217, IEEE. **(CORE 2020 Ranking: A)** Presented in section 6.2. I designed the main techniques, conducted the experiment and wrote the paper.
- S. M. M. Fattah, A. Bouguettaya and S. Mistry. Signature-based Selection of IaaS Cloud Services. IEEE International Conference on Web Services (ICWS), 2020, pp. 50-57, IEEE. **(CORE 2020 Ranking: A)** Presented in Chapter 5. I designed the main techniques, conducted the experiment and wrote the paper.
- S. M. M. Fattah, A. Bouguettaya and S. Mistry. Long-Term IaaS Provider Selection Using Short-Term Trial Experience. IEEE International Conference on Web Services (ICWS), 2019, pp. 304-311, IEEE. **(CORE 2019 Ranking: A)** Presented in Chapter 4. I designed the main techniques, conducted the experiment and wrote the paper.
- S. M. M. Fattah, A. Bouguettaya and S. Mistry. A CP-Net Based Qualitative Composition Approach for an IaaS Provider. The 19th International Conference on Web Information Systems Engineering (WISE), 2018, pp. 151-166. Springer. **(CORE 2018 Ranking: A)** Presented in section 2.6.2, I designed the main techniques, conducted the experiment and wrote the paper.
- S. Mistry, S. M. M. Fattah, and A. Bouguettaya. Sequential Learning-based IaaS Composition. ACM Transactions on Web (TWEB), 2021. (**Accepted**) Presented in section 2.6.2, I co-designed the main techniques with Sajib Mistry and wrote several of the paper.

- S. M. M. Fattah and A. Bouguettaya. IaaS Performance Signature Change Detection. IEEE Transactions on Services Computing (TSC), 2021, IEEE. (**submitted**) **(CORE 2021 Ranking: A*)** Presented in section 6.4, I designed the main techniques, conducted thwe experiment and wrote the paper.
- S. M. M. Fattah and A. Bouguettaya. IaaS Signature Change Detection with Performance Noise. 2021 International Conference on Service Oriented Computing (ICSOC), 2021, Springer. (**Accepted**) **(CORE 2021 Ranking: A)** Presented in section 6.3, I designed the main techniques, conducted the experiment and wrote the paper.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Sheik Mohammad Mostakim Fattah
School of Computer Science, Faculty of Engineering
University of Sydney
November 5, 2021

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Athman Bouguettaya
School of Computer Science, Faculty of Engineering
University of Sydney
November 5, 2021

Abstract

There are two primary subscription models for IaaS cloud services: a) pay-as-you and b) reservation. Reservation-based subscriptions are typically offered for a long-term period such as 1 to 3 years. Long-term subscriptions are typically cost-efficient than short-term subscriptions for consumers who need services for a long-term period. Large organizations such as airline companies, banks, and research institutes tend to utilize IaaS services on a *long-term basis* for economic reasons. The performance of IaaS services is a key criterion to consider when selecting a service for a long-term. Selecting a service that may exhibit poor performance in the future may cause a significant *loss of revenue* for a business organization. Most IaaS providers, however, are reluctant to provide detailed information about their long-term service performance. This research aims at developing a long-term IaaS cloud service selection framework where IaaS providers reveal limited performance information about their services. First, we propose a novel framework to find the closest match of IaaS cloud service according to a consumer's long-term QoS requirements. The proposed framework leverages free short-term trials to discover the unknown QoS performance information. A temporal skyline-based filtering method is proposed to select candidate services for short-term trials. A novel cooperative long-term QoS prediction approach is introduced that utilizes past trial experiences of similar consumers using a workload replay technique. We propose a new trial workload generation model that estimates a provider's long-term performance in the absence of past trial experiences. The confidence of the prediction is measured based on the trial experience of the consumer. Next, we propose a new long-term IaaS cloud service selection framework that utilizes a consumer's trial experience and the performance fingerprints of IaaS cloud services for the long-term selection. We design a novel equivalence partitioning-based trial strategy to discover the unknown QoS performance variability of IaaS cloud services. A trial experience transformation method is proposed to estimate the long-term performance of an IaaS cloud service. Next, we introduce a signature-based IaaS

cloud service selection framework that leverages a new significance-based trial scheme and a signature technique to discover a service's long-term performance. Next, we propose a novel event-based change detection approach to manage changes in IaaS performance signatures. A new anomaly-based event detection technique is proposed to detect changes in long-term IaaS performance behavior over time. We then propose an IaaS performance noise model to identify noise and actual changes in IaaS performance accurately. A novel categorical signature-based approach is proposed to detect the long-term performance changes using the proposed performance noise model. Finally, we introduce a signature change detection framework that leverages a sliding window-based approach and a Signal-to-Noise ratio-based approach to detect long-term changes in IaaS performance signatures. We have conducted a set of experiments based on real-world datasets to evaluate the proposed frameworks. The proposed long-term selection framework achieved almost 92% ranking accuracy. The signature-based IaaS cloud service selection framework achieved 96% ranking accuracy. The proposed changed detection frameworks achieved up to 90% change detection accuracy.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Cloud computing has become a pivotal technology of choice for most organizations to establish and manage their in-house IT infrastructures [Chaisiri *et al*., 2012]. There are two key influential factors behind this paradigm shift. One factor is the rapidly increasing maintenance costs of in-house IT infrastructures to manage unpredictable service demand. The cloud provides economic efficiency for the fluctuating service demand. Another factor is the non-adaptive nature of local IT infrastructures to rapid changes in business requirements. Cloud consumers can rent service according to their dynamic business requirements. From a cloud provider's perspective, low-cost computational resources, data storage, and higher network bandwidth offer significant economies of scale. Cloud computing provides a uniform way to access and manage computational resources, platforms, and software.

Cloud computing utilizes the service computing paradigm to deliver cloud services. The most common forms of cloud services are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Service computing is the preferred mode of providing cloud computing solutions [Armbrust *et al*., 2010]. Service computing is a fairly new paradigm that considers service as the principal component [Bouguettaya *et al*., 2017]. A service is an abstraction over data to make it useful. Services can be described, discovered, reused, and composed using its functional and non-functional properties. Service computing utilizes the power and the simplicity of services to build large-scale distributed applications.

Infrastructure-as-a-Service (IaaS) is a primary service delivery model in the cloud. IaaS models typically offer computational resources such as CPU, memory, storage, and network bandwidth in the form of various cloud services such as Virtual Machines (VMs), Virtual Storage (VS), and Virtual Private Networks (VPNs). Amazon AWS, Google Compute Engine,

and Microsoft Azure are examples of popular IaaS cloud services. An IaaS service consists of two parts: *functional* and *non-functional*. Functional attributes are set based on the purpose of the service such as computing, data storing, and networking. Non-functional attributes are the Quality of Service (QoS) attributes such as availability, response time, and throughput. QoS attributes help a consumer to select the *best-performing* services from a large number of functionally similar services [Yu and Bouguettaya, 2010].

The IaaS model offers an easier, faster, and more cost-effective alternative to manage an organization's in-house IT infrastructure in the cloud. There are two primary subscription models for IaaS cloud services: a) pay-as-you-go and b) reservation. Reservation-based subscriptions are typically offered for a long-term period such as 1 to 3 years. Long-term subscriptions are typically more cost-effective than short-term subscriptions for consumers who need services for a long-term period. Large organizations such as airline companies, banks, and research institutes tend to utilize IaaS services on a *long-term basis* for economic reasons [Ye *et al*., 2016]. Leading IaaS cloud providers such as Amazon, Google, and Microsoft offer significant discounts on long-term subscriptions. For instance, Amazon advertises up to 72% discount on its long-term subscriptions compared to its on-demand subscriptions[1]. Subscribing to an IaaS service for a long period is an important *business decision* for most organizations due to economic reasons [Mazzucco and Dumas, 2011]. *The focus of our research is the selection of an IaaS service for a long-term period.*

The performance of IaaS services is a key criterion to consider when selecting a service for a long-term [Iosup *et al*., 2014]. Selecting a service that may exhibit poor performance in the future may cause a significant *loss of revenue* for a business organization. The knowledge of the IaaS performance is therefore essential during the long-term selection. The IaaS performance is typically measured in terms of its Quality of Service (QoS) such as response time, throughput, and availability. A consumer is generally concerned with two key aspects of the IaaS performance during the long-term subscription. First, how well would a service perform for their long-term application workload? The performance of IaaS services usually varies depending on the workloads [Iosup *et al*., 2011]. For instance, a service may provide

[1] https://aws.amazon.com/ec2/pricing/reserved-instances/

better performance for compute-intensive workload than I/O-intensive workload. Second, how might the performance vary over time? Most providers typically operate in a multi-tenant environment. Therefore, the performance of their services varies over time.

Most IaaS providers are typically reluctant to reveal much information about the long-term performance of their services. The key reasons for such behavior are market competition, business secrecy, and conflicts of interest [Fattah and Bouguettaya, 2020a]. As a result, selecting an IaaS service that will be the best match for a consumer's long-term performance requirements becomes a challenge largely due to the following two key factors:

a) Incomplete IaaS advertisements: IaaS providers reveal *limited* and *short-term* QoS information in their advertisements [Wang *et al*., 2018]. IaaS advertisements typically contain a limited number of QoS attributes. For instance, disk read/write throughput, memory bandwidth, and availability are unavailable in most advertisements [Iosup *et al*., 2011]. The advertised performance information may not be representative for a long time. For instance, a consumer may want to know how the service performs in December, yet the advertised performance information is for June. Additionally, the advertised information may not be *helpful* in understanding service performance due to the lack of detailed information. For example, EC2 instances have different types of virtual CPUs (vCPUs). According to AWS advertisements[2], each vCPU can be a thread of an Intel Xeon core, an AMD EPYC core, or AWS Graviton processor. Estimating the vCPU's actual performance is difficult from such limited information [Feitelson, 2002]. Providers often advertise an average or maximum performance of their services. For instance, the network performance of some EC2 instances has a data transfer rate of up to 10-gigabits. Existing studies show that providers often fail to offer the *promised* QoS performance in the long-term period [Ye *et al*., 2016]. Therefore, relying only on IaaS advertisements is not sufficient to select a service for a long-term period.

b) Limited performance history: IaaS providers usually do not share detailed service performance history publicly due to *market competition* and *business secrecy* [Fattah *et al*., 2019]. There exist third party data collectors such as CloudHarmony, and CloudSpectator that provide summarized results or insights on the performance of cloud services. These results

[2] https://aws.amazon.com/ec2/instance-types/

are usually not fit for further analysis due to the reduced dimensions in QoS attributes and time [Li *et al*., 2010]. For instance, CloudHarmony mainly monitors network availability and does not provide any insight on response time or throughput. Moreover, collectors often use proprietary benchmarks but reveal limited information about the benchmarking process.

To the best of our knowledge, existing studies focus mainly on short-term IaaS provider selection approaches [Mistry *et al*., 2016b]. These approaches are typically inapplicable for the long-term selection due to incomplete advertisements and long-term performance variability [Fattah *et al*., 2020a]. **The aim of our research is to propose a novel framework to select the best IaaS cloud service according to a consumer's long-term performance requirements where service providers reveal limited performance information.**

1.1 Research Objective and Key Challenges

An effective way to deal with the limited performance information is to leverage free trials [Wang *et al*., 2018]. Most IaaS providers offer *free short-term trials* and encourage potential consumers to test their application workload in the cloud. For instance, Microsoft Azure offers a one-month trial period for a limited number of services to its potential consumers. Amazon AWS offers a one-month free trial on Amazon Lightsail services for 750 hours. IaaS consumers may get a first-hand experience before subscribing to a service for a long-term period. Consumers may run their application workload on different IaaS cloud services and compare their performances. The trial experience of consumers may have a considerable impact on the IaaS provider selection process [Zhu and Chang, 2014]. To the best of our knowledge, existing studies do not consider the effective utilization of free trial periods for long-term IaaS cloud service selection. **We aim to utilize free trial periods to uncover unknown QoS performance information of IaaS cloud services for long-term selection.** However, making a long-term decision based on a short-trial is challenging. A consumer's trial experience may depend on several factors such as trial workloads, the time of the trial, and the provider's QoS management policy [Scheuner and Leitner, 2018b]. The experience

from an *unplanned* short trial may not provide the complete information required for long-term service selection. We identify the following key challenges in the long-term IaaS cloud service selection using the short-term trial:

- **Number of Candidate IaaS Providers**: A large number of IaaS providers may satisfy a consumer's long-term requirements. The performance of these providers may vary considerably depending on their business strategies and infrastructures. Performing trial with every eligible IaaS is *practically* infeasible. Therefore, an effective candidate selection approach is required that will select the best candidates for the trial.
- **Temporal Restrictions**: A consumer may utilize diverse types of applications and different amount of workload over a long period. Performing necessary and sufficient amount of testing in the cloud in a short trial may not always be possible. An *unplanned* utilization of such short-term trial periods may not properly reflect the actual performance of the provider. For example, if the workloads of a consumer have a long-tailed distribution, a one-month trial with a balanced request distribution may not divulge the true performance of long-tailed workloads. Some IaaS providers (e.g., Amazon) offer long-term trials for only a few services. A consumer may not be able to *wait for such a long time* to make the selection. An effective utilization of the free trial is essential to make an informed decision.
- **Performance Variability**: It is challenging to predict long-term performance from a short trial without any additional information about the long-term performance variability of the service [Wang *et al.*, 2018]. Most IaaS providers usually operate in multi-tenant environments [Fattah *et al.*, 2020a]. IaaS performance is highly time-dependent [Iosup *et al.*, 2014]. The performance discovered in a one month trial period may not *reflect* the actual performance of a provider for the rest of the year.

1.2 Thesis Contributions

The primary focus of this work is to help a consumer to make an informed selection for a long-term period. We initially propose a novel performance-based long-term IaaS cloud service selection approach. In this approach, we assume that the experience of free trial

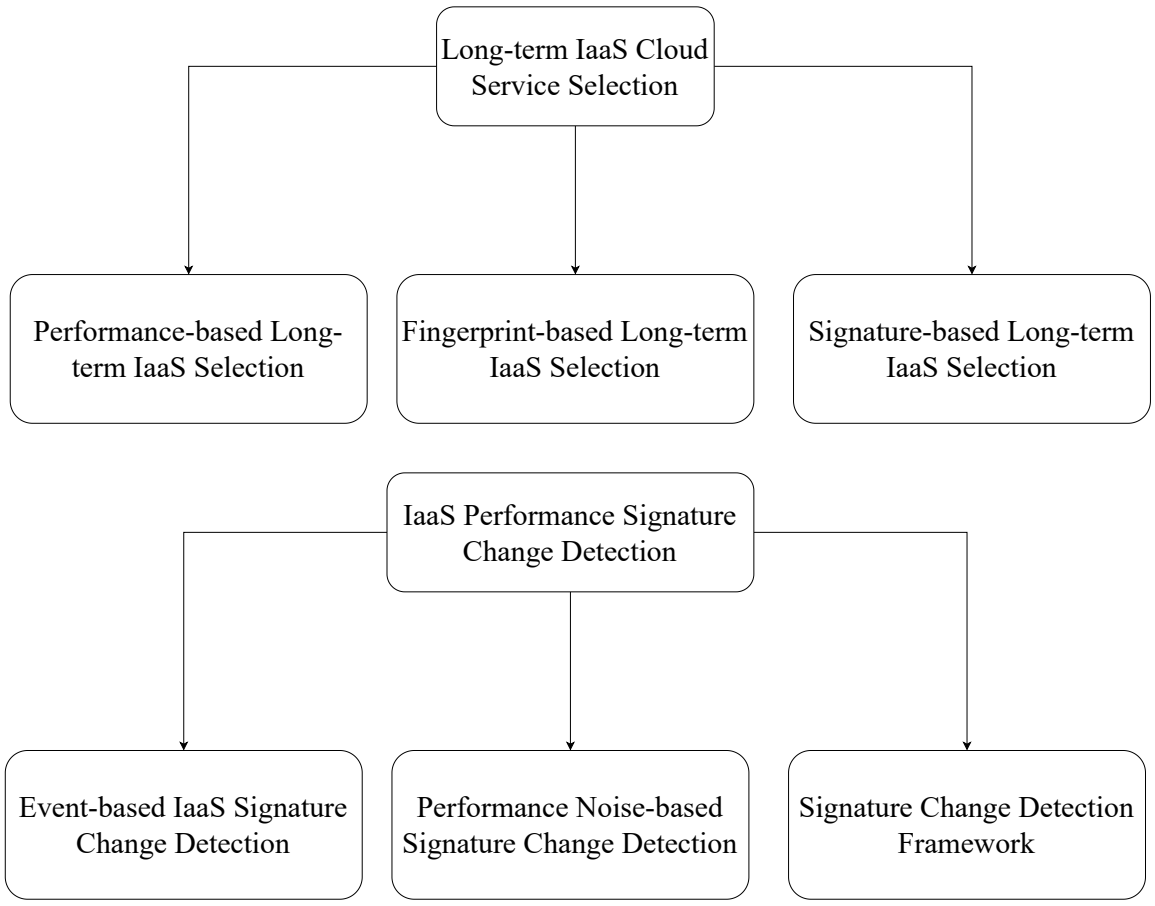


FIGURE 1.1: Key contributions

users is publicly available. We utilize the experience of free trial users to discover an IaaS service's long-term performance. We then introduce a fingerprint-based long-term IaaS selection approach where we assume that a service's long-term performance is represented as its performance fingerprint. We utilize the performance fingerprint of a service and the free trial experience of a consumer for the long-term selection. We then extend the concept of performance fingerprints to propose a new concept called IaaS performance signature that represents a service's long-term performance behavior in a privacy preserving manner. We propose a signature generation technique that is utilized for the long-term IaaS selection. Once the signature of a service is generated, it is important to keep the signature updated to reflect the service's current performance behavior. An IaaS service's performance behavior may change over time due to a number of reasons [Mi *et al.*, 2008]. For instance, a provider may upgrade its infrastructure or change its multi-tenant management policy resulting in a change

in service performance [Leitner and Cito, 2016]. It is important to detect changes in IaaS performance *as early as* possible to keep the signature updated. Therefore, we propose a set of signature change detection approaches. Figure 1.1 shows the key contributions in this thesis. The key contributions are divided into two sections. The first section focuses on the long-term IaaS selection where we introduced a set of IaaS cloud service selection framework. In the second section, we have focused on the detection of changes in long-term IaaS performance as the knowledge of IaaS performance is essential for the long-term selection. We discuss the key contributions in the following subsections.

1.2.1 Long-term IaaS Cloud Service Selection

1.2.1.1 Performance-based Long-term IaaS Selection

We propose a novel framework to select IaaS cloud services according to a consumer's long-term performance requirements. The proposed framework leverages free short-term trials to discover the unknown QoS performance of IaaS services. We design a temporal skyline-based filtering method to select candidate IaaS services for the short-term trials. A novel cooperative long-term QoS prediction approach is developed that utilizes past trial experiences of similar consumers using a workload replay technique. We propose a new trial workload generation model that estimates a service's long-term performance in the absence of past trial experiences. The confidence of the prediction is measured based on the trial experience of the consumer.

1.2.1.2 Fingerprint-based Long-term IaaS Selection

We propose a novel approach to select the best IaaS services for a long-term period where IaaS providers reveal limited performance information. The proposed approach leverages a consumer's short-term trial experiences for long-term selection. We design a novel equivalence partitioning based trial strategy to discover the temporal and unknown QoS performance variability of an IaaS servce. The consumer's long-term workloads are partitioned into multiple Virtual Machines in the short-term trial. We propose a performance fingerprint

matching approach to ascertain the confidence of the consumer's trial experience. A trial experience transformation method is proposed to estimate the actual long-term performance of the service.

1.2.1.3 Signature-based IaaS Cloud Service Selection

We propose a novel approach to select IaaS cloud services for a long-term period where the service providers offer *limited* QoS information. The proposed approach leverages free short-term *trials* to obtain the previously undisclosed QoS information. A new significance-based trial scheme is proposed using frequency distribution analysis to test a consumer's long-term workloads in a short trial. We introduce a novel IaaS *signature* technique to uniquely identify the variability of a provider's QoS performance. A Signature-based QoS Performance Discovery (SPD) algorithm is proposed that leverages the combination of free trials and IaaS signatures.

1.2.2 IaaS Performance Signature Change Detection

1.2.2.1 Event-based IaaS Signature Change Detection

We propose a novel ECA (Event-Condition-Action) approach to manage changes in *IaaS performance signatures*. The proposed approach relies on the detection of *anomalous* performance behavior in the context of IaaS performance signatures. A novel anomaly-based event detection technique is proposed. It utilizes the experience of free trial users to detect potential changes in IaaS performance signatures. A signature change detection technique is proposed using the cumulative sum control chart analysis. Additionally, a self-adjustment method is introduced to improve the accuracy of the proposed approach.

1.2.2.2 Performance Noise-based Signature Change Detection

We propose a novel framework to detect changes in the performance behavior of an IaaS service. The proposed framework leverages the concept of the IaaS signature to represent an IaaS service's long-term performance behavior. A new type of performance signature

called categorical IaaS signature is introduced to represent the performance behavior more accurately. A novel performance noise model is proposed to accurately identify both IaaS performance noise and changes in the performance behavior of an IaaS service.

1.2.2.3 Signature Change Detection Framework

We propose a novel framework to detect long-term changes in the performance behavior of an IaaS service. The proposed framework represents long-term performance behavior using IaaS signatures. The proposed approach leverages time series similarity measures and a sliding window technique to distinguish between noise and changes in IaaS performance. An SNR-based (Signal-to-Noise ratio) approach is introduced to improve the efficiency of the proposed framework.

1.3 Thesis Outline

We present the research contributions in two parts. The first part presents our work on long-term IaaS cloud service selection and the second part presents our work on IaaS performance change detection. The thesis is organized as follows:

- Chapter 2 discusses works that are closely related to our research. In this chapter, we discuss existing IaaS cloud service selection approaches and key differences between the existing work and the proposed approaches in this work.
- In Chapter 3, we present a novel IaaS cloud service selection that proposes a cooperative approach for the long-term IaaS performance discovery. First, we introduce a temporal skyline-based approach to select candidate IaaS cloud services for the free trials. Then, we introduce a long-term IaaS cloud service selection approach that leverages the experience of free trial users for the long-term service selection. Finally, we introduce a long-term service selection approach that assumes there is no historical information of the free trial users.
- Chapter 4 introduces a new concept called IaaS performance fingerprints. A performance fingerprint-based long-term selection framework is proposed in this chapter. First, we

introduce an equivalence partitioning-based approach that effectively utilizes free trials to understand the service performance for a consumer's long-term workload. Then, we discuss how to utilize the free trial experience and IaaS performance fingerprints to make a long-term selection.

- In Chapter 5, we extend the IaaS performance fingerprint concept and propose the concept of IaaS performance signatures where we consider the privacy of free trial users during the signature generation. In this chapter, we discuss how IaaS performance signatures can be generated efficiently and effectively and can be utilized to help a consumer in making an informed long-term selection.
- In Chapter 6, we introduce a set of signature change detection approaches. First, we introduce an ECA-based approach to detect changes in IaaS performance behavior. The proposed approach leverages a signature-based performance anomaly detection technique to detect changes in performance. We then introduce a novel IaaS performance model to detect changes in IaaS signature. A performance-noise based signature change detection framework is proposed. Finally, we introduce a signature change detection framework that utilizes sliding-window and SNR-based techniques to detect changes in IaaS signature.
- In Chapter 7, we conclude this thesis and briefly discuss the limitations and future work.

CHAPTER 2

Related Work

2.1 Introduction

IaaS cloud service selection is a topical research challenge in the area of cloud computing [Li *et al*., 2010]. Several IaaS cloud service selection approaches are proposed to select the optimal IaaS service based on the consumer's required QoS performance. A cloud comparison approach called CloudCom is proposed to help consumers select a cloud provider that fits their needs [Li *et al*., 2010]. CloudCom addresses three key services, i.e., elastic computing, persistent storage, and networking services, along with their metrics in the IaaS cloud. The performance of each service is measured based on the most relevant QoS attributes that may affect consumer applications directly. The IaaS cloud selection problem is modeled as a multi-criteria decision-making problem in [ur Rehman *et al*., 2012]. A variation-aware cloud selection method is proposed based on collaborative filtering techniques. We categorize the existing IaaS cloud service selection approaches into the following two groups:

(1) **IaaS selection for a short-term period:** A common approach to discover IaaS performance is to conduct short-term trial using representative application and micro-benchmarks [Scheuner and Leitner, 2019]. A generator approach is proposed to automate performance testing in the IaaS cloud [Jayasinghe *et al*., 2012]. The proposed work aims at reducing human errors and maximizing efficiency for large scale distributed experiments. A cloud benchmark outline is proposed [Binnig *et al*., 2009] that claims that traditional benchmarks (e.g., TPC benchmarks) for computer systems are not suitable for cloud performance discovery. Several studies perform experiments to understand the cloud performance for scientific application [Sadooghi *et al*., 2015]. These experiments

find that the performance of the IaaS cloud is considerably low for scientific computing. These studies focus primarily on short-term IaaS performance and do not consider the long-term consumer requirements.

(2) **IaaS selection for a long-term period:** The long-term IaaS selection approach is considered in several studies [Labbaci *et al.*, 2017; Ye *et al.*, 2012; Mistry *et al.*, 2016a]. A QoS-aware IaaS selection approach is proposed using a multi-dimensional time series. The proposed approach selects IaaS providers based on the consumer's long-term economic models. A qualitative approach is proposed using Conditional Preference Networks to select long-term IaaS providers [Mistry *et al.*, 2016a]. A QoS-aware approach is proposed to select and compose long-term cloud services using meta-heuristic approaches [Liu *et al.*, 2015]. These approaches assume that long-term performance information is given for the selection.

To the best of our knowledge, most existing IaaS selection approaches address the short-term selection, and existing long-term IaaS selection approaches are not directly applicable when available QoS performance information is limited or absent. Therefore, we leveraged existing short-term performance discovery to develop a long-term performance discovery in this thesis. Existing short-term performance discovery approaches typically include the following steps: a) Representative Workload Generation, b) Performance Benchmarking, c) QoS Prediction and Ranking, and d) Long-term IaaS Cloud Service Selection. In the following sections, we briefly discuss the state-of-the-art related to each step of the performance discovery.

2.2 Representative Workload Generation

The performance of any system often depends on its workload. Workload generation, therefore, is a significant step of performance engineering. Cloud is highly dynamic due to its multi-tenant nature. A single physical machine is shared by multiple users. The performance of the physical machine may depend on the workloads generated by its users. Hence, a user may notice performance variability of a VM depending on its workload. For example, a VM may perform better if its workload is evenly distributed over time. The VM may perform differently

depending on different types of workload, such as compute-intensive, network-intensive, or I/O-intensive.

Representative workload generation is an important part of performance evaluation [Scheuner and Leitner, 2018a], and is necessary to produce reliable results. Representative workload can be generated by collecting real-world data and creating statistical models that capture important features of real-world workloads [Feitelson, 2002] [Calzarossa *et al*., 2016a]. Workload modeling requires a deep understanding of cloud workloads and their features. Cloud workload characterization is studied extensively in the existing literature [Calzarossa *et al*., 2016b].

Most of the workload characterization studies are performed from a provider's perspective to enable effective VM allocation and consolidation, live migration, resource scheduling and so on. A study on "Google Cluster Trace" is conducted to create a statistical profile of jobs, or clusters of workload patterns [Alam *et al*., 2016]. A trace analysis on "Google Cluster Trace" is performed to understand the challenges in developing effective cloud-based resource schedulers [Reiss *et al*., 2012]. A multiple time series based workload characterization approach is proposed to predict future VM workloads [Khan *et al*., 2012]. The proposed approach searches repeatable workload patterns using cross-VM workload correlations. The correlations are generated from the dependencies among applications that are running in different VMs. Resource planning and management in the cloud require an understanding of the application workload characteristics. An application workload characterization approach is proposed to understand the effect of virtualization on the application performances [Wang *et al*., 2014]. The proposed work characterizes a three-tier application's performance on virtualized and non-virtualized environment. The resource demands at three-tier servers are compared to understand the effect of virtualization. An IaaS cloud workload characterization approach is proposed for capacity planning and performance management [Mahambre *et al*., 2012]. The relationship between different workload metrics such as CPU and memory requirements are modeled across a set of workloads. These relationships are analyzed and characterized to enable VM placement, migration, load balancing and so on. A Markovian workload characterization model is proposed to predict the performance of physical machines

TABLE 2.1: Representative workload generation techniques

Methods	**Used in**	**Strengths**	**Weaknesses**
Workload Replay	[Kessler *et al.*, 1994; Kalayappan *et al.*, 2020]	Real workloads	difficult to customize, often system specific
Piecewise Aggregated Approximation	[Mackiewicz and Ratajczak, 1993]	Easy to implement, efficient for dimensionality reduction	Real workload may not get tested
Symbolic Aggregated Approximation	[Huang *et al.*, 2016]	Significant compression ratio, good performance	Substantial information loss
Principal Component Analysis	[Mackiewicz and Ratajczak, 1993]	Finds important components of the workload	Non-parametric analysis
Perceptually Important Points	[Burtini *et al.*, 2013]	Effective compression ratio, can be used for pattern identification	Usually applicable for simple workload representation
Multiple Time Series Approach	[Khan *et al.*, 2012]	Considers correlation between different workload components	Applicable for only continuous workload representation

deployed on the cloud [Pacheco-Sanchez *et al.*, 2011]. The proposed approach utilizes *Markovian Arrival Process (MAP)* and the MAP/MAP/1 queuing model to capture the time-varying characteristics of common workloads such as heavy-tail distribution.

An IaaS consumer needs to generate representative workloads to evaluate the performance of a VM. The workload of a consumer depends on the type of its application. Different types of applications may produce workloads with different types of characteristics. Understanding these characteristics is important when measuring the effect workload attributes on the performance of VM. It is also important for resource planning and management. Finding real world traces of such applications is difficult for a consumer. Several studies propose different approaches to generate representative workloads for different applications. A number of synthetic workload generation tools and benchmarks is developed to model real world workloads. For example, SPECweb99 [Kant and Won, 2000] and SURGE [Barford and Crovella, 1998] generate workloads for web servers. A synthetic workload generation method is presented in [Bahga and Madisetti, 2011]. The proposed method performs automated workload characterization and modeling for different types of applications to extract their features. Two different methods are described for capturing the cloud application workloads: application

benchmarking and workload modeling. A synthetic workload generator is introduced, which utilizes the benchmark and workload model specification to generate representative workload for a particular application. A cloud workload specification language (GT-CWSL) is proposed to express the workload specification in a structured manner. A code generator is developed to generate workload specification for the synthetic workload generator. The proposed synthetic workload generation methodology is evaluated by modeling RUBis and TPC-W benchmarks. The results show that the synthetic workload successfully mimics the real-world workloads. We have summarized existing representative workload generation approaches in Table 2.1.

To the best of our knowledge, existing representative workload generation techniques from a consumer's perspective are mostly designed for short-term performance evaluation where a consumer's long-term service demand is not considered. We leverage existing representative workload generation techniques for the long-term performance evaluation.

2.3 Performance Benchmarking

Benchmarking is an approach that is used to evaluate the performance of a computer system with a fixed workload and configurations [Scheuner and Leitner, 2019]. The main purpose of benchmarking is to facilitate a consumer's informed decision-making. A consumer may compare between different systems based on the result of benchmarking. There exist many standard benchmarks targeting different systems and hardware; for instance, TPC-C is an established benchmark for transactional database systems [Avula and Zou, 2020]. SPEC CPU and Geekbench are examples of CPU benchmarks. StressNg is a network stress testing benchmark. The first step in designing a benchmark is to define its objectives, on which the results of a benchmark depend. A systematic way to define the objectives is to analyze the properties and constraints of the systems that need to be benchmarked [Hwang *et al.*, 2015]. A set of the priorities that should be optimized during the benchmarking process needs to be defined. Benchmarking provides a way to define these priorities and constraints in a structured manner.

Benchmarking in cloud is different from the traditional benchmarking process. The traditional benchmarking process does not consider the nature of cloud such as multi-tenancy, elasticity, hardware heterogeneity, and over-commitment of resources [Scheuner and Leitner, 2018b]. Determining the actual performance of VMs can be challenging due to these factors. Some initial ideas about how to benchmark cloud services are presented in [Binnig *et al.*, 2009]. The paper argues that traditional benchmarks such as TPC benchmarks are not suitable for cloud benchmarking, as these benchmarks are typically deployed in a managed environment with fixed configurations of software and hardware. Primary metrics of these benchmarks such as average performance or total cost of ownership assume a system that does not change over time. These benchmarks require the system to describe the ACID properties as they focus on transactional database systems.

A key advantage of cloud services is the elasticity over the traditional systems. The provider can automatically allocate new resources or remove unused resources depending on the workloads in a pay-as-you-go billing model. In the case of network failure, cloud storage providers often cannot offer strong consistency and high availability together as they use multiple data centers to support high availability and fault tolerance. Requirements for new cloud benchmarks are derived by analyzing the most important characteristics of cloud services [Binnig *et al.*, 2009]. The proposed work analyzes the TPC-W benchmark and shows why it does not satisfy the requirements of cloud service benchmarking.

Several studies propose different approaches to benchmarking the IaaS cloud. Some studies analyze the effect of virtualization on the performance of cloud applications. A comparative study is carried out to understand the impact of virtualization by benchmarking applications in terms of memory usage, network bandwidth, disk bandwidth and CPU performance [Langer and French, 2011]. A number of VMs is deployed and tested using some benchmarks and are then compared with the performance of an application on the physical machine. The results show that virtualization has a significant impact on the performance of an application. The performance of read operation from local disk and floating point operation is negatively impacted. A new custom benchmarking method is proposed to find the actual performance of different cloud providers considering the service type running on VMs [Scheuner and Leitner,

2018b]. The proposed method consists of multiple benchmark suite to gain reliable and comparable results based on application requirements. A process of developing a custom-tailored benchmark suit is described that selects benchmarks based on application requirements. The results show that the performance indicator advertised by IaaS providers is not helpful to find the actual cost/performance ratio.

A number of studies engage in performance monitoring and analysis to evaluate the performance of IaaS cloud services. Some of them analyze the performance of cloud services for special applications such as many-task scientific computing [Sadooghi *et al*., 2017; Iosup *et al*., 2011; Ostermann *et al*., 2009]. Several studies analyze the performance variability and predictability in public cloud computing and production clouds [Iosup *et al*., 2011; Leitner and Cito, 2016]. A number of commercial websites and tools are developed to monitor and compare cloud services such as CloudHarmony, CloudStatus, and Amazon Cloud Watch.

Several studies determine the QoS performance of IaaS providers by deploying VMs in the cloud. The performance behavior of small instances for service-oriented applications in Amazon EC2 is studied in [Dejun *et al*., 2010]. The proposed study benchmarks virtual instances by generating different types of workload pattern and analyzing the performance in terms of mean response time. An extensive study on the performance variance of Amazon EC2 is provided in [Schad *et al*., 2010]. The study addresses that performance unpredictability in the cloud is a significant issue for many users and often considered a key obstacle in the cloud adaption. The study finds that Amazon EC2 shows a high variance in its performance. The performance of clouds for scientific computing is analyzed using micro-benchmarks and kernels on Amazon EC2 in [Ostermann *et al*., 2009; Iosup *et al*., 2011]. The proposed study observes that tested clouds are not suitable for scientific computing due to their performance variance and low reliability. A generator approach called Expertus is proposed to automate performance testing in IaaS clouds [Jayasinghe *et al*., 2012].

To the best of our knowledge, existing benchmarking approaches are primarly designed for short-term performance evaluation of IaaS services. Therefore, most approaches do not consider the performance variability during the benchmarking process. In this thesis, we

leverage the concept of micro-benchmarking and the application bechmarking approaches for the long-term performance evaluation.

2.4 QoS Prediction and Ranking

The *experiences of existing users* are utilized to predict QoS performance of cloud providers [Wang *et al.*, 2019; Yang *et al.*, 2018; Tang *et al.*, 2016]. The rank of the providers are then measured based on the predicted performance. The collaborative filtering is a well-known approach to predict QoS performance of a service based on the experience of existing users. A collaborative filtering (CF) approach is proposed to predict QoS values based on historical QoS information provided by existing users [Wang *et al.*, 2019]. The proposed approach finds similar users based on a user's QoS requirements and utilizes similar users' experiences to predict personalized QoS ranking. A location-aware CF approach is proposed to predict missing QoS parameters for web service recommendation [Yang *et al.*, 2018]. This study suggests that the location of a user has a remarkable impact on the value of QoS attributes such as availability, response time, and throughput. The proposed location-aware CF approach improves the performance of recommendation significantly by incorporating location information of both users and services in existing similarity measurement approaches of CF. These approaches mainly focus on short-term prediction and do not consider the performance variability over a long-term.

Several studies have focused on time-aware QoS prediction approaches [Qi *et al.*, 2019; Hu *et al.*, 2014]. An approach based on time series forecasting (TSF) is studied to predict the QoS performance of cloud providers [Zadeh and Seyyedi, 2010]. The proposed approach uses Neural Networks (NN) for time series forecasting to conduct experiments. The experiments show promising results for using TFS to predict QoS performance. These approaches are focused mainly on web service selection and do not take the consumer's workload into consideration during the QoS prediction.

A QoS prediction model is proposed using naive Bayesian classifiers in [Al-Faifi *et al.*, 2018]. The proposed model uses historical information perceived by end-users to predict

different performance metrics of cloud based on different configurations of VMs. A new performance prediction method is proposed in [Scheuner and Leitner, 2018b]. The proposed approach builds classifiers based on application and micro-benchmark results to estimate cloud application performance on VMs. These approaches do not consider the long-term performance variability. As a result, these approaches cannot be applied directly for long-term performance discovery.

To the best of our knowledge, most existing QoS prediction approaches do not consider the consumer's workload during the prediction. We propose a cooperative QoS prediction approach that predicts service performance based on consumer workloads.

2.5 Long-term IaaS Cloud Service Selection

The Long-term IaaS selection and the short-term IaaS cloud service selection are fundamentally different in their nature [Ye *et al*., 2011; Zheng *et al*., 2011; Wang *et al*., 2009]. During short-term selection, only current performance of service considered at the time of the selection [Zeng *et al*., 2004; Bouguettaya *et al*., 2010]. For example, which service would give the best performance in the current situation. Unlike short-term selection, long-term selection considers the change of the performance of an IaaS service over a long-term period [Zheng *et al*., 2013]. Choice of an IaaS service at current stage will affect a consumer in the future. Most of the existing research approaches the IaaS cloud service selection problem from a short-term perspective. Few research efforts have been made to study long-term IaaS selection. Moreover, most long-term selection approaches do not consider the incomplete and limited performance information. In the following subsection, we have summarized existing research in long-term IaaS composition.

2.5.1 Long-term Selection from a Consumer's perspective

A consumer request may not be fulfilled by a single cloud service. In this case, a consumer needs to select appropriate cloud services and compose them to fulfill their request. This selection process will be based on the quality of services [Ye *et al*., 2011]. Cloud services

are categorized into two categories: application services and utility services [Ye *et al.*, 2011]. Application services are the services provided by SaaS providers where utility services are offered by IaaS providers. The value of the QoS is determined by the choice of utility services. This paper proposes QoS aware cloud service composition mechanism using a genetic algorithm to achieve large-scale composition. Application service and utility service, both are considered for this composition mechanism from a consumer's perspective. Cloud provider and consumers relationships are usually long-term and economic driven [Ye *et al.*, 2016]. A long-term QoS aware cloud service composition from a consumer's perspective approach is proposed [Ye *et al.*, 2016] that represents the long-term consumer's request in Time Series Group (TSG) for multiple QoS attributes. A prediction model is proposed based on the provider's performance history and short time advertisement to represent the provider's long-term performance in TSG. Correlation between multiple time series is considered to reduce error in the prediction model. The composition problem is then considered as a similarity search problem.

2.5.2 Long-term Selection from a Provider's Perspective

Cloud service selection is considered from the IaaS provider's perspective in several studies [Mistry *et al.*, 2015]. An IaaS provider is usually unable to satisfy all incoming customers due to fast-growing cloud consumers (i.e. SaaS providers). As a result, IaaS providers select a subset of requests of consumers to efficiently utilize a fixed set of resources and maximize profit. A prediction model is proposed to estimate the dynamic behavior of consumer requests [Mistry *et al.*, 2015]. The prediction model is used to select an optimal set of consumer requests to maximize long-term economic benefit. IaaS providers need a long-term economic model to maximize their profit from an optimal selection of requests [Armbrust *et al.*, 2009]. A long-term consumer request can be efficiently captured with the qualitative economic model instead of a quantitative economic model.

2.5.2.1 Long-term Selection using Quantitative Approaches

Several existing studies have performed the long-term IaaS selection in a quantitative approach [Mistry *et al.*, 2016b; Mistry *et al.*, 2015]. In a quantitative approach, an IaaS consumer

requests a fixed set of resources with associated QoS. For example, an IaaS consumer may request 100 CPU units, 99% availability and 8 Mbps network bandwidth. A provider's business plans are also modeled in a quantitative manner such as maximization or minimization strategies. For instance, a provider may target to gain at least 10% profit and 50% resource utilization. The provider applies these strategies during the selection process of the consumer requests. These approaches are not applicable if consumers have a variable range of requirements with conditional preferences. In the real world, a consumer may have a variable range of requirements instead of a fixed set of requests. Amazon EC2 allows the consumer to define a maximum number of VM instances that can be created on demand. For example, consumers may define 10 – 20 VM instances to be created based on their service demand.

A new optimization approach for cloud service composition is introduced [Mistry *et al.*, 2016b] that selects services based on the provider's economic expectations while considering the resource and QoS demands. The paper proposes a Hybrid Adaptive Genetic Algorithm (HAGA) to improve the quality of the composition at run-time. The main aspect of the proposed algorithm is its ability to self-adjustment to improve the prediction accuracy. The experiment shows the feasibility of the proposed approach. This work can be extended for privacy-aware cloud service composition from a provider's perspective.

2.5.2.2 Long-term Selection using Qualitative Approaches

The long-term IaaS selection in qualitative approaches provides an effective and natural way to represent provider's and consumer's qualitative preferences. The qualitative preference representation allows its users to express their preference more intuitively [Wang *et al.*, 2017]. A qualitative IaaS selection approach enables IaaS providers to define their conditional preferences in a qualitative manner. For example, a provider may specify that if a resource utilization is low, the number of customers is more important than profit. A qualitative economic model is presented for long-term IaaS requests composition [Mistry *et al.*, 2016a]. The proposed economic model captures a provider's long-term economic models in a qualitative manner.

A consumer's preferences can be captured in a qualitative manner. A qualitative approach is proposed to capture conditional preferences of IaaS consumers [Fattah *et al.*, 2018]. The proposed approach defines consumer preferences using Conditional Preference Networks (CP-net) [Boutilier *et al.*, 2004]. A CP-Net is a powerful tool to represent and reason with conditional preferences under ceteris paribus ("all else being equal") semantics. The proposed approach assumes that the preferences of consumers arrive in the form of CP-nets. These CP-nets are required to be composed to find the most suitable set of consumers according to the provider's preferences.

Testing a consumer's long-term workloads in the cloud may be cumbersome and *error-prone*. The performance of the same application in two different clouds may vary significantly. Most existing studies conduct experiments to measure *short-term performance*. Existing performance monitoring and testing approaches are not directly applicable to the long-term selection. Availability of historical datasets with detailed information is limited in the public domain due to the privacy issue. *Sharing trial experience is of less concern to cloud consumers* [Swan, 2012]; however, there are forums and websites such as Geekbench where consumers share their trial experiences. We leverage the trial experiences of past consumers to predict future QoS performance of a service or to create a performance model of the service called IaaS performance signature that captures a service's long-term performance variability.

To the best of our knowledge, most existing long-term selection approaches assume that the performance of IaaS services are known to the consumer during the selection. In this work, we assume that IaaS providers reveal limited performance information. In this regard, we develop long-term performance discovery approaches for the selection.

2.6 IaaS Performance Change Detection

The performance variability of IaaS services is addressed in several studies [Iosup *et al.*, 2014; Leitner and Cito, 2016]. The performance of IaaS cloud services is typically estimated for different applications based on short-term trials [Wang *et al.*, 2018; Fattah *et al.*, 2019]. However, most of these approaches do not consider changes in long-term IaaS performance

behavior. Several existing studies suggest that performance of IaaS services often changes periodically [Iosup *et al*., 2011]. In this regard, it is possible to estimate a service's long-term performance based on historical performance data [Fattah *et al*., 2020a]. However, an extensive study on the variability of IaaS performance is carried out which suggests that cloud performance is a "moving target" and requires re-evaluation periodically [Leitner and Cito, 2016]. Therefore, performance change detection is important for the long-term selection.

Change detection is a topical research topic that identifies abrupt changes in a process [Veeravalli and Banerjee, 2014]. It has been applied to many domains including climate change detection, speech recognition, activity recognition, and edge detection in image processing. Existing approaches for the change detection problem are categorized as either "offline" or "online" methods [Aminikhanghahi and Cook, 2017]. Offline methods analyze the entire data set at once and find where the change had occurred. Online methods for change detection monitor and analyze each data point as it becomes available from a stream or source. Online methods typically rely on the statistical properties of the process to determine the change. We identify three criteria to evaluate change point techniques: a) ability to detect changes, (b) accurately identifying the change points, and (c) the number of tests to detect changes. We apply these three criteria to evaluate the proposed change detection approaches in this thesis. A key challenge in detecting changes is to distinguish between noise and true changes in a process. In other domains, noise is typically well-defined. *However, to the best of our knowledge, there is no well-defined concept of noise in the context of IaaS performance.* In this thesis, we propose an IaaS performance noise model that helps us to accurately identify performance noise.

Change detection in a process is often performed based on anomaly detection [Ye, 2017]. Performance anomaly detection is a well-studied topic in many domains including cloud computing, distributed systems, security, and software engineering. Anomaly detection strategies are classified into four major categories in [Ibidunmoye *et al*., 2015], which are a) signature-based detection, b) observational detection, c) knowledge-driven detection, and d) flow and dependency analysis. We decided to choose the signature-based anomaly detection as it is a natural fit for our work. Signature-based detection does not require the retention

of historical information; as a result, we do not need to keep the record of past trial users to detect changes in signatures.

To the best of our knowledge, existing studies do not consider changes in the long-term IaaS performance behavior of a service. We introduce a set of long-term IaaS performance change detection approaches in the context of IaaS signatures where we leveraged existing change detection techniques from different domains.

2.7 Summary

This chapter discussed the fundamental difference between the short-term selection and long-term selection of IaaS services and existing short-term selection approaches. Existing short-term selection approaches typically involve the following steps: a) Representative workload generation, b) Performance benchmarking, c) QoS prediction and ranking, and d) Selection. We have discussed existing approaches related to each of the steps related to the selection. We have also highlighted key long-term IaaS cloud selection approaches and their limitations. We have leveraged the existing short-term performance discovery, and long-term selection approaches in this thesis to propose a set of long-term selection approaches where providers reveal limited performance information. We have also discussed existing studies about IaaS performance changes. Existing change detection approaches primarily focus on short-term changes. We have leveraged existing change detection techniques and proposed new change detection techniques in the context of IaaS performance.

CHAPTER 3

Performance-based Long-term IaaS Selection

3.1 Introduction

The long-term IaaS cloud service selection is a topical research challenge in cloud computing [Mistry *et al*., 2016b]. Selecting the right IaaS cloud service is an important *business decision* for cloud consumers [Fattah *et al*., 2020b]. IaaS cloud services are selected based on a consumer's long-term *functional* and *non-functional* requirements [Zheng *et al*., 2009]. Functional requirements are set based on the purpose of the service such as computing, data storing, and networking. Non-functional requirements are often expressed in terms of Quality of Service (QoS) attributes such as availability, response time, and throughput. QoS attributes help a consumer to select the *best-performing* services from a large number of functionally similar services. The QoS aware service selection is therefore defined as the similarity matching between a consumer's long-term QoS requirements and the expected long-term performance of IaaS services [Mistry *et al*., 2016b].

The knowledge of the IaaS services' performance is essential for long-term selection [Iosup *et al*., 2011]. Selecting a service that may perform poorly in the future, may lead to an inevitable *loss of productivity* for an organization. However, most IaaS providers are typically reluctant to divulge the *detailed* and *complete* information about their long-term QoS management policies in the dynamic multi-tenant environment [Dou *et al*., 2013]. The key reasons for such behavior are market competition, business secrecy, and conflicts of interest [Fattah and Bouguettaya, 2020a]. For example, Amazon does not disclose the actual throughput information of its vCPUs in its advertisements[1]. A consumer who has CPU-intensive workloads may

[1]https://aws.amazon.com/ec2/instance-types/

find it challenging to select Amazon for a long-term period with such limited performance information [Scheuner and Leitner, 2018b]. The performance of a VM may change over time given the dynamic nature of the cloud environment [Iosup *et al*., 2011]. As a result, advertised performance information may not reflect the actual service performance for a particular provisioning time. For example, a consumer may want to utilize some VMs in December where the advertised performance is measured in June. In such a case, the advertised information is not useful to perform the selection in December. Therefore, selecting an appropriate IaaS service is challenging without the detailed information of a service's long-term QoS performance.

To the best of our knowledge, existing approaches focus mainly on two aspects for the IaaS cloud service selection with the *incomplete* information: a) relying on IaaS advertisements, and b) gathering information from trial experiences [Wang *et al*., 2018]. IaaS advertisements typically contain limited QoS information. For instance, each vCPU may be a thread of an Intel Xeon core, an AMD EPYC core, or AWS Graviton processor according to EC2 advertisements[1]. Estimating the performance of the vCPU is difficult from such limited information [Scheuner and Leitner, 2018b]. Most providers do not differentiate between long-term and short-term services in terms of service performance. IaaS providers often advertise an average or maximum performance of their services [Persico *et al*., 2015]. For instance, the network performance of some Amazon EC2 instances has a data transfer rate of up to 10-gigabits. Existing study shows that IaaS providers often fail to offer the *promised* QoS performance [Dou *et al*., 2013]. Therefore, relying only on IaaS advertisements may not be sufficient for long-term selection.

Most IaaS providers offer *free trials*, encouraging potential consumers to test their application workloads in the cloud. For instance, Microsoft Azure offers a one month trial period for a limited number of services to its potential consumers. Application benchmarks and micro-benchmarks are usually utilized in the trial periods to discover a provider's performance in terms of various QoS attributes such as CPU speed, disk read/write latency, and network bandwidth [Scheuner and Leitner, 2018b]. Synthetic workloads are generated for the representative applications to perform *stress testing* on the providers. The results of the tests are used to compare different providers. These approaches focus mainly on the *short-term selection*

and do not consider the long-term QoS performance variability of the providers. In reality, the QoS performance of a provider varies over the long-term period due to the *multi-tenant* nature of the cloud environment [Iosup *et al*., 2014]. For example, a provider may show very good performance in the Christmas period, when the number of active consumers drops considerably. *We aim at leveraging the free short-term trial experience of consumers for the long-term IaaS cloud service selection in this work. We assume that a consumer considers only one service from a provider for the long-term selection. Therefore, we use the word "provider" and "service" interchangeably.*

We identify three key challenges to select IaaS providers for a long-term period based on short-term trial experience. First, a large number of IaaS providers may satisfy a consumer's long-term requirements. The performance of these providers may vary considerably depending on their business strategies and infrastructures. Performing a trial with every eligible IaaS is *practically* infeasible. Second, free trials are generally offered for a *short-term period*. A consumer cannot test their entire long-term workloads in the short-term trial period. Some IaaS providers (e.g., Amazon) offer long-term trials for several services. A consumer may not be able to *wait such a long time* to make the selection. Third, it is challenging to predict the long-term performance from a short-term trial without the performance variability information [Wang *et al*., 2018]. Most IaaS providers usually operate in multi-tenant environments. IaaS performance is highly time-dependent [Iosup *et al*., 2014]. The performance discovered in a one month trial period may not *reflect* the actual performance of a provider for the rest of the year. The trial experience of a consumer may depend on several factors such as trial workloads, the time of the trial, and the provider's QoS management policy [Scheuner and Leitner, 2018b]. The experience from an *unplanned* short trial may not provide the complete information required for the long-term selection.

We propose a novel framework that utilizes a consumer's short-term trial experience to select the closest-matched IaaS provider according to the consumer's long-term QoS requirements. We incorporate the experience of past trial users to predict providers' long-term performance. Experiences of past trial users may not be applicable directly to predict IaaS performance for a new consumer. The reason is that the workloads of the past trial users are most likely to

be *different* from the new consumer's workloads. We propose a cooperative long-term QoS prediction approach based on the *workload similarity* of the past trial users. The performance of the provider may change over the long-term period. The proposed framework measures the confidence of the prediction based on the trial experience of the new consumer. Our contributions in this work are summarized as follows:

- A skyline-based filtering method to select candidate IaaS providers to perform free trials.
- A cooperative long-term QoS performance prediction approach using the experience of past trial users. The confidence of the trial is measured based on the trial experience of the new consumer.
- A long-term QoS performance prediction approach using a trial workload generation model when no similar trial users are available.
- A QoS-aware selection method to choose the "closest match" IaaS provider using multidimensional time series similarity measure between the consumer performance requirements and the predicted performance of providers.

3.2 Motivation Scenario

Let us assume that a university wants to lease some general-purpose VMs for one year. Each VM requires at least 2 vCPU and 6 GB memory. The university has a minimum expected QoS performance on availability, response time, and throughput. The QoS requirements vary based on the university's seasonal demands. For example, the university may need high throughput during the examination period, and low throughput in the Christmas period. Figure 3.1(a) depicts the consumer's expected availability on three different periods (T1, T2, and T3) in a year. We assume that the university's workloads are *deterministic*, i.e., the university has estimated its future workloads for one year based on history. Figure 3.1(b) shows the consumer's CPU workloads for the $T1$, $T2$, and $T3$ periods.

Several providers in the cloud market may advertise the required type of VM to the university. Let us assume that $P1$ and $P2$ are two such providers. Both providers advertise availability information for the VMs in their IaaS advertisements. The response time and throughput

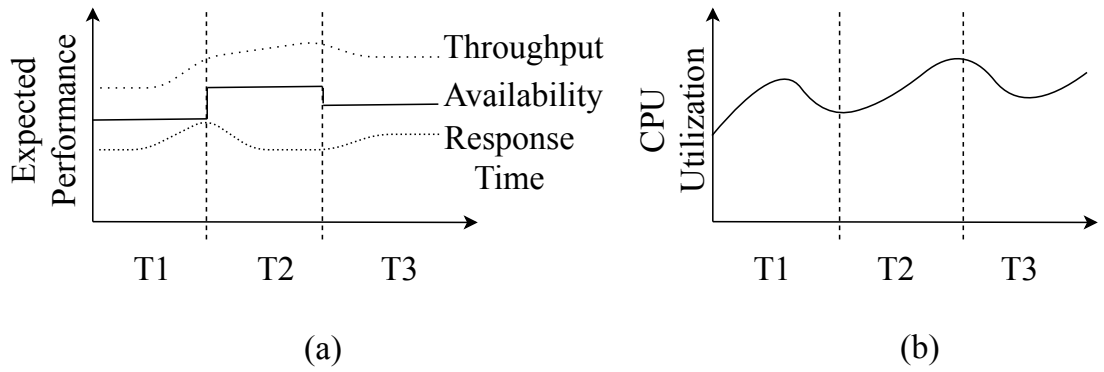


FIGURE 3.1: University's long-term requirements (a) expected QoS performance (b) long-term CPU workloads

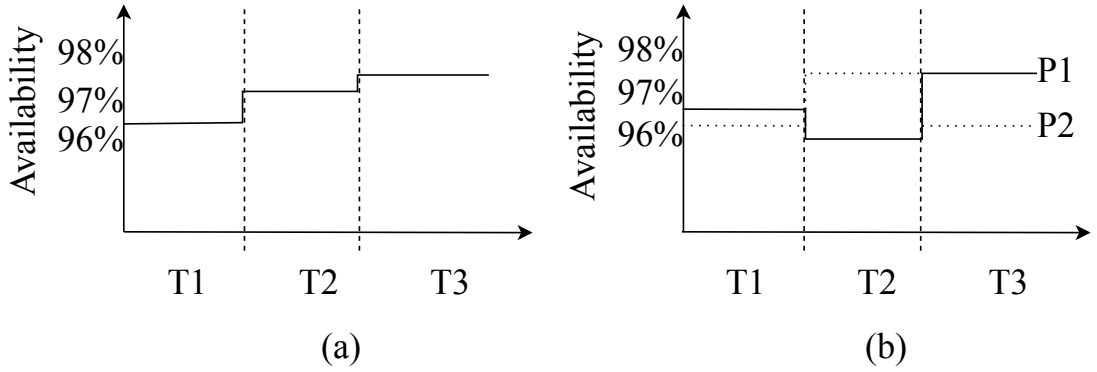


FIGURE 3.2: University's long-term requirements (a) expected availability (b) advertised availability

information are not available in the advertisements. Figure 3.2(a) shows the consumer's expected availability. Figure 3.2(b) shows the advertised availability of two providers. The consumer will select $P2$ if they select based on availability. This selection may not be a good decision as it does not consider the consumer's expected throughput and response time.

Let us assume that each provider offers a trial to the university. The university performs trials on the $T2$ period and observes the throughput and the response time of the IaaS service. Note that, the university cannot test its entire workloads in the $T2$ period. Let us assume that the university performs the free trial using only the workload of the $T2$ period (Figure 3.1(a)). If the university makes the selection considering only the observed performance in the trial period, it may not be the right decision. For example, the university may observe that $P1$ offers better throughput and response time. Selecting $P1$ may be a wrong decision as $P2$

performs better in the $T1$ and $T2$ periods. Let us assume that *the consumer has access to the experiences and workloads of past trial users*. Workloads of trial users may be different from the university's workloads. We identify the following cases:

- *Similar Trial Users*: The experience of past trial users is available where the users have similar workloads to the university. These users could be universities, colleges, banks, or other large organizations. The university performs a trial using its workloads of the $T2$ period and observes the throughput and response time of the providers. The university utilizes the experience of the trial users to predict the performance of each provider for its workloads in the $T1$ and $T3$ periods. The confidence of the prediction needs to be measured as the prediction is made based on historical information. Each provider's current performance may be improved or degraded than the experiences in the past.
- *Dissimilar Trial Users*: In this scenario, we assume that there are no trial users available who have similar workloads to the university. If the existing trial users include SaaS providers and video content providers, they are most likely to have different workloads than the university. In such a case, the university cannot completely rely on those past experiences. The university needs to perform the trial efficiently to understand the providers' performance behavior for its long-term workloads.

The proposed selection framework considers both cases for long-term selection. First, a filtering method is applied that selects candidate IaaS providers for the trial periods. Next, the proposed framework predicts the long-term QoS performance of each provider by leveraging the trial periods for similar trial users and dissimilar trial users. An IaaS provider is then selected based on the predicted QoS performance and the consumer's expected QoS performance from the candidate providers.

3.3 IaaS Selection Framework

We formulate the long-term IaaS provider selection using the following definitions and notations in Table 3.1.

TABLE 3.1: Notations and descriptions

Notation	Description
T	Required provisioning time
W	Workload requirements in time series
Q_C	Set of QoS Requirements of Consumers
l	Number of QoS parameters in Q_C
q_{ci}	The time series of the QoS parameter q_{ci}
t_n	A timestamp in T where $n = 1, 2, 3, ...T$
x_n	The value of $q_(ci)$ at the time period t_n
N	The number of IaaS providers who can satisfy the functional requirements of the consumer
P	A set of IaaS providers
A_i	The QoS advertisement of the provider P_i

- *Consumer*: A consumer is a new IaaS Consumer who requires Virtual Machines (VMs) for a long-term period.
- *Functional Requirements*: Functional requirements are defined in terms of different types of VM configurations (i.e., the number of vCPU and memory units) and the number of VMs over the long-term period.
- *Long-term Workloads*: The workload of a consumer is represented as the requested number of resource units such as CPU and memory over the long-term period.
- *QoS Requirements*: QoS requirements of a consumer is a set of QoS parameters and their minimum or average expected values for a long-term period.
- *Provider*: A provider is an IaaS provider who provisions VMs for a long-term period.
- *QoS Advertisement*: A QoS advertisement is a set of QoS parameters for a VM and the values of the QoS parameters over the long-term period.
- *Trial Periods*: A consumer can use some services with restricted conditions for free.

Let us assume that a consumer *requires* a set of general-purpose VMs of a particular configuration over a long-term period. The consumer has *variable workloads* over the long-term period. We assume that the workloads are *deterministic*, i.e., the consumer has full knowledge of its workload distribution per VM over the long-term period. The workload of the consumer can be defined as the required amount of resources (e.g., CPU time, and Memory size) at

a particular period. *We consider the workload as a combined resource requirement at a particular time and denote it as* W.

We represent the consumer's workloads and QoS requirements for the long-term period using *time series groups* (TSGs). We denote the total service usage time as T. The TSG of QoS requirements is defined as $Q_C = \{q_{c1}, q_{c2}, ..., q_{cl}\}$, where cl is the number of QoS parameters in Q_C and $q_{cl} = \{(x_n, t_n)|n = 1, 2, 3, ..., T\}$, where x_n is the value of q_{cl} at the time of t_n. We define the time series of workloads per VM as $W = \{(w_n, t_n)|n = 1, 2, .., T\}$, where w_n is the workload per VM at time t_n. Here, n denotes a timestamp.

Let us assume that there are N number of IaaS providers who can fulfill the functional requirements of the consumer. The set of the providers is denoted as $P = \{P_1, P_2, ...P_N\}$. The QoS performance of VMs varies from one provider to another provider over the long-term period. Each provider advertises long-term QoS properties in TSG for its VMs. The QoS advertisement of the provider P_i is denoted as $A_i = \{a_{i1}, a_{i2}, ...a_{ik}\}$, where ik is the number of QoS parameters in A_i. We assume $ik < l$, i.e., the number of QoS parameters in the advertisements is always less than the number of the QoS parameters in the consumer requirements. The time series of each QoS parameter of provider P_i in the advertisement is denoted as $a_{ik} = \{(y_{im}, t_{im})|im = 1, 2, 3, ..., T\}$ where y_{im} is the value of a_{ik} at the time of t_{im}. Here, im is the timestamp of the advertisement of the provider P_i. We assume that each provider advertises with the same interval, i.e., im is the *same* for all providers.

The consumer requires to predict long-term QoS performance of the providers to make an informed selection. The advertisements do not provide enough information (i.e., $ik < l$). We denote the predicted QoS performance of the provider P_i as $Q_i = \{q_{i1}, q_{i2}, ..., q_l\}$ where l is the number of QoS parameters in Q_i. The consumer requires to select a provider based on the predicted QoS performance that closely matches its expected QoS performance. Given the consumer's QoS expectations Q_C and a provider's predicted QoS performance Q_i, we use a predefined TSG distance measuring function $distance(Q_C, Q_i)$ to find the most similar provider P_s using the following equation:

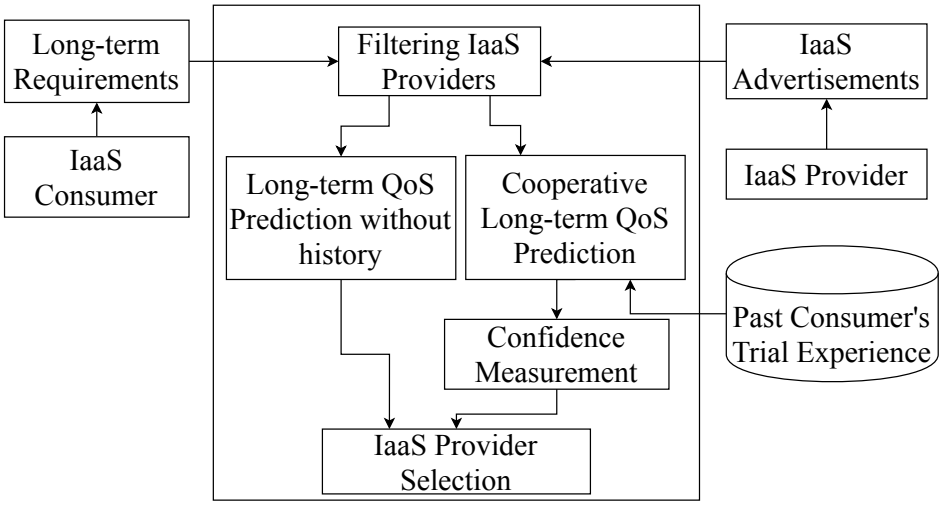


FIGURE 3.3: Long-term IaaS provider selection framework

$$P_s = \text{argmin}_{i=1}^{l}(distance(Q_C, Q_i)) \tag{3.1}$$

Figure 3.3 shows the proposed long-term IaaS provider selection framework. The framework requires a consumer's long-term requirements, i.e., expected QoS performance and workloads as the *input*. The other inputs to the framework are the long-term advertisements of a set of providers who can fulfill the functional requirements of the consumer. The framework has the following four modules:

- *Filtering IaaS Providers*: A filtering method is proposed to reduce the search space based on the similarity between the consumer requirements and the QoS advertisements of the providers. Finding such providers is a *multi-criteria decision-making problem*. Each provider may have advertisements that are the best for a particular QoS parameter at a particular time period. There may be *no clear winner* as to who provides the best advertisements for all QoS parameters over the entire period. We incorporate a *temporal skyline* [Wang *et al*., 2013] to solve the multi-criteria decision-making problem in the filtering method.
- *Cooperative Long-term QoS Prediction (CLQP)*: A consumer's long-term workloads may not be tested in a short trial. Some parts of the workloads may be tested directly in the trial period while the performance of the other parts can be inferred from past trial users.

A cooperative QoS prediction approach is applied based on the *experience of trial users* using a "workload similarity" measure technique. A confidence measurement technique is devised for the predicted performance using the trial experience.

- *Long-term QoS Prediction without History (LQP-short)*: When there is no similar past trial user, the proposed framework generates trial workloads by mapping the consumer's long-term workloads into the short-term trial periods to discover the performances. The trial workloads are generated using *time series compression techniques*.
- *QoS-aware Long-term IaaS Provider Selection (QLIS)*: The "closest match" provider is selected, where the predicted performances closely match with the consumer's requirements with the *highest trial confidence* when similar past trial users are available (CLQP). Therefore, the proposed framework does not require an *exact similarity measure* between the consumer requirements and a provider's performance. In the absence of similar trial users, the provider is selected based on the LQP-short.

3.4 Filtering IaaS Providers

We develop a filtering method to reduce the number of candidate IaaS providers for the trial. The filtering process can be modeled as single-criterion or multiple-criteria decision-making, depending on the consumer's requirements.

3.4.1 Filtering IaaS Providers based on Single Criterion

If a consumer's QoS requirement (Q_c) contains only one QoS parameter, i.e., $|Q_c| = 1$, the filtering method can be modeled based on single-criterion decision-making. For example, when a consumer only cares about the response time of the VMs, the candidate providers can be selected based on the response time without considering other QoS attributes. In such a case, we compare the consumer's long-term QoS requirement with each provider using time series similarity matching techniques. A well-known time series similarity matching technique is the Mean Absolute Error (MAE) distance [Tang *et al.*, 2016]. MAE is a fast, effective, and easy-to-implement technique where similarity is measured between each corresponding

timestamp of two time series. Equation 3.2 computes the similarity between the consumer's long-term QoS requirement q_c and a provider's advertisement (a_p) for a single QoS parameter:

$$\text{MAE}\ (q_c, a_p) = \frac{1}{n} \sum_{t=1..n} \mid q_c^t - a_p^t \mid \tag{3.2}$$

In Equation 3.2, n is the number of timestamps, q_c^t is the value of QoS parameter q_c at time t. A number of candidate providers should be selected based on the measured distance. The candidate providers can be selected using the top-K approach [Zheng *et al.*, 2012]. The top-K approach selects the best K candidates who have the minimum MEA distance from the consumer's QoS expectation. If the number of selected candidate providers is too small or too large, K can be adjusted to reduce or to increase the number of candidate providers for the trial period.

3.4.2 Filtering IaaS Providers based on Multiple Criteria

When the consumer's QoS requirements (Q_c) contain more than one QoS parameter, i.e., $|Q_c| > 1$, we can model the filtering method as a multi-criteria decision-making problem. Multi-criteria decision-making approaches are applicable when the consumer's selection depends on multiple criteria such as price, throughput, and response time. In such a case, the consumer's QoS requirements and a provider's QoS advertisements should be compared for each QoS parameter. The main issue of filtering based on multi-criteria is having *pareto-optimality* or incomparable providers. For instance, a provider may advertise VMs at the lowest price while another provider may advertise VMs with the highest performance. In such a case, there is no *clear winner* as both of them are best on different criteria.

3.4.2.1 Filtering based on Utility Function

There are several approaches to filter candidate providers based on multiple criteria. Some general approaches are utility function, conditional preference, and skyline [Jiang and Pei, 2009]. The *utility function* computes a score for each provider based on the consumer's

preference for each attribute. A consumer needs to assign weights on each QoS attributes based on their preference. For example, a consumer may put the highest weight on the price attribute as the price may be the most important attribute. The consumer may set lower weights to the less important attributes. The utility function calculates a score for each provider based on the weights of the attributes and the value of the attributes. The utility function computes the score using the following equation:

$$\text{Score}(P) = \sum_{q_c \in Q_C, a_p \in A_p} W_q \times \text{MAE}(q_c, a_p) \tag{3.3}$$

where W_q is the weight of the QoS attribute q assigned by the consumer, Q_c is the TSG of the consumer's QoS requirements, and A_p is a provider's QoS advertisements. A multi-criteria decision-making problem is then transformed into a single-criterion decision-making problem using the utility function. We can apply the top-K method using the utility scores to filter the providers for free trial.

3.4.2.2 Filtering IaaS Providers based on IaaS Skyline

The utility function based filtering method has two major *drawbacks*. First, it requires a consumer to assign weights on the QoS attributes. Second, if a consumer's preferences change, the utility function should be updated. The skyline algorithm is a well-known alternative to the utility function [Wang *et al*., 2013]. A skyline does not require a consumer to assign weights to the QoS attributes. It includes all of the outputs that can be generated by a utility function [Jiang and Pei, 2009]. The skyline concept is adopted from the real-world skyline where the most dominating items are displayed in a skyline. For example, a city skyline consists of the tallest, widest, or closest building to the viewer. Similarly, a skyline filtering method consists of the provider who advertises the best value for at least one QoS attribute.

Let us assume that there are N IaaS providers who can fulfill the consumer's functional requirements. The number of such IaaS providers is not expected to be very large given that the number of top IaaS providers is no more than 100 [2]. Therefore, we consider the value

[2] https://stackify.com/top-iaas-providers/

of N here is arbitrary. We need to find a set of candidate providers P' where $P' \subset P$ and $|P'| < |P|$, i.e., the number of selected providers is less than the number of existing providers. Let us assume that $|P'| = M$ i.e., the number of selected IaaS providers is M. The set of QoS advertisements for all IaaS providers is denoted as $A = \{A_1, A_2, A_3, ..., A_N\}$ where A_i is the advertisement of provider P_i. We need to select M number of IaaS providers from the existing N IaaS providers based on the advertisements A and the consumer requirements Q_C. We need to find a set A' where $A' \subset A$ and A' contains the closest match information according to Q_C. The advertisement contains more than one QoS parameter to compare for ranking the providers. Therefore, selecting the candidates is a multi-criteria selection problem.

We utilize a *temporal skyline* to solve the selection of candidate providers over multiple QoS criteria. The *skyline query* is an effective approach to select a set of data points that is better than any other data points in a large dataset [Jiang and Pei, 2009]. The temporal skyline is a time-series version of the skyline that selects a set of time series that are better than any other time series. We propose a temporal skyline approach to select a set of candidate providers which is better than the other sets of providers over time T.

3.4.2.3 IaaS Skyline

A provider P_i dominates another provider P_j if P_i offers equal or better QoS advertisements than P_i for all QoS parameters and P_i offers better advertisement than P_j for at least one QoS parameter. A skyline of providers contains the providers that are *pareto-optimal* and not dominated by any other providers for every QoS parameters. Figure 3.4(a) shows an example of the skyline of providers. Here, each provider offers two QoS parameters, i.e., throughput and availability. P_3 offers higher throughput and availability time than P_4 and P_6 thus P_3 dominates P_4 and P_6. Similarly, P_2 dominates P_5, P_6, and P_4. The skyline of the providers contains P_1, P_2, and P_3.

We assume that the consumer has a set of QoS parameters that are more important or *dominant* than other QoS parameters denoted as Q_D where $Q_D \subset Q_C$. For example, a consumer may consider that the throughput and availability of a VM are more important than any other QoS parameters. The weight of each dominant QoS parameter is considered equal. A provider

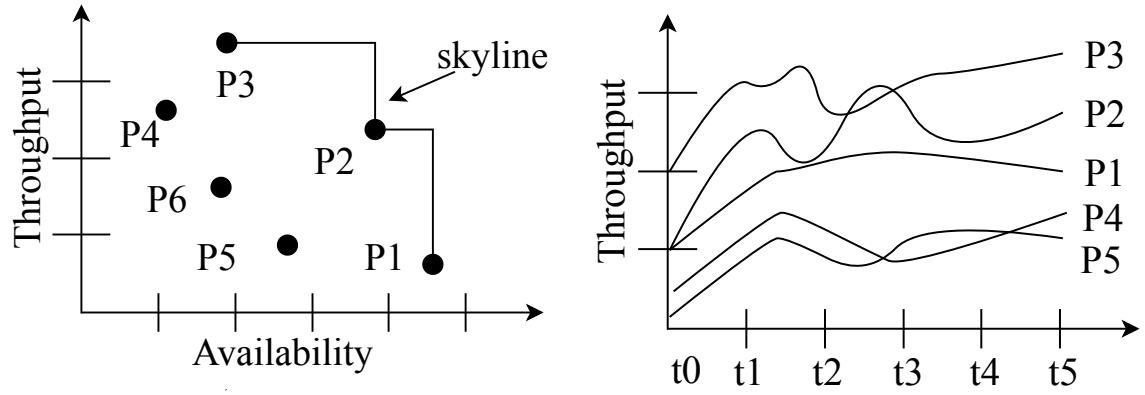


FIGURE 3.4: Filtering with skyline (a) IaaS skyline (b) temporal IaaS skyline

who does not provide throughput and availability information will not be considered for the trial. The size of Q_D can vary according to consumer preferences.

DEFINITION 1. *Dominant Provider*. A provider P_i dominates another provider P_j, denoted as $P_i \succ P_j$, if P_i provides as good as or better advertisements for all dominant QoS parameters in Q_D i.e., $\forall q \in Q_D : P_i \succeq P_j$ and $\exists q' \in Q_D : P_i \succ P_j$

DEFINITION 2. *IaaS Skyline*. The IaaS skyline of a set of IaaS providers P, denoted as SK_P, is a subset of providers that is not dominated by any other providers, i.e., $SK_P = \{p \in P | \neg \exists p' \in P : p' \succ p\}$

3.4.2.4 Temporal IaaS Skyline

We represent the long-term QoS advertisements using time series where the value of each QoS parameter may change over time. We utilize the temporal skyline introduced in [Jiang and Pei, 2009], where QoS parameters can be represented as time series .

DEFINITION 3. *Dominant QoS Time Series*. A QoS time series Q_i dominates another QoS time series Q_j in time T, denoted as $Q_i \succ Q_j$, if $\forall t \in T, Q_i \succeq Q_j$ and $\exists t' \in T, Q_i \succ Q_j$.

DEFINITION 4. *Temporal QoS Skyline*. The temporal QoS skyline of a set of QoS time series Q, denoted as ST_Q, is a subset of QoS time series that is not dominated by any other QoS time series for all timestamp t, i.e., $ST_Q = \{q \in Q | \neg \exists q' \in Q : q' \succ q\}$

Figure 3.4(b) shows a set of “throughput" time series of providers $P1$, $P2$, $P3$, $P4$, and P_5. $P3$ and $P2$ dominates all other providers for each timestamps, i.e., $t0$ to $t5$. $P3$ dominates $P2$ at $t2$ and $P2$ dominates $P3$ at $t3$. Therefore, $P3$ and $P2$ do not dominate each other and are included in the temporal QoS skyline for the interval from $t2$ and $t3$. The temporal skyline shown in Figure 3.4 considers only a single QoS parameter. In reality, the number of dominant QoS parameters Q_D in the advertisements may be more than one. We need a multiple time series approach for the skyline. We define a multiple time series (MTS)-based *IaaS skyline* and *Temporal QoS Skyline* as follows:

DEFINITION 5. *MTS-based Dominant IaaS Provider*. An IaaS provider P that has Q_D dominant QoS time series, dominates another IaaS provider P', denoted as $P \succ_{mts} P'$ if (1) $\forall t \in T, P \succeq P'$ and $\exists t' \in T, P \succ P'$ and (2) $\forall q \in Q_D, P \succeq P'$ and $\exists q' \in Q_D, P \succ P'$

DEFINITION 6. *MTS-based IaaS Skyline*. The IaaS skyline of a set of IaaS Providers P, denoted as MTS_P, is a subset of providers that is not dominated by any other providers, i.e., $MTS_P = \{p \in P | \neg \exists p' \in P : p' \succ_{mts} p\}$

We assume that each provider’s advertisement contains at least one dominant QoS parameter. Hence, the MTS-based IaaS skyline includes the providers that are not dominated by any other providers for all timestamp in T and all QoS parameters in Q_D. The existing literature show several methods of computing skylines [Wang *et al*., 2013]. Our focus is on how to apply skyline to select candidate IaaS providers rather than computing skyline efficiently. We use a nested loop (NL) algorithm [Jiang and Pei, 2009]. Algorithm 1 illustrates the NL algorithm to compute IaaS skyline.

Algorithm 1 takes a list of providers and their corresponding QoS advertisements as input in line 1. The output of the algorithm generates the IaaS skyline of the provider (line 2). Initially, we consider that each provider is a part of the skyline. Therefore, we create a list S where we assign each provider (line 3). Next, for each provider (P) in the list S (line 4-5), we compare the provider with all other providers (P_i) in S to test whether a provider is dominated by any other provider using the MTS-based dominance relation. If a provider (P) is found to be dominated by any other provider (P_i) in the list (S) (line 6), we remove P from S. Once the

algorithm finishes comparing each provider in the list, the remaining providers in S are part of the skyline (line 7). S is assigned into MTS_P which represents the final IaaS skyline and returned (line 8).

Algorithm 1 Computing Skyline using NL algorithm

```
1: Input: L, A
2: Output: MTS_P
3: S ← L;
4: for P ∈ S do
5:     for P_i ∈ S do
6:         if P_i ≻_mts P then discard P from S;
7: MTS_P ← S;
8: return MTS_P;
```

3.5 Cooperative Long-term QoS Prediction

The experiences of trial users may not be directly applicable to predict a provider's long-term performance for a new consumer's workloads. The main reason is that each user may perform a trial with different types of workloads according to their QoS requirements and have different experiences. It is highly unlikely that the new consumer's long-term workloads can be matched exactly with the trial workloads of past trial users at each period. However, the new consumer's workloads may have similarities with the trial user workloads. We utilize the experience of the users having similar trial workloads.

Collaborative filtering methods are well-known for QoS prediction from similar user experiences [Zheng *et al.*, 2012]. Traditional collaborative filtering methods for the cloud QoS prediction measure similarities between the users based on their QoS experience. For instance, a new consumer's expected QoS performance will be based on the experience of other existing consumers. In collaborative filtering, QoS values are predicted based on the similarity between the existing consumers and the new consumer. The similarity is typically measured using various attributes such as location, time, and service type. However, they do not consider user workloads during the service invocation. We leverage a traditional collaborative filtering method for the long-term QoS prediction where similarities between a consumer and past trial users are defined based on their workloads.

Let us assume that the total provisioning period has S seasonal periods. The performance of a provider does not vary significantly within a period, but it may vary considerably between the seasonal periods. The total service provisioning time $T = S \times M \times Y$ where S is the number of seasons, M is the size of each season, and Y is the number of seasonal years. For example, a consumer may need service for one year. The one year can be divided into three seasons where the size of each season is four months. Our target is to predict the QoS performance of a provider for each season for the consumer's workloads at that period. We assume that we have a history of past trial users at each season. The effects of day, night, and week are included within each season.

We develop a cooperative long-term QoS prediction **(CLQP)** approach to predict the QoS performance for each season. The proposed approach has two steps: a) finding similar users, and b) measuring the QoS performance based on similar users' experiences. We discuss these two steps in the following subsections.

3.5.1 Finding Similar Trial Users

A similarity measure technique is required to find similar users based on the consumer's workloads. Pearson Correlation Coefficient (PCC) is an effective similarity measurement technique that has been widely used in collaborative filtering for finding *similar users* [Zheng *et al.*, 2012]. It can achieve high accuracy and can be implemented easily. In a recommender system, similarity between two users is typically computed based on users' preferences, observed QoS performance, and location. In our case, we need to find similar users based on the similarity between the workload of two users. For a given interval Δt, the PCC similarity is between two sets of workloads w and w' is defined in Equation 3.4:

$$Sim(w, w')^{\Delta t} = \frac{\sum_{t=j}^{k}(w'_t - \bar{w'})(w_t - \bar{w})}{\sqrt{\sum_{t=j}^{k}(w'_t - \bar{w'})^2}\sqrt{\sum_{t=j}^{k}(w_t - \bar{w})^2}} \tag{3.4}$$

In Equation 3.4, $\bar{w}$ and $\bar{w'}$ denote the average value of w and w' in Δt. The values of workload w and w' at the timestamp t are denoted by w_t and w'_t respectively.

The PCC may not be always applicable for time series data. Whenever a workload time series segment has a steady workload (i.e., variance = 0), the PCC fails to compute the similarity. We utilize the root mean square error distance to measure the similarity of the consumer workloads using Equation 3.5.

$$\text{Sim'}\ (w, w') = \sqrt{\frac{1}{n}\sum_{t=1..n}(w_t - w'_t)^2} \tag{3.5}$$

We incorporate an extended top-K neighbor selection technique to select similar users [Tang *et al.*, 2016]. The extended top-K considers that some users may have a limited number of similar consumers. Traditional top-K algorithms may select consumers that are not similar to the new consumer. We use Equation 3.6 to select top-K similar consumers:

$$S(c) = \{c_k | c_k \in Top(c); Sim(c_k, c) > 0\} \tag{3.6}$$

$Top(c)$ finds the set of similar users c_k to the consumer c and $S(c)$ is the set of similar users who has similarity with the new consumers more than zero.

3.5.2 Measuring QoS Performance

Let us assume that the proposed framework finds k number of users who have trial workloads similar to the new consumer at a given seasonal period. We denote the number of timestamps at a particular season as n. For each timestamp t, we use the following equation to predict the QoS values of an IaaS service:

$$E_{t=1..n}(Q_c) = \frac{\sum_{i=1}^{k} Q^t_{c_k}}{k}; t \in \Delta t \tag{3.7}$$

where k is the number of the similar users, $Q^t_{c_k}$ is the observed value of a QoS parameter by a user c_k at time t. $E_{t=1..n}(Q_c)$ measures the expected QoS in the trial period Δt based on average observed performance by similar users. Equation 3.7 considers all users equally

without considering the degree of similarity of each user. For instance, if we select five users, a particular user may have the highest similarity. Considering the most similar users will not provide good prediction accuracy. Hence, we consider the *degree of similarity* during the computation of QoS performance for a consumer. The following equation is used to measure the QoS performance:

$$E'_{t=1..n}(Q_c) = \frac{\sum_{i=1}^{k}(Sim'(c,c_k)^{\Delta t} \times Q_{c_k}^{t})}{\sum_{i=1}^{k} Sim'(c,c_k)}; t \in \Delta t \tag{3.8}$$

where $Sim'(c,c_k)^{\Delta t}$ is the similarity between the consumer c and the trial user c_k and $Q_{c_k}^{t}$ is the observed QoS performance by c_k. Equation 3.8 measures the QoS performance using the weighted average of past trial experiences.

3.5.3 Performing Trial using Workload Replay

The most straight forward approach to testing a provider's performance is to use the consumer's long-term workloads directly to perform the trial. This approach is called *workload replay*. The advantage of this approach is that it performs the most "real" test of the providers as the workload contains all the real-world complexities.

A key challenge of testing is the short length of free trials. A consumer typically requires to test long-term workloads where the trial periods are short. For instance, a consumer may need a VM for one year and the length of the trial period is only one month. The consumer is unable to test its one-year workloads in one month. Let us assume that the trial period is offered in the month of "June". The result of performing the trial using the workload of "January" may be different from performing the trial using the workload of "June". If the consumer performs the trial with the workload of "June", then the performance of the provider may remain unknown for the workload of "January" unless the workloads of both months are the same. A major concern while performing the trial is the effect of the *temporal effect* on the provider's performance. A provider's performance may vary when a consumer performs the trial with the same workload in the "January" and "June".

We utilize the consumer's workload *directly* to perform the trial. Trial workloads are selected based on the month of the trial. For example, if the trial is performed in the month of "June', we apply the consumer's workload of "June". When a consumer requires service for multiple years, we take the average workload of multiple years for the trial period. The performance of the workloads for other months is inferred from the experience of past trial users as described in the previous subsections (3.5.1 and 3.5.2).

3.5.4 Measuring Confidence

The predicted QoS performance for the consumer's long-term workloads may not provide an accurate result as it is based on historical information. The current performance of the providers may change. Providers may upgrade their infrastructure or change their QoS management policy. Therefore, the consumer needs to measure the confidence of the trial for the long-term selection.

The intuition behind confidence measuring is that a consumer may have more confidence for the selection if its trial experience matches the experience of other similar consumers. However, each consumer performs trials with different workloads. Therefore, when measuring the confidence, a consumer's experience needs to be compared with other consumers who have similar trial workloads. The confidence of the prediction is measured between a consumer's observed performance in the trial and the predicted performance for the trial period. The predicted performance is computed using the experience of similar consumers. The observed and the predicted performance are both represented as time series. Therefore, we measure the similarity between observed and predicted performance to compute the confidence of the trial experience. Here, we measure the similarity based on *Euclidean distance* between the two time series. Euclidean distance is more useful when we want to scale the absolute distance. The confidence of the QoS prediction ($C^{\Delta t}$) is computed by measuring the similarity between the trial experience (E) and the predicted QoS performance (Q) using the following equation:

$$C^{\Delta t} = \sqrt{\frac{\sum_{t=j}^{k}(e_t - q_t)^2}{\Delta t}}; e_t \in E, q_t \in Q \tag{3.9}$$

where Δt is the length of the trial period, E is the observed QoS performance, Q is the predicted QoS performance in the trial period. e_t and q_t are the observed and predicted performance at timestamp t. Equation 3.9 measures similarity in terms of the Euclidean distance of the predicted and observed QoS performance. The proposed long-term selection framework selects a threshold C^{thres} for the confidence variable to filter providers based on the confidence. If the prediction of a provider has lower confidence than the threshold, the provider is discarded.

3.6 Long-term QoS Prediction without Historical Information (LQP-short)

In this section, we introduce a QoS prediction approach for the long-term selection, where we assume that there is no similar trial users. The workload replay-based trial is not suitable for performing a trial in the absence of similar trial users. The consumer needs to measure the IaaS performance for their entire workloads. We propose a trial workload generation model by leveraging existing time series compression techniques.

3.6.1 Trial Workload Generation

The performance of a cloud provider often depends on the characteristics of the workloads it serves [Feitelson, 2002]. For instance, the performance of Amazon AWS shows less performance variability for compute-intensive tasks, according to a report made by Cloud-Spectator[3]. The performance of Amazon AWS varies considerably for block storage operations. The distribution of the workloads may have a considerable impact on the performance. A provider's performance may show better performance when the workloads are evenly distributed. If the workloads come in burst or have heavy-tailed distribution, the performance may degrade [Al-Faifi *et al.*, 2018].

[3] https://cloudspectator.com/cloud-performance-reports/

Let us consider that the long-term workloads of the consumer consist of n data points, i.e., $t_1, t_2, t_3, ..., t_n$ over T period. The trial period can be divided into k data points over T_r period where $T_r << T$. If $n \leq k$, then each data point in the workload can be tested in the trial period. If $n > k$, then there are more workload data points than can be tested and compression is required. In such a case, the compression may result in some loss of data points. The amount of loss depends on *the size of* k, *the shape of the workload time series*, and *the compression method*.

We define $< W, M, T_r, k, Q_C >$ as a trial workload generation model where W denotes the long-term workload time series of a consumer, M is the compression method for the workload time series, T_r is the trial period offered by a provider, k is the number of points in T_r, and Q_C is the list of QoS parameters that needs to be measured in the trial. The goal of this model is to compress the workloads W using method M into k points to fit in T_r.

There exists a number of techniques for time series compression [Burtini *et al.*, 2013]. We apply three techniques, namely Piecewise Uniform Selection(PUS), Piecewise Aggregate Approximation (PAA), and Random Selection (RS).

(1) *Piecewise Uniform Selection (PUS):* The original workload time series is divided into k intervals [Burtini *et al.*, 2013]. Starting with the first workload, we select a workload point after each $\lceil n/k \rceil$ interval. The shape of the original workload time series may remain the same if the workload does not vary significantly over time.

(2) *Piecewise Aggregate Approximation (PAA):* The PPA method reduces the number of data points by taking average values in each interval I_i. If t_i represents a timestamp in the workload time series and n is the total number of points in the time series, then the value of the time series in I_j is calculated using the following equation:

$$I_j = \frac{1}{x} \sum_{i=(j-1)*x+1}^{i*x} t_i \text{ for } j = 1...\lceil n/x \rceil \tag{3.10}$$

The size of the original time series can be reduced by any factor by changing the value of x. For a given n data points in the workloads and k segments in the trial period, we define the minimum $x = \lceil n/k \rceil$.

(3) *Random Selection (RS):* The trial workloads are generated by selecting k number of workloads randomly from the original workloads. The random numbers are generated based on a uniform distribution in the interval (0,1).

3.6.2 Long-term QoS Prediction

The proposed framework monitors the required QoS parameters of the consumer (Q_C) during the trial period. The proposed framework expands the short-term QoS performances into the long-term performance using *time series interpolation* and *extrapolation*.

For a given set of workloads $W = \{w_1, ..., w_n\}$ and k number of points in the trial period T_r, the compression method M generates a set workload points $W' = \{w'_1, w'_2, .., w'_k\}$ for the trial. For each of these workloads, the corresponding QoS value is monitored in the trial. There are k number of values for each QoS parameter $q_{ci} \in Q_C$. There exists a one-to-one mapping between each w_i to $x_j \in q_{ci}$. The aim is to generate n number of QoS values from k number of QoS values of each $q_{ci} \in Q_C$.

The proposed framework leverages the knowledge of compression to expand the QoS values. The workload time series is divided into $\lceil n/k \rceil = x$ intervals during compression. The aim is to find $n - k$ points to determine the long-term performance. We apply the interpolation and extrapolation method to generate $n - k$ points. Interpolation is a well-known technique in the field of numerical analysis to construct new intermediate points between two points in a time series [Wiener and of Technology (Cambridge, 1950]. First, $n - k$ workload points are mapped into the different points of the time series. The mapped workload points are interpolated and extrapolated to generate intermediate workload points.

There are various algorithms to perform interpolation and extrapolation such as piece-wise constant, linear, nearest neighbor, and polynomial interpolation [Wiener and of Technology (Cambridge, 1950]. We select a commonly used interpolation technique, i.e., linear interpolation. If two consecutive workload points are denoted by (t_1, w_1) and (t_2, w_2), then an intermediate point is calculated by the following equation:

$$w = w_1 + (t - t_1)\frac{w_2 - w_1}{t_2 - t_1} \tag{3.11}$$

3.7 QoS-aware Long-term IaaS Provider Selection (QLIS)

The proposed framework selects the providers based on the confidence of the prediction when similar trial users are available. First, a subset of candidate providers is selected based on the confidence threshold (C^{thres}). This step is not performed when similar trial users are not available. Next, the proposed framework ranks the providers based on the distances of the predicted QoS performances and the consumer's QoS expectations. The predicted QoS performance and the consumer's QoS expectations are both represented using TSGs. Measuring the similarity between two TSGs can be costly when the number of QoS parameters is large. The cost of the pairwise comparison of the time series can be reduced if we can lower the dimension of the TSGs.

Principal Component Analysis (PCA) is an efficient approach to reduce the dimension of a TSG by transforming the original TSG into lower dimensional vector space [Mackiewicz and Ratajczak, 1993]. The PCA transformation is represented $D' = D \times T$ where D is the original data, T is the transformation vector, and D' is the transformed data matrix into the new vector space. Each data vector of the original space is represented by a column of D. Each row of T refers to a transformation vector. Each column of D' represents a principal component of the original vector space.

Let us consider that the set of discovered QoS is represented as TSG which is a $m \times n$ dimensional vector where m is the number of timestamps and n is the number of QoS parameters in the TSG. A vector $Q_t = (q_1^t, q_2^t, ...q_n^t)$ is created for each timestamp. Each dimension of the vector refers to a QoS time series in TSG'. We apply PCA transformation on the vector to identify the first n' QoS time series where $n' < n$ to get a new TSG TSG'. This n' contains the principal components of the discovered QoS performances. The transformation is computed as follows:

$$D'_t = D_t \times T = (D_t \times T_1, D_t \times T_2, ..D_t \times T_n) \tag{3.12}$$

where D_t is the original data, D'_t is the new data, and T_i is an n dimensional transformation vector. The values of D'_t for each timestamp is the first n' principal components of D_t. Given an n' dimensional transformation matrix $T' = (T_1, T_2, .., T'_n)$, the transformed TSG is as follows:

$$TSG' = D \times T' = (D_1 \times T', D_2 \times T', ..., D_m \times T') \tag{3.13}$$

where m represents the number of timestamps in each time series. Given the QoS requirements of the consumer $TSG1$, the transformed QoS performances $TSG2$ of a provider, and the number of the time series is n, the distance is calculated by Equation 3.14.

$$\text{distance } (TSG1, TSG2) = \sum_{i=1}^{n} \text{RMSE}(Q_i^1, Q_i^2) \tag{3.14}$$

The proposed framework ranks the providers based on normalized RMSE distances of their QoS performances with the consumer requirements.

3.8 Experiments and Results

3.8.1 Experimental Settings and Requirements

We conduct a set of experiments to evaluate the proposed framework. First, we investigate the dominant QoS attribute-based filtering process and compare it with the MTS-based temporal skyline [Wang *et al.*, 2013]. We then evaluate the effect of the proposed trial workload generation approaches on the long-term performance prediction. We then assess the performance of the proposed CLQP approach and the LQP-short approach. Finally, we evaluate the ranking accuracy of the proposed framework and compare it with the traditional trial-based ranking approaches [Li *et al.*, 2010; Wang *et al.*, 2018].

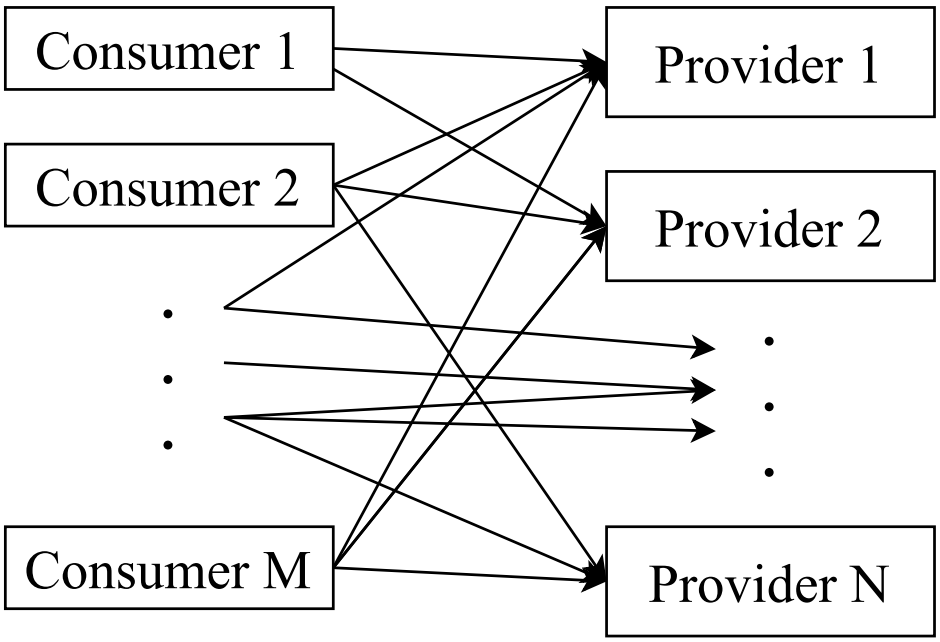


FIGURE 3.5: Experimental settings

To evaluate the proposed framework, we require an environment where a set of IaaS providers advertises their long-term performance, and a set of consumers who performs trials on different providers based on their workloads over different periods. Figure 3.5 depicts such an environment. Each consumer runs the same workload on different providers to decide which provider is the best for their workloads. The proposed framework would help a new consumer to select the closest match provider according to their QoS requirements. First, the proposed framework filters providers based on their advertisements. Next, it generates trial workloads based on the consumer's long-term workload to discover providers' performance in the trial period. Next, the framework utilizes the experience of past trial consumers to discover providers' performance on different periods outside of the trial period. Finally, it ranks the provider based on the discovered performance and the consumer's QoS requirements. Therefore, we require advertisements for a set of IaaS providers, trial workloads of a set of consumers, and QoS performance of providers at different periods for the consumers' trial workloads. In addition, we require a consumer's long-term workloads to perform the trials and the long-term performance of providers for the consumer's long-term workloads as ground truth to evaluate the proposed framework.

3.8.2 Experiment Design and Data Preparation

Finding real-world datasets for a long period that meet our experiment requirements is challenging. In particular, we require IaaS cloud workload traces and their corresponding performance datasets for a set of providers for a long-term period. To the best of our knowledge, there is no existing publicly available long-term workload-performance datasets of IaaS cloud providers. Therefore, we leverage existing short-term available datasets to synthesize datasets for our experiments. In particular, we use Eucalyptus IaaS Cloud workloads [Wolski and Brevik, 2017], and SPEC 2016 performance benchmark results [Baset *et al.*, 2017] to create experimental datasets. In the following subsections, we provide a brief description of the datasets that we used for the experiments and the data generation process to conduct the experiments. *We have made our dataset and source code publicly available to make this experiment reproducible*[4].

3.8.2.1 IaaS Advertisements Generation

IaaS advertisements in the real-world mostly contain short-term price and availability information. Therefore, we randomly create 60 IaaS advertisements that contain price, availability, throughput, and response time to understand the effect of a different number of dominant QoS parameters in the filtering process.

3.8.2.2 Workload Datasets Preparation

We utilize the Eucalyptus IaaS workload traces [Wolski and Brevik, 2017] to represent past consumers' trial workloads and a new consumer's long-term workloads. To the best of our knowledge, Eucalyptus private cloud usage traces are the closest match to validate the proposed approach with real-world datasets. The Eucalyptus traces have been utilized in several recent studies to represent a real-world cloud environment [Wolski and Brevik, 2017; Pucher, 2016]. Eucalyptus published workload traces of private cloud services, which are leveraged by several large companies [Pucher *et al.*, 2015]. The published datasets are anonymized multi-month traces scraped from the log files of six different production systems

[4] https://github.com/sm-fattah/IaaS_cloud_experiment

TABLE 3.2: Workload trace description

Parameter	Value
Number of cloud nodes	31
Number of CPU cores at each node	32
Number of timestamps	6486
Resource allocation unit	number of CPU cores
Trace attributes	VM start, stop timestamps, VM resource requests, VM id, node id

running Eucalyptus IaaS clouds. The trace contains information about both the IaaS service level (i.e., VM instance requests, request timestamps, instance lifetime, and service usage time periods) and physical level (i.e., number of cloud nodes, number of CPU cores at each node, and occupancy of physical hosts). We select a trace called "D6trace" which contains VM usage history of a large company with 50,000 to 100,000 employees [Pucher *et al.*, 2015]. It contains approximately 34 days of workloads for 31 cloud nodes where each node contains 32 cores. We selected 30 cloud nodes to represent the trial workload of past consumers and one cloud node to represent a new consumer's long-term workloads. The workload contains 6486 timestamps. To simplify the experiment, we utilized the piece-wise aggregate approximation to create 360 timestamps. We assume that each workload at a timestamp represents average CPU requests of a consumer on that timestamp. The description of the workload dataset is given by Table 3.2.

3.8.2.3 Workload-Performance Dataset Generation

Finding performance datasets of a long-term workload trace is very challenging. Therefore, we decide to synthesize the performance dataset for our selected workload traces. We develop a performance data generation framework that generates the QoS performance of a set of providers for a given workload for a given period of time. We consider performance data for three QoS attributes based on the available dataset, i.e., CPU throughput (operations/seconds), disk read and insert time (milliseconds). Figure 3.6 shows the performance data generation framework. The framework takes a consumer's workload W for a given period T , and a provider's id P as inputs. It generates corresponding QoS performance using two modules

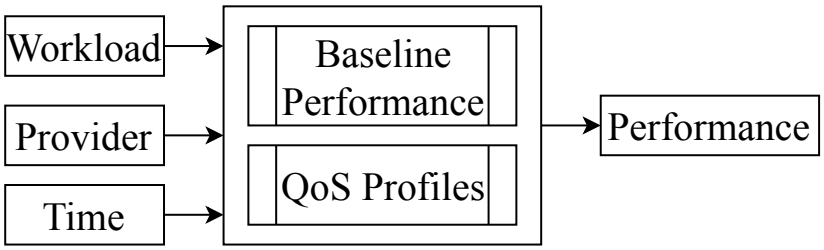


FIGURE 3.6: Data generation framework

a) Baseline performance, and b) QoS profiles of a set of IaaS providers. The baseline performance module maps each unique workload request of the Eucalyptus trace to a particular performance value. Therefore, the baseline performance represents the initial performance of a provider for a given workload. The QoS profiles module determines the final performance value of the workload at a given time for the providers based on their QoS profiles. We performed the following steps to generate the baseline performance and QoS profiles:

1) Baseline Performance: The baseline performance is generated from the benchmark results published by SPEC Cloud IaaS 2016 [Baset *et al.*, 2017]. SPEC Cloud IaaS 2016 benchmark results contain the CPU throughput (op/sec), disk insert and read response time (ms) measurements of private cloud providers. We have collected approximately 1500 performance data observations from the benchmark results. To create a *workload-QoS map*, we map each unique workload request to a unique performance value based on the resource consumption of the requests. A workload with the highest resource consumption is mapped with the lowest QoS performance value. The lowest resource consumption workload is mapped with the highest QoS performance value. The baseline QoS performance is utilized to generate long-term performance of IaaS providers with the help of *QoS profiles* of each provider.

2) QoS Profiles: We decide to create QoS profiles of five IaaS providers to simulate a real-world environment. A QoS profile determines what a provider's performance for a given workload would be at a certain point of time compared to the baseline performance. For example, if a workload with a low resource consumption is given in January to a provider, it offers 20% additional throughput compared to the baseline performance. Each QoS profile consists of two maps a) workload map, and b) seasonal map. The workload map determines

what the expected performance for a given workload would be compared to the baseline performance. Similarly, the seasonal map determines what the performance for a given time compared to the baseline performance should be. We have created the workload map and the seasonal map randomly. The rules of QoS profiles also add randomness to simulate a real-world provider who may give different performances for the same workload at the same time. The consumer and trial users are unaware of the QoS profile. They may observe the performance variability of the providers. Finally, these QoS profiles of providers are applied to the baseline performance of a given workload to generate the performance dataset of each provider.

We input the trial workloads of past consumers and a new consumer's long-term workloads from the Eucalyptus trace into the data generation framework. The framework generates performance data for each provider based on their QoS profiles and the baseline performance. The performance of the long-term workloads is utilized as the *ground truth* for the evaluation.

3.8.3 Experiment Result Analysis

3.8.3.1 Effect of Dominant QoS Parameters on IaaS Skyline

We apply the filtering on 60 IaaS providers for four QoS parameters. We select the price and availability as dominant QoS parameters. Figure 3.7 compares the performance of the temporal IaaS skyline without the dominant QoS parameters and with the dominant QoS parameters. The size of the temporal skyline without Dominant QoS parameters increases considerably with the number of providers. If there are too many IaaS providers in the cloud market, the temporal skyline without the dominant QoS parameter may not be very useful. It only discards 24 providers when the number of providers is 60. The temporal IaaS skyline with the dominant QoS parameters has a considerable impact on the total number of skyline IaaS providers. It always filters more IaaS providers than the temporal IaaS skyline. The main reason is that the size of the IaaS skyline may grow exponentially with the number of QoS parameters. The dominant QoS parameters keep the size of the skyline considerably low. The proposed method successfully reduces the number of IaaS candidates for the trial.

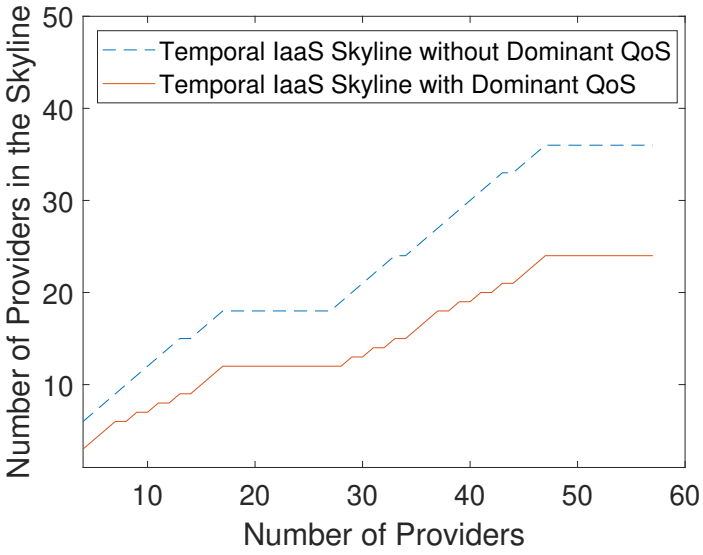


FIGURE 3.7: Temporal IaaS skyline

3.8.3.2 Evaluation of LQP-short approach

Figure 3.8(a) shows a consumer's workload for 360 timestamps. The average requested number of CPU cores per timestamp is shown in the figure. Three types of trial workloads are generated using three compression techniques (PUS, PAA, and RS). The trial QoS performances are generated for each provider based on their QoS profiles using the data generation framework. The LQP-short mainly utilizes the trial QoS performances to predict each provider's long-term QoS performance.

Figure 3.8(b) shows the actual throughput and the predicted throughput of a provider based on the LQP-short and three trial workload generation approaches. The provider exhibits variable performance for the same workload according to the actual performance graph. There are substantial performance fluctuations over different timestamps. The predicted QoS performance exhibits random behavior. The effect of different trial workload generation technique is not very noticeable. The RS based prediction shows more fluctuations than are shown by the other two approaches. The QoS predictions seem more accurate when the consumer's workloads are steady (200 to 300 days) as the trial is performed in that period. Apart from the trial period, the LQP-short approach seems to perform poorly. None of the trial workload generation approaches help to capture the temporal shift in the performance. The

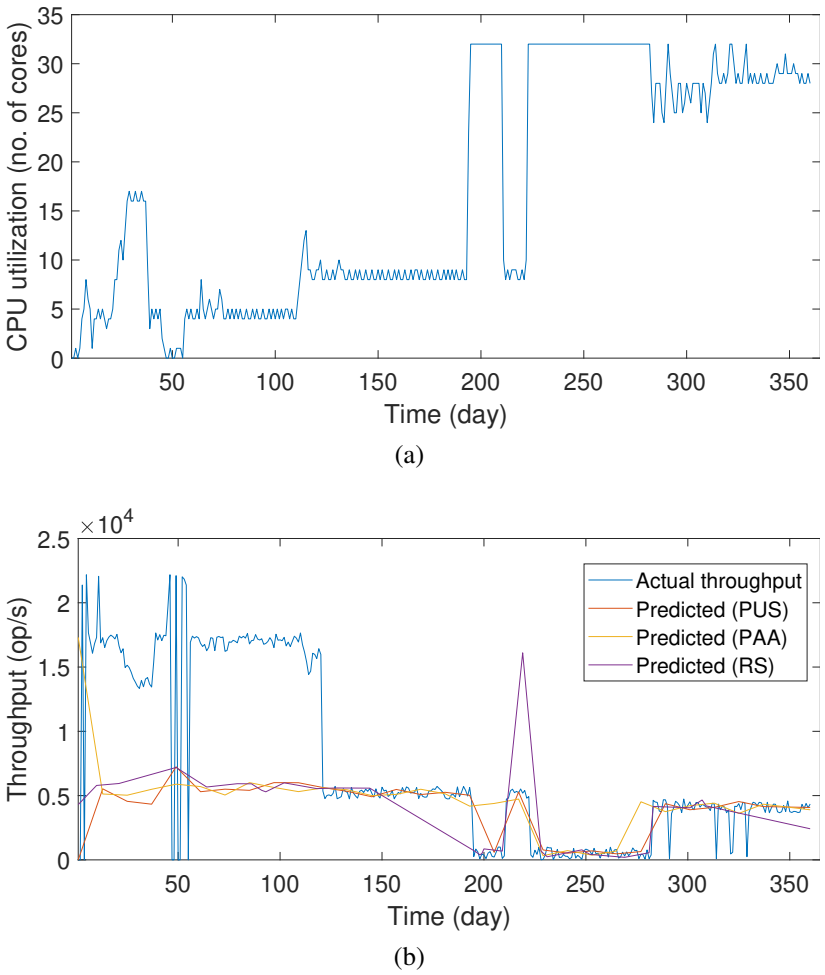


FIGURE 3.8: Performance discovery using LQP-short (a) consumer's long-term workloads (b) actual and predicted throughput of an IaaS provider

main reason for such low accuracy is because the prediction is performed without utilizing past history. The LQP-short can be considered a best-effort approach when there are no similar users available.

The prediction accuracy of the LQP-short approach is presented in Figure 3.9. The throughput prediction accuracy of each provider is shown in Figure 3.9(a) for each trial approach. The accuracy of the RS method is unpredictable across the providers as the workloads are selected randomly. The RS method has the highest NRMSE distance compared to the other two approaches for each provider. *This confirms that an unplanned short-term trial may lead to poor decision making.* The performance of the PUS approach is consistent for each provider.

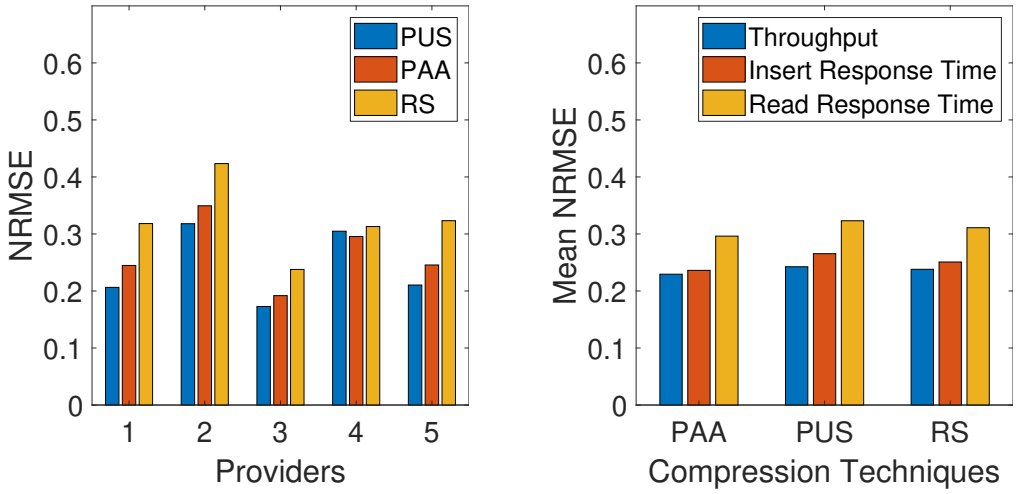


FIGURE 3.9: Prediction accuracy (NRMSE distance) (a) throughput prediction (b) mean prediction accuracy

The PUS method performs better as it retains the shape of the original workloads better than the other two approaches. Hence, choosing the types of workload has a considerable impact on trial prediction accuracy. Figure 3.9(b) shows the mean prediction accuracy across five providers for the QoS parameters throughput, insert response time, and read response time. The trial workloads generated by the RS method has the maximum NRMSE distance for throughput and read response time. The PUS has the minimum NRMSE distance as it performs well for each QoS parameter.

3.8.3.3 Evaluation of CLQP approach

We divide the total provisioning time into smaller segments to find similar trial users. The provisioning time is divided into 12 temporal segments to perform the cooperative prediction, where each segment is 30 days. We run CLQP using past trial user experiences and the consumer's long-term workloads for each month. Figure 3.10(a) illustrates the results of the cooperative long-term QoS prediction for an IaaS provider. The accuracy of the prediction improves considerably over the prediction without the history in Figure 3.8(b). The NRMSE distance between the CLQP approach and the actual performance is about 0.1343. The NRMSE distance between the sort-term approach and the actual performance is about 0.2955.

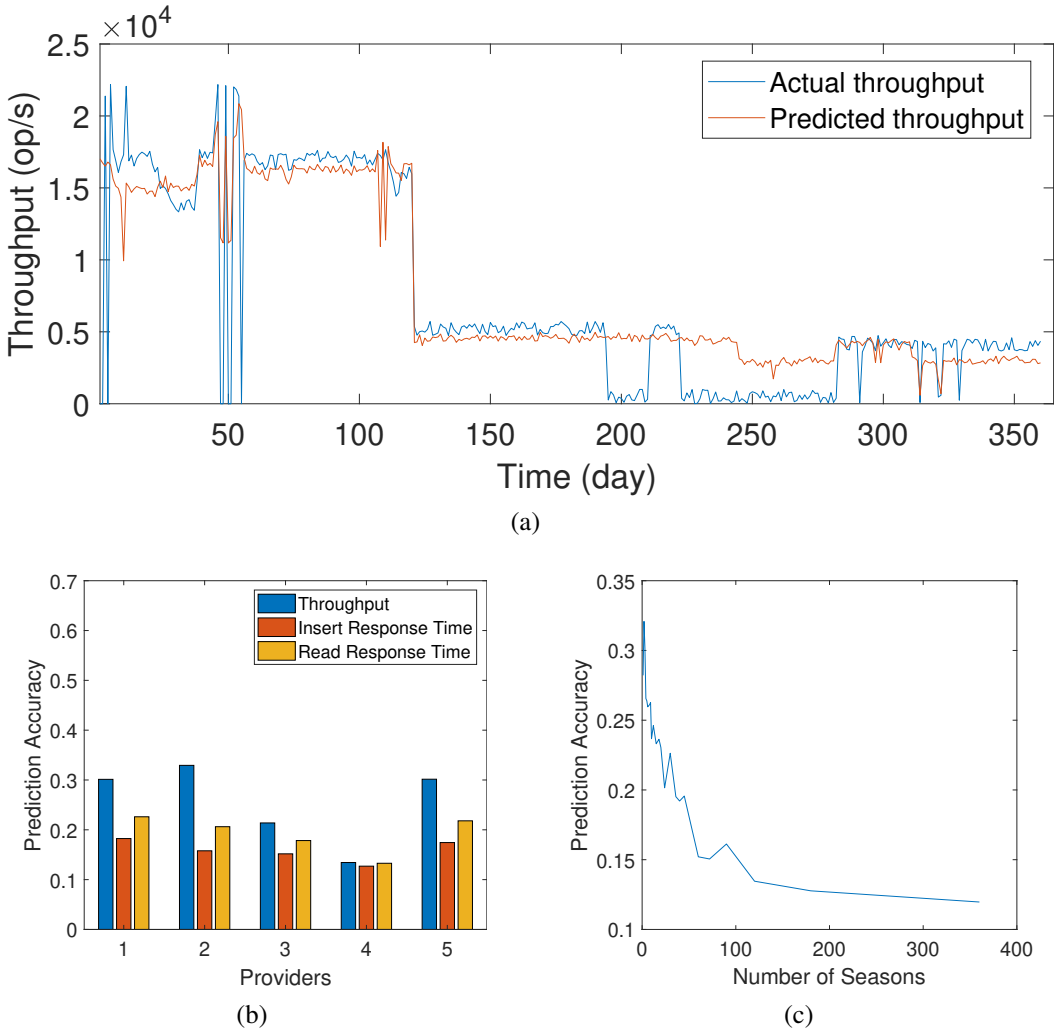


FIGURE 3.10: Performance discovery using CLQP (a) actual and predicted throughput of a provider (b) prediction accuracy of the CQLP approach (C) CQLP accuracy for variable temporal segments

Therefore, the accuracy is improved about 50%. The prediction effectively captures the temporal and workload performance variability of the provider. The randomness of the performance, i.e., different performance for the same workload at the same time is also captured effectively in the prediction. The accuracy of the predictions is shown in Figure 3.10(b) in terms of NRMSE distance. The throughput of each provider has a lower prediction accuracy as the NRMSE distance is high. The read response time for each provider has the highest prediction accuracy for each provider. It implies the effect of the performance variability may depend on the type of QoS attribute.

Our intuition is that the number of temporal segments may have a considerable impact on accuracy. When each temporal segment has a smaller length, the probability of finding a similar trial user increases. We conduct the experiment by changing the size of the temporal segments for the cooperative prediction. Figure 3.10(c) illustrates the results of cooperative QoS prediction for different sizes of temporal segments. It shows that the NRMSE distance decreases considerably with the number of the temporal segments. When the number of segments is one, i.e., the similarity is computed based on the one-year workloads of the consumer, the prediction accuracy is very low. However, if the similarity is computed based on smaller segments, the prediction accuracy increases.

3.8.3.4 Evaluation of QLIS approach

We rank the providers based on the consumer's long-term QoS requirements and the predicted QoS performance of each provider. First, we rank the provider based on the actual QoS performance and expected QoS performance of the consumer. Next, we compare this rank with the proposed approaches, i.e., ranking based on cooperative prediction and long-term prediction without historical information. We consider two traditional ranking approaches based on 1) trial performance and 2) advertised performance [Wang *et al.*, 2018]. Each rank is computed based on the NRMSE distances of the expected QoS performance and predicted QoS performance. Table 3.3 shows the ranks of the provider based on different methods. Each provider is identified numerically from 1 to 5. The actual ranking order is 5, 2, 3, 1, and 4. The cooperative ranking successfully predicts the rank of 3 providers (5, 2, and 3). The long-term prediction without history and the trial-based ranking approaches correctly rank two providers (5 and 3). The ranking of the providers based on the advertisements provides the most inaccurate ranking of the providers. The results of the rankings may change if we choose different months for the trial.

The confidence of the cooperative ranking is shown in Table 3.3 for each provider. The highest-ranked provider has the highest confidence as it has the least distance between the trial experience and the predicted performance. The first four providers in the cooperative rankings have high prediction confidence. The prediction for Provider 1 has the least confidence

TABLE 3.3: Ranking of IaaS providers

Actual Ranks	Cooperative Ranks	Long-term Ranks	Trial Ranks	Advertisements Ranks
5	5 (0.46)	5	5	1
2	2 (0.53)	4	4	3
3	3 (0.58)	3	3	5
1	4 (1.44)	2	2	4
4	1 (2.03)	1	1	2

value. The reason is that the performance of the last provider varies to a large extent in the trial period. When the confidence threshold is set to below 1, the last two providers will be discarded from the cooperative rankings.

3.8.4 Discussion

Experimental results show that relying only on IaaS advertisements or the trial experience is inadequate for long-term service selection as it often leads to the incorrect selection. Moreover, the results show that selecting appropriate trial workloads has a substantial impact on long-term performance discovery and selection. The results of the CLQP approach show that long-term performance discovery could be improved considerably with the help of past trial users' experience. Based on these findings, we suggest that consumers should perform trials based on the long-term characteristics of their workloads instead of relying only on stress testing where consumer performs short-term trial to evaluate the providers. The proposed framework could be utilized to generate trial workloads based on a consumer's workload for long-term performance discovery. The performance discovery will be improved considerably when many consumers decide to share their trial experience.

3.9 Summary

We propose a long-term IaaS provider selection framework to select the closest-matched IaaS provider according to a consumer's long-term requirements. The short-term trial periods

offered by the IaaS providers are leveraged to discover the providers' unknown QoS performance. We devise a temporal skyline-based filtering method to limit the number of candidate IaaS providers for the trial periods. Experimental results show that the filtering method effectively reduces the number of candidate providers for the trial period. The proposed CLQP approach utilizes the experience of trial users to predict the performance of a provider for the consumer's long-term workloads with a confidence measure. Experimental results show that the CLQP approach can effectively measure QoS performance. The proposed LQP-short approach discovers QoS performance without history using the proposed trial workload generation approach. Experimental results show that the LQP-short can effectively predict QoS performance with acceptable precision. Finally, a QoS-aware selection method is proposed to select the closest-matched provider where the provider's predicted performance closely matches the consumer's long-term requirements. Experimental results show that it ranks the providers successfully. The proposed framework may help many consumers choose the right provider with limited performance information. Consumers do not need to wait long to accumulate enough information to make an informed decision. The experiments are conducted using synthetic data, as finding real-world datasets that meeting the experiment requirement is challenging. However, the synthetic datasets are created by augmenting real-world datasets. Therefore, we believe the result would be similar if the experiments were conducted directly using real datasets. In the future, we will extend this work for the dynamic workloads of consumers where an online prediction approach is required.

CHAPTER 4

Fingerprint-based Long-term IaaS Selection

4.1 Introduction

In this chapter, we propose a new long-term IaaS cloud service selection approach that leverages the concept of performance fingerprints for the long-term selection. The performance fingerprint of an IaaS service represents an aggregated view of its temporal performance behavior. We assume that the performance fingerprints of IaaS services are known in this work and can be generated by collecting the experience of free trial users. We also assume that a consumer considers only one service from a provider for their long-term selection. *Therefore, we use the word "provider" and "service" interchangeably.*

We address two key challenges of using trial periods for long-term selections. First, IaaS providers typically offer free trial periods for short-term periods with limited flexibility. The consumer cannot test their long-term workloads in such short trial periods. An *unplanned* utilization of such short-term trial periods may not properly reflect the actual performance of the provider. For example, if the workloads of a consumer have a long-tailed distribution, a one-month trial with a balanced request distribution may not divulge the true performance of long-tailed workloads. Second, the performance information found in the trial periods is only applicable to a short-term period. The performance of IaaS providers varies over time due to the *dynamic* and *chaotic* nature of the cloud environment [Leitner and Cito, 2016].

The performance observed in trial periods primarily depends on the consumer's workloads and the provider's performance at that time. A provider may exhibit variable performance behavior depending on the user's application workload. For instance, a provider may show a

very good performance for a CPU-intensive workload while performing poorly for a network-intensive workload. Similarly, the provider may show a very good performance during the trial month and perform poorly in the following months. Therefore, the effect of both factors should be identified during the trial. *In this regard, we propose a novel trial strategy based on an equivalence partitioning method to capture the effect of the consumer's workloads on the provider's performance while considering the provider's temporal performance behavior.*

Selecting a provider based only on trial experience is challenging due to the performance variability of cloud services. We utilize the concept of performance fingerprints to address the issue of performance variability in the long-term selection. The performance fingerprint has two applications in the long-term selection. First, the fingerprint of a service is used to measure the confidence of the free trial experience of a consumer to evaluate whether the observed performance in the trial is the "representative" behavior of the provider. Second, the performance fingerprint and the trial experience can be utilized together to predict a provider's long-term performance behavior.

We propose a fingerprint-matching technique to ascertain the confidence of the consumer's trial experience for long-term selection. If the trial experience of a consumer is consistent with a provider's performance fingerprint, we utilize the fingerprint to predict the provider's long-term performance for the consumer's long-term workloads. The trial experience may not entirely match the performance fingerprint as it represents an aggregated view of the provider's performance regardless of the consumer's workloads. The provider may provide an isolated trial environment where a consumer may not be able to observe its actual performance. We propose a trial experience transformation technique using the provider's performance fingerprint to estimate the actual performance of the provider for the consumer's workloads. Our contributions in this work are as follows:

- An equivalence partitioning-based trial strategy using a time series compression technique that maps the consumer's long-term workloads into multiple VMs in a short-term trial period to discover an IaaS provider's unknown QoS performance.
- A performance fingerprint-matching technique to ascertain the confidence of the consumer's trial experience using the providers' performance fingerprints.

- A long-term performance discovery approach to select the best IaaS provider for a consumer using time series analysis.

4.2 Motivation Scenario

Let us assume that a university requires some general purpose VMs for one year where each VM has at least 2 vCPU and 4 GB memory. The required number of VMs and resource requirements for each VM are considered as the functional requirements of the university. We assume the university has deterministic workloads, i.e., workloads are known for one year. The university represents the workloads in terms of the number of requested resources per day. The workloads may change over time depending on the number of students, holiday periods, and so on. The university defines minimum QoS requirements on throughput, response time, and availability of the VMs. The QoS requirements may also vary over time depending on seasonal demands.

Let us assume that there are three IaaS providers - Google, Amazon, and Microsoft - who fulfil the university's functional requirements. No providers advertise their long-term performance on throughput, response time, or availability. We assume each provider offers a one-month free trial period to the university and allows the university to use three VMs. The university may run some representative benchmarks on three VMs for each day of the one-month trial period and monitor the performance of each provider to make the selection. It may lead to poor decision-making as it does not consider the university's long-term workloads and the providers' temporal performance behaviors. The performance of a provider may fluctuate in the trial period. The university requires an effective trial strategy to understand the effect of different types of workloads on the provider's performance while considering the provider's temporal performance behavior.

We assume that the performance fingerprint of each provider is known to the university. The performance fingerprint provides the university with an aggregated view of a provider's temporal performance behavior regardless of any specific type of workload distribution. Hence, the university needs to evaluate the performance of the providers using its workloads.

If the provider's performance fingerprint and the trial experience exhibit similar temporal performance behavior, the university may use the trial experience to evaluate the provider with high confidence. The university requires a fingerprint-matching technique to evaluate its trial experience. We propose a set of tools in this chapter that enables the university to leverage trial periods effectively to make an informed decision for the long-term period.

4.3 The Proposed Framework

We identify the following key challenges for the long-term selection using free trial periods:

(1) *Restriction on Trial Periods*: IaaS providers assign different types of restricted conditions on free trial periods:
   - Free trial periods are typically offered for a short-term period. Discovering long-term performances directly from short-term trials may not be possible. Amazon offers a one-year trial period for some services. A consumer may not be able to wait for such a long time to discover performances for the IaaS selection.
   - Most providers offer trial periods for only a limited number of services. For example, Amazon allows a user to trial only t2 instances from EC2 VMs. The types of VM a consumer requires may not be available for trial. In such a case, the consumer might be provided with similar yet different types of VMs for the trial. IaaS providers also restrict the number of available VMs for trial.

(2) *Temporal Performance Variability:* The performance discovered in the short-term trial periods may not always reflect the actual performance of the provider. Almost all public IaaS providers typically use multi-tenant environments to provide services to their consumers. The effect of multi-tenancy on the performance may depend on several factors such as location, workloads on the provider, and QoS management strategies that vary with time. Multi-tenancy management policies are not revealed publicly due to the business privacy of the providers. The measured performance in the trial in one month may be different in another month.

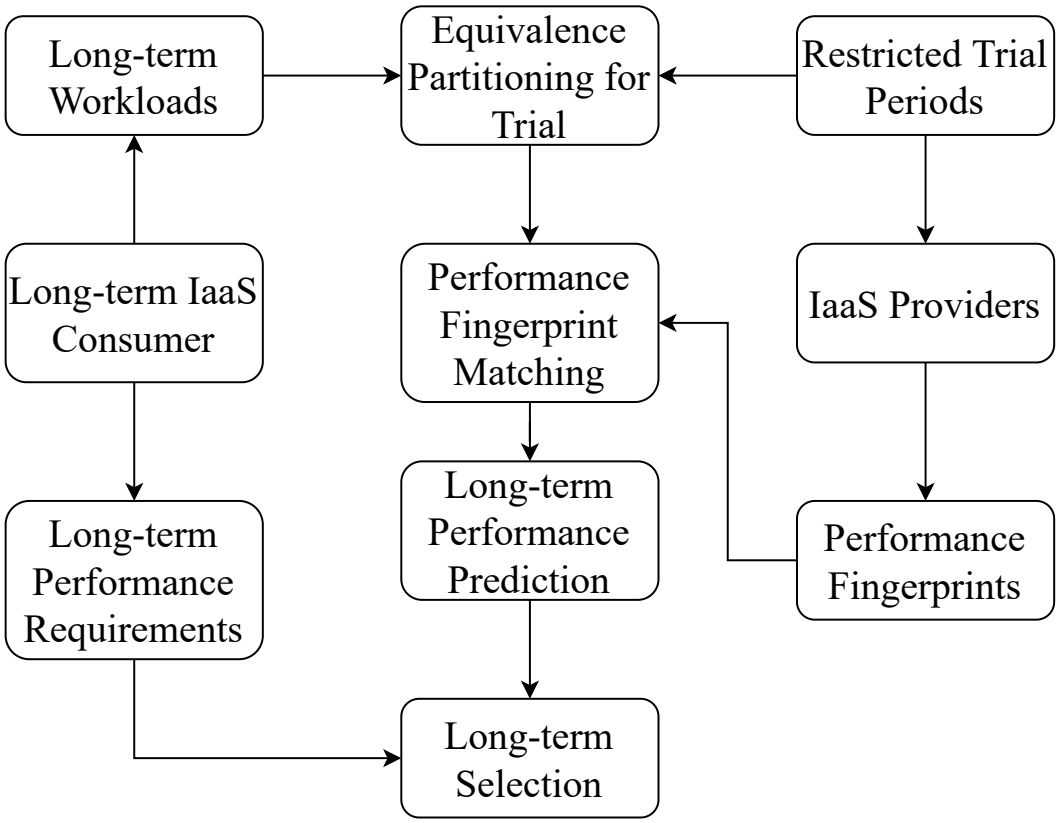


FIGURE 4.1: Long-term IaaS provider selection framework

(3) *Isolated Trial Environment:* IaaS providers may use an isolated environment for the trial users. In such a case, the trial consumers do not receive the experience of a real cloud environment. The consumers require a way to find whether they are treated differently than the existing consumers.

Figure 4.1 shows an IaaS provider selection framework that takes a consumer's long-term workload and the performance fingerprints of the providers to perform the selection. First, the proposed framework generates trial workloads using an equivalence partitioning method. Next, A performance fingerprint matching technique is applied to the trial experience to ascertain its confidence. The trial experience is then used for long-term performance prediction using the providers' fingerprints. Finally, the framework selects providers based on the consumer's long-term performance requirements. We discuss each of these steps in the following sections.

4.4 An Equivalence Partitioning-based Trial Strategy

We define an equivalent partitioning-based trial strategy where the consumer's workloads are tested in the trial period to discover a provider's performance while considering the provider's temporal performance fluctuation. For simplicity, we assume that the providers offer a fixed number of required VMs trial periods using a *continuous time-based model*. We consider the long-term workloads as time series data. We utilize time series compression techniques to capture the essential characteristics of the university's long-term workloads and map these workloads into the multiple VMs during the trial period.

4.4.1 Trial Workload Generation for Multiple VMs

Let us assume that the university's long-term workloads have n number of workload data points, i.e., $t_1, t_2, t_3, ..., t_n$ over T period. For instance, the university defines the workload as the average number of requested resources per day for one year. Each provider offers v number of VMs for T_r trial period. *We assume that the performance fluctuation within d period is negligible.* Hence, workloads for a particular VM should remain the same for every d period over T_r period to understand the effect of temporal performance behavior of a provider. Each VM may run different types of workloads to understand the effect of the provider's performance for different types of workloads. This method of partitioning the workloads is called *equivalent partitioning*.

The university's long-term workloads of n size need to be mapped with v number of VMs on T_r period. First, we partition the workloads into n/v equal parts. Let us assume that each part w contains m workload data points. Once we allocate the workloads for each VM, we need to compress the workload as w may be still very large to run in d period. The size of w may be still too large to run in d period. For example, if the university has one-year workloads and the number of VMs are 12, each part of the workload contains one month of workloads. Each VM should run one month of workloads on every day ($d = 1$) of the trial period T_r.

We need to compress each w into d period for each VM. Let us assume that d can be divided into k data points. If $m \leq k$ then each workload of w can be tested in d period. If $m > k$, then workload compression is required. The compression may incur some loss of workload information. The loss depends on *the size of* k, *the shape of the workload time series*, and *the compression method* [Burtini *et al.*, 2013].

We use a compression technique M to extract the most important workloads from w and fit into d periods. $L()$ is the loss function that calculates the loss incurred during the compression using M and th is the maximum acceptable loss. During compression $L(M) \leq th$ condition must be held. th is defined by the consumer. Let us assume d can be divided into $k-1$ intervals. Method M compresses w to fit into $k-1$ intervals. Our target is to find an optimal value for k where $L(M) \leq th$. We use Algorithm 2 to generate workload for d period. The algorithm takes each w from the university's long-term workload, the length of d, a compression technique M, a loss function $L()$, and minimum acceptable loss th (line 1). The algorithm produces trial workload tw as output (line 2). First, it determines the initial value of k by using a *workload summarization* method (line 5). The workload summarization technique generates all different workloads for given workloads. The algorithm sets the sampling rate $rate$ based on the size of the workload m and the initial value of k (line 6). The algorithm then applies the compression method M using $rate$ and calculates the amount of loss using $L()$ (line 12). If it is less than an acceptable threshold, this process continues until the amount of loss $error$ is less than the maximum acceptable loss of th (line 9). Once $error \geq th$, the algorithm stops and returns the trial workload tw (line 15).

4.4.2 Workload Compression Technique

The university's long-term workloads are represented using time series. Several approaches exist to compress time series data to generate a representative time series. We decide to use a commonly used time series compression technique called Piecewise Aggregate Approximation (PAA). The PAA method reduces the number of data points in a time series by taking average values in each interval (I_i). If t_i represents a timestamp in the workload time series and m is

Algorithm 2 Generating trial workloads

Input: w, d, M, $L()$, th
Output: tw
$m \leftarrow size(w)$;
$workloadSummary \leftarrow summary(w)$;
$k \leftarrow size(workloadSummary)$;
$rate \leftarrow ceil(m/k)$;
$errorThresh \leftarrow th$;
$error \leftarrow 0$;
while $error < errorThresh$ **do**
    if $rate == size(w)$ **then**
        break;
    $tw \leftarrow M(w, rate)$
    $error \leftarrow L(w)$
    $rate = rate + 1$
return tw;

the total number of points in the time series, then the value of the time series in I_j is calculated using the following equation:

$$I_j = \frac{1}{x} \sum_{i=(j-1)*x+1}^{i*x} t_i \text{ for } j = 1...\lceil m/x \rceil \tag{4.1}$$

The size of the original time series can be reduced by any factor by changing the value of x. For given m data points in the workloads and k segments in the trial period, we define the minimum $x = \lceil m/k \rceil$. The PAA method introduces some loss of information. For two given time series w and z, the loss is defined by the following equation:

$$L = \frac{1}{m} \sum_{i=1}^{n} |z_i - w_i| \tag{4.2}$$

The original workload time series and the compressed workload time series have a different number of workload points. We decompress the compressed workload time series to compare with the original workload time series. We find k workload points using Equation 4.1 during the compression. We apply a decompression mechanism on the compressed workload of k points to generate n points. The decompression is performed by mapping each value

of the compressed time series with $\lceil m/k \rceil$ interval into m space. The rest $m - k$ points are generated by the linear extrapolation method. After decompression both original and compressed workload time series have the same number of points. Hence, we can apply Equation 4.2 to compute the *mean absolute error*.

4.5 Performance Fingerprints

The performance information found in the trial periods is applicable for a short-term period. Many organizations such as CloudSpectator, CloudHarmony, and CloudStatus are devoted to monitoring and analyzing the performance of public IaaS cloud providers due to its growing importance. These organizations publish reports on the performance of IaaS providers using standard benchmarks. Each provider shows unique performance characteristics and exhibits different temporal performance behavior over the long-term periods. Each provider has their own unique temporal performance behavior that may depend on their provisioning policy, the number of consumers, and location.

We leverage the idea of fingerprinting to represent the temporal performance behavior of a provider. Fingerprinting techniques are well-known to identify and track a user on the Internet based on the impression left by the user [Takeda, 2012]. Fingerprinting techniques are typically used to partially or fully identify a user on the Internet by tracking their activity and preferences without any active identification. We use the concept of *performance fingerprint* of an IaaS provider to represent an aggregated view of the provider's long-term performance behavior.

DEFINITION 7. *Performance Fingerprint.* A performance fingerprint of an IaaS provider is the average performance of a set of QoS parameters over a fixed period that captures the provider's temporal performance behavior.

We denote the performance fingerprint as $F = \{Q_1, Q_2, ..Q_N\}$ where N is the number of QoS parameters and $Q_i = (P_n, t_n)|n = 1, 2, 3, ..k$. Here, t_n denotes a timestamp of T period where the average performance of Q_i is P_i. The performance fingerprint of a provider may

be known partially or completely. The partial fingerprint refers to a fingerprint that does not have information for all timestamps of a certain period.

4.5.1 Performance Fingerprint Matching

We utilize the performance fingerprint of each provider to ascertain the confidence of the trial experience. If the trial experience is consistent with a provider's performance fingerprint, then the consumer may make the selection with confidence based on the trial experience.

We assume that the complete performance fingerprint of the providers is known for T period. The trial is performed in the interval $(t_j, t_k) \in T$, i.e., $T_r = (t_j, t_k)$ where $j < k$ and $j, k \in T$ for a set of VM $v = \{v_1, v_2, ...v_p\}$. The trial performance observed by the consumer for each VM is $Q_{vi} = \{q_1, q_2, ..q_c\}$ where c is the number of QoS parameters in the consumer requirements and $q_i = \{(p_n, t_n)|n = t_j, ...t_k\}$ where p_n is the performance of q_i at the timestamp t_n. The first step of fingerprint matching is to aggregate the performance of each VM v_i for each QoS parameter $q_i \in Q_{vi}$. The aggregated performance for each QoS parameter is computed by the following equation:

$$q_i' = \text{sum}(v_1(q_i), v_2(q_i), ..., v_p(q_i)) \tag{4.3}$$

where $sum()$ represents the aggregate function, $v_j(q_i)$ represents the performance time series of QoS parameter q_i in the VM v_j in the trial period. The aggregated QoS performance of q_i for all VM is q_i'.

We denote the performance of the trial period for the consumer's aggregated workloads as $Q_{VM} = \{q_1', q_2', ..q_c'\}$. Now, we need to perform fingerprint matching between Q_{VM} and F for the trial interval (t_j, t_k). We use the *Pearson correlation coefficient* to compute the similarity between the trial experience and the performance fingerprint for each QoS parameters using the following equation:

$$r_{q'_i,q_i} = \frac{\sum_{t=j}^{k}(p'_t - \bar{p'})(p_t - \bar{p})}{\sqrt{\sum_{t=j}^{k}(p'_t - \bar{p'})^2}\sqrt{\sum_{t=j}^{k}(p_t - \bar{p})^2}} \tag{4.4}$$

where p'_t the value of observed performance and p_t is the value of the performance fingerprint at the time t of the trial period for a QoS parameter. The mean correlation coefficient for all QoS parameters is computed as follows:

$$R_{Q_{VM},F} = \frac{1}{c}\sum_{i=1}^{c} r_{q'_i,q_i} \tag{4.5}$$

The correlation coefficient measures the similarity of two time series in terms of their trends, i.e., how much the trial experience is affected by the performance fingerprint. It does not consider the actual distance from the fingerprint. We define the confidence of the trial by considering both trend and distance of the trial experience with the fingerprint using the following equation:

$$\text{Confidence} = (R_{Q_{VM},F}, \text{MNRMSE}(Q_{VM}, F)) \tag{4.6}$$

where MNRMSE is the *mean normalized root mean squared error* between the trial experience and the performance fingerprint. First, we compute the NRMSE for each QoS parameter in the trial period. The MNRMSE is computed by taking average NMRSE of all QoS parameter. The consumer defines a minimum threshold R_t and E_t for $R_{Q_{VM},F}$ and MNRMSE(Q_{VM}, F) respectively. If the confidence of the trial experience is below the thresholds, we consider it to be a partial fingerprint matching.

4.5.2 Trial Experience Transformation for Partial Matching

We transform the trial experience for partial fingerprint matching to estimate an approximate performance behavior of the provider for the consumer's workloads. The trial experience may have less correlation or higher distance with the performance fingerprint of a provider. We

need to transform the trial experience in such a way that it reduces the distance or increases the correlation between the trial experience and the performance fingerprint. In both cases, the confidence of the trial may increase.

The partial fingerprint matching indicates that the actual performance of the provider may be different from the trial experience for the consumer's workloads. We use the following equation to transform the trial experience to estimate the actual performance:

$$Q^T_{VM} = Q_{VM} + \frac{1}{2}(F - Q_{VM}) \tag{4.7}$$

where Q^T_{VM} is the transformed trial experience, F is the performance fingerprint in the trial interval, and Q_{VM} is the trial experience. Equation 4.7 transforms the trial experience for each aggregated QoS parameters by reducing the distance from the fingerprint by half. The intuitive idea behind this transformation is two-fold. First, if the provider offers an isolated trial environment, the real experience may be closer to the fingerprint rather than the trial experience. Second, the performance fingerprint does not contain information for the consumer's workload distribution. The transformation increases the confidence of the trial experience.

4.6 Long-term IaaS Provider Selection

In this section, we discuss the long-term selection process using the trial experience and the provider's performance fingerprint. First, we estimate providers' long-term performance for the consumer's workloads using the trial experience and the performance fingerprints. Next, we rank the providers based on their performance and the consumer's long-term performance requirements.

4.6.1 Long-term Performance Discovery

The university's important workloads are tested in the trial periods and the required QoS parameters are monitored. Let us assume that the trial workloads $TW = \{tw_1, tw_2, ..., tw_k\}$

are tested in v number of VMs. Each type of workload is monitored in the trial period T_r for each d intervals where the performance fluctuation in d is negligible. The QoS performance for each workload is denoted by $Q_{tw_i} = \{q_1, q_2, .., q_c\}$ where $q_i = \{(p_n, t_n)|n = 1, d, ..., Tr\}$ and p_n is the performance observed at the timestamp t_n. The consumer's long-term workloads are denoted by $LW = \{W_1, W_2, ..., W_T\}$. We need to find the performance for each W_i which is denoted by Q_{W_i}. The trial performance Q_{tw_i} and the performance fingerprint F are used to generate Q_{W_i}. We use the following steps to compute Q_{W_i}:

(1) For each $W_i \in LW$, find the closest $w_i \in TW$.

(2) For each $q'_i \in Q_{W_i}$, find $q_i \in Q_{tw_i}$.

(3) Let t'_i be the timestamp of W_i. tw_i has Tr/d number of observations. We select a timestamp t_i for tw_i where $(t_i - t_1) = (t'_i \bmod (T/Tr))$ where T is the total time and Tr is the trial period. For example, if $Tr = 30$, $T = 360$, and $t'_i = 35$ then $t_i = 5$.

(4) We compute the relative weight r_w of the fingerprint at t_i for q'_i using the following equation:

$$r_w = \frac{P_{t'_i}}{P_{t_i}} \tag{4.8}$$

where $P_{t'_i}$ and P_{t_i} are the performance of the fingerprint at timestamp t'_i and t_i respectively.

(5) The performance of q'_i at t'_i is computed as follows:

$$p'_{t'_i} = r_w * p_{t_i} \tag{4.9}$$

where $p'_{t'_i}$ and p_{t_i} are the performance q'_i at timestamp t'_i and t_i respectively.

We compute the relative weight of the performance fingerprint between the timestamps of the real workload and the trial workload. The relative weight is applied to the trial performance of the particular QoS value to compute the performance of the real workload. We perform the above steps for each $q_i \in Q_{W_i}$ to generate the long-term performance for each provider.

4.6.2 IaaS Provider Selection

We compute the distance between the estimated performance of a provider and the consumer's long-term performance requirements. The rank of each provider is computed based on their distance from the consumer's long-term requirements. We use normalized root mean squared distance to compute the distance for each QoS parameter using the following equation:

$$d(q_c, q_p) = \sqrt{\frac{1}{n} \sum_{q \in q_c, q' \in q_p, t=1..n} (q_t - q'_t)^2} \tag{4.10}$$

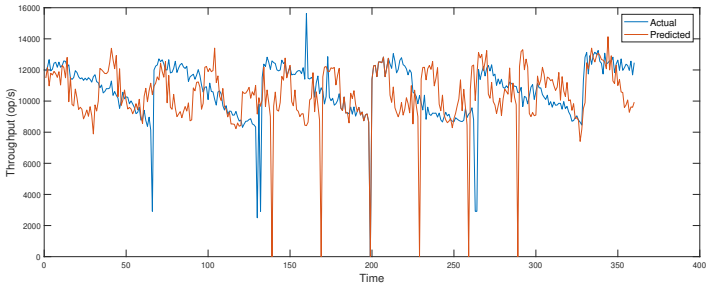
where q_c and q_p are the time series of the consumer's long-term requirements and the provider's estimated long-term performance for a particular QoS parameter, respectively. The total distance for all QoS parameter is computed by the following:

$$D(Q_c, Q_p) = \sum_{i=1}^{c} di(q_{ci}, q_{pi}) \tag{4.11}$$

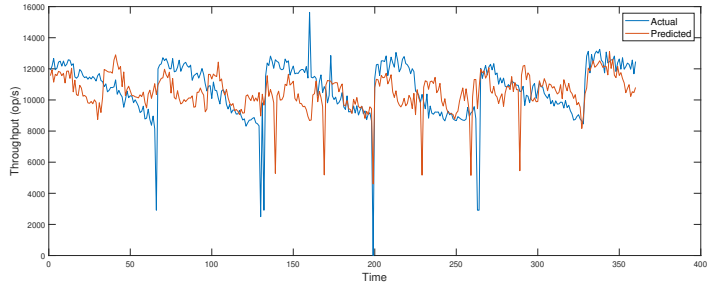
where Q_c and Q_p are the consumer's requirements and the provider's estimated performance, respectively.

4.7 Experiments and Results

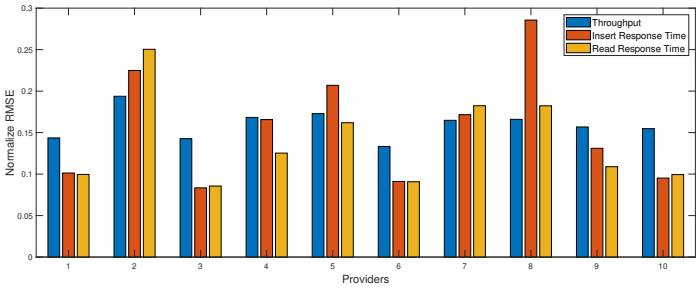
A set of experiments is conducted to evaluate the proposed approach. First, we show that the proposed trial strategy can predict a provider's long-term performance using their performance fingerprints. Next, we evaluate the effectiveness of the trial experience transformation technique considering partial fingerprint matching. Finally, we rank IaaS providers based on long-term performance prediction.



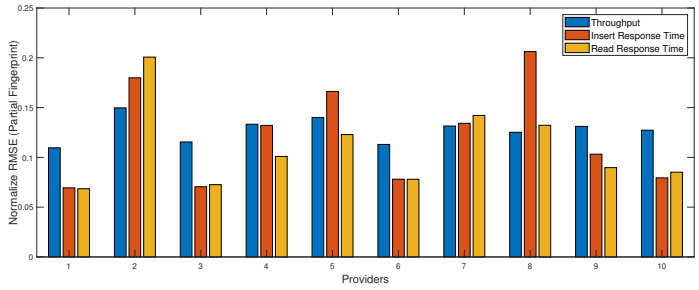
(a)



(b)



(c)



(d)

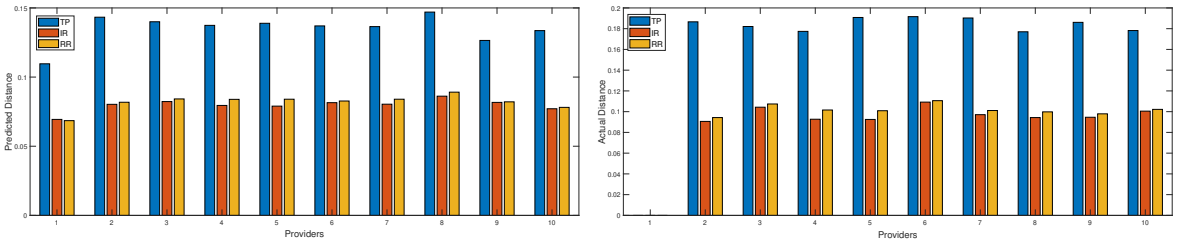
FIGURE 4.2: Long-term performance prediction (a) throughput (b) throughput with partial fingerprint (c) normalize RMSE prediction accuracy (d) normalize RMSE prediction accuracy with partial fingerprint

4.7.1 Experiment Setup

Finding real-world cloud traces for a long-term period is challenging. We generate the CPU workloads for the consumers from publicly available Eucalyptus cloud traces. The traces contain data of 6 clusters which cover continuous multi-month time frames [Nurmi *et al*., 2009]. We select one trace to generate CPU workloads for ten consumers. The QoS performance data is collected from SPEC Cloud IaaS 2016 results [Baset *et al*., 2017]. We generate QoS performances of each provider for each consumer's workloads by random replication method. Data of one month are mapped into 12-month data points where each data point is considered as an average of a single day measurement. The performance fingerprint of each provider is generated by taking the average of the observed performances of all consumers. First, we select a consumer among ten consumers as new consumer. The trial data of the selected consumer is generated using the proposed approach for 12 virtual machines and 30 days. We then find the closest matched workload from the other nine consumers to generate the performance data for the trial for each workload. The performance of the corresponding workload is considered as the trial performance of a new consumer. This approach ensures that the trial experience is affected by a provider's performance behavior.

4.7.2 Accuracy of the Performance Prediction

Figure 4.2 shows the results of a long-term performance prediction for a provider using their performance fingerprint. The performance fingerprints represent an aggregated view regardless of a consumer's workload. Hence, we can not predict the actual performance using only the provider's performance fingerprint. Figure 4.2(a) shows that the performance prediction without considering the trial experience transformation for the throughput of a provider. The performance prediction considering the partial fingerprint matching is shown in Figure 4.2(b). We use the confidence threshold $(0.5, 1)$ for the similarity and distance respectively. Once we apply the transformation, the confidence of the trial increases significantly. The prediction accuracy also improves when partial fingerprint matching is considered. Figure 4.2 depicts the long-term performance prediction for ten IaaS providers. Figure 4.2(c) shows the performance prediction without considering the partial fingerprint matching. Figure 4.2(d)



(a) (b)

FIGURE 4.3: Normalize RMSE distance between provider and consumer (a) predicted distance (b) actual distance

shows the performance prediction considering the partial fingerprint matching. The prediction accuracy is higher, i.e., lower *Normalized RMSE distance* in Figure 4.2(c) than Figure 4.2(d) which proves that the performance prediction accuracy increases with the trial experience transformation technique.

4.7.3 Accuracy of the Long-term Selection

We use the *Normalize RMSE distance* between a provider's performance and a consumer's requirements to rank each provider. The distance between the consumer's requirements and each provider's predicted performance is shown in Figure 4.3(a). The figure shows that Provider 1's results are closest to the consumer requirements for throughput, insert and read response time. Figure 4.3(a) shows the distance between the providers' actual performance and the consumer's requirement. As we generated the consumer's requirement from Provider 1's actual performance, Provider 1 has zero distance from the consumer's requirement. Therefore, the proposed approach successfully selects the optimal provider for the long-term period.

4.8 Summary

We propose a novel approach to select IaaS providers using their performance fingerprints. The proposed approach utilizes free trial periods to evaluate a provider's long-term performance. A consumer may choose a provider based on their trial experience. A novel trial strategy

using equivalence partitioning method is proposed to estimate a provider's performance for different types of workloads while considering the provider's performance variability. The trial experience is incorporated with the provider's performance fingerprint to predict long-term performance. A performance fingerprint matching technique is proposed to ascertain the confidence of the consumer's trial experience. A trial experience transformation method is proposed to improve the confidence of the consumer's trial experience. The results of experiments show that our proposed approach helps a consumer to make an informed decision when selecting an IaaS provider for the long-term period. A key limitation is that we consider only a limited number of real-world IaaS providers. The experiments are conducted using synthetic data, as finding real-world datasets that meeting the experiment requirement is challenging. However, the synthetic datasets are created by augmenting real-world datasets to capture the characteristics of real cloud environment. Therefore, we believe the result would be similar if the experiments were conducted directly using real datasets.

CHAPTER 5

Signature-based Long-term IaaS Selection

5.1 Introduction

In the previous chapter, we introduced the concept of an IaaS performance fingerprint for the long-term selection, where we assume that the fingerprint of a service is known at the time of the selection. However, generating such a fingerprint requires all consumers to publicly share their free trial experience. However, many users may not want to share their experience publicly due to privacy. Therefore, we extend the concept of performance fingerprint to propose a new concept called IaaS performance signature. The signature of a service provides an aggregated overview of a provider's relative performance behavior over time. The signature is generated in a privacy-preserving manner where free trial users do not want to share their experience publicly. The application of the IaaS signature and fingerprint remains the same in this work. In this chapter, we introduce two new concepts: a) *IaaS signature* which captures the long-term IaaS performance variability, and b) *workload significance* which addresses a consumer's future workload characteristics. We define the IaaS signature as an aggregated view of a provider's *temporal performance change* based on the collected meta-information (e.g., seasonality and trend) from past trial users. The key contributions are summarized as follows:

- An IaaS signature model that represents a provider's relative performance change over a long-term period.
- A significance-based trial scheme to discover the unknown QoS performance for the consumer's long-term workloads.

- A trial experience transformation technique to improve the confidence of the trial experience.
- A signature-based IaaS selection approach that utilizes the trial experience and IaaS signatures to discover the long-term IaaS performance.

5.2 IaaS Signatures

We adopt the concept of signature to represent a provider's long-term performance behavior for a service over a fixed period. The term "signature" is typically utilized to indicate the characteristics of an entity, work, or a piece of information that represent their identity or uniqueness. The concept of signature is used for different purposes in several domains such as computing, cryptography, and security. For instance, application performance signatures are used for resource capacity planning and performance anomaly detection [Mi *et al.*, 2008]. Signature-based malware detection is performed by antivirus software. A checksum is also a form of signature that is utilized to verify data integrity.

An IaaS signature is a relative representation of providers' performance over a fixed period for a particular service. The signature indicates a provider's performance trends and seasonality, i.e., how much a provider's performance may increase or decrease in one time compared to another time. For instance, the signature of an IaaS provider may inform that the provider's performance increase by 10% on weekend nights than regular weekdays. IaaS signature does not provide the consumer with the actual performance of a provider. A consumer would find it challenging to use the signature without performing the trial using its workloads.

IaaS Signature: An IaaS signature is a temporal representation of a provider's relative performance change over time for a service. The signature is defined by a set of QoS parameters that are relevant to the service.

We utilize signatures to measure the confidence of trial experience and to discover a provider's long-term performance. First, the trial confidence is determined by a similarity distance between the trial experience and IaaS signatures. The trial experience may be utilized to

discover the long-term service performance when the experience has high confidence. If the trial experience has low confidence, we discard the provider based on a predefined threshold. Next, we utilize the signature to estimate the provider's long-term service performance for the consumer's long-term workloads. The optimal provider is selected based on a time series similarity distance between the consumer's expected service performance and providers' predicted service performance.

5.2.1 IaaS Signature Representation

The relevant QoS attributes are the key QoS attributes to measure the performance of the service [Li *et al.*, 2010]. For example, data read/write throughput, and disk latency are the most important QoS attributes for virtual storage services. We denote the IaaS signature of a provider as $S = \{S_1, S_2, ...S_n\}$, where n is the number of QoS attributes in the signature. Each S_n corresponds to a QoS attribute Q_n. S_n denotes a time series for t period where $S_n = \{s_{n1}, s_{n2}, ...s_{nt}\}$. Here, s_nt is the relative performance of the provider at the time t for a particular QoS attribute. We use the following representation of IaaS signature:

$$S = \begin{bmatrix} s_{11} & s_{12} & .. & s_{1t} \\ s_{21} & s_{22} & .. & s_{2t} \\ s_{31} & s_{13} & .. & s_{3t} \\ .. & .. & ... & \\ s_{n1} & s_{n2} & .. & s_{nt} \end{bmatrix} \tag{5.1}$$

where each row corresponds to the QoS signature of Q_i and each column represents a timestamp t.

5.2.2 IaaS Signature Generation

We aim to represent a provider's long-term service performance changes using its signature. A provider's service performance may vary based on several factors such as the degree

of resource overbooking, the number of co-tenants and poor network conditions due to the external factors [Wang *et al*., 2018]. It is difficult to determine what are the factors behind the performance variability over time from the consumer side. However, the changes in performance often exhibit weekly, monthly, or yearly seasonality [Iosup *et al*., 2011]. Therefore, it may be possible to capture the seasonal performance changes from the experience of past trial users over different times [Wang *et al*., 2018]. Note that past trial users may not share their experience publicly due to privacy, security, and the conflict of interests with the provider [Ba-Hutair and Kamel, 2016].

It is reasonable to assume that past trial users may share their experience with a trusted non-profit organization (TNPO) for a limited period to help new consumers as they make a selection. Examples of such TNPOs are available in the public sector where privacy sensitive information about individuals needs to be shared to deliver better services [van den Braak *et al*., 2012]. For instance, health research institutes often collect data about individual patients to improve health services [Polemi, 1998]. TNPOs are responsible for data integration and distribution of collective knowledge without revealing individuals' privacy- sensitive information. In the context of cloud environment, there are few third party organizations who collect and report cloud performance based on the experience of free trial users. Geekbench[1] is a prime example of such third party organizations where trial users share their experience. In Geekbench platform, users run Geekbench provided benchmark to perform free trials on different cloud services. The experience of free trial can be accessed through Geekbench website.

We assume that past trial users share their experience with a TNPO. The TNPO generates IaaS signatures based on the aggregated experience of past trial users. The individual's experience is deleted after the generation of the IaaS signatures.

IaaS signatures can be generated in different ways based on the purpose of the signatures, which is two-fold. First, we want to ascertain the confidence of the trial experience using the signature. Second, we want to utilize the signature to predict a provider's future performance behavior using the trial experience. We represent the signature in a way that requires less

[1]https://browser.geekbench.com/

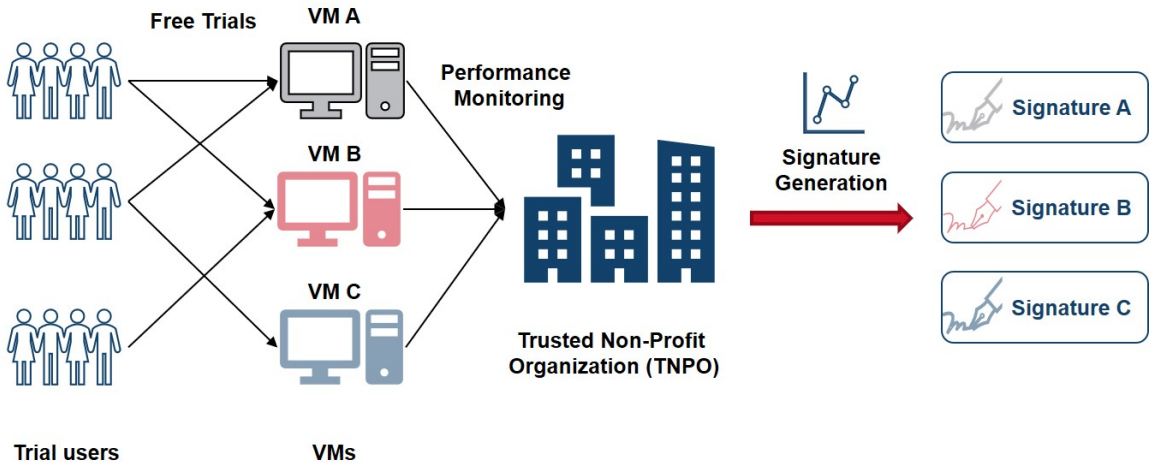


FIGURE 5.1: IaaS signature generation

detailed performance information about the provider and the past trial users. We apply a normalized averaging method to generate the signature based on the experience of past trial users.

Let us assume that three IaaS providers (A, B, and C) offer three VMs (VM_a, VM_b, and VM_c) with similar configurations (e.g., capacity, location) for free short-term trials. There exist past users who utilized the trials to find the performance of each VM at different times. Past trial users do not want to share their trial experience publicly. However, each trial user shares their experience with a Trusted Non-Profit Organization (TNPO) for a short period (Figure 5.1). The TNPO generates IaaS signatures to identify providers' long-term performance variability for each VM. The TNPO deletes users' experience once the signatures are computed. A signature provides an aggregated view of a provider's long-term performance variability. It is not possible to derive individual trial experience from the signature, and so the TNPO does not violate the privacy of past trial users. The IaaS signatures do not contain the provider's actual performance information.

Let us assume that k number of past trial users share their observed trial performance P_k over the period T for a service. We denote P_k as a set of QoS time series where $P_k = \{Q_{1k}, Q_{2k}, ..., Q_{nk}\}$. Here, Q_{ik} refers to the performance observed by the kth consumer for

the QoS attribute Q_i over the period T. We denote Q_{ik} as $Q_{ik} = \{q_{1ik}, q_{2ik}, .., q_{tik}\}$. We perform the following steps to generate the IaaS signature:

(1) For each QoS attribute Q_i, the performance observed by the trial users is collected over T period.
(2) At each timestamp $t \in T$, the average performance observed by k number of consumers is measured for each QoS attribute Q_i. The average performance over T period is denoted by $\overline{Q_{ik}}$
(3) Each $\overline{Q_{ik}}$ is normalized based on its standard deviation $\sigma(\overline{Q_{ik}})$. The normalized QoS time series groups form the IaaS signature S over T period.

The value of $s_n t$ at any t represents the relative QoS performance compared to any other time t' in Equation 5.1. This simple representation of the signature offers two benefits. First, the use of signature becomes easier once a consumer utilizes free trials based on their workloads. The performance for any other time can be found by comparing the ratio between the trial month and other time. Second, signatures can be stored and updated easily over time as storing detailed information is not necessary.

We assume that the signature provided by the TNPO is accurate and complete for period T. We assume that a provider's signatures does not drastically change over this T period. The signature mainly reflects substantial changes in the provider's service performance. The effect of the signature should be visible to most consumers in the trial period unless the provider utilizes an isolated environment.

5.3 Significance-based Trial Scheme

We devise a significance-based trial scheme to generate trial workloads based on the consumer's expected long-term workloads. The performance of a provider may depend on their workloads, e.g., service requests or tasks being processed [Calzarossa *et al.*, 2000]. Therefore, a consumer needs to evaluate the service performance of a provider based on their long-term workloads. *We assume that the long-term workloads are deterministic, i.e., the expected*

workloads are known at the time of the selection. Long-term workloads may be estimated based on the real-world workload traces. Such traces can be found in previous activity logs of the consumer. For example, large organizations often keep logs of their users' activities [Feitelson, 2002]. However, it may not be possible to run the consumer's entire long-term workloads in a short trial due to temporal restrictions. In such a case, representative trial workloads need to be generated based on the characteristics of the long-term workloads [Iosup *et al.*, 2014].

The first step to generate representative workloads is to determine the workload components (e.g., users, sessions, and applications) and workload parameters. The workload parameters are typically defined by the characteristics of the service requests such as requests arrival times, type of requests, or resource demands of different types of applications [Feitelson, 2002].

We select resource demands per second as the workload parameter without the loss of generality. We denote a consumer's expected workload time series as $w = \{w_1, w_2, ..., w_T\}$ over period of time T. Here, w_T represents the resource demand at time T. Note that it is possible to model other workload parameters as w depends on the service requirements. If multiple workload parameters need to be modeled, we may consider that w has multiple dimensions, where each dimension represents a specific workload parameter.

The next step is to characterize workloads based on the workload parameters. Workload characterization is usually performed based on statistical analysis such as clustering, specifying dispersion, PCA, and frequency distribution analysis. We use the frequency distribution analysis to characterize the consumer's long-term workloads. Finally, a subset of the long-term workloads is selected as the representative workloads for the trial. The selection criteria are defined based on the characteristics of the long-term workloads.

A consumer needs to define the selection criteria for the trial workloads carefully. Otherwise, the trial experience give useful information for the selection process. The trial needs to be performed with the workloads that have the most *significance* to the consumer. We define two

types of workload significance based on two workload parameters: a) occurrences, and b) resource consumption. The definitions of the two types of workload significance are:

- **Frequency-based Significance**: The type of workload that is expected to appear more frequently in the future than any other type of workload considered significant to the consumer.
- **Resource Consumption-based Significance**: The type of workload that is expected to demand more resources in the future than any other type of workload considered significant to the consumer.

Workload significance can be defined in terms of other criteria based on various workload parameters. For instance, a consumer may define short-term and long-term requests based on the expected execution time of the requests. We only focus on frequency-based and resource consumption-based trial workload generation. Let us assume that the trial period t has k number of timestamps and the consumer's long-term workload has n number of timestamps. We assume that $k << n$, i.e., the value of k is significantly less than the value of n. We need to generate k workloads from n workloads to perform the trial. Algorithm 3 illustrates the proposed scheme for the trial workload generation.

Algorithm 3 Significance-based trial scheme

1: **Input:** W, t, S
2: **Output:** W_t
3: $n \leftarrow size(W)$;
4: $k \leftarrow length(t)$;
5: $Uw \leftarrow unique(W)$;
6: $W_{info} = createArray(size(w))$
7: **for** each workload w in Uw **do**
8: $\quad W_{info}(w).frequency = count(w)$
9: $\quad W_{info}(w).level = level(w))$
10: $W_t = select(W_{info}, k, S)$
11: return W_t

Algorithm 3 takes the long-term workloads W, the trial period t, and the significance S as input (line 1). The output of the algorithm is W_t, which is a subset of W (line 2). First, the algorithm computes the size of W and the length of t (line 3-4). Next, it finds the unique workloads Uw in W (line 5). An array is then created called W_{info} that stores the *level* of

each workload and its *frequency* (line 6). The level of a workload is defined by its resource consumption. For example, if a workload requires 90% of the CPU units, the level is set to *high* for the workload. The level function is predefined based on the resource capacity. The frequency of each workload is stored based on its number of occurrences in W using the $count$ function (line 8). Once the map is created for each workload, a workload *selection* function is used to select k workloads from n workloads using S (line 10). The value of S determines the significance of the workloads. We use the following three criteria for S to generate the trial workload:

(1) *Frequency-based Generation (FG):* We select k trial workloads that occur most frequently in W.
(2) *Resource Consumption-based Generation (RG):* We select k trial workloads from W that have maximum resource consumption. W_{info} reaches equal to k.
(3) *Mixed Generation (MG):* We select $k/2$ workloads based on FG method and $k/2$ workloads using RG method.

The selection function can be implemented in different ways based on the workload parameters and the significance. We leave it for the future work to define workload significance using other techniques.

5.4 Signature-based IaaS Selection

5.4.1 Trial Confidence Measure

The trial confidence is determined using the similarity distance between the IaaS signature and the trial experience. The QoS performance observed in the trial should be normalized before measuring the similarity distance. We measure the similarity distance based on the shape of the signature and the trial experience for each QoS attribute. We decided to use the Pearson Correlation Coefficient (PCC) to measure the trial confidence (T_{conf}). The PCC is applied to measure the trial confidence ($T_{conf}^{Q_i}$) for each QoS attribute Q_i as follows:

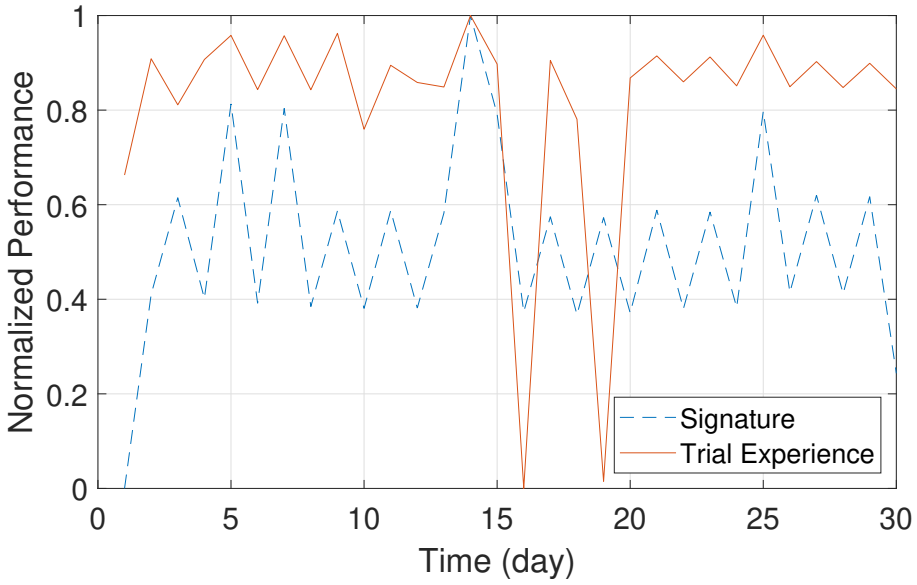


FIGURE 5.2: Trial experience matching (CPU throughput)

$$T_{conf}^{Q_i} = \frac{\sum_{t=1}^{n}(q'_t - \bar{q'})(q_t - \bar{q})}{\sqrt{\sum_{t=1}^{n}(q'_t - \bar{q'})^2}\sqrt{\sum_{t=1}^{n}(q_t - \bar{q})^2}} \tag{5.2}$$

where n is the trial length, q'_t is the normalized value of the trial performance of Q_i at time t. The normalized value of the signature is q_t at time t for the QoS attribute Q_i. The mean value of Q_i is indicated by $\bar{q'}$. The total confidence is calculated by taking the average of all confidence for each QoS attribute by the following equation:

$$T_{conf} = \frac{1}{k}\sum_{i=1}^{k} T_{conf}^{Q_i} \tag{5.3}$$

where k is the total number of QoS attributes. Figure 5.2 depicts a plot of normalized trial performance and signature for CPU throughput. The figure shows that the shape of the trial experience is similar to the signature. If the confidence is lower than a predefined threshold (e.g., less than 70%), the provider is discarded.

5.4.2 Signature-based Performance Discovery (SPD)

We utilize the IaaS signature to measure the provider's service performance beyond the trial period. The first step is to estimate the service performance for the consumer's long-term workloads based on the trial experience. We then need to apply the signature to adjust the performance of each type of workload based on the time of its appearance. For example, if a certain type of workload appears in January, then the performance of that type of workload needs to be changed using the signature. We apply Algorithm 4 to discover a provider's service performance beyond the trial period for the consumer's long-term workloads.

Algorithm 4 Signature-based performance discovery

1: **Input:** W, W_{trial}, P_{trial}, S
2: **Output:** P
3: $TotalTime = length(W)$
4: $trialLength = length(W_{trial})$
5: $S_{trial} = S(1 : trialLength)$
6: **for** each t in $TotalTime$ **do**
7: $\quad t' = NearestNeighbor(W(t), W_{trial})$
8: $\quad TransForm = S(t)/S_{trial}(t')$
9: $\quad P(t) = TransForm * P_{trial}(t')$
10: return P

Algorithm 4 takes as input the trial workload W_{trial}, trial performance P, long-term workloads W, and IaaS signature S (line 1). The algorithm returns the long-term performance P (line 2). First, the algorithm measures the length of W to estimate the total required service time (line 3). Then, the trial length is measured based on W_{trial} (line 4). The part of signature that is applicable for the trial period is taken from S based on the trial length $trialLength$ (line 5). Next, for each timestamp in the total time, the algorithm needs to measure the performance of the corresponding workload (line 6). For each workload at time t, the $NearestNeighbor$ function finds the closest workload that can be found in the trial workloads based on resource demand (line 7). We use the Euclidean distance to measure the similarity between workloads. The function $NearestNeighbor$ returns the timestamp t' that is the timestamp of the closest workload. Next, the transformation factor $TransForm$ is measured by taking the ratio between the signature of the current timestamp t and t' (line 8). The performance at the current timestamp $P(t)$ is found by multiplying the $P_{trial}(t')$ with the transformation factor

$TransForm$ (line 9). Here, $P_{trial}(t')$ is the performance of the trial workload that is closest to the current workload $W(t)$.

The performances P_{trial} and P are shown as a one-dimensional time series in the algorithm. However, the algorithm is still applicable if the performance is considered multi-dimensional, i.e., multiple QoS attributes.

5.4.3 Long-term IaaS Selection

The long-term selection is performed based on the consumer's requested performance and the predicted service performance [Ye *et al.*, 2016]. First, we normalize the value of QoS attributes based on *Min-Max Feature Scaling* to have the same scale for each QoS attribute using the following equation:

$$Q'_{it} = \frac{Q_i(t) - min(Q_i)}{max(Q_i) - min(Q_i)} \tag{5.4}$$

where $Q_i(t)$ is the value of Q_i at time t. Q'_{it} is the normalized value of $Q_i(t)$. Equation 5.4 is applied to each QoS time series of the requested and predicted performance. Next, we measure the Root Mean Squared Error (RMSE) distance from the consumer's requested performance and the discovered IaaS performance for each QoS attribute using the following equation:

$$RMSE(Q_i^r, Q_i^p) = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(Q_i^r(t) - Q_i^p(t))^2} \tag{5.5}$$

where Q_i^r and Q_i^p are the requested and predicted QoS performance of Q_i over time T. $Q_i^r(t)$ and $Q_i^p(t)$ denote the requested QoS performance and predicted QoS performance respectively at time t. Finally, the rank of each provider is measured by the following equation:

$$Rank(P) = \sum_{i=1}^{k} RMSE(Q_i^r, Q_i^p) \tag{5.6}$$

where $Rank(P)$ is the predicted rank of the provider P and k is the total number of QoS attributes.

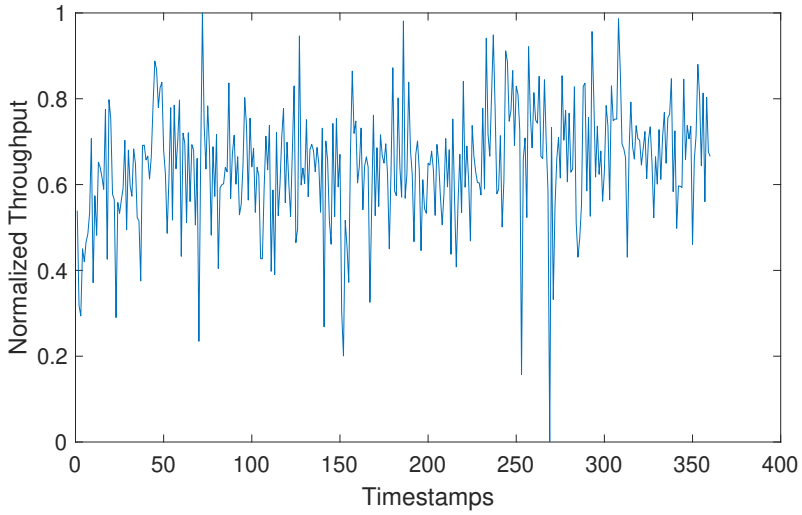
5.5 Experiments and Results

5.5.1 Experiment Setup

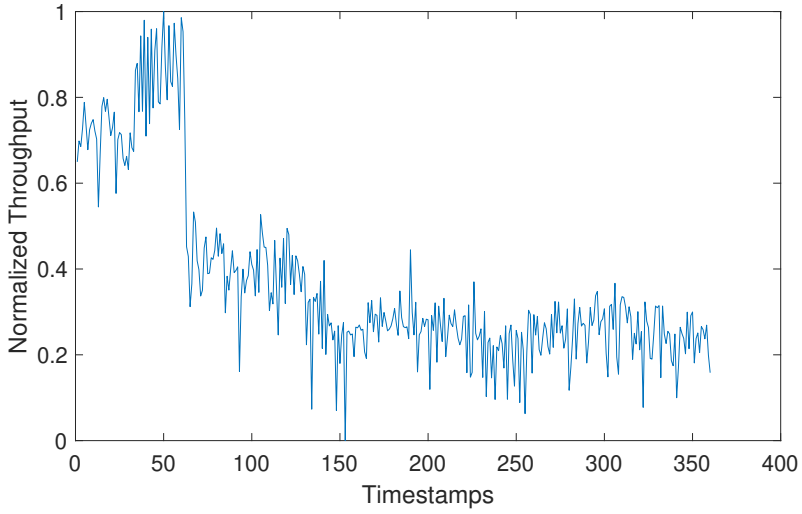
We conducted a set of experiments based on real-world datasets. The proposed SPD approach was compared against the baseline approach, i.e., LPD approach and EQ approach [Fattah *et al*., 2019]. The SPD-based long-term selection was evaluated based on the expected ranking, short-term ranking [Ye *et al*., 2016], and LPD-based ranking approaches.

5.5.1.1 Dataset from Public IaaS Providers

We collected data from Microsoft Azure and Google Compute Engine (GCP) every 15 minutes for about one month. We selected Standard A1 v2 and n1-standard-1 types of instances from Azure and GCP, respectively. Three instances for each type of VM were installed with similar configurations. Each instance ran a web server that generated a CPU-intensive load (Fibonacci number generator) for each request. Requests were generated using httperf load generator. Each request had a very small payload and the clients were installed in the same cloud. Hence, we assumed that the network performance fluctuation had little or no effect on the observed performance of each provider. The one-month data was divided into 12 partitions. Each partition was considered a one-month data. The signature of each provider was generated using the proposed approach in Section 5.2 based on the data collected from three instances of each provider as shown in figure 5.3. The trial workload was produced based on the workloads generated by httperf using the proposed approaches in Section 5.3. The trial performance of each type of workload was created based on the average performance observed in the trial period for the corresponding type of workload.



(a)



(b)

FIGURE 5.3: Signature calculated from public cloud providers (a) Azure instance (b) GCP instance

5.5.1.2 Dataset from Private IaaS Providers

We utilized the publicly available Eucalyptus IaaS workload[2], which contains about 34 days of workload data, to generate long-term consumer workloads. We generated 360 days of workload data for each consumer-based average workload per day. The long-term performance of five private IaaS providers was generated from benchmark results published by SPEC Cloud

[2] https://sites.cs.ucsb.edu/~rich/workload/

TABLE 5.1: Experimental settings

Attribute	Value
Total Time	360 days
Number of Providers	7
Trial Period Length	30 days
Number of Trial Methods	4
Trial Month	June

IaaS 2016[3]. First, we mapped each unique workload of the cluster to a unique performance value of the benchmark results. We considered the map to be the baseline performance for the workload. Next, we built long-term performance profiles for the providers where each provider showed different performance behavior based on the workloads and time. We ran the workloads from each consumer on five providers (candidates for the long-term period selection) using a short-term trial period to discover the corresponding performances of each provider. The experimental settings are shown in Table 5.1.

5.5.1.3 Baseline Approach

We defined a Long-term Performance Discovery (LPD) approach as the baseline approach to evaluate the proposed SPD approach. The trial experience contained a subset of long-term workloads and corresponding performance. We generated the performance of the long-term workloads for each provider based on the consumer trial experience. For each workload w_l in the long-term workloads, we found a workload w_t in the trial workload, where w_l and w_t have similar resource consumption. We considered the performance of w_l is equivalent to w_t.

5.5.1.4 Equivalence Partitioning-based Approach

An equivalence partitioning-based (EQ) approach is proposed in Chapter 3 where the consumer's long-term workload is partitioned based on the number of available VMs in the free trial period. Then, workloads of each partition are compressed within one day, assuming that the performance of the provider does not change considerably within a day. Each VM runs the same workload for the trial period to find the performance variability.

[3] https://www.spec.org/

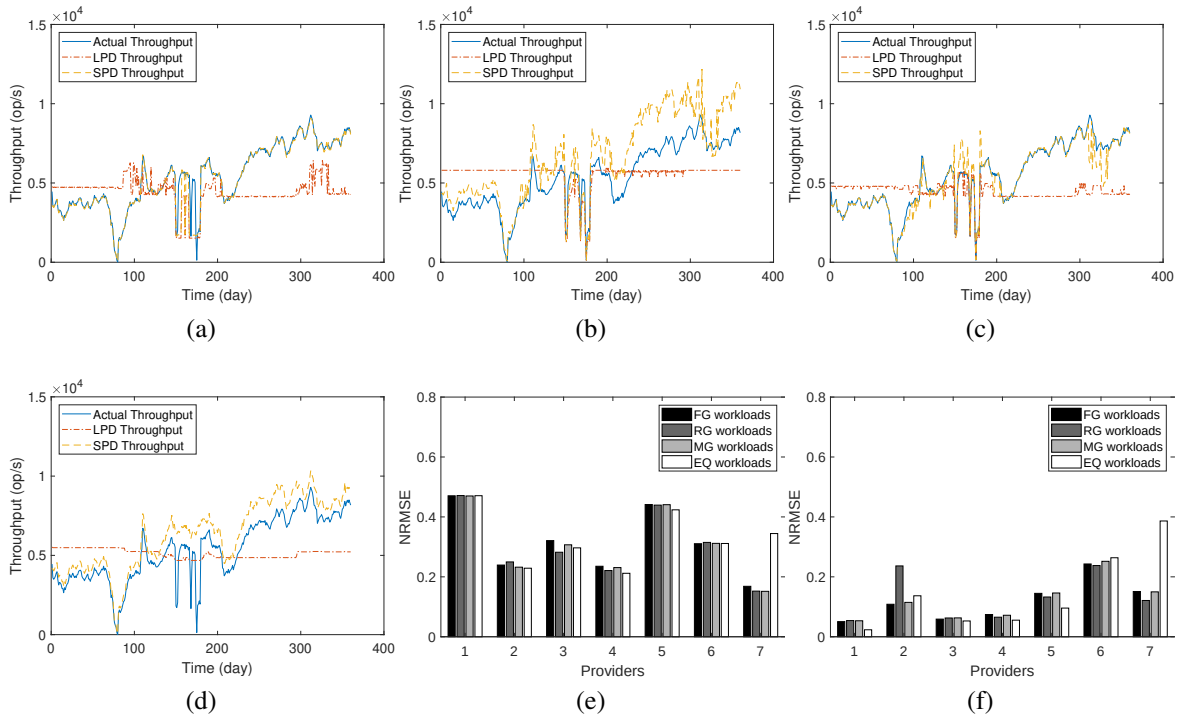


FIGURE 5.4: Long-term throughput prediction (a) FG workloads (b) RG workloads (c) MG workloads (d) EQ workloads (e) LPD approach error (f) SPD approach error

5.5.2 Evaluation of Long-term Performance Discovery

Figure 5.4 shows the results of the long-term IaaS performance discovery. Figure 5.4(a), (b), (c), and (d) show the predicted CPU throughput of a provider using the FG, RG, MG, and EQ trial schemes defined in section 5.3. Each figure shows the LPD throughput, SPD throughput, and actual throughput. The predicted performance using the LPD approach exhibits similar behavior in each figure. The LPD predicted performance cannot capture the temporal performance shifts. It is noticeable that the predicted performance remains on the same performance level of the trial month (151-180 days). The LPD approach can be useful to predict the performance of the providers that provide services with good performance isolation. The SPD approach predicts the throughput more accurately compared to the LPD approach as shown in each figure (Figures 5.4(a), (b), (c), and (d)). The SPD approach utilizes the shape of the signature to estimate long-term IaaS performance. Hence, the predicted performance has a similar shape to the signature. Figure 5.4(e) and (f) show the accuracy of the predicted

performance using normalized RMSE (NRMSE) distances for seven providers. Providers 6 and 7 are public IaaS provider and the rest are private IaaS providers. The accuracy of the SPD approach (Figure 5.4(f)) is considerably higher than the LPD approach (Figure 5.4(e)).

5.5.3 Effect of Trial Schemes in Performance Discovery

The effects of different trial schemes are noticeable in Figures 5.4(e) and (f). The LPD approach exhibits less performance variability for different trial schemes as it does not consider the provider's long-term performance variability. The prediction accuracy of the SPD approach varies considerably based on the selected trial scheme. The RG scheme-based SPD approach (Figure 5.4(f)) shows the lowest accuracy compared to the other approaches (Figure 5.4(f)). This result is due to the characteristics of the consumer's long-term workloads. The RG scheme selects mainly resource-intensive (i.e., requires high resource usage) workloads similar to the traditional *load* and *stress* testing based approaches. Hence, traditional load and stress testing techniques may not provide good accuracy for long-term performance discovery.

The FG scheme-based SPD approach shows the maximum estimation accuracy compared to the other trial schemes (Figure 5.4 (f)). The reason for this is that this approach utilizes most frequently occurring workloads in the consumer's long-term workloads. The maximum number of workloads is tested in this scheme. The estimation errors for the MG and EQ scheme remain in between the FG and RG schemes. The MG scheme is built using the FG and RG scheme. As a result, the NRMSE for the MG scheme is in between FG and RG schemes. The EQ scheme shows poor performance for public providers as the EQ scheme depends on the number of available VMs to run the experiments.

5.5.4 Evaluation of IaaS Ranking

The ranking of the providers based on different approaches is shown in Table 5.2. We measure the *expected ranking* of the providers to evaluate the proposed selection approach. The expected ranking is computed based on the NRMSE distance between the consumer's throughput requirement and a provider's actual throughput. We rank the providers based on

TABLE 5.2: Ranking of IaaS providers

Rankings	Orders
Expected	p1 $<$ p4 $<$ p2 $<$ p3 $<$ p5 $<$ p6 $<$ p7
Short-term	p1 $<$ p6 $<$ p7 $<$ p3 $<$ p4 $<$ p2 $<$ p5
LPD	p2 $<$ p4 $<$ p5 $<$ p3 $<$ p1 $<$ p6 $<$ p7
SPD	p1 $<$ p4 $<$ p3 $<$ p2 $<$ p5 $<$ p6 $<$ p7

three approaches using the FG scheme. First, we rank the providers based on the short-term trial experience. The short-term ranking cannot rank the providers correctly compared to the expected ranking. Therefore, the short-term selection approach is not applicable for the long-term period. Next, we rank the providers based on the predicted performance using the LPD approach, which does not rank most providers correctly. Hence, the selection based on the trial experience without considering the long-term performance may lead to incorrect provider selection. Finally, we rank the providers based on the predicted performance using the SPD approach that ranks most providers correctly. We use the following utility function to measure the *weighted ranking error* (E) of each ranking approach:

$$E = \frac{1}{W} \sum_{i=1}^{n} W_p \times (R_p - R_a)^2 \tag{5.7}$$

where n is the number of providers, R_p is the predicted rank, and R_a is the actual rank. The lower value of E means higher ranking accuracy, and higher value indicates lower ranking accuracy. W_p denotes the weight of each provider. The weight is assigned using the provider's expected ranking. If the expected ranking is high, then the weight is considered high. Table 5.2 shows the accuracy of the rankings on the error column. The SPD-based estimation shows the lowest ranking error, which indicates the highest level of accuracy.

5.6 Summary

We introduce a novel approach to select the optimal IaaS service according to a consumer's long-term QoS requirements. The proposed approach leverages free trials and IaaS signatures to discover long-term service performance of IaaS providers. The experiment results using

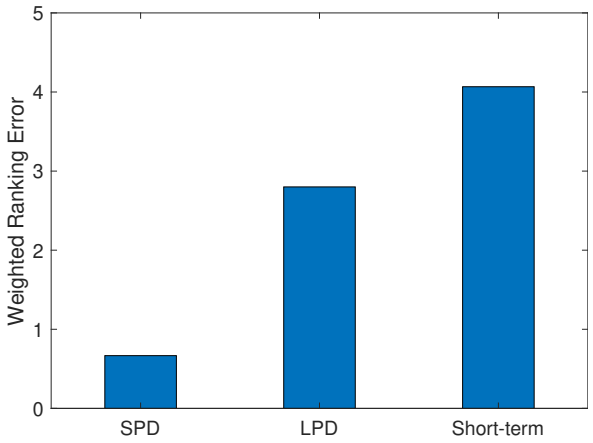


FIGURE 5.5: Weighted ranking error

the real-world datasets show that the proposed SPD approach effectively discovers long-term service performance using different trial schemes. We conclude that the selection of an appropriate trial scheme plays an important role in the long-term performance discovery. The results also confirm that the proposed approach ranks the IaaS services effectively using the IaaS signatures and the consumer's trial experience. In this chapter, we focus on deterministic workloads; however, in the future, we will explore long-term IaaS selection for stochastic workloads as well.

CHAPTER 6

IaaS Signature Change Detection

6.1 Introduction

An understanding of the IaaS services' performance is paramount in determining which services are the best fit for the consumers' required QoS [Iosup *et al.*, 2011]. Despite that, IaaS providers typically reveal very limited performance information in their advertisements due to market competition and business secrecy [Wenmin *et al.*, 2011]. Effective utilization of free trials offered by IaaS providers is a unique way to deal with the limited performance information for the long-term selection [Wang *et al.*, 2018]. However, free trial experiences do not provide enough information to make the best service selection for a long-term period. The key reason is that the performance of IaaS services changes periodically due to the multi-tenant nature of the cloud [Iosup *et al.*, 2011]. The observed performance in a trial in one month may change if the trial is performed in a different month. Therefore, making a long-term commitment based only on short trials does not always lead to the best service selection [Zhu and Chang, 2014].

IaaS performance signatures offer an effective alternative to deal with the unknown service performance variability for the long-term selection [Mi *et al.*, 2008; Fattah *et al.*, 2020b]. An IaaS performance signature represents the *expected* performance behavior of an IaaS service over a long period of time. For instance, the signature of a VM may indicate that its response time is expected to increase by 10% in January over the response time for December. A consumer's trial experience of a service and its corresponding signature can be utilized to make a better selection for the long-term period. A signature-based IaaS selection approach is proposed in the previous chapter. The proposed approach generates signatures using the

experience of past trial users. However, the proposed approach does not consider the *dynamic* nature of performance signatures.

IaaS performance signatures are dynamic in nature and may need to be re-evaluated over a long period of time for a number of reasons [Mi *et al.*, 2008]. For instance, a provider may upgrade their infrastructure or change their QoS management policy, resulting in a change in service performance [Leitner and Cito, 2016]. In such a case, the IaaS performance signature may need to be updated to be representative of the new performance profile of the service. It is therefore important to detect the change in IaaS performance *as early as* possible to make sure its signature reflects the *current* performance behavior of the service. *We focus on the detection of long-term changes in IaaS performance behavior.* We assume that the long-term performance behavior of an IaaS service is represented by its performance signature. Therefore, the IaaS performance change detection problem is transformed as the IaaS signature change detection problem in this work.

We identify two key challenges in IaaS performance change detection in the context of IaaS signatures. The first challenge is in determining the time at which the signature needs to be re-evaluated for the change detection. A change in the service performance behavior may occur at any point of time. The challenge is to accurately identify change points in time where there is a high likelihood of performance change occurrences. This is also known as the *Change Point Detection* problem [Aminikhanghahi and Cook, 2017]. The second challenge is the ability to differentiate between *noise* and *actual changes* in IaaS performance. Noise refers to the irregular or anomalous performance behavior that may not necessarily indicate long-term performance changes [Moens and Zénon, 2019]. For instance, a major failure of computing infrastructure may negatively impact the performance of an IaaS service at a point of time without necessarily indicating a long term change in performance behavior. Given the multi-tenant and dynamic nature of the cloud, noise in IaaS performance is very common [Doelitzscher *et al.*, 2013]. Therefore, the challenge is to accurately identify performance noise to detect actual performance changes.

To the best of our knowledge, existing research has not given enough attention to the long-term IaaS performance change detection problem [Fattah *et al.*, 2019]. We focus on the

signature change detection in this chapter. First, we assume that there is no noise in the IaaS performance. In this case, we propose a novel ECA-based approach to detect changes in IaaS performance.We then introduce a new performance noise-based change detection approach that considers the performance noise during the change detection. Finally, we introduce a change detection framework that leverages change detection techniques from signal processing domain to detect changes in IaaS signatures.

6.2 ECA-based Signature Change Detection

We propose a novel *Event-Condition-Action* (ECA) approach to manage changes in IaaS performance signatures. The ECA model is a simple yet powerful tool that has been extensively used in databases, cognitive computing, and semantic web. In the ECA model, when an *event* is detected, a *condition* is checked, and a resulting *action* is executed [Liu and Özsu, 2009]. Although the ECA model is quite standard, the way we have used it and the context in which it has been used is novel. The ECA model is typically used in databases to trigger an action when a condition is met. We have leveraged the ECA model to detect change points in IaaS performance behavior. The proposed ECA approach relies on the detection of anomalous performance behavior to detect changes in IaaS performance. In particular, the proposed approach consists of two main parts: a) an *anomaly-based event detection* technique that determines when to trigger the re-evaluation of a signature, and b) *a signature change detection* method that leverages time series change detection techniques to re-evaluate existing IaaS performance signatures. In addition, we introduce a self-adjustment method to improve the performance of the proposed ECA approach using a feedback loop from the outcome of the signature change detection. In summary, we propose a novel framework for the detection of accurate changes in IaaS performance signatures. Accuracy is achieved over time through continuous testing of the re-evaluated signatures, which may lead to either (1) confirming the previous signature changes, or (2) invalidating the previous signature changes. The key contributions of this section is as follows:

- An anomaly-based event detection to determine when to trigger the testing of an IaaS performance signature.
- A signature change detection that leverages time series change detection techniques to re-evaluate the existing IaaS performance signatures.
- A self-adjustment method to improve the accuracy of the proposed ECA approach over time using a feedback loop.

6.2.1 Proposed ECA Approach

We utilize the signature representation and the generation from the last chapter. Therefore, we skip the details about the IaaS signatures in this chapter. We apply an ECA approach to manage changes in IaaS signatures. The ECA model is especially useful when an action needs to be performed based on a condition that needs to be satisfied. According to the ECA model, an event determines when to trigger an action, the condition defines how to evaluate the event, and the action sets the execution plan in response to the event. An event is typically a special indicator that informs a system that an action may need to be performed. An example of events in security software could be defined as the deletion of a large number of files at once. The security software may start the evaluation of the event, i.e., the deletion of a large number of files to find out whether it is a result of a security attack or a user action.

Anomalous performance behavior is a potential indicator of IaaS performance change [Mi *et al.*, 2008]. An anomalous performance behavior is the *deviation* from the expected performance of an IaaS service. The expected performance is represented by its performance signature. Performance anomalies are typically common in cloud environment [Ibidunmoye *et al.*, 2015]. Anomalies may occur due to unexpected events faced by the IaaS provider such as a sudden increase in the workload of the physical system, a power failure, or natural disasters. As a result, experiencing performance anomalies in the free trial period may be normal in the cloud. However, the frequent occurrences of performance anomalies in the free trial period may indicate changes in IaaS performance, thus requiring a re-evaluation of the existing signature [Mi *et al.*, 2008]. Therefore, we define the event for the IaaS performance change detection based on the frequent occurrences of performance anomalies.

DEFINITION 8. *Event.* An event is the frequent occurrences of performance anomalies that are experienced by the free trial users within a fixed period of time.

The frequency is initially defined as an arbitrary number or threshold f which can be adjusted in the self-adjustment step. Once an event is detected, it needs to be evaluated to detect whether the signature has been changed. If the event satisfies the condition, the signature needs to be updated. The condition and action are defined as follows:

DEFINITION 9. *Condition.* The condition is the process of testing an event to ascertain changes in IaaS performance.

DEFINITION 10. *Action.* The action is the process of updating the present IaaS performance signature to reflect the changes in the IaaS performance.

We utilize the above three definitions as the basis for the proposed ECA approach. In the following sections, we describe the three parts of the proposed approach: a) an anomaly-based event detection, b) a signature change detection and signature update (condition and action respectively), and c) a self-adjustment method to improve the accuracy of the proposed approach.

6.2.2 Anomaly-based Event Detection

We utilize the free trial experience and the existing IaaS performance signatures to detect performance anomalies. The events are detected based on the performance anomalies. First, we measure the similarity between the trial experience of a consumer and the signatures to detect performance anomalies [Ibidunmoye *et al.*, 2015]. When a user's trial experience is similar to the current signature, the signature is considered to be representative of the expected service performance. When the trial experience does not exhibit a similar performance behavior as represented by its signature, we consider it to be an anomalous performance behavior. The signature represents the relative performance behavior as a time series. As a result, the shape of the time series needs to be considered in order to measure the similarity rather than the value of each data point in the signature time series. There are numerous

approaches in the existing literature to measure time series similarity based on the shape such as Pearson Correlation Coefficients (PCC), Euclidean Distance (ED), Spearman Correlation (SC), Cosine Similarity (CS), Symbolic Aggregate Approximation (SAX), and Dynamic Time Warping (DTW). Some of these methods may be applied to measure the similarities between the trial experience and the signatures. We briefly discuss how to apply the PCC, CS, and ED for the similarity measure to detect performance anomalies in the context of IaaS performance signatures as they are most widely used techniques and easier to implement than other techniques.

Let us denote the trial experience of a user by E_Q where E_Q denotes the performance of an IaaS service in the free trial period T_f. Here, $T_f << T$, i.e., the free trial period is significantly less than the required provisioning period T. We represent E_Q as a time series $E_Q = \{q_1, q_2, ...q_n\}$ where n is the number of timestamps in t. E_Q needs to be normalized before measuring the similarity with an IaaS performance signature. Let E'_Q be the normalized trial performance where the normalization is performed based on its standard deviation. We denote E'_Q as $E'_Q = \{q'_1, q'_2, ...q'_n\}$. Let the signature of an IaaS service for the trial period T_f be S_Q for the QoS attribute Q where $S_Q = \{s_1, s_2, s_3, ...s_n\}$. The similarity between the normalized trial experience (E'_Q) and the signature of a service during the trial period (S_Q) using the Euclidean distance $(S(E'_Q, S_Q)^{ED})$ is computed by the following equation:

$$S(E'_Q, S_Q)^{ED} = \sqrt{\sum_{t=1}^{n}(s_t - q'_t)} \tag{6.1}$$

Similarly, the similarity measure using the Pearson Correlation Coefficients is computed using the following equation:

$$S(E^N_Q, S_Q)^{PCC} = \frac{\sum_{t=1}^{n}(s_t - \bar{s})(q'_t - \bar{q'})}{\sqrt{(s_t - \bar{s})^2}\sqrt{(q'_t - \bar{q})^2}} \tag{6.2}$$

where $\bar{q'}$ and $\bar{s}$ are the mean values of q' and s within the trial period T_f. The cosine similarity of the trial experience is measured by the following equation:

$$S(E_Q^N, S_Q)^{CS} = \cos\theta = \frac{\sum_{t=1}^{n} s_t q'_t}{\sqrt{\sum_{t=1}^{n}(s_i)^2}\sqrt{\sum_{t=1}^{n}(q')_i^2}} \tag{6.3}$$

Each of the above equations provides us with a similarity value between the trial experience and the corresponding IaaS performance signature. In the case of the Euclidean distance, the lower the distance, the higher the similarity.

A similarity threshold needs to be defined to determine how much deviation of the performance from the signature should be considered as the performance anomaly. We define a similarity threshold S_{thresh} for each technique. The threshold is used to distinguish between the normal performance behavior and performance anomalies. The initial threshold is defined during the signature generation process based on the experience of the past trial users' experience. Let us assume that there are N number of past trial users. The experience of the past trial users is denoted by $E_P = \{E_1, E_2, ...E_N\}$. The initial similarity threshold T_S for anomaly detection is defined as follows:

$$T_S = \min_{i=1}^{N} S(E_i, S_Q)^M \tag{6.4}$$

where M denotes the similarity measure method, i.e., PCC, ED, or CS. The threshold for different similarity measure technique can be different. When a new user performs a trial their observed performance has a similarity lower than the T_S, we consider it an anomalous performance behavior of the service. The value of the similarity threshold S_{thresh} is adjusted based on the experiments.

The event for signature change detection is defined as the frequent occurrences of performance anomalies within a fixed period of time as mentioned earlier. Therefore, we define an anomaly threshold for the event detection and denote as F_{thresh} which represents the minimum number of occurrences of the performance anomalies within a period of time T_f. The value of T_f is the length of the free trial period. We assume that each provider offers the same length of free trial without the loss of generality. The value of F_{thresh} can be initially defined as the number of past trial users within each T_f period who have the minimum similarity between

their experience and the corresponding signature. For example, if there are five past trial users who have the minimum similarity T_S with the present signature, then F_{thresh} is initialized as five. In such a case, the number of past trial users who have the minimum similarity during the signature generation process is considered as the usual number of performance anomalies within the T_f period. When the number of performance anomalies exceeds F_{thresh}, we consider it an event that needs to be evaluated for signature change detection. We update the value of F_{thresh} over time to detect the signature change effectively in the self-adjustment step based on the experiments.

6.2.3 Signature Change Detection

An event indicates that a signature may need to be re-evaluated. When an event is detected, the present signature needs to be tested to evaluate the event. This testing is the condition part of the proposed ECA approach. The main concern in the signature change detection is to differentiate between the performance anomalies and performance changes. This is similar to the signature processing domain, where the noise is a major concern for signal change detection. For instance, a voice recognition program has to differentiate between the noises in the environment and the voice of a new speaker. There exists a number of approaches for change detection in a signal or time series based on supervised, semi-supervised, or unsupervised methods [Aminikhanghahi and Cook, 2017]. We chose an unsupervised method called CUSUM which is a sequential analysis technique for small change detection in a time series. The CUSUM control chart is a simple and effective technique that is used in several areas such as signal processing, image processing, and intrusion detection in computer networks and security systems [Veeravalli and Banerjee, 2014].

A CUSUM control chart monitors the deviation of the individual or a group of samples from a target mean. Let us assume that the observation of a process P has the sequence $x_1, x_2, ...x_n$ with an estimated average of m_x and standard deviation s_x. The upper limit and the lower limit of the cumulative sum are defined by the following equations:

$$UL_i = \begin{cases} \max\left(0, UL_{i-1} + x_i - m_x - \frac{1}{2}ns_x\right), & i \geq 1 \\ 0, & i = 1 \end{cases} \tag{6.5}$$

$$LL_i = \begin{cases} \min\left(0, LL_{i-1} + x_i - m_x + \frac{1}{2}ns_x\right), & i \geq 1 \\ 0, & i = 1 \end{cases} \tag{6.6}$$

where UL_i is the upper limit, LL_i is the lower limit, n is the *minimum detectable shift* from the target mean. The process P is considered in violation of CUSUM criteria at the sample x_i if it obeys $UL_i > cs_x$ or $LL_i < -cs_x$ where c represents the control limit. The value c is adjustable and represents the number of standard deviations that the upper and lower cumulative sums are allowed to drift from the target mean.

Once an event is detected within a period of time T_f, we recompute a new signature (S^N) based on the trial experience of all the users within that period of time using the signature generation technique described in the previous chapter. The CUSUM control chart is applied to the new signature based on Equations 6.5 and 6.6. The target mean m_x and the standard deviation s_x are set based on the existing signature S within T_f period. The values of c and n are set based on the standard practice of CUSUM, which are s_x and $5s_x$ respectively. Once we detect the change in the IaaS performance signature within a time window of T_f, the existing part of the signature is replaced by the new signature.

6.2.4 Self-adjustment of the ECA Approach

When an event is detected and evaluated based on the proposed signature change detection technique, the outcome will be either a true positive or a false positive. A true positive implies that the signature needs to be updated. A false positive indicates that the signature does not need to be updated. The number of false positives can be reduced by adjusting the similarity threshold S_{thresh} for the anomaly detection and the anomaly threshold F_{thresh} for the event detection. For example, let us assume that the anomaly threshold for the event detection is set to five performance anomalies for a one-month trial period. If the TNPO detects five

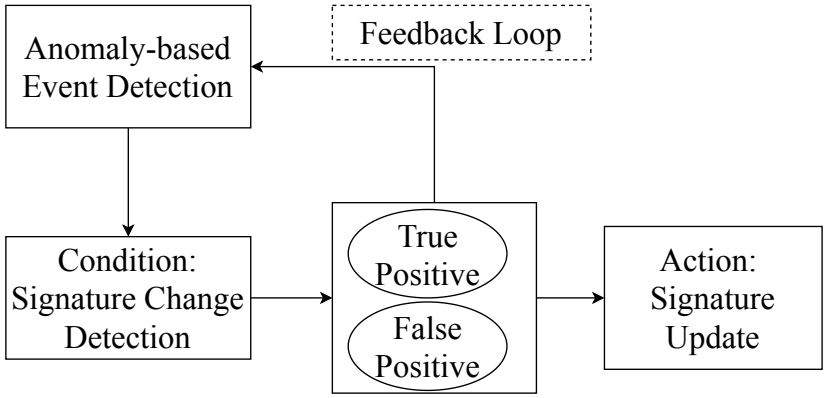


FIGURE 6.1: Self-adjustment of the ECA approach

anomalies in every month and the outcome is a true positive, then the anomaly threshold should have been reduced earlier to detect the change in IaaS performance. Similarly, if the outcome is a false positive every time, we need to increase the anomaly threshold for event detection. We apply a self-adjustment method using a feedback loop from the outcome of the signature change detection to the event detection so as to change the anomaly threshold.

Figure 6.1 shows the proposed self-adjustment method using a feedback loop. The outcome of the condition checking is fed to the anomaly-based event detection module. When the number of true positives or false positives within a predefined period of time T' exceeds a predefined threshold Z, the event detection module updates the frequency threshold F_{thresh}. The values of Z and T' are set by the TNPO. The frequency threshold is updated linearly based on the change detection outcome. When the outcome of signature change detection exceeds the true positive threshold then F_{thresh} is incremented by one. If the outcome exceeds the false positive threshold, then F_{thresh} is decremented by one.

6.2.5 Experiments and Results

A series of experiments were conducted to evaluate the proposed ECA approach. We identified two key attributes: a) the number of false positives, and b) the change detection delay to evaluate the proposed approach.

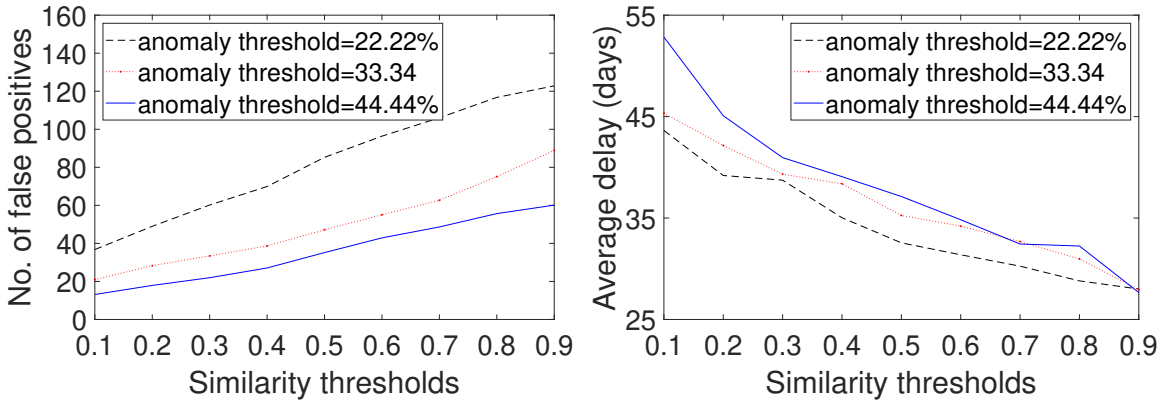
TABLE 6.1: Experimental settings

Variable Name	Values
Total number of simulation	100
Total provisioning period	360 days
Trial length of each consumer	30 days
Total number of IaaS performance signatures	5
Total number of Consumers	18
Similarity thresholds	0.1 to 0.9
Anomaly Thresholds	10% to 100%

6.2.5.1 Experiment Setup

Finding real-world workload traces and performance datasets for a long-term period was very challenging. Thus, we utilized the publicly available workload traces and performance data to mimic the long-term cloud environment. We used the Eucalyptus IaaS workload, which contains six workload traces of a production cloud environment, to generate the trial workloads of different consumers[1]. We selected a trace that contained 34 days of workloads of a large company with 50,000 to 100,000 employees. We partitioned the data into 360 parts and considered each partition as an average workload of one day to create a one-year workload dataset. The long-term performance of five IaaS providers was generated from the benchmark results published in SPEC Cloud IaaS 2016 [Fattah *et al.*, 2019]. We augmented the workload traces with the performance data to generate a long-term workload-performance dataset of five IaaS providers. We created the signature of each provider using the approach in the previous chapter. The experimental settings are shown in Table 6.1. We have created approximately 100 new signatures to simulate the change in IaaS signatures from the signature of five IaaS providers. To create a new signature, first we selected a random index in one of the IaaS signatures. This index is called change index. We then computed the moving average for each 10 data points in the signature starting from the change index to the rest of the signature. The original signature is then altered by replacing with the moving average values from the change index to the end of the signature.

[1] https://www.cs.ucsb.edu/~rich/workload/



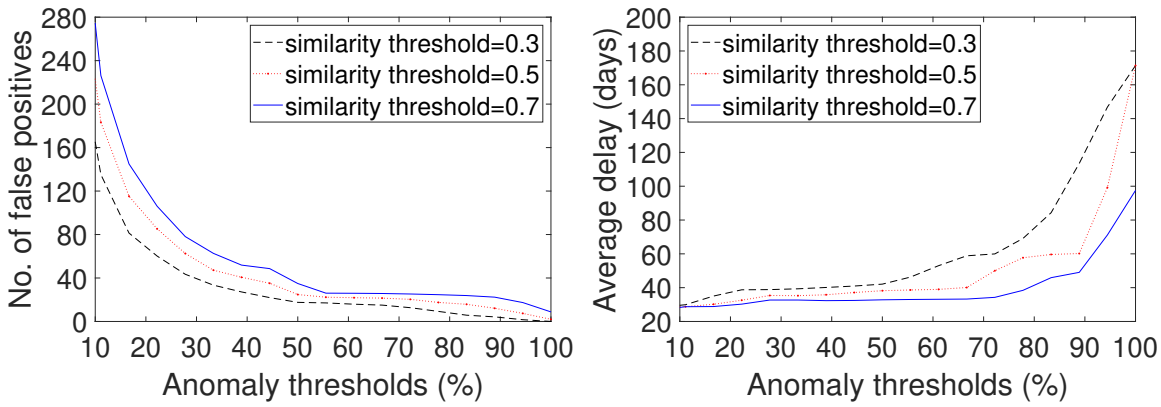
(a) (b)

FIGURE 6.2: Effects of different similarity thresholds in change detection in (a) number of false positives (b) average delay

6.2.5.2 Evaluation and Discussion

The proposed approach aims at reducing the number of false positives and the change detection delay varying the similarity threshold and the anomaly threshold for the anomaly and event detection, respectively. We discuss only the results of similarity measure using PCC as it provides the best results. Figure 6.2(a) shows the number of false positives that are generated before the actual change detection for the different values of the similarity thresholds. The anomaly thresholds are set from 22.22% to 44.44% of the total number of consumers within a given trial month. The number of false positives increases with the increase of similarity thresholds according to Figure 6.2(a). The reason for this is that when the similarity threshold is increased, the number of detected anomalies increases. As a result, the number of detected events also increases resulting in a high number of false positives. The number of false positives directly affects the delay in signature change detection. The average delay in change detection is illustrated in Figure 6.2(b). The average delay is reduced with the increase of similarity threshold for anomaly detection. This implies that when the number of false positives increases, the average detection delay is reduced due to the increasing number of testing.

The anomaly threshold for the event detection impacts the result in the opposite way of changing the similarity thresholds. Figures 6.3(a) and (b) illustrate the effect of changing

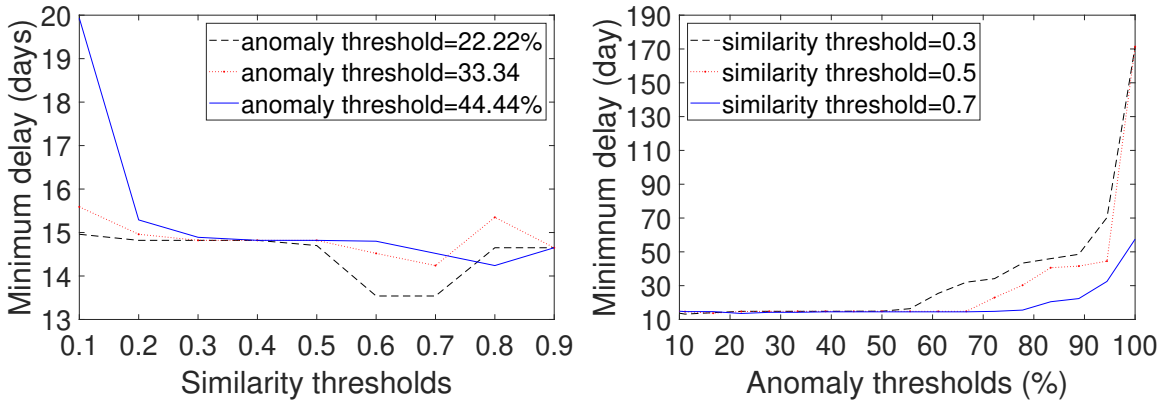


(a) (b)

FIGURE 6.3: Effects of different anomaly thresholds in change detection in (a) number of false positives (b) average delay

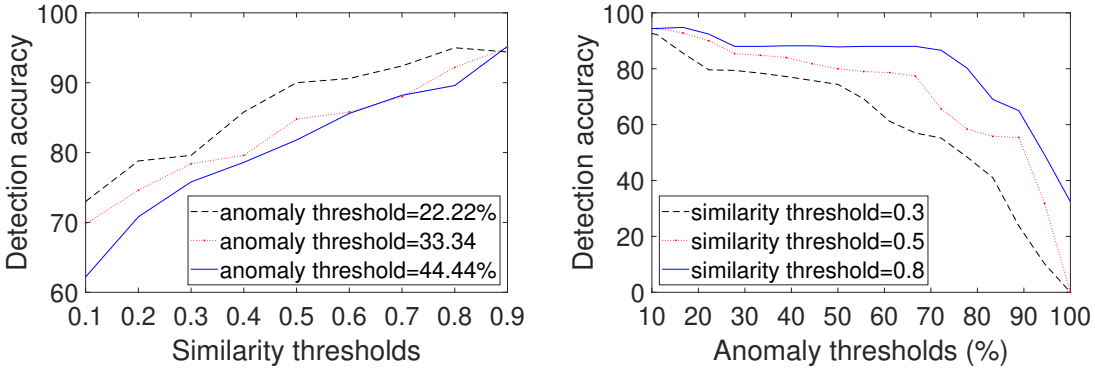
the anomaly detection threshold on the number of false positives and the detection delay, respectively. The number of false positives decreases *exponentially* with the increase of the anomaly threshold. For instance, when the anomaly threshold is at 100% of the total consumer, the number of false positives becomes almost zero. The reason for such a result is that when the anomaly threshold is increased, the proposed framework accepts a higher number of performance anomalies as the normal behavior of the service. As a result, when the anomaly threshold is 100% of the total trial users at a point of time, an event is detected only if every trial user observes anomalous performance behavior. Similarly, Figure 6.3(b) depicts that the increase in the anomaly threshold increases the average detection delay. The reason is that when the anomaly threshold is increased, the number of detected events becomes lower. When the number of events is decreased, the number of testing of signature is also decreased. This result could be inferred from the impact of anomaly thresholds on the number of false positives. Intuitively, if the number of false positives decreases, the average detection delay should increase because of the lower number of performed tests.

Figure 6.2(b) shows that the average change detection delay varies between 30 to 55 days, which is reasonable given the one-month trial period window. Figure 6.3(b) shows that the average detection delay varies between 15 to 180 days. The high value in the result could indicate that the proposed approach is unable to detect changes in some signatures. Figure 6.4 shows that the actual delay for most of the cases is much lower. Figures 6.4(a) and (b)



(a) (b)

FIGURE 6.4: Minimum delay in change detection for (a) different similarity thresholds (b) different anomaly thresholds



(a) (b)

FIGURE 6.5: Accuracy of the change detection for (a) different similarity thresholds (b) different anomaly thresholds

show the minimum change detection delay for different similarity thresholds and anomaly thresholds, respectively. The value of minimum change detection varies from 15 days to 60 days in most cases. The high value of the average change detection delay indicates that the proposed approach is unable to detect some of the changes at all. As a result, the average delay in change detection increases.

If we wait for a long time, the proposed approach may be able to detect changes in all signatures. However, waiting for an uncertain period to detect the change is unrealistic. We therefore set a *time window* T_w. We evaluate the proposed approach in terms of its ability to

detect changes in each signature within T_w. We set the value of T_w to 60 days based on the average change detection delay as shown in Figure 6.2(b). If a change in a signature is not detected within the first 60 days of the actual change, we consider that the proposed approach is unable to detect the change for that particular signature. Figure 6.5(a) shows that the detection accuracy increases from 40% to 95% with the increase of the similarity threshold. This indicates that the proposed approach is able to detect the change for up to 90% of the signatures within the first 60 days when the similarity threshold is very high. It is important to note that high accuracy leads to a higher cost in terms of the number of performed tests. Figure 6.5(b) shows that the change detection accuracy decreases from 95% to below 10% with the increase of anomaly threshold for the event detection. This result is also consistent with the previous results. The high anomaly threshold leads to a lower number of testing which results in lower accuracy in the change detection. The accuracy results indicate that the similarity threshold and anomaly threshold need to be adjusted separately for each signature to improve the performance, which can be accomplished using the proposed self-adjustment method.

6.3 Performance Noise-based Signature Change Detection

A key challenge in long-term IaaS performance change detection is to accurately identify IaaS performance noise and true changes in the performance behavior of a service. Noise in signal processing typically refers to unwanted disturbances in electrical signals, which is usually generated during the capture, storage, transmission, processing, or conversion of the signal [Whalen, 2013]. In the context of IaaS performance, noise can stem from noisy neighbor effects (co-tenancy), periodic updates, or unwanted disruptions in service provisioning [Varadarajan *et al.*, 2012]. The IaaS performance change detection problem has two phases: a) change point detection, and b) change detection. IaaS performance noise should be considered in both phases as it may affect the performance of the change detection process. The proposed signature-based change detection approach in the previous chapter considers the performance noise only in the change detection phase during the re-evaluation of the existing performance signatures.

In this section, we introduce a new change detection approach where performance noise is considered during the change point detection. The performance observed by each trial user may be affected by the performance noise. In such a case, the performance of a change point detection process may degrade in terms of the number of false positives if the effect of noise is not considered. For instance, the performance observed by a large number of consumers may not be *exactly* similar to the corresponding performance signature due to the noise. In such a case, a change point detection algorithm will trigger a false alert.

We propose a novel change detection framework that aims at identifying performance changes more effectively. The proposed framework introduces a new type of IaaS performance signature called *categorical IaaS signature*. The categorical IaaS signature models performance behavior more *accurately* than the *general IaaS signature* introduced in [Fattah *et al*., 2020b] as the general IaaS signature does not consider the effect of different categories of workloads, i.e., CPU-intensive, I/O-intensive, and memory-intensive, on IaaS performance. The proposed framework utilizes a heuristic-based approach to determine noise in IaaS performance. In this approach, the categorical signature and the general signature are utilized to define performance noise bandwidth. The performance noise bandwidth is updated over time to detect performance changes more accurately. The key contributions are summarized as follows:

- A new type of IaaS performance signature named Categorical IaaS Signature that models an IaaS service's long-term performance behavior based on different categories of workload.
- A novel performance noise model that defines the noise bandwidth based on the categorical and general IaaS signatures.
- A performance change detection model that leverages the proposed performance noise model to detect changes in IaaS performance.

6.3.1 General IaaS Performance Signatures

The general IaaS performance signature is first introduced in Chapter 5. The general signature of an IaaS service is represented based on its relative performance changes over time, i.e., how much a service's performance may increase or decrease in one time period compared

to another time period. For example, the general signature of a VM may inform that its response time is expected to increase by 5% on weekend nights over the response time on regular weekdays. The general signature focuses mainly on the effect of seasonality on IaaS performance. It assumes that the effect of different types of workload on the observed performance is not substantial compared to the effect of seasonal performance variability. Therefore, this signature is called general signature as it considers all types of workload equally. Note that the signature does not provide the exact performance of a service. Therefore, a consumer is unable to select a service based only on its signature. Instead, the consumer needs to perform the trial with their application workloads and utilizes the trial experience and the IaaS signature to estimate the long-term service performance [Fattah *et al*., 2020b].

DEFINITION 11. *General IaaS Performance Signature.* An IaaS performance signature is a temporal representation of relative performance changes of an IaaS service over a long period.

6.3.2 Categorical IaaS Performance Signatures

In this subsection, we introduce a new type of signature called categorical IaaS performance signature. For simplicity, we refer to the categorical IaaS performance signature as the categorical signature and the general IaaS performance signature as the general signature. The motivation behind creating the categorical signature is to produce a more accurate signature that captures the effect of different types of workload on IaaS performance behavior. The performance of an IaaS service may depend on the workload it runs [Feitelson, 2002]. Therefore, IaaS providers often advertise CPU-intensive, memory-intensive, or network-intensive VMs. For instance, Amazon EC2 offers a wide range of compute-optimized, storage-optimized, and memory-optimized instances[2].

IaaS workloads can be categorized based on several workload parameters such as resource requirements, request arrival rates, and workload distribution. Without loss of generality, we only consider resource requirements as workload parameters for categorization in this work. Therefore, workload categories will be CPU-intensive, memory-intensive, and I/O intensive. The proposed workload categorization is applicable for any other workload parameters. Let

[2]https://aws.amazon.com/ec2/instance-types/

us assume there are N_c types of workload based on resource requirements of consumer requests. Therefore, we create N_c number of categorical signatures. A categorical signature is represented as:

$$S_c = \begin{bmatrix} s_{c11} & s_{c12} & .. & s_{c1t} \\ s_{c21} & s_{c22} & .. & s_{c2t} \\ s_{c31} & s_{c13} & .. & s_{c3t} \\ .. & .. & ... & \\ s_{cn1} & s_{cn2} & .. & s_{cnt} \end{bmatrix} \tag{6.7}$$

where S_c represents the signature for c categories of workloads. Here, c is one of the categories in N_c. The rest of the attributes of Equation 6.7 are the same as the general signature.

6.3.3 Categorical IaaS Performance Signature Generation

The key difference between the categorical signature generation and the general signature generation is the consideration of different workload categories. First, we define a set of categories (C) based on the resource requirements where $C = \{1, 2, 3, ...N_c\}$. For each category, we define the criteria that determine the category of each request (workload). Let us assume that a consumer's request has R number of attributes, where each attribute denotes a resource in the VM such as vCPU, storage, or memory. For each attribute (a), we define a minimum resource requirement M_a. If a request has more than M_a amount of resource requirement for the attribute a, we consider that request as an a-intensive request. For example, if a request has 80% of CPU usage requests, then we consider that request as a CPU-intensive request. According to this approach, a request can exist in multiple categories of workloads. The minimum resource requirement for each attribute is defined experimentally, i.e., the different threshold is considered as the minimum resource requirement for each category to find the most effective threshold. Once we define the category for each workload, we create the categorical signature as follows:

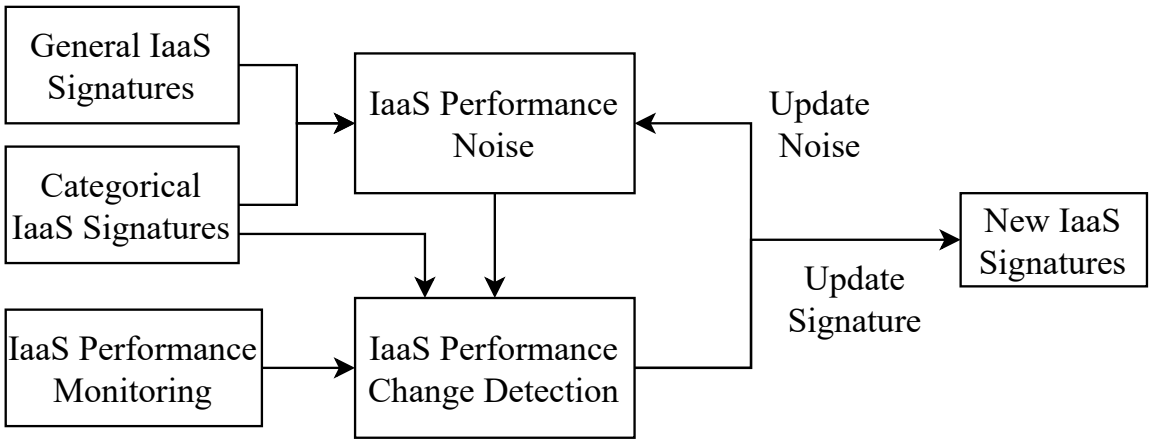


FIGURE 6.6: IaaS Signature change detection framework

(1) For a QoS attribute Q, the performance observed by the trial users is collected over time T.

(2) For each category a at each trial length δT, we identify k number of a-intensive requests. The average performance ($\overline{Q_k}$) is measured for each QoS attribute.

6.3.4 Proposed Change Detection Framework

In this subsection, we discuss the proposed change detection framework as shown in Figure 6.6. The proposed framework consists of two key components: a) IaaS performance noise and b) IaaS performance change detection. The performance noise is initially defined by the general signature and the categorical signature. The performance noise is then updated dynamically based on the observed performance of the free trial users. The change detection framework utilizes the knowledge of IaaS performance noise and the categorical IaaS signature to detect changes in the categorical signature based on the observed performance by the free trial users. The change detection framework updates the knowledge about the performance noise based on the observed performance over time.

6.3.5 IaaS Performance Noise

A key step in identifying changes in IaaS performance is to accurately determine the noise in IaaS performance. We define the noise in IaaS performance as the deviation from the expected

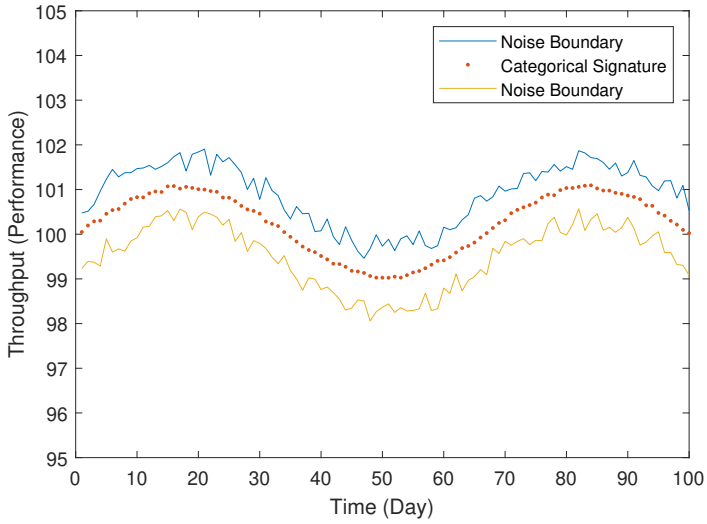


FIGURE 6.7: IaaS performance noise bandwidth

performance behavior as represented by the signature of an IaaS service. The key challenge in defining the performance noise is to determine the amount of performance fluctuation from the expected performance behavior. A boundary must be defined, which will determine whether the observed performance fluctuations can be considered as the noise or a permanent change in the performance behavior. In signal processing, image processing, and other domains, there are many approaches to define and detect different types of noises such as White noise, Gaussian noise, and Salt and pepper noise. *To the best of our knowledge, there is no definitive way of defining noise in the case of IaaS performance behavior.* Therefore, we propose a heuristic-based approach using the general signature and the categorical signature to define the initial performance noise boundary of an IaaS service. We call it *IaaS performance noise bandwidth*. The noise bandwidth is updated over time based on the observed performance behavior of an IaaS service. The performance noise bandwidth is defined as follows:

DEFINITION 12. *IaaS Performance Noise Bandwidth.* The surrounding area created by the acceptable fluctuation from the expected performance of an IaaS service is the IaaS performance noise bandwidth of that service.

The amount of acceptable fluctuation is initially defined by the general signature and the categorical signature as shown in Figure 6.11. The distance between the general signature and the categorical signature D is computed for each timestamp. D is then considered as

the acceptable deviation from the expected performance as represented by the categorical signature. Any observed performance that has the maximum deviation D from the categorical signature is considered be noisy performance. The noise bandwidth is later updated based on the observed performance by the free trial users in the change detection step.

6.3.6 IaaS Performance Change Detection

Detecting changes in IaaS performance requires monitoring the current performance behavior of an IaaS service. We assume that the TNPO continues to monitor the experience of free trial users after creating the signatures. When most of the users' trial experience does not *match* with the corresponding categorical signature, the existing signature needs to be re-computed. We represent the signatures and the trial experience as time series. Therefore, the matching of trial experience and signature has two parts: a) distance, and b) shape. When the observed performance of a user has a distance from the categorical signature within the performance noise bandwidth, and the shape of the observed performance is similar to the categorical signature, we assume that there is no change in performance. We identify the following cases during the matching based on the shape and the distance:

(1) Case 1: Most of the users' observed performance is within the noise bandwidth, and the shape of the performance is similar to the corresponding categorical signatures. In this case, no action is taken.
(2) Case 2: Most of the users' observed performance is outside the noise bandwidth, and the shape of the performance is not similar to the corresponding categorical signatures. In this case, signatures are required to be re-computed.
(3) Case 3: Most of the users' observed performance is within the noise bandwidth, and the shape of the performance is not similar to the corresponding signatures. In this case, we reduce the size of the performance noise bandwidth.
(4) Case 4: Most of the users' observed performance is outside but adjacent to the noise bandwidth, and the performance shape is similar to the corresponding categorical signatures. In this case, we increase the size of the performance noise bandwidth.

Let us assume that the noise bandwidth at timestamp t is defined by d^+ and d^- where d^+ is the distance from the categorical signature to the noise boundary on the upper side of Y-axis, and d^- is the distance from the categorical signature to the noise boundary on the downside of Y-axis. Therefore, we need to measure whether the observed performance d is in between d^+ and d^- at each timestamp. The first two cases are straightforward. We define a threshold Th. When Th percentage of the users' observed performance matches with case 1 or case 2, we either take no action or update the signature. The value of T_h is set experimentally. In case 3, if T_h percentage of users' performance is within the noise bandwidth and their shape does not match, we reduce the performance noise bandwidth. We experimentally define a similarity threshold T_s, which determines the minimum acceptable similarity between observed performance and the categorical signature. After reducing the bandwidth, we apply the change detection process again for each user's observed performance. In case 4, we increase the size of the noise bandwidth based on the observed performance and apply the change detection process again. We define a threshold δd, which determines how much noise bandwidth needs to be increased or decreased in cases 3 and 4. The value of δd is set based on experiments.

6.3.7 Experiments and Results

6.3.7.1 Experiment Setup and Datasets

A series of experiments were conducted to evaluate the proposed change detection approach. We identified two key attributes: a) average delay and b) ability to detect changes or detection accuracy to evaluate the proposed approach. The proposed approach is compared with the existing IaaS performance changed detection approach proposed in Section 6.2.

The experiment setup is same as section 6.2.5.1. In addition, we identified the following two key variables in this experiment that drives the performance of the proposed approach:

- Similarity Threshold: The similarity threshold indicates the minimum similarity between the shape of the observed performance in the trial of a consumer and the corresponding signature. The similarity threshold is utilized to determine shape-based similarity.

TABLE 6.2: Experimental settings

Variable Name	Values
Total provisioning period	360 days
Trial length of each consumer	30 days
Total number of IaaS performance signatures	5
Total number of Consumers	18
Similarity thresholds	.6 to 0.9
Anomaly Thresholds	60% to 90%

- Anomaly Threshold: The proposed change detection framework relies on the trial experience of the majority of the users. Based on the observations of the majority of the users, we either confirm change or update performance noise. The anomaly threshold defines the minimum number of users that is considered as the majority of the users.

6.3.7.2 Evaluation and Discussion

We evaluate the proposed approach in terms of the average delay to detect signature changes and its ability to detect true changes in signature. The expectation is to reduce the average delay to detect the change in performance and increase the accuracy of detecting changes. Here, accuracy refers to the true positives, i.e., how many changes the proposed approach is able to detect. Figure 6.8 depicts the results of experiments. Figures 6.8(a) and (b) show the average delay in detecting changes. Figure 6.8(a) shows the average delay for different similarity thresholds. There is no trend visible that indicates that there is a linear relationship between the similarity threshold and average change detection delay. The figure shows that the average delay is the minimum when the similarity threshold is about 90%. However, the average also depends on the anomaly threshold. When the anomaly threshold is about 70%, the average delay is minimum in most cases in Figure 6.8(a). Similarly, Figure 6.8(b) shows the average delay for different anomaly thresholds. It also shows no common trend in the average detection delay based on the anomaly threshold. The average delay is the minimum when the anomaly threshold is about 70%, and the similarity threshold is about 80%.

The average delay is not the only attribute to measure the performance. We consider the accuracy of the proposed approach in terms of its ability to identify true changes correctly. Figures 6.8(c) and (d) show the accuracy of the proposed approach. In Figure 6.8(c), the

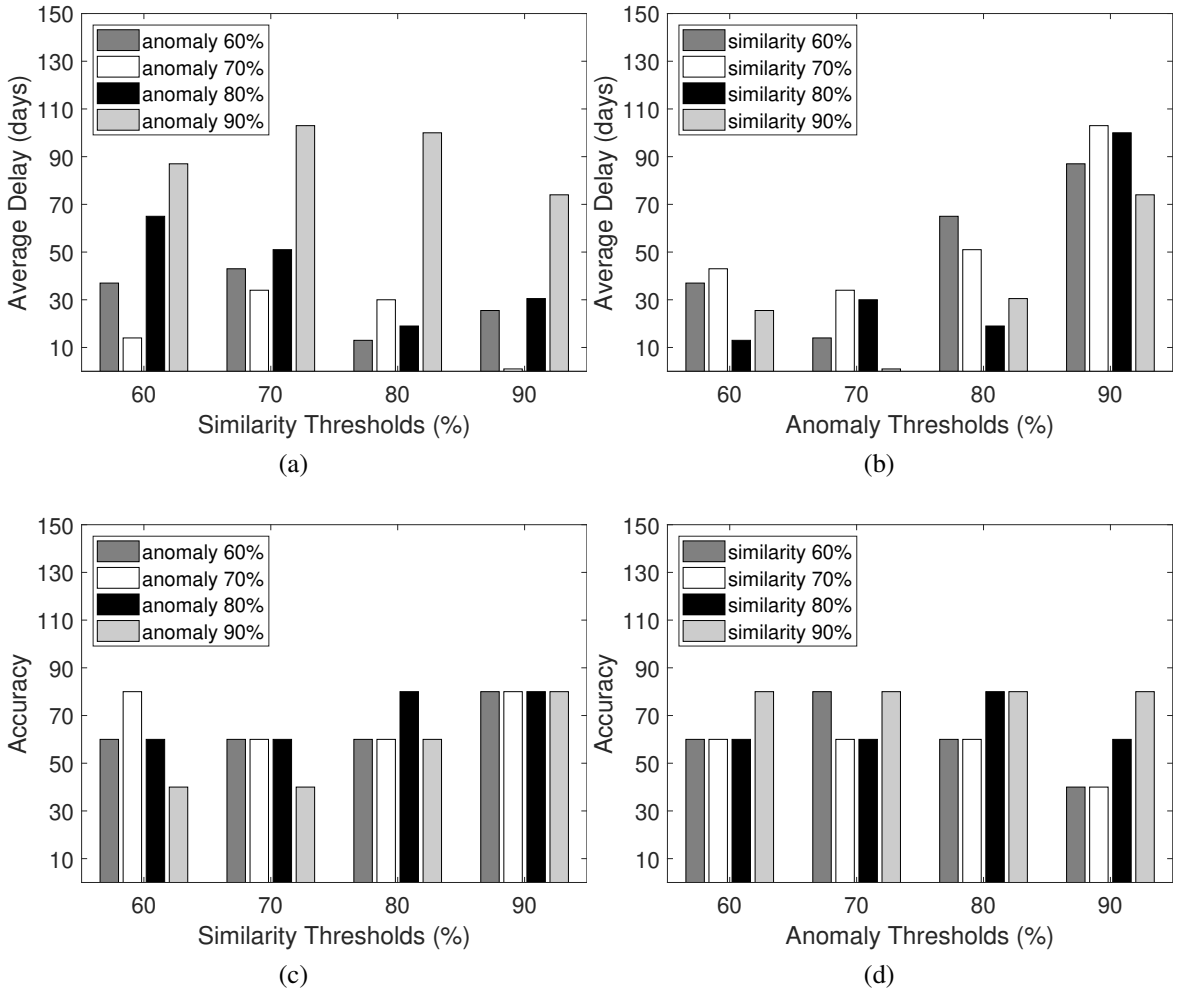
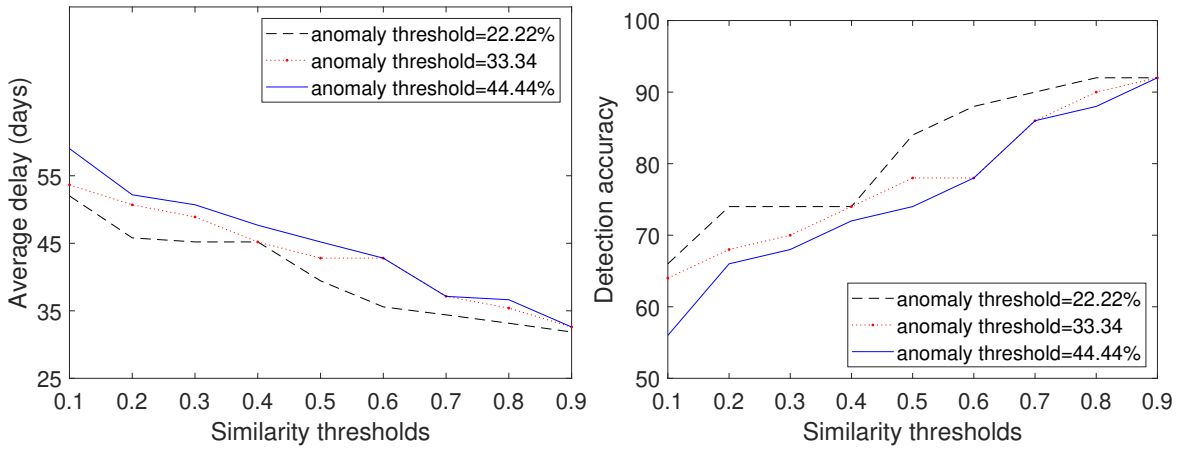


FIGURE 6.8: Experiment results (a) average delay for variable similarity thresholds (b) average delay for variable anomaly thresholds (c) accuracy for variable similarity thresholds (d) accuracy for variable anomaly thresholds

accuracy is illustrated with respect to the different similarity thresholds. The accuracy of the proposed approach is about 80% when the similarity threshold is 90%. The effect of different anomaly thresholds is not very substantial on the accuracy according to the figure. Figure 6.8(d) illustrates the accuracy with respect to the anomaly threshold. When the anomaly threshold is about 90%, that means 90% of the users' experience do not match the corresponding signature, and the similarity threshold is about 90%, while the accuracy of the proposed approach is about 80%. The proposed approach finds the changes in IaaS performance based on an iterative approach that conditionally updates the performance noise.



(a) (b)

FIGURE 6.9: Performance of the ECA approach (a) average delay (b) accuracy

Therefore, the change detection process stops when the suitable performance noise bandwidth is measured, confirming whether there is a change in the signature.

We have implemented the proposed ECA approach in [Fattah and Bouguettaya, 2020b] and applied it to our dataset. The result of the ECA approach is illustrated in Figure 6.9. Figure 6.9(a) shows the average delay for different similarity thresholds and anomaly thresholds in the ECA approach. The average delay in this approach can be 35 days to 55 days, depending on the similarity and anomaly thresholds. The average delay in our approach can be from 2 days to 110 days, depending on the similarity and the anomaly thresholds. Choosing the right similarity and anomaly threshold provides a better result than the ECA approach in terms of average change detection delay. The detection accuracy in Figure 6.9 shows that the ECA approach provides accuracy from 60% to 90%, depending on the similarity and anomaly thresholds. The proposed approach in this work has an accuracy of about 60% to 80%. However, it does not produce any false positives, while the proposed ECA approach in [Fattah and Bouguettaya, 2020b] produces a significant number of false positives.

6.4 Signature Change Detection Framework

We propose a new signature change detection framework in this section. The proposed framework leverages change detection techniques from signal processing domain to detect changes. The focus of this framework is to distinguish the true changes in IaaS performance from the changes that are caused by performance noise during the change detection. Noise in signal processing typically refers to the unwanted disturbance in electrical signals [Whalen, 2013]. There are numerous types of noises in the signal processing domain. In the context of IaaS performance, we define three types of IaaS performance noises: spikes, attenuation, and distortion. We propose a novel change detection framework that identifies noises and true changes in long-term IaaS performance behavior. The proposed framework leverages time series similarity measure techniques and a sliding window approach to identify noise in IaaS performance. In addition, we introduce a *Signal-to-Noise Ratio* or SNR-based approach to improve the performance of the proposed framework. We conducted a set of experiments based on real-world datasets to evaluate the proposed framework.

- A novel IaaS performance change detection framework that utilizes experience of free trial users and existing IaaS signatures to detect long-term changes in IaaS performance behavior.
- An IaaS performance noise model that represents different types of noise in IaaS performance signatures.
- A sliding window-based change detection technique to detect changes in IaaS performance signatures when there is no prior knowledge about the performance noise.
- A Signal-to-Noise Ratio based approach to detect changes in IaaS performance signature when prior knowledge about performance noise is available.

6.4.1 Proposed Framework

IaaS performance change detection typically consists of two parts: a) change point detection (CPD), and b) change detection. The CPD process identifies the points in time at which a change in performance may occur. Once a change point is identified, the change detection

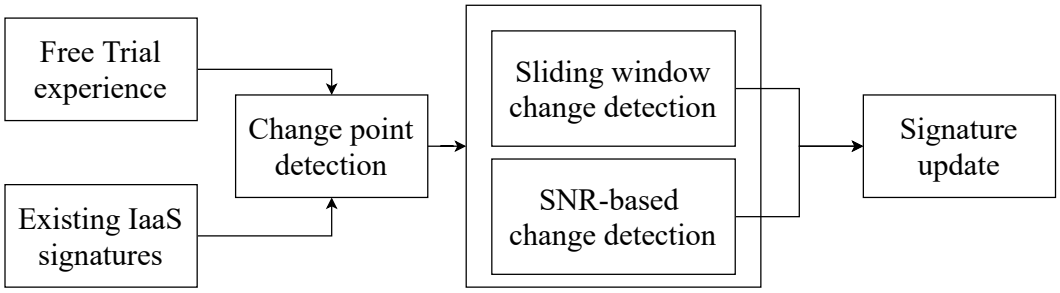


FIGURE 6.10: IaaS signature change detection framework

process recomputes the signature and evaluates whether the signature needs to be updated. The signature is then updated based on the evaluation of the change detection process. As our main focus is on the change detection process, we assume that the CPD process will be performed using the proposed approach in [Fattah and Bouguettaya, 2020b]. The reason for including the CPD process is to provide a holistic view of the change detection process. Figure 6.10 shows the proposed change detection framework. The framework takes as input the free trial experiences of potential consumers and existing IaaS signatures. The change points are detected by comparing the free trial experiences and existing IaaS signatures. Once a change point is identified, the change detection module applies two change detection approaches. The first approach assumes there is no prior knowledge about IaaS performance. This approach relies on time series similarity measure techniques and a sliding window approach. The second approach assumes that we have prior knowledge about the performance noise. The performance change is detected using prior knowledge about performance noise. In this approach, we utilize signal-to-noise ratio to differentiate between noise and change in IaaS performance. In the following subsections, we provide a brief overview of the change point detection and the change detection approaches.

6.4.2 Change Point Detection

Change point detection is formally known as the problem of finding abrupt changes in data when a property of a system or an entity changes. Change point detection is a pre-requisite of change detection, i.e., identifying the points in time when the likelihood of change occurrences is high. We utilize the proposed ECA approach in Section 6.2 in this framework.

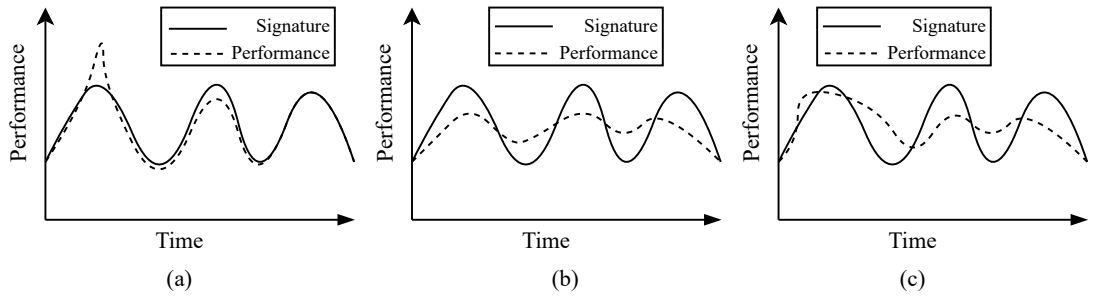


FIGURE 6.11: IaaS performance noise (a) spike, (b) attenuation, and (c) distortion

6.4.3 IaaS Signature Change Detection

6.4.3.1 IaaS Performance Noise

A change point indicates that a change may have occurred in IaaS performance behavior. After a change point has been detected, the existing signature needs to be tested to ascertain whether a revaluation of the signature is required. The main challenge in this ascertainment is to differentiate between the noise and change in performance. We define the noise as the irregular or abnormal performance behavior in the performance of an IaaS service with respect to the corresponding IaaS signature. We identify the following three types of noise that may appear in IaaS performance as shown in Figure 6.11.

(1) **Spikes**: Spikes in IaaS performance typically originate from various uncertain and sudden changes in performance (Figure 6.11(a)). For instance, an IaaS provider may face a major power failure of its infrastructure. These types of noise typically exist for a short period. The performance becomes normal once the provider resolves the issue.

(2) **Attenuation**: The second type of noise is caused by attenuation. Attenuation in a signal usually refers to the loss of power in signal strength; in the case of an IaaS signature, it has a similar effect (Figure 6.11(b)). The performance of an IaaS service may not increase or decrease as expected according to the signature. However, the performance has a similar shape to its signature. Attenuation may occur due to the effect of multi-tenancy in the cloud.

(3) **Distortion**: Distortion refers to random noise in performance (Figure 6.11(c)). These types of noise are very irregular and do not follow any particular pattern. Distortion may be originated from multi-tenancy, scheduled maintenance, security attacks, sudden hazards, and so on. Differentiating these types of noises from performance change is very challenging.

First, we develop a change detection approach that applies a sliding window-based detection technique that identifies the noises in IaaS performance. In this case, we assume that we do not have any historical knowledge about performance noise. Next, we develop an SNR-based approach to improve the performance of the change detection process. In this step, we assume that we have prior knowledge about the nature of noise. In the following subsections, we describe these two approaches.

6.4.3.2 Sliding Window Change Detection

Once a change point has been detected, the signature is recomputed based on the experience of current trial users using the method described in Chapter 5. The existing signature and the recomputed signature need to be compared to detect changes in performance. Both signatures are represented as time series. Therefore, we may utilize existing time series similarity measure techniques to compute the similarity between the two signatures. There are numerous approaches in the existing literature to measure time series similarity based on the shape and distance such as Pearson Correlation Coefficients (PCC), Euclidean Distance (ED), Spearman Correlation (SC), Cosine Similarity (CS), Symbolic Aggregate Approximation (SAX), and Dynamic Time Warping (DTW). However, most of these techniques do not consider noise in time series during the similarity measure. Therefore, we develop a sliding window-based noise detection technique to differentiate between noise and changes in IaaS performance. The sliding window-based approach leverages existing time series similarity measure techniques. In particular, we will use PCC and RMSE (Root Mean Squared Error) to measure similarities between two signatures based on shape and distance. We utilize these two similarity measure techniques as they are effective, easy to implement, and most commonly used techniques.

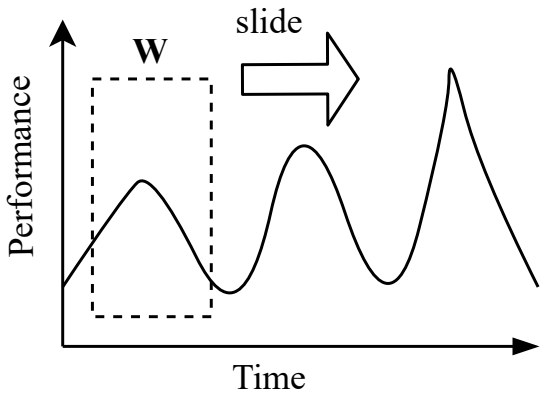


FIGURE 6.12: Sliding window approach to identify performance noise

We start with the assumption that there is no noise in the IaaS performance. Therefore, we measure the similarity based on PCC and RMSE distance between the existing signature and the recomputed signature. The PCC is used to compare the shape between two time series using the following equation:

$$Similarity(S, S')^{PCC} = \frac{\sum_{t=1}^{n}(s_t - \bar{s})(s'_t - \bar{s'})}{\sqrt{(s_t - \bar{s})^2}\sqrt{(s'_t - \bar{s})^2}} \tag{6.8}$$

where $\bar{s'}$ and $\bar{s}$ are the mean values of s' and s within the period T_n. S and S' are the existing and recomputed signature respectively. The next step is to measure the distance between the two signatures using RMSE, as follows:

$$RMSE(S, S') = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(S(t) - S'(t))^2} \tag{6.9}$$

where S and S' are the existing and recomputed signature over time T. $S(t)$ and $S'(t)$ denote the value of S and S', respectively at time t.

When the existing signature and the recomputed signature have high similarity and a low RMSE distance, we assume that the existing signature does not require re-evaluation. Therefore, the detected change point would be considered a false positive. The thresholds S^P and S^R for the PCC and RMSE, respectively, need to be predefined. We defined these thresholds

experimentally. If the PCC or RMSE are lower than these thresholds, we need to decide whether the existing signature should be updated. If we can identify the noise in the recomputed signature, and if by removing the noise the similarity increases considerably, we do not change the existing signature.

If the shapes of the signatures are very similar and the RMSE distance is within an acceptable threshold T^D, we consider it to be the effect of attenuation. Therefore, we do not change the existing signature in this case. If the PCC value is low and the RMSE distance is also low, they indicate the shape has been altered. It could have resulted from either performance change or noise. If it is because of noise, then the noise is either spike or distortion. We can detect spikes in IaaS performance using a sliding window-based approach as shown in Figure 6.12. In this approach, we start with an assumption that the lower value of the PCC is originated from spikes. Therefore, if we can identify and remove spikes from the recomputed signature, the similarity score will increase. Hence, we define a window W. The length of W should be greater than the size of the spike to identify the noise. Therefore, the length of W should be defined based on the size of the spike.

We scan the signature from beginning to end using the W length, and each time we remove the W portion of the signature and recompute the value of PCC. If the similarity score increases and reaches up to the threshold S^P, then we consider it to be noise. Therefore, we do not update the existing signature. We are unable to distinguish between the change and noise in this approach if the noise is originated from distortion. In this case, we need to rely on experience about the noise.

6.4.3.3 SNR-based Change Detection

We discuss the change detection approach with prior knowledge about performance noise. In this approach, we rely on historical information rather than time series similarity measures. As a result, we do not need to identify different types of noise separately. We leverage a noise detection technique from the signal processing domain to identify noise in IaaS performance.

We assume that once a signature has been generated, the TNPO monitors the performance of free trial users for a period T (e.g., one year). In this period, we assume that the IaaS signature of a service does not change considerably. Therefore, the TNPO monitors performance without any intervention to the existing signature to understand performance noise. The TNPO monitors the noise and measures the Signal-to-Noise Ratio (SNR). SNR is used to measure the level of the desired signal and the level of background noise. SNR is typically expressed as the ratio of signal power to noise power, often expressed in decibels. A ratio higher than 1:1 (greater than 0 dB) indicates more signal than noise. In the context of change detection, we consider the existing IaaS signature as the desired signal and the IaaS performance noise as background noise. The SNR value is computed as follows:

$$SNR = \frac{E(S^2)}{E(N^2)} \tag{6.10}$$

where S is the signal and N is the noise. E refers to the expected value, i.e., in this case, mean square of S and N. When the noise has an expected value of zero, then the denominator is its variance rather than the mean. We assume that the TNPO divides the T period into d number of segments. The value of d can be adjusted based on the noise in the different seasonal periods, i.e., daily, weekly, monthly. The TNPO requires recomputing the signature in these d periods to compute the noise. The noise (N) can be computed by measuring the difference between the existing signature and the recomputed signature using the following equation:

$$N = S - S^r \tag{6.11}$$

where S is the existing signature and S^r is the recomputed signature in the noise learning period. For each d period, we measure the SNR value. Once the monitoring period is finished, we start the change detection process. Once we detect a change point, we recompute the signature based on the experience of current users. We then compute the SNR value of the current signature compared to the recomputed signature. If the current SNR value is close to any existing SNR values, we consider the change point as a false positive and do not change

the existing signature. We define a threshold T^{SNR} experimentally to determine the minimum SNR distance between the current SNR and past SNR values. The change detection process can be improved by adjusting this threshold.

6.4.4 Experiments and Results

A series of experiments are conducted to evaluate the proposed sliding window and SNR-based approaches. The proposed approaches are compared with the existing IaaS performance change detection approach, i.e., cumulative sum control chart (CUSUM) proposed in Section 6.2.

6.4.4.1 Experiment Setup

Finding real-world workload traces and performance datasets for a long-term period was very challenging. Thus, we utilized the publicly available workload traces and performance data to mimic the long-term cloud environment. We used the Eucalyptus IaaS workload, which contains six workload traces of a production cloud environment, to generate the trial workloads of different consumers[3]. We selected a trace that contained 34 days of workloads of a large company with 50,000 to 100,000 employees. We partitioned the data into 360 parts and considered each partition as an average workload of one day to create a one-year workload dataset. The long-term performance of five IaaS providers was generated from the benchmark results published in SPEC Cloud IaaS 2016 [Fattah *et al.*, 2019]. We augmented the workload traces with the performance data to generate a long-term workload-performance dataset of five IaaS providers. We created the signature of each provider using the approach in the previous chapter. We have created approximately 100 new signatures to simulate the change in IaaS signatures from the signature of five IaaS providers. To create a new signature, first we selected a random index in one of the IaaS signatures. This index is called change index. We then computed the moving average for each 10 data points in the signature starting from the change index to the rest of the signature. There are various types of noise known in signal processing domain such as white noise, black noise, Gaussian noise, Cauchy noise.

[3] https://www.cs.ucsb.edu/~rich/workload/

TABLE 6.3: Experimental Settings

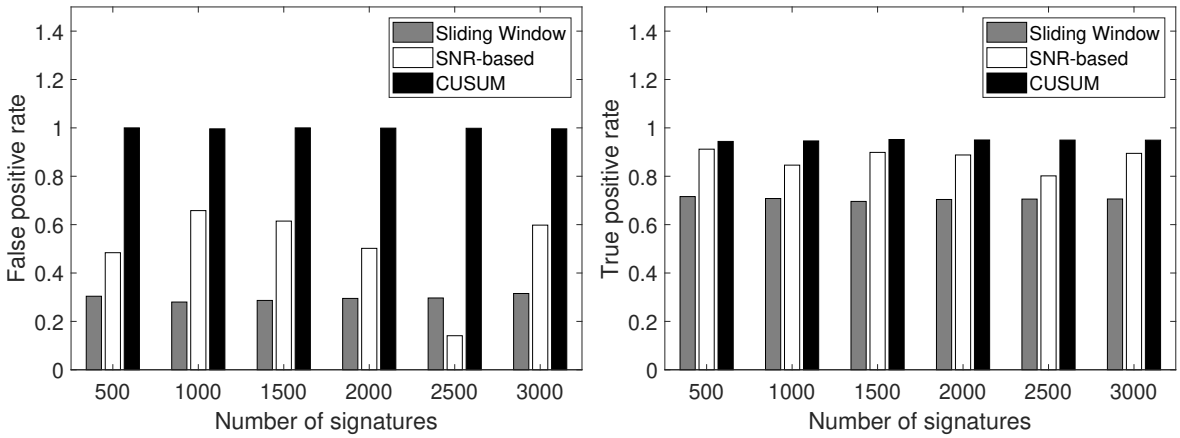
Variable Name	Values
Total provisioning period	360 days
Trial length of each consumer	30 days
Total number of IaaS performance signatures	5
Total number of Consumers	18
Similarity thresholds	.6 to 0.9
Anomaly Thresholds	60% to 90%

We chose to utilize additive white Gaussian noise as it a commonly used noise model used in information theory to mimic the effect of many random processes that occur in nature. We have created new signatures to simulate the change in IaaS signatures from the signature of five IaaS providers. To create a new signature, first we selected a random index in one of the IaaS signatures. This index is called change index. We then computed the moving average for each 10 data points in the signature starting from the change index to the rest of the signature. The original signature is then altered by replacing with the moving average values from the change index to the end of the signature. The experimental settings are shown in Table 6.3."

6.4.4.2 Evaluation and Discussion

The aim of the experiments is to evaluate the proposed approaches in terms of their ability to differentiate between noise and true performance changes. Therefore, we consider four attributes: 1) True Positives (TP) (indicating correct change detection), 2) False Positives (FP) (indicating incorrect change detection), 3) True Negatives (TN) (indicating correct noise detection), and 4) False Negatives (FN) (indicating incorrect noise detection). First, we look at the false positive rate and the true positive rate of the proposed approaches. The false positive rate (also known as false alarm ratio) indicates the expectancy of the false positive ratio and is calculated as follows:

$$\text{False Positive Rate (FP rate)} = \frac{FP}{FP + TN} \tag{6.12}$$



(a) (b)

FIGURE 6.13: Experiment results (a) false positive rate (b) true positive rate

The lower value of false positive rate indicates better performance. Similarly, true positive rate (also known as recall or sensitivity) refers to the ability to identify correct changes and is calculated as follows:

$$\text{True Positive Rate (TP rate)} = \frac{TP}{TP + FN} \tag{6.13}$$

Figure 6.13(a) shows the FP rate of the three different change detection approaches. The experiment is run with five different sample sizes of IaaS signatures. The X axis indicates the sample size of each iteration. The Y axis indicates the false positive rates. Figure 6.13(a) shows that the sliding window approach has the lowest average FP rate compared to the other two approaches (around 0.3 in most cases). The SNR-based approach has a higher FP rate than the sliding window approach and a lower FP rate than the CUSUM approach. The CUSUM approach has the highest false positive rate. Similarly, Figure 6.13(b) shows the TP rate for the three approaches. The sliding window has the lowest TP rate and the CUSUM has the highest TP rate. The SNR-based approach provides a TP rate higher than the sliding window and lower than the CUSUM. These results indicate that the CUSUM is unable to distinguish between noise and actual change in IaaS performance. Therefore, it considers all types of noise and change as true change. As a result, it has the highest FP rate and TP rate. The sliding window-based approach offers the lowest FP rate. However, its ability to detect

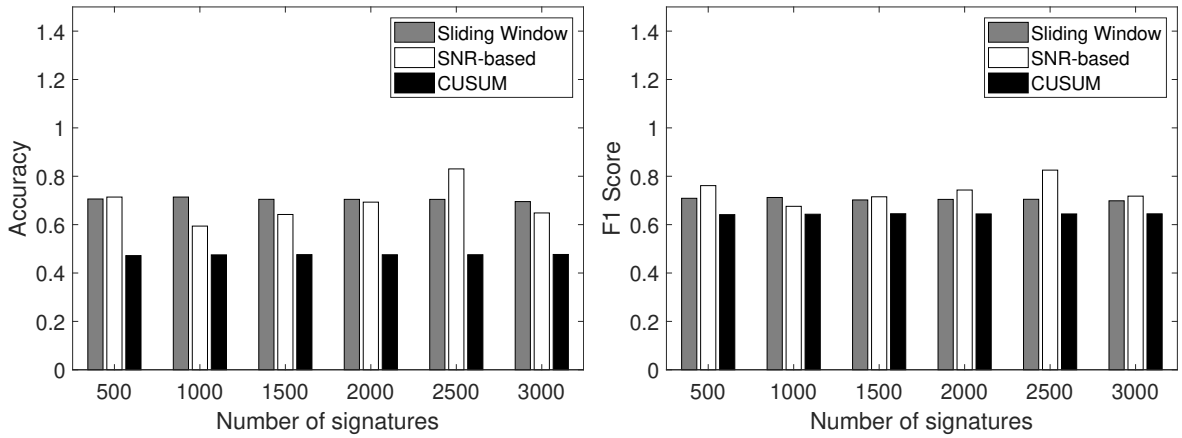


FIGURE 6.14: Experiment results (a) accuracy (b) $F1$ score

true changes is also the lowest compare to the other approaches. The SNR-based approach exhibits balanced FP and TP rate compared to the other two approaches. As a result, the SNR-based approach can be considered to have a better ability to distinguish between the noise and changes in IaaS performance. Now, we measure the accuracy and the $F1$ score of these approaches to further evaluate the results. Accuracy and $F1$ score are computed as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.14}$$

$$F1 \text{ score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6.15}$$

Accuracy refers to the ability to correctly detect changes compared to the total number of observations. Figure 6.14(a) shows the accuracy of the three approaches. It shows that the sliding window has the highest accuracy most of the time and CUSUM has the lowest accuracy. The SNR-based approach has an accuracy higher than the CUSUM and lower than the sliding window approach. From the accuracy results, the sliding window approach appears to be the best-performing approach. However, the accuracy metric for change detection is not always suitable, because a large number of true negatives in the data can cause bias. In

such a case, the $F1$ score is a better indicator of performance measure. $F1$ score takes both false positives and false negatives into account and is, therefore, a better indicator for uneven class distribution of data. Figure 6.14(b) shows the $F1$ score of the three approaches. The CUSUM has the lowest $F1$ score and the SNR-based approach has the highest $F1$ score. The sliding window has a lower $F1$ score compared to the SNR-based approach. This indicates that the SNR-based approach has a better ability to distinguish between the noise and changes in IaaS performance compared to the other two approaches. This conclusion is the same as the conclusion derived from the FP rate and the TP rate from Figure 6.13.

6.5 Summary

Detecting changes in long-term IaaS performance is important as it will help new consumers select the best services according to their long-term QoS requirements. In this chapter, we propose a set of approaches to detect changes in IaaS performance as represented by its signature. In this case, the IaaS performance signature may need to be updated to be representative of the new performance profile of the service. First, we introduce a novel ECA approach to detect changes in IaaS performance, which would warrant changes in the corresponding IaaS signature. The proposed approach relies on detecting anomalous performance behavior from the experience of free trial users to detect changes in IaaS performance. A novel anomaly-based event detection technique is proposed to determine when to trigger the re-evaluation of IaaS signatures. We then introduce a categorical signature-based approach to detect performance noise. Next, a novel IaaS performance model is introduced to identify changes in IaaS performance. Finally, we propose a novel signature change detection framework that utilizes sliding window-based and SNR-based techniques to identity noise and changes in IaaS performance.

CHAPTER 7

Conclusion and Future Work

IaaS cloud is one of the most popular forms of cloud computing as it provides cloud consumers with a range of benefits such as fast migration, low maintenance cost for IT infrastructure, and high scalability. Most business organizations prefer to migrate and manage their in-house infrastructure in the IaaS cloud. There are two types of subscriptions in the cloud: a) pay- as-you-go, and b) reservation. Large organizations tend to utilize IaaS cloud services on a reservation basis for a long-term period, e.g., one to three years. Subscribing to an IaaS service for a long-term period is an important business decision for IaaS consumers. Selecting a service that may perform poorly in the future may incur a substantial loss for an organization. IaaS consumers are provided with a free short-term trial to facilitate a long-term decision. However, selecting a service for a long-term period based on only a short-term trial is challenging. In this regard, this thesis addresses the key challenges in selecting an IaaS service for a long-term period based on short-term trials considering a consumer's long-term performance requirements. The key contributions of this research are presented in two parts. In the first part, we propose a set of long-term selection frameworks to help a new consumer in long-term selection. In the selection, we address three key challenges of the long-term selection: a) candidate service provider selection for the free trials, b) effective utilization of free trials, and c) long-term performance variability. In the second part, we focus on the long-term performance change detection as this element may affect the selection considerably. In this context, we address the key challenges in long-term performance change detection: a) change point detection, b) change detection, and c) performance noise identification.

Chapter 3 proposes a long-term IaaS provider selection framework to select the closest-matched IaaS provider according to a consumer's long-term requirements. The short-term

trial periods offered by the IaaS providers are leveraged to discover the providers' unknown QoS performance. We proposed a temporal skyline-based filtering method to limit the number of candidate IaaS providers for the trial periods. Then, we considered two possible cases: a) experience of free trial users is available, and b) experience of free trial users is unavailable. In the first case, we proposed a co-operative long-term QoS prediction (CLQP) approach to discover a service's long-term performance. The proposed CLQP approach utilizes trial users' experiences to predict a provider's performance for the consumer's long-term workloads with a confidence measure. Experimental results show that the CLQP approach can effectively measure QoS performance. In the second case, a long-term QoS prediction approach without history is proposed that utilizes a set of trial workload generation approaches. Experimental results show that the LQP-short can effectively predict QoS performance with acceptable precision. Finally, a QoS-aware selection method is proposed to select the closest match provider where the provider's expected performance closely matches the consumer's long-term requirements.

In Chapter 4, we approached the long-term selection from a different perspective, where we leveraged the concept of performance fingerprints. The performance fingerprint of an IaaS service represents a service's expected performance behavior over a long-term period. Here, we assume that the performance fingerprint of a service is known during the selection. The performance fingerprint of a service is incorporated with a consumer's trial experience for the long-term selection. A novel trial strategy using the equivalence partitioning method is proposed to estimate a service's performance for different types of workloads while considering the service's performance variability. A performance fingerprint-matching technique is proposed to ascertain the confidence of the consumer's trial experience. A trial experience transformation method is proposed to improve the confidence of the consumer's trial experience.

In Chapter 5, we extended the concept of performance fingerprints into performance signatures. Here, we assumed that free trial users may not share their experiences publicly to protect their privacy. Therefore, free trial users share their experience to a trusted non-profit organization that generates and provides the performance signature of a service to potential consumers. A

Signature-based QoS Performance Discovery (SPD) algorithm is proposed, which leverages the combination of free trials and IaaS signatures. A new significance-based trial scheme is proposed using frequency distribution analysis to test a consumer's long-term workloads in a short-term trial.

In Chapter 6, we introduced a novel ECA approach to detect changes in IaaS performance, which would warrant changes in the corresponding IaaS signature. The proposed approach relies on the detection of anomalous performance behavior from the experience of free trial users to detect changes in IaaS performance. A novel anomaly-based event detection technique is proposed to determine when to trigger the re-evaluation of IaaS signatures. The experiment results show that the proposed approach is able to accurately detect changes in IaaS performance that warrant re-evaluation of the corresponding IaaS signature. We then introduced a more advanced performance change detection approach. In this approach, we introduced the concept of categorical and general IaaS performance signature to define performance noise more accurately. We introduced a new IaaS performance noise model to identify performance change accurately. Finally, we introduced a signature change detection framework approach that considers performance noise during the change detection. The key challenge in performance change detection is to differentiate between noise and changes in IaaS performance. We defined three types of noise in IaaS performance behavior, i.e., spikes, attenuation, and distortion. We utilized time series similarity measuring techniques and a sliding window technique to identify noise in IaaS performance. We proposed an SNR-based approach to improve the performance of change detection. In this approach, we utilized the Signal-to-Noise Ratio to measure noise levels in IaaS performance to detect changes.

7.1 Discussion

This research focuses on the long-term selection of IaaS cloud services given a consumer's long-term QoS requirements using performance discovery. A set of trial strategies is proposed to effectively utilize free trial periods based on the consumer's long-term workloads. The concept of an IaaS performance signature is proposed to deal with long-term performance

variability for the long-term selection. The long-term performance behavior of a service may change over time. In this case, the performance signature may need to be updated over time to reflect the current performance behavior of an IaaS service. We proposed a set of approaches to detect changes in the IaaS performance behavior of an IaaS service. Our research shows that relying only on IaaS advertisements or the trial experience is inadequate for the long-term selection as it often leads to making an incorrect choice. Experimental results show that selecting appropriate trial workloads has a substantial impact on the long-term selection. The result also shows that the long-term selection can be improved significantly by utilizing the proposed signature-based selection approach. Although this research helps a consumer in making an informed long-term selection, there are several limitations that may open up new research directions. We briefly discuss the key limitations and future work below.

7.2 Limitations

We assumed that a consumer's long-term workloads are deterministic during the long-term selection, i,e., a consumer knows how they will utilize the service over long-term period. However, a consumer's service usage pattern may change over period of time. In such case, deterministic workload model may not lead to the optimal selection. We also assumed that the required service for a consumer will be available for free trials. Although this assumption is realistic as provider typically would allow free trial for most services, however, if the service is not available for free trial, the proposed selection frameworks would not be able to perform the selection.

We proposed an equivalence strategy-based trial strategy where multiple VMs are utilized to evaluate service performance. The accuracy of the performance discovery may depend on the number of available VMs. When only one VM is available for the free trial, discovering service performance could be more challenging in this context.

We introduced a signature-based IaaS cloud service selection framework where the accuracy of the signature depends on the number of consumers that share their trial experience to

a trusted third party. Therefore, if most users are reluctant to share their experience, the accuracy of the signature will degrade significantly.

A key limitation of this work is the use of synthetic datasets in the experiment. We have used publicly available short-term datasets to conduct the experiments due to the unavailability of suitable long-term workload-performance datasets. However, the synthetic datasets are created by augmenting real-world datasets to capture the characteristics of real cloud environment. Therefore, we believe the result would be similar if the experiments were conducted directly using real datasets.

7.3 Future Work

The limitations of this work could lead to several new research directions. To overcome the limitation of deterministic workloads, we investigate how to develop a dynamic workload prediction model that would help a consumer to perform long-term selection. This model could be build up by utilizing the past service usage pattern of a consumer over a long period. When a particular service is unavailable for the free trial, we may investigate the use of available services to predict the performance of the required service. In this case, we may utilize collaborative QoS recommendation techniques to predict service performance.

The proposed equivalence trial strategy in chapter 4 would mainly work if there are multiple VMs are available for the free trial. However, a possible research direction could be developing more advance trial strategy that would work even if only one VM is available. In this case, we may need to consider partitioning the available time in a way that would allow us to simulate multiple VM environment.

We may investigate how to develop an incentive model that would encourage free trial users to share their experience publicly. It will lead to improve the accuracy of the signature. In chapter 6, we proposed an event-based IaaS signature change detection framework. The proposed framework aims at detecting long-term changes in IaaS performance of an IaaS service and updating its performance signature accordingly. The proposed framework assumes

that the signature of a service is generated accurately prior to the change detection. However, if the accuracy of the signature is lower, the performance of the proposed change detection may degrade. In this context, a new change detection model could be developed that considers the accuracy of the signature during the change detection.

We then introduced a new type of IaaS performance signature called categorical IaaS signature. The categories are defined based on the resource requests of the consumer's workload such as CPU-intensive, I/O-intensive, and network-intensive. However, it is possible to categorize the consumer workloads based on workload distributions such as uniform, heavy- tailed, and self-similar. In this context, the proposed categorical signature generation technique may be extended to capture the effect of the workload distribution of the performance. Finally, we introduced an IaaS signature detection framework where performance noise is considered during the change detection. The proposed framework primarily focuses on three types of noise during the change detection. A possible extension could comprise development of a new framework that can identify other performance noises.

A possible extension of this research is to extend the application of IaaS performance signatures. For example, the performance signature of an IaaS service could be utilized to detect long-term performance anomalies. Another possible extension of this research is to consider the long-term IaaS cloud service composition, where a consumer may need to select multiple IaaS cloud services for the long-term selection.

Bibliography

[Al-Faifi *et al*.2018] Abdullah Mohammed Al-Faifi, Biao Song, Mohammad Mehedi Hassan, Atif Alamri, and Abdu Gumaei. 2018. Performance prediction model for cloud service selection from smart data. *Future Generation Computer Systems*, 85:97–106.

[Alam *et al*.2016] Mansaf Alam, Kashish Ara Shakil, and Shuchi Sethi. 2016. Analysis and clustering of workload in google cluster trace based on resource usage. In *IEEE Intl Conference on Computational Science and Engineering and IEEE Intl Conference on Embedded and Ubiquitous Computing and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering*, pages 740–747. IEEE.

[Aminikhanghahi and Cook2017] Samaneh Aminikhanghahi and Diane J Cook. 2017. A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367.

[Armbrust *et al*.2009] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. 2009. Above the clouds: A berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

[Armbrust *et al*.2010] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2010. A view of cloud computing. *Communications of the ACM*, 53(4):50–58.

[Avula and Zou2020] Raghu Nandan Avula and Cliff Zou. 2020. Performance evaluation of tpc-c benchmark on various cloud providers. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0226–0233. IEEE.

[Ba-Hutair and Kamel2016] Mohammed N Ba-Hutair and Ibrahim Kamel. 2016. A new scheme for protecting the privacy and integrity of spatial data on the cloud. In *IEEE Second International Conference on Multimedia Big Data*, pages 394–397. IEEE.

[Bahga and Madisetti2011] Arshdeep Bahga and Vijay Krishna Madisetti. 2011. Synthetic workload generation for cloud computing applications. *Journal of Software Engineering and Applications*, 4(07):396.

[Barford and Crovella1998] Paul Barford and Mark Crovella. 1998. Generating representative web workloads for network and server performance evaluation. In *ACM SIGMETRICS Performance Evaluation Review*, volume 26, pages 151–160. ACM.

[Baset *et al*.2017] Salman Baset, Marcio Silva, and Nicholas Wakou. 2017. Spec cloud™ iaas 2016 benchmark. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering*, pages 423–423.

[Binnig *et al*.2009] Carsten Binnig, Donald Kossmann, Tim Kraska, and Simon Loesing. 2009. How is the weather tomorrow? towards a benchmark for the cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*, pages 1–6.

[Bouguettaya *et al*.2010] Athman Bouguettaya, Surya Nepal, Wanita Sherchan, Xuan Zhou, Jemma Wu, Shiping Chen, Dongxi Liu, Lily Li, Hongbing Wang, and Xumin Liu. 2010. End-to-end service support for mashups. *IEEE Transactions on Services Computing*, 3(3):250–263.

[Bouguettaya *et al*.2017] Athman Bouguettaya, Munindar Singh, Michael Huhns, Quan Z Sheng, Hai Dong, Qi Yu, Azadeh Ghari Neiat, Sajib Mistry, Boualem Benatallah, Brahim Medjahed, et al. 2017. A service computing manifesto: the next 10 years. *Communications of the ACM*, 60(4):64–72.

[Boutilier *et al*.2004] Craig Boutilier, Ronen I Brafman, Carmel Domshlak, Holger H Hoos, and David Poole. 2004. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191.

[Burtini *et al*.2013] Giuseppe Burtini, Scott Fazackerley, and Ramon Lawrence. 2013. Time series compression for adaptive chart generation. In *26th IEEE Canadian Conference on Electrical and Computer Engineering*, pages 1–6. IEEE.

[Calzarossa *et al*.2000] Maria Calzarossa, Luisa Massari, and Daniele Tessera. 2000. Workload characterization issues and methodologies. In *Performance Evaluation: Origins and Directions*, pages 459–482. Springer.

[Calzarossa *et al*.2016a] Maria Carla Calzarossa, Marco L Della Vedova, Luisa Massari, Dana Petcu, Momin IM Tabash, and Daniele Tessera. 2016a. Workloads in the clouds. In *Principles of Performance and Reliability Modeling and Evaluation*, pages 525–550. Springer.

[Calzarossa *et al*.2016b] Maria Carla Calzarossa, Luisa Massari, and Daniele Tessera. 2016b. Workload characterization: A survey revisited. *ACM Computing Surveys*, 48(3):48.

[Chaisiri *et al*.2012] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. 2012. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5(2):164–177.

[Dejun *et al*.2010] Jiang Dejun, Guillaume Pierre, and Chi-Hung Chi. 2010. Ec2 performance analysis for resource provisioning of service-oriented applications. In *International Conference on Service-Oriented Computing*, pages 197–207. Springer.

[Doelitzscher *et al*.2013] Frank Doelitzscher, Martin Knahl, Christoph Reich, and Nathan Clarke. 2013. Anomaly detection in iaas clouds. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 387–394. IEEE.

[Dou *et al*.2013] Wanchun Dou, Xuyun Zhang, Jianxun Liu, and Jinjun Chen. 2013. Hiresome-ii: Towards privacy-aware cross-cloud service composition for big data applications. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):455–466.

[Fattah and Bouguettaya2020a] Sheik Mohammad Mostakim Fattah and Athman Bouguettaya. 2020a. Event-based detection of changes in iaas performance signatures. In *IEEE International Conference on Services Computing*, pages 210–217. IEEE.

[Fattah and Bouguettaya2020b] Sheik Mohammad Mostakim Fattah and Athman Bouguettaya. 2020b. Event-based detection of changes in iaas performance signatures. In *IEEE International Conference on Services Computing*, pages 210–217. IEEE.

[Fattah *et al*.2018] Sheik Mohammad Mostakim Fattah, Athman Bouguettaya, and Sajib Mistry. 2018. A cp-net based qualitative composition approach for an iaas provider. In *International Conference on Web Information Systems Engineering*, pages 151–166. Springer.

[Fattah *et al*.2019] Sheik Mohammad Mostakim Fattah, Athman Bouguettaya, and Sajib Mistry. 2019. Long-term iaas provider selection using short-term trial experience. In *IEEE International Conference on Web Services*, pages 304–311. IEEE.

[Fattah *et al*.2020a] Sheik Mohammad Mostakim Fattah, Athman Bouguettaya, and Sajib Mistry. 2020a. Long-term iaas selection using performance discovery. *IEEE Transactions on Services Computing*.

[Fattah *et al*.2020b] Sheik Mohammad Mostakim Fattah, Athman Bouguettaya, and Sajib Mistry. 2020b. Signature-based selection of iaas cloud services. In *IEEE International Conference on Web Services*, pages 50–57. IEEE.

[Feitelson2002] Dror G Feitelson. 2002. Workload modeling for performance evaluation. In *IFIP International Symposium on Computer Performance Modeling, Measurement and Evaluation*, pages 114–141. Springer.

[Hu *et al*.2014] Yan Hu, Qimin Peng, Xiaohui Hu, and Rong Yang. 2014. Time aware and data sparsity tolerant web service recommendation based on improved collaborative filtering. *IEEE Transactions on Services Computing*, 8(5):782–794.

[Huang *et al*.2016] Tian Huang, Yongxin Zhu, Yafei Wu, Stéphane Bressan, and Gillian Dobbie. 2016. Anomaly detection and identification scheme for vm live migration in cloud

infrastructure. *Future Generation Computer Systems*, 56:736–745.

[Hwang *et al.*2015] Kai Hwang, Xiaoying Bai, Yue Shi, Muyang Li, Wen-Guang Chen, and Yongwei Wu. 2015. Cloud performance modeling with benchmark evaluation of elastic scaling strategies. *IEEE Transactions on parallel and distributed systems*, 27(1):130–143.

[Ibidunmoye *et al.*2015] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. 2015. Performance anomaly detection and bottleneck identification. *ACM Computing Survey*, 48(1):1–35.

[Iosup *et al.*2011] Alexandru Iosup, Nezih Yigitbasi, and Dick Epema. 2011. On the performance variability of production cloud services. In *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 104–113. IEEE.

[Iosup *et al.*2014] Alexandru Iosup, Radu Prodan, and Dick Epema. 2014. Iaas cloud benchmarking: approaches, challenges, and experience. In *Cloud Computing for Data-Intensive Applications*, pages 83–104. Springer.

[Jayasinghe *et al.*2012] Deepal Jayasinghe, Galen Swint, Simon Malkowski, Jack Li, Qingyang Wang, Junhee Park, and Calton Pu. 2012. Expertus: A generator approach to automate performance testing in iaas clouds. In *IEEE International Conference on Cloud Computing*, pages 115–122. IEEE.

[Jiang and Pei2009] Bin Jiang and Jian Pei. 2009. Online interval skyline queries on time series. In *IEEE 25th International Conference on Data Engineering*, pages 1036–1047. IEEE.

[Kalayappan *et al.*2020] Rajshekar Kalayappan, Avantika Chhabra, and Smruti R Sarangi. 2020. Chunkedtejas: A chunking-based approach to parallelizing a trace-driven architectural simulator. *ACM Transactions on Modeling and Computer Simulation*, 30(3):1–21.

[Kant and Won2000] Krishna Kant and Youjip Won. 2000. Performance impact of uncached file accesses in specweb99. In *Workload Characterization for Computer System Design*, pages 87–104. Springer.

[Kessler *et al.*1994] Richard E. Kessler, Mark D Hill, and David A Wood. 1994. A comparison of trace-sampling techniques for multi-megabyte caches. *IEEE Transactions on Computers*, 43(6):664–675.

[Khan *et al.*2012] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. 2012. Workload characterization and prediction in the cloud: A multiple time series approach. In *IEEE Network Operations and Management Symposium*, pages 1287–1294. IEEE.

[Labbaci *et al.*2017] Hamza Labbaci, Brahim Medjahed, and Youcef Aklouf. 2017. A deep learning approach for long term qos-compliant service composition. In *International Conference on Service-Oriented Computing*, pages 287–294. Springer.

[Langer and French2011] Steve G Langer and Todd French. 2011. Virtual machine performance benchmarking. *Journal of digital imaging*, 24(5):883–889.

[Leitner and Cito2016] Philipp Leitner and Jürgen Cito. 2016. Patterns in the chaos—a study of performance variation and predictability in public iaas clouds. *ACM Transactions on Internet Technology*, 16(3):1–23.

[Li *et al*.2010] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. 2010. Cloudcmp: comparing public cloud providers. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14. ACM.

[Liu and Özsu2009] Ling Liu and M Tamer Özsu. 2009. *Encyclopedia of database systems*, volume 6. Springer New York, NY, USA:.

[Liu *et al*.2015] Shengcai Liu, Yufan Wei, Ke Tang, A Kai Qin, and Xin Yao. 2015. Qos-aware long-term based service composition in cloud computing. In *IEEE Congress on Evolutionary Computation*, pages 3362–3369. IEEE.

[Mackiewicz and Ratajczak1993] Andrzej Mackiewicz and Waldemar Ratajczak. 1993. Principal components analysis (pca). *Computers and Geosciences*, 19:303–342.

[Mahambre *et al*.2012] Shruti Mahambre, Purushottam Kulkarni, Umesh Bellur, Girish Chafle, and Deepak Deshpande. 2012. Workload characterization for capacity planning and performance management in iaas cloud. In *IEEE International Conference on Cloud Computing in Emerging Markets*, pages 1–7. IEEE.

[Mazzucco and Dumas2011] Michele Mazzucco and Marlon Dumas. 2011. Reserved or on-demand instances? a revenue maximization model for cloud providers. In *IEEE International Conference on Cloud Computing*, pages 428–435. IEEE.

[Mi *et al*.2008] Ningfang Mi, Ludmila Cherkasova, Kivanc Ozonat, Julie Symons, and Evgenia Smirni. 2008. Analysis of application performance and its change via representative application signatures. In *IEEE Network Operations and Management Symposium*, pages 216–223. IEEE.

[Mistry *et al*.2015] Sajib Mistry, Athman Bouguettaya, Hai Dong, and A Kai Qin. 2015. Predicting dynamic requests behavior in long-term iaas service composition. In *IEEE International Conference on Web Services*, pages 49–56. IEEE.

[Mistry *et al*.2016a] Sajib Mistry, Athman Bouguettaya, Hai Dong, and Abdelkarim Erradi. 2016a. Qualitative economic model for long-term iaas composition. In *International Conference on Service-Oriented Computing*, pages 317–332. Springer.

[Mistry *et al*.2016b] Sajib Mistry, Athman Bouguettaya, Hai Dong, and Alex Kai Qin. 2016b. Metaheuristic optimization for long-term iaas service composition. *IEEE Transactions on Services Computing*, 11(1):131–143.

[Moens and Zénon2019] Vincent Moens and Alexandre Zénon. 2019. Learning and forgetting using reinforced bayesian change detection. *PLoS computational biology*, 15(4):e1006713.

[Nurmi *et al*.2009] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, and Dmitrii Zagorodnov. 2009. The eucalyptus open-source cloud-computing system. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 124–131. IEEE.

[Ostermann *et al*.2009] Simon Ostermann, Alexandria Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. 2009. A performance analysis of ec2 cloud computing services for scientific computing. In *IEEE International Conference on Cloud Computing*, pages 115–131. Springer.

[Pacheco-Sanchez *et al*.2011] Sergio Pacheco-Sanchez, Giuliano Casale, Bryan Scotney, Sally McClean, Gerard Parr, and Stephen Dawson. 2011. Markovian workload characterization for qos prediction in the cloud. In *IEEE International Conference on Cloud Computing*, pages 147–154. IEEE.

[Persico *et al*.2015] Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescapè. 2015. Measuring network throughput in the cloud: The case of amazon ec2. *Computer Networks*, 93:408–422.

[Polemi1998] Despina Polemi. 1998. Trusted third party services for health care in europe. *Future Generation Computer Systems*, 14(1-2):51–59.

[Pucher *et al*.2015] Alexander Pucher, Emre Gul, Rich Wolski, and Chandra Krintz. 2015. Using trustworthy simulation to engineer cloud schedulers. In *IC2E*, pages 256–265. IEEE.

[Pucher2016] Alexander Ernst Pucher. 2016. *Using Workload Prediction and Federation to Increase Cloud Utilization*. Ph.D. thesis, UC Santa Barbara.

[Qi *et al*.2019] Lianyong Qi, Ruili Wang, Chunhua Hu, Shancang Li, Qiang He, and Xiaolong Xu. 2019. Time-aware distributed service recommendation with privacy-preservation. *Information Sciences*, 480:354–364.

[Reiss *et al*.2012] Charles Reiss, Alexey Tumanov, Gregory R Ganger, Randy H Katz, and Michael A Kozuch. 2012. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *Proceedings of the Third ACM Symposium on Cloud Computing*, page 7. ACM.

[Sadooghi *et al*.2015] Iman Sadooghi, Jesús Hernández Martin, Tonglin Li, Kevin Brandstatter, Ketan Maheshwari, Tiago Pais Pitta de Lacerda Ruivo, Gabriele Garzoglio, Steven Timm, Yong Zhao, and Ioan Raicu. 2015. Understanding the performance and potential of cloud computing for scientific applications. *IEEE Transactions on Cloud Computing*, 5(2):358–371.

[Sadooghi *et al.*2017] Iman Sadooghi, Jesús Hernández Martin, Tonglin Li, Kevin Brandstatter, Ketan Maheshwari, Tiago Pais Pitta de Lacerda Ruivo, Gabriele Garzoglio, Steven Timm, Yong Zhao, and Ioan Raicu. 2017. Understanding the performance and potential of cloud computing for scientific applications. *IEEE Transactions on Cloud Computing*, 5(2):358–371.

[Schad *et al.*2010] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. 2010. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment*, 3(1-2):460–471.

[Scheuner and Leitner2018a] Joel Scheuner and Philipp Leitner. 2018a. A cloud benchmark suite combining micro and applications benchmarks. In *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, pages 161–166. ACM.

[Scheuner and Leitner2018b] Joel Scheuner and Philipp Leitner. 2018b. Estimating cloud application performance based on micro-benchmark profiling. In *IEEE International Conference on Cloud Computing*, pages 90–97. IEEE.

[Scheuner and Leitner2019] Joel Scheuner and Philipp Leitner. 2019. Performance benchmarking of infrastructure-as-a-service (iaas) clouds with cloud workbench. In *Companion of the 2019 ACM/SPEC International Conference on Performance Engineering*, pages 53–56.

[Swan2012] Melanie Swan. 2012. Crowdsourced health research studies: an important emerging complement to clinical trials in the public health research ecosystem. *Journal of medical Internet research*, 14(2):e46.

[Takeda2012] Keiji Takeda. 2012. User identification and tracking with online device fingerprints fusion. In *IEEE International Carnahan Conference on Security Technolog*, pages 163–167. IEEE.

[Tang *et al.*2016] Mingdong Tang, Zibin Zheng, Guosheng Kang, Jianxun Liu, Yatao Yang, and Tingting Zhang. 2016. Collaborative web service quality prediction via exploiting matrix factorization and network map. *IEEE Transactions on Network and Service Management*, 13(1):126–137.

[ur Rehman *et al.*2012] Zia ur Rehman, Omar K Hussain, and Farookh K Hussain. 2012. Iaas cloud selection using mcdm methods. In *IEEE Ninth International Conference on e-Business Engineering*, pages 246–251. IEEE.

[van den Braak *et al.*2012] Susan W van den Braak, Sunil Choenni, Ronald Meijer, and Anneke Zuiderwijk. 2012. Trusted third parties for secure and privacy-preserving data integration and sharing in the public sector. In *Proceedings of the 13th Annual International Conference on Digital Government Research*, pages 135–144. ACM.

[Varadarajan *et al.*2012] Venkatanathan Varadarajan, Thawan Kooburat, Benjamin Farley, Thomas Ristenpart, and Michael M Swift. 2012. Resource-freeing attacks: improve

your cloud performance (at your neighbor's expense). In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 281–292. ACM.

[Veeravalli and Banerjee2014] Venugopal V Veeravalli and Taposh Banerjee. 2014. Quickest change detection. In *Academic Press Library in Signal Processing*, volume 3, pages 209–255. Elsevier.

[Wang *et al*.2009] Hongbing Wang, Shizhi Shao, Xuan Zhou, Cheng Wan, and Athman Bouguettaya. 2009. Web service selection with incomplete or inconsistent user preferences. In *International Conference on Service-Oriented Computing*, pages 83–98. Springer.

[Wang *et al*.2013] Hao Wang, Chao-Kun Wang, Ya-Jun Xu, and Yuan-Chi Ning. 2013. Dominant skyline query processing over multiple time series. *Journal of Computer Science and Technology*, 28(4):625–635.

[Wang *et al*.2014] Xiajun Wang, Song Huang, Song Fu, and Krishna Kavi. 2014. Characterizing workload of web applications on virtualized servers. In *Workshop on Big Data Benchmarks, Performance Optimization, and Emerging Hardware*, pages 98–108. Springer.

[Wang *et al*.2017] Hongbing Wang, Hualan Wang, Guibing Guo, Yangyu Tang, and Jie Zhang. 2017. Measuring similarity of users with qualitative preferences for service selection. *Knowledge and Information Systems*, 51(2):561–594.

[Wang *et al*.2018] Wei Wang, Ningjing Tian, Sunzhou Huang, Sen He, Abhijeet Srivastava, Mary Lou Soffa, and Lori Pollock. 2018. Testing cloud applications under cloud-uncertainty performance effects. In *IEEE 11th International Conference on Software Testing, Verification and Validation*, pages 81–92. IEEE.

[Wang *et al*.2019] Shangguang Wang, Yali Zhao, Lin Huang, Jinliang Xu, and Ching-Hsien Hsu. 2019. Qos prediction for service recommendations in mobile edge computing. *Journal of Parallel and Distributed Computing*, 127:134–144.

[Wenmin *et al*.2011] Lin Wenmin, Dou Wanchun, Luo Xiangfeng, and Jinjun Chen. 2011. A history record-based service optimization method for qos-aware service composition. In *IEEE International Conference on Web Services*, pages 666–673. IEEE.

[Whalen2013] Anthony D Whalen. 2013. *Detection of signals in noise*. Academic press.

[Wiener and of Technology (Cambridge1950] Norbert Wiener and Mass.) Massachusetts Institute of Technology (Cambridge. 1950. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Technology Press.

[Wolski and Brevik2017] Rich Wolski and John Brevik. 2017. Qpred: Using quantile predictions to improve power usage for private clouds. In *IEEE International Conference on Cloud Computing*, pages 179–187. IEEE.

[Yang *et al*.2018] Yatao Yang, Zibin Zheng, Xiangdong Niu, Mingdong Tang, Yutong Lu, and Xiangke Liao. 2018. A location-based factorization machine model for web service

qos prediction. *IEEE Transactions on Services Computing*.

[Ye *et al*.2011] Zhen Ye, Xiaofang Zhou, and Athman Bouguettaya. 2011. Genetic algorithm based qos-aware service compositions in cloud computing. In *International Conference on Database Systems for Advanced Applications*, pages 321–334. Springer.

[Ye *et al*.2012] Zhen Ye, Athman Bouguettaya, and Xiaofang Zhou. 2012. Qos-aware cloud service composition based on economic models. In *International Conference on Service-Oriented Computing*, pages 111–126. Springer.

[Ye *et al*.2016] Zhen Ye, Sajib Mistry, Athman Bouguettaya, and Hai Dong. 2016. Long-term qos-aware cloud service composition using multivariate time series analysis. *IEEE Transactions on Services Computing*, 9(3):382–393.

[Ye2017] Kejiang Ye. 2017. Anomaly detection in clouds: Challenges and practice. In *the first Workshop on Emerging Technologies for software-defined and reconfigurable hardware-accelerated Cloud Datacenters*, page 6. ACM.

[Yu and Bouguettaya2010] Qi Yu and Athman Bouguettaya. 2010. Computing service skylines over sets of services. In *IEEE International Conference on Web Services*, pages 481–488. IEEE.

[Zadeh and Seyyedi2010] Mahmod Hosein Zadeh and Mir Ali Seyyedi. 2010. Qos monitoring for web services by time series forecasting. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 5, pages 659–663. IEEE.

[Zeng *et al*.2004] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. 2004. Qos-aware middleware for web services composition. *IEEE Transactions on Software Engineering*, 30(5):311–327.

[Zheng *et al*.2009] Huiyuan Zheng, Weiliang Zhao, Jian Yang, and Athman Bouguettaya. 2009. Qos analysis for web service composition. In *IEEE International Conference on Services Computing*, pages 235–242. IEEE.

[Zheng *et al*.2011] Zibin Zheng, Hao Ma, Michael R Lyu, and Irwin King. 2011. Qos-aware web service recommendation by collaborative filtering. *IEEE Transactions on services computing*, 4(2):140–152.

[Zheng *et al*.2012] Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R Lyu, and Jianmin Wang. 2012. Qos ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1213–1222.

[Zheng *et al*.2013] Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R Lyu, and Jianmin Wang. 2013. Qos ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1213–1222.

[Zhu and Chang2014] Dong Hong Zhu and Ya Ping Chang. 2014. Investigating consumer attitude and intention toward free trials of technology-based services. *Computers in Human*

Behavior, 30:328–334.

APPENDIX A

TABLE A.1: Notations and descriptions

Notation	Description
T	Required provisioning time
W	Consumer workloads in time series
Q_C	Set of QoS requirements of consumers
l	Number of QoS parameters in Q_C
q_{ci}	The time series of the QoS attribute q_{ci}
t_n	A timestamp in T where $n = 1, 2, 3, ...T$
x_n	The value of $q_(ci)$ at the time period t_n
N	The number of IaaS providers who can satisfy the functional requirements of the consumer
P	A set of IaaS providers
P_s	Selected IaaS provider
A_i	The QoS advertisement of the provider P_i
ST_Q	Temporal QoS skyline
MTS_p	MTS-based IaaS skyline
δt	Time interval
S_c	Set of similar trial users
$C^{\delta t}$	Confidence of the QoS prediction
L	Amount of loss of information
$T_{conf}^{Q_i}$	Trial confidence for Q_i
S	IaaS Performance Signature
Q'_{it}	Normalized value of $Q_i(t)$
$RMSE(Q_i^r, Q_i^p)$	Root Mean Squared Error between required QoS (Q_i^r) and predicted QoS (Q_i^p)
$R(P)$	Rank function
$S(E_Q, S_Q)^{ED}$	Euclidean distance between normalized trial experience and signature
$S(E_Q, S_Q)^{PCC}$	Pearson Correlation Coefficients normalized trial experience and signature
$S(E_Q, S_Q)^{CS}$	Cosine distance normalized trial experience and signature
T_s	Similarity threshold
UL_i	Upper limit of cumulative sum
LL_i	Similarity threshold
N	Performance Noise

TABLE A.2: Abbreviations and descriptions

Abbreviations	Description
IaaS	Infrastructure as a Service
VM	Virtual Machine
vCPU	Virtual CPU
CLQP	Cooperative Long-term QoS Prediction
LQP-short	Long-term QoS Prediction without History
QLIS	QoS-aware Long-term IaaS Provider Selection
MAE	Mean Absolute Error
MTS	Multiple Time Series
ECA	Event-Condition-Action
SNR	Signal-to-Noise Ratio
RMSE	Root Mean Squared Error
NRMSE	Normalized Root Mean Squared Error
PCA	Principle Component Analysis
FG	Frequency-based Generation
RG	Resource Consuption-based Generation
MG	Mixed Generation
SPD	Signature-based Performance Discovery
LPD	Long-term Performance Discovery
EQ	Equivalence Partitioning-based Approach
CUSUM	Cumulative Sum Control Chart
DTW	Dynamic Time Warping
SAX	Symbolic Aggregate Approximation
SC	Spearman Correlation (SC)
CS	Cosine Similarity
PCC	Pearson Correlation Coefficient
TP	True Positives
FP	False Positives