Universidad de Guadalajara

Centro Universitario de Ciencias Económico-Administrativas (CUCEA)

Doctorado en Tecnologías de Información

Doctoral Thesis

A Regression Tree Predictive Model for Virtual Machine Startup Time in IaaS clouds

AUTHOR: **Yatheendraprakash Govindaraju**

With specialization in:

Distributed Systems

Review commitee:

Dr. Hector Alejandro Duran Limon	Director
Dr. Efren Mezura-Montes	Co-Director
Dra. Liliana Ibeth Barbosa-Santillán	Reviewer
Dr. Arturo Chavoya Peña	Reviewer
Dr. Mario Siller	Reviewer
Dra. Maria Elena Meda Campana	Reviewer

Zapopan, Jalisco, March 2020

Dedication

To my family

Whose continuous moral support made this possible.

Acknowledgments

It is laborious to mention all the people who made this research work possible by their continuous support during my doctoral program. However, I would like to thank a few of them, without their support, it would not have been possible to finish my dissertation.

I would like to thank my family for their moral and sometimes, financial supports. In addition, I would like to thank my friend Oswaldo Renato Velasco Campos and his family whose moral support made my stay in this country possible. Also, I would like to express my thanks to my friend, Oscar Santana Alvarez for all his personal and professional support.

I would like to express my gratitude to my thesis director, Hector Alejandro Duran Limon, who guided me throughout this thesis, with all my shortcomings.

I wish to acknowledge the help provided by the technical and support staff in the DTI, CUCEA.

To all of you, for your direct or indirect support ... Thank you!

Summary

Cloud computing provides effective ways to rapidly provision computing resources over the Internet. For better management of resource provisioning, the system requires to predict service-level agreements (SLAs) such as virtual machine (VM) startup times under various conditions of computing resources. The VM startup time is an important SLA parameter, which can impact other SLA parameters such as service initiation time and VM scale out times. By predicting VM startup times, Cloud providers can improve Cloud users' expectations. However, little research has been carried out to define approaches to predict VM startup times. In this thesis, we propose a Regression Tree model for predicting average, minimum, and maximum VM startup times. To test the efficiency of our model, we implemented the model in an OpenStack test environment. The test results show that our model predicts VM startup times with an average accuracy of 91.81%.

Contents

List of Tables

List of figures

CHAPTER 1

Introduction and Motivation

1.1 Introduction

Cloud computing provides a convenient means of remote on-demand and pay per use access to computing resources, by extensive use of virtualization. Cloud computing enables the application service providers to lease data center capabilities for deploying applications depending on user Quality of Service (QoS) requirements. The management of quality of service requirements is challenging to the performance and costs of the cloud services. Mainly the IaaS service providers need to maximize the profits and also guarantee QoS requirements of their customers. Hence, one of the prime objectives of Infrastructure as a Service (IaaS) providers is to optimally allocate the resources to improve the utilization, i.e. minimize the cost. Generally, the Quality of service requirements are clearly stated in a Service level agreement (SLA) between the IaaS service providers and their users. If the resources are allocated without considering these SLA parameters, Service providers may not guarantee any QoS for the users. Hence, it is important to understand how the SLA parameters will change for different real-time scenarios of the environment. There are multiple approaches proposed by academic and industry researchers to address these challenges. Basically, all these techniques propose a different framework which are mostly applicable to multiple Virtual Machines (VMs) of single application and/or multi application environments.

Each cloud provider (CP) ensures a certain quality of service (QoS) for cloud users. The performance of cloud service is measured by such QoS guarantees, which are stated

in a SLA . The main challenge for CPs is to ensure the stated QoS guarantees are met within the optimal operating cost of the data centers. Nowadays, the cloud is becoming popular and data centers host VMs of different applications. However, current approaches in IaaS from the cloud provider perspective consider only one SLA parameter, which involves response time[1]. There are other important SLA parameters which are not taken into account such as boot time[2] ,Scale up time[3], and scale down time[4]. Therefore, cloud providers can offer only a poor level of QoS guarantees to the users. It is important that such kinds of parameters are also considered in an SLA so that the cloud users can offer a minimal level of QoS to their end users. Cloud users can improve their revenue by providing a better client satisfaction.

Providers employ different SLA parameters for each type of cloud services. For example, the SLA document of the Amazon EC2 service [1] (part of IaaS products) describes the Monthly uptime SLA parameter, whereas the SLA document of Amazon S3 service (part of SaaS products) includes an Error rate SLA parameter. There are several works that have addressed SLA modeling in cloud computing [2], [3], [4], [5], [6]. Bruneo [3], for example, proposed a stochastic model for analyzing utilization, availability, service time, and waiting time of data centers. Wu et al. [5] proposed an algorithm to schedule customer requests with the aim of minimizing both costs and response time. Lakra et al. [6] propose a multi-objective task scheduling algorithm to guarantee minimum overall execution times. Rao et al. [10] propose a fuzzy controller based on a QoS framework to guarantee response time and throughput. Overall, the service guarantees (modeled as SLA parameters) that are more commonly considered by current research and practice are response time [2], [3], [4], [5], [6] and throughput [2], [6]. The major public cloud providers like Amazon EC2 [1], RackSpace [7], Microsoft Azure [8], and Google Cloud [9] consider as SLA parameters both monthly uptime percentage (which regards availability) and response time. Some other SLA parameters have received less attention such as scale-up time, scale down time, and reliability [10], [11]. However, to the best of our knowledge,

[1]The time taken to complete a user request
[2]The time taken to instantiate a VM
[3]The time taken to increase a specific number of VMs
[4]The time taken to decrese a specific number of VMs

VM startup times guarantees have not been considered so far. The VM startup time is the time taken to boot and run a virtual machine instance. The VM startup time also plays a great impact in technological advancements of other cloud services such as Platform as as Service(PaaS) and Software as a Service(SaaS). As most of the services of PaaS and SaaS are being deployed on top of the Virtual Machines, the estimation of VM startup times may lead to great impact on setting the performance expectations of PaaS and SaaS services too. Though the impact of VM startup times is high on performance metrics of CPs, very few studies have been carried out on this area. A model to predict VM startup times under different resource conditions can address this need of CPs.

1.2 Motivation

This section provides a description of the problem at hand and defines the context domain of the current work. The importance of the given research problem and its consequences are discussed.

1.2.1 Definition of the context domain

Before defining the research problem of this work, it is important to outline the scope of its context domain. In this work, the context domain will be limited to cloud computing environment for Infrastructure service providers and the Service level agreement parameter, virtual machine startup time.

1.2.2 Cloud computing

As stated before, a cloud computing environment provides convenient means to access virtualized computing resources. The virtualization of computing resources and its access are provided at the IaaS level. We can say virtual machines are building blocks of this IaaS service. Sometimes, PaaS and SaaS layers are created on top of this building block i.e. a virtual Machine. For our problem at hand, we will consider only these VMs. Thus, setting the research context limited to the virtual machines.

1.2.3 VM startup time

In the IaaS service level and other services that are built on top of IaaS, the characteristics of VM have a great impact on the performance of that service. Particularly, VM startup time (or some times called boot time) directly impact the SLA parameters such as scale-up times. For example, the scale-up time is the time taken to add the required number of VMs, which is the maximum value of VM startup times of the requested VMs. Similarly, scale-down time is the minimum value of the same set. Hence, our research context is limited to VM startup times, specifically average, minimum and maximum VM startup times. By predicting VM startup times, IaaS cloud end users can calculate how soon their

servers will be ready for use and SaaS providers can calculate service initiation time (i.e. the time taken to initiate a VM and bring up its services online). Currently, public cloud providers do not offer guarantees regarding VM startup times. This is relevant since VM startup times in a public cloud may vary from 44 seconds to 810 seconds, this depending on the cloud provider and operating system of the VM [12]. Not having support for this kind of guarantee can negatively impact the QoS perceived by the cloud end user. For example, consider an e-commerce Web site that receives a sudden and unpredicted workload increase. In this case, it is required to know how long it will take to bring up additional virtual servers to decide whether an adaptation strategy based on either a vertical or a horizontal scale up is more suitable to maintain the cloud SLA. Under normal runtime conditions of any IaaS cloud environment, VM startup time of each VM in a batch of requested VMs varies from one another. For example, in an OpenStack cloud setup, where 15 VMs are requested, all 15 VMs can have different VM startup times. In such scenarios, it can be helpful for both cloud providers and cloud users to know the average, minimum, and maximum VM startup times. However, there are several factors that affect such startup times such as VM image size and instance type [12] as well as CPU and disk utilization [13]. Given that multiple factors are involved, predicting such startup times can be challenging.

1.3 Defnition of the research problem

As described in the research context, there exists a concern for cloud providers about finding or predicting VM startup time which impacts SLA parameters of the IaaS layer and sometimes, other top service layers. Based on this concern, the following research problem has been defined.

VM startup time directly influences the quality of service of IaaS layers and in turn, the applications running on this layer. Hence, by finding or predicting average, minimum and maximum VM startup times, the CPs and application developers can more effectively manage end-user expectations of VM startup times and of related SLA parameters (e.g. service initiation time).

1.4 Hypothesis and research questions

With the pointers discussed in the previous sections, we have defined the following hypothesis. **A prediction model can be developed to predict average, minimum and maximum VM startup times. This considering a maximum prediction error of 12 %.**

In accordance with the stated hypothesis, we defined the following research questions to take forward this research.

- What are the generic factors impacting the virtual machine startup time in IaaS Cloud platforms?
- What kind of prediction techniques could be employed for VM startup time prediction?
- What are the steps of the method to design and evaluate a VM startup time prediction model?

1.5 Research Variables

The VM startup time depends upon the current resource utilization of physical nodes such as CPU, memory, and network conditions. It also depends on the characteristics of the requested VMs like image size of the VMs as well as the total number of VMs requested. Apart from physical node conditions and VM characteristics, there are other factors like data center interconnectivity. In this work we consider as the independent variables *Uc, Um, Un, Is,* and *N,* whereas the dependent variable is *VMst* as shown in the equation below.

$$VMst = f(Uc, Um, Un, Is, N)$$

where,

VMst	-VM startup time
Uc	-CPU utilization of the physical node
Um	-Memory utilization of the physical node
Un	-Network utilization of the physical node
Is	-Image size of requested VM
N	-number of VMs requested

1.6 Research goal

The main objective of this research is to define predictive models for average, minimum and maximum VM startup times.

The specific objectives are the following:

1. Analyze and identify the impact factors for VM startup times.
2. Find a suitable prediction methodology that could be applied for VM startup time prediction.
3. Design and implement prediction models for average, minimum and maximum VM startup times.
4. Evaluate and refactor the model design in order to keep the prediction error under 12%.

1.7 Research work scope and assumptions

As stated in the research goal sections, the primary objective of this work is to design and implement prediction models for VM startup times. In order to achieve the mentioned specific objectives, the scope and assumptions for this goal have been defined.

- The designed prediction models are applicable to the IaaS cloud environments where cluster resource usage metrics (such as CPU and network load as well as memory consumption) are available.
- Although the designed models consume utilization metrics of the cluster and do not include the means to obtain those metrics, the tools for obtaining these metrics are not within the scope of this work.
- The designed prediction models can be used on the platforms where, apart from VMs, other kinds of loads exist.

- The designed models are evaluated under real-world conditions like when other VMs are executing in the same environment.
- The impact of application-specific characteristics which are running inside the VMs are outside the scope of the designed models.

1.8 Structure of the thesis

This thesis is structured as follows. Chapter 2 details the theoretical context related to SLA parameters, VM startup time, and predictive methodologies. Chapter 3 presents the proposed method to design and implement the VM startup time prediction model. Chapter 4 evaluates the prediction model and discusses the results. Finally, Chapter 5 summarizes this research work and provides future work directions.

CHAPTER 2

Theoretical context

The focus of this chapter is to provide a detailed description of related concepts and a review of relevant work regarding the research problem. The first part provides an introduction to the IaaS cloud and the second part presents the relevant related work.

2.1 IaaS cloud

The cloud computing paradigm is gaining popularity in the last decade with a substantial day to day increase in the user base and commercial providers. The main purpose of cloud computing is to offer computing, storage, and software "as a service" [14]. Out of these, the provisioning of computing and storage services is categorized as "Infrastructure as a Service" (IaaS) and it is achieved by virtualization of these resources. Rajkumar et al. [14] state that "Offering virtualized resources (computing, storage and communication) on demand is known as Infrastructure as a Service)". The authors also define the desired features of a cloud as: 1. Self-service, immediate access to resources; 2. Per usage metering and billing, customers only pay for the resources used; 3. Elasticity, rapid provision of requested resources; and 4. Customization, all the provisioned resources are highly customizable like access to virtual servers. The elasticity feature is achieved by employing virtualization and on-demand resource provisioning. The cloud providers manage these aspects via cloud infrastructure management tools.

As mentioned previously, the basic building blocks of IaaS are virtual machines. The creation and management of these virtual machines are handled by infrastructure man-

agement tools. The calculation or prediction of VM startup time directly impacts the elasticity of the cloud.

2.2 Related work

The Cloud QoS Models that have been proposed so far e.g. [2], [15], [16] define different SLA parameters. For instance, Guerout et al. [2] classify SLA parameters into four categories, namely performance, dependability, security, and cost. As this work specifically addresses the SaaS cloud model, the SLA parameters discussed in this work are either inadequate or not required by the other cloud models. Alhamad et al. [15] propose a list of SLA parameters for each type of cloud service models and categorizes them into functional and non-functional requirements for cloud users. Baset et al. [16] compare SLA parameters of different public cloud providers and conclude that Cloud providers only focus on availability and/or request completion rate even though a wider SLA support is required by cloud users. Below we present related work, which we classify into the following categories: QoS parameters in resource allocation frameworks, prediction methods of QoS parameters, and VM startup time related works.

2.2.1 QoS parameters in Resource allocation frameworks

Van et al. [17] propose a framework to address the VM allocation and consolidation problem by considering high-level QoS guarantees (i.e. application level QoS). The authors proposed two-layer decision modules, namely local decision module (LDM) and global decision module (GDM). The LDM monitors application specific SLA parameters whereas the GDM enables VM consolidation without affecting SLA parameters. Two application level SLA parameters, namely response time and throughput, are considered in this work. This framework can be applicable to IaaS and SaaS providers. Wu et al. [5] propose two algorithms to map the user requests to VMs, in order to maximize profit by reusing VMs. The first algorithm maps the requests to VMs which have maximum space availability whereas the second allocates the requests to VMs with minimum available space. Both algorithms consider only the response time SLA parameter. Wu et al. [18] propose three algorithms to manage the admission control and scheduling problem at the SaaS level. The algorithms aim at maximizing the profits by minimizing the number of VMs, by rescheduling or exploiting penalty delays. This work considers both application

SLA parameters like deadline and budget and service initiation time at the provider side. Li et al. [19] propose an energy efficient and QoS aware model, based on the particle swarm optimization technique, to address the task allocation problem. This work considers response time, resource cost, and throughput as SLA parameters to optimize energy costs. Wang et al. [20] propose a particle swarm optimization method to solve an energy and QoS aware VM placement problem. This work considers response time, throughput, availability, and reliability as SLA parameters and can be applicable at the SaaS level. Singh et al. [21] propose a QoS-aware resource scheduling framework to consolidate and schedule cloud workloads. The cloud workloads are clustered using K-means on the basis of QoS weights. The authors, also propose cost-time based scheduling policies to schedule those clustered workloads. This work considers execution times as an SLA parameter and could be employed by SaaS providers. Samreen et al. [22] propose a machine learning based decision-making model to select public cloud instances. This work considers VM instance costs and application execution times as SLA parameters. Based on a polynomial regression technique, the application execution time is predicted to decide a suitable VM instance in public clouds. Zainelabden et al. [23] propose a conceptual framework to identify SLA violations. The framework monitors both the cloud data center metrics and application metrics to identify any SLA violation. Although no specific SLA parameters are mentioned in this work, it can be applicable for both availability and response time SLA parameters for SaaS providers. The work of Stantchev et al. [13] provides a framework for negotiating and enforcing the SLA between business owners and IT infrastructure providers (i.e. CPs). The proposed work considers the response time, throughput, and transaction rates as SLA parameters.

2.2.2 QoS prediction methods

There are several efforts addressing QoS prediction in cloud environments. Table 2.1 shows an overview of such efforts along with the SLA parameters involved in the proposed approaches. Some of the approaches that are employed at the SaaS level mainly focus on the response time SLA parameter [24], [11], [25], [26], [27], [28],[10]. Ma et al. [24] propose a collaborative QoS prediction approach based on cloud model theory and time

series analyses to predict response times of multiple users in a cloud environment. The prediction and time series analysis consider periodic variations of the QoS parameters and user application requirements. This approach can only be applied at the SaaS level. Xu et al. [11] propose a QoS prediction model based on a matrix factorization technique. The user features and service features are factorized in order to predict the QoS parameters. This model can be applicable to SaaS cloud models, which considers response time, throughput and reliability of the system.

Tabla 2.1: Summary of related works

Related Work	**Problem**	**Technique**	**QoS Parameters**	**Cloud Model**
H Ma et al. [24]	QoS Prediction, Service Selection	Time series analysis	Response Time	SaaS
Y Xu et al. [11]	QoS Prediction,Service Selection	Matrix Factorization	Response time, Throughput, Reliability	SaaS
K Su et al. [25]	QoS Prediction,Service Selection	K-means Clustering	Response Time, Throughput	SaaS

Z Chen et al. [26]	QoS Prediction,Service Selection	Clustering and Matrix Factorization	Response Time, Throughput	SaaS
W Zhang et al. [27]	VM migration,QoS Prediction	Markov Decision Process	Response Time	SaaS
R Karim et al. [29]	QoS prediction,Vertical Cloud Service Composition	Tensor Factorization	Response Time, Throughput	SaaS ,IaaS, DaaS (Data as a Service)
Z Ye et al. [30]	QoS Prediction,Service composition	Time series Group	Response Time, Throughput, Cost	SaaS, IaaS
P zhang et al. [31]	QoS Prediction, Service Selection	Bayesian Network Model	Response Time	IaaS, PaaS, SaaS

D Geebelen et al. [10]	QoS Prediction, QoS Aggregation, Service composition	Composite Time series,Kernel Based Quantile Regressor	Response Time, Throughput, Reputation, Reliability, Availability, Cost	SaaS
W Hussain et al. [32]	Resource Allocation, QoS Violation Prediction	Top-k Nearest Neighbors, Fuzzy Inference Model	Response Time, Throughput	PaaS, SaaS
B Tang et al. [28]	QoS Violation Prediction	Naïve Bayesian Model	Response Time	SaaS
Wu H. et al. [33]	VM launch overhead reference model	Non-Heuristic Model	VM launch overhead	IaaS, SaaS, PaaS

Su et al. [25] propose a trust-aware QoS prediction model based on the K-Means clustering technique. The QoS data for predictions is obtained from user-based clustering, which includes historical QoS data from trustworthy users and Service based Clustering

that includes historical QoS data for similar services. Based on the trustworthiness of users and services, a set of top-K QoS data points of similar services are used for carrying out the prediction. This work basically considers response time and throughput as QoS parameters. Chen et al. [26] propose a collaborative QoS prediction approach that uses a geographical neighborhood-based matrix factorization technique. The historical QoS parameters of each service from geographical neighbors of similar services are used to build the unified matrix factorization model that is employed to predict the QoS parameters. Although no specific QoS parameters are explicitly mentioned, as this model uses historic values to predict the corresponding QoS parameter, it can be applicable for parameters like response time and throughput. Zhang et al. [27] propose a QoS prediction model based on Markov decision processes to address a problem of QoS aware VM migration. The proposed framework predicts the response time of the services and triggers the VM migration if needed. Tang et al. [28] propose an SLA violation prediction technique based on a Naïve Bayesian Model. This approach considers historical QoS data along with a flexible number of features to predict the SLA violations on response time. Geebelen et al. [10] propose a QoS aggregation and prediction model by using composite time series and Kernel-based quantile regressor approaches respectively, to address the service composition problem. This work considers the following QoS parameters: response time, throughput, reputation, reliability, availability, and cost.

Some other approaches to QoS prediction are targeted to the IaaS level [32] [29] [30]. Hussain et al. [32] propose an SLA Violation prediction model based on the Top-k nearest neighbors approach to address a resource allocation problem. This approach considers response time and throughput of user transactions to predict the SLA violations of response time and throughput. Karim et al. [29] propose an end-to-end QoS prediction approach based on a tensor factorization technique. From the end user perspective, QoS parameters may not depend on a single cloud service but on the composition of multiple services. This approach vertically composes all related cloud services and apply tensor factorization to determine the response time and throughput of the system from the end user perspective. Ye et al. [30] propose a model for the service composition problem based on multivariate QoS prediction of the time series group technique. This model analyzes historical QoS

data of services and predicts the response time, throughput, and cost of services. This can be applicable to both the SaaS and IaaS models. Finally, an approach working at the IaaS, PaaS, and SaaS levels is proposed by Zhang et al. [31] whereby a QoS prediction model based on a Bayesian Network model is used to determine the response times.

2.2.3 VM startup times

There have been a few efforts towards analyzing the factors that have a big impact on VM startup times [34] [35]. Mao et al. [34] analyze the performances of three major public cloud provider platforms, namely Amazon EC2, Microsoft Azure, and RackSpace, based on VM startup times. According to this work, the VM OS image size has a linear relationship with VM startup time across all the platforms, whereas the VM instance type only affects the RackSpace and Azure platforms. Scaling multiple VM instance requests shows a linear performance behavior in the EC2 and Azure platforms. Although the authors are able to determine some of the most important factors that affect the performance of VM startup times, their work does not provide a predictive model for VM startup times. Razavi et al. [35] propose a VM image content consolidation technique to reduce the VM startup times at the PaaS level. This work analyzes how VM image size affects the VM startup times in Amazon EC2 platform but does not address VM startup time prediction.

Wu H. et al. [33] propose a reference model for predicting VM launching overhead (i.e. VM startup times). The reference model involves running a number of VM startup time benchmarks in a particular cloud platform to obtain a prediction model specific to such a platform. However, this can be an arduous and complex task. Although the reference model considers CPU and disk utilization of the system, the approach has not been tested under network traffic congestion conditions. Another limitation of this reference model is that only the CPU load of VMs is assumed, this restricting the cloud platform from running another kind of CPU-intensive tasks.

Although there are several works addressing the issue SLA models and some other efforts have tackled QoS prediction in cloud environments, there is little research regarding the prediction of VM startup times.

2.3 Discussion

This section presents the key elements found after analyzing the related works mentioned above. The first key elements are the SLA parameters and models proposed for the cloud. We can broadly classify SLA parameters into application-level SLAs and non-application specific SLAs. The SLA parameters like response time, throughput, service initiation time and availability are examples of application-specific parameters, whereas VM startup time is an example of a non application-specific parameter. The works [5], [13], [17], [18], [19], [20], [21], [22] and [23] address the first category, i.e. application-specific SLA parameters. The works [34], [35] and [33] address the second category, i.e. non-application specific SLA parameters. Although some works address different challenges concerning VM startup times, little work has been carried out to tackle the problem of predicting VM startup times.

The second key elements are the QoS prediction models of these approaches. The works [10], [25], [26], [27], [28], [32], [29], [30], [31] address the QoS prediction problem of Application-specific SLA parameters, whereas only the work in [36] addresses the, non-application specific SLA parameter, VM startup time parameter. This highlights the need for further research regarding QoS prediction approaches for VM startup times. The third key elements are the applicability of these approaches to the service layers of the cloud. All the above mentioned works are applicable to the SaaS layer whereas only [31], [29] and [36] are applicable to the IaaS layer.

2.4 Conclusion

This chapter presented the related works on SLA parameters, QoS prediction models and its applicability to different cloud service layers. Based on this, we were able to define our research goal. The objective of our proposed model will address the need for predicting VM startup times in the service layer of the IaaS cloud. The advantage of our proposed model is applicability, which could be implemented on any platform where the resource usage metrics are available.

CHAPTER 3

Regression tree prediction model

3.1 Introduction

This chapter describes the method to design regression tree prediction models for average, minimum and maximum VM startup times. At the beginning of this chapter, the phases of this method are defined. Later, the phases are detailed and we show how they are applied to our models.

3.2 Background on Regression trees

Classification and Regression Trees (CARTs) are a supervised machine learning technique that use a binary recursive partitioning procedure to create classification or regression trees. The type of tree to be created depends on the target values. Classification trees are used for categorical target values whereas the Regression Trees are employed for numerical target values. Say n samples of a variable Y in a training set has p feature variables, namely *X1, X2, ... Xp*. The decision tree is built by recursively partitioning the X space into k disjoint sets, *A1, A2,.. Ak*, such that the predicted value Y is j if X belongs to Aj, for j = *1, 2, ... k* [14]. The process of building this tree is called "tree induction". Table 3.1 shows the basic terminology of decision trees.

Multiple approaches have been proposed for building decision trees. CARTs use a binary splitting process to create the tree. In other words, a root node is split into two child nodes (or decision nodes). These child nodes are further split into two grandchildren

Tabla 3.1: Terminology used in Regression tree

Term	Description
Root Node	Represents entire sample / training set
Branch	Part of entire tree
Splitting	Process of dividing a node into two of more branches
Decision Node	A Node split into sub-nodes
End node	No further splits of nodes
Pruning	Process of removing a branch of a decision node

nodes and so on. X Wu et al. [37] describe this process which we represent in Algorithm 1.

Algorithm 1 Decision Tree building

```
1: endnode ← TrainingData(X)
2: C ← AttributeValue
3: rootnode ← endnode
4:
5: procedure SPLIT
6:     for node do
7:         if sample size is large or not in same target class then
8:             if X ≠ C then
9:                 AssignLeft
10:            else
11:                AssignRight
12:        else
13:            continue
```

3.3 Overall approach

The methodology to develop a Regression tree predictive model consists of the following phases: 1. Data collection, 2. Data preprocessing, 3. Training phase, 4. Apply the pruning process, 5. Validate the model. A brief introduction to these phases is given below.

Data collection phase The data collection phase is the process of obtaining the data relevant to the problem at hand. In our case, this is the process to collect the relevant data about VM startup times. A dataset with average, minimum, and maximum VM

startup times along with the dependent variables of the models is produced as output of this phase.

Data preprocessing phase Usually, before applying a machine learning process, the collected dataset needs to go through a transformation process in order to accommodate the needs of the chosen machine learning process. The transformation process may include cleaning and normalization of the data. The input of this process is the raw collected data. This phase produces a dataset that can be used to train the model.

Training phase In this phase, the preprocessed dataset is received as input in order to produce a regression tree model as output. The regression tree model is built using the CART algorithm [37].

Prunning phase The CART algorithm does not have any internal measures to improve the performance of the tree model. Such improvement can be achieved by either employing a cross-validation or a pruning process after the training phase, i.e. building the trees of the model. In our case, we employed a pruning process.

Validation phase This phase determines the accuracy of the prediction models. It takes a new dataset as input and predicts the target values. Then the predicted target values are compared with actual values to determine the prediction accuracy.

The Regression Tree model was implemented with the well-known "Statistics and Machine Learning tool box" of MATLAB. Further details of how each phase was carried out is provided below.

3.4 Data collection phase

As said earlier, the VM startup time SLA parameter is mainly used as a performance metric to evaluate public and private cloud infrastructures. Previous research efforts [34] have identified that both factors, VM image size and VM type, negatively impact the performance of this SLA parameter. Given that the study in [34] was conducted over public clouds, the authors could not find the impact of CPU utilization nor the memory and network usage. When a VM is being created and allocated to a physical node, the resource utilization of that physical node influences the allocation of resources to the

VM. Considering this, in this work we propose the following impact factors for VM start times: a) VM image size, b) CPU utilization of the physical node on which a VM is running, c) memory utilization of the physical node on which a VM is running, d) network utilization and e) number of concurrent VM creation requests. In addition, we propose three Regression Tree-based models for predicting average, minimum, and maximum VM startup times, respectively. These prediction models use the mentioned impact factors as features to estimate the VM startup times. The models are identical except that training and testing data varies depending on the targets. A dataset is created by collecting the above-mentioned impact factors for a given cloud environment.

3.5 Data preprocessing phase

The data collected from the previous phase are in the raw form. Hence, the data needs to be formatted in a suitable form whereby the training phase is able to process such data. The training data record should consist of target values followed by features. A sample of the training dataset is shown in Figure 3.1.

```
AverageTime,minimum,maximum,imagesize,vmType,noVMreq,CPUUtilization,MemoryUtilization,NetworkUtil
16.395,12.65,19.81,12.7,1,14,0.1646,4.569963972,0.0021296
6.24,5.03,7.45,12.7,2,5,5.6242,7.835791609,0.0037552
88.84,87.98,90.55,572.49,3,9,1.0531,6.895604876,0.0047984
101.95,101.95,101.95,894.56,4,4,1.4709,5.547008229,0.003092
7.78,7.78,7.78,12.7,1,1,35.3069,60.54704637,0.0026672
9.115,8.9,9.33,12.7,2,7,99.9021,61.31866775,0.0029624
94.235,94.09,94.38,572.49,3,10,67.9481,7.012623825,93.6810216
19.314,14.58,24.12,12.7,1,17,33.909,42.52278191,64.5691104
9.165,8.49,9.84,12.7,2,9,98.9755,47.58872317,0.0172704
86.515,86.43,86.6,572.49,3,7,33.3361,7.077628885,0.0027544
118.585,118.01,119.16,894.56,4,5,35.2449,9.642499235,0.006036
9.695,8.9,10.49,12.7,1,6,29.1114,27.39958354,0.0025544
5.73,5.73,5.73,12.7,2,3,88.8452,26.30158362,0.0031888
75.41,75.41,75.41,572.49,3,4,61.8587,6.377508151,94.2478568
104.94,104.94,104.94,894.56,4,2,34.951,7.641588377,0.0047048
10.34,9.42,11.26,12.7,1,5,34.2091,34.84000562,70.1903424
12.3475,6.07,17.08,12.7,2,14,99.9834,42.57324572,0.0030832
```

Figure 3.1: Sample formatted training dataset

3.6 Training phase

Since the Regression Tree technique is a nonparametric method, there is no need for normalizing or scaling of the training data. During the training phase, the MATLAB fitrtree function builds the Regression Tree model by using a binary split process, which follows the CART tree induction procedure. Part of the Regression Tree model is shown in Figure 3.2.

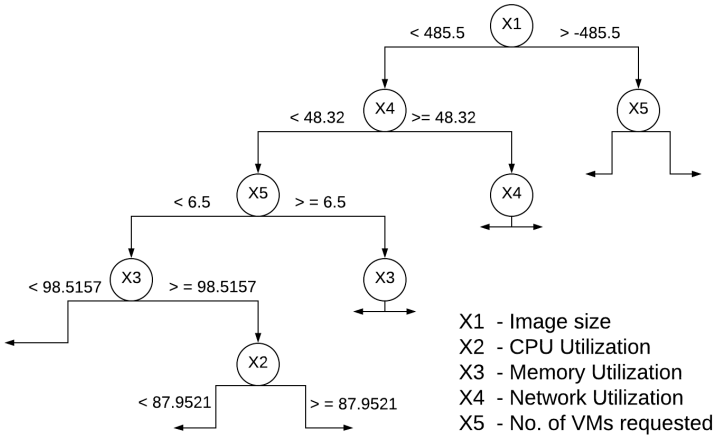


Figure 3.2: Visualization of the regression tree model

3.7 Pruning phase

The trained model then goes through the pruning process to avoid overfitting of the training data. Initially, the 10-fold cross-validation was applied, but with unsatisfactory results. Therefore, we applied the pruning process, which improved the prediction accuracy. The next chapter discusses the effect of the pruning process.

We have performed a pruning process based on the root mean square error (RMSE) of the test data. This process is described in Algorithm 2.

Algorithm 2 Pruning process

$TD \leftarrow TreeDepth$
$RMSE \leftarrow RootMeanSquareError\ at\ Current\ level$
$RMSE\_LOW \leftarrow Lowest\ Value$
$PL \leftarrow Prune\ level$
$OPL \leftarrow Optimized\ prune\ level$
$Train\ Model$
$PL \leftarrow TD - 1$
procedure PRUNE(TD, PL)
    $Evaluate\ Model$
    if $RMSE \leq RMSE\_LOW$ **then**
        $RMSE\_LOW \leftarrow RMSE$
        $PL \leftarrow PL - 1$
    else
        $OPL \leftarrow PL + 1$

The goal of this process is to identify the pruning level at which the RMSE of the test data is the lowest. First, the tree depth (TD) of the trained model and the RMSE is calculated (line 2 and 3). This initial RMSE is set for both RMSE and RMSE_LOW. The prune level is set to one level less than the tree depth (line 7). Now the pruned model is evaluated and we calculate the RMSE (line 10). This RMSE is compared with RMSE_LOW. If the RMSE decreases (i.e. the current RMSE is less than RMSE_LOW) then the pruning level is reduced one more level and we redo the pruning (line 8 to 12). This process continues until a prune level at which the RMSE increases is reached. This increase in the RMSE indicates that our optimum prune level has been crossed. Hence the optimum prune level is the current prune level plus one (line 14). The optimum prune level is used in the implementation of the model.

3.8 Validation phase

We considered the following metrics for validating the prediction accuracy: Mean Squared Error (MSE) and Squared Correlation coefficient (R^2). At the problem at hand, VM startup times may have a different range of values. For example, in an idle environment, the VM startup time for a Tiny VM image is around 20 seconds whereas for a Small VM image it is around 200 seconds. Since the MSE is more sensitive to extreme values, the Mean Absolute Error (MAE) was included in the metrics. Table 3.2 shows these metrics along with their equations.

Tabla 3.2: Performance Metrics for prediction

Metric	**Formula**
Error	E = Actual Value Predicted Value
Absolute Error	$\|E\|$
Mean Squared Error	$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$
Mean Absolute Error	$MAE = \frac{1}{n}\sum_{i=1}^{n}\|f(x_i) - y_i\|$
Squared Correlation Coefficient	$R^2 = \frac{(n\sum_{i=1}^{n} f(x_i)y_i - \sum_{i=1}^{n} f(x_i)\sum_{i=1}^{n} f(y_i))^2}{((n\sum_{i=1}^{n} f(x_i)^2) - (\sum_{i=1}^{n} f(x_i))^2 (n\sum_{i=1}^{n} y_i{}^2 - (\sum_{i=1}^{n} y_i)^2)}$
Average Error Percentage	$\%ofError = \frac{1}{n}\sum_{i=1}^{n}(\frac{(f(x_i)-y_i)}{f(x_i)} * 100)$

In the following chapter, we present an evaluation of our VM startup time prediction model.

CHAPTER 4

4. Model Evaluation

4.1 Introduction

This chapter presents the results of the evaluation of the proposed regression tree models for average, minimum and maximum VM startup times. This chapter is organized as follows. Section 4.2 describes the experiment system characteristics. Section 4.3 shows the accuracy of the prediction on all three models. This chapter ends with a discussion of the results in Section 4.4.

4.2 Experiment setup

Our experiments were run in a private cloud setup based on the Nadimit cluster located at CUCEA, University of Guadalajara. The cluster involves 4 nodes where each node includes two quad-core Xeon 5500 2.0 GH and 16 GB of main memory. The cluster nodes are interconnected via Ethernet switch. The cluster runs Linux Centos 7, kernel 3.10.0 along with OpenStack (Liberty). OpenStack was configured to use the KVM hypervisor and the “Provider Networks” (see below). The Master node acted as a Controller and the remaining four nodes acted as Compute nodes. Since the networking option "Provider Networks" configuration does not require a dedicated "Network node", all the five nodes were configured as "Compute Nodes". The hardware configurations of each node are shown in Table 4.1. The resource utilization of each node was monitored and recorded

with the help of Zabbix, a monitoring tool. Table 4.2 shows the metrics recorded for each type of resource.

Tabla 4.1: Characteristics of Cluster Nodes

Resource	**Availability**
Number of CPUs	8
Memory	16 GB
Network Bandwidth	100Mbps

Tabla 4.2: Zabbix Metrics Setup

Resource	**Metrics**
CPU	User Utilization of CPU
	System Utilization of CPU
Memory	Total Memory
	Available Memory
Network Bandwidth	Network incoming Traffic
	Network Outgoing Traffic

We defined a project setup along with a dedicated zone (with four computing nodes) in the OpenStack configuration. This is to make sure that all the VM requests are accommodated among the nodes assigned to the given zone. For this experiment, we have used four off the shelf VM flavors, namely Nano VM, Tiny VM, Small VM and Medium VM. The characteristics of these VM flavors are given in Table 4.3. We consider these flavors cover variations in the main characteristics of a VM flavor.

Tabla 4.3: VM flavor characteristics

Name	**VCPUs**	**Total Disk**	**RAM**	**Operating System**
m1.nano	1	1GB	64MB	Cirros
m1.tiny	1	5GB	512MB	Cirros
m1.small	1	20GB	2048 MB	Debian
m1.medium	2	20GB	4GB	Centos

In a real time environment, each node might have a different level of utilization for each resource at any given time. To mimic these random utilization levels of resources in nodes,

load generators scripts were scheduled to run on each node. The tools CPUloadgenerator [38] and fillmem were used to create CPU load and memory loads, respectively. Fillmem is a python script developed by us which allocates a random amount of memory to current processes. For each run of these tools, a random percentage of the load was created for the Compute nodes. The CPU load generator script was scheduled to run at the 5th and 35th minute of an hour. This script calls the CPUloadgenerator program to create a random load percentage on a given core. A random number generator ($RANDOM – inbuilt bash function) defined in a bash script determines how many CPU cores (between 0 to 7) need to be selected on a run. This is applied to all the processors of the Compute nodes. The random load percentage is chosen between 0 % to 100 % along with the CPU core on which this load needs to be imposed. The selected percentage of the load is applied to the random selection of CPU for 30 minutes whereas the remaining CPUs are left idle. After 30 minutes, the script will be recalled with a different random percentage of load and different random selection of CPU cores.

The memory load generator script is scheduled to run at the 7th and 32nd minute of an hour. This script calls the fillmem program to load a random amount of memory. Similarly to the CPU load generator, this script also uses a random number generator to determine the amount of memory load to be applied in a run. The random amount of memory falls between 1 to 16,000 MB. This amount of memory is per node. Each node has its own memory load generator whereby a different amount of memory load is simultaneously applied in each node.

The network traffic load between nodes is generated by a script (developed by us) that uses the ping command along with random packet sizes. The network load generator triggers every 1st minute of an hour and ends on the 59th minute. This script uses the ping command to induce the network load between nodes. The ping command transmits a given size of packets to a remote Compute node for one minute. Once one minute has elapsed, a new random packet size is obtained and transmitted. Also, this script considers the next node in the host list will be considered as a remote Compute node. For example, the network load generator on the Compute node 1 pings to Compute node 2, which in turns pings to Compute node 3 and so on till the last Compute node pings to Compute

node 1. The packet size is random, which is determined each second, involves between 1 and 1,000,000 bytes.

The CLI tool of OpenStack is used to create, launch and shutdown the VMs. The VM request generator script is scheduled to run at the 11th and 41st minute of an hour. For each run, this script requests nano VMs, tiny VMs, medium VMs and small VMs. The number of VMs to be requested will be determined by a random number generator. The range of this random number generated depends on the type of VMs requested. For tiny VMs, this range is between 1 to 10 whereas for small VMs, the range is 1 to 5. All the scripts mentioned above ran for 24 hours. In addition, the CPU, Memory and Network load generators are scheduled to be executed in different minutes of the hour, in order to make sure those loads are active at the same time of the execution of the VM creation scripts.

Out of the features mentioned in Chapter 3, as shown in the Table 4.4, features 3, 4 and 5 were obtained by the Zabbix tool, whereas features 1 and 5 were retrieved from the VM creation requests.

Tabla 4.4: Features and target of our model

Number	**Features**	**Target**
1	VM image size	Average, Minimum and Maximum VM StartupTime
2	VM Type	
3	CPU Utilization of a given physical node	
4	Memory utilization of a given physical node	
5	Network utilization of a given physical node	
6	Number of concurrent VM creation requests	

4.3 Results and discussion

In this section, the experimental results are presented and the performance of the proposed prediction models are evaluated. We first present the details of training datasets, followed by statistics regarding the mitigation of the overfitting. Then this section describes the distribution analysis of the training dataset. We then analyze a number of performance metrics of the average VM startup time model for the WRK3 node with statistical measures and graphical plots.

We created a training dataset with 3300 records and a prediction dataset with 600 records. We used MATLAB to create the Regression Tree models. The first 2700 records of the training dataset were used for training the model and the remaining records were used for testing. All the records of the prediction dataset were used to evaluate the trained model. Once the training phase was completed, the model was tested with the remaining 600 records.

During our training and evaluation phase, we have identified the overfitting in the models. As discussed before, we employed the pruning process to overcome this challenge. Table 4.5 shows the average percentage of error with and without pruning. As shown, the average percentage of error has been considerably reduced with pruning. For instance, in the case of node WRK3, we obtained 26.35% as an average prediction error of VM startup time (ST) with a tree depth of 400. When we applied the pruning process, the error percentage was reduced to 8.18%. In the same way, the minimum VM ST prediction error was reduced to 11.32 % and the maximum VM ST prediction was reduced to 11.45 %. The pruning process reduces data with outliers. Hence, this process helps to avoid an overfitting scenario.

For each compute node, we have created prediction models and the results are tabulated in Table 4.6.

Figure 4.1 shows the distribution of measured average VM startup times. It can be seen in this figure that startup times of VM type 1 and type 2 are overlapping, whereas type 3 and type 4 are grouped separately. Table 4.7 shows the corresponding mean and standard deviation for average VM startup times per VM type. Although, the standard

Tabla 4.5: Average percentage of error with and without Pruning

Node	Target	Tree Depth	% of Error (without pruning)	Prune level	% of Error (with pruning)
WRK3	Average VM ST	400	26.35	380	8.18
	Minimum VM ST	356	22.05	340	11.32
	Maximum VM ST	400	32.06	394	11.45
WRK4	Average VM ST	300	25.7	280	9.67
	Minimum VM ST	296	22.72	289	14.46
	Maximum VM ST	288	27.68	277	9.69
WRK5	Average VM ST	358	25.7	340	9.08
	Minimum VM ST	359	17.59	339	12.63
	Maximum VM ST	393	32.6	385	10.01
WRK6	Average VM ST	336	22.35	328	10.17
	Minimum VM ST	316	19.18	310	14.93
	Maximum VM ST	327	27.97	318	10.01

Tabla 4.6: Performance indicators for Average VM startup time prediction

Compute Node	Phase	R^2	MSE	MAE
WRK3	Testing	0.7034	0.00023	42.7106
	Prediction	0.9973	9.8886	2.3113
WRK4	Testing	0.6649	0.0003	46.4439
	Prediction	0.9772	8.6053	2.0733
WRK5	Testing	0.9895	44.1466	5.4684
	Prediction	0.9978	7.5566	1.96
WRK6	Testing	0.9920	37.7606	5.0511
	Prediction	0.9969	10.02	2.2887

deviation of type 3 and type 4 is 8.04% and 4.25%, respectively, we can also observe that the relative standard variation for type 1 and type 2 is 37.88% and 40.58%, respectively; which is considerable higher than 8.19% that is the average error of our approach. Future work will extend the dataset to obtain more complex scenarios involving higher standard deviation values for type 3 and type 4.

For example, in the case of node WRK3 we obtained the following. This testing process produced a squared correlation coefficient of 0.7034 and a Mean Squared error of 0.00023. The prediction dataset was then passed on to the model and was also evaluated. The result

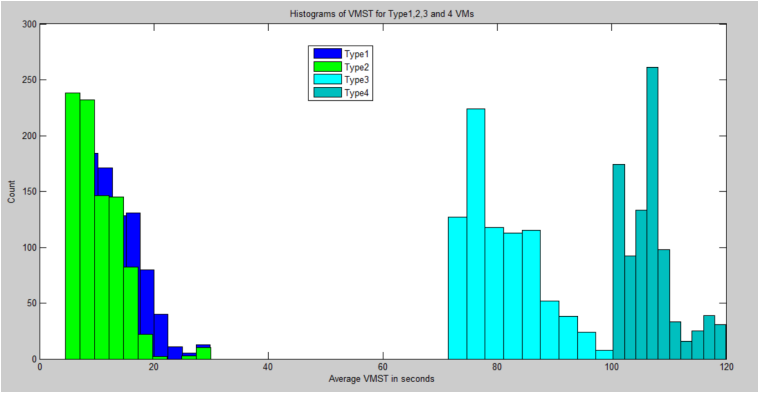


Figure 4.1: Histograms of Average VM startup times grouped by VM Type

Tabla 4.7: Statistics of training dataset per VM group

VM Type	Mean	Standard Deviation	Relative Standard Deviation
Type1 – Nano	12.7537	4.8375	37.88
Type2 - Tiny	10.2564	4.1666	40.58
Type3 - Small	80.9602	6.5125	8.04
Type4 - Medium	106.8654	4.5532	4.25

showed an R^2 of 0.9973 and an MSE of 9.8886. We can see that our model predicted the VM startup time with an accuracy of 70.34% during the testing phase whereas a 98.886% of accuracy was obtained during the prediction phase. Even with a high prediction accuracy, the MAE is around 2.3113. In general, the MSE of the prediction model increases when the prediction input value range moves away from the training input range [39]. In other words, a large MSE implies underfitting of the prediction model. Our prediction model has a considerably small MSE, even though there are two different ranges of VM startup times. Both the nano and tiny VMs have a VM startup time in the range of 20 seconds whereas medium and small VMs are in the range of 150 seconds. This causes large MSE values. Table 8 shows the values of R^2, MSE, and MAE in each phase. Figure 4.2 shows the plot of the actual and predicted values in the prediction phase, whereas Figure 4.3a shows a histogram of errors and Figure 4.3b shows the histogram of absolute errors.

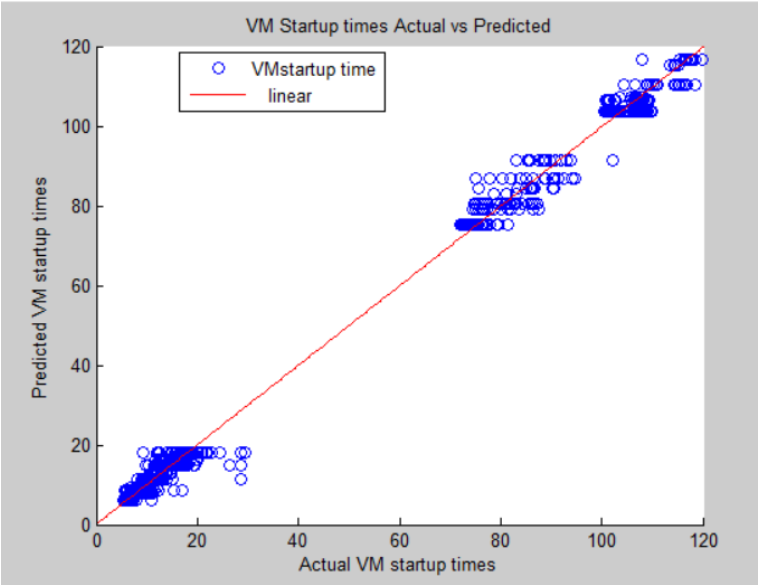
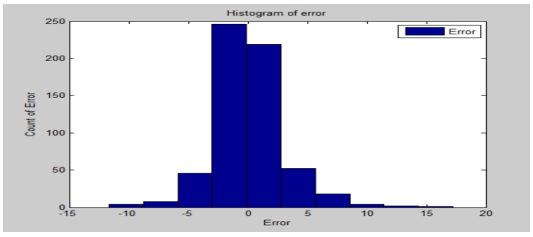
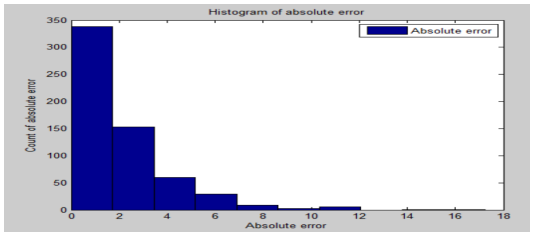


Figure 4.2: Average VM startup times. Actual vs predicted values



(a) Average VM startup times - Histogram of error

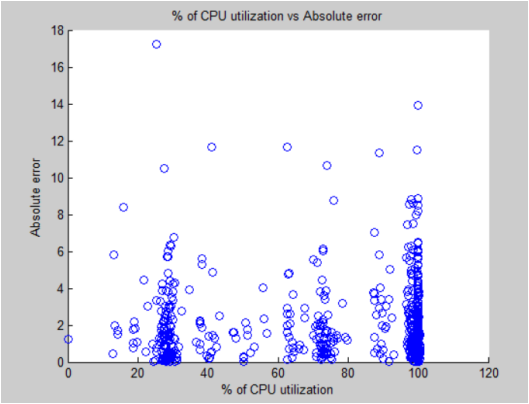


(b) Average VM startup times - Histogram of absolute error

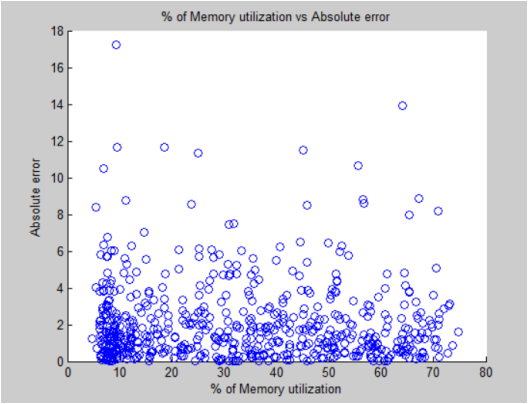
Figure 4.3: Histograms - Average VM Startup times

Given that the prediction accuracy values we obtained in each node were very similar, as can be seen in Table 4.6, we only present the results of the node WRK3. However, the results of the other nodes can be consulted in our git repository [40]. In addition, for the sake of brevity, we only present the analysis of the average VM startup time. Nevertheless, the results of the minimum and maximum startup time are very similar to the results of the average startup time and can be consulted in our git repository [40]. Further related plots are included in Appendix A.

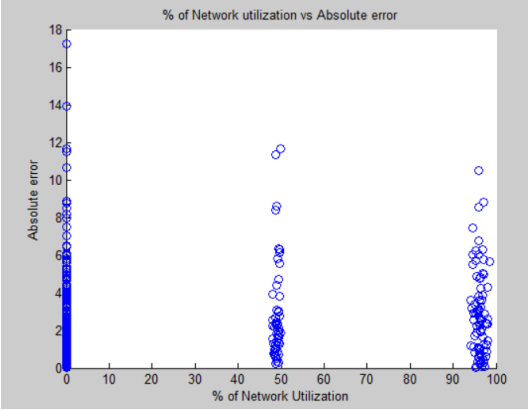
In figure 4.2, the average VM startup times are plotted against their corresponding predicted values. The red line in the plot shows, as it supposed to be, that actual and predicted values follow a linear relationship.



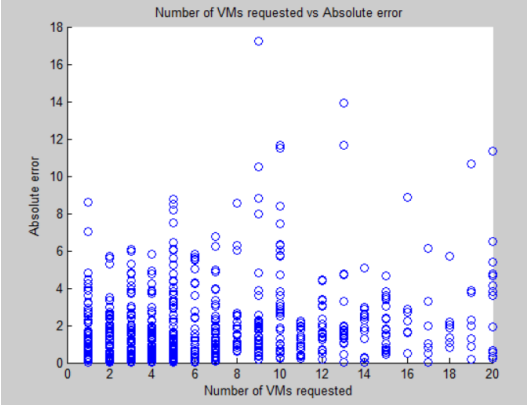
(a) Percentage of CPU utilization vs absolute error



(b) Percentage of memory utilization vs absolute error



(c) Percentage of network utilization vs absolute error



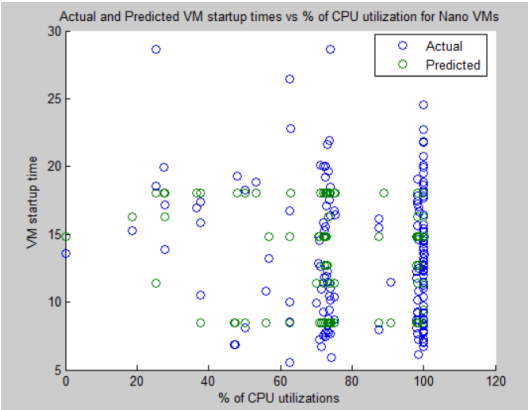
(d) Number of VMs requested vs absolute error

Figure 4.4: Average VM Startup times - features vs absolute error

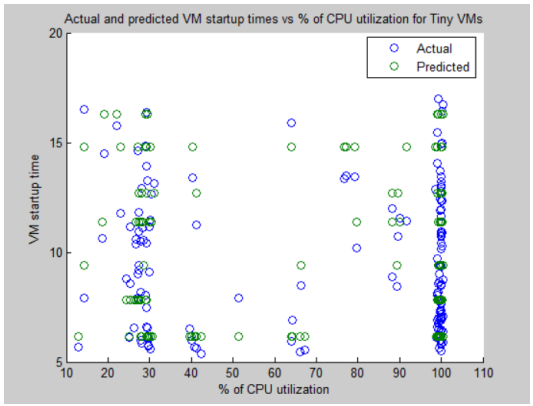
As mentioned earlier, there are four types of VM images used in this experiment and each one has a different range of VM startup times. Even in this scenario, the MSE is considerably small. Hence MAE has been considered as another performance metric. The MAE for the prediction phase is 3.9. The histogram of errors and absolute errors is shown in figures 4.3a and 4.3b, respectively.

In general, a histogram of errors should have a symmetric shape and be concentrated near to zero. The histogram of absolute errors shows this expected behavior whereas the histogram of errors concentrated near to zero. In general, the correlation coefficient R^2 measures how well the data fitted into the model. But it does not help to understand the accuracy of the predictions. The performance metrics MSE and MAE are one way

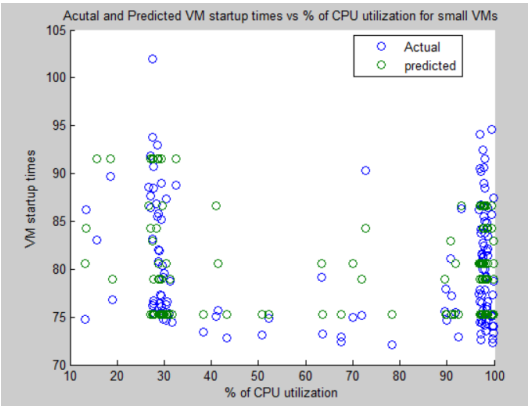
to evaluate the accuracy of the predictions. Such metrics were mentioned in the above paragraph. Another way is to scatter in a plot the residuals (prediction errors) against the features and/or against time. In our prediction model, the prediction value (i.e. the VM startup time) is irrelevant to the time of the day. Hence, the residuals (i.e. the absolute errors) are plotted against the features in figures 4.4a, 4.4b, 4.4c, and 4.4d. Most of the error data is concentrated around zero which implies that the error band is small.



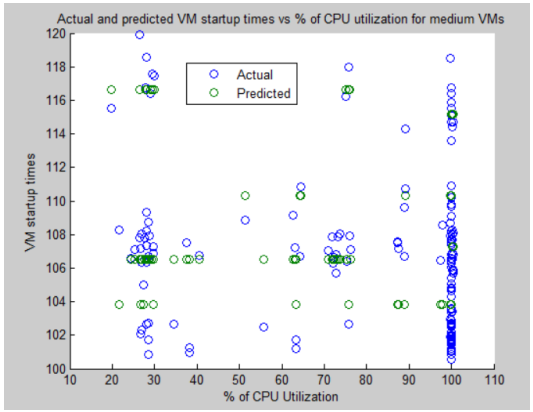
(a) Actual and Predicted Average VM startup times (in seconds) vs percentage of CPU utilization for nano VMs



(b) Actual and Predicted Average VM startup times (in seconds) vs percentage of CPU utilization for tiny VMs



(c) Actual and Predicted Average VM startup times (in seconds) vs percentage of CPU utilization for small VMs



(d) Actual and Predicted Average VM startup times (in seconds) vs percentage of CPU utilization for medium VMs

Figure 4.5: Average VM Startup times by VM types vs CPU Utilization

Figure 4.5b shows the plot of actual and predicted VM startup times versus the percentage of CPU utilization of tiny VMs. In Figure 4.5b, the actual and predicted values are almost superimposed and clustered together. This shows how close are the predicted values to the actual values. Figure 4.5c, figure 4.5d and figure 4.5a show the same for small

, medium and nano VMs, respectively. By comparing figures 4.5a, 4.5b, 4.5c, and 4.5d we can conclude that the prediction of VM startup times for Tiny VMs is more accurate than that of small and medium VMs. As mentioned in Section 4, Tiny VM, Small VM, and Medium flavors have different configuration and operating systems. The nano and tiny VMs use CirrOS whereas Small and Medium VMS employ Debian and CentOs, respectively. The higher accuracy of the prediction of Tiny VMs may be due to the fact that the behavior complexity of CirrOS is smaller than that of Debian and CentOs. Further study is needed to identify these particular differences.

We have evaluated our Regression Tree model against two different prediction approaches with the same test dataset. We chose one heuristic method, namely Artificial Neural networks (ANNs), and a standard regression prediction method, namely the Support Vector Regression (SVR) prediction method [41]. For this purpose, we selected the ANN model discussed by Islam et al. [42]. We created a feed-forward neural network with 5 hidden layers along with purelin transfer function.

In the case of SVR, we selected the standard SVR model [41] with RBF kernel. The two most important parameters of an RBF kernel are C and gamma. These two parameters are unknown for any given training set of any problem. The parameter C is defined as a tradeoff between flatness and tolerated deviation of target values, and is also known as a penalty factor. A small C parameter value leads to underfitting whereas a large value causes overfitting. The gamma parameter is the inverse of the standard deviation of the RBF kernel function, which is used as a similarity measure between two points (input vectors). If the gamma parameter has a small value, then the RBF kernel has a large variance. In this case, two different input vectors can be considered similar even when they are far from each other. The best approach is to randomly vary the value of these parameters within a calculated range to find the best values for each parameter. The best values are those that produce a high rate of accuracy in the predictions. We used a grid search involving a 10-fold cross-validation process to select the C and gamma parameters for a given training dataset. For our test dataset, the optimum C and gamma values are 8 and 0.03125. Both models (i.e. ANNs and SVR) were implemented in MATLAB and received the same test datasets. The results are listed in Table 4.8 below.

Tabla 4.8: Comparison of the Prediction Models for average VM startup time

Approach	**MAE**	**MSE**	**Median of Absolute error**	**Average % Error**	**Relative Standard Deviation**
ANN [42]	3.08	15.4	3.08	10.72%	1.83
SVR	2.46	10.68	1.67	8.96%	1.807
Regression Tree	2.3113	9.8886	1.45	8.1807 %	1.27

Table 4.8 shows that the ANN approach has an MAE of 3.08 and an MSE of 15.4, but has an average error of the 10.72%, which is slightly larger than the average error of our Regression Tree model. The SVR approach provides an MAE and an MSE slightly larger than those of our Regression Tree model. Table 4.8 also shows the median of absolute errors for each approach. As it can be seen, the Regression Tree model provides the smallest median of absolute error (i.e. 1.45) when compared to the other two approaches (i.e. ANN with 3.08 and SVR with 1.67). The Regression Tree model has the smallest average percentage error. This clearly shows that our Regression Tree model provides a better prediction accuracy than the other two approaches. The test datasets and MATLAB scripts we employed in the experiments are available in our git repository [40].

The Table 4.9 and Table 4.10 show the same measurements for minimum and maximum startup time, respectively. The measurements of the minimum startup time model are similar to those of the average startup time prediction model. Regarding the measurements of the the maximum startup time prediction model, although the SVR model provides a better percentage of average error, the Regression Tree model offers a slightly better relative standard deviation.

Tabla 4.9: Comparison of the prediction models for minimum VM startup times

Approach	**MAE**	**MSE**	**Median of Absolute error**	**Average % Error**	**Relative Standard Deviation**
ANN [42]	3.61	19.87	7.77	16.67%	2.54
SVR	2.57	10.608	5.61	12.25%	2.33
Regression Tree	2.202	8.78	4.713	11.327 %	1.42

Tabla 4.10: Comparison of the prediction models for maximum VM startup times

Approach	**MAE**	**MSE**	**Median of Absolute error**	**Average % Error**	**Relative Standard Deviation**
ANN [42]	3.48	28.81	6.32	13.17%	2.14
SVR	2.83	26.6	2.83	9.002%	1.91
Regression Tree	2.84	33.17	4.35	11.45 %	1.66

Finally, Table 4.11 shows the execution times taken by each model in the training, testing, and prediction phases. These times were measured under the same environment conditions and with the same input datasets. The ANN model takes more time to train whereas testing and prediction times are almost in line with the other models. The training phase of the SVR model requires less time than the ANN model, but more than that of the Regression Tree model. The poor performance of the ANN, which deserves further analysis, may be due to the fact that, depending of the data analyzed, it may require, besides the usual training process, a careful topology design (number of hidden layers, number of neurons per hidden layer, type of connections, etc.). Perhaps, for our particular dataset, such design requires more attention and such analysis is now part of our future work. In terms of execution times, the Regression Tree model outperforms the other models and shows nearly two orders of magnitude of better performance in the last two phases.

Tabla 4.11: Comparison of the Execution times (in seconds) of the Prediction Models

Phase	**ANN**	**SVR**	**Regression Tree**
Training	4.052810s	0.143945s	0.012049s
Testing	0.158625s	0.015330s	0.001321s
Prediction	0.007648s	0.024942s	0.002393s

Although the prediction accuracy of the SVR model is pretty close to that obtained by the Regression Tree model, an important advantage of the latter over the other two models is that the Regression Tree graphical model helps to understand the phenomenon represented in the data. In our particular case, a performance analysis can be helpful to

detect and avoid performance bottlenecks. For instance, part of the generated tree model of our testbed platform where the features located in the upper nodes of the tree are those whose values are more important for the prediction process. As we can see, the impact level of the features is as follows (ordered from a higher impact to a lower impact): VM image size (feature 1), number of VM requested (feature 5), network utilization (feature 4), CPU utilization (feature 2), and memory utilization (feature 3). Such order suggests that the VM image is critical to predict an accurate startup time, whereas the CPU utilization is useful, but less important. The VM image size has the greatest impact given that this feature is the root element in the tree. Then, those factors that have more splits in the tree, particularly at the higher levels, encompassing a larger number of nodes have a greater impact. For example, we can see that the second level subtree based on feature 5 is much larger compared to the subtree based on feature 4. Although figure 3.2 shows only one part of the tree model, the interested readers can download the related files from our repository [40] and visualize the complete tree model. We can observe that the predicted values follow two different range of values depending on two sets of features. Features 1 and 5 are not specifically related to physical hosts, rather, these features are related to the characteristics of a VM. The other three features (i.e. 2, 3, and 4) determine the fine-grained division of predicted values. It is important to note that these three features represent resource utilization of the given physical machine. Based on this behavior, it is safe to state that VM characteristics determine VM startup times in terms of coarse-grained values, whereas resource availability of a given physical machine customizes the accuracy of the predicted values.

CHAPTER 5

Conclusion and future work

5.1 Introduction

This dissertation presented a method to design regression tree prediction models for VM startup times in the IaaS clouds. As we have mentioned, the availability of VM startup times will help the cloud providers to set the user and application expectations right. With this prediction model, the cloud providers can plan VM provisioning and better quality of service parameters. We have reviewed the literature to identify the relevant works that are published in the field of cloud computing and QoS predictions. We were able to identify multiple QoS models and parameters for Cloud computing environments. But, we found a lack of VM startup time prediction methods able to be applied to IaaS cloud platforms.

5.2 Summary

As mentioned above, the research goal reported in this thesis is to develop a method for designing VM startup time prediction models for the IaaS cloud environments. Chapter 1 elaborated about the need for such prediction models. Additionally, this chapter provided a detailed view of the methodology by defining the research problem at hand, hypothesis, research questions, and the generic and specific goals for this research.

Chapter 2 presented the theoretical context of the research domain. The research context was defined to increase the focus of the problem at hand and then we presented

the literature analysis of QoS parameters and prediction models in the defined research area.

Chapter 3 provided a description of the proposed method to design the Regression tree VM startup time model for the cloud environment. This chapter also detailed the phases of the predictive modeling process along with how these phases are carried out within our proposed method.

Chapter 4 presented the detailed characteristics of the experiment / testbed setup which were used to collect the data for the testing and evaluation phase of our model. Chapter 4 also presented the distribution analysis of the test dataset followed by the accuracy of the prediction models. Finally, a detailed overall discussion about the results was presented.

5.3 Answers to research questions

In Chapter 1, we defined three research questions to provide a guideline for our research. The following paragraph provides the answers to those research questions.

- **What are the generic factors impacting the virtual machine startup time in IaaS Cloud platforms?** There are different factors that affect virtual machine startup times in the IaaS cloud. Although there is a lack of works focusing on this topic, through [3][31][54] we have identified some of the generic impact factors like VM image size, VM types and resource availability. We further limited the resource availability impact factor into CPU, Memory and Network utilization. In summary, the factors impacting the virtual machine startup times are: a) VM image size, b) CPU utilization of the physical node on which a VM is running, c) memory utilization of the physical node on which a VM is running, d) network utilization and e) number of concurrent VM creation requests.

- **What kind of prediction techniques could be employed for VM startup time prediction?** In the theoretical context section, we have presented various prediction techniques used for QoS prediction. Out of those, we have identified the

CART algorithm to implement our model. The Classification and Regression Trees algorithm is a supervised learning method and does not require a lot of preprocessing like the one needed for the normalization of the data. The challenges of overfitting were mitigated by using an appropriate level of pruning. Chapter 3 gives details about the pruning process employed in our model. As mentioned in Chapter 4, ANN and SVR techniques could be employed for the problem at hand, but with their own drawbacks. For example, the SVR technique needs the normalization of the training and prediction datasets. Chapter 4 has also shown that the Regression tree model performs slightly better than the other two techniques.

- **What are the steps of the method to design and evaluate a VM startup time prediction model?** We proposed a method based on the Regression tree supervised learning technique, to design the VM startup time prediction. We have presented a step by step method to design and implement our model in Chapter 3. The method consists of a data collection phase, data preprocessing phase, training phase, pruning phase, and the validation phase.

The data collection phase describes the process of collecting the data needed from our testbed whereas the data preprocessing presents the cleanly formatted dataset for the training phase. We have used the CART algorithm to build the regression tree prediction model and then applied the pruning process to mitigate the overfitting. Finally, the predictive model has been evaluated based on the performance metrics defined in Chapter 3.

5.4 Contributions

The main contribution of this thesis is the proposed method to design a regression tree prediction model in the IaaS cloud. The designed model is capable of predicting average, minimum and maximum VM startup times in an IaaS cloud. This assuming we have access to the resource utilization (i.e. CPU, memory and network utilization) and VM characteristics (i.e. VM image size and VM type). The models generated for average VM

start up time by the proposed method have an average accuracy of 91%, which is within the 12% error range stated in the research goal.

An important characteristic of the proposed models is that they can be applied to any cloud environment where the physical node resource utilization is available, as mentioned before.

We have designed three prediction models involving average, minimum and maximum VM startup times, respectively. The designed models accept the same predictors but different target parameters. The trained models were then evaluated based on the defined performance metrics. As shown in Table 8, our models have shown similar performance across different physical nodes and a maximum MAE of 2.311. The minimum R^2 of the prediction is 97.72 %.

Another important contribution is showing that the regression trees (CART) are a suitable technique for developing a prediction model for VM startup times in the IaaS cloud environments.

Finally, a number of papers were published reporting some of the results of this research. The abstracts of these papers are included in Appendix B.

- Yatheendraprakash Govindaraju, Hector A. Duran-Limon, 2016. **A QoS and Energy Aware Load Balancing and Resource Allocation Framework for IaaS Cloud Providers.** In Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing(UCC '16), Shanghai, China. 410–415.

- Yatheendraprakash Govindaraju, Hector A. Duran-Limon, Efren Mezura-Montes, 2020. **A Regression Tree Predictive Model for Virtual Machine Startup Time in IaaS clouds.** Cluster computing. The paper is "Accepted with Minor revisions" and Currently in the status of "revisions being processed". (JCR Q3 with impact factor of 1.851)

5.5 Future work

The future path for our research is as follows.

- **Create a VM allocation framework by adopting the proposed VM startup time prediction models.** Cloud providers need a complete framework to effectively manage resource allocations. By adopting the VM startup time prediction models, the VM allocation framework could be more effective in handling VM startup time expectations.

- **Expand the work to a wider range of VM types.** As the need for cloud computing increases, cloud providers offer a wider range of VM services, suitable for different needs of cloud users, like compute-optimized VMs and memory-optimized VMs. These specific characteristics of the VMs can be incorporated into our prediction model to effectively predict the VM startup times.

- **Improve prediction accuracy.** Currently, we have employed the pruning process to mitigate the overfitting in our models. There is a possibility of incorporating the tree selection techniques that could be employed for improving the accuracy of the current models.

- **Reduce the outliers in the VM startup times.** One of the drawbacks of any machine learning technique is the impact of poor quality of the training data. In Chapter 4, we presented the quality of our training data and it was shown that VM type 3 and 4 have outliers which directly affect the performance of our models. A larger testbed is needed to collect sufficient datasets for these VM types.

- **Explore the support for upcoming compute technologies.** The Cloud providers are moving towards the containerization services along with Virtual machine services. Our prediction models are applicable to only Virtual machine services for the time being. The proposed models could be extended to improve the Container services too.

5.6 Concluding Remarks

Service level agreement parameters are important to set the right expectations between cloud users and providers. Each service model employs different SLA parameters

in accordance with the users' needs. Among these parameters, at the IaaS level, the VM startup time is an important SLA parameter, which impacts both infrastructure and application-level performance.

This thesis has investigated a method to design models to predict average, minimum, and maximum VM startup times in IaaS cloud environments. Although there is scope for improving the evaluation for a wider variety of VM types, the work presented in this thesis provides a stepping stone for the future development of works on VM startup times in the IaaS cloud.

Bibliography

[1] "Amazon compute service sla (2019)." [Online]. Available: https://aws.amazon.com/compute/sla

[2] T. Guérout, S. Medjiah, G. Da Costa, and T. Monteil, "Quality of service modeling for green scheduling in Clouds," *Sustainable Computing: Informatics and Systems*, vol. 4, no. 4, pp. 225–240, dec 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S221053791400047X

[3] D. Bruneo, "A stochastic model to investigate data center performance and qos in IaaS cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 560–569, mar 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6473795/

[4] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning," in *2011 IEEE Nineteenth IEEE International Workshop on Quality of Service*. IEEE, jun 2011, pp. 1–9. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5931341

[5] L. Wu, S. K. Garg, and R. Buyya, "SLA-Based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, 2011, pp. 195–204. [Online]. Available: http://www.cloudbus.org/papers/SLA-SaaS-CCGrid2011.pdf http://ieeexplore.ieee.org/document/5948610/

[6] A. V. Lakra and D. K. Yadav, "Multi-Objective Tasks Scheduling Algorithm for Cloud Computing Throughput Optimization," *Procedia Computer Science*, vol. 48, pp. 107–113, 2015.

[7] "Rackspace cloud sla(2019)." [Online]. Available: https://www.rackspace.com/information/legal/cloud/sla

[8] "Microsoft azure sla (2019) ." [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/?v=17.23h610

[9] "Google cloud sla(2019)." [Online]. Available: https://cloud.google.com/compute/sla

[10] D. Geebelen, K. Geebelen, E. Truyen, S. Michiels, J. A. K. Suykens, J. Vandewalle, and W. Joosen, "QoS prediction for web service compositions using kernel-based quantile estimation with online adaptation of the constant offset," *Information Sciences*, vol. 268, pp. 397–424, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2013.12.063

[11] Y. Xu, J. Yin, S. Deng, N. N. Xiong, and J. Huang, "Context-aware QoS prediction for web service recommendation and selection," *Expert Systems with Applications*, vol. 53, pp. 75–86, 2016. [Online]. Available: http://ac.els-cdn.com/S0957417416000208/1-s2.0-S0957417416000208-main.pdf?_tid=de58d4bc-3a91-11e7-a15b-00000aab0f27&acdnat=1494978563_c8f98222e9e279760bfc04ae2cd0aae4

[12] D. Serrano, S. Bouchenak, Y. Kouki, F. A. de Oliveira Jr., T. Ledoux, J. Lejeune, J. Sopena, L. Arantes, and P. Sens, "SLA guarantees for cloud services," *Future Generation Computer Systems*, vol. 54, pp. 233–246, apr 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X15000801

[13] V. Stantchev and C. Schröpfer, "Negotiating and enforcing QoS and SLAs in grid and cloud computing," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5529. Springer, Berlin, Heidelberg, 2009, pp. 25–35. [Online]. Available: http://link.springer.com/10.1007/978-3-642-01671-4_3

[14] R. Buyya, J. Broberg, and A. Gosécinéski, *Cloud computing : principles and paradigms*. Wiley, 2011. [Online]. Available: https://www.wiley.com/en-us/Cloud+Computing%3A+Principles+and+Paradigms-p-9780470887998

[15] M. Alhamad, T. Dillon, and E. Chang, "Conceptual SLA framework for cloud computing," in *4th IEEE International Conference on Digital Ecosystems and Technologies - Conference Proceedings of IEEE-DEST 2010, DEST 2010.* IEEE, apr 2010, pp. 606–610. [Online]. Available: http://ieeexplore.ieee.org/document/5610586/

[16] S. A. Baset, "Cloud SLAs," *ACM SIGOPS Operating Systems Review*, vol. 46, no. 2, p. 57, jul 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2331576.2331586

[17] H. N. Van, F. D. Tran, and J. M. Menaud, "SLA-aware virtual resource management for cloud infrastructures," *Proceedings - IEEE 9th International Conference on Computer and Information Technology, CIT 2009*, vol. 1, pp. 357–362, 2009.

[18] L. Wu, S. Kumar Garg, and R. Buyya, "SLA-based admission control for a Software-as-a-Service provider in Cloud computing environments," in *Journal of Computer and System Sciences*, vol. 78, no. 5, 2012, pp. 1280–1299. [Online]. Available: www.elsevier.com/locate/jcss

[19] H. Li, G. Zhu, Y. Zhao, Y. Dai, and W. Tian, "Energy-efficient and QoS-aware model based resource consolidation in cloud data centers," *Cluster Computing*, pp. 1–11, may 2017. [Online]. Available: http://link.springer.com/10.1007/s10586-017-0893-5

[20] S. Wang, A. Zhou, C. H. Hsu, X. Xiao, and F. Yang, "Provision of Data-Intensive Services Through Energy-and QoS-Aware Virtual Machine Placement in National Cloud Data Centers," *IEEE Transactions on Emerging Topics in Computing*, vol. 4, no. 2, pp. 290–300, 2016.

[21] S. Singh and I. Chana, "QRSF: QoS-aware resource scheduling framework in cloud computing," *The Journal of Supercomputing*, vol. 71, no. 1, pp. 241–292, sep 2014. [Online]. Available: http://link.springer.com/10.1007/s11227-014-1295-6

[22] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," in *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 557–563.

[23] A. A. Z. A. Ibrahim, D. Kliazovich, and P. Bouvry, "Service Level Agreement Assurance between Cloud Services Providers and Cloud Customers," in *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, 2016.

[24] H. Ma, H. Zhu, Z. Hu, W. Tang, and P. Dong, "Multi-valued collaborative QoS prediction for cloud service via time series analysis," *Future Generation Computer Systems*, vol. 68, pp. 275–288, 2017. [Online]. Available: http://ac.els-cdn.com/S0167739X16303971/1-s2.0-S0167739X16303971-main.pdf?_tid=26b4773e-3a92-11e7-bef1-00000aab0f01&acdnat=1494978684_e67b377b7592333afd69f173df27c0ea

[25] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for web service recommendation," *Knowledge-Based Systems*, vol. 115, pp. 55–65, 2017. [Online]. Available: http://ac.els-cdn.com/S0950705116303331/1-s2.0-S0950705116303331-main.pdf?_tid=ae08a7e2-3a91-11e7-80ae-00000aab0f6c&acdnat=1494978482_11223ebaf0de1d5bd9be0c2f37f1ea0b

[26] Z. Chen, L. Shen, and F. Li, "Exploiting Web service geographical neighborhood for collaborative QoS prediction," *Future Generation Computer Systems*, vol. 68, pp. 248–259, 2017. [Online]. Available: http://ac.els-cdn.com/S0167739X16303533/1-s2.0-S0167739X16303533-main.pdf?_tid=1d22a98a-3c1c-11e7-b5b2-00000aab0f26&acdnat=1495147890_a1ea343ed71787a847104f5e38338392

[27] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "SEGUE: Quality of service aware edge cloud service migration," in *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom*, 2017, pp. 344–351.

[28] B. Tang and M. Tang, "Bayesian model-based prediction of service level agreement violations for cloud services," in *Proceedings - 2014 International Symposium on Theoretical Aspects of Software Engineering, TASE 2014*, 2014, pp. 170–176.

[29] R. Karim, C. Ding, A. Miri, and M. S. Rahman, "End-to-End QoS Prediction Model of Vertically Composed Cloud Services via Tensor Factorization," in *Proceedings - 2015 International Conference on Cloud and Autonomic Computing, ICCAC 2015*, 2015, pp. 158–168.

[30] Z. Ye, S. K. Mistry, A. Bouguettaya, and H. Dong, "Long-term QoS-aware Cloud Service Composition using Multivariate Time Series Analysis," *Services Computing, IEEE Transactions on*, vol. PP, no. 99, p. 1, 2014.

[31] P. Zhang, Q. Han, W. Li, H. Leung, and W. Song, "A Novel QoS Prediction Approach for Cloud Service Based on Bayesian Networks Model," in *IEEE International Conference on Mobile Services*, vol. 33, no. 1. IEEE, jun 2016, pp. 24–28. [Online]. Available: http://ieeexplore.ieee.org/document/7787062/ http://ieeexplore.ieee.org/document/7368034/

[32] W. Hussain, F. Hussain, and O. Hussain, "Allocating optimized resources in the cloud by a viable SLA xodel," in *2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016*, 2016, pp. 1282–1287.

[33] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, K. Chadwick, and S. Y. Noh, "A Reference Model for Virtual Machine Launching Overhead," *IEEE Transactions on Cloud Computing*, vol. 4, no. 3, pp. 250–264, jul 2016. [Online]. Available: http://ieeexplore.ieee.org/document/6953063/

[34] M. Mao and M. Humphrey, "A performance study on the VM startup time in the cloud," in *Proceedings - 2012 IEEE 5th International Conference on Cloud Computing, CLOUD 2012*. IEEE, jun 2012, pp. 423–430. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6253534

[35] K. Razavi, L. M. Razorea, and T. Kielmann, "Reducing VM startup time and storage costs by VM image content consolidation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8374 LNCS. Springer, Berlin, Heidelberg, 2014, pp. 75–84. [Online]. Available: http://link.springer.com/10.1007/978-3-642-54420-0_8

[36] F. K. Dosilovic, M. Brcic, and N. Hlupic, "Explainable artificial intelligence: A survey," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., jun 2018, pp. 210–215.

[37] X. Wu and V. Kumar, *The Top Ten Algorithms in Data Mining*, 2009. [Online]. Available: https://xa.yimg.com/kq/groups/81025504/1871639475/name/The_Top_Ten_Algorithms.pdf#pa

[38] "CPUloadgenerator." [Online]. Available: https://github.com/GaetanoCarlucci/CPULoadGenerato

[39] M. H. C. M. U. DeGroot and M. J. C. M. U. Schervish, *Probabilty and Statistics 4th ed.*, 2002.

[40] "Repository for source code related to vm startup time." [Online]. Available: https://github.com/yathee/VMstModel

[41] Chih-Wei Hsu, Chih-Chung Chang and C.-J. Lin, "A Practical Guide to Support Vector Classification," *BJU international*, vol. 101, no. 1, pp. 1396–400, 2008. [Online]. Available: http://www.csie.ntu.edu.tw/ http://www.csie.ntu.edu.tw/ cjlin/papers/guide/guide.pdf

[42] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.05.027

APPENDIX A

Plots of minimum and maximum VM start up time (VMST) models results

A.1 Plots of minimum VMST model prediction results for WRK3

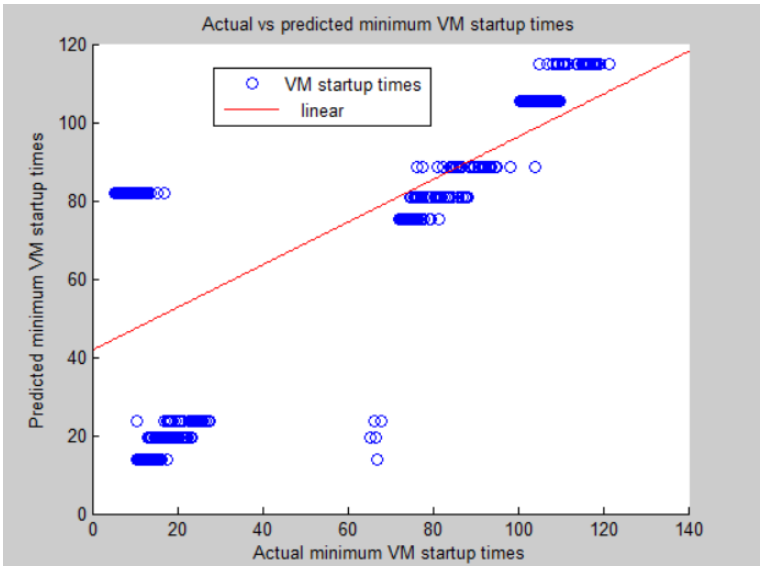
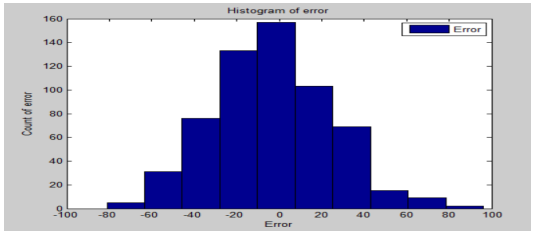
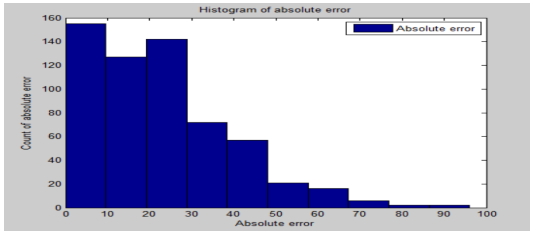


Figure A.1: Minimum VM startup times. Actual vs predicted values

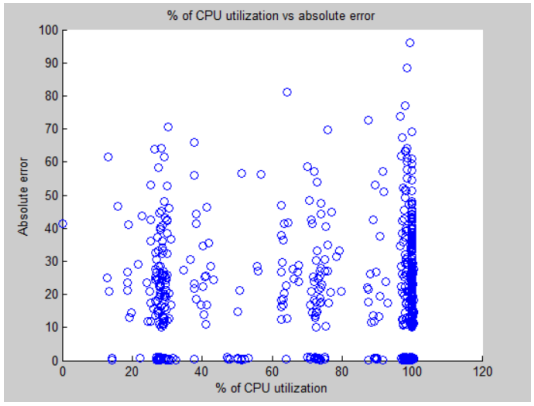


(a) Minimum VM startup times - Histogram of error

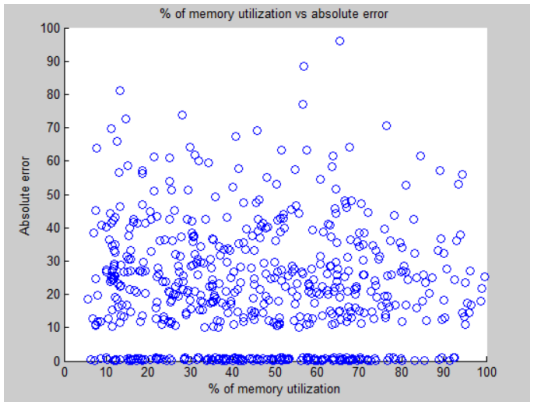


(b) Minimum VM startup times - Histogram of absolute error

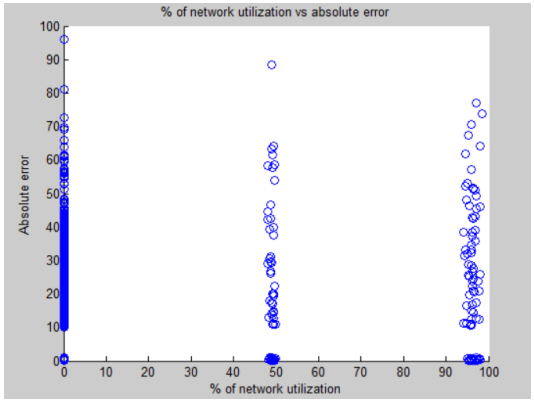
Figure A.2: Histograms - Minimum VM Startup times



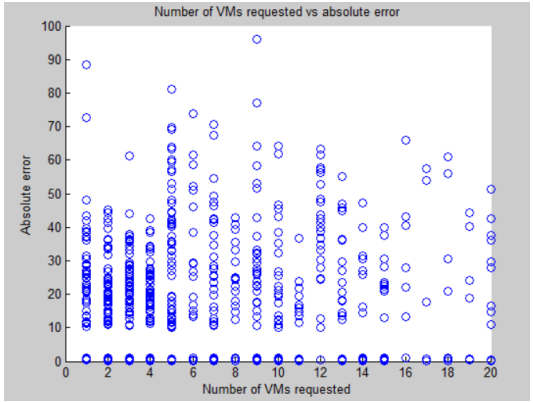
(a) Percentage of CPU utilization vs absolute error



(b) Percentage of memory utilization vs absolute error



(c) Percentage of network utilization vs absolute error



(d) Number of VMs requested vs absolute error

Figure A.3: Minimum VM Startup times - features vs absolute error

A.2 Plots of maximum VMST model prediction results for WRK3

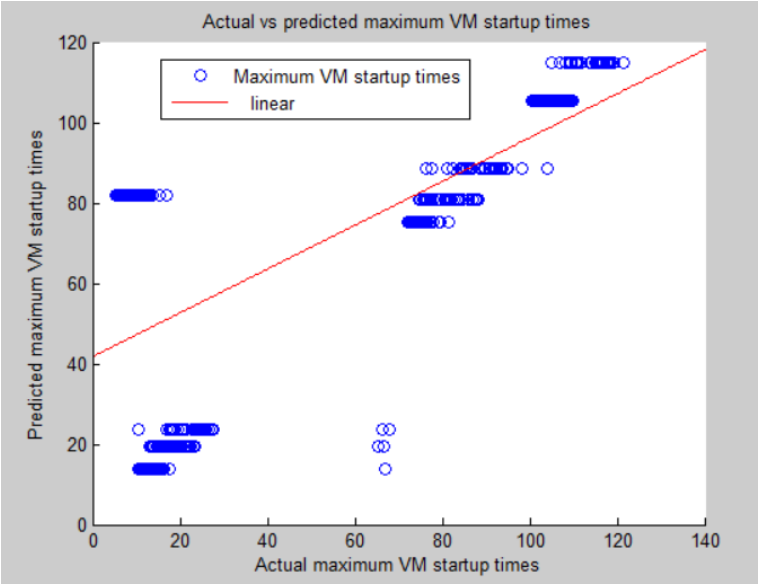
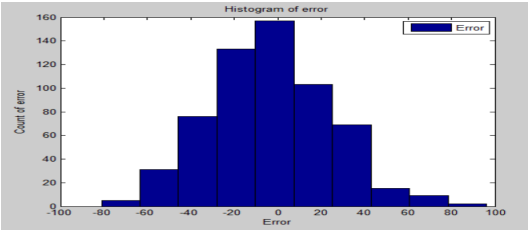
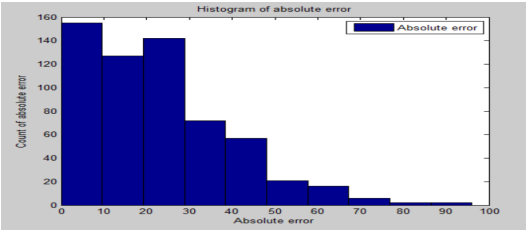


Figure A.4: Maximum VM startup times. Actual vs predicted values

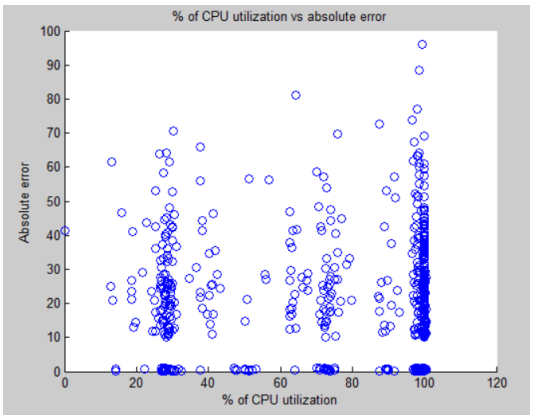


(a) Maximum VM startup times - Histogram of error

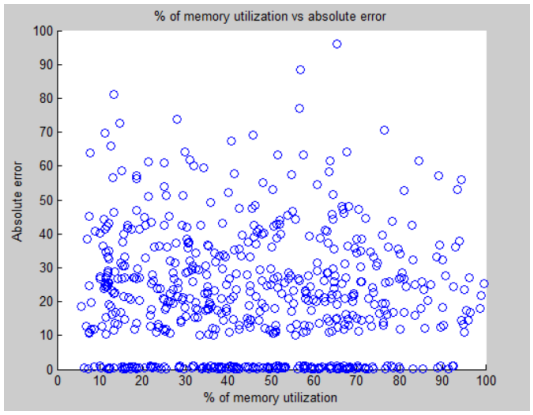


(b) Maximum VM startup times - Histogram of absolute error

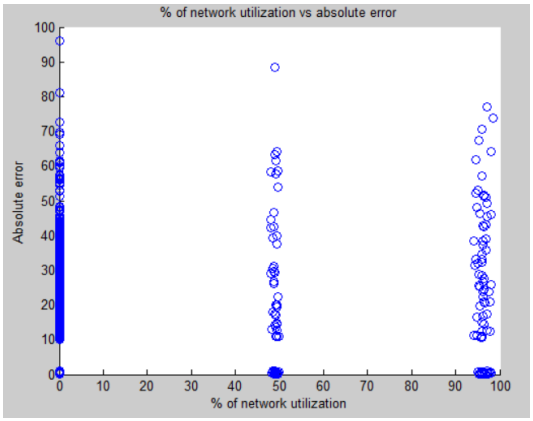
Figure A.5: Histograms - Maximum VM Startup times



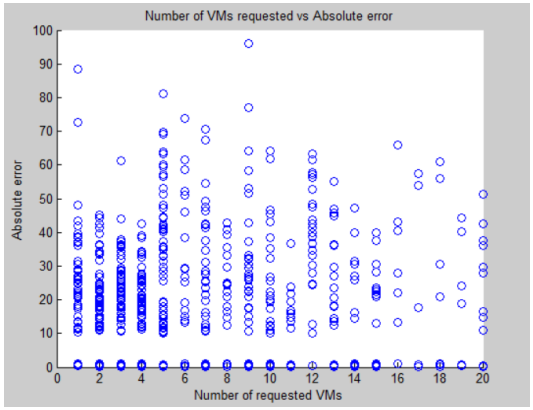
(a) Percentage of CPU utilization vs absolute error



(b) Percentage of memory utilization vs absolute error



(c) Percentage of network utilization vs absolute error



(d) Number of VMs requested vs absolute error

Figure A.6: Maximum VM Startup times - features vs absolute error

APPENDIX B

Publications

B.1 A QoS and Energy Aware Load Balancing and Resource Allocation Framework for IaaS Cloud Providers.

- Yatheendraprakash Govindaraju, Hector A. Duran-Limon, 2016. **A QoS and Energy Aware Load Balancing and Resource Allocation Framework for IaaS Cloud Providers.** In Proceedings of the 9th IEEE/ACM International Conference on Utility and Cloud Computing(UCC '16), Shanghai, China. 410–415.

The exponential growth of cloud based applications and the increased number of cloud users have given rise to new challenges for cloud service providers, especially for Infrastructure as a Service (IaaS) providers. This exponential growth of datacenters increases the energy consumption, along with its carbon footprints. Hence, better energy aware techniques not only reduce energy costs, but they are also helpful for reducing pollution in the environment. Some of the main challenges for cloud providers include increasing the resource utilization and minimizing energy consumption while maintaining the quality of service (QoS) offered to the users. There are many load balancing and resource allocation techniques proposed to handle these challenges. Out of these techniques, only a few consider QoS goals for IaaS service providers. However, none of these techniques includes service level agreement (SLA) parameters related to the Virtual Machine (VM) life cycle such as VM startup times. In this paper, we propose a novel approach to address this problem.

B.2 A Regression Tree Predictive Model for Virtual Machine Startup Time in IaaS clouds.

- Yatheendraprakash Govindaraju, Hector A. Duran-Limon, Efren Mezura-Montes, 2020. **A Regression Tree Predictive Model for Virtual Machine Startup Time in IaaS clouds.** Cluster Computing. The paper is "Accepted with Minor revisions" and Currently in the status of "revisions being processed". (JCR Q3 with impact factor of 1.851)

Cloud computing provides effective ways to rapidly provision computing resources over the Internet. For better management of resource provisioning, the system requires to predict service-level agreements (SLAs) such as virtual machine (VM) startup times under various conditions of computing resources. The VM startup time is an important SLA parameter, which can impact other SLA parameters such as service initiation time and VM scale out times. By predicting VM startup times, Cloud providers can improve Cloud users' expectations. However, little research has been carried out to define approaches to predict VM startup times. In this thesis, we propose a Regression Tree model for predicting average, minimum, and maximum VM startup times. To test the efficiency of our model, we implemented the model in an OpenStack test environment. The test results show that our model predicts VM startup times with an average accuracy of 91.81%.