
**UNIVERSIDADE FEDERAL DO CEARÁ**

**CENTRO DE TECNOLOGIA**

**DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE TELEINFORMÁTICA**

**DEBORAH MARIA VIEIRA MAGALHÃES**

# WORKLOAD MODELING AND PREDICTION FOR RESOURCES PROVISIONING IN CLOUD

**FORTALEZA**

**2017**

DEBORAH MARIA VIEIRA MAGALHÃES

# WORKLOAD MODELING AND PREDICTION FOR RESOURCES PROVISIONING IN CLOUD

Tese apresentada ao Curso de Programa de Pós-Graduação em Engenharia de Teleinformática do Departamento de Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Orientador: Prof. Dr. Danielo G. Gomes

Coorientador: Prof. Dr. Rajkumar Buyya

FORTALEZA

2017

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

M165w Magalhães, Deborah Maria Vieira.
Workload Modeling and Prediction for Resources Provisioning in Cloud / Deborah Maria Vieira Magalhães. – 2017.
100 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia de Teleinformática, Fortaleza, 2017.
Orientação: Prof. Dr. Danielo Gonçalves Gomes.
Coorientação: Prof. Dr. Rajkumar Buyya.

1. Cloud Computing. 2. Simulation. 3. Workload Modeling. 4. Workload Profiling. 5. Prediction. I. Título.

CDD 621.38

---

DEBORAH MARIA VIEIRA MAGALHÃES

WORKLOAD MODELING AND PREDICTION
FOR RESOURCES PROVISIONING IN CLOUD

Tese apresentada ao Curso de Programa de Pós-Graduação em Engenharia de Teleinformática do Departamento de Engenharia de Teleinformática do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Engenharia de Teleinformática. Área de Concentração: Sinais e Sistemas

Aprovada em: ___ /___ /_____

BANCA EXAMINADORA

---

Prof. Dr. Danielo G. Gomes (Orientador)
Universidade Federal do Ceará (UFC)

---

Profa. Dra. Rossana Maria de Castro Andrade
Universidade Federal do Ceará (UFC)

---

Prof. Dr. José Marques Soares
Universidade Federal do Ceará (UFC)

---

Prof. Dr. Paulo Romero Martins Maciel
Universidade Federal de Pernambuco (UFPE)

---

Prof. Dr. Daniel Cardoso Moraes de Oliveira
Universidade Federal Fluminense (UFF)

I dedicated this work to my parents and friends.

## ACKNOWLEDGEMENTS

To my parents, Olga and Sérgio, who taught me by example the value of commitment.

To my supervisor, Prof Danielo, for his patience and support that accompany me since the master's degree. Thank you for your friendship.

To my dear friend Andrea, for the courage and persistence that marked all battles that she fought by my side.

To friends that I met at GREat lab, I would like to thank Mauricio and Sergio for the partnership that raises thoughtful discussions contributing to my personal and professional growth. Thanks also to Adyson, Belmondo, Carlos André, Cristiano, Ismayle, Jefferson, Mariana, Neemias Gabriel, Paulo Arthur and Thalisson for providing me with a collaborative and pleasant research environment.

To Rodrigo for the constant friendship and support with images creation process. To Lana for taking me in AAMC and motivating me to see the world in a generous, creative and innocent way. My thanks also for the help with the references. To Italo for being such a generous and helpful person who gave me support in difficult times. To Rainara, for the support and encouragement that always contagious, besides the meaningful comments.

To Tici for the support and companionship, besides enhancing my sense of aesthetics and organization.

To Suelen, for being my philosophy master with dedication, generosity, and grace, awakening in me the desire to be a disciple. To Sandra for the friendship full of complicity.

To professors José Marques, Rossana Andrade, Paulo Maciel and Daniel de Oliveira for availability and excellent suggestions.

To professors Atslands and Paulo Armando for availability, tips, and shared materials and for inspiring me as a professional.

To the professors of the DETI, who contributed in a decisive way to my professional formation, especially Prof. Paulo Cortez for making the LisaMini scientific application available.

To the Clouds Lab who welcomed me during the sandwich period and contributed to my personal and professional life.

To the Coordination for the Improvement of Higher Education Personnel (Capes) for subsidizing part of the doctorate and to the National Counsel of Technological and Scientific Development (CNPq) to support my sandwich period in Clouds Lab.

To Darilu, Rute, Janaína and all GREat professionals for always support me and offer the infrastructure to carry out this work.

"É preciso saber o que aprendemos, ou seja, extrair das páginas dos livros aquelas ideias que, em sua medida, podem se incorporar a nós, sobretudo quando as aceitamos como válidas e necessárias. É preciso aprender a agir; saber equivocar-se dia a dia e recomeçar com a alegria de quem conquista um caminho novo. Mas, sobretudo, para além dos erros e desilusões, fazer com que se mova algo em nós e no mundo"

(Délia Steinberg Guzman)

## RESUMO

A avaliação de políticas de gerenciamento de recursos em nuvens computacionais é uma tarefa desafiadora, uma vez que tais ambientes estão sujeitos a demandas variáveis de usuários com diferentes perfis de comportamento e expectativas de Qualidade de Serviço (QoS). Fatores como *overhead* da camada de virtualização, indisponibilidade de dados e complexidade de cargas de trabalho altamente heterogêneas dificultam a modelagem e caracterização de aplicações hospedadas em nuvens. Neste contexto, caracterizar e modelar a carga de trabalho (ou simplesmente carga) é um passo importante na sistematização da análise e simulação do desempenho de políticas de gerenciamento dos recursos computacionais e uma estratégia particularmente útil antes da implantação física das nuvens. Nesta tese de doutorado, é proposta uma metodologia para modelagem e caracterização de carga visando criar perfis de utilização de recursos em Nuvem. Os padrões de comportamento das cargas são identificados e modelados sob a forma de distribuições estatísticas as quais são utilizadas por um controlador preditivo a fim de estabelecer a complexa relação entre a utilização dos recursos e a métrica de tempo de resposta. Desse modo, o controlador realiza ajustes no percentual de utilização do recursos a fim de manter o tempo de resposta observado pelo o usuário dentro de um limiar aceitável. Assim, nossa proposta apoia diretamente políticas de provisionamento de recursos cientes da Qualidade de Serviço (QoS). A metodologia proposta foi validada através de aplicações com características distintas: uma aplicação científica para o auxílio do diagnóstico de doenças pulmonares e uma aplicação Web que emula um site de leilões. Os modelos de desempenho computacional gerados foram confrontados com os dados reais através de métodos estatísticos gráficos e analíticos a fim de avaliar sua acurácia e todos os modelos apresentaram um percentual de erro inferior a 10%. A modelagem proposta para o controlador preditivo mostrou-se efetiva pois foi capaz de dinamicamente manter o tempo de resposta próximo ao valor esperado, com erro percentual absoluto médio ($MAPE$) $= 4.36\%$ sem violação de SLA.



**Palavras-chave:** Caracterização, Modelagem, Simulação, Cargas de Trabalho, Predição, Computação em Nuvem

## ABSTRACT

The evaluation of resource management policies in cloud environments is challenging since clouds are subject to varying demand coming from users with different profiles and Quality de Service (QoS) requirements. Factors as the virtualization layer overhead, insufficient trace logs available for analysis, and mixed workloads composed of a wide variety of applications in a heterogeneous environment frustrate the modeling and characterization of applications hosted in the cloud. In this context, workload modeling and characterization is a fundamental step on systematizing the analysis and simulation of the performance of computational resources management policies and a particularly useful strategy for the physical implementation of the clouds. In this doctoral thesis, we propose a methodology for workload modeling and characterization to create resource utilization profiles in Cloud. The workload behavior patterns are identified and modeled in the form of statistical distributions which are used by a predictive controller to establish the complex relationship between resource utilization and response time metric. To this end, the controller makes adjustments in the resource utilization to maintain the response time experienced by the user within an acceptable threshold. Hence, our proposal directly supports QoS-aware resource provisioning policies. The proposed methodology was validated through two different applications with distinct characteristics: a scientific application to pulmonary diseases diagnosis, and a web application that emulates an auction site. The performance models were compared with monitoring data through graphical and analytical methods to evaluate their accuracy, and all the models presented a percentage error of less than 10 %. The predictive controller was able to dynamically maintain the response time close to the expected trajectory without Service Level Agreement (SLA) violation with an Mean Absolute Percentage Error (MAPE) = 4.36%.



**Keywords:** Cloud Computing, Simulation, Workload Modeling, Workload Profiling, Prediction.

## LIST OF FIGURES


Figure 1 – Workload Modeling Proposal and Resource Management Scope. . . . . 22
Figure 2 – Prediction methods taxonomy (adapted from (LORIDO-BOTRAN *et al.*, 2014)). . . . . 35
Figure 3 – Model based Prediction Control Process (adapted from (CAMACHO; ALBA, 2013)). . . . . 38
Figure 4 – Workload modeling scenario (adapted from (FEITELSON, 2014)). . . . . 46
Figure 5 – Workload modeling process. . . . . 46
Figure 6 – Resource management sequence. . . . . 51
Figure 7 – MISO system. . . . . 52
Figure 8 – Workload generation diagram. . . . . 65
Figure 9 – Statistical analysis of number of instructions. (a) Histogram of Browsing profile (median of 100 rounds); (b) Number of instruction per user session for Apache and MySQL . . . . . 69
Figure 10 – Statistical analysis of instructions number. (a) Histogram of Bidding profile (median of 100 rounds); (b) Number of instruction per user session for Apache and MySQL . . . . . 70
Figure 11 – Exploratory analysis of the observed CPU utilization for a browsing user session. (a) Browsing profile: scatterplot of CPU utilization (b) Browsing profile: limited axis with interval between 600 and 650 s . . . . . 71
Figure 12 – Quantile-Quantile plots. (a) GL with mle method; (b) GEV with PWM method 73
Figure 13 – Comparison of histogram of observed number of instructions and probability density function of simulated number of instructions (a) Browsing profile; (b) Bidding profile. . . . . 75
Figure 14 – MISO Model (scenario 01): response time behavior with setpoint range [300ms-1000ms] . . . . . 77
Figure 15 – MISO Model (scenario 02): response time behavior with setpoint range [300ms-1850ms] . . . . . 77
Figure 16 – Impact of LisaMini execution in different instances. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate; (e) Disk Write Rate . . . . . 79

Figure 17 – Impact of LisaMini execution on A2 instance considering different geographic locations. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate . . . . . 80
Figure 18 – Impact of LisaMini execution on A1 instance considering different geographic locations. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate . . . . . 81

## LIST OF TABLES


Table 2 – Summary of related work. . . . . 31
Table 3 – Studies inclusion/exclusion criteria. . . . . 36
Table 4 – Private cloud: Physical sever and VMs specifications . . . . . 56
Table 5 – Experiment # 1 - RUBiS client configuration . . . . . 58
Table 6 – Experimental design. . . . . 61
Table 7 – Descriptive statistics: location, dispersion and shape. . . . . 62
Table 8 – MPC: experimental design parametrization. . . . . 67
Table 9 – Observed metrics statistics: number of instructions, CPU, memory, and disk utilization, and response time. . . . . 69
Table 10 – Distributions and parameters for number of instructions. . . . . 71
Table 11 – *MLE* parameters estimation of response time and CPU, memory and disk utilization metrics. . . . . 72
Table 12 – Kolmogorov-Smirnov test: number of instructions. . . . . 74
Table 13 – Kolmogorov-Smirnov test: response time and cpu, memory and disk utilization. 74
Table 14 – WMW test . . . . . 76
Table 15 – Estimated MSE error for transfer functions. . . . . 77
Table 16 – Experiment 1 - Distribution parameters of different instances . . . . . 83
Table 17 – Experiment 2 - Distribution parameters of different instances in different locations . . . . . 83
Table 18 – Kolmogorov-Smirnov test: Experiment 1 . . . . . 83
Table 19 – Kolmogorov-Smirnov test: Experiment 2 . . . . . 84

## LIST OF ACRONYMS

| | |
|---|---|
| ADF | Augmented Dickey-Fuller |
| CPU | Central Processing Unit |
| CT | Computerized Tomography |
| DVFS | Dynamic Voltage and Frequency Scaling |
| FFT | Fast Fourier Transform |
| GEV | Generalized Extreme-Value |
| GL | Generalized Lambda |
| GoF | Goodness-of-Fit |
| GPD | Generalized Pareto Distribution |
| GWD | Generalized Weibull Distributions |
| HF | Histogram Fitting |
| HPC | High-Performance Computing |
| HTTP | HyperText Transfer Protocol |
| IaaS | Infrastructure as a Service |
| IT | Information technology |
| JEE | Java Enterprise Edition |
| JVM | Java Virtual Machine |
| KPSS | Kwiatkowski-Phillips-Schmidt-Shin |
| KS | Kolmogorov-Smirnov |
| MAPE | Mean Absolute Percentage Error |
| MCC | Mobile Cloud Computing |
| MIMO | Multiple Input Multiple Output |
| MIPS | Million Instructions Per Second |
| MISO | Multiple Input Single Output |
| MLE | Maximum Likelihood Estimation |
| MPC | Model Predictive Control |
| MPS | Maximum Product of Spacing |
| MSE | Mean Squared Error |
| PaaS | Platform as a Service |
| PID | Proportional Integral Derivate |
| PWM | Probability Weighted Moments |

| | |
|---|---|
| Q-Q | Quantile-Quantile |
| QM | Quantile Matching |
| QoS | Quality de Service |
| QT | Queuing Theory |
| RUBiS | Rice University Bidding System |
| SaaS | Software as a Service |
| SISO | Single Input Single Output |
| SLA | Service Level Agreement |
| SLO | Service Level Objective |
| VM | Virtual Machine |
| VMM | Virtual Machine Manager |
| WMW | Wilcox Mann-Whitney |

**CONTENTS**


**1 INTRODUCTION** . . . . . 17
**1.1 Contextualization** . . . . . 17
**1.2 Objectives** . . . . . 19
**1.3 Research Questions and Methodology** . . . . . 19
**1.4 Contributions** . . . . . 21
**1.5 Scope** . . . . . 22
**1.6 Thesis Organization** . . . . . 23
**2 BACKGROUND AND RELATED WORK** . . . . . 25
**2.1 Resource Management in Cloud Computing** . . . . . 25
**2.2 Workload Profiling** . . . . . 26
***2.2.1 Challenges of Cloud Workload Modeling*** . . . . . 27
***2.2.2 Importance of Cloud Workload Modeling*** . . . . . 28
***2.2.3 Related Work in Cloud Workload Modeling*** . . . . . 29
**2.3 Modeling Techniques for Application Prediction** . . . . . 31
***2.3.1 Systematic Mapping*** . . . . . 34
***2.3.2 Importance of Cloud Workload Prediction*** . . . . . 35
***2.3.3 Challenges of Cloud Workload Prediction*** . . . . . 36
***2.3.4 Model Predictive Control (MPC)*** . . . . . 37
*2.3.4.1 Prediction* . . . . . *38*
*2.3.4.2 Optimizer* . . . . . *40*
*2.3.4.3 Control Law* . . . . . *41*
***2.3.5 Related Work in Model Predictive Control*** . . . . . 41
**2.4 Concluding Remarks** . . . . . 43
**3 CLOUD WORKLOAD MODELING AND PREDICTION** . . . . . 45
**3.1 Workload Modeling Methodology** . . . . . 45
***3.1.1 Experimental Design and Setup*** . . . . . 47
***3.1.2 Monitoring and Trace*** . . . . . 47
***3.1.3 Exploratory Analysis*** . . . . . 47
***3.1.4 Parameters Estimation*** . . . . . 47
***3.1.5 Goodness of Fit*** . . . . . 49
*3.1.5.1 Quantile-Quantile Plots* . . . . . *49*

*3.1.5.2 Kolmogorov and Smirnov Test . . . 50*
**3.2 Simulation . . . 50**
**3.3 Prediction . . . 51**
***3.3.1 Cloud Scenario . . . 51***
***3.3.2 Model Assumptions . . . 52***
**3.4 Concluding Remarks . . . 53**
**4 MATERIAL AND METHODS . . . 54**
**4.1 Monitoring and Tracing: RUBiS . . . 54**
***4.1.1 Testbed Configuration . . . 56***
***4.1.2 Servers Metrics . . . 56***
***4.1.3 Client Metrics . . . 57***
***4.1.4 Experimental Design . . . 58***
**4.2 Monitoring and Tracing: LisaMini . . . 58**
***4.2.1 Testbed Configuration . . . 59***
***4.2.2 Metrics . . . 59***
***4.2.3 Experimental Design . . . 60***
*4.2.3.1 Monitoring Experiments . . . 60*
**4.3 Workload Profiling . . . 61**
***4.3.1 Exploratory Analysis . . . 61***
***4.3.2 Parameters Estimation . . . 62***
***4.3.3 Goodness of Fit . . . 63***
**4.4 Model Simulation and Validation . . . 64**
***4.4.1 Workload Generation . . . 64***
***4.4.2 Simulation Validation . . . 65***
**4.5 Prediction and Control . . . 65**
**4.6 Concluding Remarks . . . 67**
**5 RESULTS AND DISCUSSION . . . 68**
**5.1 RUBiS Profiling . . . 68**
***5.1.1 Statistical Analysis . . . 68***
***5.1.2 Parameters Estimation . . . 70***
***5.1.3 Goodness of Fit . . . 72***
**5.2 Simulation and Validation . . . 75**

**5.3** **Prediction and Control** . . . 76
**5.4** **LisaMini Profiling** . . . 78
***5.4.1*** ***Monitoring*** . . . 78
*5.4.1.1* *Experiment #1: Different instances* . . . *78*
*5.4.1.2* *Experiment #2: Different locations* . . . *79*
***5.4.2*** ***Performance Modeling*** . . . 82
***5.4.3*** ***Godness of Fit*** . . . 82
***5.4.4*** ***Discussion of Results*** . . . 84
**5.5** **Concluding Remarks** . . . 85
**6** **CONCLUSION** . . . 87
**6.1** **Research Questions Analysis** . . . 87
**6.2** **Publications** . . . 90
**6.3** **Future Work** . . . 90
**BIBLIOGRAPHY** . . . 92
**GLOSSARY** . . . 100

# 1 INTRODUCTION

This thesis proposes a methodology for Workload characterization and modeling aiming at resource utilization profile in cloud environments. Section 1.1 contextualizes this work by providing an overview of the resource provisioning problem in cloud computing. Section 1.2 details the thesis goals. Section 1.3 presents the research questions and discusses the methodology employed through this work. Section 1.4 presents the thesis contributions and Section 1.5 states the thesis scope. Finally, Section 1.6 lists the remaining structure of this document.

## 1.1 Contextualization

As cloud computing has grown in size, reach and popularity, it has supported various types of applications such as Web serving, scientific High-Performance Computing (HPC), social networking, and gaming (BUYYA *et al.*, 2009). Such applications are subjected to different Quality of Service (QoS) aspects which are specified in a Service Level Agreement (SLA) negotiated between cloud providers and customers. The failure to comply with a SLA could compromise the QoS and result in penalties to the cloud provider (AMIRI; MOHAMMAD-KHANLI, 2017).

The development of computational Resource Management policy that supports QoS is a challenging task (MAGALHÃES *et al.*, 2014). To evaluate the performance of these policies is even more challenging (COUTINHO, 2014) since clouds observe varying demand, its physical infrastructure has different sizes, software stacks, and physical resource configurations, and users have different profiles and QoS requirements (BUYYA *et al.*, 2009). Therefore, to evaluate these policies under same conditions several times is not a trivial task.

In this context, the Workload Modeling and characterization enable a systematic analysis and simulation of resource management policies performance, which brings benefits to cloud providers, researchers and advanced professionals working in the area. The application models allow for adjustments to fit particular situations, controlled modification of parameters, repetitions of evaluation conditions, and generalization of Pattern found in the applications (FEITELSON, 2014).

For cloud providers, the evaluation and simulation of resource management policies allow for the improvement of their QoS systems before the real deployment in large scale environments. Moreover, the simulation of workloads based on realistic scenarios enables the

production of trace logs, which are scarce in some level of cloud environments due to business and confidentiality concerns (MORENO *et al.*, 2013) (CHEN *et al.*, 2010).

However, workload modeling and characterization are especially challenging when dealing with a highly dynamic environment, such as cloud data centers, because (i) heterogeneous hardware is present in a single data center; (ii) the virtualization layer involves overheads caused by I/O processing and interactions with the Virtual Machine Manager (VMM); and (iii) complex workloads are composed of a wide variety of applications submitted at any time, and with different characteristics and user profiles. All these factors contribute to a lack of methodologies to characterize the different behavioral patterns of cloud applications. Therefore, researchers and providers will benefit from a well-defined methodology which contains a process to achieve realistic models. They can use and extend it to generate their resource management policies and analysis.

Application modeling increases the understanding of typical patterns presented in the Cloud workloads and leads to a better management of available resources (MORENO *et al.*, 2013; KAVULYA *et al.*, 2010). From the workload characterization, performance models can be built to help us to answer important questions such as: How can cloud usage levels be affected by user behavior? How can the energy efficiency of a cloud be improved as long as the quality of service offered to users is maintained?

To ensure scalability, flexibility, and effectiveness, cloud providers need quickly to provide computing resources in accordance with the dynamic applications demand (KHAN *et al.*, 2012; COUTINHO *et al.*, 2013; BRAGA *et al.*, 2014). To this end, performance models are critical to improving resource management since they allow to continuously predict the workload performance for a particular hardware and software stack (MANVI; SHYAM, 2014).

A prediction process begins with the construction of a performance model, comprising the behavior pattern exhibited by the application over the time. Several studies in the literature have presented different techniques to predict the application behavior, such as Linear Regression (RIZVANDI *et al.*, 2012), Support Vector Regression (SVR) (YANG *et al.*, 2012), Hidden Markov Chain (KHAN *et al.*, 2012), Artificial Neural Networks (ANN) with interactive training model (KUNDU *et al.*, 2010), and Bucket Method (DENG *et al.*, 2012). After the construction of a predictive model, the Accuracy of the predictor must be evaluated. The system's ability to estimate the resource demand in subsequent time intervals is critical to cloud providers because it directly helps to adjust the resources dynamically to comply the QoS requirements.

## 1.2 Objectives

This thesis focuses on workload modeling and prediction. Consequently, how an application uses a computational resource (CPU, disk, memory) and how this usage varies according to the time are described and used to build realistic performance models. These models are used to estimate the resource demand of each application deployed in cloud environments. This way, the computational resources can be allocated in an efficient manner.

Therefore, the **main goal of this thesis is to develop a workload profiling methodology to produce accurate performance models that offer a good approximation to the real scenario**. Such methodology can be used to generate different synthetic loads relating to relevant applications in cloud computing and support analysis and simulation of resource utilization in cloud environments.

Since the elasticity is critical to cloud providers, the second goal of this thesis is the development of a predictive model capable of capturing the complex relationship between the resource utilization profile of the application and its performance metrics (execution time, response) in order to guarantee the QoS. Such model should estimate this relationship in a future time to capture the workload variation and guide resource allocation ensuring the cloud environment elasticity.

## 1.3 Research Questions and Methodology

Our main research hypothesis is that **well-defined methodology to generate workload performance models in a cloud scenario combined with an efficient prediction technique could improve the Infrastructure as a Service (IaaS) resource provisioning while maintaining the desired QoS**. To guide this investigation, four fundamental research questions associated with the workload monitoring and prediction were suggested as follows.

**Research Question #1**: How do different user profiles impact on resource utilization behavior?

**Research Question #2**: Do different VM instances and geographic locations affect the resources usage patterns sufficient to justify specific models for each of them?

**Research Question #3**: How to develop predictive models of resources to guide the cloud elasticity so that the workload variation has the minimum impact on performance and availability?

**Research Question #4**: How to develop models that are able to predict application performance considering multiple parameters such as processor, memory, network and disk usage to comply with QoS?

With respect to the methodology employed in this thesis, we carried out the 5 stages described below:

1. Literature review: the literature is investigated to raise the motivation and key challenges to workload modeling and prediction. From this point, significant gaps are identified: a lack of methodologies to characterize the different behavioral patterns of cloud applications, and a lack of detail regarding the controller parameterization, which compromises the reproducibility of the proposals and; and a lack of formal relationship description between the input and output of model predictive controls;
2. Monitoring: firstly, a design experiment is defining, including decisions as workloads, metrics, factors, environment setup, etc. Later, the monitoring scenarios have been implemented with a view to evaluating the impact of different factors on user profiles, geographical locations, and VM instance types. And, finally, the traces are recorded and stored for later analysis;
3. Profiling: the modeling process is performed in three steps as illustrated in Figure 1,
    a) Exploratory Analysis, which analyzes the data characteristics and determines the candidate distributions to represent the model;
    b) Parameter Estimation, uses estimation methods to set the parameters of the distributions, selected in advance, based on the collected samples;
    c) Good of Fitness Test, which are graphical and analytical methods to evaluate whether the distributions and their respective parameters provide satisfactorily approximation to the empirical data;
4. Prediction: a Model Predictive Control (MPC) is used to estimate the number of physical resources needed to maintain the predicted output close to the application SLA. To predict the application performance is used a linear technique called State Space Model;
5. Discrete-event simulation: the performance workload models are implemented as an extension of a CloudSim simulator (CALHEIROS *et al.*, 2011). After, validation is performed to show that the simulator is able to represent the observed data accurately. The simulation accuracy is evaluated through graphical and statistical hypothesis test approaches.

## 1.4 Contributions

Workload characterization and modeling problems have been addressed through the creation of simple and generic models to generate synthetic workloads (FEITELSON, 2014). Several papers have focused on resource utilization imposed by jobs, tasks and requisitions (CHEN *et al.*, 2010; GANAPATHI *et al.*, 2010; KAVULYA *et al.*, 2010; GROZEV; BUYYA, 2013; MORENO *et al.*, 2013).

From these related work, we claim the workload characterization and modeling can provide an understanding of non-trivial tradeoffs and behavior prediction in cloud scenarios. However, frequently, we note that such works did not adequately justify their choices. Unfortunately, this makes the experiment reproducibility difficult and, consequently, their conclusions as well. In the expectation of improve the researh reproducibility, our main contribution is a methodology for workload characterization and modeling aiming at resource utilization profile in cloud environments. This methodology can be used to generate accurate and easily manageable models that match behavior patterns for different user profiles, geographic locations, and VMs configurations. This well-defined methodology contains steps and justifications to achieve the distributions and parameters derived from the application analyses so that our models can be easily reproduced and reused.ns are modeled in the form of statistical distributions. Therefore, the patterns fluctuate based on realistic parameters in order to represent dynamic environments. A model validation is provided through visual and analytical methods with a view to show that the model adequately represents the observed patterns.

We can also highlight some secondary contributions as follows:

1. Implementation of the proposed models as an extension of the CloudSim simulator (CALHEIROS *et al.*, 2011), making the model available to researchers' practical use, supported by the cloud data center environment abstraction that CloudSim provides;
2. A case study supporting Central Processing Unit (CPU) utilization forecasting associated with Dynamic Voltage and Frequency Scaling (DVFS) management in order to improve energy saving;
3. A predictive model that fits the complex relationship between an application resource utilization profile and its performance metrics in order to support resource allocation policies aware of QoS.

Figure 1 illustrates the thesis contributions in a resource management context. Our major contribution is highlighted in the workload modeling box which can divide into the core

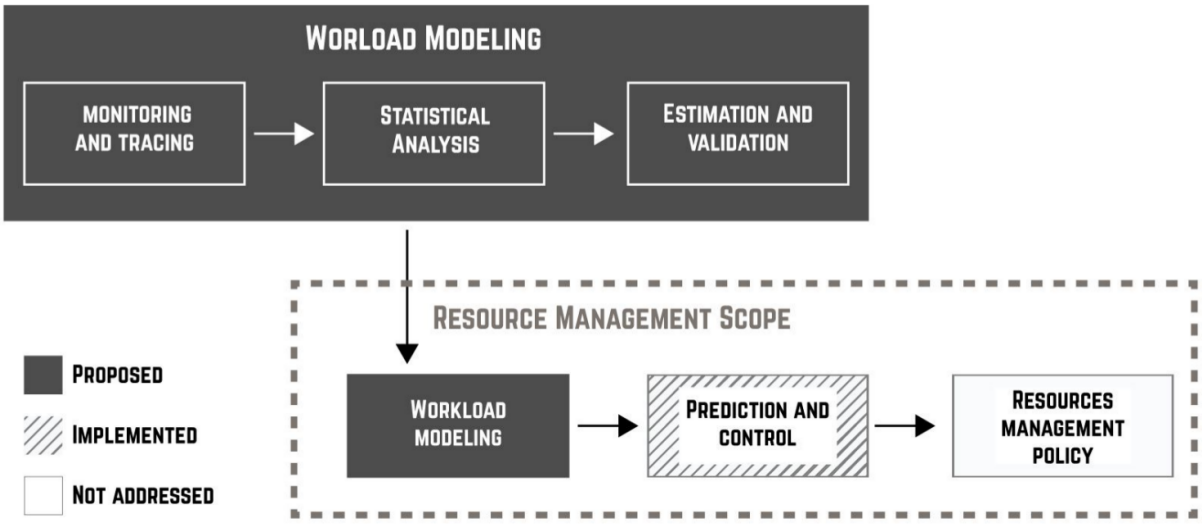



Figure 1 – Workload Modeling Proposal and Resource Management Scope.

steps: (i) data monitoring and tracing; (ii) statistical analysis, which identifies the workload patterns, and (iii) estimationn and validation, it estimates the statistical distribution parameters and, finally, evaluate with the statistical model offers a good fit to the empirical data. Therefore, the statistical models generated from the methodology proposed are inputs to implementation of the model predictive control (secondary contribution) which adjusts the resource utilization values to maintain the application response time within an acceptable QoS. Such estimated values can be directly used by the resource provisioning policy to guide its decision making, but provisioning policy is not addressed in this thesis.

## 1.5 Scope

Some concepts and aspects identified in the cloud computing literature have not been addressed in this thesis. However, they are related and discussed throughout the text. The main focus of this thesis is on the workload characterization and modeling, and the secondary one is an application performance prediction and control, as illustrated in Figure 1. It is an important highlight that the workload concept has different interpretations in the literature. In this thesis, we assume the workload as the Virtual Machine (VM) resource utilization (CPU, memory, disk) and the application performance metric is the response time. The following items let clear what is out of the thesis scope:

- Autonomic Computing: despite this thesis support the monitoring and analysis phases of autonomic loop, this work do not consider autonomic computing;
- Multi Site Clouds: multi site clouds consist of a vast number of small data centers spread

across a geographic area, appealing to service providers who already have distributed development facilities interconnected by high-speed networks (COUTINHO, 2014). We do not consider multi site clouds;

- Provisioning: the model predictive control estimates the number of resources required to run the application to meet user expectations. Therefore, this work supports resource allocation strategies but does not implement the strategy itself, as illustrated in Figure 1. A provisioning procedure can be attached at any time to this work without impacting it;
- Application independent: here two different applications were used to validate the workload modeling methodology, (i) a scientific application for pulmonary disease diagnoses (Section 4.1) and, (ii) a Web application that simulates an action site (Section 4.2). However, our proposal for generating statistical models that represent the applications can be applied to any workload;
- Deployment: a cloud environment provides basically three service models. The SaaS model provides specific purpose software systems that are available to users over the Internet and accessible through a thin client interface. The PaaS model provides an operating system, programming languages and development environments for the applications, aiding the implementation of software systems. And, finally, the IaaS model makes it easier and more accessible to provide resources such as servers, network, storage, and other key computing resources to build an on-demand application environment (COUTINHO, 2014). This thesis is dedicated to the IaaS implementation model, not addressing aspects of the other service models.

## 1.6 Thesis Organization

The remainder of this dissertation is organized as follows: Chapter 2 presents the theoretical references that guide the development of this thesis, as well the literature papers related to our proposal. In addition, Chapter 2 highlights the importance and challenges of workload modeling and prediction in the cloud. Chapter 3 presents the proposed workload modeling methodology, as well as the validation and simulation approaches of the models achieved via methodology. In addition, it discusses important parameters of the predictive control that are critical to fit a realistic scenario. Chapter 4 presented the experimental design of the monitoring experiments and described how the proposed methodology was implemented and how the application model validation was conducted as an extension of the CloudSim simulator.

Chapter 5 presents the results achieved in the monitoring, profiling, prediction and, validation and simulation phases of this work. Finally, in Chapter 6, the main thesis conclusions are presented, as well as the future perspectives to improve the results achieved in this work and to support new proposals related to workload profiling and prediction in cloud scenarios.

# 2 BACKGROUND AND RELATED WORK

In this chapter, we discuss relevant concepts in the context of resource management in cloud environments. In addition, the importance and challenges of workload profiling and prediction for efficient resource provisioning are presented. We also present some related work, their differences and similarities to this thesis.

## 2.1 Resource Management in Cloud Computing

Cloud computing can be defined as a model that enables access over the network, on demand to a set of configurable computing resources (networks, servers, storage devices, applications, and services) that can be quickly acquired and released with a minimal management effort or service provider interaction (YOUSEFF *et al.*, 2008).

Clouds provide scalability, quality of service, and inexpensive computing infrastructure that can be accessed via a simply and pervasive way (ROSSI *et al.*, 2016). Further to fast elasticity and broad access, cloud computing relies on the characteristics of self-service on demand, resource pooling, and measured service. In addition, it has three service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) (MELL; GRANCE, 2009).

This thesis supports the elasticity characteristic of IaaS, where the cloud provider offers VMs, storages, firewalls, load balancer, and network devices (MANVI; SHYAM, 2014). Elasticity is related to the ability of the system to automatically adapt to changes in demand for resources by provisioning and deprovisioning them at runtime (COUTINHO *et al.*, 2015; HERBST *et al.*, 2013). As the elastic modalities in cloud environments, there is elasticity in two directions: vertical and horizontal. Vertical elasticity (scale-up), usually consists of adding more processors or memory on the same host increasing its capacity. Horizontal elasticity (scale-out) refers to the addition of more hosts to the environment. This thesis directly supports the horizontal.

The dynamic cloud demand may result in under-provisioning of resources that promotes the violation of SLA, causing dissatisfaction of user and reduced profit. On the other hand, overprovisioning increases operating costs by wasting energy and computing resources. Therefore, the provider must have an efficient resource management policy to ensure the minimum amount of resources required to comply with the SLA (AMIRI; MOHAMMAD-KHANLI,

2017). In this context, resource management for IaaS leads to benefits, such as scalability, quality of service, optimal utility, and cost effectiveness.

Dynamic resource allocation in the cloud depends on virtualization technologies. Such technologies, despite the progress, take around 10 minutes [1] to create a VM and make it fully operational. And, the response time to changes in workload in order to achieve the desired performance is fundamental to the elasticity (COUTINHO *et al.*, 2015). Therefore, the prediction of future demand is essential for the implementation of rapid elasticity and the effective provisioning of resources. The prediction method estimates the variation in future demand, giving the resource manager sufficient time to provide computing resources properly (AMIRI; MOHAMMAD-KHANLI, 2017).

## 2.2 Workload Profiling

The problems of workload characterization and workload modeling were addressed previously by creating a simple and general model to generate synthetic workloads similar to real systems (FEITELSON, 2014). The modeling is obtained via monitoring of the infrastructure during workload execution. The idea is to find patterns of behavior on the collected data. In the context of this work, workload modeling is used for analysis and simulation of resource utilization in cloud environments. This type of evaluation is critical for resource provisioning so as to fulfill SLA. For this type of evaluation, good characterization and understanding of workload burstiness are crucial.

An important aspect that should be considered during model creation is the degree of detail. A model with the fewest number of parameters possible is easier to manage. At the same time, it must be able to fully characterize all the important attributes of workloads.

When the collected traces are directly used to represent a workload, greater precision can be obtained. However, those results can not be extrapolated to other system sizes or environment settings, because the traces show a static picture of the system, leading to results limited to the scale of the used infrastructure . If there is a need to capture different software stacks, physical resources, and their configurations, using traces becomes expensive, since much time and money need to be invested in the configuration of various software stacks and changing the configuration and number of the machines.

Models appear to circumvent traces limitations through the adjustment, where it is

[1] https://serverfault.com/questions/43884/speeding-up-launch-of-amazon-ec2-windows-instances

possible to adjust the workload to fit the particular situation. Thus, the configuration of physical resources and software stacks are model parameters, and a new workload is generated according to these parameters. Furthermore, using a model rather than data traces brings others advantages, such as (FEITELSON, 2014): (i) controlled modification: the parameters of the model can be modified, one at a time, in order to investigate their influence, while other parameters are kept constant, (ii) repetitions: it is possible to repeat experiments under similar, but not identical, statistical conditions, (iii) additional features: missing attributes in the traces can be inserted into models. Therefore, the models prevent the need for application installation and configuration, and they provide a controlled input for researchers evaluate their theoretical mechanisms.

### *2.2.1 Challenges of Cloud Workload Modeling*

The modeling and characterization of workloads are especially challenging when dealing with a highly dynamic environment such as a cloud, for various reasons, as discussed following:

1. Diversity of Hardware Platforms: Information technology (IT) managers update about 20% to 25% of their platforms every year (MULIA *et al.*, 2013), resulting in the combination of different hardware in the same data center. In these computational infrastructures, models are used to predict application performance in order to improve the resource management. The needs of applications for resources vary over time, promoting VMs relocation to maximize the IT infrastructure performance. When a VM is migrated from one host to another, the performance model can only be applied if the target host has the same hardware as the source host, since the same workload shows different performances according to physical resources capacity of the machines that host the applications. Moreover, other VMs co-located in the target host also impact the workload performance. Given the environment heterogeneity, in many cases, the performance model needs to be completely rebuilt. Another solution is to create *a priori* models of application performance for all target hosts (KUNDU *et al.*, 2010).
2. Business and Confidentiality: due to business and confidentiality reasons there are insufficient cloud trace logs available for analysis. Thus there is a lack of methodologies to characterize the different behavioral patterns of cloud applications (MORENO *et al.*, 2013; CHEN *et al.*, 2010). Some efforts in this direction were initiated, for example, by the availability of Google TraceLog (REISS *et al.*, 2011) and Yahoo!, enabling approaches to

analysis and characterization of these data (ZHANG *et al.*, 2011; MISHRA *et al.*, 2010; WANG *et al.*, 2011; KAVULYA *et al.*, 2010);

3. Workload Complexity: a cloud hosts a wide variety of applications that could be submitted at any time, with different characteristics and user profiles, which have heterogeneous and competing QoS requirements (BUYYA *et al.*, 2009). This leads to complex workloads depending on users behavior and resource consumption. Thus, it is challenging to predict workload patterns over time. MULIA *et al.* (2013) proposed a set of cloud workload categories and established a relationship among them and critical computer resources in order to detect and reduce resource contention. The authors argue that the list of cloud workload categories is a tool for common communication among service providers, IT managers, and cloud users, what facilities the SLAs, purchase capital, and future computer architecture design decisions;
4. Virtualization Layer: the virtualization layer causes an overhead caused by I/O processing and interactions with the VMM. As a result, additional resources are needed. This overhead depends on the used hardware platform. Thus, it is essential to consider these factors to model workloads in order to improve estimation of application resources requirements when they are consolidated in virtual environments. In this context, WOOD *et al.* (2008) proposed an approach to estimate the CPU requirements of applications when they are transferred to a virtual environment.

### *2.2.2 Importance of Cloud Workload Modeling*

Workload modeling increases the understanding about the typical cloud workload patterns and leads to more informed decisions to better manage resources (MORENO *et al.*, 2013; KAVULYA *et al.*, 2010). From the workload characterization, performance models can be constructed to support research domains, like energy-efficiency and resource management and answer important research questions, such as: how are overall cloud data center utilization levels affected by user behavior? How can cloud data center energy efficiency be improved as long as the QoS offered to the users is maintained?

To ensure scalability, flexibility, and cost-effectiveness, cloud providers need to provide computing resources according to the dynamic applications demands (KHAN *et al.*, 2012). To this end, performance models are critical to improving resource management because they allow predicting the performance of an application for a given set of hardware resources

and software stack continuously.

The prediction process starts with the building of the performance model, comprising the behavior patterns exhibited by the application over time. fter the construction of the prediction model, the predictor accuracy is evaluated. Thus, providers can use the performance model to predict how workload patterns will change, and dynamically scale resources to meet the expected demand.

In addition, workload modeling in cloud computing gives rise to performance analysis and simulation, which brings benefits to cloud providers and researchers as it allows: (i) the evaluation through simulation of resource management policies allows the improvement of cloud providers systems QoS; (ii) the evaluation of these policies without deployment and execution in the expensive large scale environments and (iii) the simulation of realistic cloud environments with controlled modification, adjustment, repetition and additional features. The use of simulation models enables the production of trace logs based on realistic scenarios, filling a gap identified in this area (MORENO *et al.*, 2013). For example, CHEN *et al.* (2010) used several combinations of parameters in a simulated Hadoop application with the purpose of generating different workloads that arise in a real cloud production environment.

### *2.2.3 Related Work in Cloud Workload Modeling*

The conflict of priorities between cloud providers, aiming high resource utilization with low operating costs, and MapReduce (DEAN; GHEMAWAT, 2008) applications users addressing small execution time, led to characterization and modeling of this application type (CHEN *et al.*, 2010; GANAPATHI *et al.*, 2010; KAVULYA *et al.*, 2010). CHEN *et al.* (2010) developed a tool for generating realistic workloads with the goal of analyzing the tradeoff between latency and resources utilization. In view of this, they used traces from one dedicated Hadoop cluster at Facebook. These traces are modeled through statistical distributions that mimic the workload characteristics. However, the traces do not contain important information such as skewness, which is crucial in the choice of distributions that represent the data, and resource utilization, which is essential for the development of the proposed work. In the attempt to obtain resource utilization data, statistical models were used to generate the job stream processed in the environment provided by Amazon's Elastic Cloud Computing (EC2). However, the authors do not present information concerning the distributions, parameters and Goodness-of-Fit (GoF) tests. Thus, the reproduction of the model by other researchers is not possible.

GANAPATHI *et al.* (2010) proposed a model to predict the execution time of MapReduce jobs in order to maximize performance while minimizing costs. A workload generator based on statistical models was used to guide the prediction. However, the authors did not consider the full distribution to build the model. Instead, distribution is estimated from the 1st, 25th, 75th and 99th percentiles, causing information loss that compromises the accuracy of the model.

KAVULYA *et al.* (2010) characterized resource utilization patterns and sources of failures to predict job completion times in MapReduce applications. They did not offer an analysis of data justifying the distributions used. Furthermore, they use only the method of Maximum Likelihood Estimation (MLE), which is quite sensitive to outliers. Our approach uses different estimation methods, and the results show that the estimator has great influence on the predictor accuracy. KAVULYA *et al.* (2010) presented the Kolmogorov-Smirnov (KS) values with no critical value. Thus, it is not clear which set of distributions and parameters provide a good fit to the modeled data. Also, the values of the KS presented lose representativeness, since no critical value was defined. Thus, it is not clear which distributions and their parameters provide a good fit to the modeled data.

Most work discussed in this section focused on resources utilization imposed by jobs, tasks, and requests. However, different types of users directly impact on the workload present in the cloud. In view of this, MORENO *et al.* (2013) and GROZEV; BUYYA (2013) created models based on resource utilization and users behavioral patterns. Moreno et al. (MORENO *et al.*, 2013) proposed a new approach for characterizing the Google trace log in the context of both user and task in order to derive a model to capture resource utilization patterns. As in our study, the authors developed the model in the CloudSim simulator and contrasted the logged data against simulation experiments. However, although the model parameters are presented, estimation methods and the results of the GoF tests are omitted. These factors compromise the use of the model by other researchers.

Grozev and Buyya (GROZEV; BUYYA, 2013) also made use of the RUBiS workload and implemented the model in the CloudSim. The CPU load is modeled in terms of an average number of instructions required for a user session. However, the average is not robust, which makes it easily influenced by peak usage. Also, the authors adopt a non-statistical approach for modeling. Thus, they did not analyze dispersion and shape measurements, which are relevant for a statistical workload characterization.

Table 2 – Summary of related work.

| Authors | Modeling Approach | Application Type | Exploratory Analysis | Parameter Estimation | GoF Test |
|---|---|---|---|---|---|
| Chen et al. (CHEN *et al.*, 2010) | Distribution Analysis | MapReduce | Yes | No | No |
| Ganapathi et al. (GANAPATHI *et al.*, 2010) | Distribution Analysis | MapReduce | No | No | No |
| Kavulya et al. (KAVULYA *et al.*, 2010) | Distribution Analysis | MapReduce | Yes | Yes (single) | Yes |
| Moreno et al. (MORENO *et al.*, 2013) | Distribution Analysis & Cluster | MapReduce | Yes | No | No |
| Grozev and Buyya (GROZEV; BUYYA, 2013) | Analytical Model | Web | No | No | No |
| **Present Work** (MAGALHAES *et al.*, 2015) | **Distribution Analysis** | **Web** | **Yes** | **Yes (multi)** | **Yes** |

From these related work, we observe that the application characterization and modeling support researchers and cloud providers in understanding complex tradeoffs and predicting applications behavior. This understanding leads to more efficient techniques for resources management. However, the lack of a well-defined methodology, containing steps to achieve distributions, estimate parameters, and GoF tests, prevents models reproduction and usage. A summary of the related work is presented in Table 2.

## 2.3 Modeling Techniques for Application Prediction

Amiri and Mohammad-Khali (AMIRI; MOHAMMAD-KHANLI, 2017) and LORIDO-BOTRAN *et al.* (2014) proposed taxonomies to classify methods capable of to predict the behavior of a different application in order to support auto-scaling in the context of cloud computing. Such taxonomies have in common three main groups as depicted in Figure 2: Queuing Theory (QT), Machine Learning and Control Theory.

Queuing models have been used to model the Internet and traditional servers applications. In a simple elastic cloud application scenario, a single queue represents the load balancer that dispatches requests to $n$ VMs. In a more complex scenario (multi-tier applications), a queueing network can be used, where each tier is a queue with one or multiple servers. A queue system consists of a fixed architecture and a set of known parameters such as inter-arrival time distribution, service time distribution, and a number of servers. The main limitation of QT models is the failure to adapt to application behavior changes; consequently, the model need be recomputed (with analytical or simulation-based tools) when there are changes in either the architecture or the parameters. In addition, solving nonlinear equations at runtime is a complex

and costly task. Thus, QT is an expensive alternative for dynamic ambients. As an advantage, QT technique does not require a training phase (LORIDO-BOTRAN *et al.*, 2014).

Machine learning techniques are applied to dynamically construct the model of resource consumption under a specific amount of workload. In this way, different applications can utilize the auto-scalers without customized settings and preparations. They are also more robust to changes and can self-adaptively adjust the model on the fly in the presence of notable events. In order hand, machine learning approaches take time to converge to a stable model and thus causes the auto-scaler to perform poorly during the active learning period. Certainly, the application performance is affected in this process. Furthermore, the time that is taken to converge is hard to predict and varies case by case (QU *et al.*, 2016).

Online machine learning in the auto-scalers context can be divided into two types(see 2): reinforcement learning, and regression. The reinforce learning algorithm's target is to generate a table specifying the best provision or deprovision action under each state. The learning process is similar to a trial-and-error approach. The learning algorithm chooses an individual operation and then observes the result. If the result is positive, the auto-scaler will be more likely to take the same action next time when it faces a similar situation.

Regression determines the relationship among variables via a function based on observed data. Finally, it uses this relation to making predictions. In the context of resource estimation, auto-scaler can record system utilization, application performance, and the workload for regression. As the training proceeds and more data are available, the predicted results also become more accurate. Although, regression requires the user to determine the function type (linear or quadratic). In the case of auto-scaling Web applications, it is usually safe to assume a linear function [CHENHAO QU, 2016]. In order to achieve an accurate prediction, the model should be able to extract all workload behavior patterns. Most of the machine learning techniques, such as Neural Network and Markov Model, are based on fixed pattern length. Thus, they cannot extract all useful patterns that extrapolate the fixed length. In addition, The workload behavior changes might start after training the prediction model. The model should be retrained to adapt to workload changes. Gathering new data and retraining the models may very time-consuming (AMIRI; MOHAMMAD-KHANLI, 2017).

The control theory goal is to maintain the value of one or more controlled variables (control outputs), close to a desirable set point, via manipulated variables (control inputs) adjustment. The control systems are divided into three groups: (i) Open-loop, the system output

is calculated based only on the current input and the system model, *i.e.*, this control is unaware when the output reaches the desired set point; (ii) Feedback, the output of the system is observed. Thus, the controller can correct output deviations from the set point and; (iii) feed-forward, predicts the output errors and reacts before the error actually occurs. The prediction may fail. Thus, the combination of feedback and feed-forward controllers is commonly used. The controller performance depends on the application model and the controller itself (reactive or proactive). Many researchers consider that control theory has a great potential when combined with resource prediction (LORIDO-BOTRAN *et al.*, 2014).

Patikirikorala and Colman (PATIKIRIKORALA; COLMAN, 2010) highlight the advantages of applying feedback controllers in cloud platforms. Feedback control is able to handle unpredictable workload conditions, nonlinear characteristics and provides effective control under disturbance and un-modeled dynamics. Furthermore, there is systematic design process to develop an effective controller with well-proven tools and methodologies to calculate the controller tuning parameters and validate the controller. In addition, there are many standard controller algorithms available from control literature with formal stability proofs. Also, the authors discuss the well-established feedback control schemes in three categories, as shown in Figure 2: fixed gain controllers, adaptative controller and Model Predictive Control (MPC).

The Fixed gain controllers are widely adopted due to their simplicity. However, once the model parameters are set, they remain fixed during the controller execution. An example of this class is the Proportional Integral Derivate (PID).

In contrast, the adaptive control addresses some of the limitations of fixed gain controllers, by adjusting the controller tuning parameters online without human intervention. The adaptative control constructs the dynamic model of the system in each sampling instance. This type of controller is suitable for workloads with smooth variations. However, in the presence of sudden burst, the online estimation process may not capture the dynamics of the system. In addition, the parameters are updated online, but the control law remains the same (PATIKIRIKORALA; COLMAN, 2010).

Reactive systems might not be able to scale in case of sudden workload bursts because it takes several minutes to deploy or migrate a VM image, boot the VM operating system and application, and have the server fully operational (MAO; HUMPHREY, 2012). In view of this, Lorido-Botran et al. LORIDO-BOTRAN *et al.* (2014) concluded that efforts should be redirected towards developing proactive auto-scaling systems able to predict future needs

and acquire the corresponding resources with enough anticipation, maybe using some reactive methods to correct possible prediction errors.

In this context, we adopt in this work a proactive and self-optimizing technique called MPC. The MPC uses the dynamic model and predicts future behavior of the system to come up with the decisions to optimize the future behavior. In addition, MPC is attractive for multi-objective scenarios and to derive close to optimal decisions in the presences of complex policies and constraints.

The MPC establishes a relationship between the application performance and the workload and guarantees that the performance does not exceed a predefined threshold, avoiding SLA violations. The MPC treatment of constraints is simple, and these can be systematically included during the design process (CAMACHO; ALBA, 2013). The main MPC drawback is the need for an appropriate model of the system. The MPC algorithm is based on prior knowledge of the model and is independent of it, but the benefits obtained will be affected by the discrepancies existing between the real process and the model used (CAMACHO; ALBA, 2013). Nonlinear controllers can model the application behavior with accuracy, however, they require complex mathematical calculations. In this context, some controllers assume that the application behavior is linear. This makes the model less computationally costly, but there is instability associated with the controller (AMIRI; MOHAMMAD-KHANLI, 2017).

The MPC could be used to several inputs and outputs variables, representing the behavior of Single Input Single Output (SISO) and Multiple Input Multiple Output (MIMO) systems. In this thesis, we model a Multiple Input Single Output (MISO) scenarios.

### *2.3.1 Systematic Mapping*

This search strategy was adapted from systematic review proposed by MADNI *et al.* (2016) and it involves the formulation of research questions, a search of different databases, keywords definition, and inclusion/exclusion criteria presented in Table 3. The main goal of this search strategy is to address the following questions:

- What are the motivations to workload prediction in Cloud?
- What are the main challenges in cloud workload prediction?
- What are the main motivations for using the MPC approach?

The search decision for the required research works from 2016 to 2017 and the databases chosen are *Google Scholar* and *Web of Science*. On the basis of the topic and the

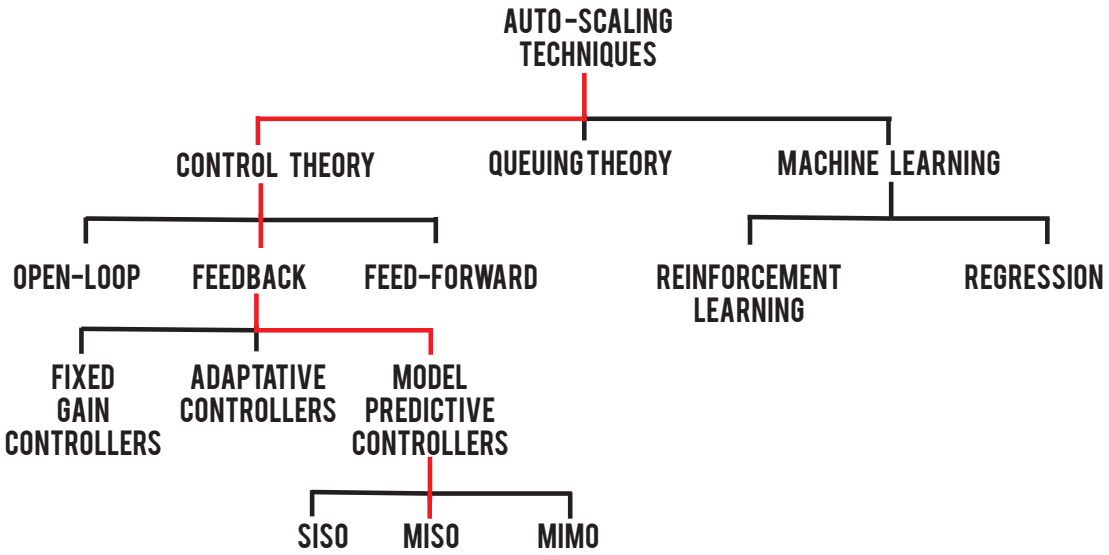



Figure 2 – Prediction methods taxonomy (adapted from (LORIDO-BOTRAN *et al.*, 2014)).

proposed research questions, we define the searching keywords as a first step to formulating the search string. The following keywords *Cloud Computing*, *Resource Management*, *Model Predictive Control* are considered. We use the logical operator AND for connecting the keywords, resulting in the following search string: "Cloud Computing" AND "Resource Management" AND "Model Prediction Control".

Most of the studies were screened out because their titles or abstracts were not relevant to be incorporated in this review or they did not meet the selection criteria. The inclusion and exclusion criteria are defined in Table 3. The initial search resulted in a total of 49 studies, which were condensed to 12 studies on the basis of their titles and abstracts. Even if the paper did not cite the MPC in the title or abstract, it could have a satisfactory content in the subject forward its body. Therefore, in this first phase, a search for the MPC keyword was used throughout the studies.

### *2.3.2 Importance of Cloud Workload Prediction*

Infrastructure as a Service (IaaS) resource management is critical to cloud computing since it allows for optimizing the resource utilization, which assists the management cost reduction, energy savings, *etc*. Also, the resource management ensures QoS by reduction of overhead and latency and improvement of throughput (WEINGARTNER *et al.*, 2015). In this work, we focus on the IaaS provider's perspective.

IaaS resource management task involves resource allocation, provisioning, require-

Table 3 – Studies inclusion/exclusion criteria.

| Inclusion criteria | Exclusion criteria |
|---|---|
| The study focuses on resource management in Cloud Computing; | The study does not focus on resource management in another area; |
| The study considers the Infrastructure as a Service (IaaS) for resource management only; | The study does not consider the Software as a Service (SaaS) or Platform as a Service (PaaS); |
| The study is written in English only; | The study is not written in the English language; |
| The study is peer-reviewed and published in scholarly society; | The study is not peer-reviewed such as workshop, descriptions and technical reports; |
| The study is published in journals or conferences; | The study is not published in the form of books, abstracts, editorials or keynotes; |

ment mapping, adaptation, discovery, brokering, estimation, and modeling, each one with their own challenges (MANVI; SHYAM, 2014). This thesis focuses on resource modeling and estimation. Consequently, the use of computing resource (CPU, disk, memory) by an application and how this usage varies according to time are described and used to build performance models. These models are used to estimate the resource demand of each application deployed in the cloud environment. Thereby, resources can be allocated in an efficient manner.

Furthermore, the system's ability to predict the resource demand in subsequent time intervals directly helps the system to adjust the resources dynamically to fulfill the QoS requirements. Thus, this work supports the monitoring (M) and proactive analysis phase analysis(A) phases of the MAPE loop of auto-scaling process (LORIDO-BOTRAN *et al.*, 2014). Manvi and Shyam (MANVI; SHYAM, 2014) raise open challenges in resource mapping, provisioning, allocation, and adaptation.

### *2.3.3 Challenges of Cloud Workload Prediction*

Workload prediction is especially challenging when applied in a highly dynamic and elastic environment with resource contention scenario, multidimensional resource demands (CPU, network, memory, disk) for multitier applications, that must meet service-level agreements (SLAs) in the face of non-stationary workloads (ANGLANO *et al.*, 2016). The cloud hosts a wide variety of applications deployed and destroyed at any time, with different characteristics and user profiles, which have heterogeneous and competing QoS requirements (CALHEIROS *et al.*, 2011). This leads to complex workloads depending on users' behavior and resource consumption.

The use of past demands to predict future demands cannot be accurate regarding Web-based interactive applications (PADALA *et al.*, 2009). For this reason, the estimation models need process each one of the computational resources in a real time manner to support the dynamic resource management. Additionally, the cloud system should automatically adjust the resources to the workload handled by the application with minimum human intervention (LORIDO-BOTRAN *et al.*, 2014).

To obtain accurate prediction models, all behavioral patterns of workload should be captured by prediction models. For machine learning methods such as Neural Networks, Support Vector Machine, Markov Model, based on fixed pattern length, it is an important use sliding window to capture patterns in differents regions of workload. Otherwise, the method needs to be retrained. The technique should guarantee the model robustness, *i.e.*, the model predictive ability to generalize on inexperienced workloads. Moreover, most of these methods present discriminative learning, *i.e.*, they do not learn workload pattern explicitly. They learn the model parameters to optimize a utility function. But, parameters are of little interest to the resource manager. Therefore, some methods are needed to extract knowledge from a model and improve its transparency. Another challenge is regarding the absence of a standard testbed to evaluate the prediction techniques. This can lead to models that super fit specific workloads (AMIRI; MOHAMMAD-KHANLI, 2017).

### *2.3.4 Model Predictive Control (MPC)*

Proactive auto-scaling systems able to predict future needs with enough anticipation are critical in clouds (LORIDO-BOTRAN *et al.*, 2014). Since anticipation is critical, the model predictive control automates the scaling task, complementing the capacity planning and workload migration approaches. Model Predictive Control does not designate a specific control strategy but rather a range of control methods which make explicit use of a process to obtain the control signal by minimizing an objective function (CAMACHO; ALBA, 2013).

The goal of the MPC is to detect the occurrence of process shifts quickly and to take the corrective action before SLA violation happens, which means maintain the *Predicted Output* close to the desired control signal (*i.e. Reference Trajectory*). To this end, the *Predicted Output* is compared with the *Reference Trajectory* and the difference between them is applied as feedback to the *Optimizer*. This difference is called *Future Errors*, as presented in Figure 3. The *Optimizer* solves an optimization problem taking into account a predefined cost function

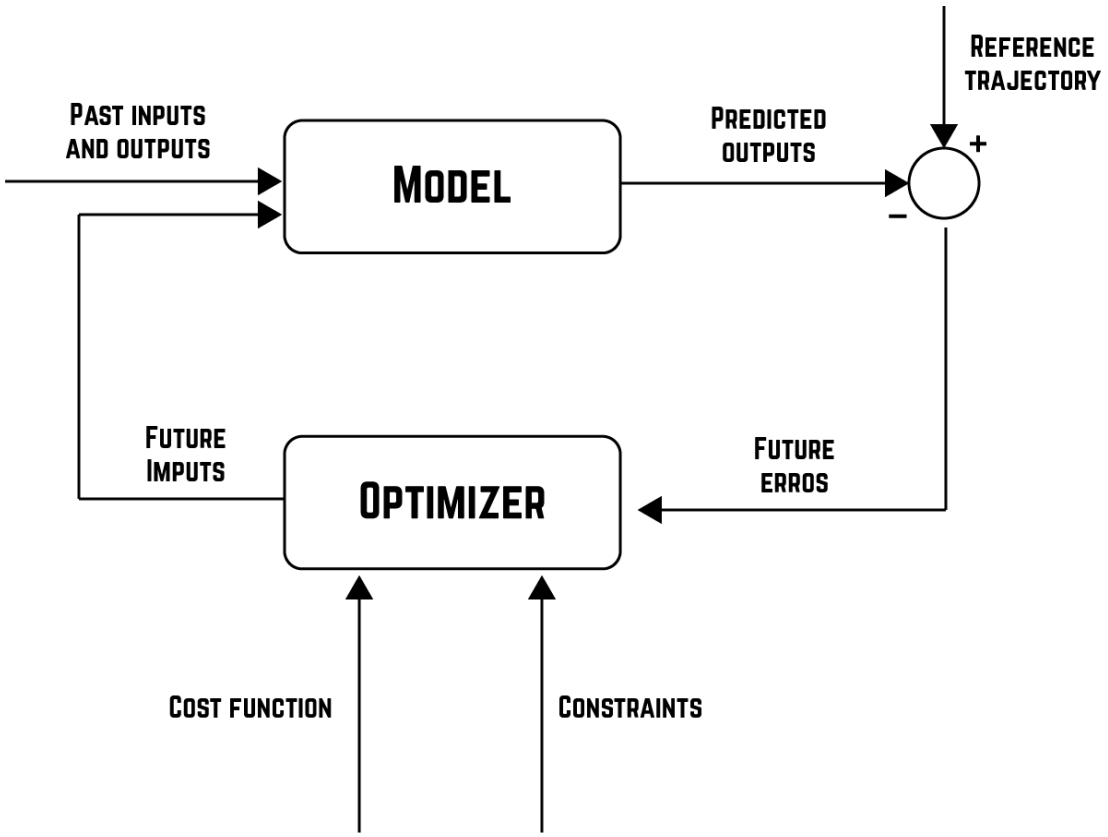



Figure 3 – Model based Prediction Control Process (adapted from (CAMACHO; ALBA, 2013)).

to minimize the *Future Errors*. To define the best current control action which predicted input trajectory gives the lowest numerical value to the cost. The MPC reduces the errors over time by adjustment of the control variables (CPU, Memory, and Disk utilization).

To model the complex relationship among the dependent and independent variables, the MPC predicts the future outputs, based on past inputs and outputs, and on the proposed optimal future control actions (Section 2.3.4.1). These actions are calculated by the optimizer (Section 2.3.4.2) taking into account the cost function, the future error, as well as the constraints.

#### *2.3.4.1 Prediction*

The MPC use is determined by the necessity to calculate the predicted output at future instants $\hat{y}(t+k|t)$ for $k = 1...N$, where $N$ is the prediction horizon at each instant $t$. The model might use different techniques to represent the relationship between the outputs and the measurable inputs; the most commonly used are impulse response, step response, transfer function, and state space model (CAMACHO; ALBA, 2013). In this work, the last two techniques are adopted.

In the MPC, a formal relationship between the input and output is modeled to determine how a change in the former affects the output. This formal relationship is denoted as transfer function or state space function or performance model (LORIDO-BOTRAN *et al.*, 2014).

**Transfer Function**: the Transfer Function Model is widely used in the industry (CAMACHO; ALBA, 2013). Although the derivation of the controller is more difficult, it requires fewer parameters. The concept of transfer function $G = B/A$ so that the output is given by:

$$A(z^{-1})y(t) = B(z^{-1})u(t), \tag{2.1}$$

where $A(z^{-1}) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n a z^{-na}$, $B(z^{-1}) = 1 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n b z^{-nb}$, and $z^{-1}$ is the backward shift operator. Thus, the prediction is given by:

$$\hat{y}(t+k|t) = \frac{B(z^{-1})}{A(z^{-1})} u(t+k|t). \tag{2.2}$$

This representation is valid for unstable processes as well, and it has the advantage that a set of few parameters are required, although *a priori* knowledge of the process is critical, especially that of the order of the A and B polynomials.

**State Space Model**: the system is described by a linear discrete time model, called State Space, and represents the dynamics of a $n^{th}$ order system as a first differential equation represented by a $N$ size vector. It has the following representation:

$$x_{t+1} = Ax_t + Bu_t, \tag{2.3}$$

$$y_t = Cx_t + D_t, \tag{2.4}$$

where, $t$ denotes time, $x_t$ is the state (vector), $u_t$ is the input or control, $y_t$ is the output and $d_t$ is the disturbance. All this terms are time dependent. A, B, C and D are constants of model represented by matrices. In this work, the disturbance is not considered, i.e, D=0. A system that can be represented by an $n^{th}$ order differential equation with $r$ inputs and $m$ outputs, $A_{[nxn]}$ is

the state matrix, $B_{[nxr]}$ is the input matrix and $C_{[mxn]}$ is the output matrix. The prediction for this model is given by:

$$\hat{y}(t+k|t) = C\hat{x}(t+k|t) = C[A^k x(t) + \sum_{i=1}^{k}(A^{i-1})Bu(t+k-i|t)]. \tag{2.5}$$

The state space model has the advantage of multivariable processes can be used in a straightforward manner. The control law is simply the feedback of a linear combination of the state vector (CAMACHO; ALBA, 2013).

#### *2.3.4.2 Optimizer*

The future output $y$ is calculated by optimizing a determined criterion to keep the process as close as possible to the reference trajectory $w(t+k)$. This criterion usually takes the form of a quadratic function of the errors between the predicted output signal and the reference trajectory. The control effort $\Delta u$ is included in the objective function in most cases. The general expression for such an objective function is:

$$J(N_1,N_2,N_u) = \sum_{j=N_1}^{N_2}\delta(j)[\hat{y}(t+j|t) - w(t+j)]^2 + \sum_{j=1}^{N_u}\lambda(j)[\Delta u(t+j-1)]^2, \tag{2.6}$$

where $N_1$ and $N_2$ are the minimum and maximum prediction horizons and $N_u$ is the control horizon. The coefficients $\delta(j)$ and $\lambda(j)$ are the future behavior, usually constant values or exponential sequences are considered. An advantage of MPC is that the reference is known *a priori*. Thus, the system can react before the change has effectively been made, avoiding delay in the process response.

Regarding constraints, in practice, all processes are subject to them. Normally, bounds in the amplitude and in the slew rate of the control signal and limits in the output will be considered.

$$u_{min} \leq u(t) \leq u_{max}, \quad \forall t, \tag{2.7}$$

$$y_{min} \leq y(t) \leq y_{max}, \quad \forall t. \tag{2.8}$$

By adding these constraints to the objective function, the minimization becomes more complex, so that the solution cannot be obtained explicitly as in the unconstrained case.

#### *2.3.4.3 Control Law*

In order to obtain values $u(t+k|t)$ it is necessary to minimize the objective function $J$ of Equation 2.6. To this end, the predicted outputs $\hat{y}(t+k|t)$ are computed as a function of past values of inputs and outputs and future control signals. An analytical solution can be obtained for the quadratic criterion if the model is linear and there are not constraints. Otherwise, an iterative method of optimization should be used. This is not a trivial task since there will be $N_2 - N_1 + 1$ independent variables. To reduce this degree of freedom a certain structure may be imposed by the control law. This structuralizing of the control law produces an improvement in robustness and in the general behavior of the system.

The control law structure is sometimes imposed by the use of the control horizon concept $(N_u)$ that consists of considering that after a certain interval $N_u < N_2$ there is no variation in the proposed control signals:

$$\Delta u(t+j-1) = 0, \quad j > N_u. \tag{2.9}$$

Quadratic programming methods have to be used in the presence of constraints. However, an explicit solution does exist for certain types of constraints, for example, when the condition that the output attains the reference value at a determined instant is imposed, this method is used in Constrained Receding Horizon Predictive (CAMACHO; ALBA, 2013).

### *2.3.5 Related Work in Model Predictive Control*

PAUL *et al.* (2016) present a framework for modeling and distributing the jobs among geographically distributed data centers. This framework uses MPC based algorithms in order to reduce the cost of electricity and to increase renewable energy penetration. Thus, the MPC optimization function has three major goals. Firstly, it minimizes the electricity cost incurred by the cloud provider. Secondly, it maximizes the renewable energy usage at all locations. Thirdly, it minimizes the number of switchings of the servers. The MPC based framework was able to appropriately address the issue of server switching and the adverse effects associated with it, reducing the variance of an average cost of electricity in 15.12% to 20.74%. This result mitigates the electricity price risk and represents a considerable amount of financial cost savings.

To address workload fluctuations on Apache Storm, FARAHABADY *et al.* (2016b) propose an MPC-based schedule that improves performance with highly dynamic behavior.

The controller allocates computing resources based on decisions that consider the future states of non-controllable disturbance parameters, *e.g.*, arriving rate of tuples or resource utilization in each worker node while ensuring the required QoS for the individual data streams. The MPC-based solution was able to response to changes in a variation of workload or resource capacity, while it maintains the CPU cores utilization of active working nodes around 80% and achieves an average of 31% improvement in resource utilization. However, the solution needs to find an automatic way to manage situations where unusual burst streaming of data happens abruptly.

The applications executing in virtual platforms compete for resources and interfere with each others' performance, resulting in their performance variability/degradation. FARAHABADY *et al.* 2016a present a dynamic resource allocation solution based on a model predictive controller that aims at increasing the overall resource utilization of servers while achieving QoS level demands by users. The solution improves the overall resource utilization (MIMO) by 32%, without a significant impact on the QoS. Such improvement results in a fewer number of active servers and in turn contribute to an overall energy saving by 33%.

ANGLANO *et al.* 2016 propose a dynamic vertical scaling resource management framework, based on MPC fuzzy control. The control is able to dynamically adjust the CPU and memory capacity allocated to each VM, in the face of nonstationary workloads and resource contention, in order to precisely match the workload needs. The main advantage of authors work is the controller takes decisions by considering, at each control interval, the interference of dynamic memory adjustment on vCPU utilization, and the sign of the performance error. In addition, this related work details the control parametrization. As in our work, the authors makes use of RUBiS as multi-tier application; further, the authors also model MISO controllers. To avoid the negative effects originated by the lack of coordination between CPU and memory allocation the controllers interact in a cascade control strategy, whereby the MISO CPU Controller controls the MISO Mem Controller, which takes that vCPU capacity as input. As a disadvantage, the rule base design is challenging. Simple rules do not require complex computing, but they could not manage the resources in an efficient manner. In contrast, achieve complex rules might be difficult and time-consuming. In addition, they are more sensitive to the workload changes.

SONG *et al.* 2016 propose a hybrid solution composed of a rule-based control and a model predictive control to find the optimal load distribution, which can lower the outlet air temperature of servers. The inlet and outlet air temperatures of servers are important because

they need cooling to operate reliably. The inlet temperature of the server is mainly affected by the data center cooling system (stable). In contrast, the outlet air temperatures are largely affected by the load assigned to the servers. An unreasonable allocation of the load will result in high-temperature areas in the outlet aisle of the rack. The objective of our control strategies is to minimize the maximum outlet temperature of servers. The rule-based control strategy was making the fullest use of servers with low outlet air temperature. The MPC approach solves an optimal load distribution while minimizing the maximum outlet air temperatures of servers and maintaining the quality of service. The simulation results show a temperature drop between 0.5 and 1 centigrade degrees comparing to others literature approaches.

Dynamic environments like the cloud, in general, present non-linear and workload-dependent behavior. However, PADALA *et al.* (2009) demonstrated that such behavior could be approximated locally by a linear model (model simplicity), the online controller is periodically re-estimated with real-time measures of the variables of interest, allowing the model to adapt to a different operation and workload conditions. ZHU; AGRAWAL (2010) used MPC in their proposal and show that it was able to adjust parameters so that the application maximizes resource utilization with little overhead. In addition, the authors showed that the control model could capture the application behavior in an accurate manner.

In practice, MPC has proved to be a reasonable strategy for resource management in cloud environments. In general, we observe in the related works a lack of detail regarding the controller parameterization, which compromises the reproducibility of the proposals. In addition, in the MPC theory, the formal relationship between the input and output (transfer function) is modeled to determine how the input affects the output. This formalization facilitates the researchers' practical use. The transfer function can be used directly in the controller for testing and validation purposes.

## 2.4 Concluding Remarks

The workload modeling and characterization support simulation which brings benefits to researchers and providers. The models allow for adjustments to fit particular situations, controlled modification of parameters, repetitions of evaluation conditions, and generalization of patterns found in the application, providing a controlled input for researchers. For cloud providers, they allow the improvement of their systems QoS without deployment and execution in large scale environments and production of trace logs based on realistic scenarios.

However, workload modeling and characterization are especially challenging when dealing with a highly dynamic and heterogeneous environment, such as cloud data centers, contributing to the lack of methodologies to characterize the different behavioral patterns of cloud applications.

The prediction of resource demand directly helps the system to adjust the resources dynamically to fulfill the QoS requirements. Therefore, reactive systems might not be able to scale in case of sudden workload bursts because it takes several minutes to deploy or migrate a VM image. In view of this, efforts should be redirected towards developing proactive auto-scaling systems able to predict future needs and acquire the corresponding resources with enough anticipation.

The knowledge acquired in this chapter allowed to understand the choices made in the next chapter. As the planning of this work, the following chapter presents the workload profiling methodology and the prediction approach used.

# 3 CLOUD WORKLOAD MODELING AND PREDICTION

In this chapter, we present the proposed workload modeling methodology, discussing all the steps to achieve the statistical models that represent the workload. After, we introduce the model simulation through graphical and statistical hypothesis test approaches. Finally, we show the model predictive control responsible for taking corrective actions based on predictions to maintain the QoS acceptable with an efficient resource utilization. This controller counts on the state space model predictor to estimate the application performance given support resource allocation policies.

## 3.1 Workload Modeling Methodology

Figure 4 presents the methodology used to create the performance models of applications (MAGALHAES *et al.*, 2015). The modeling begins with the measured data about the workload. The workload is composed of requests submitted by *Users*. These requests generate a *Demand* to the application server present in *Cloud*. The requests are processed by server meanwhile the metrics of interest are *Monitoring and Tracing*, and stored in a *Trace Data*.

After the experiments, the *Statistical Analysis* enables the understanding and quantification of the characteristics of data. Then, a *Performance Modeling* is conducted to reach the statistical models that represent the application performance for a specific scenario.

The *Performance Modeling* consists of finding the mathematical functions (statistical distributions) that are capable of representing the observed metrics. The modeling process is composed of three steps: (i) exploratory analysis (Section 3.1.3), (ii) parameter estimation (Section 3.1.4) and (iii) GoF tests (Section 3.1.5), as shown in Figure 5. The first one analyzes the data characteristics and determines the candidate distributions to represent the model. The second step uses estimation methods to set the parameters of the model based on the collected samples, given the selected distribution chosen in advance. In the last step, the tests are composed of graphical and analytical methods to evaluate whether the distributions and their respective parameters provide approximation satisfactorily to the empirical data.

Therefore, the *Statistical Models* are created from the trace and used for either *Prediction* or *Simulation*. In the *Simulation* case, the *Statistical Models* are used as input to generate the users behavior that will be integrated with the set of classes that represent the application's entities modeled in the cloud infrastructure simulator (CALHEIROS *et al.*, 2011)

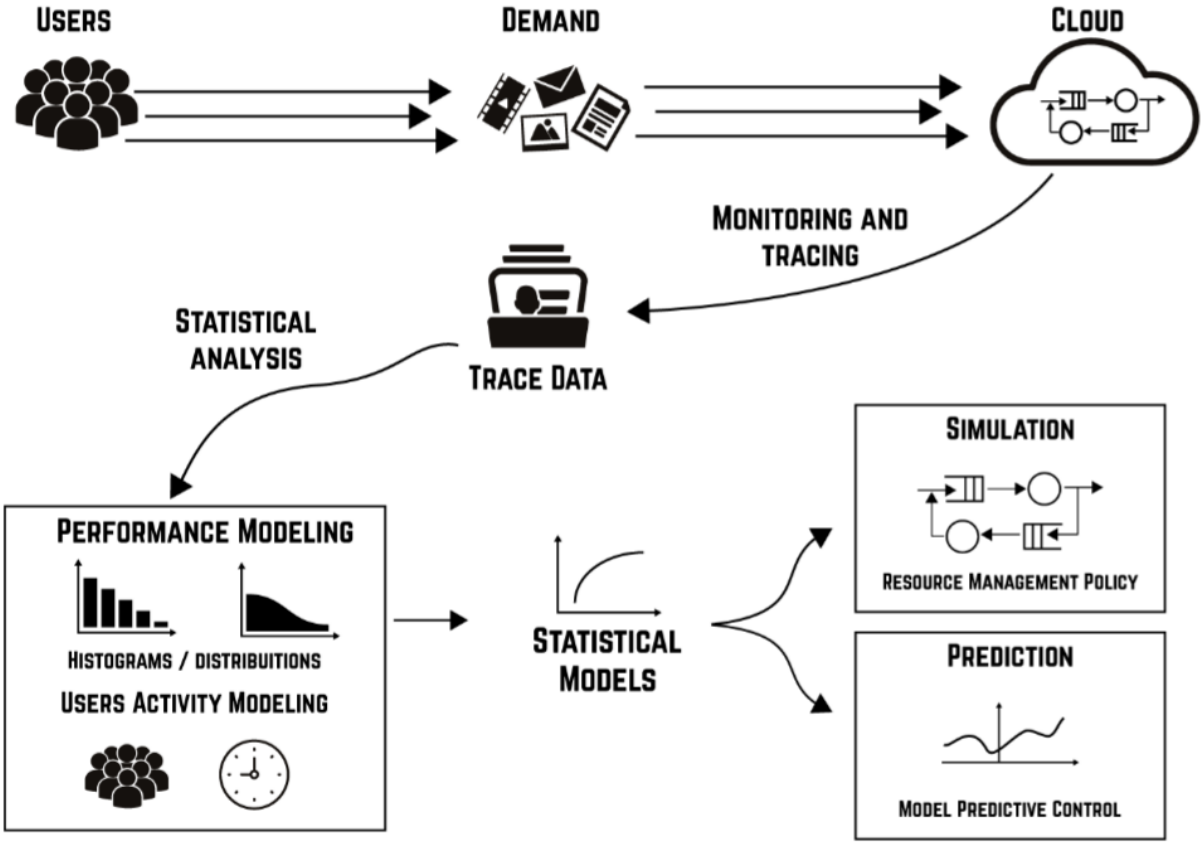



Figure 4 – Workload modeling scenario (adapted from (FEITELSON, 2014)).


Figure 5 – Workload modeling process.

and, finally, the proposed model can be validated.

In the *Prediction* case, *Statistical Models* can be used to build a performance predictor to assist policies for management and provisioning of resources and energy saving for this class of application. Moreover, the performance of the application can be examined in different scenarios through the insertion and adjustment of parameters in the model.

The methodology proposed in this work define how to achieve performance models with a set of rules, methods, and a Process. To this end, we instantiate the process with a well defined steps described in the Figure 5. All steps will be described in the following subsections.

### *3.1.1 Experimental Design and Setup*

This step includes the definition of application and metrics to be observed. Also, it comprises the testbed configuration and experimental design definition.

### *3.1.2 Monitoring and Trace*

This step includes the definition of monitoring tools and configuration. Also, it comprises the implementation of gathering scripts, and definition of data organization scheme.

### *3.1.3 Exploratory Analysis*

Understanding the characteristics of the available sample is important to decide in advance which distributions better approximate the collected data. In this direction, the first step is to identify the variables types, for example, discrete and continuous or qualitative and quantitative.

Since the data are quantitative, descriptive statistics such as mean, standard deviation and kurtosis provide a way to characterize the data as to their shape, location, and dispersion. They also offer an interesting way to summarize them. Such description is necessary to assist the choice of possible distributions that can provide a good fit to data sample.

Many statistical techniques used to, for example, estimate parameters, validate the adherence of the data to the model, assume that the data are normal. Therefore, it is interesting to perform a pre-processing data. Then, the data normality has to be verified via normality tests. Holding the results, changes could be made to modify the data scale to make the normality assumption plausible.

### *3.1.4 Parameters Estimation*

The identification of patterns in the observed data supports the modeling of application. In this work, the patterns are modeled in the form of statistical distributions. This section describes the set of steps used to find a mathematical function capable of representing the observed metrics, exploratory analysis, parameter estimation, and goodness of fit tests.

Thus, the target metric has $n$ observations, $x_1, x_2, ..., x_n$, from an unknown population. The goal is to find the probability density function (pdf) $f(x, \theta)$, where $\theta$ is the vector of parameters estimated from the available sample, that represents the data adequately.

After choosing the distributions that can represent mathematically the data, it is necessary to estimate the parameters of the models. One option is to calculate the moments of the sample and use them as estimators for the moments of the distribution. However, this approach is highly sensitive to outliers, especially for the third and fourth moments, what limits its utilization in the case of distributions used in this work (FEITELSON, 2014; CHALABI *et al.*, 2012b).

Different estimation methods are presented (CHALABI *et al.*, 2012b) :

- MLE: considering $n$ independent observations $x_1, x_2, ..., x_n$ from a sample belonging to a distribution defined by a set of parameters $\theta$ of the probability density function $f$. The mle consists in the determination of values of $\theta$ that maximize the log-likelihood function defined by

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} \ln(f(x_i;\theta)). \tag{3.1}$$

- Histogram Fitting (HF) : the observed data is represented in the form histogram and the probabilities at the midpoint of their bars are fitted to the GL density function. The method Freedman-Diaconis (FREEDMAN; DIACONIS, 1981) is used in order to define how many bars of histogram should be considered in order to obtain the best fit. This is a robust method based on the inter-quartile range;
- Quantile Matching (QM) : this method consists of finding the parameter values that minimize the difference between the theoretical quantiles and sample quantiles. Given a set of probabilities, $p$, as a sequence of values in the range $0 < p < 1$. The quantile matching method is defined by the cumulative probability distribution $F$ with a set of parameters $\theta$ determined by

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i}^{n} (F^{-1}(p_i|\theta) - Q(p_i))^2, \tag{3.2}$$

  where $n$ is the cardinality of $p$, $F^{-1}(p|\theta)$ is the quantile function of GL distribution, and $Q$ is the quantile function of the sample number of observed instructions.
- Probability Weighted Moments (PWM) (GREENWOOD *et al.*, 1979): this method is a generalization of the method of classic moments with better performance, where the

weighted probability moments of a random variable $X$, with cumulative distribution function $F$ and set of parameters $\theta$, is defined by

$$M(p,r,s) = E[X^p(F(X))^r(1-F(X))^s], \tag{3.3}$$

where $p$, $r$ and $s$ are real positive numbers. When $r$ and $s$ are integers, $F^r(1-F)^s$ can be expressed as a linear combination of powers of $F$ or powers of $(1-F)$, ergo, the distribution is summed through the moment $M(1,r,0)(r=0,1,...)$ or the moment $M(1,0,s)(s=0,1,...)$. Assuming the notation $M(1,0,s)(s=0,1,...)$ used by Hosking et al. (HOSKING *et al.*, 1985), we have

$$\mu_s = E[X(1-F_\theta(X))^s], s=0,1,2,... \tag{3.4}$$

The estimator of the first moment is found by simply assigning the value 0 to $s$ and replacing $\mu_0$ by the value obtained by calculating the first moment of the sample of data. The same process is also used for estimation of the other moments.

- Maximum Product of Spacing (MPS) Estimator (CHENG; AMIN, 1983): the parameter estimates are the values that make the spacing between the cumulative distribution functions of the random variates equal. This method is more successful than mle because it is equivalent to maximizing the geometric means of the probability spacing.

### *3.1.5 Goodness of Fit*

Once the selected parameters and their distributions are known, the next step is deciding which model fits to the data through the GoF tests. The GoF statistics check whether the empirical and theoretical data belong to the same distribution. In this study, two methods are used to assess if the selected distributions provide a good fit to the data: one graphic method using the Quantile-Quantile (Q-Q) plots, and one analytical method using the KS test.

#### *3.1.5.1 Quantile-Quantile Plots*

The Q-Q plots technique consists of calculating empirical and theoretical distribution quantiles and plotting one in terms of the other. If the distributions are equal, the points on the graph are equidistant from the ordinates and abscissas axis, leading to a 45 degrees inclination straight line (FEITELSON, 2014).

In addition to checking whether two data sets belong to the same distribution, Q-Q Plots method allows one to check other aspects simultaneously, for example, the presence of outliers.

#### *3.1.5.2 Kolmogorov and Smirnov Test*

The KS test evaluates the hypothesis that the observed data belongs to a population that follows one or more probability distributions. The test can be defined as the hypotheses: (i) **Null Hypothesis ($H_0$):** the observed data follows the specified distribution and (ii) **Alternative Hypothesis ($H_a$):** the observed data does not follow the specified distribution. This test calculates the maximum deviation ($D$) between the cumulative distribution functions of empirical and theoretical data, for all samples (FEITELSON, 2014):

$$D_n = \max_{i=1...n} \{|\frac{1}{n} - F(x_{(i)})|, |F(x_{(i)}) - \frac{i-1}{n}|\}, \tag{3.5}$$

where $F(x)$ is the theoretical distribution, and $n$ is the sample size of the data observed. In general, the value of $D$ is compared with a threshold called critical value to determine if the null hypothesis is accepted or rejected. If the maximum deviation is less than the critical value, the null hypothesis is accepted. Otherwise, it is rejected. The critical value is extracted from predefined tables (STEPHENS, 1974), according to the sample size ($n$) and the significance level ($\alpha$) determined. In this work, we fix the sample size $n = 100$ and the significance level at $\alpha = 0.05$.

## 3.2 Simulation

The statistical models should be implemented as an extension of a Cloud simulator. Validation will be realized to show that the simulator is able to represent the observed data accurately. Model Predictive Control does not designate a specific control strategy but rather an wide range of control methods which make explicit use of a model of the process to obtain the control signal by minimizing an objective function.

The graphical and statistical hypothesis test approaches were used to evaluate the accuracy of the model in the simulator by comparing the simulated against the observed data. The Wilcox Mann-Whitney (WMW) hypothesis test (GOLD, 2007) consists of evaluating the hypothesis that the simulated data belong to the population that follows the probability


Figure 6 – Resource management sequence.

distribution of the observed data. If $p - value > \alpha$, the hypothesis cannot be rejected. Otherwise, the null hypothesis is rejected. The level of significance was set at $\alpha = 0.05$ for all the tests.

## 3.3 Prediction

An efficient resource management is essential to avoid under and over provisioning of resources. The resource management comes from the workload monitoring to the resource provisioning policy. In this context, workload profiling is useful since it can produce workloads with different intensities of the fluctuations and the burstiness. This improves the prediction methods adaptability and accuracy because such methods can be evaluated under different workloads (AMIRI; MOHAMMAD-KHANLI, 2017). Therefore, the statistical models generated as a result of the methodology proposed in this thesis are inputs to the model prediction control which adjusts the resource utilization values to maintain the application response time within an acceptable QoS. Such estimated values can be directly used by the resource provisioning policy to guide its decision making, as shown in Figure 6, whose the highlighted boxes represent the steps covered by this thesis. According to Section 2.3.5, there is a lack of detail regarding the controller parameterization. The greatest MPC drawback is the need for an appropriate model of the process to be available. The control performance is directly affected by the discrepancies existing between the real process and the model used.

### *3.3.1 Cloud Scenario*

In this thesis, it was considered a representative application that runs on today's virtualized infrastructures, a 2-tier Web application that implements an auction site prototype modeled as eBay.com. The CPU, memory and disk utilization is monitored on the application server at VM in order to verify the behavior of the use of physical resources while processing requests from a user. Our system consists of a MISO model shown in Figure 7, where we have


Figure 7 – MISO system.

three inputs: $u_1$, corresponding to the CPU; $u_2$, corresponding to memory; And $u_3$ corresponding to the disk. The system has the response time as output $y$ as well.

Dynamic resource allocation in the cloud depends on virtualization technologies. The VM provisioning technology is able to provide a VM in minutes. The streaming VM technology allows for a VM preview before it is entirely prepared but still take time until a proportion of it is ready. The rapid VM cloning technique allows a VM to be created in seconds. However, the VM preparing involves software installation, automatic patching, security checking, thus, the acceleration of VM creation cannot reduce the overall waiting time significantly. Such technologies take around 10 minutes to create a VM and make it fully operational. Our prediction window was set in 1200 milliseconds. Thus, the actions taken by the resource provisioning strategy can be fully completed and considered before further provisioning decisions are made.

In our model, the reference trajectory is set to response time value that does not compromise the QoS. In literature, IQBAL *et al.* 2010 adopt an SLA that enforces one-second maximum average response time requirements for static and dynamic resources of the RUBiS application. WANG; WANG (2011) select 700 ms to the set point of the response time controller. BODIK *et al.* (2007) define the 99th percentile of response times is less than 300 ms. So, our set point is fixed in values between 300ms to 1000ms.

ZHU *et al.* (2009) have performed experiments with the same 2-tier Web application adopted in this work. They found a behavior which is consistent with the application being CPU-bound. Our experiments corroborate these results. Therefore, we attribute the biggest weight to the CPU comparing with memory and disk utilization.

### *3.3.2 Model Assumptions*

**Linear Model**: in general, clouds present non-linear behavior, however, PADALA *et al.* (2009) demonstrated that such behavior could be approximated locally by a linear model, which facilitates optimization as well as off-line analysis of expected closed-loop controller behavior.

**Disturbance:** a disturbance model describes the behavior not reflected by the process model, including the effect of nonmeasurable inputs, and model errors. In this work, the disturbance was disregarded, so the value zero is assigned to the constant in the model described in Equation 2.4 ($D = 0$);

**Noise**: we assume only the presence of white noise in the input and output channels. It is generated such that the standard deviation of each measurement is approximately 1% of its original value under normal operating conditions.

## 3.4 Concluding Remarks

A well-defined methodology is described in this chapter to obtain statistical models in a realistic manner in three steps: an exploratory analysis, parameter estimation and good of fitness test. The exploratory analysis is critical to define a good distribution group candidate to represent the data. Also, different parameter estimation methods are presented which is crucial to achieving realistic models, since different methods can offer more robust parameters to a specific distribution, offering a more accurate model. Finally, it is interesting to use graphic and analytic methods to support the decision if two data sets belong to the same distribution.

Since performance models allow predicting of how the application patterns will change, we present the MPC as a solution that estimates resources utilization while maintaining this estimation inside an output trajectory (response time, execution time) acceptable in QoS perspective.

Reactive systems might not be able to scale in case of sudden workload bursts frequently present in the cloud. Such environments, in general, present non-linear and workload-dependent behavior. Therefore, proactive auto-scaling systems capable of make decisions with enough anticipation are needed. The MPC offers a local approximation of system behavior via a linear model. Furthermore, the controller is periodically updated with real-time measures, allowing the model to adapt to workload conditions. Thus, the MPC is capable of represents a complex system in a simple manner.

# 4 MATERIAL AND METHODS

To achieve the workload modeling methodology, a series of decisions must be made in advance, such as: defining the workload (Web and scientific applications) used and which characteristics should be modeled (Sections 4.1 and 4.2), setting the environment of experimentation (Sections 4.1.4 and 4.1.4), determining which metrics are evaluated and how data should be collected (Sections 4.1.2, 4.1.3 and 4.2.2). In addition, this chapter describe the decisions, statistical tests and methods used to achieve the performance models (Section 4.3). Moreover, this chapter presents a study case where an implementation of the Web application model (Section 4.4) is developt as an extension of the CloudSim simulator (CALHEIROS *et al.*, 2011) to predict CPU performance to support energy efficiency domain.

## 4.1 Monitoring and Tracing: RUBiS

The workload behavior can be expressed by the type and amount of resources that it demands. Currently, different workloads are running in the cloud, such as: Web Serving, scientific HPC, social networking, and gaming, among others. FOSTER *et al.* (2008) emphasize that the cloud environment is suitable for interactive real-time applications. However, researchers have been focusing on provisioning and scheduling techniques for batch applications while cloud resource management in interactive applications is not as explored (GROZEV; BUYYA, 2013; DUONG *et al.*, 2012). In view of this, we utilize the reference RUBiS [1] benchmark in order to evaluate the impact of the user in the resources consumption patterns. In the context of cloud computing, it has been employed in the construction of a performance model of a 3-tier application in cloud environments (GROZEV; BUYYA, 2013) and on the evaluation of a Benchmark-as-a-Service platform, enabling the determination of the bottleneck tier and to tune the Java Enterprise Edition (JEE) servers to improve application performance (TCHANA *et al.*, 2014). The methodology is not limited to Rice University Bidding System (RUBiS) and it can be extrapolated to other workload categories. Feitelson (FEITELSON, 2014) discusses several use cases covering different workloads, where the methodology can be applied.

The RUBiS benchmark is an auction site, based on eBay.com, implementing functionalities, such as selling, browsing, and bidding. It is widely used in the literature to evaluate the scalability of application servers. This benchmark is composed of the RUBiS Client and

[1] http://rubis.ow2.org/

Tomcat and MySQL servers. The Client is responsible for simulating the user's behavior by generating requests that are processed by the servers. User's behavior comprises different interactions.Among them, search for items by category or region, buy or sell items, postcomments, etc. The RUBiS Client provides two user profiles: browsing profile,including only read interactions with the database, and the bidding profile,which includes 85% read and 15% reading and writing in a database(CECCHET *et al.*, 2003).

An e-commerce Web application has an interesting feature that is the dependencebetween user actions, *i.e.*, a particular action depends on the previous one. For example, a user can not buy a product that has not been selected. RUBiS represents this characteristic through a hierarchical model based on Markovchain (BOLCH *et al.*, 2006), where there are tables specifying the probability of a user who is on a given page, to visit another page or exit the system. In this work, we achieved the inter-dependence between actions by a generative hierarchical model, which mimics the process that generates the observed behavior (FEITELSON, 2014).

Since the load is generated by the users, we create a model that simulates the dynamic behavior of the users using the system. Such behavior involves the concepts of user population, user session, and think time. The user population is the set of users who are interacting with the system at the same time. It covers the model of arrival of new users as well as the residence time in the same system. Thus, the user population defines the sizes of user sessions. The residence time from users in the system is variable and fluctuates around a certain value (FEITELSON, 2014). Our experiments involve a single user, however the population size and arrival model of new users are parameters of the proposed model, so they are easily configured.

The user session consists of the pattern of a user's activity during the period that she uses the system. In the case of e-commerce, this pattern is a sequence of request, such as interactions of searching, adding to the shopping list, and making a purchase. The time interval between two requests of the same session is called think time (FEITELSON, 2014). The intensity with which the user uses the system is determined by user profile (browsing or bidding). RUBiS reproduces the think time by negative exponential distribution with mean equals to 7 seconds, as defined by TCP-W benchmark (MENASCE, 2002). It uses the same distribution for setting the session time of the user in the system but, in this case, the mean of distribution is equal to 15 minutes.

Table 4 – Private cloud: Physical sever and VMs specifications

| Instance Name | Configuration |
|---|---|
| Rubis Client | m1.small \| 2GB RAM \| 1 VCPU \| 20GB Disk |
| Servers | m1.medium \| 4GB RAM \| 2 VCPU \| 40GB Disk |
| Physical Server | 64GB RAM \| 12 VCPU (Xeon 6C E5-2620) \| 2.1TB Disk |

### *4.1.1 Testbed Configuration*

The experimental environment consists of a private cloud comprising 3 high-performance servers interconnected through a Gigabit Ethernet switch. The management of physical resources is accomplished via OpenStack [2] (Grizzly), which provides compatibility with Amazon EC2 and S3, thus allowing for migration of applications between these platforms. Our experiments have 2 types of VMs, small and medium, whose hardware configurations are described in Table 4. For compatibility with the monitoring tool Bro (PAXSON, 1999), the RUBiS version 1.4.3 is used.

The client VM is configured with the RUBiS Client, which contains the load generator. The server VM hosts a Linux, Apache, MySQL and PHP (LAMP) stack. The Linux version on the server is the same used on the client (Ubuntu 12.04.2 LTS).We opted for the multi-threaded version of Apache Web server (2.2.22) because it is faster and consumes less memory in most situations (ZERIGO, 2014).The MySQL relational database (14.14) is configured with a maximum number of connections equal to the number of virtual CPUs on the server VM. Finally, we use version 5.3.10 of the PHP scripting language. This VM also hosts the Apache and MySQL servers. The monitoring tools running both the client and the servers are detailed in Section 4.1.2.

### *4.1.2 Servers Metrics*

The metric CPU is monitored on the server in order to verify the behavior of the use of physical resources while processing requests from a user. These data are essentially important to define a profile of resource utilization, which assists directly in the management of physical resources. The monitoring of resources is accomplished through the Linux sysstat package (SYSSTAT, 2013), which provides the “sar” tool. This tool allows for monitoring of individual resources, including CPU utilization (%) per core. The sampling interval was set to 1 second for all monitored resources.

Another metric observed on the server is the number of instructions. The RUBiS

[2] http://www.openstack.org/

benchmark randomly generates the size of user session from a negative exponential distribution. This value is static for any CPU that processes the session. Changing this to time, according to the CPU clock, is reasonable if all CPUs have the same characteristics (architecture, core number, manufacturer, *etc.*). In the case of the characteristics of CPU from VMs are heterogeneous, estimating the execution time is a difficult task. Therefore, the aim of capturing this metric is to create a model to characterize the user session in terms of the number of instructions executed, Million Instructions Per Second (MIPS). Thus, the session execution time will vary according to the processing capacity of the hardware.

Two services are directly affected by users requests in the servers (MySQL and Apache). Thus, we measure the total number of instructions performed by these services during the execution of a user session. To this end, we made use of the Pin instrumentation framework (LUK *et al.*, 2005) that offers a tool called "inscount" able to count the number of instructions used by an application.

### *4.1.3 Client Metrics*

One of the main motivations for the use of Benchmark is the ability to measure the QoS experienced by users in several controlled and repeated scenarios. However, a recent study (HASHEMIAN *et al.*, 2012) shows that the response time presented by the RUBiS Client is very inaccurate due to the Java Virtual Machine (JVM) version and the TCP connection management policy provided by Java networking library used by the benchmark. In order to work around the problem, we use a more current JVM (1.7) and measure the response time by a set of independent tools, *i.e.*, we do not consider the response time reported by the RUBiS Client. To this end, we adopt the same definition and methodology for monitoring response time proposed in this study in order to obtain a realistic measure and avoid erroneous conclusions. In this thesis, the QoS is evaluated through the response time, which is a metric of interest for Web applications, that must respond to users in a timely manner. This information is useful for decision making with respect to resource management and provisioning.

Once the connection between client and server is established, the client sends an HyperText Transfer Protocol (HTTP) request to the server, which in turn confirms the arrival of the request through an acknowledgment package. After processing the request, the server sends the HTTP response via one or more packets. The time between the delivery of the HTTP request and receiving the first packet of the HTTP response is called reply time. The time to transfer all

Table 5 – Experiment # 1 - RUBiS client configuration

| Configuration | Value |
|---|---|
| Number of Clients | 1 |
| User Profile | Browsing/Bidding |
| Session Run Time | 15 minutes |
| Think Time | 7,8 seconds |
| Up Ramp Time | 2 mim |
| Down Ramp Time | 1 mim |

the packets that constitute the HTTP response is called transfer time and the response time is the sum of the reply time and transfer time. We use the tools “tcpdump” (TCPDUMP/LIBPCAP, 2013) and “Bro” to measure the response time of HTTP requests experienced by the user.

### *4.1.4 Experimental Design*

To accurately model the impact of user behavior on the processor, we define the total number of instructions that are executed and how they arrive in the processor over the course of a user session. Therefore, we observed the CPU utilization behavior during the user session, which is the second metric analyzed in this study.

The sample of the two observed metrics was obtained through 100 executions of the same experiment, which consists of the submission of requests from a single client to Apache and MySQL servers. The data is captured throughout the processing of the user session on the server. This process is repeated for the two user profiles presented by RUBiS. The RUBiS Client was configured using the parameters presented in Table 5. Considering a multi-tier application in a cloud environment, LLOYD *et al.*(2013) show that that the inter-tier network impact on the overall performance variability is negligible compared to the impact of the CPU load. Therefore, since Apache and MySQL are in the same VM, the effect of internal network is not considered in our model.

## 4.2 Monitoring and Tracing: LisaMini

The cloud environment hosts multiple purposes applications, from running batch jobs to Web applications, which have heterogeneous and competing QoS requirements (CALHEIROS *et al.*, 2011). Vondra and Sedivy (VONDRA; SEDIVY, 2013) indicate an efficient way to maximize the resource utilization of cloud computing infrastructure through the mix of interactive and non-interactive workloads. To address a realistic and efficient scenario, we adopted a

heterogeneous mix composed for a non-interactive (scientific application) and an interactive (web application) workloads:

The lung and lower respiratory infections diseases are in the top major killers during the past decade. Furthermore, lower respiratory infections are the higher cause of death in low-income countries [3]. In this scenario, the LisaMini workload consists in the use of computational vision techniques for lung segmentation from chest Computerized Tomography (CT) scans. This tool supports the diagnosis, assessment of lung disease progression and responses to treatments.

The LisaMini procedure is as follows: the patient exam images are loaded, then the lung segmentation is performed and, finally, quantitative data of segmentation are calculated (volume, density, a percentage of lung aeration) and stored in a log, as a batch application. There are no user interaction and no network communication during the LisaMini execution. The LOLA11 [4] set of chest CT scans is processed by LisaMini.

### *4.2.1 Testbed Configuration*

The images processing and segmentation time are essential for a fast diagnosis. Consequently, if there is an emergency demand, the elasticity provided by the computational clouds allows for you to expand the number of resources allocated to minimize the processing time. In addition, the cloud provides a robust infrastructure where the application is available at any time. All monitoring experiments were performed using three different instances of Microsoft Azure, such as A2 (4 cores, 7GB of memory), A0(1 core, 1.75 GB of memory), and A1(2 cores, 3.5 GB of memory). Theses instances were setting with Linux (Ubuntu 14.04 LTS).

### *4.2.2 Metrics*

Six different operational metrics were monitored in order to verify the physical resources usage patterns during the application execution: (i) CPU utilization, consists of CPU utilization percentage promoted by user processes and system, taking into account all core instance. Once the environment is dedicated, the impact on the system CPU consumption is small; (ii) CPU I/O wait corresponds to the CPU percentage which is waiting for I/O operations, considering all instance cores; (iii) memory utilization, percentage of memory used; (iv) memory swap, percentage of swap memory used; (v) disk read rate, number of disk sectors read per

[3] http://who.int/mediacentre/factsheets/fs310/en/
[4] http://lola11.com/

second; (vi) disk write rate, number of disk sectors written per second. The network was not taken into account as the application does not use this resource. The disk read and write rate is important to plan the input-output operations per second (iops) of the storage system that can be sustained. These metrics are essentially important to define a profile of resource utilization and their monitoring was accomplished through the Linux sysstat package [5].

### *4.2.3 Experimental Design*

#### *4.2.3.1 Monitoring Experiments*

The unpredictability of the cloud application performance is a challenge for researchers and providers (ARMBRUST *et al.*, 2010). Different factors can interfere in the performance variation, such as location, instance type, processor models, resources not virtualized, daily cycle, week cycle (SCHAD *et al.*, 2010). In this way, the LisaMini application was performed for lung segmentation of 10 patients from the LOLA11 set of chest CT scans (11-20), while operating metrics are monitored and stored in logs. The number of 10 patients was chosen in order to ascertain the difference in processing patterns of different patients.

Herein, the instance types and location factors were considered, resulting in the two experiments. Table 6 summarize the experimental design.

**Experiment 1 (Different instances):** 3 Azure's instances were considered, all of them, located in the Western United States to address the question "Do different instances affect the resources usage patterns sufficient to justify specific models for each of them?"

**Experiment 2 (Different locations):** 2 instances (A1, A2) were started in 3 locations (East USA, Occidental Europe, Southeast Asia) to address the question: "Do different locations of the same instance affect the resources usage patterns sufficient to justify specific models for each of them?"

The sampling interval was set to 1 second for all monitored resources. The observed metrics figures represent the median of 5 executions of the same experiment. The median was selected by reason of to be a more robust statistic that average, *i.e.*, less susceptible to outliers. A larger number of executions were collected, but all patterns were identified in 5 executions. The average statistics ($\mu$) and variance ($\sigma^2$) were used to assist the resource usage patterns comparison in the experiments. Regarding the metrics, the change in the number of patients only

[5] http://sebastien.godard.pagesperso-orange.fr/

Table 6 – Experimental design.

| **Characteristics** | **Description** |
|---|---|
| System | Microsoft Azure (A0, A1, A2 instances) |
| Metrics | CPU utilization, CPU I/O wait, memory utilization, memory swap utilization, disk write rate, disk read rate |
| Fixed parameters | Experiment 1: workload, location; Experiment 2: workload, instance (A1 and A2) |
| Configurable factors | Experiment 1: instances (A0,A1,A2); Experiment 2: location (East USA, Western Europe, Southeast Asia) |
| Evaluation technique | Measurements |
| Workload | LOLA11 (11-20) |
| Data analysis | Interpretation of normality testes, shape characteristics, graphics, the median behavior of operational metrics |
| Results presentation | Tables, histograms, line charts, qq-plots, density plots |

impacts in time execution. No additional pattern was found.

The swap memory consumption remained zero during application execution in all experiments. Thus, its behavior is not shown.

## 4.3 Workload Profiling

### *4.3.1 Exploratory Analysis*

Understanding the characteristics of the available sample is important to decide in advance which distributions better approximate the collected data. We performed an exploratory data analysis through the statistics summarized in Table 9.

Since the data is quantitative, one way to characterize them is to capture the sample location measurements. These measures give an idea of how the data are concentrated in relation to a central measure, such as the mean and median. Measures of dispersion and shape, such as skewness and kurtosis, are also used.

In the direction of the goal, the first step was to identify the variables types. All metrics observed in this study are quantitative and continuous. Then, the data normality was verified via the Shapiro-Wilk's test (SHAPIRO; WILK, 1965). Holding the test results, the changes were made in order to change the data scale to make the normality assumption plausible. Finally, the log, square-root, and inverse transformations were applied to data.

Table 7 – Descriptive statistics: location, dispersion and shape.

| Statistic | Type |
|---|---|
| minimum | location |
| 1st Quartile | location |
| median | location |
| mean | location |
| 3rd Quartile | location |
| maximum | location |
| std. deviation | dispersion |
| skewness | shape |
| kurtosis | shape |

### *4.3.2 Parameters Estimation*

Since the characteristics of the sample are known, the distributions candidates are selected considering both user profiles. The Generalized Lambda (GL) distribution is able to represent different shapes, including those with negative skewness (RAMBERG; SCHMEISER, 1974). Therefore, we use it as an alternative to represent the observed number of instructions. The GL is a generalization of the four parameters of lambda distribution family (HASTINGS *et al.*, 1947): the first parameter is the location, represented by the median ($\mu$), the second parameter is the scale, represented by the inter-quartile range ($\sigma$), the last two parameters are related to shape, featuring the skewness ($\alpha_3$) and the steepness of the distribution ($\alpha_4$), respectively.

The GL parameterization used in this work was proposed by CHALABI *et al.* (2012a), where the parameters of location and scale are defined in terms of the related quantile statistics. The advantage of this parameterization is that it reduces the empirical data parameter estimation to the problem of estimation of two parameters since the parameters of location and scale can be directly estimated from their robust sampling estimators. The quantile of the sample can always be estimated even when there are no moments of the distribution.

Another alternative selected is the Generalized Extreme-Value (GEV) distribution (JENKINSON, 1955), widely used in the literature to model natural phenomena, especially in the field of hydrology (HOSKING *et al.*, 1985; MARTINS; STEDINGER, 2000). Moreno et al. (MORENO *et al.*, 2013) use GEV to model the consumption of CPU and memory from the data provided by the Google MapReduce Cloud TraceLog (REISS *et al.*, 2011). This distribution has three parameters, location ($\mu$), scale ($\sigma$), and shape ($\xi$) and can represent a variety of forms that include three different distributions: (i) Gumbel, when $\xi = 0$, (ii) Frechet, when $\xi < 0$, and (iii) Weibull, when $\xi > 0$.

In order to represent the CPU utilization, 21 different continuous distributions with positive skewness were analyzed for different user profiles, including the Generalized Pareto Distribution (GPD). The GPD is widely used in finance, analysis of extreme events, and reliability studies (HOSKING; WALLIS, 1987; EMBRECHTS *et al.*, 1997). It has 3 parameters: location ($\mu$), scale ($\sigma$) and shape ($\xi$). This distribution has three parameters, location ($\mu$), scale ($\sigma$) and shape ($\xi$) and it includes three different distributions as special cases: (i) Exponential, when $\xi = 0$, (ii) Pareto, when $\xi < 0$, (iii) Uniform, when $\xi = 1$. For comparison, the estimate parameters for the exponential distribution, location ($\mu$) and scale ($\sigma$), also are presented.

For comparison purpose, the parameters of the Normal distribution(symmetric), location ($\mu$) and scale ($\sigma$), and Gumbel distribution(positively skewness), location ($\mu$) and scale ($\beta$) are also estimated by the method of maximum log-likelihood.

The packages "fitgl" (CHALABI *et al.*, 2012a) and "fExtremes" (COLES, 2001) from the statistical tool R (R Core Team, 2013) are used for estimation of the parameters of GEV and GL, respectively.

After choosing the distributions those that are able to represent the data mathematically, it is necessary to estimate the parameters of the models.

### *4.3.3 Goodness of Fit*

In order to perform the GoF tests, the statistics verify whether the empirical and theoretical data belong to the same distribution. In this study, two methods are used to assess if the selected distributions provide a good fit to the data: one graphic method using Q-Q plots, and one analytical method using KS test.

The KS test has an important limitation: the parameters from $F(x)$ must be known in advance and not estimated from the observed data, as performed in this work. There are special tables that can be used in such cases. However, these tables exist only for a restricted set of distributions (STEPHENS, 1974).

In order to circumvent this limitation, we used the bootstrapping provided by the Matching (SEKHON, 2011) package from the R statistical tool. The bootstrapping process starts with the calculation of the value of $D$ from the original data. After that, 100 samples are generated with replacement from $F(x)$ and, then, the KS test is calculated. The sampling followed by calculation of the KS test is repeated 1000 times in order to build the distribution of $\hat{D}$. Thus, the originally calculated value of $D$ is located in the distribution of $\hat{D}$. If $D$ drops in the

distribution body or $p - value > \alpha$, the null hypothesis cannot be rejected. Otherwise, $D$ fall into the tails of the distribution, $p - value < \alpha$, the null hypothesis is rejected.

## 4.4 Model Simulation and Validation

Validation is performed to show that the simulator is able to represent the observed data accurately. CloudSim is a toolkit that enables modeling and simulation of cloud computing systems and environments, including data centers, service brokers, provisioning, and allocation policies. The energy modeling in CloudSim allows for switching on/off of hosts and virtual machine migration for energy savings and integration of energy models. Furthermore, it provides the simulation of the five different modes of DVFS (Dynamic Voltage and Frequency Scaling) from the Linux kernel.

Thereby, the application modeling was developed in CloudSim because this simulator contains abstractions for representing cloud infrastructures, power consumption, and it allows energy-aware simulations, especially using DVFS.

### *4.4.1 Workload Generation*

The process of generating the load simulator is shown in Figure 8, all classes presented by the *Workload Generation* were implemented in this work and extends the CloudSim. At the start of the simulation, the *ECommerceApp* class is instantiated with the parameters number of users and arrival rate of users, which can be a fixed value or generated by a distribution, and, finally, the user profiles (browsing and bidding). With these parameters, the *ECommerceApp* instantiates the *UBehavior* class responsible for encapsulating the users behavior. This behavior is defined by the statistical distributions that represent the total number of instructions and CPU utilization, presented in Section 5.1.3. The models are developed using the R statistical language (R Core Team, 2013) that communicate with CloudSim through the *REngine* library.

Once the user's behavior, i.e. the number of requests and their arrival time and size in terms of a number of instructions, is known, a *USession* is instantiated for each user. Afterwards, *USession* instantiates the set of *Request* that will compose it. In each simulation step, one or more requests are processed until all have been completed.

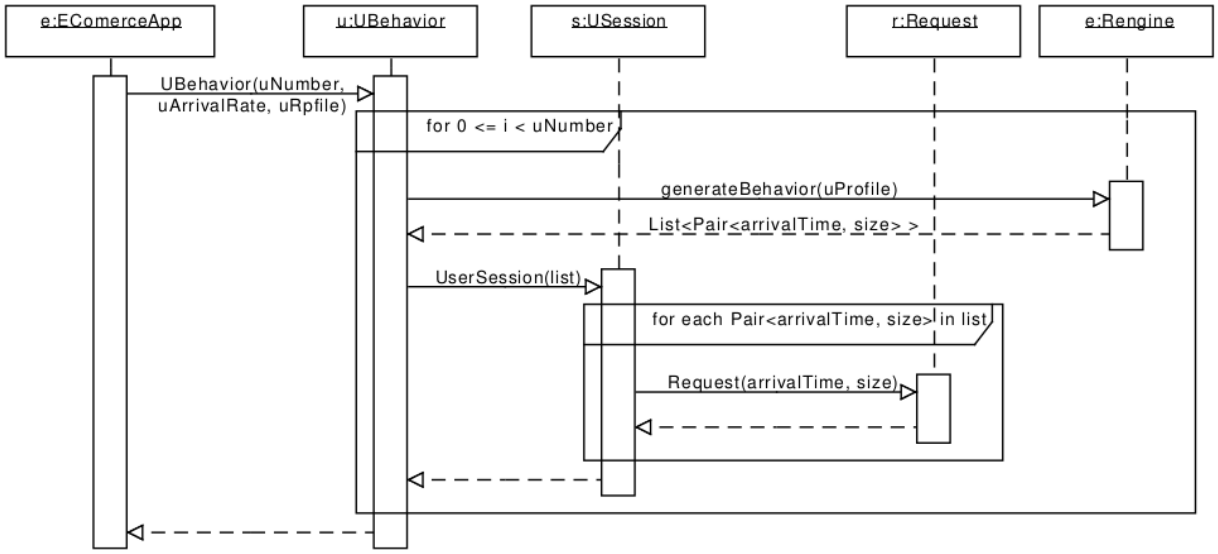



Figure 8 – Workload generation diagram.

### *4.4.2 Simulation Validation*

The WMW statistical hypothesis test, described in the Section 4.4 was used to evaluate the accuracy of the model in the simulator.

The simulations were conducted in a simulated cloud computing data center environment provided by CloudSim, whose hosts and VMs follow the same configuration described in Table 4. The simulation is set according to the scenario where data was collected (see Session 4.1.4). Thus, it is the submission of requests by a single user that is processed by the application server, in this case, the VM. The duration of the session was set according to Table 5 and each experiment consisted of 1000 rounds of simulation.

## 4.5 Prediction and Control

The relationship formalization between the input and output is crucial to determine how the input affects the output. The Fast Fourier Transform (FFT) is used in this work to estimate the transfer functions. It identifies patterns in the time series corresponding to inputs and output. The FFT decomposes a signal time series into components of different frequencies. The dominant frequencies will correspond to the repeating patterns (LORIDO-BOTRAN *et al.*, 2014). The result is presented as follows:

$$TFresptime|CPU = \frac{-1397s \quad + \quad 2.379e04}{s^2 \quad + \quad 21.31s \quad + \quad 12.6}, \tag{4.1}$$

$$TFresptime|mem = \frac{-2.062e04s^3 - 1594s^2 - 7016s + 80.82}{s^4 + 6.645s^3 + 2.589s^2 + 2.385s + 0.5684}, \quad (4.2)$$

$$TFresptime|disk = \frac{-4.112e06s^3 - 3.035e05s^2 - 1.114e06s + 2.088e05}{s^4 + 3.141e04s^3 + 3384s^2 + 1.009e04s + 185.2}. \quad (4.3)$$

The transfer functions formalizations acan be directly used by researchers' practical use, such as, testing, comparisons, and validation purposes.

To assess the ability of MPC to adjust the resources (CPU, disk, memory) as needed to meet a response time acceptable in the user perspective, we carry out some experiments by means simulation in MPC Matlab Toolbox [6].

To evaluate the performance of MPC model, we propose three experimental scenarios: in the first scenario we change the reference trajectory in steps from 300ms to 1s. The objective is to verify if the MPC is able to adapt throughout the simulation, *i.e.*, if it can correct its trajectory through the adjustment of the inputs (MISO); (ii) in the second scenario we consider a stressing test, *i.e.*, to vary the reference trajectory to a point where the system can no longer reflect it. In view of this, we can define a scope for which the MPC given the inputs (MISO) works correctly; In the MISO scenarios, a higher weight is assigned to the CPU as discussed in Section 3.3.1, the weights and other parameters for each scenario are presented in Table 8.

To quantify the performance of the transfer functions and the MPC, three error measures was used: future error, Mean Squared Error (MSE) and MAPE. To define how the transfer functions offer a good match to input/output data, the MSE is calculated. Further, the MPC performance is computed via MAPE based on future errors. The MSE is calculated as follows, where $n$ is the size of data vector:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \quad (4.4)$$

The future error represents the difference between the reference value $w(k)$ and the response time estimated value $\hat{y}$ for $k = 1...N$, where $N$ is the prediction horizon at each instant $t$, as ilustrated in Figure 3 and computed in Equation 4.5:

$$e_{fut}(k) = w - \hat{y}(k), \quad (4.5)$$

[6] https://www.mathworks.com/products/mpc.html

Table 8 – MPC: experimental design parametrization.

| **Parameter** | **Value** |
|---|---|
| Simulation Time | 300 seconds |
| Prediction Horizon | 1200 seconds |
| Control Horizon | 3 |
| Time Sampling | 0.1 |
| **Scenario 01** | |
| Number of Inputs | 3 (CPU, Memory, Disk) |
| Number of Outputs | 1 (Response Time) |
| Setpoint | range [300, 500, 1000] |
| Input Weights | CPU: 0.3 ; Memory: 0.1; Disk: 0.1 |
| **Scenario 02** | |
| Number of Inputs | 3 (CPU, Memory, Disk) |
| Number of Outputs | 1 (Response Time) |
| Setpoint | range [300, 500, 1000, 1850] |
| Input Weights | CPU: 0.3 ; Memory: 0.1; Disk: 0.1 |

Since the future error is known, the MAPE is computed as follows:

$$MAPE = \frac{1}{N}\sum_{k=1}^{N}(\frac{|e_{fut}(k)|}{w}), \tag{4.6}$$

## 4.6 Concluding Remarks

Computational resources metrics, such as CPU, memory, and disk, becomes essential for providers and clients because they are often associated with elasticity (COUTINHO, 2014). In the monitoring experiments, these metrics were observed for two relevant applications: a scientific application for pulmonary diseases diagnoses and a Web application that simulates an auction site. Such applications were deployed in a private and a public cloud.

The experiments were designed and conducted in order to answer the *Research Question #1* and *Research Question #2*: “Do different VM instances and geographic locations affect the resources usage patterns sufficient to justify specific models for each of them?” “How do different user profiles impact resource utilization behavior?”

# 5 RESULTS AND DISCUSSION

This chapter presents the results of the applications monitoring experiments described in Sections 4.1 and 4.2. In addition, the results from the three phases of the proposed methodology are presented, and the statistical models are achieved. Also, the results of the fit tests are presented in order to verify if the simulated data are consistent with the observed data for both users profiles.

The Web application models were implemented as an extension of the CloudSim simulator and the results are discussed in Section 5.2. In addiction, these models are used as input in MPC in order to estimate the resource utilization values to maintain the application response time within an acceptable QoS, the simulation results are presented in Section 5.3.

## 5.1 RUBiS Profiling

### *5.1.1 Statistical Analysis*

Table 9 presents the descriptive statistics related to the sum of the number of instructions consumed by Apache and MySQL services, CPU, memory and disk utilization for both user behavior, and response time for Biding profile. Regarding the number of instructions and memory, the negative value of skewness is reinforced because the median is greater than the average. This characteristic is clearly observed in the histograms of the number of instructions consumed by Apache and MySQL services (Figures 9a and 10a) through the long left tail relative the right tail. The negative skewness was primordial in the choice of distributions used to fit the number of instructions.

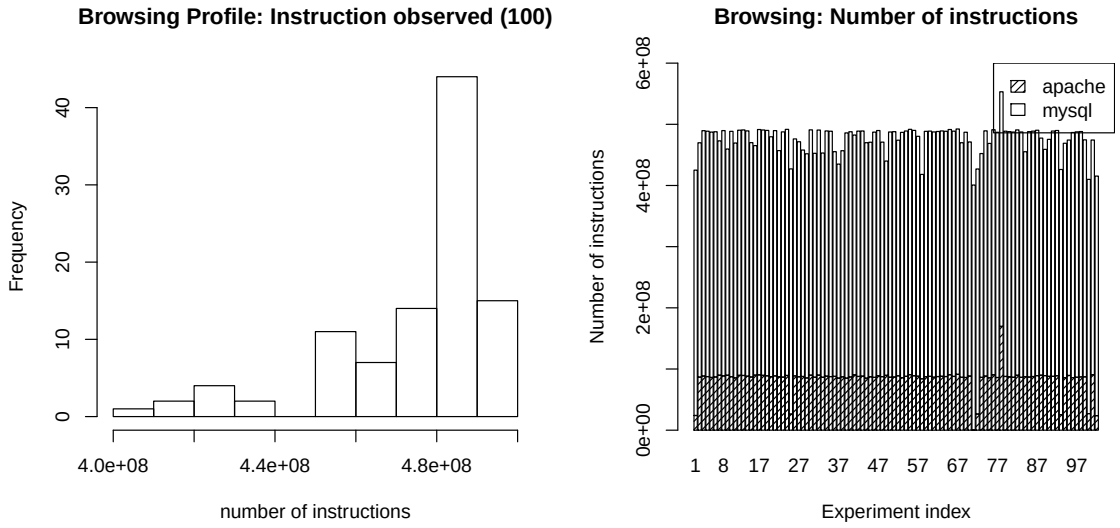
Table 9 also shows high kurtosis values for CPU, disk and response time. This characteristic is seen in Figures 9a and 10a through a well pronounced peak, near the median. These metrics have many time intervals equal or close to zero. Therefore, the non-zero values promote a large scale difference contributing to the presence of peaks.

The change from browsing to bidding profile implies in a memory consumption increment. However, the disk consumption is much lower compared to memory consumption because, in general, e-commerce applications are in memory, *i.e.*, the information is transferred from disk to memory (cache) to avoid slow Web response times.

Figures 9b and 10b show the number of instructions executed by the CPU concerning the Apache and MySQL services for each of the 100 user sessions performed during the

Table 9 – Observed metrics statistics: number of instructions, CPU, memory, and disk utilization, and response time.

| Statistic | Type | Browsing | | | | Bidding | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Instructions | CPU (%) | RAM (%) | Disk (TPS) | Instructions | CPU (%) | RAM (%) | Disk (TPS) | Resp. Time (ms) |
| minimum | location | 4.00e+08 | 0 | 18.92 | 0 | 4.26e+08 | 0 | 58.17 | 0 | 0.74 |
| 1st Quartile | location | 4.69e+08 | 0 | 19.97 | 0 | 4.72e+08 | 0 | 58.76 | 0 | 0.85 |
| median | location | 4.87e+08 | 0 | 20.02 | 0 | 4.87e+08 | 0 | 58.84 | 0 | 0.87 |
| mean | location | 4.75e+08 | 0.59 | 20.01 | 0.11 | 4.79e+08 | 0.55 | 58.82 | 0.08 | 1.00 |
| 3rd Quartile | location | 4.89e+08 | 0.50 | 20.08 | 0 | 4.89e+08 | 0.50 | 58.92 | 0 | 0.90 |
| maximum | location | 4.92e+08 | 82.50 | 20.13 | 26 | 4.91e+08 | 79.90 | 59.01 | 8 | 9.42 |
| std. deviation | dispersion | 2.02e+07 | 4.99 | 0.11 | 1 | 1.38e+07 | 4.80 | 0.16 | 0.43 | 0.68 |
| skewness | shape | -1.67 | 12.28 | -3.10 | 18.05 | -1.52 | 13.00 | -2.05 | 8.28 | 8.80 |
| kurtosis | shape | 2.25 | 156.89 | 17.21 | 418.48 | 2.07 | 175.84 | 5.06 | 107.78 | 91.97 |



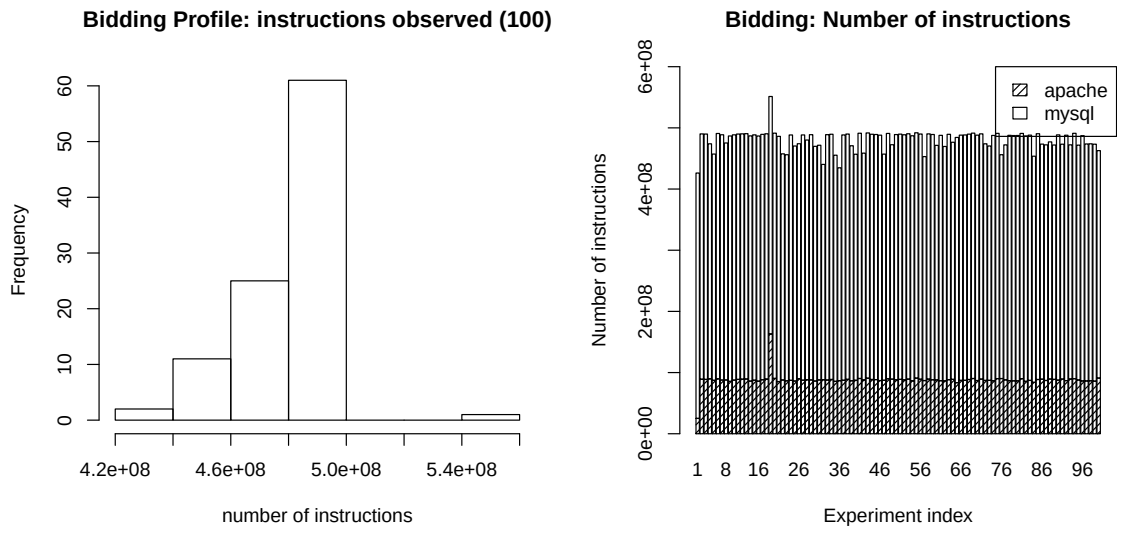

(a) (b)

Figure 9 – Statistical analysis of number of instructions. (a) Histogram of Browsing profile (median of 100 rounds); (b) Number of instruction per user session for Apache and MySQL

experiment. In both profiles, MySQL requires more processing than Apache. Furthermore, none of these services are bottlenecks for the application in this experimental setting.

Figure 11a depicts the scatterplot of the percentage of CPU utilization over a user session, where we found a higher CPU consumption at the beginning of the session. This consumption decays rapidly to zero or close to zero and so continues until the end of the session. The concentration of data in a single well-pronounced peak near the median with fast decay reinforced the high kurtosis presented in Table 4.

Due to the large difference in scale between the percentages of CPU utilization reflected in Figure 11a, the values of the axes are limited in order to observe the CPU utilization




(a) (b)

Figure 10 – Statistical analysis of instructions number. (a) Histogram of Bidding profile (median of 100 rounds); (b) Number of instruction per user session for Apache and MySQL

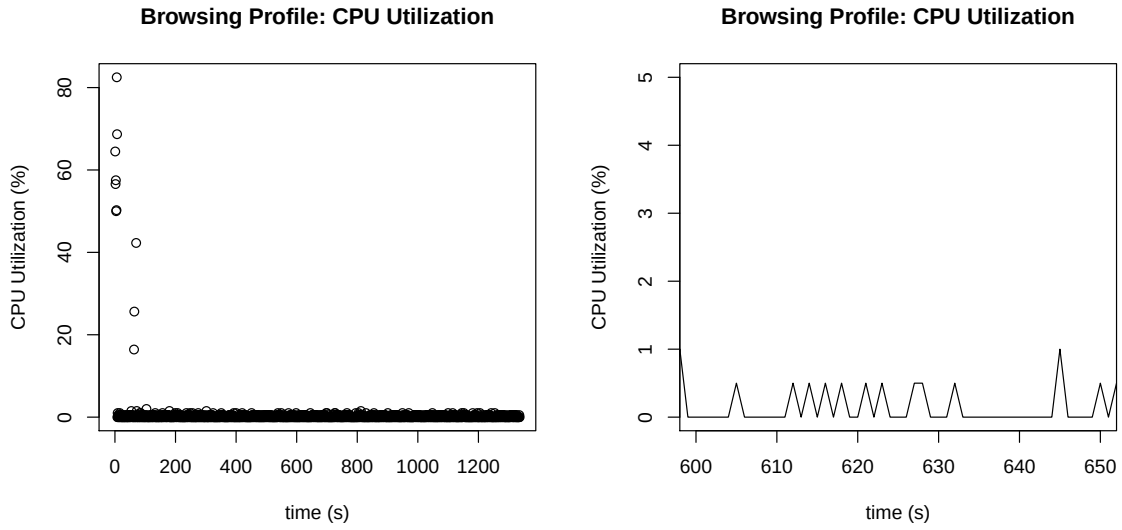
behavior when it is equal or close to zero, as shown in Figure 11b. The same pattern of behavior is identified in the bidding profile.

Figure 11b shows that instructions arrive at the processor in bursts followed by periods of inactivity, because of think times of users, for both profiles. However, the number of instructions in each cycle and the frequency with which they occur varies over time, so there is no pattern about where the peaks and troughs of cycles will happen, indicating a stationary time series. Thus, the data are subjected to a statistical hypothesis test of stationarity, where the Augmented Dickey-Fuller (ADF) (SAID; DICKEY, 1984) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) (KWIATKOWSKI *et al.*, 1992) tests are computed. In these tests, the significance level is fixed at $\alpha = 0.05$. Both tests showed that the data are stationary: for ADF, $p - value = 0.01$ and, for KPSS, $p - value = 0.1$.

Through exploratory analysis conducted in this section, it is possible to select the candidate distributions for the number of instructions metric considering both user profiles, in agreement with the characteristics of the sample.

### *5.1.2 Parameters Estimation*

Table 10 shows the values of the estimated parameters for the selected distributions in combination with the different estimation methods for the number of instructions considering both user profiles. The GL distribution has four estimated parameters, because the sample is the




(a) (b)

Figure 11 – Exploratory analysis of the observed CPU utilization for a browsing user session. (a) Browsing profile: scatterplot of CPU utilization (b) Browsing profile: limited axis with interval between 600 and 650 s

Table 10 – Distributions and parameters for number of instructions.

| Distribution | Estimation Method | Estimated Parameters | |
|---|---|---|---|
| | | Browsing | Bidding |
| GL | MLE | $\hat{\alpha_3} = -9.509e-01$, $\hat{\alpha_4} = 8.055e-01$ | $\hat{\alpha_3} = -9.414e-01$, $\hat{\alpha_4} = 9.814e-01$ |
| | HF | $\hat{\alpha_3} = -9.635e-01$, $\hat{\alpha_4} = 3.264e-02$ | $\hat{\alpha_3} = -4.442e-01$, $\hat{\alpha_4} = 4.790e-01$ |
| | QM | $\hat{\alpha_3} = -9.744e-01$, $\hat{\alpha_4} = 1.347e-02$ | $\hat{\alpha_3} = -9.539e-01$, $\hat{\alpha_4} = 2.102e-02$ |
| | MPS | $\hat{\alpha_3} = -9.504e-01$, $\hat{\alpha_4} = 9.834e-01$ | $\hat{\alpha_3} = -9.483e-01$, $\hat{\alpha_4} = 9.777e-01$ |
| GEV | MLE | $\hat{\xi} = -3.015e+00$, $\hat{\sigma} = 4.230e+07$ | $\hat{\xi} = 7.116e-01$, $\hat{\sigma} = 6.077e+07$ |
| | PWM | $\hat{\xi} = -1.324e+00$, $\hat{\sigma} = 1.839e+07$ | $\hat{\xi} = -8.767e-01$, $\hat{\sigma} = 1.523e+07$ |

same for all estimation methods. The median values for the browsing profile ($\hat{\mu} = 4.873e+08$) and for the bidding profile ($\hat{\mu} = 4.877e+08$), as well as inter-quartile range for the browsing profile ($\hat{\sigma} = 1.919e+07$) and for the bidding profile ($\hat{\sigma} = 1.735e+07$) remain constant for all the combinations.

In contrast, the shape parameters ($\alpha_3$) and ($\alpha_4$) of the GL distribution have their values influenced by the estimation method. Similarly, the value of the location parameter for the browsing profile ($\hat{\mu} = 4.783e+08$) and for the bidding profile ($\hat{\mu} = 4.820e+08$) of the GEV distribution remains constant, while the scale parameters ($\sigma$) and shape ($\xi$) are influenced by the estimation method. Therefore, if the selected fitting distributions have shape parameters, it is important to verify the existence of outliers in the sample in order to choose the appropriated estimation method. Figure 12 shows how sensitive the *MLE* and *PWM* estimation methods are to the presence of outliers.

Table 11 – *MLE* parameters estimation of response time and CPU, memory and disk utilization metrics.

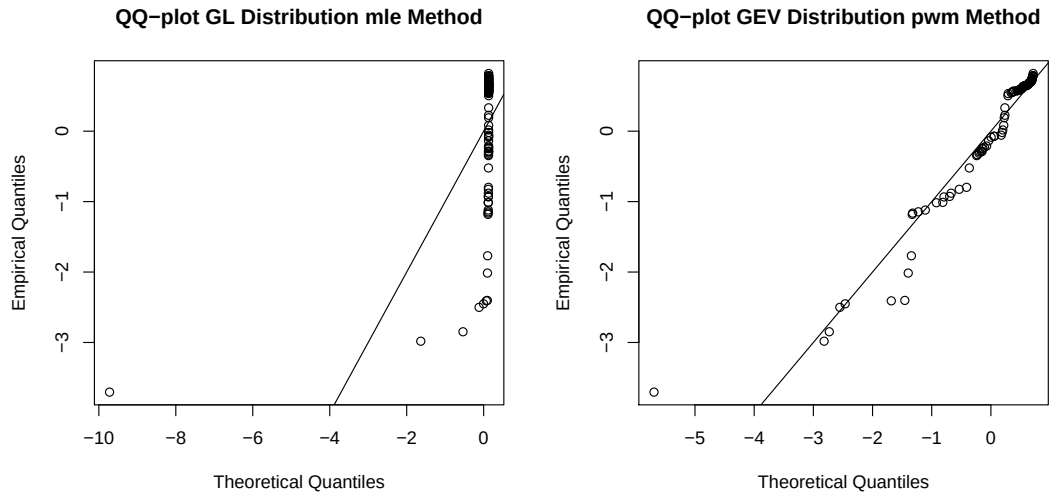
| Metric | User Profile | Distribution | Estimated Parameters |
|---|---|---|---|
| CPU Utilization | Browsing | GWD | $\hat{\xi} = 5.386$, $\hat{\mu} = 0.976$, $\hat{\sigma} = 0.014$ |
| | Bidding | GEV | $\hat{\xi} = -0.259$, $\hat{\mu} = 0.576$, $\hat{\sigma} = 0.004$ |
| Memory Utilization | Browsing | GEV | $\hat{\xi} = -0.221$, $\hat{\mu} = 0.049$, $\hat{\sigma} = 8.526e-06$ |
| | Bidding | Error(3P) | $\hat{\xi} = 2.062$, $\hat{\mu} = 7.669$, $\hat{\sigma} = 3.0985e-04$ |
| Disk Utilization | Browsing | GEV | $\hat{\xi} = -0.369$, $\hat{\mu} = 0.977$, $\hat{\sigma} = 0.003$ |
| | Bidding | Error(3P) | $\hat{\xi} = 2.042$, $\hat{\mu} = 0.967$, $\hat{\sigma} = 0.003$ |
| Response Time | Bidding | GEV | $\hat{\xi} = -0.290$, $\hat{\mu} = 1086.7$, $\hat{\sigma} = 10.898$ |

Table 11 contains the parameters estimated through the *MLE* method, for the distributions that represent the CPU, memory and disk utilization, and response time metrics and offer the best fit for the data, according to the KS test. The GEV distribution best fits CPU (Bidding), memory (Browsing), disk (Browsing), and response time (Bidding) enhancing the results found in Moreno et al. (MORENO *et al.*, 2013) that uses GEV to model the consumption of CPU and memory from the data provided by the Google Cloud TraceLog (REISS *et al.*, 2011). The other scenarios are covered by the Generalized Weibull Distributions (GWD) and Error (3P). All these distributions have a shape parameter that allows a better fitting.

### *5.1.3 Goodness of Fit*

Figure 12 shows the Q-Q graphs for the following pairs distribution/estimation method: GL/*MLE* and GEV/*PWM*. For each graph, the reference line is plotted. It can be noticed that the pair GEV/*PWM* quantiles (Figure 12b), in terms of the observed data quantiles, have greater proximity to the reference line. Thus, this pair is a strong candidate to represent the observed number of instructions behavior. It is interesting to compare the pairs GEV/*PWM* and GL/*MLE* (Figure 12a). So, it can be observed how sensitive the *mle* estimation method is to the presence of outliers. Therefore, if the data set has significant presence of outliers, the mle is not the best option.

Table 12 presents the values of $D$ and $p-value$ test statistics for the number of instructions observed to the distributions of probability specified in Section 4.3.2. Considering the browsing profile, there are only 3 cases where the null hypothesis cannot be rejected because the $p-value > 0.05$: GL/*MLE*, GL/*MPS* and GEV/*PWM*. Considering the profile bidding, there are 2 cases where the null hypothesis cannot be rejected: GL/*mle* and GL/*MPS*. Other important information that can be inferred from this table is that the results are sensitive to the applied estimation method. The GL distribution with *HF* and *QM* estimation methods presents




(a) (b)

Figure 12 – Quantile-Quantile plots. (a) GL with mle method; (b) GEV with PWM method

performance near or below to the symmetric and positive asymmetric distributions.

The estimation methods *PWM* and *MPS* perform better than the *MLE* method for both GEV and GL distributions, for both profiles. This can be justified by the fact that the *MLE* method is equivalent to maximizing the geometric mean. Therefore, it is highly sensitive to outliers. Figure 12a shows the presence of outliers in the data. Furthermore, the *MLE* method provides good results for a small sample size, while the *PWM* and *MPS* are more robust methods.

The GEV distribution with parameters estimated using the *PWM* method presents the best fit to browsing profile. In contrast, the GL distribution with parameters estimated using the *MPS* method presents the best fit for the bidding profile.

In accordance with the presented results, the models are defined representing the number of instructions executed during a user session for both user profiles. These models aim to characterize user session based on the number of instructions instead of session runtime. Thus, the session runtime will vary according to the characteristics of the CPU. On the other hand, if the model characterized the session runtime, this value would be constant regardless the processor. Also, establishing a relationship between session runtime for different processors in a heterogeneous environment is a complex task, since several factors impact the runtime such as a number of cores, clock frequency, and architecture.

The GoF tests defined the distribution/parameters pairs more apt to represent the observed metrics. These pairs are used to compose the model that represents the resources demand of the Web application according to user profiles. Table 13 shows the $p - value(max)$,

Table 12 – Kolmogorov-Smirnov test: number of instructions.

| Distribution/Estimation Method | Browsing | | Bidding | |
|---|---|---|---|---|
| | D | p-value | D | p-value |
| GL/MLE | 0.130 | 0.368 | 0.120 | 0.431 |
| GL/HF | 0.210 | 0.016 | 0.240 | 0.009 |
| GL/QM | 0.230 | 0.012 | 0.240 | 0.006 |
| GL/mps | 0.120 | 0.465 | 0.120 | 0.433 |
| GEV/mle | 0.200 | 0.030 | 0.450 | 2.2e-16 |
| GEV/PWM | 0.100 | 0.675 | 0.160 | 0.146 |

Table 13 – Kolmogorov-Smirnov test: response time and cpu, memory and disk utilization.

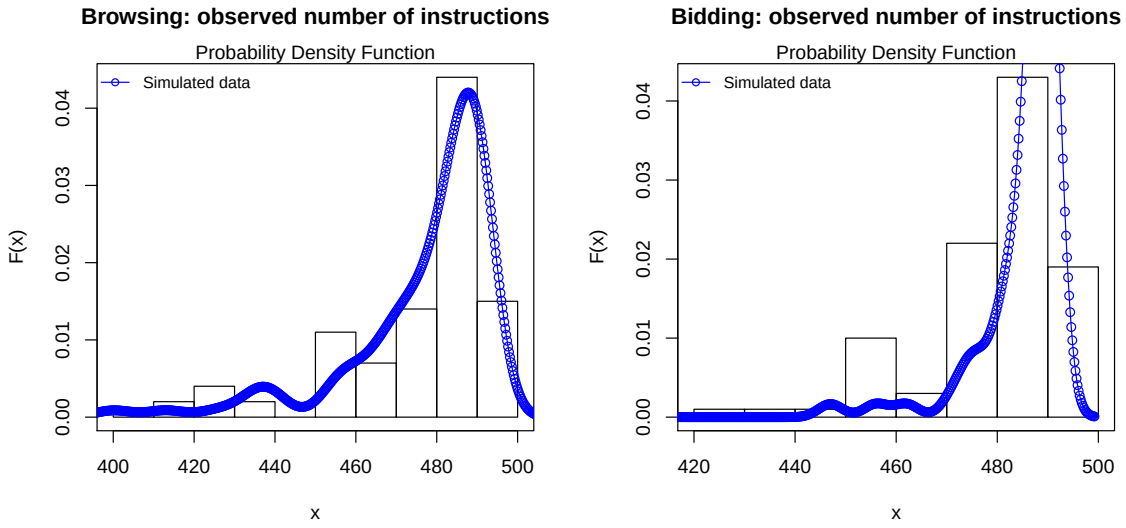
| Metric | Browsing | | | Bidding | | |
|---|---|---|---|---|---|---|
| | p-value (max) | p-value (min) | error (%) | p-value (max) | p-value (min) | error (%) |
| CPU utilization | 0.901 | 0.007 | 6 | 0.982 | 0.008 | 1 |
| Memory utilization | 0.998 | 0.065 | 0 | 0.992 | 0.018 | 2 |
| Disk utilization | 0.994 | 0.032 | 1 | 0.994 | 0.044 | 1 |
| Response Time | - | - | - | 0.983 | 0.036 | 1 |

$p-value(min)$ and $error(\%)$ calculated based on 500 replications of KS test. An error is computed when sampled simulated data does not belong to the same sample data observed, i.e., $p-value < 0.05$. No resource has an error greater than 6%, indicating that the models can correctly represent the collected data.

Therefore, the Browsing model is represented by the GEV distribution with $\hat{\xi} = -1.324e+00, \hat{\mu} = 4.873e+08, \hat{\sigma} = 1.839e+07$ parameters, representing the total number of instructions that will compose the user session, and distribution GWD with $\hat{\xi} = 5.386, \hat{\mu} = -0.976, \hat{\sigma} = 0.014$ parameters, representing how the total of instructions will be distributed throughout the session based on CPU utilization.

The Bidding model, on the other hand, is modeled by the GL distribution with $\hat{\mu} = 4.877e+08, \hat{\sigma} = 1.735e+07, \hat{\alpha_3} = -9.483e-01, \hat{\alpha_4} = 9.777e-01$ parameters, representing the total number of instructions, and the distribution GEV with $\hat{\xi} = -0.259, \hat{\mu} = 0.576, \hat{\sigma} = 0.004$ parameters, representing the CPU utilization. The models that represents the disk and memory demands of the two profiles and the response time experienced by the Bidding profile are presented in Table 11.

Due to trace or model unavailability, unrealistic assumptions are made in the literature about the workload (LI, 2010), such as a set of requisitions with fixed inter-arrival times, simple Poisson models to represent instruction arrivals and exponentially distributed session time. However, in this work, we achieved different results from assumptions commonly made, such as




(a) (b)

Figure 13 – Comparison of histogram of observed number of instructions and probability density function of simulated number of instructions (a) Browsing profile; (b) Bidding profile.

stationarity in the data observed, the distribution representing the arrival of instructions in the processor is GWD, and the distributions that represent the total number of instructions, which is a metric correlated with session time, are GEV and GL.

## 5.2 Simulation and Validation

The graphical validation is shown in Figure 13, while the WMW test results are reported in Table 14. Figure 13 shows a comparison of the histogram of observed data against probability density function of the simulated data to the number of instructions metric, considering both user profiles. The simulated data are consistent with the observed data for both profiles. However, visually, the Browsing profile provides a better approximation to the observed data. This result is reinforced by Table 14, where the Browsing profile has an error three times smaller than the Bidding profile.

Table 14 shows the maximum and minimum $p-values$ obtained through the WMW test applied to a set of 100 samples of observed data and 100 samples of simulated data. For the total of 10,000 comparisons, the error was also calculated. The error is computed when sampled simulated data does not belong to the same sample data observed, i.e., $p-value < 0.05$. The error for all metrics is less than 10%. Among the mathematical transformations applied to the CPU utilization metric, the inverse transformation offers the best results, reducing the Wilcox

Table 14 – WMW test

| Metric | Browsing | | | Bidding | | |
|---|---|---|---|---|---|---|
| | p-value (max) | p-value (min) | error (%) | p-value (max) | p-value (min) | error (%) |
| Number of instructions | 0.999 | 0.033 | 3.000 | 0.995 | 0.012 | 9.000 |
| CPU utilization | 0.996 | 0.026 | 6.000 | 0.998 | 0.033 | 8.000 |
| Memory utilization | 0.992 | 0.046 | 4.000 | 0.999 | 0.016 | 1.000 |
| Disk utilization | 0.998 | 0.034 | 1.000 | 0.999 | 0.029 | 8.000 |
| Response time | - | - | - | 0.998 | 0.024 | 4.000 |

error rate of 34% to 6%, considering the Browsing profile.

Then, we can conclude the implementation of web application modeling in the CloudSim simulator is capable of producing data to represent both user profiles accurately. Thus, it can be used by researchers to build performance models and to produce trace logs based on realistic scenarios and extrapolating the results with controlled modification of parameters such as a number of users, software stack, and physical and virtual machine configuration. Furthermore, this implementation contributes to the development of performance models to support emerging cloud computing research domains, such as resource allocation in Mobile Cloud Computing (MCC) in which the trade-off between time and energy is a management challenge (GHASEMI-FALAVARJANI *et al.*, 2015).

## 5.3 Prediction and Control

Table 15 shows through MSE error how the transfer functions offer a good fit to input/output data. Figure 14 illustrates the response time behavior throughout the simulation and shows that, for the range between [300ms - 1000ms], the MPC can adjust itself to the output meet the reference trajectory and at no time exceeds the established threshold, meeting the expectations of the user. Therefore, the controller model proposed in this work was effective for the Web application in the designed scenario with error $MAPE = 4.36442(\%)$. PADALA *et al.* 2009 propose a combination of an online model estimator and a MIMO resource controller. The model estimator captures the complex relationship between application performance and resource allocations, while the MIMO controller allocates the right amount of multiple virtualized resources to achieve application Service Level Objective (SLO). They use RUBiS in the experimental evaluation and achieve a $MAPE = 4.2(\%)$.

Figure 15 shows the behavior of the response time for a setpoint range varying from 300 to 1850 milliseconds. We note that at 1800 ms, the controller can not make the input

Table 15 – Estimated MSE error for transfer functions.

| | TF (resptime ~ cpu) | TF (resptime ~ mem) | TF (resptime ~ disk) |
|---|---|---|---|
| MSE | 120.7 | 118.5 | 118.6 |

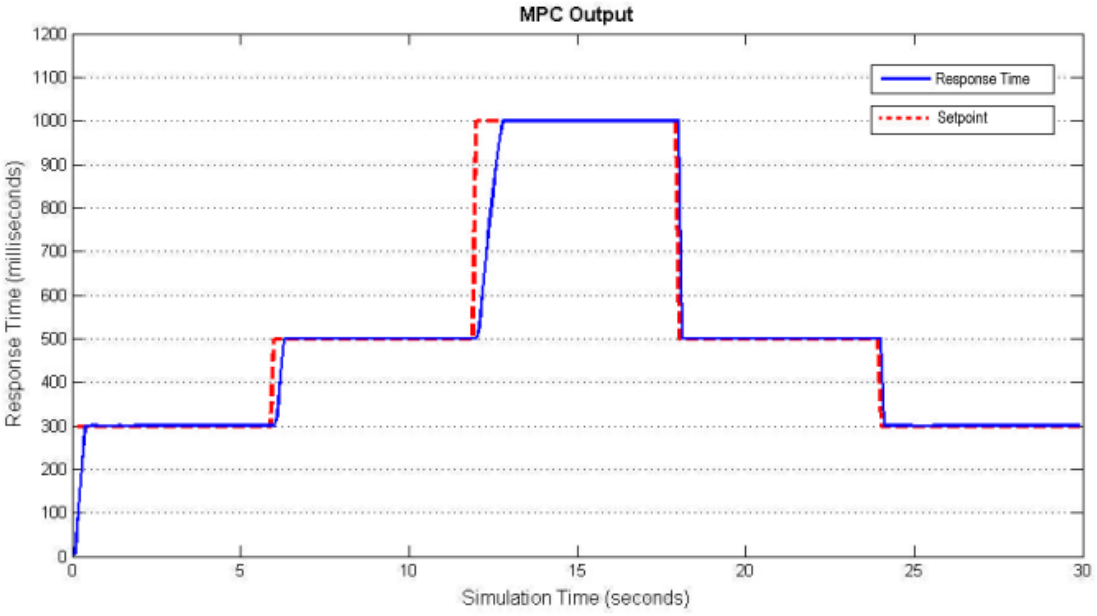



Figure 14 – MISO Model (scenario 01): response time behavior with setpoint range [300ms-1000ms]

adjustments to promote a response time near the setpoint. Therefore, the controller modeling is effective up to the setpoint value of 1800 milliseconds. As can be observed in this scenario, a higher error was achieved ($MAPE = 6.27517(\%)$).

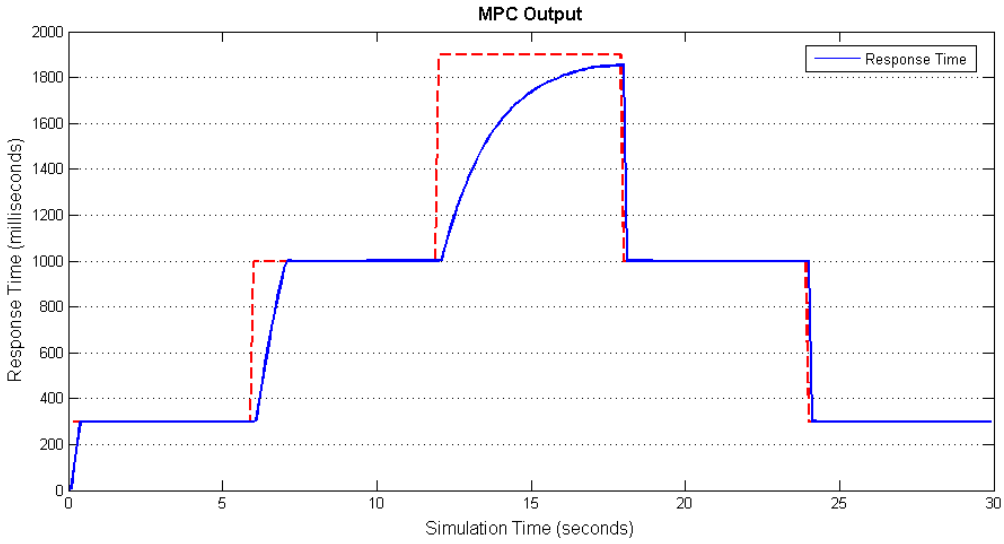



Figure 15 – MISO Model (scenario 02): response time behavior with setpoint range [300ms-1850ms]

## 5.4 LisaMini Profiling

### *5.4.1 Monitoring*

The sampling interval was set to 1 second for all monitored resources. The observed metrics figures represents the median of 5 executions of the same experiment. The median was selected by reason of to be a more robust statistic that average, i.e, less susceptible to outliers. A larger number of executions were collected. Nonetheless, additional patterns were not identified to a higher number of executions 5. The average statistics ($\mu$) and variance ($\sigma^2$) were used to assist the resource usage patterns comparison in the experiments. Regarding the metrics, the change in the number of patients only impacts in time execution. No additional pattern was found.

The swap memory consumption remained zero during application execution in all experiments. Thus, its behavior is not shown.

#### *5.4.1.1 Experiment #1: Different instances*

Figure 16 presents the resource usage behavior for each selected instance. Experiments were carried out considering the Azure's A0 instance. However, an error promoted by the lack of memory abruptly interrupts the images processing. For this reason, the A0 results were not shown and considered in performance modeling.

Also in Figure 16, two distinct range in CPU and memory utilization percentage can be observed. The A2 instance ($\mu = 27.643, \sigma^2 = 23.635$) presented the CPU utilization percentage less than A1 instance ($\mu = 53.633, \sigma^2 = 36.750$). Despite of the values in distinct range, CPU utilization presents the same trend behavior in both instances (*vide* Figure 16a). Once more, memory utilization for A2 instance ($\mu = 43.656, \sigma^2 = 4.349$) is lower than the A1 instance ($\mu = 71.953, \sigma^2 = 18.642$) and the same behavior pattern was found, according to Figure 16c. The difference in the value range is also evident in the disk read rate (Figure 16d), while A2 instance did not perform disk reads, A1 instance had the average read rate of $\mu = 492.475$ with great variance $\sigma^2 = 692526.799$ represented by peak from the 37 seconds.

Otherwise, the CPU I/O wait (Figure 16b) and disk write rate (Figure 16e) showed similar behavior patters for the instances. In addition, we observed that the instances showed the similar experiment execution time.

None of instances had the CPU utilization percentage reaching 100%. Further, the

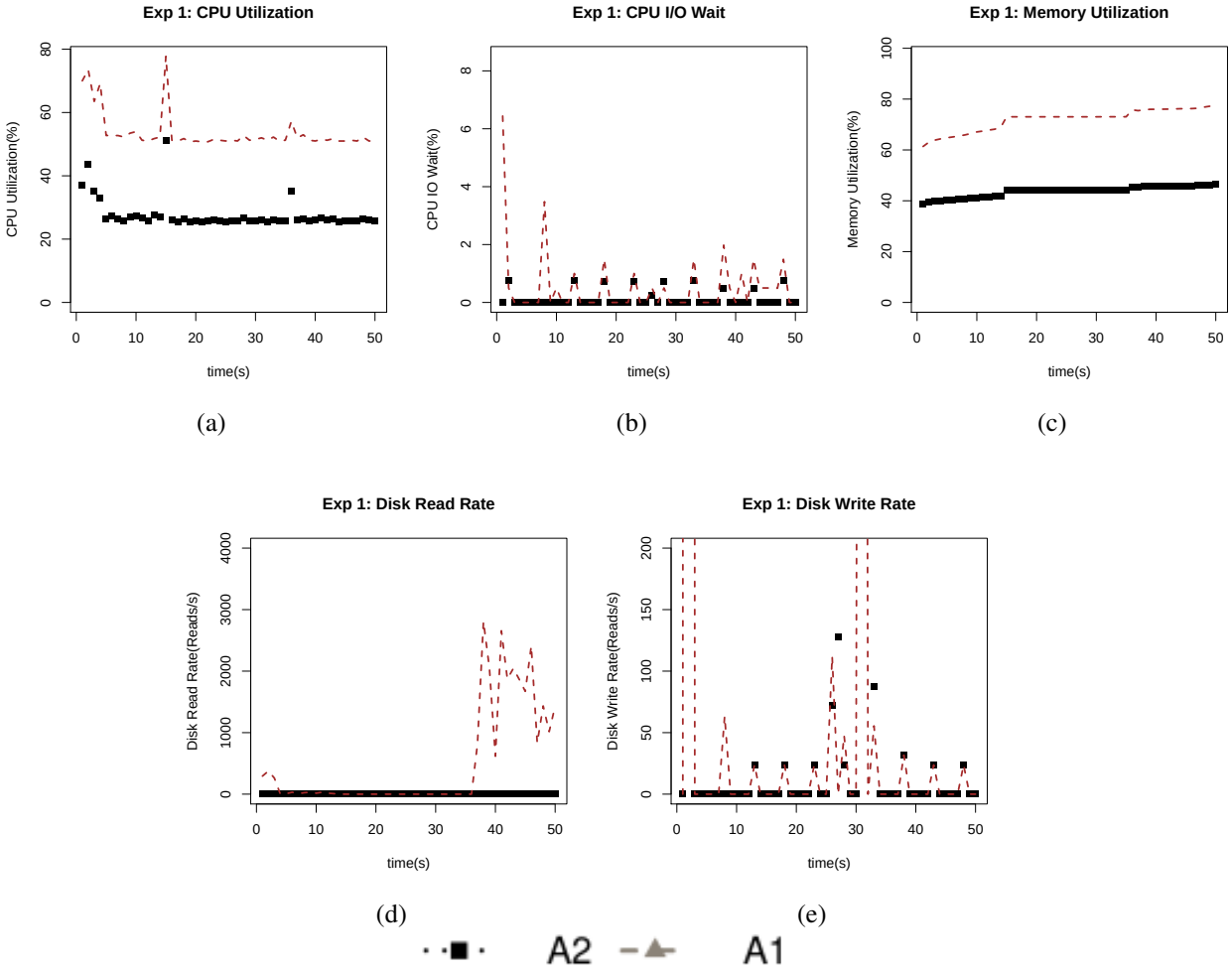



Figure 16 – Impact of LisaMini execution in different instances. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate; (e) Disk Write Rate

time that CPU spent waiting for the completion of an I/O operation had not exceeded 7 %. Consequently, there was no indication of performance problems associated with CPU for those instances. However, the A1 instance presented a significant memory consumption.

#### *5.4.1.2 Experiment #2: Different locations*

Figure 17 shows the resource utilization behavior of A2 instance for each localization. The memory utilization percentage has two distinct ranges of values. The USA ($\mu = 43.656, \sigma^2 = 4.349$) presented a much lower memory consumption presented by Asia ($\mu = 58.30, \sigma^2 = 3.470$) and Europe ($\mu = 58.30, \sigma^2 = 3.470$). Although the experiment involved three different locations, we found only two value ranges. This is because the Asia showed the same pattern of Europe. The other metrics showed the same location behavior pattern.

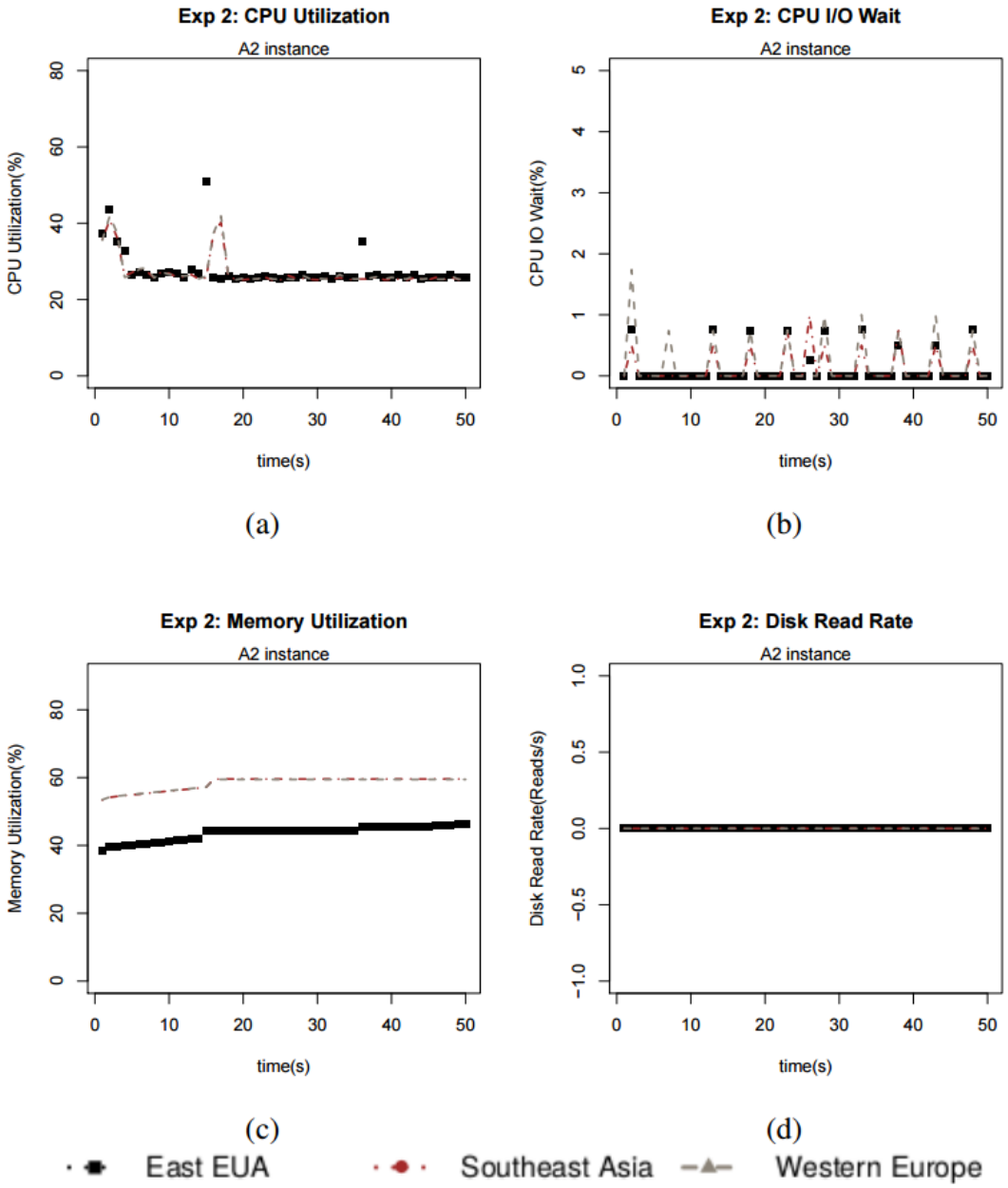



Figure 17 – Impact of LisaMini execution on A2 instance considering different geographic locations. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate

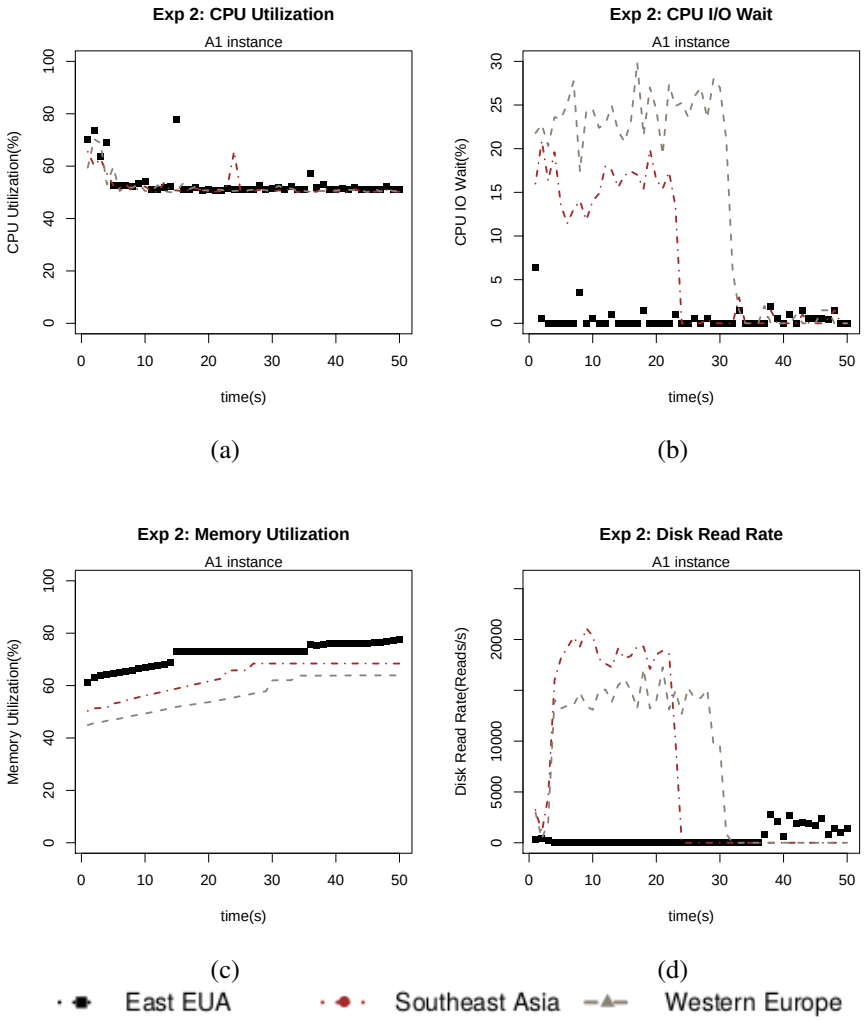



Figure 18 – Impact of LisaMini execution on A1 instance considering different geographic locations. (a) CPU Utilization; (b) CPU I/O Wait; (c) Memory Utilization; (d) Disk Read Rate

Figure 18 shows the resource utilization behavior of A1 instance for each localization. The CPU utilization percentage shows a pattern similar in the USA, Asia and Europe. The other metrics have different value ranges for the different locations. However, again the USA presents distinct behavior while Asia and Europe have similar behavior.

In the USA, the CPU utilization ($\mu = 0.507, \sigma^2 = 1.199$) has a percentage of waiting for I/O operations much lower and with less variation compared with Asia ($\mu = 15.126, \sigma^2 = 135.493$) and Europe ($\mu = 7.476, \sigma^2 = 64.763$). This result is also reflected in the disk write rate, where we have USA ($\mu = 492.475, \sigma^2 = 692526.799$), Europe ($\mu = 7471.7, \sigma^2 = 81402321.732$) and Asia ($\mu = 7717.907, \sigma^2 = 50042849.65$). The memory consumption percentage presented three different ranges: USA ($\mu = 71.953, \sigma^2 = 18.642$), Asia ($\mu = 56.566, \sigma^2 = 44.190$) and Europe ($\mu = 63.080, \sigma^2 = 38.722$).

### *5.4.2 Performance Modeling*

To determine the best candidate to fit the distribution date, we evaluate the fit presented by the 21 continuous distributions and create a subgroup with those with the most promising results. For each selected distribution, we generated 100 random samples and calculate the KS test. From the results, we set the $p-value$ maximum, $p-value$ minimum and rate of the null hypothesis ($H_0$) rejection. The $H_0$ consists of the assumption that the observed data follow the specified distribution. $H_0$ is rejected when the $p-value$ is less than significance level. In this work, we fix the significance level at $\alpha = 0.05$.

We elected the distribution with the lowest rejection rate, according to Tables 18 and 19. When more than one distribution offer the same rejection rate, we observed the Q-Q plots to support the decision. The entire set of distributions obtained from this procedure is presented in Table 16 and Table 17 for experiments 1 and 2, respectively. These tables show the types of transformation performed on the data, the distributions and their parameters that will compose the models.

Previous studies indicate the GEV as a distribution that offers good fit to represent resource utilization, which reinforces the results found in this work. Also, the Weibull distribution with three parameters offers the best fit to different metrics of A1 instance. The frequency with which the distributions as Weibull (3P) and GEV provide the best fit for the selected metrics are justified through the third parameter. Such distributions besides representing location and dispersal measures, can also represent shape, which provides greater precision in the fitting.

There is no distribution in common between the A1 and A2 instances for the same metric. This discrepancy directly reflects the difference of behavior between these instances. The Disk Read Rate of A2 instance is null as is seen in Figure 16 and the Disk Write Rate metric shows the same behavior pattern for both instances.

### *5.4.3 Godness of Fit*

Table 18 shows the higher rejection rates for CPU utilization (4% and 9%) for the Experiment 1. It presents rejection of 1% for the percentage of memory utilization. Regarding to the other metrics, the null hypothesis cannot be rejected for either of the 100 tests. This indicates that the models are consistent with the presented data. Table 19 shows that all rejection rates are under 6% for A1 and A2 instances.

Table 16 – Experiment 1 - Distribution parameters of different instances

| Metric | Transformation | | Distribution | | Estimated Parameters | |
|---|---|---|---|---|---|---|
| | A1 | A2 | A1 | A2 | A1 | A2 |
| United States (East Cost) | | | | | | |
| CPU utilization | inverse | inverse | Weibull (3P) | Weibull | $\hat{\mu} = 6.165$, $\hat{\sigma} = 0.017$, $\hat{\xi} = 0.001$ | $\hat{\sigma} = 0.037$, $\hat{\xi} = 80.894$ |
| CPU I/O Wait | inverse | inverse | Weibull (3P) | GEV | $\hat{\mu} = 0.702$, $\hat{\sigma} = 0.132$, $\hat{\xi} = 4.055$ | $\hat{\mu} = 0.929$, $\hat{\sigma} = 0.019$, $\hat{\xi} = -0.331$ |
| Memory utilization | log10 | inverse | GEV | Logistic | $\hat{\mu} = 1.861$, $\hat{\sigma} = 0.003$, $\hat{\xi} = -0.310$ | $\hat{\mu} = 0.022$, $\hat{\sigma} = 8.336e-05$ |
| Disk Read Rate | inverse | - | Weibull (3P) | - | $\hat{\mu} = 0.362$, $\hat{\sigma} = 0.201$, $\hat{\xi} = 3.040$ | - |
| Disk Write Rate | log10 | log10 | GEV | GEV | $\hat{\mu} = 0.388$, $\hat{\sigma} = 0.119$, $\hat{\xi} = -0.205$ | $\hat{\mu} = 0.388$, $\hat{\sigma} = 0.119$, $\hat{\xi} = -0.205$ |

Table 17 – Experiment 2 - Distribution parameters of different instances in different locations

| Metric | Transformation | | Distribution | | Estimated Parameters | |
|---|---|---|---|---|---|---|
| | Asia | EU | Asia | EU | Asia | EU |
| A2 instance | | | | | | |
| Memory utilization | - | inverse | - | GEV | - | $\hat{\mu} = 7.326e-05$, $\hat{\sigma} = 0.017$, $\hat{\xi} = -0.224$ |
| A1 instance | | | | | | |
| CPU I/O Wait | inverse | sqrt | Normal | Gamma | $\hat{\mu} = 15.126$, $\hat{\sigma} = 135.49$ | $\hat{\sigma} = 0.016$, $\hat{\xi} = 137.380$ |
| Disk Read Rate | sqrt | log2 | Weibull (3P) | Normal | $\hat{\mu} = 36.267$, $\hat{\sigma} = 25.576$, $\hat{\xi} = 3.307$ | $\hat{\mu} = 5.643$, $\hat{\sigma} = 0.891$ |

Table 18 – Kolmogorov-Smirnov test: Experiment 1

| Metric | A1 | | | A2 | | |
|---|---|---|---|---|---|---|
| | p-value maximum | p-value minimum | Ho Reject (%) | p-value maximum | p-value minimum | Ho Reject (%) |
| United States (East Cost) | | | | | | |
| CPU utilization | 1 | 0.014 | 4 | 0.929 | 0.004 | 9 |
| CPU I/O Wait | 0.133 | 0.09 | 0 | 0.99 | 0.07 | 0 |
| Memory utilization | 0.994 | 0.081 | 0 | 0.956 | 0.008 | 1 |
| Disk Read Rate | 0.966 | 0.065 | 0 | - | - | - |
| Disk Write Rate | 1 | 0.106 | 0 | 1 | 0.106 | 0 |

Table 19 – Kolmogorov-Smirnov test: Experiment 2

| Metric | Asia | | | EU | | |
|---|---|---|---|---|---|---|
| | p-value maximum | p-value minimum | Ho Reject (%) | p-value maximum | p-value minimum | Ho Reject (%) |
| A2 instance | | | | | | |
| Memory utilization | - | - | - | 1 | 0.008 | 4 |
| A1 instance | | | | | | |
| CPU I/O Wait | 0.962 | 0.016 | 1 | 0.993 | 0.02 | 2 |
| Disk Read Rate | 0.966 | 0.065 | 0 | 0.964 | 0.019 | 5 |

### *5.4.4 Discussion of Results*

Experiment 1 (see Figure 16) aims at verifying the influence of different instances types in the pattern of resources utilization. The smaller resource capacity is presented by the A1 instance, which the increase of utilization percentage of its resources in relation to A2 instance. Through this experiment, we identified a difference in the behavior pattern of the disk read rate between instances. This disparity is due to A2 instance has twice the memory of A1 instance. Hence, the A2 instance can carry all medical images to be processed in memory. Consequently, there is no disk reads. In other hand, the A1 instance does not have enough memory to load all the images. Then, the operating system transfers as many files to the cache in order to avoid swap memory usage. The remaining files are accessed from disk, which explains the peaks of disk write rate presented by A1 instance at the end of medical image processing.

Both instances show similar behavior regarding to the disk write rate. This similarity is due to a log creation which contains information about the lung segmentation processing for different patients. This log is composed for a header (patient name, exam date, exam serial number), that justified the initial peak of the writing rate, and statistics (Pulmonary density Average, Percent normal regions, medium and hyper aerated) for the left lung, right and both, which promotes other peaks. The size and information presented by the log is the same regardless of location and type of instance. Therefore, all experiments present the same disk write rate pattern.

The CPU I/O wait metric is directly impacted by the disk access. That is why the disk write rate and CPU I/O wait show similar behavior. The difference in amplitude of CPU I/O wait peaks between instances is justified by the fact that disk access is slower than memory access. Since A1 performs disk read, it presents higher amplitudes then A2. Finally, we conclude that A1 and A2 instances have different patterns of CPU utilization, memory utilization, CPU

I/O wait, and disk read which justify specific models for each instance. In addition, the LisaMini is a memory oriented application, which prevents the use of A0 instances.

Experiment 2 (see Figures 17 and 18) verifies the influence of different locations in the resource utilization patterns. Asia and Europe have a similar behavior, even in different range of values. In contrast, the USA shows a very different behavior compared with the other locations. This difference is attributed to data centers heterogeneity and the resources affected in this experiment were memory, disk read and CPU I/O wait, justifying specific models according to location.

In the A1 instance presents a discrepancy in the disk reads behavior pattern, which impacts on CPU I/O wait behavior. Unlike CPU and memory, the disk is not dedicated to a particular instance. It is shared between instances of a data center. Therefore, the performance is influenced by the shared resource availability. Since the experiments were performed in 3 different time zones, the number of users can vary greatly depending on the time and location, influencing the availability of resources and, consequently in their performances. Furthermore, Azure use more than one type of disc.

## 5.5 Concluding Remarks

In the experiments that aim at verifying if the user profile impacts on the performance models, it was observed that the change from browsing to bidding profile implies in a memory consumption increment. However, the disk consumption is much lower compared to memory consumption because, in general, e-commerce applications are in-memory, *i.e.*, the information is transferred from disk to memory (cache) to avoid slow Web response times.

In the experiments that aim at verifying if the instances types impacts on the performance models, it was observed that only *log10* and *inverse* appears between best fits. Also, the General Extreme Value (GEV) distribution best fits CPU I/O wait (A2 instance), Memory and Disk Write Rate (A1 instance). The other resources are covered by the Weibull and Logistic distributions.

In the experiments that aim at verifying if the geographic location impacts on the performance models, it was observed that Asia and Europe have a similar behavior, even in different range of values. In contrast, the USA shows a very different behavior compared with the other locations. This difference is attributed to data centers heterogeneity and the resources affected in this experiment were memory, disk read and CPU I/O wait, justifying specific models

according to location. Since the experiments were performed in 3 different time zones, the number of users can vary greatly depending on the time and location, influencing the availability of resources and, consequently in their performances.

From the validation of model simulation experiments, we can conclude the implementation of web application modeling in the CloudSim simulator is capable of producing data to accurately represent both user profiles. For the total of 10,000 comparisons, the error was also calculated. The error is computed when sampled simulated data does not belong to the same sample data observed, i.e., $p - value < 0.05$. The error for all metrics is less than 10%. Among the mathematical transformations applied to the CPU utilization metric, the inverse transformation offers the best results, reducing the Wilcox error rate of 34% to 6%, considering the Browsing profile.

# 6 CONCLUSION

Clouds are facing complex workloads composed of a wide variety of applications, where the behavior is defined by users with different profiles and QoS requirements. Such workloads, along with insufficient trace logs available for analysis of resource utilization are the main causes for the lack of methodologies and models that characterize the applications hosted in the cloud. In order to fill this gap, in this thesis, a web application model that fits the behavioral patterns of different user profiles is proposed. This model supports analysis and simulation of CPU utilization in cloud environments.

Here, a well-defined methodology is adopted to support implementation and validation of simulation models as presented in this work. The proposed model was implemented as an extension of the CloudSim simulator, and the validation was conducted through graphic and statistical hypothesis methods, which showed that the Web application modeling in CloudSim generates data that fit the different user profiles accurately. This implementation allowed a case study that supports CPU utilization forecasting associated with DVFS management in order to improve energy savings. This is critical to cloud providers that need to quickly grant resources to meet the application demand in an energy efficient manner. Such a model can also be utilized to improve other research domains, *e.g.*, such as resource optimization and failure-analysis.

Based on our model and experiments, we can highlight the following:

- Different results from assumptions commonly made (LI, 2010) were achieved, such as (i) stationarity instead of fixed arrival times to represent instructions arrival in processor (Section 5.1.1), (ii) Generalized Pareto distribution instead of simple Poisson models to represent instruction arrivals, and (iii) Generalized Extreme Value and Generalized Lambda distributions instead of Exponential distribution to represent session time (Section 5.1.2);
- MPC showed to be an accurate technique with respect to Web application modeling and control parameterization proposed in this work . It can provide a straightforward support to SLA-aware resource provisioning policies (Section 5.3).

## 6.1 Research Questions Analysis

The research questions presented in section 1.3 are discussed as follows:

**Research Question 1:** How do different user profiles impact on resource utilization

behavior?

Monitoring experiments were designed to this end (see Subsection 4.1.4) and results are shown in the Subsection 5.1.1. The type of user profile (i.e., user behavior model) imposed a strong influence on the CPU utilization so that different statistical distributions were found to represent the total number of instructions. Therefore, user behavior must be considered in workload modeling to reflect realistic conditions.

The GEV distribution with parameters estimated using the PWM method presents the best fit to browsing profile. In contrast, the GL distribution with parameters estimated using the mps method presents the best fit for the bidding profile. There is no distribution in common between the user profiles for the same metric. This discrepancy directly reflects the difference in behavior between these instances.

**Research Question 2:** Do different VM instances and geographic locations affect the resources usage patterns sufficient to justify specific models for each of them?

Monitoring experiment was designed to this end (see Subsection 4.2.3.1) and results are shown in the Subsection 5.4.1. Through this experiment, we conclude that A1 and A2 instances have different patterns of CPU utilization, memory utilization, CPU I/O wait, and disk read which justify specific models for each instance.

From Table 16, we note the General Extreme Value (GEV) distribution best fits CPU I/O wait (A2 instance), Memory and Disk Write Rate (A1 instance). The other resources are covered by the Weibull and Logistic distributions. There is no distribution in common between the A1 and A2 instances for the same metric. This discrepancy directly reflects the difference in behavior between these instances.

Monitoring experiment was designed to accomplish this goal (see Subsection 4.2.3.1) and results are shown in the Subsection 5.4.1. Through this experiment, we conclude that the geographical location impacts on the resource utilization patterns. Asia and Europe have a similar behavior, even in the different range of values. In contrast, USA shows a very different behavior compared to the other locations. This difference is due to data centers heterogeneity and the resources affected in this experiment were the memory, disk read and CPU I/O wait. Then, we need specific models according to the geographical location.

The A1 instance presents a discrepancy in the disk reads behavior pattern, which impacts on CPU I/O wait for behavior. Unlike CPU and memory, the disk is not dedicated to a particular instance. Instead, it is shared between instances of a data center. Therefore, the

workload performance (response time, throughput) can change according to the shared resource availability. Since the experiments were performed in 3 different time zones, the number of users can vary greatly depending on the time and location, influencing the availability of resources and, consequently their performance.

**Research Question 3:** How to develop predictive models of resources to guide the cloud elasticity so that the workload variation has the minimum impact on performance and availability?

In this thesis, a model predictive control capable of capturing the complex relationship between the resource utilization profile of the application and its user performance metric (response time and execution time) was used to support QoS-aware resource allocation policies.

The monitoring of computational resources, such as CPU, memory, and disk, becomes essential for both providers and clients because such metrics are often associated to elasticity (COUTINHO, 2014). Therefore, the relationship between CPU, memory and disk utilization with the application response time are used as MPC's input. The MPC takes the corrective action, which means adjusting the inputs to keep the predicted output (response time, execution time) close to the reference trajectory.

To this end, the predicted output is compared to the reference trajectory and the difference between them, called future errors, is applied as feedback to the optimizer to minimize them. The controller is periodically updated with real-time measures, allowing the model to adapt to workload conditions. To define the best current control action, MPC reduces the errors over time by adjustment of the input variables (see Section 2.3.4). Thus, the adjusted input could be used in a straightforward manner to support resource allocation policies.

**Research Question 4:** How to develop models that are able to predict application performance considering multiple parameters such as processor, memory, network and disk usage?

Monitoring experiments were designed and implemented (see Sections 4.1 and 4.2) in order to capture the performance behavior (CPU, memory, disk) of two different applications in private and public cloud: RUBiS and LisaMini. Since the behavior was recorded, a well-defined methodology, described in the Section 3.1, was followed to obtain the performance models in the form of statistical distributions, where patterns fluctuate based on realistic parameters in order to represent dynamic environments (see Subsections 5.1.2 and 5.4.2).

Such performance models fulfill the input role in the MPC that uses the state space

model, described in Section 2.3.4.1, to compute the predicted behavior over some horizon. The state space models are a linear technique capable of representing the dynamics of a $n^{th}$ order system as a first differential equation represented by a $N$ size vector. Thereby, the dependence of the predictions on future control choices is linear, and this facilitates optimization as well as off-line analysis of expected closed-loop behavior.

## 6.2 Publications

The list of publications published so far from work in this thesis includes journal and conference papers as follows:

- **MAGALHÃES, D. M. V.**; CALHEIROS, R. N.; BUYYA, R.; GOMES, D. G. "Workload modeling for resource usage analysis and simulation in cloud computing". Computers & Electrical Engineering, v. 47, p. 69–81, 2015. ISSN 0045-7906;
- SILVA JUNIOR, L. S. ; **MAGALHÃES, D. M. V.**; Gomes, Danielo G. "Modelagem e Simulação de Offloading para Computação Móvel em Nuvem". In: 7o. SBCUP - Simpósio Brasileiro de Computação Ubíqua e Pervasiva (CSBC 2015 - SBCUP), 2015, Recife-PE. p. 91-100 (in Portuguese);
- **MAGALHÃES, D. M. V.** ; SOARES, J. M. ; Gomes, Danielo G. "Alocação dinâmica de máquinas virtuais para economia de energia". Novas Edições Acadêmicas - NEA, 2014. 132 p, ISBN: 978-3-639-61919-5 (in Portuguese).

## 6.3 Future Work

Following up the work developed during this Ph.D. thesis, here is a list of potential perspectives to be improved:

1. The cloud environments hosts multiple purposes applications, from running batch jobs to web applications, which have heterogeneous and competing QoS requirements (CALHEIROS *et al.*, 2011). Vondra and Sedivy (VONDRA; SEDIVY, 2013) indicate an efficient way to maximize the resource utilization of cloud computing infrastructure through the mix of interactive and non-interactive workloads. Address a realistic and efficient scenario is desired to adopt a heterogeneous mix composed of a non-interactive (scientific application) and an interactive (web application) workloads;
2. Resource Allocation Policies: since this work achieves the performance models for two

different applications. In addition, a model predictive control was used to establish the relation between computational resource metrics and user performance metrics, to capture the workload variation and guide resource allocation ensuring the cloud environment elasticity. A straightforward extension of our work is the development of a resource allocation policy to meet the demand of cloud applications while maintaining the user experience desired;

3. Transfer function: in the literature different methods are used to generate the transfer function, such as ARMA(X), Kalma filter, Smoothing splines. According to Lorido (LORIDO-BOTRAN *et al.*, 2014), Fuzzy models have been used to establish the relationship between the input and output variables. It is relevant to compare the performance of the FFT method used in this work with different techniques;
4. Correlation between resources: Lorido (LORIDO-BOTRAN *et al.*, 2014) shows that most of the methods of prediction focus on more than one workload metric. However, they ignore the correlation between them. It is relevant to investigate the correlation between the resource utilization metrics to provide more understandable results for resource management.

# BIBLIOGRAPHY

AMIRI, M.; MOHAMMAD-KHANLI, L. Survey on prediction models of applications for resources provisioning in cloud. **Journal of Network and Computer Applications**, v. 82, p. 93–113, 2017. ISSN 1084-8045. Disponível em: <//www.sciencedirect.com/science/article/pii/S1084804517300231>.

ANGLANO, C.; CANONICO, M.; GUAZZONE, M. Fcms: A fuzzy controller for cpu and memory consolidation under sla constraints. **Concurrency and Computation: Practice and Experience**, Wiley Online Library, 2016.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. **Commun. ACM**, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/1721654.1721672>.

BODIK, P.; SUTTON, C.; FOX, A.; PATTERSON, D.; JORDAN, M. Response-time modeling for resource allocation and energy-informed slas. In: **Proc. Workshop on Statistical Learning Techniques for Solving Systems Problems (MLSys' 07), Whistler, BC, Canada**. [S.l.: s.n.], 2007.

BOLCH, G.; GREINER, S.; MEER, H. de; TRIVEDI, K. S. **Queueing networks and Markov chains: modeling and performance evaluation with computer science applications**. [S.l.]: John Wiley & Sons, 2006.

BRAGA, R. A.; GOMES, D. G.; SOARES, J. M. **ElasticCluster: Explorando a Ociosidade de Clusters Virtualizados**. [S.l.]: Novas Edições Acadêmicas - NEA, 2014.

BUYYA, R.; RANJAN, R.; CALHEIROS, R. N. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In: IEEE. **High Performance Computing & Simulation, 2009. HPCS'09. International Conference on**. [S.l.], 2009. p. 1–11.

CALHEIROS, R. N.; RANJAN, R.; BELOGLAZOV, A.; ROSE, C. A. F. D.; BUYYA, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. **Software: Practice and Experience**, John Wiley & Sons, Ltd., v. 41, n. 1, p. 23–50, 2011.

CAMACHO, E.; ALBA, C. **Model Predictive Control**. 2. ed. Springer London, 2013. (Advanced Textbooks in Control and Signal Processing). ISBN 9780857293985. Disponível em: <https://books.google.com.br/books?id=tXZDAAAAQBAJ>.

CECCHET, E.; CHANDA, A.; ELNIKETY, S.; MARGUERITE, J.; ZWAENEPOEL, W. Performance comparison of middleware architectures for generating dynamic web content. In: **Proceedings of the International Conference on Middleware**. [S.l.]: Springer-Verlag, 2003. p. 242–261.

CHALABI, Y.; SCOTT, D. J.; WUERTZ, D. **An asymmetry-steepness parameterization of the generalized lambda distribution**. [S.l.], 2012. Disponível em: <http://EconPapers.repec.org/RePEc:pra:mprapa:37814>.

CHALABI, Y.; WÜRTZ, D.; TROYER, M. **New Directions in Statistical Distributions, Parametric Modeling and Portfolio Selection**. [S.l.]: ETH, 2012.

CHEN, Y.; GANAPATHI, A. S.; GRIFFITH, R.; KATZ, R. H. **Towards understanding cloud performance tradeoffs using statistical workload analysis and replay**. [S.l.], 2010. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-81.html>.

CHENG, R. C. H.; AMIN, N. A. K. Estimating parameters in continuous univariate distributions with a shifted origin. **Journal of the Royal Statistical Society**, v. 45, n. 3, p. 394–403, 1983.

COLES, S. **An introduction to statistical modeling of extreme values**. [S.l.]: Springer-Verlag, 2001.

COUTINHO, E. F. **FOLE: Um framework conceitual para avaliação de desempenho da elasticidade em ambientes de computação em nuvem**. Tese (Doutorado) — Universidade Federal do Ceará, Fortaleza, CE, Brasil, 2014.

COUTINHO, E. F.; GOMES, D. G.; SOUZA, J. N. d. An analysis of elasticity in cloud computing environments based on allocation time and resources. **IEEE Latin America Conference on Cloud Computing and Communications (LatinCloud)**, IEEE, p. 7–12, 2013.

COUTINHO, E. F.; SOUSA, F. R. de C.; REGO, P. A. L.; GOMES, D. G.; SOUZA, J. N. de. Elasticity in cloud computing: a survey. **Annals of telecommunications - annales des télécommunications**, v. 70, n. 7, p. 289–309, 2015. ISSN 1958-9395. Disponível em: <http://dx.doi.org/10.1007/s12243-014-0450-7>.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. **Communications of the ACM**, ACM, v. 51, n. 1, p. 107–113, 2008.

DENG, Y.; MENG, X.; ZHOU, J. Self-similarity: Behind workload reshaping and prediction. **Future Generation Computer Systems**, Elsevier, v. 28, n. 2, p. 350–357, 2012.

DUONG, T. N. B.; LI, X.; GOH, R.; TANG, X.; CAI, W. QoS-aware revenue-cost optimization for latency-sensitive services in IaaS clouds. In: **Proceedings of the 16th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)**. [S.l.: s.n.], 2012. p. 11–18.

EMBRECHTS, P.; KLÜPPELBERG, C.; MIKOSCH, T. **Modelling Extremal Events: For Insurance and Finance**. [S.l.]: Springer, 1997.

FARAHABADY, M. R. H.; LEE, Y. C.; ZOMAYA, A. Y.; TARI, Z.; SONG, A. A model predictive controller for contention-aware resource allocation in virtualized data centers. In: **IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)**. [S.l.: s.n.], 2016. p. 277–282.

FARAHABADY, M. R. H.; SAMANI, H. R. D.; WANG, Y.; ZOMAYA, A. Y.; TARI, Z. A qos-aware controller for apache storm. In: **2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)**. [S.l.: s.n.], 2016. p. 334–342.

FEITELSON, D. G. **Workload Modeling for Computer Systems Performance Evaluation**. [S.l.]: Cambridge University Press, 2014. In press, available at http://www.cs.huji.ac.il/ feit/wlmod/wlmod.pdf.

FOSTER, I.; ZHAO, Y.; RAICU, I.; LU, S. Cloud computing and grid computing 360-degree compared. In: **Proceedings of the Grid Computing Environments Workshop**. [S.l.: s.n.], 2008. p. 1–10.

FREEDMAN, D.; DIACONIS, P. On the histogram as a density estimator:l2 theory. **Probability Theory and Related Fields**, v. 57, n. 4, p. 453–476, 1981.

GANAPATHI, A.; CHEN, Y.; FOX, A.; KATZ, R.; PATTERSON, D. Statistics-driven workload modeling for the cloud. In: **Proceedings of International Conference on the Data Engineering Workshops**. [S.l.]: IEEE, 2010. p. 87–92.

GHASEMI-FALAVARJANI, S.; NEMATBAKHSH, M.; GHAHFAROKHI, B. S. Context-aware multi-objective resource allocation in mobile cloud. **Computers & Electrical Engineering**, Elsevier, v. 44, p. 218–240, 2015.

GOLD, A. Understanding the mann-whitney test. **Journal of Property Tax Assessment and Administration**, UNIVERSITY OF ULSTER, v. 4, n. 3, p. 55, 2007.

GREENWOOD, J. A.; LANDWEHR, J. M.; MATALAS, N. C.; WALLIS, J. R. Probability weighted moments: definition and relation to parameters of several distributions expressable in inverse form. **Water Resources Research**, v. 15, n. 5, p. 1049–1054, 1979.

GROZEV, N.; BUYYA, R. Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments. **The Computer Journal**, p. , 2013.

HASHEMIAN, R.; KRISHNAMURTHY, D.; ARLITT, M. Web workload generation challenges – an empirical investigation. **Software: Practice and Experience**, John Wiley & Sons, Ltd, v. 42, n. 5, p. 629–647, 2012.

HASTINGS, C.; MOSTELLER, F.; TUKEY, J.; WINSOR, C. Low moments for small samples: a comparative study of statistics. **Annals of Mathematical Statistics**, v. 18, n. 3, p. 413–426, 1947.

HERBST, N. R.; KOUNEV, S.; REUSSNER, R. H. Elasticity in cloud computing: What it is, and what it is not. In: **ICAC**. [S.l.: s.n.], 2013. p. 23–27.

HOSKING, J. R. M.; WALLIS, J. F. Parameter and quantile estimation for the generalized pareto distribution. **Technometrics**, American Society for Quality Control and American Statistical Association, v. 29, n. 3, p. 339–349, 1987.

HOSKING, J. R. M.; WALLIS, J. R.; WOOD, E. F. Estimation of the generalized extreme-value distribution by the method of probability-weighted moments. **Technometrics**, American Statistical Association, v. 27, n. 3, p. 251–261, 1985.

IQBAL, W.; DAILEY, M. N.; CARRERA, D. Sla-driven dynamic resource management for multi-tier web applications in a cloud. In: **Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing**. Washington, DC, USA: IEEE Computer Society, 2010. (CCGRID '10), p. 832–837. ISBN 978-0-7695-4039-9. Disponível em: <http://dx.doi.org/10.1109/CCGRID.2010.59>.

JENKINSON, A. F. The frequency distribution of the annual maximum (or minimum) values of meteorological elements. **Quarterly Journal of the Royal Meteorological Society**, v. 87, p. 158–171, 1955.

KAVULYA, S.; TAN, J.; GANDHI, R.; NARASIMHAN, P. An analysis of traces from a production MapReduce cluster. In: **Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)**. [S.l.: s.n.], 2010. p. 94–103.

KHAN, A.; YAN, X.; TAO, S.; ANEROUSIS, N. Workload characterization and prediction in the cloud: A multiple time series approach. In: **Proceedings of the IEEE Network Operations and Management Symposium (NOMS)**. [S.l.: s.n.], 2012. p. 1287–1294.

KUNDU, S.; RANGASWAMI, R.; DUTTA, K.; ZHAO, M. Application performance modeling in a virtualized environment. In: **Proceedings of the 16th International Symposium on High Performance Computer Architecture (HPCA)**. [S.l.]: IEEE, 2010. p. 1–10.

KWIATKOWSKI, D.; PHILLIPS, P. C.; SCHMIDT, P.; SHIN, Y. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? **Journal of Econometrics**, v. 54, p. 159–178, 1992.

LI, H. Realistic workload modeling and its performance impacts in large-scale escience grids. **IEEE Transactions on Parallel and Distributed Systems**, v. 21, n. 4, p. 480–493, 2010.

LLOYD, W.; PALLICKARA, S.; DAVID, O.; LYON, J.; ARABI, M.; ROJAS, K. Performance implications of multi-tier application deployments on infrastructure-as-a-service clouds: Towards performance modeling. **Future Generation Computer Systems**, v. 29, n. 5, p. 1254–1264, 2013.

LORIDO-BOTRAN, T.; MIGUEL-ALONSO, J.; LOZANO, J. A review of auto-scaling techniques for elastic applications in cloud environments. **Journal of Grid Computing**, Springer Netherlands, v. 12, n. 4, p. 559–592, 2014. ISSN 1570-7873. Disponível em: <http://dx.doi.org/10.1007/s10723-014-9314-7>.

LUK, C.-K.; COHN, R.; MUTH, R.; PATIL, H.; KLAUSER, A.; LOWNEY, G.; WALLACE, S.; REDDI, V. J.; HAZELWOOD, K. Pin: Building customized program analysis tools with dynamic instrumentation. In: **Proceedings of the Conference on Programming Language Design and Implementation**. [S.l.]: ACM, 2005. p. 190–200.

MADNI, S. H. H.; LATIFF, M. S. A.; COULIBALY, Y.; ABDULHAMID, S. M. Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. **Cluster Computing**, p. 1–45, 2016. ISSN 1573-7543. Disponível em: <http://dx.doi.org/10.1007/s10586-016-0684-4>.

MAGALHAES, D.; CALHEIROS, R. N.; BUYYA, R.; GOMES, D. G. Workload modeling for resource usage analysis and simulation in cloud computing. **Computers & Electrical Engineering**, v. 47, p. 69–81, 2015. ISSN 0045-7906. Disponível em: <http://www.sciencedirect.com/science/article/pii/S004579061500302X>.

MAGALHÃES, D.; SOARES, J. M.; GOMES, D. G. **Alocação dinâmica de máquinas virtuais para economia de energia**. [S.l.]: Novas Edições Acadêmicas - NEA, 2014.

MANVI, S. S.; SHYAM, G. K. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. **Journal of Network and Computer Applications**, v. 41, p. 424–440, 2014. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804513002099>.

MAO, M.; HUMPHREY, M. A performance study on the vm startup time in the cloud. In: **Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing**. IEEE Computer Society, 2012. (CLOUD '12), p. 423–430. ISBN 978-0-7695-4755-8. Disponível em: <http://dx.doi.org/10.1109/CLOUD.2012.103>.

MARTINS, E. S.; STEDINGER, J. R. Generalized maximum-likelihood generalized extreme-value quantile estimators for hydrologic data. **Water Resources Research**, v. 36, n. 3, p. 737–744, 2000.

MELL, P.; GRANCE, T. The nist definition of cloud computing. **National Institute of Standards and Technology**, v. 53, n. 6, p. 50, 2009.

MENASCE, D. TPC-W: a benchmark for e-commerce. **Internet Computing**, IEEE, v. 6, n. 3, p. 83–87, 2002.

MISHRA, A. K.; HELLERSTEIN, J. L.; CIRNE, W.; DAS, C. R. Towards characterizing cloud backend workloads: Insights from Google compute clusters. **SIGMETRICS Performance Evaluation Review**, ACM, v. 37, n. 4, p. 34–41, 2010.

MORENO, I.; GARRAGHAN, P.; TOWNEND, P.; XU, J. An approach for characterizing workloads in Google cloud to derive realistic resource utilization models. In: **Proceedings of 7th International Symposium on Service Oriented System Engineering (SOSE)**. [S.l.]: IEEE, 2013. p. 49–60.

MULIA, W. D.; SEHGAL, N.; SOHONI, S.; ACKEN, J. M.; STANBERRY, C. L.; FRITZ, D. J. Cloud workload characterization. **IETE Technical Review**, v. 30, n. 5, p. 382–397, 2013.

PADALA, P.; HOU, K.-Y.; SHIN, K. G.; ZHU, X.; UYSAL, M.; WANG, Z.; SINGHAL, S.; MERCHANT, A. Automated control of multiple virtualized resources. In: **Proceedings of the 4th ACM European Conference on Computer Systems**. ACM, 2009. (EuroSys '09), p. 13–26. ISBN 978-1-60558-482-9. Disponível em: <http://doi.acm.org/10.1145/1519065.1519068>.

PATIKIRIKORALA, T.; COLMAN, A. Feedback controllers in the cloud. In: **Proceedings of APSEC**. [S.l.: s.n.], 2010.

PAUL, D.; ZHONG, W.-D.; BOSE, S. K. Energy efficiency aware load distribution and electricity cost volatility control for cloud service providers. **Journal of Network and Computer Applications**, v. 59, p. 185 – 197, 2016. ISSN 1084-8045.

PAXSON, V. Bro: a system for detecting network intruders in real-time. **Computer Networks: The International Journal of Computer and Telecommunications**, Elsevier, v. 31, n. 23–24, p. 2435–2463, 1999.

QU, C.; CALHEIROS, R. N.; BUYYA, R. Auto-scaling web applications in clouds: A taxonomy and survey. **CoRR**, abs/1609.09224, 2016. Disponível em: <http://arxiv.org/abs/1609.09224>.

R Core Team. **R: A Language and Environment for Statistical Computing**. Vienna, Austria, 2013. Disponível em: <http://www.R-project.org>.

RAMBERG, J. S.; SCHMEISER, B. W. An approximate method for generating asymmetric random variables. **Communications of the ACM**, ACM, v. 17, n. 2, p. 78–82, 1974.

REISS, C.; WILKES, J.; HELLERSTEIN, J. L. **Google cluster-usage traces: format + schema**. [S.l.], 2011. Disponível em: <http://code.google.com/p/googleclusterdata/wiki/TraceVersion2>.

RIZVANDI, N. B.; TAHERI, J.; MORAVEJI, R.; ZOMAYA, A. Y. On modelling and prediction of total cpu usage for applications in mapreduce environments. In: **Proceedings of the 12th International Conference on Algorithms and Architectures for Parallel Processing**. [S.l.]: Springer-Verlag, 2012. p. 414–427.

ROSSI, F. D. *et al.*

**Performance-Aware Energy-Efficient Cloud Orchestration** — Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, RS, Brazil, 2016.

SAID, S. E.; DICKEY, D. A. Testing for unit roots in autoregressive-moving average models of unknown order. **Biometrika**, v. 71, n. 3, p. 599–607, 1984.

SCHAD, J.; DITTRICH, J.; QUIANÉ-RUIZ, J.-A. Runtime measurements in the cloud: Observing, analyzing, and reducing variance. **Proc. VLDB Endow.**, VLDB Endowment, v. 3, n. 1-2, p. 460–471, set. 2010. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/1920841.1920902>.

SEKHON, J. S. Multivariate and propensity score matching software with automated balance optimization: The matching package for r. **Journal of Statistical Software**, v. 42, n. 7, p. 1–52, 2011.

SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). **Biometrika**, JSTOR, p. 591–611, 1965.

SONG, M.; ZHU, H.; FANG, Q.; WANG, J. Thermal-aware load balancing in a server rack. In: **2016 IEEE Conference on Control Applications (CCA)**. [S.l.: s.n.], 2016. p. 462–467.

STEPHENS, M. A. Edf statistics for goodness of fit and some comparisons. **Journal of the American Statistical Association**, v. 69, n. 347, p. 730–737, 1974.

SYSSTAT. 2013. Disponível em: <http://sebastien.godard.pagesperso-orange.fr/>.

TCHANA, A.; DILLENSEGER, B.; PALMA, N. D.; ETCHEVERS, X.; VINCENT, J.; SALMI, N.; HARBAOUI, A. A self-scalable and auto-regulated request injection benchmarking tool for automatic saturation detection. **IEEE Transactions on Cloud Computing**, p. , 2014.

TCPDUMP/LIBPCAP. 2013. Disponível em: <http://www.tcpdump.org/>.

VONDRA, T.; SEDIVY, J. Maximizing utilization in private iaas clouds with heterogenous load through time series forecasting. **International Journal on Advances in Systems and Measurements**, v. 6, n. 1-2, p. 149–165, 2013.

WANG, G.; BUTT, A. R.; MONTI, H.; GUPTA, K. Towards synthesizing realistic workload traces for studying the hadoop ecosystem. In: **Proceedings of the 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems**. [S.l.]: IEEE, 2011. p. 400–408.

WANG, X.; WANG, Y. Coordinating power control and performance management for virtualized server clusters. **IEEE Transactions on Parallel and Distributed Systems**, v. 22, n. 2, p. 245–259, 2011. ISSN 1045-9219.

WEINGARTNER, R.; BRäSCHER, G. B.; WESTPHALL, C. B. Cloud resource management: A survey on forecasting and profiling models. **Journal of Network and Computer Applications**, v. 47, p. 99–106, 2015. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804514002252>.

WOOD, T.; CHERKASOVA, L.; OZONAT, K.; SHENOY, P. Profiling and modeling resource usage of virtualized applications. In: **Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware**. [S.l.]: Springer-Verlag, 2008. p. 366–387.

YANG, H.; LUAN, Z.; LI, W.; QIAN, D. Mapreduce workload modeling with statistical approach. **Journal of Grid Computing**, Springer, v. 10, n. 2, p. 279–310, 2012.

YOUSEFF, L.; BUTRICO, M.; SILVA, D. D. Toward a unified ontology of cloud computing. In: IEEE. **Grid Computing Environments Workshop, 2008. GCE'08**. [S.l.], 2008. p. 1–10.

ZERIGO. **Apache: multi-threaded vs multi-process (pre-forked)**. 2014. Disponível em: <http://www.zerigo.com/article/apache_multi-threaded_vs_multi-process_pre-forked>.

ZHANG, Q.; HELLERSTEIN, J.; BOUTABA, R. Characterizing task usage shapes in Google compute clusters. In: **Proceedings of the 5th International Workshop on Large Scale Distributed Systems and Middleware**. [S.l.: s.n.], 2011. p. 2–3.

ZHU, Q.; AGRAWAL, G. Resource provisioning with budget constraints for adaptive applications in cloud environments. In: **Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing**. [s.n.], 2010. (HPDC '10), p. 304–307. ISBN 978-1-60558-942-8. Disponível em: <http://doi.acm.org/10.1145/1851476.1851516>.

ZHU, X.; UYSAL, M.; WANG, Z.; SINGHAL, S.; MERCHANT, A.; PADALA, P.; SHIN, K. What does control theory bring to systems research? **ACM SIGOPS Operating Systems Review**, ACM, v. 43, n. 1, p. 62–69, 2009.

# GLOSSARY

**A**

**Accuracy:** the degree to which the estimated/simulated/predicted value conforms to the measured value.

**B**

**Benchmark:** To compare different systems, they must be evaluated under equivalent conditions, and, in particular, with the same workload. Therefore, a set of workloads are ported to different systems and used as the basis for comparison. Such standardized workloads are called benchmarks.

**P**

**Pattern:** a particular way in which a set of properties or characteristics (resource usage: CPU, memory, disk utilization) show themselves regularly repeted over time for a specific scenario.
**Process:** sequence of interdependent and linked procedures which, at every stage, consume one or more resources (time, computational resources) to convert inputs (logs) into outputs (performance patterns). These outputs then serve as inputs for the next stage until a known goal or end result is reached (model).

**R**

**Resource Management:** IaaS resource management is the complex task of providing the right amount of resources to achieve the required Service Level Agreement (SLA), while keeping the overall cost low.

**W**

**Workload:** Amount of work or number of work units assigned to a particular resource over a given period. In this thesis, consider a workload that is composed of jobs or user requests that use computational resources, such as, CPU, disk, memomy.
**Workload Modeling:** Workload modeling is the attempt to create a simple and general model, which can then be used to generate synthetic workloads for reliable performance evaluations of computer systems. Workload modeling always starts with measured and recorded data about the workload-related events that happened in a certain system. But, it is not limited to just observing and recording, the essence of modeling is abstraction (generalization and

simplification). Considering a log with data about tens of thousands of jobs may contain hundreds of thousands of numbers. It is ludicrous to claim that the exact values of all these numbers are important. What is important are the underlying patterns. It is these patterns that are represented by model, typically in the form of statistical distributions.