

Assessing and Forecasting Energy and Ecological Efficiency on Cloud Computing Platforms



Josep Subirats, *Supervisor:* Jordi Guitart

Department of Computer Architecture

Universitat Politècnica de Catalunya

A thesis submitted for the degree of
Master on Computer Architecture, Networks and Systems

September 2013

I would like to dedicate this thesis to my parents, Marina and Daniel.

Acknowledgements

The work presented in this thesis can be understood as a big puzzle which has been finished thanks to the pieces provided by many people. First of all, I would like to express my most sincere gratitude to my Master Thesis advisor Dr. Jordi Guitart for his guidance, neverending support and wisdom during this last year. The present thesis would not be in your hands without the numerous socratic discussions we had and his constant help. I would also like to thank my boss, Dr. Jordi Torres, for having given me the opportunity to work in the Barcelona Supercomputing Centre.

I have met lots of people in this centre who have also contributed with their puzzle piece. Many thanks to Dr. Íñigo Goiri, my initial mentor in this Cloud Computing adventure and author of EMOTIVE. Thanks to Dr. Enric Tejedor for having created such a fantastic programming model as COMPSs, and to Francesc Lordan, for his valuable help and instruction on how to use it to generate a batch workload. Thank you, Oriol Fitó and Jorge Ejarque, for your continuous and selfless help in debugging the Ecoefficiency Tool. The list of remaining work mates and people at BSC who have contributed on having a fantastic working atmosphere and that also provided me with valuable insights is almost infinite: Mario Macías, Gemma Reig, Raül Sirvent, Roger Rafanell, Juan Luís Pérez, Cesare Cugnasco, Álvaro Villalba, Sergio Mendoza, David Carrera, Yolanda Becerra... Thank you all and to all those people who, despite not appearing in the previous list, have contributed in one way or another in the realisation of this work.

During my involvement in the OPTIMIS project [1] I have also met several people who I would like to thank because of their help and indirect involvement in this work. They are: Django Armstrong, Tomas Forsman, Pierre Gilet, Enric Pages and Juan Luis Prieto.

Last but not least, I would also like to thank my family and friends for their continuous support during this last year.

This work has been supported by the OPTIMIS project and has been partly funded by the European Commissions IST activity of the 7th

Framework Program under contract number 257115. This work expresses the opinions of the author and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work. It is also supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625, by the Generalitat de Catalunya under contract 2009-SGR-980. The conducted experimentation has been possible thanks to the computing resources provided by the Umeå University, Sweden. Special thanks to Johan Tordsson for having made this collaboration possible.

Abstract

During the last years, IaaS providers have become interested in reducing their infrastructure energy consumption while preserving the agreed QoS with their clients. They want to optimize their infrastructure energy efficiency and to do so their VM placement algorithms need to know the current and future energy efficiency at different levels (Virtual Machine, node, infrastructure and service levels) and for potential actions such as service deployment or VM deployment, migration or cancellation.

This thesis contributes by providing a mathematical formulation for the previous aspects, as well as the design of a CPU utilization estimator used to calculate the aforementioned forecasts. A tool capable of performing such calculations in a real environment has been developed and tested in the context of this thesis. The conducted experimentation has been focused on finding the configuration parameter values that provide the best precision when forecasting the energy efficiency of the infrastructure for different workload types.

The obtained results indicate that the correct adjustment of these parameters leads to considerable precision improvements. Web workloads' induced resource utilizations are better forecasted when using parameter values which make the estimators more focused on noise filtering, even if they react slowly to changes. Batch workloads are quite the opposite: the parameter values have to make the estimators very reactive to utilization changes to obtain a good overall precision. When these two workloads are combined, the batch workload predominates and more reactive predictors are required. In fact, it has been stated that the obtained precision when running pure batch or mixed workloads partially depends on each execution, concretely on the amount of queued tasks during it. Finally, it has been observed that the forecasts precision degradation as such forecasts are performed for a longer time period in the future is smaller when running web workloads rather than when running batch or mixed ones, because of their nature.

Contents

Contents	v
List of Figures	vii
Nomenclature	xvi
1 Introduction	1
2 Background and State of Art	5
2.1 Background	5
2.2 State of the Art	8
3 Assessment and Forecast of the Eco-Efficiency for Infrastructure Providers	13
3.1 Energy and Ecological Efficiency	13
3.2 Energy Efficiency Assessments and Predictions	15
3.2.1 Energy Efficiency at Node Level	15
3.2.2 Energy Efficiency at Infrastructure Level	18
3.2.3 Energy Efficiency at VM Level	18
3.2.4 Energy Efficiency at Service Level	20
3.3 Energy Efficiency Predictions for Potential Actions	21
3.3.1 Service Deployment	21
3.3.2 VM Deployment (unknown placement)	24
3.3.3 VM Deployment (known placement)	26
3.3.4 VM Cancellation	27
3.3.5 VM Migration (unknown placement)	28
3.3.6 VM Migration (known placement)	29
4 CPU Estimator	32
5 Off-line Power Model	35

6	Experimentation and Validation	38
6.1	Methodology to generate web workload	38
6.1.1	SPECweb2005 Overview	39
6.1.2	Testing environment	40
6.1.3	Workload generation	42
6.2	Methodology to generate batch workload	45
6.2.1	COMP Superscalar Overview	47
6.2.2	Testing environment	47
6.2.3	Workload generation	49
6.3	Methodology to generate mixed workload	51
6.4	Model validation with web workload	53
6.4.1	Experiment 1: predictors' parameter tuning	54
6.4.2	Experiment 2: last samples error tuning	57
6.4.3	Experiment 3: final results	60
6.5	Model validation with batch workload	62
6.5.1	Experiment 1: predictors' parameter tuning	62
6.5.2	Experiment 2: last samples error tuning	66
6.5.3	Experiment 3: final results	67
6.6	Model validation with heterogeneous workload	69
6.6.1	Experiment 1: predictors' parameter tuning	69
6.6.2	Experiment 2: last samples error tuning	72
6.6.3	Experiment 3: final results	74
7	Conclusions and Future Work	77
7.1	Conclusions	77
7.2	Future Work	79
8	Publications	81
Appendix A: Detailed results for the experimentation with web workload		82
Appendix B: Detailed results for the experimentation with batch workload		86
Appendix C: Detailed results for the experimentation with mixed workload		89
References		92

List of Figures

2.1	Typical OPTIMIS environment	6
2.2	Data used by the Ecoefficiency Tool	8
4.1	Variable estimator overall design	33
5.1	Real power consumption incurred by variable CPU utilizations . .	36
5.2	Real power consumption based on fully-occupied CPUs	37
6.1	SPECweb2005 controlled parameters during the experiment . . .	40
6.2	Virtual Machine distribution (web workload experimentation) . .	42
6.3	Variable web workload generation	43
6.4	Effect of starting multiple SPECweb2005 clients within a single iteration	46
6.5	Virtual Machine distribution (batch workload experimentation) .	48
6.6	Batch workload generation using COMPSs	49
6.7	Batch workload intensity	51
6.8	Arrival rate (batch workload experimentation)	52
6.9	Virtual Machine distribution (mixed workload experimentation) .	52
6.10	Mean obtained CCC for different M values (moving average pre- dictor) in Experiment 1 (Web)	55
6.11	Mean obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Web)	56
6.12	Mean obtained CCC for different $Samples_LR$ values (linear re- gression predictor) in Experiment 1 (Web)	56
6.13	Mean obtained CCC for different $Alpha_DES$ values (double ex- ponential smoothing predictor) in Experiment 1 (Web)	57
6.14	Infrastructure's energy efficiency assessments and forecasts during Experiment 1 (Web)	58
6.15	Mean obtained CCC for different $N_Samples$ values (determines the "best forecast") in Experiment 2 (Web)	59
6.16	Infrastructure's energy efficiency assessments and forecasts during Experiment 2 (Web)	59

LIST OF FIGURES

6.17	Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 3 (Web)	60
6.18	Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (Web)	61
6.19	CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions in Experiment 3 (Web) . . .	62
6.20	Mean obtained CCC for different M values (moving average predictor) in Experiment 1 (Batch)	63
6.21	Mean obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Batch)	64
6.22	Mean obtained CCC for different $Samples_LR$ values (linear regression predictor) in Experiment 1 (Batch)	64
6.23	Mean obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Batch)	65
6.24	Infrastructure’s energy efficiency assessments and forecasts during Experiment 1 (Batch)	65
6.25	Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Batch)	66
6.26	Infrastructure’s energy efficiency assessments and forecasts during Experiment 2 (Batch)	67
6.27	Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (Batch)	68
6.28	CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions in Experiment 3 (Batch) . .	69
6.29	Mean obtained CCC for different M values (moving average predictor) in Experiment 1 (Mixed)	70
6.30	Mean obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Mixed)	70
6.31	Mean obtained CCC for different $Samples_LR$ values (linear regression predictor) in Experiment 1 (Mixed)	71
6.32	Mean obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Mixed)	71
6.33	Infrastructure’s energy efficiency assessments and forecasts during Experiment 1 (Mixed)	72
6.34	Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Mixed)	73
6.35	Infrastructure’s energy efficiency assessments and forecasts during Experiment 2 (Mixed)	73
6.36	Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (first execution, Mixed)	74

LIST OF FIGURES

6.37	CCC degradation of the infrastructure's energy efficiency forecasts for different time interval predictions in Experiment 3 (first execution, Mixed)	75
6.38	Infrastructure's energy efficiency assessments and forecasts during Experiment 3 (second execution, Mixed)	75
6.39	CCC degradation of the infrastructure's energy efficiency forecasts for different time interval predictions in Experiment 3 (second execution, Mixed)	76
A.1	Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Web)	82
A.2	Individual obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Web)	83
A.3	Individual obtained CCC for different $Samples_LR$ values (linear regression predictor) in Experiment 1 (Web)	83
A.4	Individual obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Web)	84
A.5	Individual obtained CCC for different $N_Samples$ values (determines the "best forecast") in Experiment 2 (Web)	84
A.6	Individual obtained CCC for different $N_Samples$ values (determines the "best forecast") in Experiment 3 (Web)	85
B.1	Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Batch)	86
B.2	Individual obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Batch)	87
B.3	Individual obtained CCC for different $Samples_LR$ values (linear regression predictor) in Experiment 1 (Batch)	87
B.4	Individual obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Batch)	88
B.5	Individual obtained CCC for different $N_Samples$ values (determines the "best forecast") in Experiment 2 (Batch)	88
C.1	Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Mixed)	89
C.2	Individual obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Mixed)	90
C.3	Individual obtained CCC for different $Samples_LR$ values (linear regression predictor) in Experiment 1 (Mixed)	90
C.4	Individual obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Mixed)	91

LIST OF FIGURES

C.5	Individual obtained CCC for different $N_{Samples}$ values (determines the “best forecast”) in Experiment 2 (Mixed)	91
-----	---	----

Nomenclature

Glossary

BeSIM	Back-end SIMulator (SPECweb2005 component)
BLO	Business Level Objective
BREEAM	Building Research Establishment Environmental Assessment Methodology
CASBEE	Comprehensive Assessment System for Building Environmental Efficiency
CCC	Concordance Correlation Coefficient
COMPSs	COMP Superscalar
CPU	Central Processing Unit
CU	Computer Units (generic performance metric)
DBMS	DataBase Management System
Domain-0	Privileged domain of Xen Hypervisor
EMOTIVE	Elastic Management Of Tasks In Virtualized Environments
FIB	Facultat d'Informàtica de Barcelona (Barcelona School of Informatics)
FIFO	First In First Out
HM	Holistic Manager (OPTIMIS component)
IaaS	Infrastructure as a Service
IO	Input/Output

NOMENCLATURE

IP	Infrastructure Provider (IaaS Provider)
IPC	Instructions per Cycle
ISO	International Organization for Standardization
LEED	Leadership in Energy and Environmental Design
MFLOPS	Millions of Floating-point Operations Per Second
MI	Monitoring Infrastructure (OPTIMIS component)
MIPS	Millions of Instructions Per Second
MPC	Memory Accesses per Cycle
MWIPS	Millions of Whetstone Instructions Per Second
OCCI	Open Cloud Computing Interface
OLTP	OnLine Transaction Processing
OPTIMIS	Optimized Infrastructure Services
OS	Operating Systems
OVF	Open Virtualization Format
PaaS	Platform as a Service
PM	Programming Model
PUE	Power Usage Effectiveness
QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
SaaS	Software as a Service
SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SP	Service Provider (SaaS Provider)
SPARC	Scalable Processor ARChitecture

SPC	Storage Performance Council
SPEC	Standard Performance Evaluation Corporation
SUT	System Under Test
TPC	Transaction Processing Performance Council
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMM	Virtual Machine Manager (OPTIMIS component)

List of Symbols

$\%B$	Percentage of brown energy used by a IaaS provider
$\overline{Perf_{CPU}}$	Average performance per CPU in the whole IaaS provider infrastructure
$\overline{Pwr_{incr}}$	Average incremental power consumption per CPU in the whole IaaS provider infrastructure
ρ	System utilization
$\hat{R}_i$	Forecasted real power consumption of node i
$\hat{U}_0^i$	Forecasted CPU utilization of Domain-0 of node i
$\hat{U}_{VM_j^i}$	Forecasted CPU utilization of VM j in node i
$\widehat{EcEf}_{infr}$	Forecasted ecological efficiency value of the whole IaaS provider infrastructure
$\widehat{EcEf}_{serv_k}$	Forecasted ecological efficiency of service k
$\widehat{EcEf}_{VM_j}$	Forecasted ecological efficiency of VM j
$\widehat{EnEf}_{infr}$	Forecasted energy efficiency value of the whole IaaS provider infrastructure
$\widehat{EnEf}_{serv_k}$	Forecasted energy efficiency of service k
$\widehat{EnEf}_{VM_j}$	Forecasted energy efficiency of VM j

NOMENCLATURE

$\widehat{Perf}_{node_i}$	Forecasted performance of node i in Computing Units (CU)
$\widehat{Perf}_{VM_j}$	Forecasted performance of VM j in Computing Units (CU)
$\widehat{Pwr}_{node_i}$	Forecasted power consumption of node i
$\widehat{Pwr}_{VM_j}$	Forecasted power consumption of VM j
$\widehat{EcEf}_{node_i}$	Forecasted ecological efficiency value of node i
$\widehat{EnEf}_{node_i}$	Forecasted energy efficiency value of node i
α_{DES}	First configuration parameter of the double exponential smoothing predictor
α_{ES}	Configuration parameter of the exponential smoothing predictor
β_{DES}	Second configuration parameter of the double exponential smoothing predictor
c	Number of servers (execution slots)
I_{DAY_i}	Workload intensity during office hours (8am to 8pm)
I_{NIGHT_i}	Workload intensity during out of office hours (8pm to 8am)
M	Configuration parameter of the moving average predictor
m_k	Number of VMs of service k
N	Number of nodes of the whole Iaas provider infrastructure
$N_{Samples}$	Number of error samples to average to calculate the mean absolute error of the internal predictors
n_i	Number of VMs being executed in node i
N_{TOTAL}	Total number of sessions for the week from 5th to 11th April 2010 in the FIB workload
N_{DAY_i}	Number of sessions during office hours (8am to 8pm)
N_{NIGHT_i}	Number office during out of office hours (8pm to 8am)
$nodes$	Set of nodes, index i

NOMENCLATURE

p	Number of processors
PUE	Power Usage Effectiveness
R_i	Real power consumption of node i
$Samples_LR$	Configuration parameter of the lineal regression predictor
$services$	Set of services, index k
T_{HOURL}	Time that a simulated hour lasts during the experiment execution
U_0^i	CPU utilization of Domain-0 of node i
$U_{CPU_{def}}$	Default initial utilization per CPU
$U_{VM_j^i}$	CPU utilization of VM j in node i
VMs	Set of virtual machines, index j
VMs_i	Subset of virtual machines being executed in node i , index j
w	Number of workers
$\%E_e$	Percentage of energy coming from an energy source e
$\delta_{cert.}$	Boolean variable: equals to 1 if the provider owns all the certifications required in the Service Manifest
λ_{JOB}	Job arrival rate
λ_{TASK}	Task arrival rate
$\lambda_{JOB_{MAX\ n\ Workers}}$	Maximum job arrival rate that the system can cope with
μ_{TASK}	Task service rate
$\rho_{En.Sources,\%B}$	Conversion factor from power consumption $[W]$ to CO2 emission rate $[\frac{gr}{s}]$
CPU_{alloc_i}	Total number of CPUs allocated to services in node i
CPU_{node_i}	Total number of CPUs in node i
CPU_{serv_k}	Number of CPUs required by service k
CPU_{VM_j}	Number of CPUs required by VM j

$EcEf$	Ecological efficiency
$EcEf_{node_i}$	Ecological efficiency assessment of node i
$EcEf_{infr}$	Ecological efficiency assessment of the whole IaaS provider infrastructure
$EcEf_{serv_k}$	Ecological efficiency assessment of service k
$EcEf_{VM_j}$	Ecological efficiency assessment of VM j
Em_e	Delivered emissions per Watt per second associated to energy source e
$EnEf_{node_i}$	Energy efficiency assessment of node i
$EnEf_{infr}$	Energy efficiency assessment of the whole IaaS provider infrastructure
$EnEf_{serv_k}$	Energy efficiency assessment of service k
$EnEf_{VM_j}$	Energy efficiency assessment of VM j
$Perf_{max_i}$	Maximum performance of node i in Computing Units (CU)
$Perf_{node_i}$	Actual performance of node i in Computing Units (CU)
$Perf_{VM_j}$	Actual performance of VM j in Computing Units (CU)
Pwr_{incr_i}	Incremental power consumption per CPU in node i
Pwr_{max_i}	Maximum power consumption of node i (when U=100%)
Pwr_{min_i}	Minimum power consumption of node i (when U=0%)
Pwr_{node_i}	Actual power consumption of node i
Pwr_{VM_j}	Actual power consumption of VM j

Chapter 1

Introduction

Over the last years, Cloud computing has become a consolidated computing paradigm that allows sharing different kinds of resources over the network in an automated way. These resources can be, for instance, networks, servers¹, storage, applications or services. Different types of providers have emerged, which can be classified depending on the service model they adopted, namely [2]:

- Software as a Service (SaaS) providers: they provide complete applications to the user that are running on a Cloud infrastructure. These applications can be accessed using a thin client interface, such as a web browser.
- Platform as a Service (PaaS) providers: they offer means to the consumer to deploy its own applications to the Cloud infrastructure. These applications are created using programming languages, libraries, services and tools supported by the provider.
- Infrastructure as a Service (IaaS) providers: they provide low level computing resources on which the user can run its own software, such as operating systems and applications. These resources can include processing power, storage or networks, and can be provided physically or virtually (most normally).

Typically, IaaS providers allow the user to create Virtual Machines (VMs) and establish Virtual Local Area Networks (VLANs) between them. A Virtual Machine is a replica of the underlying physical machine and the user feels as if running directly on the physical machine [3]. Different Operating Systems (OS) can be installed inside of them, and each VM can execute tasks independently of the ones being executed by other VMs.

¹In this publication, the words “node”, “server”, “physical server”, “physical machine” will be indistinctly used to refer to a computer.

The use of VMs allows the creation of services that can ask for more or less resources depending on their instantaneous demand variation (scalability). For instance, by simply adding a new VM replica to an already existing service, it can process the requests of a bigger number of users of that service. Another advantage of using VMs is that they offer the possibility to be migrated from one physical host to another without stopping its execution, transparently to the user.

Back in 2006, the annual energy consumption of the datacenters in the United States was already of about 61 Billion kWh, which is equivalent to the energy consumed by 5.8 million average U.S. households [4]. From that year, this amount of energy has significantly increased. What is more, given that most of the energy produced in U.S. and around the world comes from burning coal and natural gas [5], the datacenter's energy consumption has a direct impact on climate change [6].

An individual IaaS provider typically owns a very large number of physical hosts, where the different users' VMs are deployed. Altogether, they consume a massive amount of energy which represents about 30 percent of a their datacenter's operating expenses [7]. Moreover, IaaS providers have to pay fines if their carbon footprint exceeds the limits imposed by international regulations such as the Kyoto Protocol [8]. As a consequence, IaaS providers have a great interest in reducing the total power consumption of their physical hosts.

Each server has a power consumption that is not proportional to its resource utilisation. Even though the power consumption grows linearly with regards to its utilisation [9], if the host is not used it still has an associated idle power consumption. Using VM migration, multiple VMs can be consolidated into a single physical server and the remaining idle servers can be turned off, avoiding their idle power consumption and therefore reducing the overall datacenter's energy consumption.

Until recently, IaaS providers only cared about providing the required performance to their users, regardless of the incurred energy consumption. Green Cloud computing is envisioned to achieve efficient processing while minimising the energy consumption [10]. Energy efficiency is defined as the ratio between the useful output of a process and the energy input to this process [11]. It is a generic term which can be applied in multiple fields, but when talking about Green Cloud computing, improving the energy efficiency refers to using less energy to produce the same amount of services or useful output. Therefore, the ultimate target of Green Cloud computing is maximising the energy efficiency.

IaaS providers need tools to measure and predict the energy efficiency of their infrastructure in real time (this includes the performance and the power being delivered), as well as the generated carbon emissions. The current available tools are in fact benchmarks which are able to evaluate the energy efficiency of a node or

set of nodes in offline mode, but they are not able to evaluate it in real time while executing general applications, or to predict it for a given amount of time in the future. Some publications use real-time metrics affecting the energy efficiency to drive their VM placement algorithms, but not the energy efficiency itself. What is more, IaaS providers use VM placement algorithms which need to know what will be the impact of performing a VM deployment/migration/cancellation on the infrastructure's energy efficiency in order to optimise it. Even though some of these calculations have been addressed by the related literature, they have been only evaluated in simulated environments, but no tools have been provided to IaaS providers to gather such information.

In this context and taking into account the limitations of the current available tools, the present thesis tackles the following innovations:

- Develop mechanisms to assess and forecast the energy efficiency and the ecological efficiency (defined in Section 3.1) of a Cloud IaaS provider's infrastructure in real time. Different levels will be considered: infrastructure level, node level, service level and VM level. The performed forecasts will be calculated taking into account the previous CPU utilizations of all the VMs involved in the calculation (depending on the considered level), as well as the maximum performance that each node can provide. Then, predictions on performance and power consumption will be made and subsequently used to predict the energy/ecological efficiency. With this information, IaaS providers are able to continuously monitor the energy efficiency of the different levels and predict possible violations to predefined thresholds.
- Develop mechanisms to predict the energy and ecological efficiency of a Cloud IaaS provider infrastructure for potential actions, including: service deployment, VM deployment, VM migration and VM cancellation. These predictions are calculated based on the same information as the previous point, but additionally considering the changes in the performance and power consumption if the potential action is performed. With this information, the placement algorithms can determine whether performing a VM migration, for instance, can lead to increase the overall infrastructure energy/ecological efficiency.
- Develop a mechanism to estimate the future CPU utilization of a VM. An estimator which combines the forecasts of 4 different and independent predictors has been developed to this end. The obtained predictions are used to predict the performance delivered to each VM as well as the power consumption of a given node (also necessary to predict the energy efficiency), using the model described in Chapter 5.

The present thesis is structured as follows. In Chapter 2, the necessary theoretical and practical concepts needed to understand the contents of this thesis are provided, as well as a dissertation describing the work which has already been performed in the related literature. Chapter 3 contains all the mathematical models developed to evaluate and forecast the energy efficiency at different levels (infrastructure, node, service and VM levels) for the current status of the provider, as well as to forecast it for potential actions including VM deployment, migration, cancellation and service deployment. Therefore, it contains the full description of the contributions of this thesis. Chapters 4 and 5 contain, respectively, the description of the CPU estimator and the offline power model which are used when predicting the energy efficiency at the different aforementioned levels, for a given amount of time in the future. In Chapter 6, the developed tool is evaluated using two kinds of realistic workload and the configuration parameters of the CPU estimator described in Chapter 4 are tuned in order to obtain a good precision when forecasting the energy efficiency of the infrastructure. Finally, in Chapter 7, the conclusions drawn from this work are presented, and additional possible innovations and research lines are provided.

Chapter 2

Background and State of Art

2.1 Background

Along with the outburst of Cloud computing came a surge of interest in Green Computing. The latter is a field where the engineers and developers try to optimise the energy efficiency of various computing processes either by intervening into the hardware or the software architecture.

Energy efficiency is a generic term defined as the ratio between the useful output of a process and the energy input to this process [11]. The useful output mentioned in this general definition can be the tones of a product or even the number of instructions executed per each Joule consumed. In some publications, as in [12], this ratio's numerator and denominator are inversed and referred to as energy intensity. Nevertheless, they still implicitly contain information about the energy efficiency of a process.

Note that this definition (useful output per Joule) is equivalent to the amount of useful output per second divided by the amount of Joules consumed during that second, that is, Watts. In this thesis, the adopted metric to measure the energy efficiency has been MWIPS/W. The MWIPS refers to the Millions of Whetstone Instructions per Second performed with respect to those performed by a SPARCstation 20-61 with 128 MB RAM, a SPARC Storage Array, and Solaris 2.3. The MWIPS of each physical node have been measured using the UnixBench benchmark [13].

A tool to measure the energy and ecological efficiency (see Section 3.1) has been developed as a component called the Ecoefficiency Tool, as part of the OPTIMIS project [14].

The OPTIMIS (Optimized Infrastructure Services) project, developed under the FP7-ICT-2007-1 program, is a very extense project involving many different components. For the sake of simplicity, only the basic concepts necessary to understand the role of the Ecoefficiency Tool in this environment and the conducted

The available nodes and the BLO are configured in the Holistic Manager (HM) component. If the BLO seeks to maximize the energy/ecological efficiency of the infrastructure, the VMM will choose the destination node that maximizes the infrastructure's energy/ecological efficiency if the VM is deployed in it. When the node to host a particular VM has been chosen, the VMM issues the VM creation by sending its VM description to the virtualization middleware.

OPTIMIS currently supports 3 virtualization middleware solutions: EMOTIVE Cloud [15], OpenNebula [16] and OpenStack [17]. During the experimentation of this thesis, EMOTIVE Cloud has been used. This middleware allows creating/destroying/migrating VMs, and can be accessed as a REST-compliant Web service. The provided interfaces follow the OCCI 1.0 specification [18], and the VM descriptors used during VM deployment follow the OVF 1.0.0 standard [19].

When the BLO is established, the HM configures the Ecoefficiency Tool in such a way that if it detects that the infrastructure's energy/ecological efficiency is (or is going to be) below a given threshold, it will send back notifications to the HM. To this end, the Ecoefficiency Tool is continuously monitoring the energy/ecological efficiency of the infrastructure, and also predicting it for a given time period in the future (in order to detect potential threshold violations).

Note that the performed predictions need to be as accurate as possible in order to avoid unnecessary and incorrect notifications. This is the reason why the experimentation in this thesis has focused on trying to maximize the precision of the tool when performing these forecasts, by appropriately tuning the parameter configuration of the tool's CPU estimators (see Chapter 4).

If the HM receives a notification from the Ecoefficiency Tool, it evaluates possible actions to perform in order to improve the infrastructure's energy/ecological efficiency, such as migrating a VM. The HM can also be asked to replicate or to cancel a given VM by other OPTIMIS components. In any of these cases (VM migration/deployment/cancellation), the HM asks the Ecoefficiency Tool to calculate what will be the impact in the infrastructure's energy/ecological efficiency, by using the methods described in Section 3.3.

As it can be observed in Figure 2.2, the Ecoefficiency Tool retrieves three different kinds of data in order to calculate/predict the energy/ecological efficiency, namely:

- Performance data: it is collected offline, for each node, using the UnixBench benchmark. This benchmark determines the maximum Millions of Whetstone Instructions per Second (MWIPS) that each node is able to perform with respect to those performed by a SPARCstation 20-61 with 128 MB RAM, a SPARC Storage Array, and Solaris 2.3.

-
- CPU utilization data: it is captured in real time by the Monitoring Infrastructure component (MI), using the *virt-top* utility [20]. The collected information is inserted into the OPTIMIS database, from where the Ecoefficiency Tool is able to retrieve it later. The information provided by *virt-top* includes the individual CPU utilization of each VM of the node where it is executed.
 - Power data: it is obtained from a power sensor using the SNMP protocol. The nodes to be monitored are connected to this power sensor. This data is also obtained by the MI and inserted into the OPTIMIS database, from where the Ecoefficiency Tool retrieves it later. The captured data includes the instantaneous real power consumption of each node in Watts.

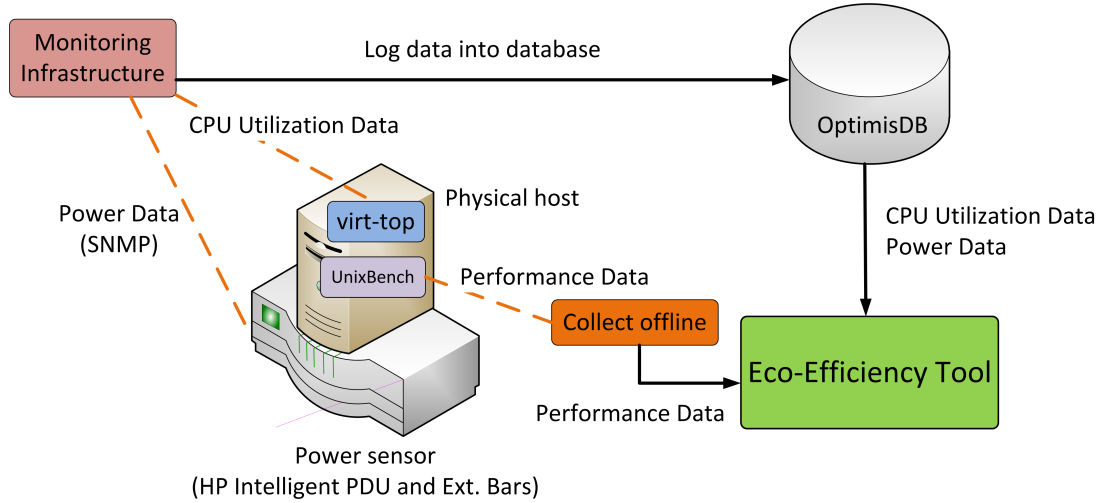


Figure 2.2: Data used by the Ecoefficiency Tool

2.2 State of the Art

In the last years there have been several efforts that introduced the concept of energy efficiency into the Cloud computing field and proposed related solutions and concepts. In [21] the authors perform an introductory, though interesting survey regarding the impact of energy-saving strategies for the management of integrated systems and Clouds specifically, and conclude in certain suggestions, such as reducing the software and hardware related energy cost, improving load balancing, reducing energy consumption due to communication and take into consideration the CO₂ emissions resulting from datacenters. These concepts are

aligned with the design principles that have been used in the solution adopted in this thesis.

In [22] different hardware configurations' impact on energy consumption and performance for typical applications are evaluated, and it is concluded that different workloads need different hardware configurations to achieve performance and energy efficiency. What is interesting of this publication is that in order to obtain energy efficiency assessments which can be compared across different workloads, a metric of the form iops/J (input/output operations per Joule) is derived. Filesystem performance metrics of the form iops/s can not be used to compare between different workloads, as each of them targets a different application domain. A similar idea is used in this publication. Given that the CPU utilization incurred by a VM depends on the application it is running and the underlying hardware, it can not be used as a universal performance metric comparable across different platforms. In order to be able to compare the computing performance delivered across nodes with different architectures, a metric of the form MWIPS (see Section 2.1) will be used. While in [22] the evaluated energy efficiency was in the form of read/write operations by energy unit, this thesis assesses the amount of computation power delivered per unit of real power consumed by the node.

The Storage Performance Council (SPC) also developed a couple of benchmarks called SPC-1C/E and SPC-1/E, which also evaluate the energy efficiency of a storage subsystem [23]. These benchmarks report the energy efficiency with a metric of the form $\frac{T}{s \cdot W}$, where T are the I/O operations per time unit s and per electricity consumed W in Watts. The benchmark is pretty similar to the one presented in [22], but while in [22] the maximum energy efficiency was measured for different workload types and system configurations, in the SPC benchmarks the energy efficiency of a given system is evaluated for 3 different levels of IO load and using a mix of 8 different IO patterns modeled after common OLTP applications (e.g., DBMS, email server). These different load levels are established at 0%, 50% and 80% of the maximum measured load during the performance benchmarks. What was found interesting of this publication, and has also been applied in the models of this thesis, is the fact that the energy efficiency is not a metric only to be measured when the system is fully loaded, but also at intermediate load levels. In fact, the proposed models in this thesis are not restricted to a limited number of load levels, but to all the possible load levels between 0% and 100%.

The Transaction Processing Performance Council (TPC) released a set of benchmarks initially targeted to measure the performance of a transactional application. Later, they released an additional benchmark, called TPC-E which allowed to measure the energy consumption corresponding to the work performed by the performance benchmarks, and to merge both performance and energy information to produce a metric of the form $\frac{W \cdot s}{T}$, where the electricity (W) per

time unit (S) is divided by the total amount of transactions (T) [23]. Note that in this benchmark, the energy efficiency is presented inverted with regards to the formulations of the previously referred publications. This notation allows the benchmark to present the resulting metric for the overall system under test (SUT), called the “primary metric”, as a sum of “secondary metrics”, all of them in the units “Watts per Performance”. This is possible because both primary and secondary metrics share the same denominator (performance). In this context, the “secondary metrics” represent the amount of power consumed by each of the subsystems of the SUT to deliver the obtained performance of the benchmark, therefore they represent the contribution of each of the subsystems to the overall benchmark result. Given that in this thesis the energy efficiency definition is not of the form “Watts per Performance” but “Performance per Watt”, it is not possible to apply a similar approach to calculate the energy efficiency of the whole infrastructure out of the energy efficiency of the nodes comprised in it. As it will be seen in Section 3.2.2, additional formulation will be necessary to calculate the energy efficiency of the whole infrastructure, although it will still leverage from some results of the formulation to calculate the energy efficiency of a single node.

Finally, the Standard Performance Evaluation Corporation (SPEC) has also released benchmarks to measure the energy efficiency of a system executing a server side Java application (SPECpower_ssj2008 benchmark) and of a system emulating a web server serving different kinds of web workloads (SPECweb.2009 benchmark) [23]. The SPEC benchmarks also evaluate the consumed energy and performance at multiple load levels, like SPC-1C/E and SPC-1/E. The provided metric is of the form $\frac{T}{W}$, where T can represent server side Java operations or web transactions, depending on the benchmark.

From all the related work analysed so far, it can be concluded that SPEC benchmarks are the ones providing a metric for energy efficiency most similar to the one studied in this thesis. However, it has to be observed that so far, only benchmarks to measure the “offline energy efficiency” of a given application have been considered in the current literature, but there are no tools to measure the real time energy efficiency of a system. Moreover, what is interesting from an IaaS provider point of view is to be able to maximize the energy efficiency of its infrastructure, and to do that it needs tools to actually predict the future energy efficiency that it will have. This is also not addressed by the analysed benchmarks. This is actually one of the main innovations of the present publication: being able to assess and predict the real time energy efficiency of the infrastructure.

Energy efficiency assessments and forecasts can be used by placement algorithms in order to maximize the energy efficiency of the infrastructure. For instance, in [24] two task consolidation algorithms are presented which are used to optimise the task placement of a process in a Cloud infrastructure taking into account the energy efficiency. On the one hand, these algorithms are targeted

for batch applications and take into account resource utilizations and power consumptions when deciding in which node a task must be run. In this thesis, the provided energy efficiency metric can be used by VM placement algorithms (not studied in this publication) independently of the underlying workload type (web, batch, etc.). Moreover, the resource utilization incurred by a given VM (or task) is not the same if it runs in nodes with different characteristics. Therefore, in the present thesis performance is considered as the universal metric which can be translated for each specific node into resource utilization and in turn into power consumption. On the other hand, the algorithms in [24] perform placement optimisation when a task has to be scheduled to a particular node, but not during its execution. In Cloud computing environments, tasks can be encapsulated in VMs, which can be migrated while running. So, the impact of migrating a VM on the infrastructure's energy efficiency has to be studied in order to allow placement algorithms to take it into consideration. This way, energy efficiency can be optimised not only during VM deployment but also during VM migration.

The authors in [25] present a resource management technique that applies to Cloud datacenters that leverages from energy efficiency measurements. The solution proposed mainly focuses in minimizing the energy consumption by migrating and consolidating VMs while keeping the QoS level as high as possible. The findings of that experiment showed that there is great potential in reducing the energy consumption by optimising the resource management in a Cloud infrastructure.

In [26], the authors observed that co-located VMs with high memory accesses per cycle (MPC) resulted in higher cache conflict, which decreased its number of executed instructions per cycle (IPC) and hence increased its execution time. This can be translated as performance degradation and higher energy consumption due to this longer execution time. Also, VMs with high IPC were observed to cause a higher CPU utilization, which lead to a greater power consumption. The authors then conclude that co-scheduling VMs with heterogeneous characteristics is beneficial from energy and performance point of view. With this idea in mind, they propose a VM placement algorithm which takes into account the real-time MPC and IPC measurements of each VM to obtain a VM placement solution which potentially maximises the VM performance and minimizes the VM energy consumption.

While the previously described benchmarks (SPC-1C/E, SPC-1/E, TPC-E, SPECpower_ssj2008 or SPECweb_2009) are used to measure the offline energy efficiency of a node or set of nodes, the algorithm described in [26] considers real-time metrics, but not the energy efficiency itself. It actually considers metrics which potentially affect the energy efficiency (MPC and IPC). In this publication, the energy efficiency is directly considered as both the delivered performance and power are measured using real-time metrics and power-measuring devices.

It also has to be observed that both in [24] and [25] the evaluation of the models is performed by means of simulation, and not with real experimentation. In [26], they actually conduct their experimentation on a state of the art testbed. The experimentation of this thesis has also been conducted on real physical servers, using real CPU utilization and power data gathered by the Monitoring Infrastructure, as described in Section 2.1.

So, tools to evaluate the energy efficiency at real time during VM operation and to be able to consider the possibility of performing VM deployments or migrations are needed by Cloud providers in order to optimise the energy efficiency of their infrastructure. Taking into consideration the exposed requirements, additional effort has been put in the present work to allow the Ecoefficiency Tool to forecast the infrastructure's energy efficiency for a variety of potential actions including VM deployment/migration/cancellation or service deployment, which can then be used by VM placement algorithms to maximize the energy efficiency in real time.

Chapter 3

Assessment and Forecast of the Eco-Efficiency for Infrastructure Providers

As stated in Chapters 1 and 2, VM placement algorithms need to know the energy efficiency at which a node or a set of nodes are running. Some service contracts might also establish constraints regarding the minimum energy efficiency that a VM or the whole service should accomplish. What is more, these algorithms also need to know what will be the effect of performing certain actions (such as VM deployment, migration, cancellation or service deployment) on the energy efficiency of the whole infrastructure.

This chapter contains the description of the main research body conducted in this thesis, and addresses all the aforementioned aspects. In Section 3.1, the definition of energy efficiency from an IaaS provider point of view is provided, along with the definition of a very closely related metric: the ecological efficiency. Then, this definition is extended in Section 3.2 for each of the levels of interest: node, infrastructure, VM and service levels. Finally, Section 3.3 describes how to calculate the impact on the infrastructure energy efficiency if a VM is deployed, migrated or canceled, or if a service is deployed.

3.1 Energy and Ecological Efficiency

Energy efficiency is defined as *the ratio between the useful work performed and the energy consumed to do it* [11]. Note that this definition is equivalent to the ratio between the amount of work performed each second and the power delivered during that second (see Equation 3.1).

$$EnEf = \frac{Useful\ work}{Energy[J]} = \frac{\frac{Useful\ work}{s}}{\frac{Energy[J]}{s}} = \frac{Performance}{Power[W]} \quad (3.1)$$

Following this last definition, our aim is to obtain a formula which evaluates this relationship at physical node level in the first place (Section 3.2.1). The obtained formula will then be extended to obtain similar definitions for the remaining levels of our interest: infrastructure level (Section 3.2.2), virtual machine level (Section 3.2.3) and service level (Section 3.2.4).

An IaaS provider can be interested in having energy efficiency information for all of the aforementioned levels, as it can be used by its placement algorithms. These algorithms will make decisions in order to optimize, among other things, the energy efficiency of the whole infrastructure while at the same time fulfilling the SLA agreements with the provider's clients. However, a SaaS is only interested in the energy efficiency achieved by a given service, and not about the individual energy efficiency of the VMs comprising it or the nodes where they are being executed.

Note that the performance units mentioned in Equation 3.1 will not be the same when considered from an IaaS or a SaaS perspective. This aspect will be further discussed in Section 3.2.4.

The energy efficiency gives an idea of the performance delivered per unit of power consumed, but it does not reflect the environmental impact caused to deliver that performance. In order to consider it, a new metric is defined: the ecological efficiency.

Ecological efficiency is defined as the ratio between the useful work performed and the carbon dioxide emitted to do it. As it can be seen in Equation 3.2, the ecological efficiency definition is equivalent to the amount of work performed each second divided by the carbon dioxide emission rate (in $\frac{gr}{s}$) during that second.

$$EcEf = \frac{Useful\ work}{CO_2\ emissions[gr]} = \frac{\frac{Useful\ work}{s}}{\frac{CO_2\ emissions[gr]}{s}} = \frac{Performance}{CO_2\ em.\ rate\ [\frac{gr}{s}]} \quad (3.2)$$

This definition is very similar to the one of energy efficiency. The only change present in the formula is the denominator. In fact, the carbon dioxide emission rate (in $\frac{gr.CO_2}{s}$) can be calculated from the power consumption using the conversion factor $\rho_{En.Sources,\%B}$ presented in Equation 3.3. As it can be seen, this factor depends in the first place on the percentage of brown energy used by the provider (range 0.0-1.0) ($\%B$). From this percentage of brown energy, the percentage of energy coming from each brown energy source is represented by $\%E_e$ (range 0.0-1.0), while the delivered emissions per Watt per second associated to this same energy source are represented by Em_e (units in $\frac{gr}{W.s}$). The carbon dioxide emissions associated to green energy sources are assumed to be negligible.

$$CO_2 \text{ em. rate } \left[\frac{gr}{s} \right] = \overbrace{\left(\%B \cdot \sum_{e \in En.Sources} \%E_e \cdot Em_e \left[\frac{gr}{W \cdot s} \right] \right)}^{\rho_{En.Sources, \%B}} \cdot Power[W] \quad (3.3)$$

Using this notation, the relationship expressed in Equation 3.4 between the energy and the ecological efficiency can be deduced. Note that the relationship depends on the energy type being used by the IaaS provider, therefore even while having the same energy efficiency, two providers can have different ecological efficiencies. For the sake of simplicity, only the energy efficiency calculation formulas and explanations will be provided in this thesis, since the ecological efficiency can be deduced from them using this simple conversion factor. In addition, the term “eco-efficiency” will be used to refer to either energy or ecological efficiency indistinctly.

$$EcEf = \frac{Performance}{\rho_{En.Sources, \%B} \cdot Power(W)} = \frac{EnEf}{\rho_{En.Sources, \%B}} \quad (3.4)$$

3.2 Energy Efficiency Assessments and Predictions

In the following subsections, different considerations concerning eco-efficiency assessment calculations and predictions are discussed. As commented before, only the energy efficiency formulas and explanations will be tackled, as the ecological efficiency can be easily calculated, for a given provider, with the conversion factor exposed in Section 3.1. Section 3.2.1 introduces the methodology followed to evaluate and predict the energy efficiency of a single physical node of an IaaS, while Section 3.2.2 uses these results to do it at an IaaS level. In Section 3.2.3 these calculations are discussed when performed at VM level, while in Section 3.2.4 these results are used to assess and predict the energy efficiency of a service running on several VMs. The symbols used throughout this section and their meanings can be found in the Nomenclature.

3.2.1 Energy Efficiency at Node Level

An IaaS is not interested on which kind of work is being performed by the applications being run in their infrastructure, but on the resource utilisation they incur in the system. So, from an IaaS point of view, the useful work per second carried out by a server running several VMs can correspond to the amount of computing

performance delivered to them during that second, based on the assumption that the applications being run on the VMs are computing-intensive.

A first approach could be to evaluate the delivered computing performance per second as the amount of CPU time which has been assigned to the VMs running in the node during that second (amount of CPU utilisation). However, it is obvious that a 1990 computer with 50% of CPU utilisation will not give the same performance as a 2010 computer with the same CPU utilisation.

Possible metrics to measure the delivered performance per second could be MIPS or MFLOPS in this sense, given that they are independent of the underlying hardware and allows to compare the delivered performance per second in heterogeneous clusters. In particular, the MWIPS metric (Millions of Whetstone Instructions Per Second) is currently being used by the Eco-Efficiency Tool as it measures both integer and floating-point arithmetic performance. However, all these possible metrics will be referred in a general way as Computing Units (CU) throughout the rest of this thesis. Therefore, a CU will relate to the amount of delivered performance per second.

The maximum computing power (performance per second) which can be delivered by a node i is represented as $Perf_{max_i}$ and can be measured as the number of CUs delivered by the node when its processors are running at full speed with 100% of CPU utilisation. By multiplying $Perf_{max_i}$ by the sum of CPU utilisations (normalized to 1) incurred by the VMs running in the node, we can obtain the amount of computing power delivered by the node to the VMs (see Equation 3.5), assuming that the distribution of computing power among the VMs is proportional to their CPU utilisation. These assumptions are valid when running computing-intensive applications.

Note that the CPU utilisation of the privileged domain (referred to as Domain-0 in Xen Hypervisor nomenclature) has not been included in this sum, since it is not considered to be useful work. In fact, it can be considered as the overhead needed to run them. Also observe that the CPU utilisation values range from 0 to 100 independently of the number of processors present in the node.

$$Perf_{node_i} = \sum_{j \in VMs_i} \frac{U_{VM_j^i}}{100} \cdot Perf_{max_i} \quad (3.5)$$

On the other hand, the amount of power consumed by the node to execute these VMs along with the privileged domain corresponds to the real power and not to the apparent power, given that the real power reflects the energy really consumed and charged for by utilities [27]. It also needs to be taken into account the power overhead needed to run the node (cooling, lighting and power delivery losses). Following the definition of PUE [28], the total power consumption will be the real power consumed by the node multiplied by the PUE of the data-

center. Therefore, the total power consumption incurred by the node, including overheads, is depicted in Equation 3.6.

$$Pwr_{node_i} = R_i \cdot PUE \quad (3.6)$$

Taking into consideration Equations 3.5 and 3.6, the energy efficiency of a node from an IaaS point of view can be calculated using Equation 3.7.

$$EnEf_{node_i} = \frac{Perf_{node_i}}{Pwr_{node_i}} \quad (3.7)$$

One way to forecast the node energy efficiency for a given time in the future would be to keep track of the previous assessment results and extrapolate them using a linear regression. However, this approach does not take into account that the VMs running in a particular host can have completely independent behaviours. In order to overcome this problem, each of the variable terms of Equations 3.5 and 3.6 are predicted separately. Then, the energy efficiency is forecasted using the same relationship as in Equation 3.7.

On the one hand, the CPU utilisations of each VM and the Domain-0 are forecasted using an estimator which combines four prediction methods: moving average, linear regression, exponential smoothing and double exponential smoothing (see Chapter 4). Each CPU estimator uses as an input the previous CPU utilisation samples of the VM and outputs the best CPU utilisation forecast based on the last errors obtained by each of the prediction methods. Note that each VM/Domain-0 use an independent CPU estimator. The node's maximum computing performance $Perf_{max_i}$ will remain constant, as it is inherent to the node. Therefore, the performance forecast of a node can be calculated as exposed in Equation 3.8.

$$\widehat{Perf}_{node_i} = \sum_{j \in VMs_i} \frac{\widehat{U}_{VM_j^i}}{100} \cdot Perf_{max_i} \quad (3.8)$$

On the other hand, the node's real power consumption can be estimated by using an offline power characterization of the node, like in [9], and evaluating it for the total forecasted CPU load. This characterization is depicted as function $\widehat{R}_i(U_i)$, which gives as an output the estimated real power consumption in Watts of node i for a given CPU utilisation in the range of 0-100, independently of its number of cores (see Chapter 5 for more details). The PUE is inherent to the datacenter and can be assumed to be invariant for short time elapses. Considering these two exposed factors, the forecasted total power consumption incurred by the node, including overheads, can be calculated as in Equation 3.9. Note that the Domain-0 forecasted CPU utilisation ($\widehat{U}_0^i$) also has to be considered in the

estimation of the real power consumption as it is part of the CPU utilisation of the physical node.

$$\widehat{Pwr}_{node_i} = \widehat{R}_i \left(\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j^i} \right) \cdot PUE \quad (3.9)$$

Finally, the predicted node energy efficiency can be calculated using Equation 3.10. This approach accounts for the different behaviour of the VMs running in a particular host.

$$\widehat{EnEf}_{node_i} = \frac{\widehat{Perf}_{node_i}}{\widehat{Pwr}_{node_i}} \quad (3.10)$$

3.2.2 Energy Efficiency at Infrastructure Level

In order to describe the energy efficiency at infrastructure level following the general definition in Equation 3.1, the total performance and power delivered by all the nodes of the infrastructure must be calculated. Using Equations 3.5 and 3.6, this calculation is very simple and the resulting energy efficiency formula at infrastructure level results as depicted in Equation 3.11.

$$EnEf_{infr} = \frac{\sum_{i \in nodes} Perf_{node_i}}{\sum_{i \in nodes} Pwr_{node_i}} \quad (3.11)$$

Following the same reasoning as in Equation 3.11, the forecasted energy efficiency at infrastructure level can be calculated by forecasting each of the terms of Equation 3.11, which in fact means using Equations 3.8 and 3.9 to build the final formula, resulting in Equation 3.12.

$$\widehat{EnEf}_{infr} = \frac{\sum_{i \in nodes} \widehat{Perf}_{node_i}}{\sum_{i \in nodes} \widehat{Pwr}_{node_i}} \quad (3.12)$$

3.2.3 Energy Efficiency at VM Level

The amount of useful work carried out by a VM from an IaaS point of view corresponds to the fraction (normalized $U_{VM_j^i}$) of the maximum performance of the node where it is running ($Perf_{max_i}$) consumed by this VM, as depicted in Equation 3.13.

$$Perf_{VM_l} \mid l \in VMs_i = \frac{U_{VM_l^i}}{100} \cdot Perf_{max_i} \quad (3.13)$$

In order to determine the approximate power consumption incurred by a VM, its proportional part of the total power consumption R_i of the node where it is being executed is derived based on the VM's CPU utilisation, including overheads. This proportional distribution with regards to the CPU utilisation is accomplished with the second term of Equation 3.14. As in [27], a linear relationship between a node's real power consumption and its total CPU utilisation is assumed. Since Domain-0's CPU utilisation is required to run all of the n_i VMs being executed in node i , it is considered as an overhead which is uniformly distributed between them. Therefore, the real power consumption of a VM corresponds to the first two terms of Equation 3.14, which is in turn multiplied by the datacenter's PUE in order to determine the total amount of power needed to run the VM in the provider's premises.

$$Pwr_{VM_l} \mid l \in VMs_i = R_i \cdot \frac{U_{VM_l^i} + \frac{U_0^i}{n_i}}{U_0^i + \sum_{j \in VMs_i} U_{VM_j^i}} \cdot PUE \quad (3.14)$$

The energy efficiency of a VM can be calculated, using the results of Equations 3.13 and 3.14, as in Equation 3.15.

$$EnEf_{VM_l} \mid l \in VMs_i = \frac{Perf_{VM_l} \mid l \in VMs_i}{Pwr_{VM_l} \mid l \in VMs_i} \quad (3.15)$$

As in Section 3.2.1, the forecasted VM energy efficiency can be computed by estimating both the performance and the power consumption for that particular time in the future. The performance delivered in the future only depends on the future CPU utilisation that this VM will cause in the node where it runs and on the maximum performance which this node can deliver, which is inherent to the node and remains constant, as shown in Equation 3.16.

$$\widehat{Perf}_{VM_l} \mid l \in VMs_i = \frac{\widehat{U}_{VM_l^i}}{100} \cdot Perf_{max_i} \quad (3.16)$$

The future power consumption of a VM corresponds to the proportional part of the future real power consumption of the node where the VM is running. As in Equation 3.9, the future real power consumption of the node is obtained from the offline power model described in Chapter 5, using as input parameter the sum of all the forecasted CPU utilisations of all the VMs running in the node, as well as Domain-0's one (estimated as previously described in Section 3.2.1). The proportional factor which describes the part of this power consumed by the VM

is calculated as in Equation 3.14, by forecasting all of its terms. The resulting consumed power is multiplied by the datacenter's PUE (can be considered constant for short time elapses) to account for the extra power needed to sustain the infrastructure, finally resulting in Equation 3.17.

$$\widehat{Pwr}_{VM_l} \mid l \in VMs_i = \widehat{R}_i \left(\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j^i} \right) \cdot \frac{\widehat{U}_{VM_l^i} + \frac{\widehat{U}_0^i}{n_i}}{\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j^i}} \cdot PUE \quad (3.17)$$

Once the VM's performance and power have been estimated using the previous formulas, the expected energy efficiency of a VM can be calculated as in Equation 3.18.

$$\widehat{EnEf}_{VM_l} \mid l \in VMs_i = \frac{\widehat{Perf}_{VM_l} \mid l \in VMs_i}{\widehat{Pwr}_{VM_l} \mid l \in VMs_i} \quad (3.18)$$

3.2.4 Energy Efficiency at Service Level

A service is a set of components which can interact between them in order to provide a complete functionality to a client. Usually, these components are encapsulated in a set of VMs which can run in different hosts or even in different datacenters of different IaaS providers. Therefore, the energy efficiency of a service can be calculated as the sum of the performance delivered to all the VMs composing the service divided by the sum of their power consumption. Equation 3.19 summarizes this idea.

$$EnEf_{serv_k} = \frac{\sum_{j \in serv_k} Perf_{VM_j}}{\sum_{j \in serv_k} Pwr_{VM_j}} \quad (3.19)$$

In order to calculate the energy efficiency of a service for a future time, each of the delivered performance and power terms present in Equation 3.19 must be predicted, resulting in Equation 3.20.

$$\widehat{EnEf}_{serv_k} = \frac{\sum_{j \in serv_k} \widehat{Perf}_{VM_j}}{\sum_{j \in serv_k} \widehat{Pwr}_{VM_j}} \quad (3.20)$$

Note that a SaaS provider can select multiple IaaS providers to run different partitions of a given service. Locally, each IaaS provider will assess and forecast

the energy efficiency of each service partition as described in Equations 3.19 and 3.20, given that an IaaS provider is not necessarily aware that it is running just a partition of a wider service. The SaaS provider will use the same formulas, but it will have to separately ask each IaaS provider for the total performance and power being delivered to the service partition (a subset of VMs of the whole service) deployed in that particular IaaS provider, and aggregate all the performance and power deliveries separately in order to calculate the overall service's energy efficiency.

The approach described in this section can be useful for both IaaS and SaaS providers during service operation because it gives an idea of the computing power delivered to the service by Watt consumed. However, SaaS providers can be also interested in the performance delivered by the application in terms of application metrics like response time or throughput instead of the CUs it consumes. In this case, in order for a SaaS to assess and forecast the energy efficiency of a given service during its operation, it needs to know the power consumed by the VMs comprising it (or its estimation when forecasting the energy efficiency), so it can come up with metrics like throughput per Watt consumed, or session rate per Watt consumed.

3.3 Energy Efficiency Predictions for Potential Actions

In the following subsections, the mathematical models used to predict the energy efficiency impact on the infrastructure of an IaaS provider when performing actions such as service deployment or VM deployment, cancellation or migration are discussed.

3.3.1 Service Deployment

When a new service deployment is issued to a SP provider, it must decide to which IP provider it will forward the service creation request in order to issue the creation. In some cases, the owner of the service may have specified a minimum desired eco-efficiency level (either energy and/or ecological efficiency) for that service (when evaluated at service level, see Section 3.2.4). Moreover, the service owner may also have specified a given set of environmental-aware certifications which the IP where the service will be deployed must possess (i.e. LEED, BREEAM, CASBEE, ISO14000, etc.). In this case, the SP provider needs to determine whether the minimum required eco-efficiency for that service can be guaranteed by the IP provider where the service will be deployed, as well as if it meets the specified certification requirements.

To predict the energy efficiency at service level, the performance this service will require and the power consumption that it will incur need to be determined.

An estimation of the performance which will be delivered to a service can be computed by calculating the mean performance delivered to each of the CPUs that the service will use, and aggregate them. If the placement of the components (typically a component is encapsulated in a VM) of a service was known beforehand, the way to calculate the performance delivered to each of them would be similar as in Equation 3.16. Nevertheless, as the component placement is not determined until the service has been accepted by the IP provider, this approach cannot be applied, because the maximum performance of each of the destination nodes is not known beforehand. Neither is the CPU utilisation incurred by the VMs in each of them.

A fair way to overcome this limitation consists of assuming that each of the components will be deployed in nodes whose maximum performance is the mean performance of all the nodes of the infrastructure, and each of the CPUs occupied by the service's VMs will have the same default initial load (in our experimentation, 80%). Therefore, all the CPUs of all of the VMs of the service will receive the same amount of performance. So, just by multiplying the total number of CPUs of the service by the mean performance that each of these CPUs will receive (maximum performance that the CPU can deliver multiplied by the normalized utilisation of that CPU), the total performance delivered to the service can be calculated. Equation 3.21 summarizes the previous reasoning. The way to calculate the mean maximum performance per CPU is provided in Equation 3.22.

$$\widehat{Perf}_{serv_k} | serv_k \text{ depl.} = CPU_{serv_k} \cdot \overline{Perf}_{CPU} \cdot \frac{U_{CPU_{def}}}{100} \quad (3.21)$$

$$\overline{Perf}_{CPU} = \frac{\sum_{i \in nodes} \frac{Perf_{max_i}}{CPU_{node_i}}}{N} \quad (3.22)$$

In order to calculate the power consumption incurred by the service, the total number of CPUs which are going to be used by the service is multiplied by the mean power consumption per CPU to calculate the overall power consumption of the service, as depicted in Equation 3.23. Given that it is not known beforehand in which nodes the service VMs are going to be executed, the mean energy consumption per CPU of the whole datacenter will be determined.

As discussed in Chapter 5, the power consumption of a single node can be characterized as an idle power Pwr_{min_i} , which is independent of the node load, and a variable power which follows a linear trend with regards to the CPU utilisation. Therefore, each CPU's power consumption can be modelled as an idle part and a variable part which depends on the CPU utilisation.

On the one hand, the idle part is a “necessary power consumption” for a node to be able run, it can be considered as an overhead. If only one CPU of a node is active all this overhead is attributed to this CPU, while if n CPUs are active within this node, each one can be considered to consume $\frac{Pwr_{min_i}}{n}$ Watts of idle power. This idea is extrapolated when calculating the idle power consumption associated to a single CPU when considering the whole infrastructure. In this case, the idle power associated to a single CPU corresponds to the total idle power of all the nodes of the infrastructure divided by the total number of CPUs already allocated to the VMs in the whole infrastructure plus the additional CPUs which will be used by the service under deployment. This corresponds to the first part of the parenthesis of Equation 3.23.

On the other hand, the variable part corresponds to the second part of the parenthesis of Equation 3.23. As detailed in Chapter 5, the incremental power Pwr_{incr_i} is the extra power consumption of a node when one of its CPUs is occupied at 100%. Note that because the destination node is not known, the fairest approach is to consider the mean incremental power of all the nodes of the infrastructure, which can be calculated as in Equation 3.24. Given that the CPU utilisation of a VM depends on the performance of the node where it is deployed, a tentative initial default CPU utilisation $U_{CPU_{def}}$ is assumed (for instance of 80%) due to the fact that the destination node is not known. Therefore, the variable part of the power consumption of each CPU of the service VMs will be the mean incremental power multiplied by the normalized CPU utilisation of that CPU. It is worth mentioning that the overall VMs’ CPU utilisation has been considered to be equally distributed among all of the CPUs in this formulation.

$$\widehat{Pwr}_{serv_k} | serv_k \text{ depl.} = CPU_{serv_k} \cdot \left(\frac{\sum_{i \in nodes} Pwr_{min_i}}{CPU_{serv_k} + \sum_{i \in nodes} CPU_{alloc_i}} + \overline{Pwr_{incr}} \cdot \frac{U_{CPU_{def}}}{100} \right) \cdot PUE \quad (3.23)$$

$$\overline{Pwr_{incr}} = \frac{\sum_{i \in nodes} Pwr_{incr_i}}{N} \quad (3.24)$$

Taking into account the explanations in this section, the energy efficiency of a service upon its deployment can be calculated as in Equation 3.25. The term $\delta_{cert.}$ is a boolean variable which is equal to 1 if all the required environmental-aware certifications specified in the service manifest (service contract specified

by the service owner) are in possession of the IP provider where the service is to be deployed, and 0 otherwise. Therefore, if the IP provider has the required certifications, the forecasted energy efficiency of the service will be different of 0 and the service might be deployed in that provider (if the specified minimum energy efficiency requirements are met). In case the IP provider does not have any of the required certifications, the returned energy efficiency forecast will be 0 and that particular IP provider will not be selected for the service deployment.

$$\widehat{EnEf}_{serv_k} \mid serv_k \text{ depl.} = \delta_{cert.} \cdot \frac{\widehat{Perf}_{serv_k} \mid serv_k \text{ depl.}}{\widehat{Pwr}_{serv_k} \mid serv_k \text{ depl.}} \quad (3.25)$$

Similarly, from an IP provider point of view, it is required to know which will be the impact of deploying a service in the infrastructure. Depending on the forecasted impact and taking into account the BLOs of the IP provider, the service will be accepted or rejected by the IP provider. The forecasted energy efficiency of the whole infrastructure of an IP provider upon a service deployment is depicted in Equation 3.26. The only difference with regards to Equation 3.12 is that the extra performance being delivered to the new service as well as the power it will consume are now included in the nominator and the denominator of the formula, respectively. Note that as in Equation 3.25, the boolean term $\delta_{cert.}$ is used to consider the fact that if the IP provider does not possess all the required environmental-aware certifications specified in the service manifest, no extra performance or power will be delivered to the service, because it will be actually rejected.

$$\widehat{EnEf}_{infr} \mid serv_k \text{ depl.} = \frac{\sum_{i \in nodes} Perf_{node_i} + \delta_{cert.} \cdot \widehat{Perf}_{serv_k} \mid serv_k \text{ depl.}}{\sum_{i \in nodes} Pwr_{node_i} + \delta_{cert.} \cdot \widehat{Pwr}_{serv_k} \mid serv_k \text{ depl.}} \quad (3.26)$$

3.3.2 VM Deployment (unknown placement)

During a given service operation, its virtual machines suffer variations in the resource demand depending on the users' workload. In some situations, these VMs become overloaded and may need to be replicated to cope with the extra load. If this replication takes place the global power consumption and thereby energy associated costs will increase. One could aim for the energy efficiency of the infrastructure by using consolidation, but if the replicated VMs cannot fit in the currently active physical hosts, new physical hosts will have to be turned on and then the global energy efficiency will decrease. So, the impact of deploying a VM in terms of energy efficiency of the whole infrastructure needs to be predicted.

In this section, the forecasted energy efficiency of the infrastructure if a new VM is deployed to a yet unknown physical host is evaluated. If the placement algorithm then determines that a new VM deployment can lead to a situation which better satisfies the provider's BLOs, it will proceed by evaluating to which destination node should it be deployed (using the calculations described in Section 3.3.3).

The total computing performance which will be delivered by the infrastructure corresponds to the one which would be delivered if the VM was not deployed (first term of Equation 3.27) plus the extra performance delivered to the newly created VM (second term of the same equation). Given that the destination node is yet unknown, the average performance delivered per CPU is multiplied by the number of CPUs of the VM to calculate the second term of the equation. This approach is a particular case of what is done in Equation 3.21 to calculate the future performance delivered to a service under deployment. In this case only the number of CPUs of a single VM are considered, and not of a complete service. Refer to Section 3.3.1 for more details.

$$\widehat{Perf}_{infr} | VM_l \text{ depl.} = \sum_{i \in nodes} \widehat{Perf}_{node_i} + CPU_{VM_l} \cdot \overline{Perf_{CPU}} \cdot \frac{U_{CPU_{def}}}{100} \quad (3.27)$$

The expected power consumption can also be calculated by taking Equation 3.23 as a reference and particularizing it for a single VM of CPU_{VM_l} CPUs, as depicted in Equation 3.28.

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l \text{ depl.} = \sum_{i \in nodes} \widehat{Pwr}_{node_i} + CPU_{VM_l} \cdot \left(\frac{\sum_{i \in nodes} Pwr_{min_i}}{CPU_{VM_l} + \sum_{i \in nodes} CPU_{alloc_i}} + \right. \\ \left. + \overline{Pwr_{incr}} \cdot \frac{U_{CPU_{def}}}{100} \right) \cdot PUE \end{aligned} \quad (3.28)$$

Finally, the potential energy efficiency of the infrastructure upon a VM deployment to a yet unknown destination physical host can be calculated using the previously obtained formulas and combining them as shown in Equation 3.29.

$$\widehat{EnEf}_{infr} | VM_l \text{ depl.} = \frac{\widehat{Perf}_{infr} | VM_l \text{ depl.}}{\widehat{Pwr}_{infr} | VM_l \text{ depl.}} \quad (3.29)$$

3.3.3 VM Deployment (known placement)

Either as part of a service deployment or because of a VM replication due to a high resource demand, the provider's placement algorithm needs to know which will be the energy efficiency impact at infrastructure level of deploying a new VM into a given destination node.

As in Equation 3.27, the total computing performance delivered by the infrastructure will correspond to the one which would be delivered if no changes were made plus the extra performance delivered to the newly created VM. In this case, given that the destination node is known, a more precise calculation of the additional delivered performance can be made.

As commented before, the CPU utilisation of a given node ranges from 0 to 100 regardless of its number of CPUs. Therefore, if a single CPU of the destination node is occupied at 70%, this will cause an amount of CPU utilisation in the whole node of $\frac{70\%}{CPU_{node_{dst}}}$. Assuming an initial default CPU utilisation in all of the CPUs occupied by the VM of $U_{CPU_{def}}$, the extra CPU utilisation incurred in the destination host by each of the CPUs of the VM will be $\frac{U_{CPU_{def}}}{CPU_{node_{dst}}}$. If the VM has CPU_{VM_l} CPUs, then the additional CPU utilisation caused by the VM in the destination host will be $\frac{CPU_{VM_l}}{CPU_{node_{dst}}} \cdot U_{CPU_{def}}$.

The additional performance delivered can then be calculated by multiplying the normalized CPU utilisation incurred by the VM in the destination node by the maximum performance which can be delivered by it, as depicted in the second term of Equation 3.30. As previously stated, the first term of the formula simply estimates the natural evolution of the delivered performance in the infrastructure when no changes are present.

$$\widehat{Perf}_{infr} | VM_l depl. = \sum_{i \in nodes} \widehat{Perf}_{node_i} + \frac{CPU_{VM_l}}{CPU_{node_{dst}}} \cdot \frac{U_{CPU_{def}}}{100} \cdot Perf_{max_{dst}} \quad (3.30)$$

The power consumption of all the nodes of the infrastructure excluding the destination node will follow its natural trend, independently of the VM deployment. Only the trend in the power consumption of the destination node will be affected. To forecast it, the offline power model described in Chapter 5 is used, which only needs to know the forecasted CPU utilisation in the host as an input. In this case, it will consist of the future CPU utilisation of the privileged domain (Domain-0) and those of the already running VMs (which will follow its original trend), plus the additional CPU utilisation caused by the new VM. This last term is calculated as in Equation 3.30.

$$\widehat{Pwr}_{infr} | VM_l \text{ depl.} = \sum_{\substack{i \in nodes \\ i \neq dst}} \widehat{Pwr}_{node_i} + \widehat{R}_{dst} \left(\widehat{U}_0^{dst} + \sum_{j \in VMs_{dst}} \widehat{U}_{VM_j^{dst}} + \frac{CPU_{VM_l}}{CPU_{node_{dst}}} \cdot U_{CPU_{def}} \right) \cdot PUE \quad (3.31)$$

Finally, the energy efficiency of the provider's infrastructure if a VM l is deployed into a given destination node can be calculated as in Equation 3.32.

$$\widehat{EnEf}_{infr} | VM_l \text{ depl.} = \frac{\widehat{Perf}_{infr} | VM_l \text{ depl.}}{\widehat{Pwr}_{infr} | VM_l \text{ depl.}} \quad (3.32)$$

3.3.4 VM Cancellation

In this section, our aim is to determine the potential energy efficiency at infrastructure level for the hypothetical case in which a VM l that is being executed at node src is cancelled (shut down).

When a VM l is cancelled, it will cease to ask for CPU time to the node it is being run (src). So, the forecasted useful performance delivered by node src will have to exclude VM l 's CPU utilisation from the original calculation in Equation 3.8. The rest of nodes of the infrastructure will not suffer any modification in this sense, as the number of VMs they host will remain constant. This reasoning is summarized in Equation 3.33.

$$\widehat{Perf}_{infr} | VM_l \text{ canc.} = \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Perf}_{node_i} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \frac{\widehat{U}_{VM_j^{src}}}{100} \cdot Perf_{max_{src}} \quad (3.33)$$

Similarly, the expected power consumption of the nodes of the infrastructure will follow Equation 3.9 except for node src . For this particular node, its expected power consumption can be induced using the same reasoning as in Equation 3.9, but excluding VM l 's CPU utilisation from the calculation.

$$\widehat{Pwr}_{infr} | VM_l \text{ canc.} = \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Pwr}_{node_i} + \widehat{R}_{src} \left(\widehat{U}_0^{src} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \widehat{U}_{VM_j^{src}} \right) \cdot PUE \quad (3.34)$$

Taking into consideration the previous results, the expected energy efficiency of the infrastructure if a VM l is cancelled can be calculated as in Equation 3.35.

$$\widehat{EnEf}_{inf_r} \mid VM_l \text{ canc.} = \frac{\widehat{Perf}_{inf_r} \mid VM_l \text{ canc.}}{\widehat{Pwr}_{inf_r} \mid VM_l \text{ canc.}} \quad (3.35)$$

3.3.5 VM Migration (unknown placement)

Energy efficiency forecasts at infrastructure level when VMs are migrated between nodes are needed by placement optimization scheduling policies. For instance, the placement algorithm might be interested in consolidating a VM which is being executed alone in a server. If that VM is migrated, the source node can be turned off in order to avoid consuming unnecessary idle power. Another possible situation can be when a VM is demanding more resources than the ones which its hosting node can provide. In this case, it becomes necessary to migrate this VM to a node with more resources.

Initially, a rough estimation of the impact of this migration can be done by estimating it without selecting a specific destination node for the VM, as it will be studied in this section. Based on this initial evaluation, if the placement algorithm considers that it is worth performing this migration, it will then study which is the best node to receive the migrated VM by performing the calculations described in Section 3.3.6 for each potential destination node.

When a VM is migrated, its performance requirement trend is not affected, as migration is transparent to the applications being executed inside of the VM. Therefore, regardless to which node the VM is migrated into and given that its future performance will not be affected by this change, the expected total performance being delivered by the infrastructure will remain as in the nominator of Equation 3.12.

A VM migration can be understood as a VM which ceases to be executed in its original node *src* (VM cancellation) and starts to be executed in another node (VM deployment), in terms of power consumption. This reasoning is followed in Equation 3.36, where the two first terms are the same as in Equation 3.34 (representing a VM cancellation in its source node), and the third term represents the additional energy consumption incurred by the VM once migrated. This third term is calculated following the same reasoning as in Equation 3.23: the power consumption of the VM is estimated by multiplying its number of CPUs by the mean power consumption per CPU of the datacenter (see Section 3.3.2 for more details). Note that in this case, as no additional CPUs will be allocated (because the VM is being migrated), the total idle power consumption is considered to be distributed among the already allocated CPUs in the datacenter.

$$\begin{aligned}
\widehat{Pwr}_{infr} \mid VM_l migr_{UnkP} = & \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Pwr}_{node_i} + \left(\widehat{R}_{src} \left(\widehat{U}_0^{src} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \widehat{U}_{VM_j^{src}} \right) + \right. \\
& \left. + CPU_{VM_l} \cdot \left(\frac{\sum_{i \in nodes} Pwr_{min_i}}{\sum_{i \in nodes} CPU_{alloc_i}} + \overline{Pwr}_{incr} \cdot \frac{U_{CPU_{def}}}{100} \right) \right) \cdot PUE
\end{aligned} \tag{3.36}$$

Following the general formula of energy efficiency described in Equation 3.1 and taking into consideration the aspects described in this section, the forecasted energy efficiency at infrastructure level for a potential VM l migration is presented in Equation 3.37.

$$\widehat{EnEf}_{infr} \mid VM_l migr_{UnkP} = \frac{\widehat{Perf}_{infr}}{\widehat{Pwr}_{infr} \mid VM_l migr_{UnkP}} \tag{3.37}$$

3.3.6 VM Migration (known placement)

As it was exposed in Section 3.3.5, once the placement algorithm of an IaaS provider determines that performing a VM migration to an undefined node can help meeting the business level objectives of that provider, it will then evaluate in detail to which specific node this VM should be migrated.

It has already been stated that the required computing power of a VM will be the same regardless of the node where it is running. As explained previously, it will correspond to the maximum computing power which can be delivered by the node it is running in (node A) multiplied by the CPU utilisation (normalized to 1) of the VM in that node. The VM will require the same computing power when running in another node (node B). As the maximum computing power of node B is a constant value, this implies that the CPU utilisation in node B is the only factor which will be modified in order to provide the same number of CUs to the VM in both cases. This reasoning is summarized in Equation 3.38.

$$\begin{aligned}
Perf_{VM_j} \mid j \in VMs_A = Perf_{VM_j} \mid j \in VMs_B \implies \\
\implies Perf_{node_A} \cdot \frac{U_{VM_j^A}}{100} = Perf_{node_B} \cdot \frac{U_{VM_j^B}}{100}
\end{aligned} \tag{3.38}$$

If a VM is migrated into a given node and taking into account Equation 3.38, the forecasted CPU utilisation from the source node need to be scaled as depicted

in Equation 3.39 in order to estimate the CPU utilisation which will require in the destination node.

$$\hat{U}_{VM_j^{dst}} = \left(\frac{Perf_{max_{src}}}{Perf_{max_{dst}}} \right) \cdot U_{VM_j^{src}} \quad (3.39)$$

The future delivered infrastructure performance will not be affected by the fact that a VM is migrated, because the VM performance requirement is independent of the node where it is being executed, as explained in the previous section. Therefore, the only thing to be determined is how a VM migration affects to the overall power consumption in the infrastructure. With this information, it will be possible to determine the energy efficiency at infrastructure level for this potential action.

Equation 3.40 describes the overall power consumption of the infrastructure when a VM is migrated to a known destination node. In this case, the only two nodes which will be affected in terms of power consumption trends will be nodes *src* (source) and *dst* (destination). The rest of nodes' power consumption is estimated in the first term of the formula. For the source node, its real power consumption is estimated using the offline power model described in Chapter 5. The CPU utilisations of the Domain-0 and the VMs running in that node are used as an input for the model, except that of VM *l*'s, as it will stop being executed in this node. A similar approach is followed in the destination node, but in this case the additional CPU utilisation incurred by VM *l* (calculated using Equation 3.39) is also considered when applying the offline power model to estimate the future power consumption. Note that both the source and destination node power consumptions need to be multiplied by the datacenter PUE in order to consider the extra power overhead needed to feed the cooling systems, management infrastructure, etc.

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l migr_{KP} = & \sum_{\substack{i \in nodes \\ i \neq src \\ i \neq dst}} \widehat{Pwr}_{node_i} + \left(\hat{R}_{src} \left(\hat{U}_0^{src} + \sum_{\substack{j \in VM_{src} \\ j \neq l}} \hat{U}_{VM_j^{src}} \right) + \right. \\ & \left. + \hat{R}_{dst} \left(\hat{U}_0^{dst} + \sum_{j \in VM_{dst}} \hat{U}_{VM_j^{dst}} + \hat{U}_{VM_l^{dst}} \right) \right) \cdot PUE \end{aligned} \quad (3.40)$$

Equation 3.41 evaluates the forecasted energy efficiency at infrastructure level for a potential VM *l* migration to a specific destination node.

$$\widehat{EnEf}_{infr} \mid VM_l migr_{KP} = \frac{\widehat{Perf}_{infr}}{\widehat{Pwr}_{infr} \mid VM_l migr_{KP}} \quad (3.41)$$

Chapter 4

CPU Estimator

To forecast the eco-efficiency at the different levels described in this document, the CPU utilizations of the VMs running in the infrastructure need to be forecasted separately, as well as that of the privileged domain of each host (Domain-0 in the Xen nomenclature). To this end, a general variable estimator has been developed. It relies on forecasts provided by four different predictors and selects one of them appropriately to obtain the “best forecast” of a given variable. The overall design of the variable estimator is presented in Figure 4.1.

As it can be observed in the figure, the variable estimator consists in four different predictors: the moving average, the exponential smoothing, the linear regression and the double exponential smoothing predictors. Further information about these predictors can be found in [29]. All these predictors individually take the past and present observed values of the variable to predict (from time $t - n$ until t) in its input and provide a forecast of its expected value in time $t + \tau$ in its output, where τ is the time difference between the current time and the future instant to predict. Note that each estimator can be configured with one or more configuration parameters. In the experimentation section, these parameters will be appropriately tuned in order to obtain a better precision when forecasting the CPU utilization of a given virtual machine.

The variable estimator provides two interfaces: the *addValue(t, value)* and the *obtainBestForecast(t+ τ)* methods. When the *addValue(t, value)* method is invoked, a new sample of the variable is inserted at the last position of the *realValues* array, corresponding to time t . The already existing samples are shifted to the left. If the *realValues* array’s size was already n (a discussion of the value of n is provided in short), the oldest sample is simply discarded.

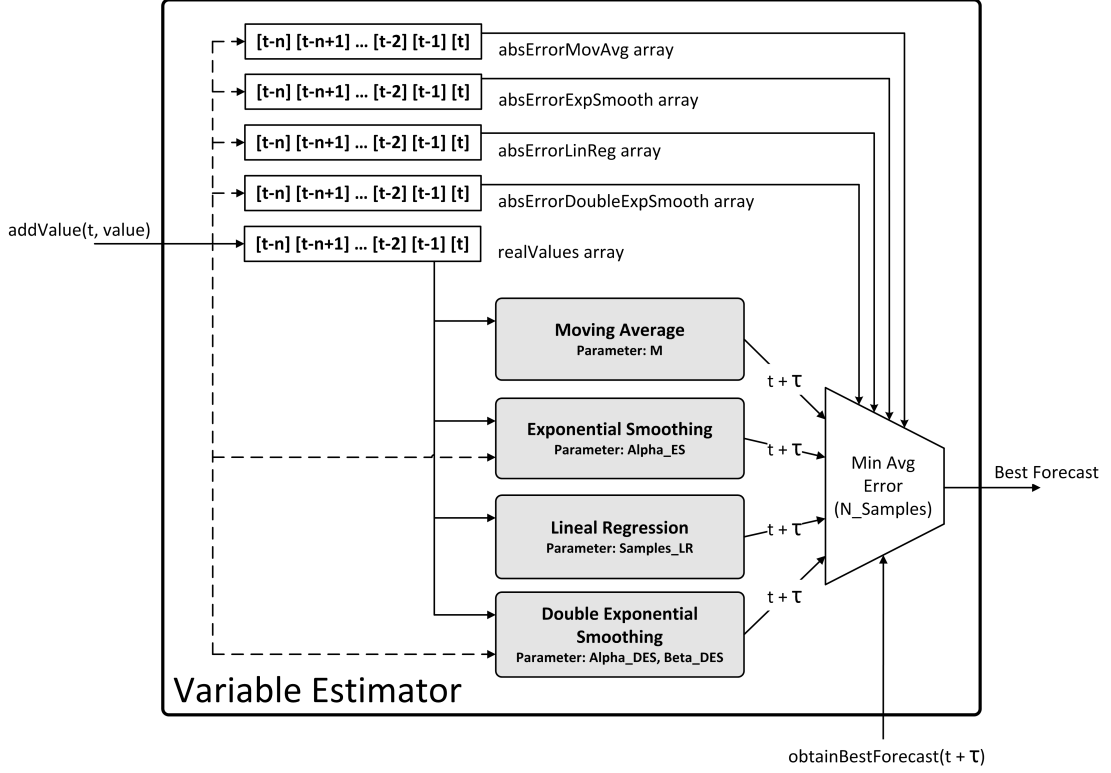


Figure 4.1: Variable estimator overall design

In addition, *addValue* also updates the absolute error arrays (also of size n) of the different estimators. In particular, it calculates the absolute error that each estimator would have if the sample at the current time (time= t) was predicted using the samples of the *realValues* array until time= $t-1$. This calculated value is inserted in the last position of each absolute error array, displacing the previous absolute errors to the left and discarding the oldest one if the array's size was already n .

Finally, *addValue* also updates the state of the exponential and the double exponential smoothing predictors taking into consideration the newly inserted value.

The *obtainBestForecast(t+τ)* method firstly evaluates, for all the predictors, which has had the minimum mean absolute error during the last $N_Samples$ predictions. To do this, it takes all the absolute error arrays of all the predictors and calculates its average values over the last $N_Samples$ samples for each of them. Depending on the result, it selects the forecast provided by the predictor with the smallest mean average absolute error over the last $N_Samples$ as the “Best Forecast” for time $t+τ$.

The value of $N_Samples$ is configurable and might affect to the precision of the predictor. In the experimentation section, the value of $N_Samples$ which maximizes the precision of the variable estimator when predicting the CPU utilization of a VM will be determined.

All the absolute error and real values arrays have a size of n samples. The value of n has to be at least $Samples_LR$ for the Lineal Regression predictor to be able to work correctly, and it also has to be at least equal to $N_Samples$ so the last $N_Samples$ samples of the absolute error can be averaged when selecting the predictor with the smallest mean average error. To simplify the implementation of the variable estimator, the same size n has been chosen both for the real values and the absolute error arrays. Therefore, n should be at least $\max(Samples_LR, N_Samples)$.

Chapter 5

Off-line Power Model

When doing forecasts of the energy efficiency at any of the levels described in this document, the need of being able to predict which will be the power consumption of a given physical host arises. In [9], it is observed that the main contributors to the overall dynamic power consumption of a physical host are the CPU and the memory. Given that the activity levels of the rest of the components correlate well with the CPU activity, this publication suggests a linear and a non-linear model to approximate the power consumption based on the CPU utilization. While the non-linear model approximates the relationship between the power consumption and the CPU utilization with less error, the linear model still has a good precision, it is simpler to implement and provides a better runtime performance. The linear model has been the one implemented in this work to perform the power consumption forecasts.

An off-line power model has been constructed for each of the nodes of the infrastructure. To generate it, one VM has been deployed in each of the hosts, pinned to all the hosts' CPUs. Different CPU utilization levels have been induced in these VMs, ranging from 0% to 100% in small steps, thus making the CPU utilization of its hosting node to vary equally. The *stress* workload generator [30] and the *cpulimit* tool [31] have been used together for this purpose. Note that, according to the adopted notation, a CPU utilization of 100% corresponds to having all the CPUs of the node occupied at 100%.

These variable workloads allowed to perform multiple real power measurements for different CPU utilizations for all the nodes. As in [9], the obtained results, shown in Figure 5.1, confirm that the power consumed by a node closely follows a linear tendency with respect to its CPU utilization. Nevertheless, it can be observed that in all of them there is a small variability of the power consumed for a given CPU load. This is due to the power consumption of other components, like memory, network devices or hard disks, as described in [32]. However, the obtained linear model is precise enough for the objectives of this thesis.

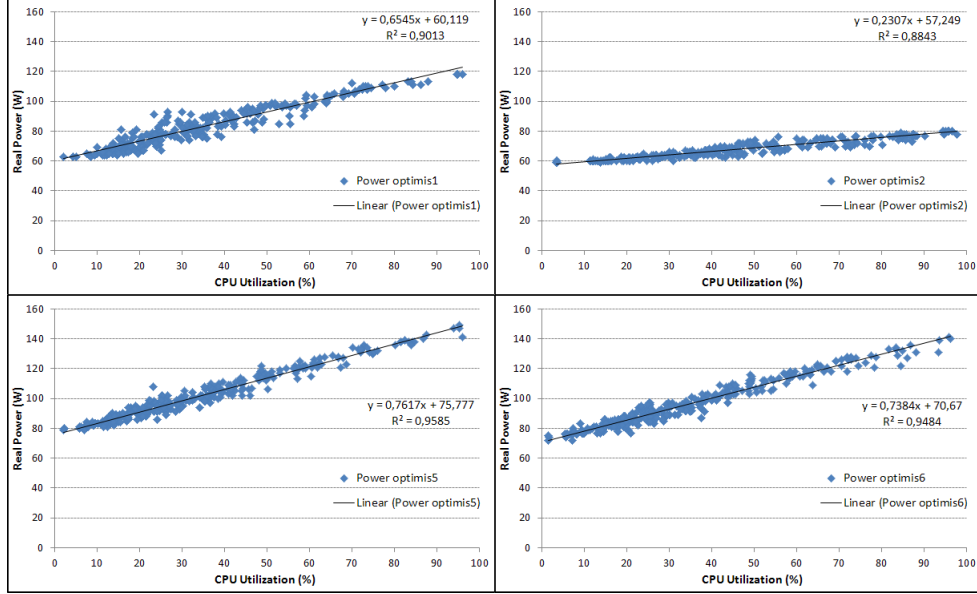


Figure 5.1: Real power consumption incurred by variable CPU utilizations

The off-line linear power model is then computed by calculating the regression line (least squares) of all the captured points for each node. This results in a power model for each node of the form $\hat{R} = a \cdot U_{CPU_i} + b$, which can then be used to estimate the future power consumptions for a given CPU load. Being Pwr_{min_i} the value of the regression line when the CPU utilization is 0% and Pwr_{max_i} the one at 100%, the power model in Equation 5.1 calculates the estimated power consumption for a CPU utilization of x .

$$\hat{R}_i(x) = \frac{Pwr_{max_i} - Pwr_{min_i}}{100} \cdot x + Pwr_{min_i} \quad (5.1)$$

The horizontal axis of the previously obtained model can be seamlessly changed by an axis representing the average number of occupied CPUs of the node. For instance, when the CPU utilization of a 8-processor node is 50%, on average 4 CPUs are being used, while if the utilization is 100%, the 8 CPUs of the node are being completely used. A graphical representation of this alternative approach is presented in Figure 5.2.

From this alternative approach, the additional power consumption of the node when one of its inactive CPUs becomes active and loaded at 100% can be calculated as in Equation 5.2 (see the graphical representation of Pwr_{incr_i} in Figure 5.2 for further clarification).

$$Pwr_{incr_i} = \frac{Pwr_{max_i} - Pwr_{min_i}}{CPU_{node_i}} \quad (5.2)$$

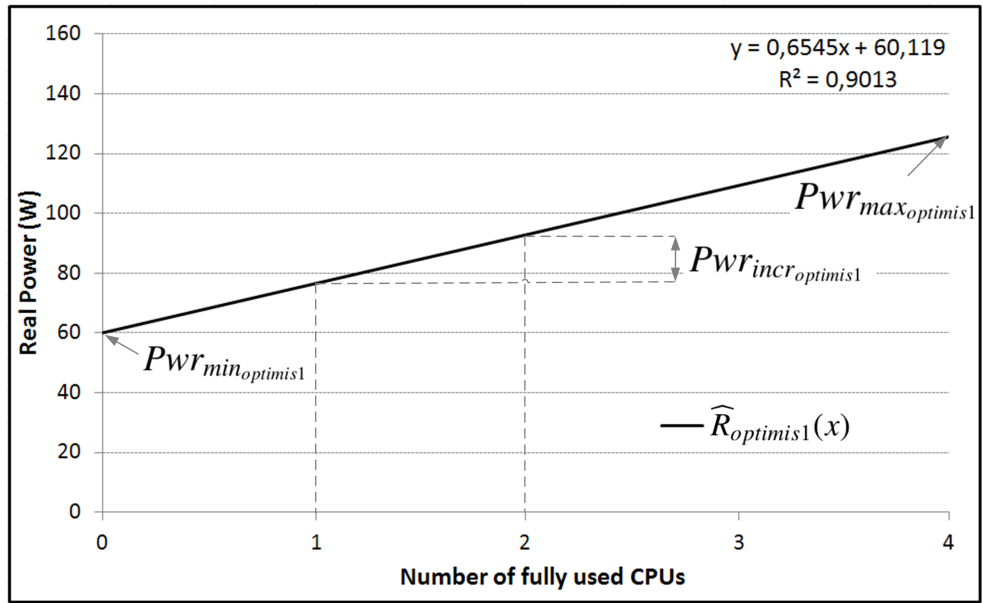


Figure 5.2: Real power consumption based on fully-occupied CPUs for node "optimis1"

Chapter 6

Experimentation and Validation

Many mathematical models have been deduced in the context of this thesis. Of all of them, it was considered that the assessment and forecast of the energy efficiency of the whole infrastructure was the most important one, because it is the factor that IaaS providers want to optimise and it is also the basis of many of the forecasts for potential actions described in Section 3.3.

Therefore, the objective of the conducted experimentation has been to evaluate and improve the precision of the forecasts of the energy efficiency of the infrastructure when running different workload types. To this end, the parameters of the variable estimators used to forecast the CPU utilisation of all the VMs running in the infrastructure were appropriately tuned for each workload type.

In Sections 6.1, 6.2 and 6.3, the respective procedures followed to generate realistic web, batch and mixed workloads are explained in detail. Then, in Sections 6.4, 6.5 and 6.6, the Ecoefficiency Tool precision is evaluated against these different workloads, its predictor parameters are appropriately tuned and the obtained precision results are presented.

6.1 Methodology to generate web workload

A realistic web workload has been generated based on the SPECweb2005 benchmark, which is described in Section 6.1.1. The servers attending this workload need to be encapsulated in virtual machines in order to be considered as “useful work” by the infrastructure provider. The details of this encapsulation can be found in Section 6.1.2. The number of concurrent clients accessing the web servers at a given time follows the real access pattern of the Barcelona School of Informatics web page¹ of the 5th and 6th of April 2010. The way the final

¹<http://www.fib.upc.edu>

workload has been generated, taking into account the aforementioned web traces, can be found in Section 6.1.3.

6.1.1 SPECweb2005 Overview

SPECweb2005 is a benchmark developed by the Standard Performance Evaluation Corporation (SPEC), widely used to evaluate the performance of a system acting as a web server servicing static and dynamic page requests. It is able to simulate three different scenarios: a banking site, an e-commerce site and a support site, based on the client's behavior when accessing different web pages stored in a web server and the responses it gets.

In the conducted experiments, both the e-commerce as well as the support site have been used. In the e-commerce site, the clients visit a shopping site, browse its catalog and "purchase" products, using both encrypted (about 5%) and unencrypted data. In the vendor support site, the clients are able to look at a set of pages and download files from them, such as manuals or software, using unencrypted connections.

The benchmark has 4 component types:

- The *Clients* build connections to the web server, send page requests and receive the answers from the web server. There can be more than one Client component accessing the web server during a simulation, each of them generating a given number of simultaneous sessions.
- The *Web server* or SUT (System Under Test) receives and processes the requests coming from the clients. For some of the transactions, it needs to access the Back-end SIMulator.
- The *Back-end SIMulator* (BeSIM) simulates the database and application parts of the application.
- The *Prime Client* is the component responsible of initializing all the aforementioned components, monitors the execution of the benchmark and collects and evaluates the results once the benchmark completes.

The Prime Client allows to configure many aspects of the simulation. Nevertheless, in this document only the ones relevant to the conducted experiments will be detailed. Although each benchmark run can perform multiple iterations of a given experiment, only a single iteration of it is performed. The reason for this is provided in the following Section. Four configurable parameters are of special interest (see Figure 6.1):

- `THREAD_RAMPUP_SECONDS`: time during which new user sessions are started in an increasing manner.

-
- `WARMUP_SECONDS`: time before the actual measurement of the benchmark takes place, during which the system prepares its caches.
 - `RUN_SECONDS`: time during which the benchmark is run and results are recorded.
 - `THREAD_RAMPDOWN_SECONDS`: time during which the user sessions are stopped in a decreasing manner.

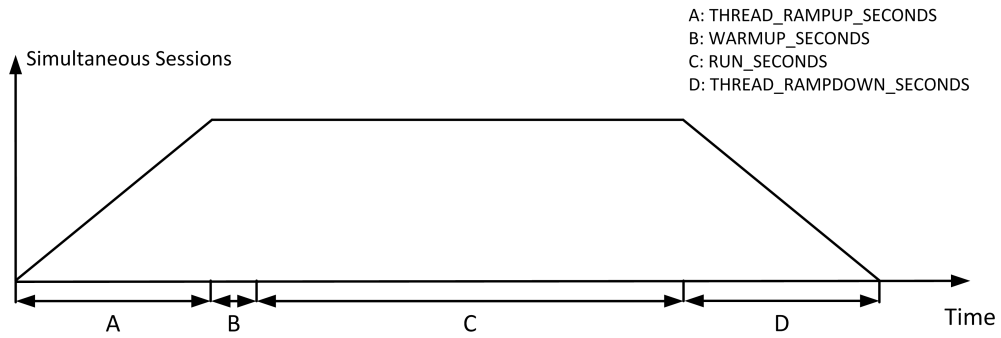


Figure 6.1: SPECweb2005 controlled parameters during the experiment

The benchmark measures, for each iteration, the percentage of requests served with a response time lower than 2 seconds (QoS of GOOD), lower than 4 seconds (QoS of TOLERABLE) and greater than 4 seconds (QoS of FAIL).

6.1.2 Testing environment

The SPECweb2005 benchmark components have been encapsulated in three different Virtual Machine types:

- The *Webserver VM* contains the logic of the *Web server (SUT)* component of the SPECweb2005 architecture.
- The *BeSIM VM* contains the logic of the *Back-end SIMulator* component of the SPECweb2005 architecture.
- The *Client VM* contains the logic of both the *Client* and *Prime Client* components of the SPECweb2005 architecture. In particular, the *Prime Client* encapsulated in this VM is only able to control the *Client* which is also encapsulated in it.

This type of encapsulation allows to create multiple *Client VMs*, all of them interacting with the same *Web server* component. Although in the original SPECweb2005 architecture there is only one *Prime Client* controlling all the

Clients, using this approach together with VM contextualization the number of simultaneous sessions that each *Client VM* generates can be independently established. Moreover, the Webserver and BeSIM VM web servers have been configured following the tune up recommendations published in [33]. The characteristics of the created VMs can be found on Table 6.1.

VM type	Webserver VM	BeSIM VM	Client VM
Number of cores	4	2	1
RAM Memory	6 GB	3 GB	1 GB
Operating System	Debian 6.0	Debian 6.0	Debian 6.0

Table 6.1: SPECweb2005 VM characteristics

One *Webserver VM* and one *BeSIM VM* are hosted in the Cloud provider per each deployed service. That is, a *Webserver VM* and a *BeSIM VM* are needed to provide the required service to access an e-commerce website, while another set of *Webserver VM* and *BeSIM VM* need to be deployed to provide access to a support website. These four VMs will cause a realistic load in the infrastructure provider resources, which will be measured in order to assess the energy efficiency of the provider’s infrastructure.

The *Client VMs* for each scenario will be deployed in an external set of nodes, and will be the ones generating load to the system. The EMOTIVE middleware has been installed in these nodes, and will be directly used by the experiment application to create and destroy VMs in each iteration of the experiment. The necessary configuration parameters of each VM (see Section 6.1.3) will be set in each VM OVF description and passed to the VM by means of VM contextualization. Note that this EMOTIVE installation is external to the OPTIMIS environment, as it is directly used by the experiment application without interacting with any OPTIMIS component.

The VM distribution of the conducted experiment is presented in Figure 6.2. Note that only the communications of the *Client VM#1* (of each scenario) with its corresponding *Webserver VM* and *BeSIM VM* are depicted in the picture, to obtain a better visualization. The *Client* SPECweb2005 component inside of each *Client VM* directly interacts with the *Webserver VM* (solid arrow), while the *Prime Client* in each *Client VM* interacts with both the *Webserver VM* and *BeSIM VM* to configure them (dashed arrows).

The physical nodes where the aforementioned VMs were deployed have the characteristics described in Table 6.2.

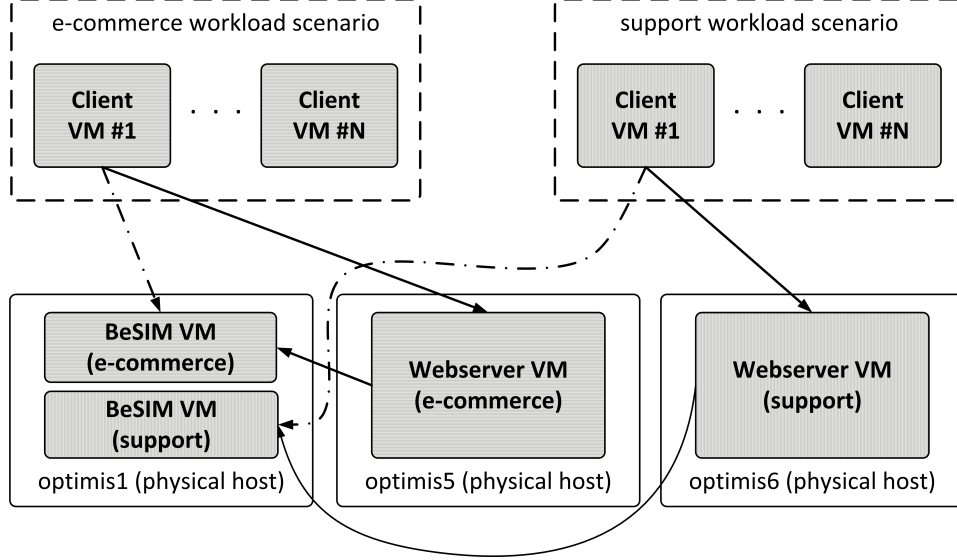


Figure 6.2: Virtual Machine distribution (web workload experimentation)

6.1.3 Workload generation

As discussed before, SPECweb2005 benchmark has been chosen since it mimics the behaviour of real Internet users under different scenarios. In particular, the e-commerce and support scenarios described above have been used. To run the benchmark, several parameters need to be configured, being the number of simultaneous sessions the main one influencing in the web server's resource utilisation. By default, this parameter is constant during the complete execution of the benchmark (reached once `THREAD_RAMPUP_SECONDS` seconds have passed, see Figure 6.1). However, to really mimic the workload of real Internet sites, this parameter needs to be modified along the execution of the experiment.

In order to have a variable number of simultaneous sessions along the execution of the experiment, the SPECweb2005 benchmark is executed multiple times (for each of the scenarios), in a fashion as the one displayed in Figure 6.3. As it can be observed, an independent SPECweb2005 benchmark is launched in each iteration and its execution starts precisely when the previous benchmark enters the `THREAD_RAMPDOWN_SECONDS` phase (see Figure 6.1). Thus, while the previous benchmark run starts reducing the number of client threads connected to the web server, the current benchmark run starts increasing them. Therefore, the global number of simultaneous sessions changes smoothly from one value to the other across different iterations.

	optimis1	optimis2
Number of processors	4	2
Processor type	Intel Xeon X3430	Intel Pentium G6950
Processor frequency (GHz)	2.40	2.80
RAM Memory (GB)	10	8
Operating System	Debian 6.0.7	Debian 6.0.7
Hypervisor	Xen 4.0.1	Xen 4.0.1
Pwr_{min} (W)	60.119	57.249
Pwr_{max} (W)	125.569	57.249
$Perf_{max}$ (MWIPS)	2171.6	1215.4
Deployed VMs	BeSIM VM (e-commerce) BeSIM VM (support)	$\langle none \rangle$
	optimis5	optimis6
Number of processors	4	4
Processor type	Intel Xeon X3330	Intel Xeon X3330
Processor frequency (GHz)	2.66	2.66
RAM Memory (GB)	7	7
Operating System	Debian 6.0.7	Debian 6.0.7
Hypervisor	Xen 4.0.1	Xen 4.0.1
Pwr_{min} (W)	75.777	70.67
Pwr_{max} (W)	151.947	144.51
$Perf_{max}$ (MWIPS)	2173.96	2179.84
Deployed VMs	Webserver VM (e-commerce)	Webserver VM (support)

Table 6.2: Node characteristics

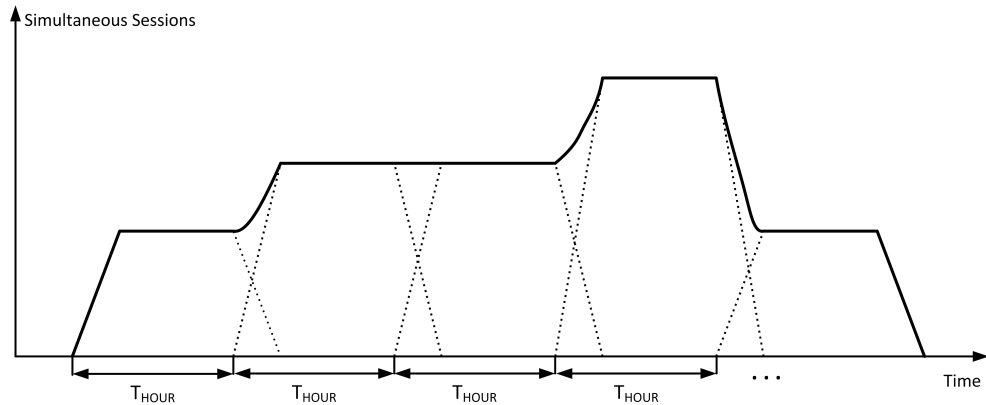


Figure 6.3: Variable web workload generation by overlapping individual SPECweb2005 benchmark executions

The web traces of the Barcelona School of Informatics (FIB) web page¹ have been used as a reference to set the number of simultaneous sessions at a given time during the experiments. From these web traces, the number of received requests per hour were first derived. Then, since the number of simultaneous sessions during an hour is proportional to the number of requests received during that hour, the FIB web traces have been used to set the number of simultaneous sessions along the experiments. The simultaneous sessions from Monday 5th to Tuesday 6th April 2010 have been simulated in the experiments. It has been assumed that both the e-commerce and support workloads will follow the same trend, as they are Web workloads too, but scaled by a different proportionality factor.

Note that the simulated time is different from the execution time of the experiment. The FIB web traces are used to determine the number of simultaneous sessions during a simulated hour (corresponding to the different hours captured during 5th and 6th April 2010 in the web traces). This hour is simulated during T_{HOURL} seconds of the experiment execution ($T_{HOURL} = 480s = 8min$ in the conducted experimentation). After T_{HOURL} seconds, the experiment changes the number of simultaneous sessions based on the next value of received requests per hour of the FIB web traces. Also observe that if a forecast is performed (either CPU or energy efficiency forecast) for 1 minute (of the experiment execution time) ahead in the future, this actually represents $\frac{60}{8} = 7.5$ minutes of the simulated time.

In the conducted experiments, the e-commerce workload is reduced by a factor of 55 with regards to the one provided by FIB, while the support workload's reduction factor is 110. These values have been chosen by evaluating the obtained QoS (in a separate benchmark execution) for the maximum amount of simultaneous sessions that occur for these particular reduction factors during the full experiment execution. In particular, the maximum number of simultaneous sessions are of about 800 for the e-commerce scenario and 400 for the support one, providing a GOOD QoS for more than 90% and 60% of the requests in the conducted separate benchmarks, respectively. When choosing these values, it has been considered that the e-commerce client browsing experience has to be "almost perfect" for the client to be satisfied, while having to wait for a longer time to download a particular file is not so critical for the client satisfaction.

Each *Client VM* can cope with a limited number of simultaneous sessions without becoming overloaded. In the conducted experiments, it was determined that a maximum of 200 simultaneous sessions could be handled safely by a *Client VM* with the specifications provided in Table 6.1. So, in each iteration of the experiment, more than one *Client VM* may need to be created in order to generate

¹<http://www.fib.upc.edu>

the desired simultaneous sessions for that period.

Given that the VM creation is sequential, this causes that the *Client VMs* have a different amount of time to execute the benchmark run within an “hour period”. Note that “hour period” refers to an hour of the FIB web traces, but it just corresponds to several minutes in the experiment execution. In order to reach the maximum number of simultaneous sessions at the same moment in all the *Client VMs* in a given iteration, the `THREAD_RAMPUP_SECONDS` is modified in each of them when they are started, as depicted in Figure 6.4. So, the `THREAD_RAMPUP_SECONDS` in each VM is reduced depending on the time spent to create the previous VMs of the iteration. Note that it cannot be reduced indefinitely. If it is detected that the necessary `THREAD_RAMPUP_SECONDS` is going to be less than a certain safety time (in the experiments, 20 seconds were considered), the execution time ($T_{WARMUP} + T_{RUN}$) is reduced in order to assure a minimum `THREAD_RAMPUP_SECONDS`.

As it can be observed in Figure 6.4, the effect of starting multiple *Client VMs* in each iteration causes that the increasing of simultaneous sessions is not as linear as it would be if only a single *Client VM* was started. This could cause some declines in the number of simultaneous sessions at hour transitions, if the approach presented in Figure 6.3 is followed. To mitigate this problem, the execution time is slightly extended, causing a greater overlap between the workload trapezoids.

6.2 Methodology to generate batch workload

To generate a realistic batch workload, the COMP Superscalar (COMPSs) Programming Model has been used. In Section 6.2.1, a brief overview of the features of COMPSs is provided. In a COMPSs environment, two main entities are present: a master and several workers. The workers are actually the ones causing a high resource utilisation when processing batch tasks. So, in order to be considered as “useful work” by the infrastructure provider, the master and the workers have been encapsulated in two different VM kinds, whose details and distribution in the testbed can be found in Section 6.2.2. Although the COMPSs PM was not originally conceived to mimic a batch workload, an application which generates job arrivals (consisting on multiple tasks) according to a Poisson process makes it work as such. The description of the adopted approach to achieve it can be found in Section 6.2.3.

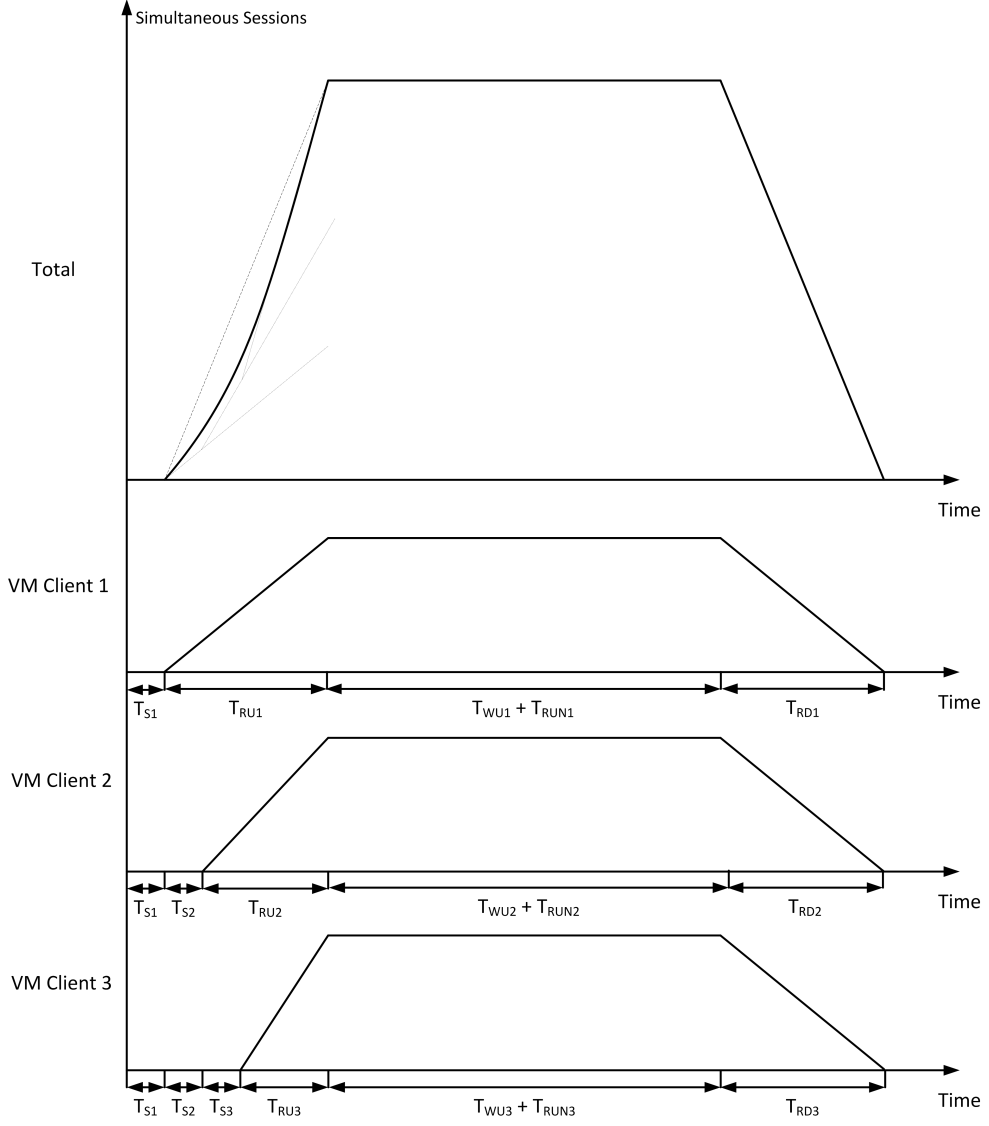


Figure 6.4: Effect of starting multiple SPECweb2005 clients within a single iteration. T_{Si} = Client i starting time (includes VM creation time). T_{RUi} = THREAD_RAMPUP_SECONDS of VM i . T_{WUi} = WARMUP_SECONDS of VM i . T_{RUNi} = RUN_SECONDS of VM i . T_{RD_i} = THREAD_RAMPDOWN_SECONDS of VM i .

6.2.1 COMP Superscalar Overview

COMP Superscalar (COMPSs) is a programming model which facilitates the development of applications that run on top of distributed infrastructures such as Clusters, Grids and Clouds. In a nutshell, the COMPSs model has three key characteristics [34]:

- Sequential programming: the model is based on sequential programming, that is, users only need to code their application as usual, using sequential coding and without caring about parallelization or distribution.
- Infrastructure unaware: the model abstracts the application from the underlying distributed infrastructure. Therefore, applications are portable between infrastructures with different characteristics.
- No APIs, but common languages: COMPSs does not require the use of any special API call, pragma or construct in the application. Everything is coded using common languages such as Python, C++ or Java.

COMPSs parallelizes an application sequential code by splitting it into parts (named tasks) which can have data dependencies among them. It is the COMPSs runtime system that exploits the inherent parallelism of such tasks at execution time. In order to leverage from such benefits, the original sequential code must be modified: the programmer must indicate which methods within the application must be considered as tasks. Then, when the code is executed the COMPSs runtime instruments the application by replacing the method calls by invocations to the runtime system. In turn, the runtime system is the responsible of finding dependencies between tasks (method invocations), scheduling their execution to the available resources and contacting with the nodes to execute them remotely and in a distributed fashion.

6.2.2 Testing environment

An application which generates a batch-like workload using the COMPSs programming model has been encapsulated in two different Virtual Machine types:

- *Master VM*: it is where the batch workload generator application and the main part of the COMPSs environment are placed. This batch workload generator simulates the arrival of job submissions to a batch scheduler, following the model described in Section 6.2.3. Such jobs are split into tasks, whose execution is distributed among the different *Worker VMs* (see below). In fact, each job submission is implemented as an invocation to a

method which has been marked as to be executed using the COMPSs programming model. Therefore, when the application starts its execution it turns on the COMPSs runtime system, which is placed in the same *Master VM*, and it repetitively executes the marked method. The COMPSs runtime system captures such method invocations and distributes the task executions among the different *Worker VMs*. When all the *Worker VMs* are busy executing previously submitted tasks, it queues the execution of any additional task. When any of these VMs finish their task execution, it triggers the execution of a new task in that VM (the first one in the queue). Therefore, using this approach the system emulates a distributed system executing batch applications.

- *Worker VMs*: they are in charge of executing the tasks issued in the *Master VM* and assigned to them. To this end, the *Master VM* establishes an SSH connection with the *Worker VM*, where a small COMPSs component is placed. This component receives a message from the *Master VM* containing the method name and the class implementing it, along with the necessary data and/or objects. Therefore, the *Worker VMs* are the ones actually causing a high resource utilisation in the Cloud infrastructure during the experimentation, given that the tasks being executed are CPU intensive.

The characteristics of the aforementioned VMs can be found in Table 6.3.

VM type	Master VM	Worker VM
Number of cores	2	2
RAM Memory	2 GB	2 GB
Operating System	Debian 5.0.8	Debian 5.0.8

Table 6.3: COMPSs VM characteristics

In the scenario where the energy efficiency precision is evaluated using batch workload, a single *Master VM* is in charge of controlling and scheduling tasks to 6 *Worker VMs*, as displayed in Figure 6.5 (the node characteristics are the same as in Table 6.2):

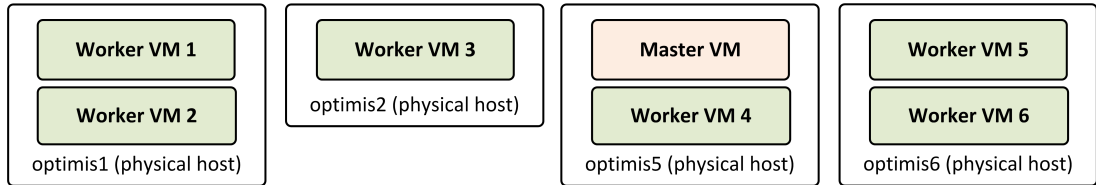


Figure 6.5: Virtual Machine distribution (batch workload experimentation)

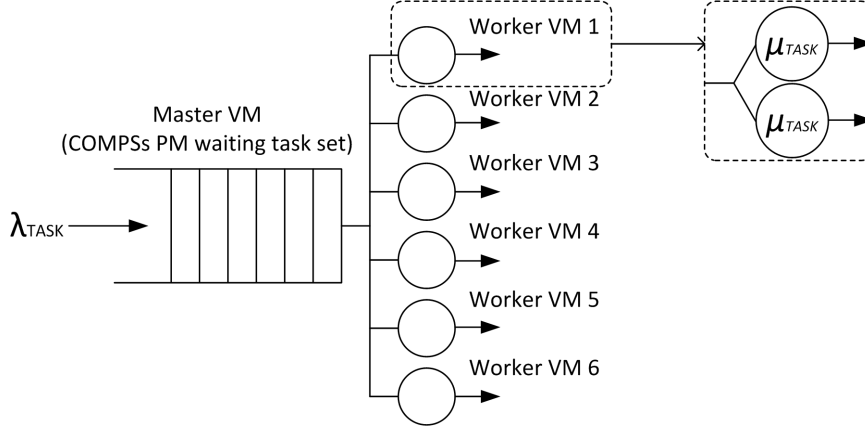


Figure 6.6: Batch workload generation using COMPSs, with $p = 2$ and $w = 6$.

6.2.3 Workload generation

The COMPSs Programming Model (COMPSs PM) has been used in order to generate a batch workload. When a new job is launched, several tasks are submitted to the COMPSs PM, which assigns its execution to the different slots present in the Worker VMs. In general, each Worker VM has p processors that can execute p tasks simultaneously and independently. When a job arrives at the COMPSs PM runtime its tasks are automatically assigned to Worker VMs with at least one empty slot, if there is any. However, if all the slots are already occupied, the unassigned tasks are retained in a *waiting task set*, until one of the slots becomes free and the COMPSs PM decides to execute one of these waiting tasks in it. Note that the waiting tasks are not necessarily picked up in FIFO order from the waiting task set. They are picked one by one and therefore executed in a batch fashion. However, for the stability analysis of this section, the waiting task set can be understood as a queue.

The application generates job arrivals according to a Poisson process with rate λ_{JOB} . Note that since each job consists of k tasks, $\lambda_{TASK} = k \cdot \lambda_{JOB}$. Assuming that the service times of each slot of the Worker VMs have an exponential distribution with parameter μ_{TASK} , the whole system can be modeled as a M/M/ c system, following the Kendall notation [35]. The system model can be observed in Figure 6.6.

In this model, the system utilisation has the formula $\rho = \frac{\lambda_{TASK}}{c \cdot \mu_{TASK}}$, where c represents the number of servers. Note that if all the Worker VMs have the same number of processors, $c = p \cdot w$, being c the number of execution slots, p the number of processors per Worker VM and w the number of Worker VMs. In this context, ρ has to be less than 1 for the queue to be stable. Therefore, for a given μ_{TASK} , the maximum λ_{TASK} that the system can cope with is:

$$\rho = \frac{\lambda_{TASK}}{c \cdot \mu_{TASK}} < 1 \Rightarrow \lambda_{TASK} < c \cdot \mu_{TASK} \Rightarrow \lambda_{TASK} < p \cdot w \cdot \mu_{TASK} \quad (6.1)$$

Considering that $\lambda_{TASK} = k \cdot \lambda_{JOB}$ and the deduction of Equation 6.1, the maximum job submission rate follows the formula of Equation 6.2.

$$\lambda_{JOB} < \frac{p \cdot w}{k} \cdot \mu_{TASK} \quad (6.2)$$

Firstly, the execution time of a job was measured in the experimentation testbed. Two independent measurements were performed, which were of 7.14 and 7.18 minutes. The mean execution time thus was of 7.16 minutes. Each task of the job was directly executed in an independent slot, without queuing. Note that for the conducted experimentation, all the executed tasks were approximately of the same length. So, the execution time in each slot to serve one task was also of 7.16 minutes. Therefore, the μ_{TASK} per execution slot was:

$$\mu_{TASK} = \frac{1}{7.16} = 0.1397 \frac{\text{servings}}{\text{minute}} \quad (6.3)$$

With this information, the maximum mean arrival rate for any number of Worker VMs can be easily calculated using the formula already deduced in Equation 6.2. For instance, during the experimentation with pure batch workload, 6 Worker VMs ($w=2$) of 2 processors each ($p=2$) were used to execute jobs consisting of 3 tasks each ($k=3$). The maximum arrival rate in this case is:

$$\lambda_{JOB_{MAX \ 6 \ Workers}} = \frac{2 \cdot 6}{3} \cdot 0.1397 = 0.5588 \frac{\text{arrivals}}{\text{minute}} \quad (6.4)$$

Having a constant job arrival date throughout a given day is not a realistic situation: users tend to submit more jobs while they are working in their offices, whereas less jobs are submitted during the night. To mimic this situation and consider a different job arrival intensity depending on the day of the week, the FIB web traces were used as a reference and processed as follows:

- Sum the number of sessions for the week from 5th to 11th April 2010. This variable is named N_{TOTAL} .
- For each day i , the number of sessions from 8am to 8pm (office hours) and from 8pm to 8am (out of office hours) were added up in two separate variables, called N_{DAY_i} and N_{NIGHT_i} .
- For each day, the workload intensity I during all the office hours is calculated as $I_{DAY_i} = \frac{N_{DAY_i}}{N_{TOTAL}}$, while for the night hours the formula is $I_{NIGHT_i} = \frac{N_{NIGHT_i}}{N_{TOTAL}}$.

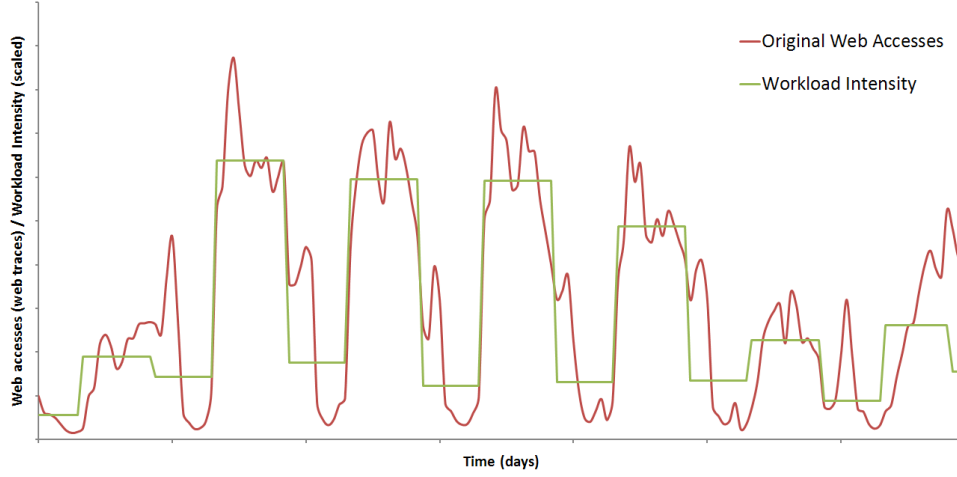


Figure 6.7: Batch workload intensity calculated for the week from 5th to 11th April 2010.

The resulting workload intensity waveform for the processed week is depicted in Figure 6.7. Note that the workload intensity has been scaled in this picture in order to appreciate its correspondence with the original web workload.

Finally, the workload intensity waveform is scaled in such a way that its mean value corresponds with the 90% of the maximum theoretical arrival rate obtained in Equation 6.4. The obtained waveform is the one which has been used to establish the arrival rate at each “hour period”. Remember that an hour period of the experiment execution (T_{HOURL}) represents an hour of the simulated experiment (refer to Section 6.1.3 for more details). Note that although during some periods the arrival rate will be greater than the theoretical maximum, this aspect will be compensated during night hours. As long as the mean arrival rate is below the theoretical maximum (the 90% reduction factor establishes an additional safety margin), the system will not become saturated. The resulting arrival rate throughout the conducted experiment in Section 6.5 (a pure batch workload with 6 workers) is depicted in Figure 6.8.

6.3 Methodology to generate mixed workload

The precision of the Ecoefficiency Tool has also been tested against a mixed workload, consisting of both web and batch workloads. The procedure to generate such workloads can be found in Sections 6.1 and 6.2, respectively.

In this scenario, both the e-commerce and support web servers were deployed, together with their respective application servers. The batch workload was gen-

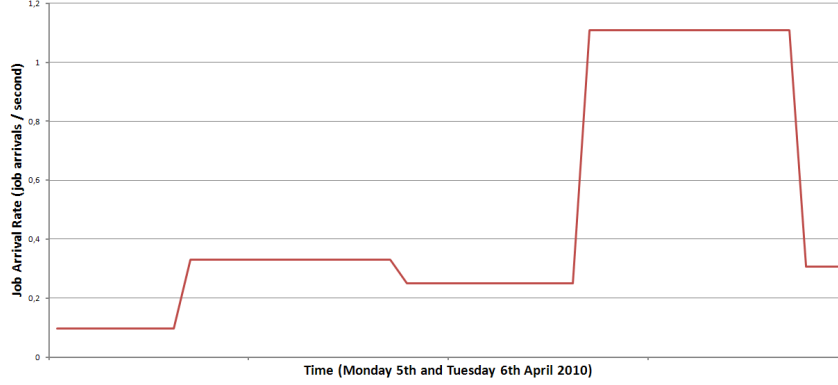


Figure 6.8: Arrival rate throughout the experiment in Section 6.5. The simulated days are April the 5th and 6th of 2010.

erated with a Master VM with 2 Worker VMs. Given that the available testbed had a limited number of CPUs and RAM memory, the two BeSIM VMs and the Master VM were consolidated in a single physical host (optimis2). Note that as each of them uses 2 CPUs, a total of 6 CPUs would be required to run them without having to share processor time between them. However, due to the fact that all of them had a very low CPU utilisation and RAM requirements, they were forced to share the 2 CPUs of the physical host between them. Their RAM memory was also slightly reduced in order for them to fit in the same node. No significant performance degradation was observed because of these actions. The final VM distribution can be observed in Figure 6.9.

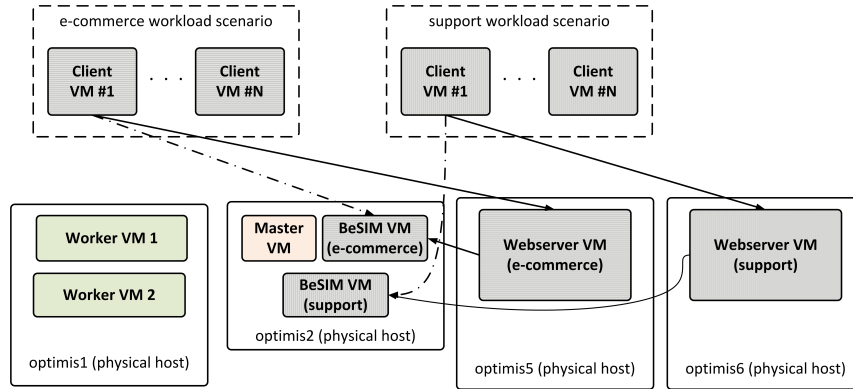


Figure 6.9: Virtual Machine distribution (mixed workload experimentation)

The maximum job arrival rate had to be recalculated for this scenario. Using Equation 6.2, the new calculated value was:

$$\lambda_{JOB_{MAX} \ 2 \ Workers} = \frac{2 \cdot 2}{3} \cdot 0.1397 = 0.1863 \frac{arrivals}{minute} \quad (6.5)$$

As in Section 6.2.3, the job arrival rate waveform was appropriately scaled so its mean value throughout the week from 5th to 11th April 2010 was equal to the 90% of the new calculated maximum value.

6.4 Model validation with web workload

In the following sections, the experiments performed to validate and improve the accuracy of the forecasts performed by the Eco-efficiency Tool when running a realistic web workload are presented. In particular, the target of this experimentation has been to improve the accuracy of the forecasts at the infrastructure level, which are the ones used by the placement algorithms of the infrastructure provider to make optimization decisions.

The two scenarios (e-commerce and support) were deployed as displayed in Figure 6.2. The conducted experiment consisted in assessing the energy efficiency (the ecological efficiency can be easily derived from it as described in Equation 3.4) of the whole infrastructure and forecasting it one minute ahead in the future. Then, after one minute the energy efficiency was evaluated again and the obtained energy efficiency assessment could be compared with the forecast which had been made in the previous step. At the same time, the CPU utilisation of each VM was also monitored, as well as the forecasted values provided by the predictors of the variable estimators (see Chapter 4) associated to each of them.

Therefore, two kinds of assessments-forecasts pairs were captured throughout the experiment: these of the overall energy efficiency of the infrastructure and these of the CPU utilisation of each VM. Note that for the CPU utilisation of a given VM, each assessment was compared with the forecasts provided by each of the individual predictors (moving average, exponential smoothing, linear regression and double exponential smoothing) as well as the “best forecast” provided by its associated variable estimator. Refer to Chapter 4 for more details.

This capturing procedure was repeated during all the experiment, and finally the obtained set of assessment and forecast samples are compared by means of the Concordance Correlation Coefficient (CCC).

The CCC, described in [36], measures the agreement between two variables and can be used to evaluate the degree of agreement among raters. In the context of this thesis, it has been used to evaluate the degree of agreement between the performed forecasts for a given time and the assessment value at that same time, by comparing their values. Note that, in fact, each assessment has been compared with the forecast which was performed in the previous iteration of the experiment.

As described in [36], the CCC between two vectors of length N can be calculated as:

$$\rho_c = \frac{2 \cdot s_{xy}}{s_x^2 + s_y^2 + (\bar{x} - \bar{y})^2} \quad (6.6)$$

Where:

$$\bar{x} = \frac{1}{N} \cdot \sum_{n=1}^N x_n \quad (6.7)$$

$$s_x^2 = \frac{1}{N} \cdot \sum_{n=1}^N (x_n - \bar{x})^2 \quad (6.8)$$

$$s_{xy} = \frac{1}{N} \cdot \sum_{n=1}^N (x_n - \bar{x}) \cdot (y_n - \bar{y}) \quad (6.9)$$

Note that in all the experiments, the energy efficiency of the infrastructure is assessed and forecasted by the Ecoefficiency Assessment Tool installed in the infrastructure provider. Nevertheless, in order to be able to determine the CPU forecasts made for each VM, by each internal predictor of its associated variable estimator (as well as the resulting “best forecast”) and for each possible configuration parameter of each predictor, an external probe was installed in the same system. This external probe received the values of the current CPU utilisations of all the running VMs, each time they were measured by the Ecoefficiency Assessment Tool. It also received a notification each time the Ecoefficiency Assessment Tool issued the forecast of the CPU utilisation of a given VM for a given time stamp in the future, therefore allowing the external probe to perform the same forecast for all the predictors of that VM, using multiple configuration parameters in each of them.

During all the experiments, the following parameters were used: $T_{HOURL} = 480s$ (8 minutes), $T_{RUNi} = 420s$ (7 minutes), $T_{RDi} = 120s$ (2 minutes), $T_{WUi} = 10s$ and $T_{RUmin} = 25s$. The same nomenclature as in Figures 6.3 and 6.4 is used.

6.4.1 Experiment 1: predictors’ parameter tuning

In this first experiment, the workload described in Section 6.1 was used. The accuracy of each internal predictor of each variable estimator (see Chapter 4) was separately evaluated for different configuration parameter values, in order to obtain the value which provides the highest precision when the CPU utilisation of a virtual machine is predicted when running a pure web workload. The CCC was used as a criteria to compare the precision provided by the different parameter values: the one providing the greatest CCC was considered to be the best one.

The CPU utilisations of the *Webserver VMs* of the e-commerce and the support scenarios were evaluated and forecasted continuously, and the CCC between the expected CPU utilisations and the real values was calculated for each of them, for each parameter value of each internal predictor. The *BeSIM VMs* were not considered in this calculation, given that they were support VMs not causing a significant resource utilisation.

The obtained results when the CPU utilisation is predicted by each of the internal predictors (moving average, exponential smoothing, linear regression and double exponential smoothing predictors) can be found in Figures 6.10, 6.11, 6.12 and 6.13, respectively. Note that these results are the averaged results of those obtained in the e-commerce and support *Webserver VMs*.¹

When the CPU utilisation is predicted using the moving average predictor (Figure 6.10), a value of $M=2$ maximises the mean obtained CCC.

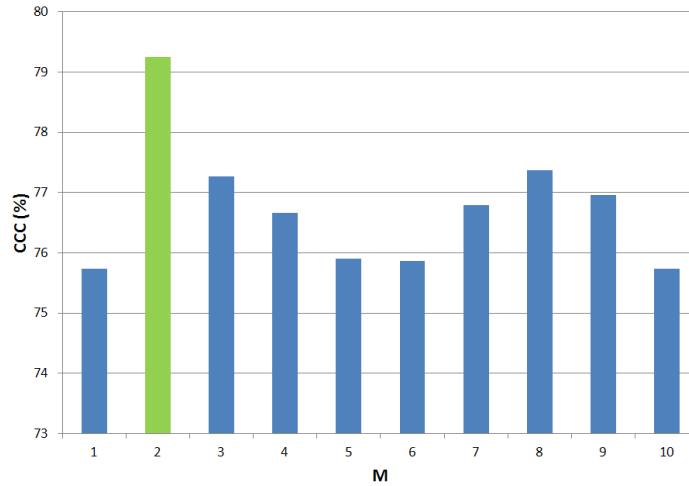


Figure 6.10: Mean obtained CCC for different M values (moving average predictor) in Experiment 1 (Web)

On the other hand, when the forecasts are made using the exponential smoothing predictor (Figure 6.11), it is maximized using a value of $\alpha_{ES}=0.8$, although 0.7 almost provides the same result.

For the linear regression predictor (Figure 6.12), a value of $LR=11$ is the best option. Considering the energy efficiency assessments displayed in Figure 6.14 below, it can be observed that its waveform is actually very noisy. So, by using such a high value, the linear regression predictor actually filters the noise of the last 11 observed CPU samples in order to predict the next one, which is consistent with the used web workload.

¹Separate results for each of these VMs can be found in Appendix A

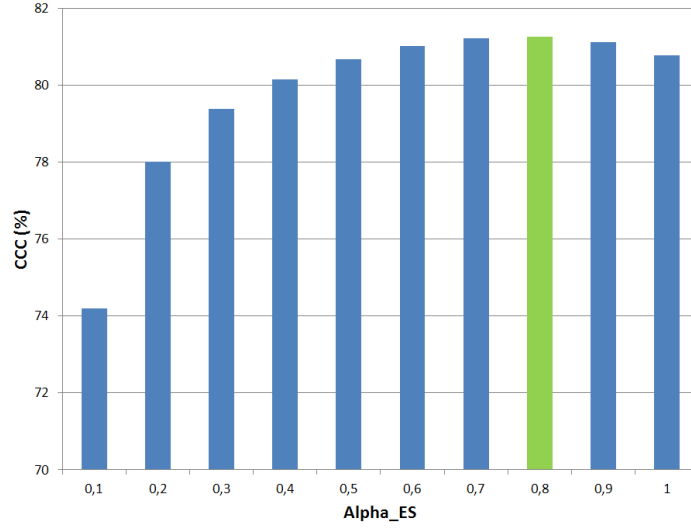


Figure 6.11: Mean obtained CCC for different Alpha_ES values (exponential smoothing predictor) in Experiment 1 (Web)

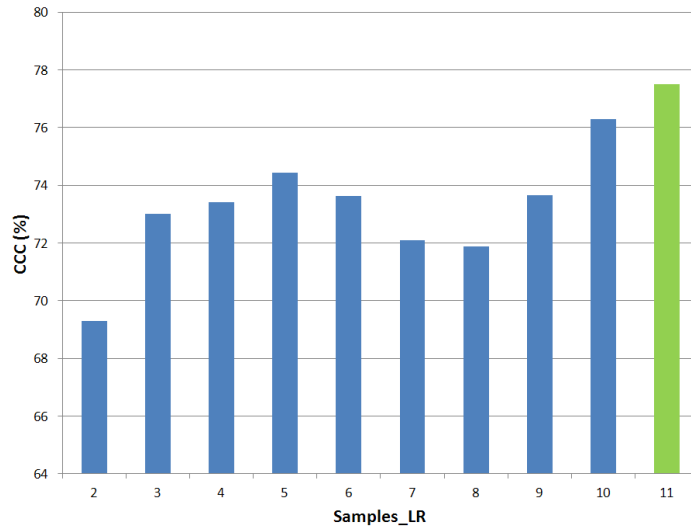


Figure 6.12: Mean obtained CCC for different Samples_LR values (linear regression predictor) in Experiment 1 (Web)

The double exponential smoothing is configured by means of two parameters: Alpha_DES and Beta_DES . For simplicity, the different parameter values have been calculated using Equation 6.10, by increasing the value of δ from 0.0 to 1.0 in steps of 0.1. Note that these parameters do not necessarily need to be related

in this way. As displayed in Figure 6.13, a value of $Alpha_DES=0.36$ (with a corresponding $Beta_DES$ of 0.1111) maximises the obtained CCC.

$$Alpha_DES = 1 - (1 - \delta)^2 \quad Beta_DES = \frac{\delta^2}{1 - (1 - \delta)^2} \quad (6.10)$$

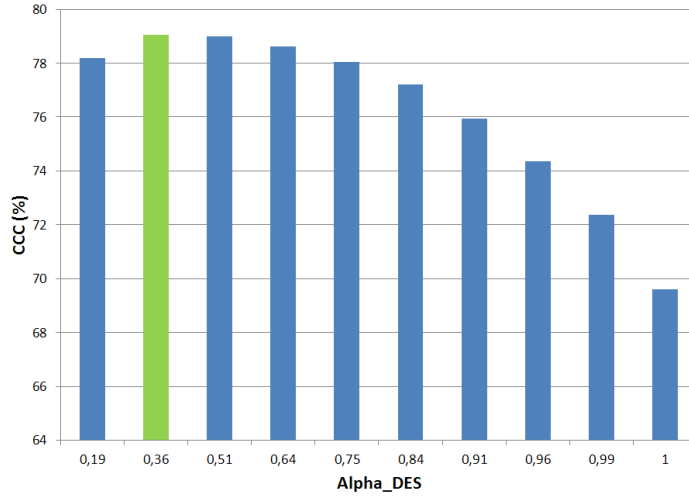


Figure 6.13: Mean obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Web)

Apart from validating the CPU estimators, during this experiment the energy efficiency was also internally assessed and forecasted by the Ecoefficiency Tool, which was configured with the following initial parameter values for its CPU estimators: $M = 3$, $Alpha_ES = 0.2$, $Samples_LR = 10$, $Alpha_DES = 0.36$ and $Beta_DES = 0.1111$. The obtained CCC was of 82.30%. Figure 6.14 displays how the energy efficiency forecasts closely follow the trend of the assessments over time quite well.

6.4.2 Experiment 2: last samples error tuning

The “best forecast” (see Chapter 4) of the CPU utilisation of the VMs running in the infrastructure is the one used by the Ecoefficiency Assessment Tool to calculate, among other things, the infrastructure’s energy efficiency. Therefore, a better precision when determining it may lead to an improvement in the overall precision of the infrastructure energy efficiency forecasts. Taking into account the design of the CPU estimator described in Chapter 4, the parameter which has direct influence when determining which predictor is chosen to provide the “best forecast” is $N_Samples$. $N_Samples$ refers to the number of samples of

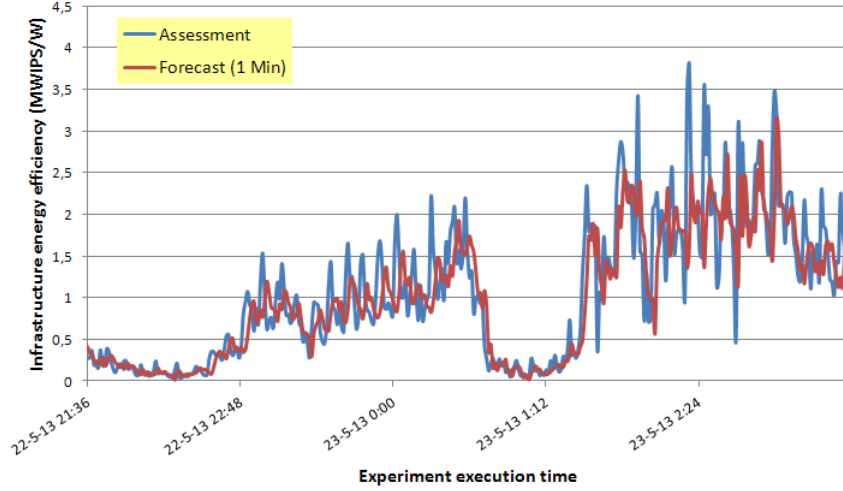


Figure 6.14: Infrastructure’s energy efficiency assessments and forecasts during Experiment 1 (Web)

the absolute error (of the previous forecasts of each individual predictor) which are considered to calculate the current mean absolute error of each predictor of the CPU estimator.

The experiment described in the previous section was repeated, but configuring the CPU estimators of the Ecoefficiency Assessment Tool so they used the optimal parameters for each predictor (determined in Experiment 1: $M=2$, $Alpha_ES=0.8$, $Samples_LR=11$, $Alpha_DES=0.36$ and $Beta_DES=0.1111$). In this second experiment, the impact of obtaining the “best forecast” by taking into consideration a value for $N_Samples$ which ranged from 1 to 10 (both inclusive) was evaluated. The obtained results can be observed in Figure 6.15. The results display the fact that the obtained precision is not highly affected by the value of $N_Samples$. Nevertheless, the one providing the highest CCC corresponds to $N_Samples=8$, which will be the one used in the next experiment.

As in the previous experiment, in this one the energy efficiency was also assessed and forecasted internally by the Ecoefficiency Tool, which was now configured with the optimal predictor configuration values. Note that in this experiment, the optimal $N_Samples$ value had not been determined yet and it was set to 1.

This configuration modification was reflected in the precision results obtained between the assessments and forecasts of the infrastructure’s energy efficiency: the resulting CCC was of 84.95%. Therefore, the use of the optimal parameters obtained in Experiment 1 led to a better precision when forecasting the energy efficiency of the infrastructure.

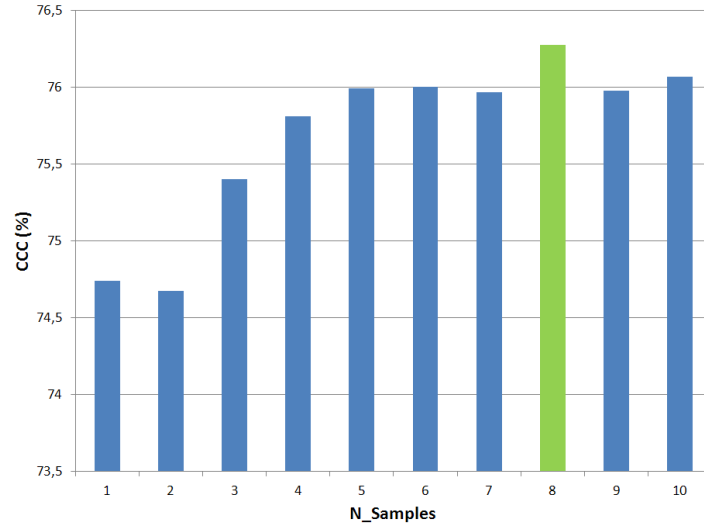


Figure 6.15: Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Web)

Figure 6.16 displays how the energy efficiency forecasts follow the evolution of the assessments closer than in the previous experiment. In particular, it can be observed how the peaks are predicted more accurately.

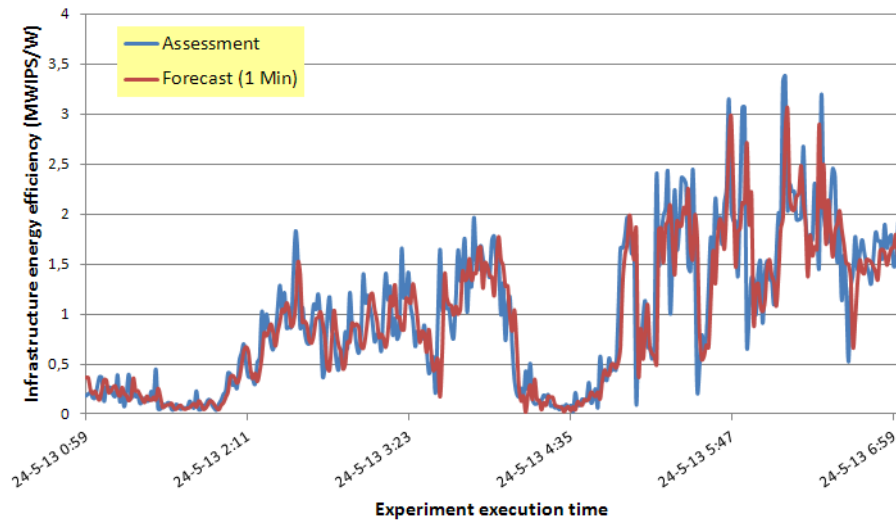


Figure 6.16: Infrastructure’s energy efficiency assessments and forecasts during Experiment 2 (Web)

One interesting outcome of Experiment 2 was that, in fact, the precision of the exponential smoothing estimator with $Alpha_ES=0.8$ happened to be better than the “best forecast” prediction, both in the support (CCC: 71.51%) and in the ecommerce (CCC: 83.05%) scenarios, for all the possible values of $N_Samples$. This fact suggests that the way in which the individual forecasts of the predictors of the CPU estimators are combined might not be the optimal one. Further study on this aspect will be conducted as future work.

6.4.3 Experiment 3: final results

Finally, the experiment was repeated again with the same parameters as in the previous experiment but setting a value of $N_Samples=8$ in the configuration of the CPU estimators of the Ecoefficiency Assessment Tool. Nevertheless, given that a low variation in the precision was observed in the last experiment when $N_Samples$ was modified, the external probe was configured to still perform the “best forecasts” for a value of $N_Samples$ ranging from 1 to 20, both inclusive.

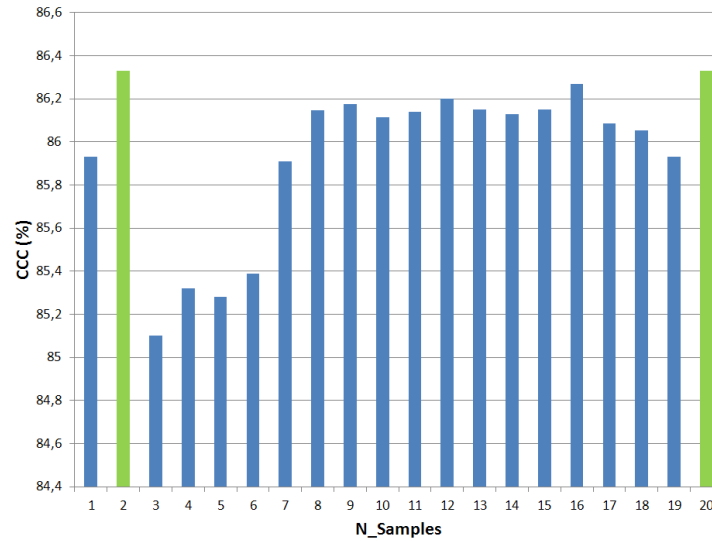


Figure 6.17: Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 3 (Web)

Figure 6.17 demonstrates that there is not a clear value for $N_Samples$ which provides the best precision, as the CCC is maximized for $N_Samples=2$ and $N_Samples=20$ in this experiment, while in the previous experiment it was maximized for $N_Samples=8$. Therefore, as discussed at the end of Experiment 2, the way the forecasts of the individual predictors inside of the CPU estimator are combined need to be studied further.

However, with $N_Samples=8$ the quality of the energy efficiency forecasts performed by the Ecoefficiency Assessment Tool clearly increased when compared to Experiment 2. In this experiment, the obtained CCC was of 90.35%. Although there is not a clear variation of the precision when modifying $N_Samples$, in both experiments it was observed that with $N_Samples=1$, the precision was worse than when selecting $N_Samples=2, 8$ or 20 . This partly justifies that in this experiment, given that $N_Samples$ was not equal to 1 (which was used in Experiment 2), the precision increased. On the other hand, in this experiment the variance of the energy efficiency assessments was lower than in the previous experiments (observe Figure 6.18), which allowed the predictions performing better.

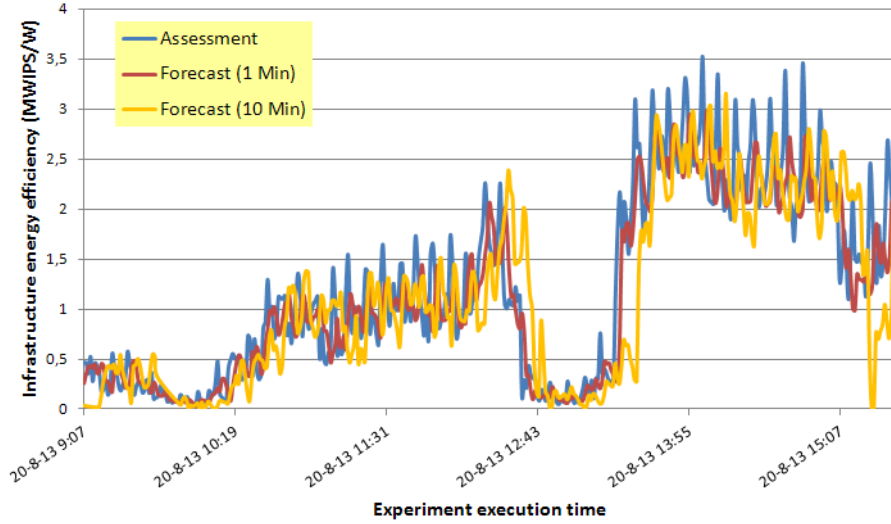


Figure 6.18: Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (Web)

Finally, the precision of the Ecoefficiency Tool when performing forecasts for different time intervals in the future was measured. Note that these time intervals represent the experiment execution time and not the simulated time (refer to Section 6.1.3 for more details). The results can be visualized in Figure 6.19. Clearly, the CCC suffers a degradation when the prediction is performed for more than one minute in the future. In fact, it can be concluded that for web workload the precision remains stable for predictions up to 7 minutes in the future and degrades from 8 minute predictions onwards. It has to be taken into account that given the time scaling ($T_{HOURL} = 8$ minutes, see Section 6.1) performed when conducting this experiment, 7 minutes in the simulation corresponds to 52.5 minutes in the “real world”.

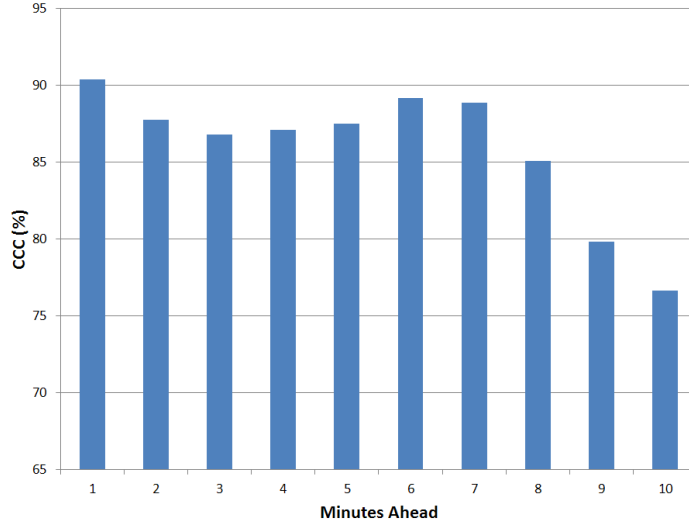


Figure 6.19: CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions in Experiment 3 (Web)

6.5 Model validation with batch workload

The Eco-efficiency Tool was also validated using a batch workload which was generated following the steps described in Section 6.2. The set of conducted experiments were the same ones as in Section 6.4.

The individual predictor’s configuration parameters (see Chapter 4) which maximized the mean CCC when predicting the CPU utilisation of all the deployed VMs were determined in the first conducted experiment (Section 6.5.1). Then, the optimum number of absolute error samples to consider when calculating the “best forecast” of the CPU utilisation of a VM (refer to Chapter 4 for more details) was determined in the second experiment (Section 6.5.2). Finally, in the third experiment (Section 6.5.3) these parameter values were introduced in the Ecoefficiency Tool to evaluate if its prediction precision increased. Moreover, the degradation of the CCC when performing such forecasts for multiple time intervals in the future was determined.

6.5.1 Experiment 1: predictors’ parameter tuning

The conducted experiment in this section was exactly the same as the one described in Section 6.4.1 but using a batch workload (described in Section 6.2)

instead. Refer to that section for more details.¹

In this case, the VMs causing high resource utilisations were the 6 *Worker VMs* (distributed as detailed in Section 6.2.2). The *Master VM* was not considered in the calculations because its resource utilisation was negligible. The CCC between the forecasted CPU utilisations and the real values was calculated for each *Worker VM*, for each parameter value and internal predictor.

Similarly to when a web workload was used, the results show that for a batch workload a value of $M=2$ for the moving average predictors also leads to the highest CCC value (Figure 6.20).

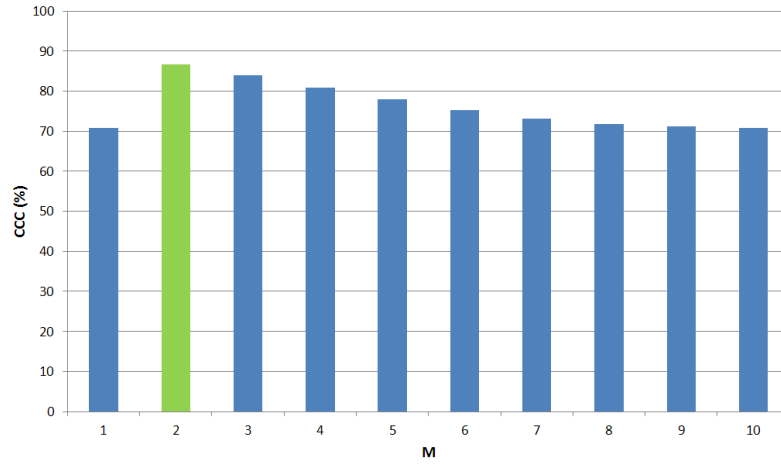


Figure 6.20: Mean obtained CCC for different M values (moving average predictor) in Experiment 1 (Batch)

Nevertheless, for the exponential smoothing predictors the CCC becomes maximized for $Alpha_ES=1$ (Figure 6.21). This means that the best forecast when using this predictor type is actually the value of the last observed sample. This discovery makes complete sense for a batch workload, as tasks tend to use the CPU where they are allocated (slot) at a constant 100% utilisation while they are executed, and not at intermediate utilisation levels as in the web workload. Therefore, the best estimation of CPU utilisation is the present one, as the task will keep using the whole CPU while it is being executed in a VM.

The linear regression predictors show the best precision when using a value of $Samples_LR=4$ (Figure 6.22), and not 11 as in the web workload. There are two reasons to explain this phenomena. On the one hand, the CPU utilisation changes happen very abruptly when running a batch workload (when a tasks starts or stops being executed). Therefore, the predictor needs to be very reactive when

¹Separate CCC results for each of the VMs considered in the present and following sections can be found in Appendix B

these changes happen. If a smaller value of *Samples_LR* is used, the predictor becomes adjusted to the new CPU utilisation values faster. On the other hand, given that when a task is running the CPU utilisation tends to be very stable, not many values need to be considered to predict the next one, as it will be very similar to the current one (this fact has also been observed in the exponential smoothing predictor).

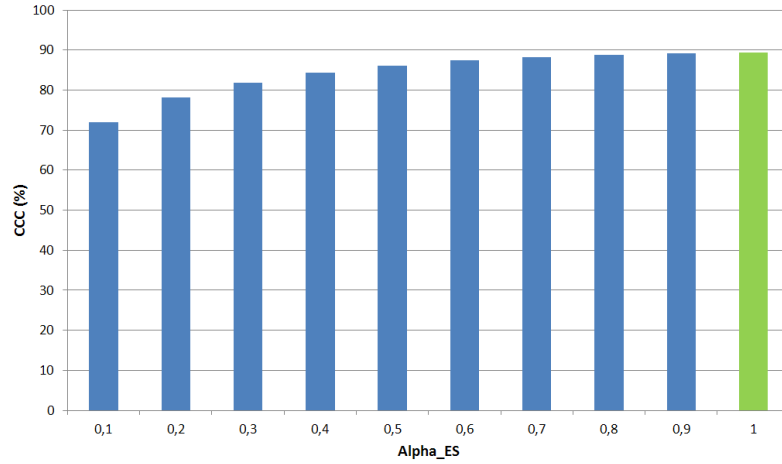


Figure 6.21: Mean obtained CCC for different *Alpha_ES* values (exponential smoothing predictor) in Experiment 1 (Batch)

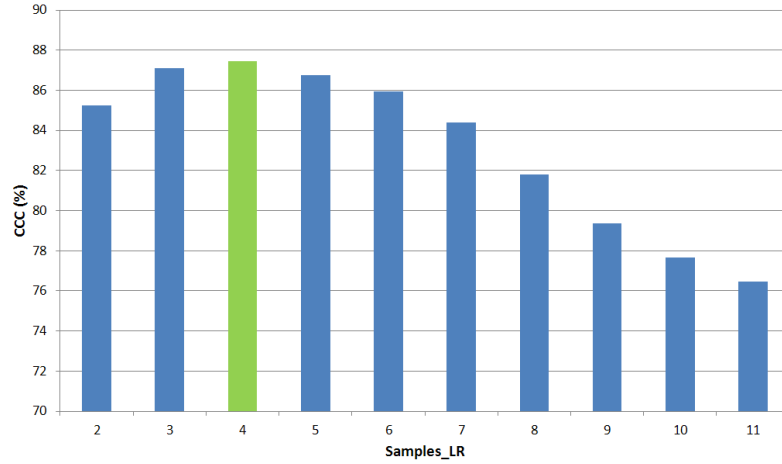


Figure 6.22: Mean obtained CCC for different *Samples_LR* values (linear regression predictor) in Experiment 1 (Batch)

The double exponential smoothing predictor obtained the greatest CCC when *Alpha_DES*=0.84 (with a corresponding *Beta_DES*=0.43), as it can be observed in Figure 6.23.

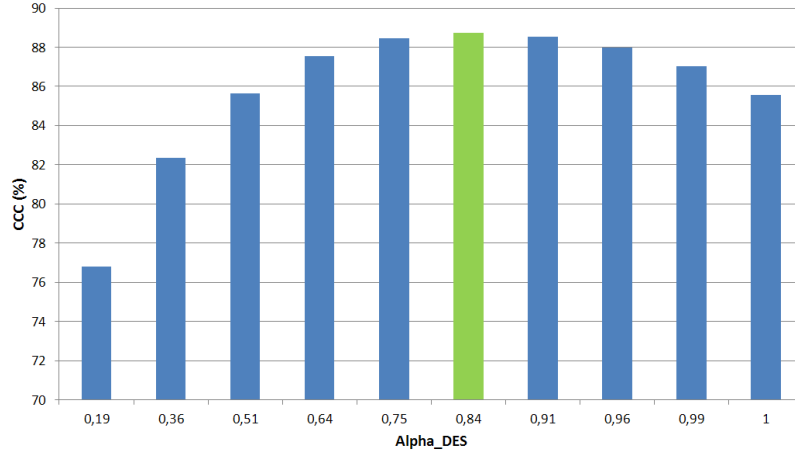


Figure 6.23: Mean obtained CCC for different $Alpha_DES$ values (double exponential smoothing predictor) in Experiment 1 (Batch)

In this first experiment, the Ecoefficiency Tool was configured with its initial default configuration values ($M = 3$, $Alpha_ES = 0.2$, $Samples_LR = 10$, $Alpha_DES = 0.36$ and $Beta_DES = 0.1111$). In this case, the infrastructure energy efficiency forecasts (performed for one minute ahead in the future) obtained a CCC of 88.36%, and can be observed in Figure 6.24. The objective of the following experiments will be to improve this result. Nevertheless, the obtained CCC is already very high.

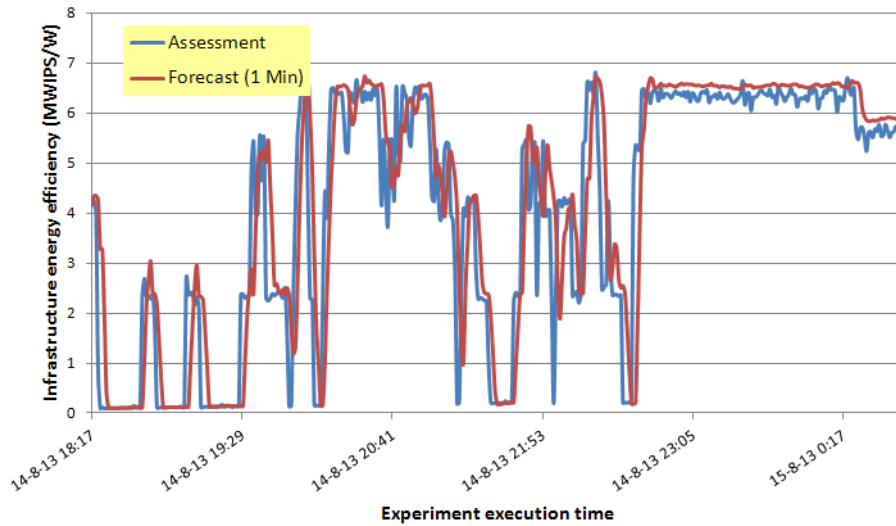


Figure 6.24: Infrastructure's energy efficiency assessments and forecasts during Experiment 1 (Batch)

6.5.2 Experiment 2: last samples error tuning

The conducted experiment in this section was exactly the same as the one described in Section 6.4.2 but using a batch workload (described in Section 6.2) instead. Refer to that section for more details.

After configuring the optimum parameter values obtained in the previous experiment inside of the Ecoefficiency Tool, the previous experiment was repeated in order to determine the value of $N_Samples$ which provided the best precision for the “best forecasts” (see Chapter 4) of the VM CPU estimators. The obtained results are displayed in Figure 6.25. In this case, the values of $N_Samples=1$ and $N_Samples=2$ were the ones maximising the CCC. The chosen value has been $N_Samples=1$, since it makes the estimator more reactive and this is more appropriate when working with a batch workload.

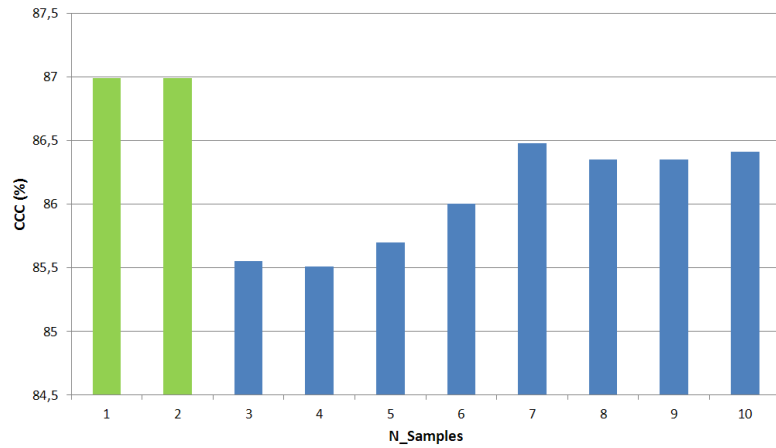


Figure 6.25: Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Batch)

The Ecoefficiency Tool was configured with the values obtained in the previous experiment ($M=2$, $Alpha_ES=1.0$, $Samples_LR=4$, $Alpha_DES=0.84$ and $Beta_DES=0.43$). Note that, casually, the initial value of $N_Samples$ was actually the optimum one which was found during this experiment. The infrastructure energy efficiency forecasts (performed for one minute ahead in the future) obtained a CCC of 87.99%, and can be observed in Figure 6.26.

In the previous experiment (without having tuned any parameter) the CCC was actually greater, 88.36%. If the two experiment’s energy efficiency waveforms are compared (Figures 6.24 and 6.26), it can be seen that in the first experiment there were some tasks being queued (as the energy efficiency looks like a “mountain” during the first simulated day) but not during the first simulated day of the second experiment. So, the CPU utilisation of the VMs during the second

experiment performed more abrupt variations than in the first experiment, which was more stable. These variations are actually the hardest values to predict and therefore causing a decrease in the overall CCC.

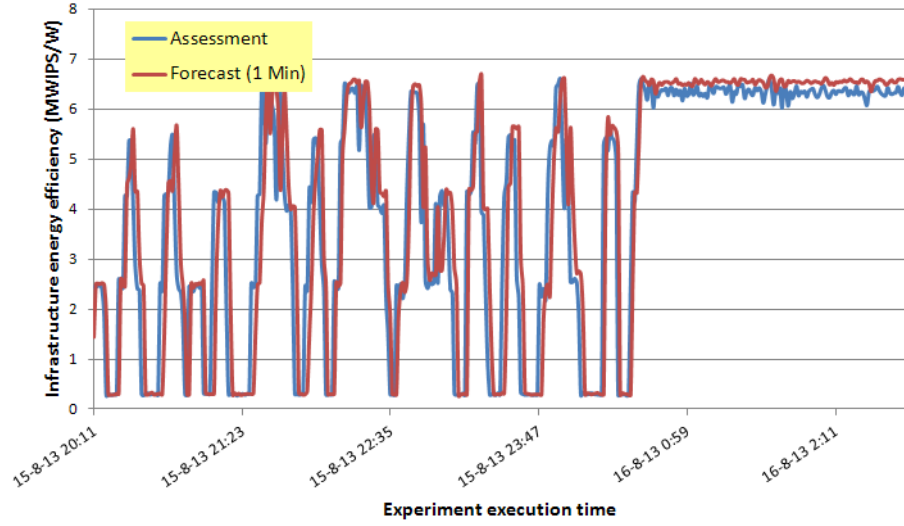


Figure 6.26: Infrastructure’s energy efficiency assessments and forecasts during Experiment 2 (Batch)

6.5.3 Experiment 3: final results

In the previous experiment, it was determined that a value of $N_Samples=1$ was the one providing the values for the “best forecasts” (refer to Chapter 4) that maximized the overall CCC of the CPU estimators. In fact, this value had already been set in the previous experiment, as it was the initial default value when the experiments were started. Therefore, it is reasonable to expect that the CCC for the energy efficiency forecasts of the infrastructure will be similar to the previous experiment.

In fact, this time the obtained CCC for the infrastructure energy efficiency forecasts was greater: 91.35%. From the waveform displayed in Figure 6.27, it can be observed that the energy efficiency did not experiment so many variations as in the previous experiment. This explains why the obtained CCC was greater in this experiment.

So, on average it can be concluded that the CCC is improved when the individual predictor configuration parameters are tuned, but it is highly dependant on the way the tasks are submitted to the COMPSs PM. If tasks are queued, the CPU utilisation is more stable, easier to predict and therefore leads to a greater CCC. However, if more CPU utilisation variations take place (due to new tasks

arriving or dying when there are no tasks in the queue waiting to be executed), the CCC tends to decrease.

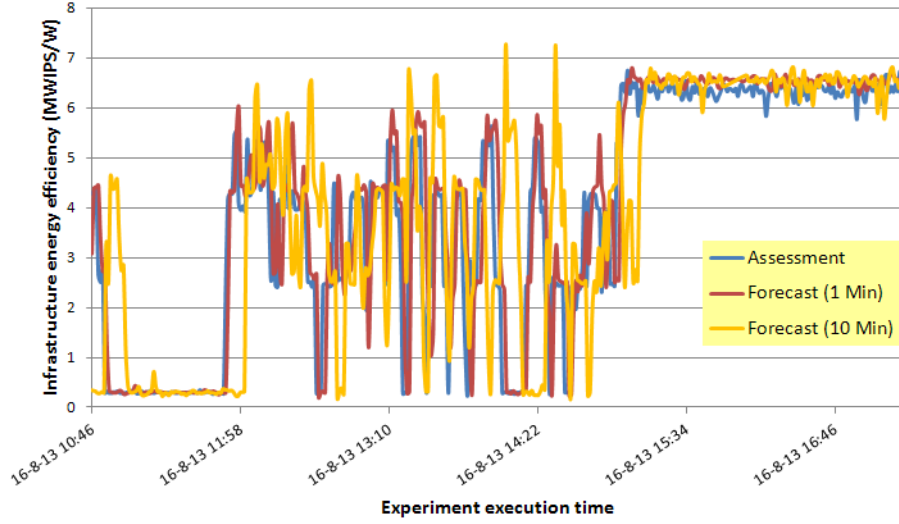


Figure 6.27: Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (Batch)

Finally, the degradation of the CCC as the energy efficiency of the infrastructure is forecasted for longer time spans in the future has been evaluated. The obtained results are depicted in Figure 6.28. In this case, it can be observed how the degradation is more pronounced than when operating with web workload. Note that these time intervals represent the experiment execution time and not the simulated time (refer to Section 6.1.3 for more details).

Observe that while in the web workload the CCC decreased 10% when the energy efficiency was predicted 9 minutes ahead in the future, the same 10% degradation takes place when it is predicted for only 3 minutes in the future when a batch workload is used. This is explained by that fact that in batch workloads the energy efficiency can change very abruptly, from values around 5 MWIPS/W to 0 MWIPS/W (for instance, when a task ends its execution and there are no tasks waiting in the scheduler queue). Such variations do not take place when a web workload is used.

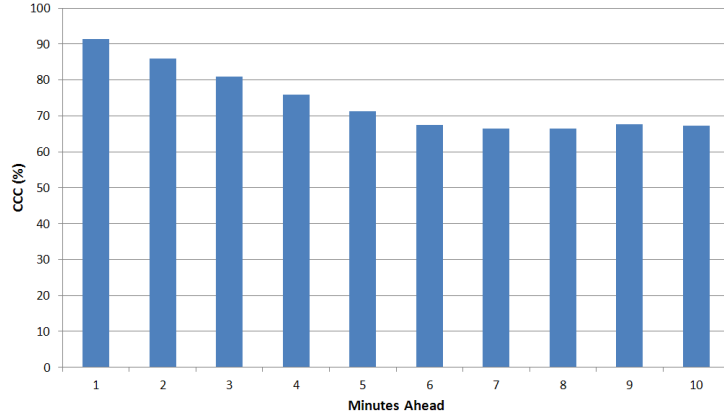


Figure 6.28: CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions in Experiment 3 (Batch)

6.6 Model validation with heterogeneous workload

Finally, the Eco-efficiency Tool was validated with a mixed workload consisting on a combination of a web and a batch workload, generated as described in 6.3. The set of conducted experiments were the same ones as in Section 6.4.

Like in the previous experimentation procedures, the first experiment (Section 6.6.1) seeks to obtain the optimum configuration parameters for the individual predictors of the CPU estimators, while in the second experiment (Section 6.6.2) the optimum number of absolute error samples to decide upon the forecast provided by one predictor or another is determined. Refer to Chapter 4 for more details on the design of the CPU estimator. Finally, all the obtained optimum configuration parameter values are set in the Ecoefficiency Tool and the same experiment is repeated once more (Section 6.6.3) to determine if an improvement is achieved in the CCC between the assessments and forecasts of the energy efficiency of the whole infrastructure.

6.6.1 Experiment 1: predictors’ parameter tuning

In this first experiment, the resulting optimal parameter values were $M = 2$ for the moving average predictor (Figure 6.29), $Alpha_ES = 1.0$ for the exponential smoothing predictor (Figure 6.30), $Samples_LR = 4$ for the linear regression predictor (Figure 6.31) and $Alpha_DES = 0.75$ (with a corresponding $Alpha_DES = 0.2933$) for the double exponential smoothing predictor (Fig-

ure 6.32). These aforementioned values maximised the mean obtained CCC of the forecasts provided by each individual predictor.¹ Note that the configuration values obtained for the moving average, exponential smoothing and linear regression predictors coincide with the ones that were obtained when running a pure batch workload. Only the double exponential smoothing predictor has obtained a different result when operating with a mixed workload, which actually resides in between the values obtained for a pure web workload ($Alpha_DES=0.36$) in Section 6.4.1 and for a pure batch workload ($Alpha_DES=0.84$) in Section 6.5.1.

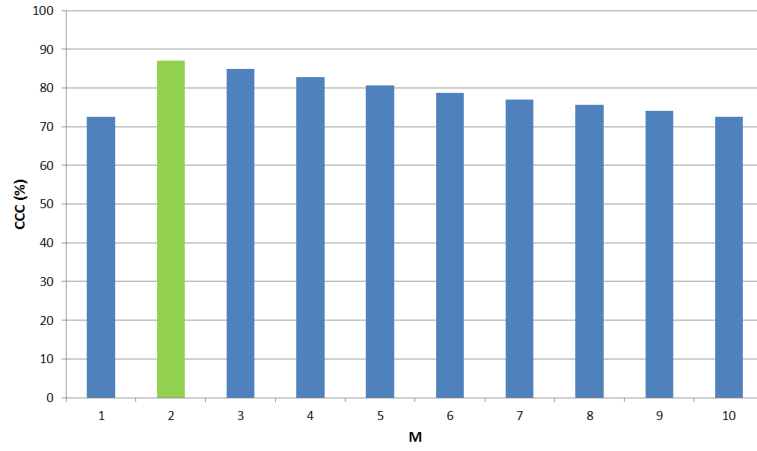


Figure 6.29: Mean obtained CCC for different M values (moving average predictor) in Experiment 1 (Mixed)

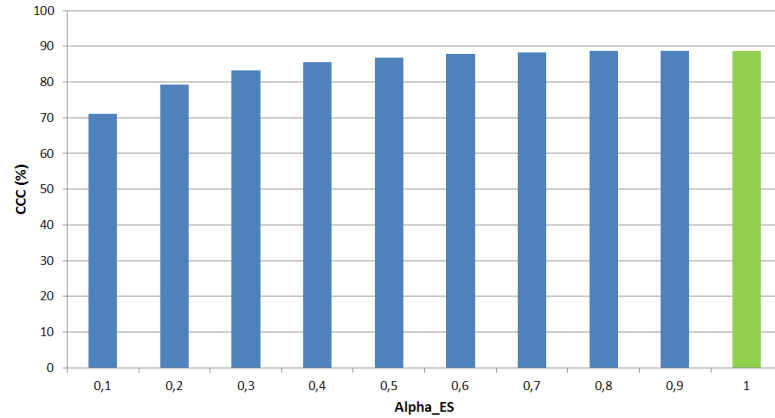


Figure 6.30: Mean obtained CCC for different $Alpha_ES$ values (exponential smoothing predictor) in Experiment 1 (Mixed)

¹Separate CCC results for each of the VMs considered in the present and following sections can be found in Appendix C

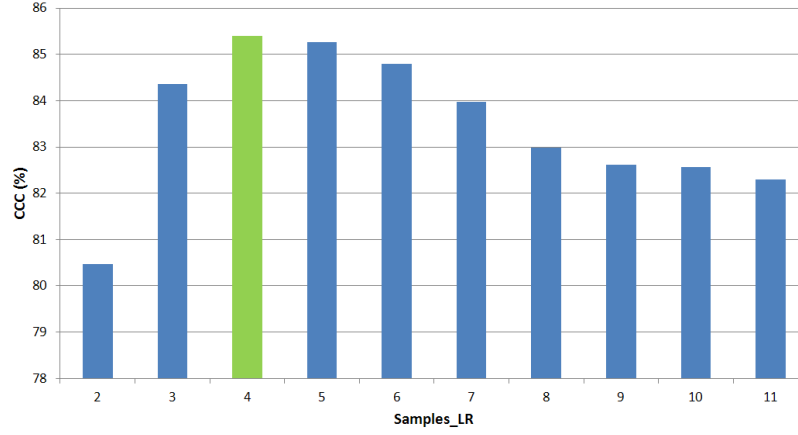


Figure 6.31: Mean obtained CCC for different *Samples_LR* values (linear regression predictor) in Experiment 1 (Mixed)

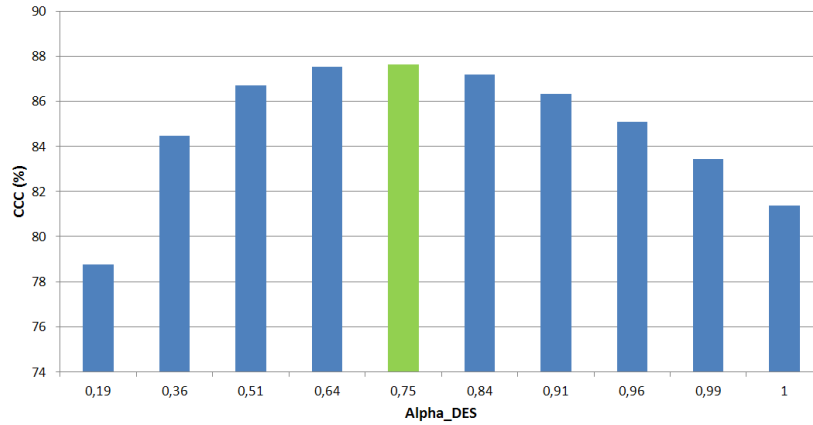


Figure 6.32: Mean obtained CCC for different *Alpha_DES* values (double exponential smoothing predictor) in Experiment 1 (Mixed)

Therefore, it can be concluded that when operating with a mixed workload consisting of a combination of web and batch workloads, the batch workload predominates over the web one because of its abrupt changes in resource utilisation (when tasks are submitted or finish their execution). So, the configuration parameters tend to be those which make the internal predictors more reactive, and not so focused on noise filtering as when operating with pure web workload. Moreover, both the exponential and double exponential predictors give more predominance to the last observed samples rather than the ones previously observed in the past (therefore making them more reactive), as when a pure batch workload is used.

Figure 6.33 displays the infrastructure energy efficiency assessments and forecasts performed one minute ahead in the future during this experiment. It can be observed how the energy efficiency waveform is a combination of the ones obtained when using web and batch workloads. It can be observed how in the second simulated day, the tasks of the batch workload were actually queued, as the energy efficiency is more stable. Additionally, the noise caused by the web workload can also be observed during this second simulated day. In this experiment, the obtained CCC between them has been of 78.19%. The objective of the following experiments will be to improve this result.

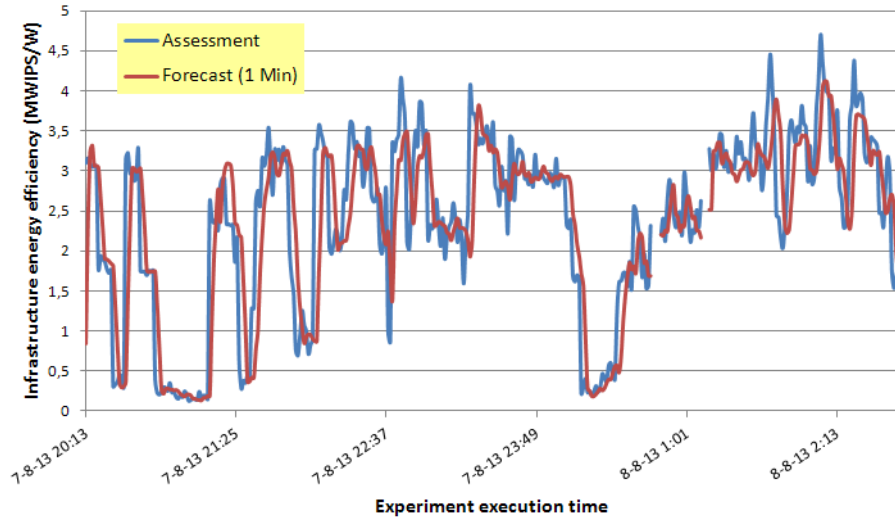


Figure 6.33: Infrastructure’s energy efficiency assessments and forecasts during Experiment 1 (Mixed)

6.6.2 Experiment 2: last samples error tuning

After configuring the optimal parameter values obtained in the previous experiment in the Ecoefficiency Tool, the same experiment was repeated. The objective in this experiment was to determine the value of $N_Samples$ which provided the set of “best forecasts” with the highest precision (greatest CCC). Refer to Chapter 4 for more details of the internals of the CPU estimator. As it can be observed in Figure 6.34, the predictor using a value of $N_Samples=2$ achieved the greatest CCC, even though the one using $N_Samples=1$ was also very close. Observe that when using a pure batch workload, a value of $N_Samples=1$ was obtained. Therefore, this result also suggests that when using a mixed workload, the batch workload predominates and therefore better results are achieved when using more reactive predictors.

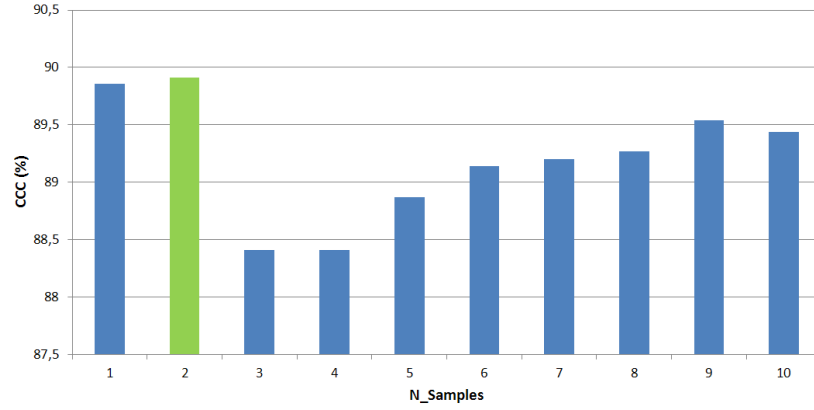


Figure 6.34: Mean obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Mixed)

In this experiment, the CCC between the infrastructure energy efficiency assessments and forecasts was of 90.03%. So, a good improvement has been achieved compared to the previous experiment. In this case the second simulated day was more “stable”, given that the batch tasks were queued throughout all the day. This factor has also played an important role in improving the CCC, and not only the newly used configuration parameter values.

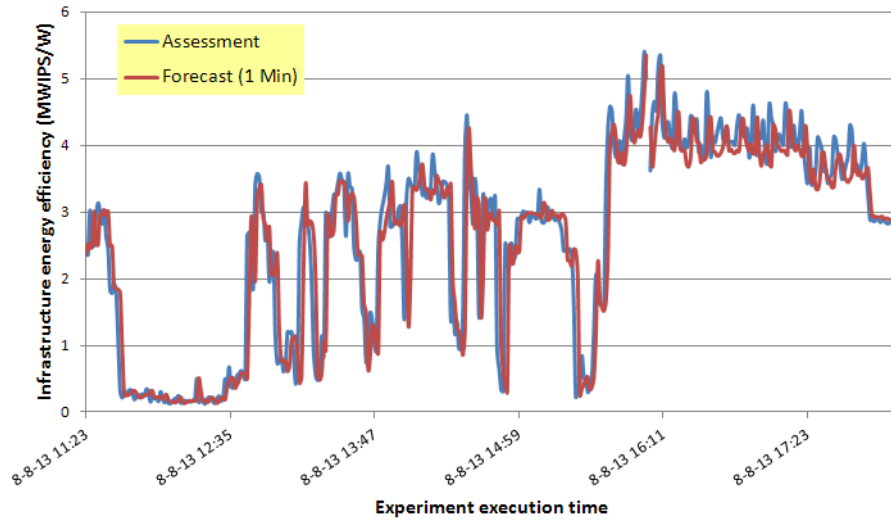


Figure 6.35: Infrastructure’s energy efficiency assessments and forecasts during Experiment 2 (Mixed)

6.6.3 Experiment 3: final results

In this last experiment, all the obtained optimal configuration parameter values were introduced in the Ecoefficiency Tool. The energy efficiency evolution over time and its forecasts for 1 and 10 minutes ahead in the future are displayed in Figure 6.36. As it can be observed, this execution in particular was very “stable” because more batch tasks were submitted and they became queued from the first simulated day. Remember that the task arrivals are modelled following a Poisson process, which is actually random. In this case, the obtained CCC between the energy efficiency assessments and forecasts (for 1 minute ahead in the future) was of 92.73%.

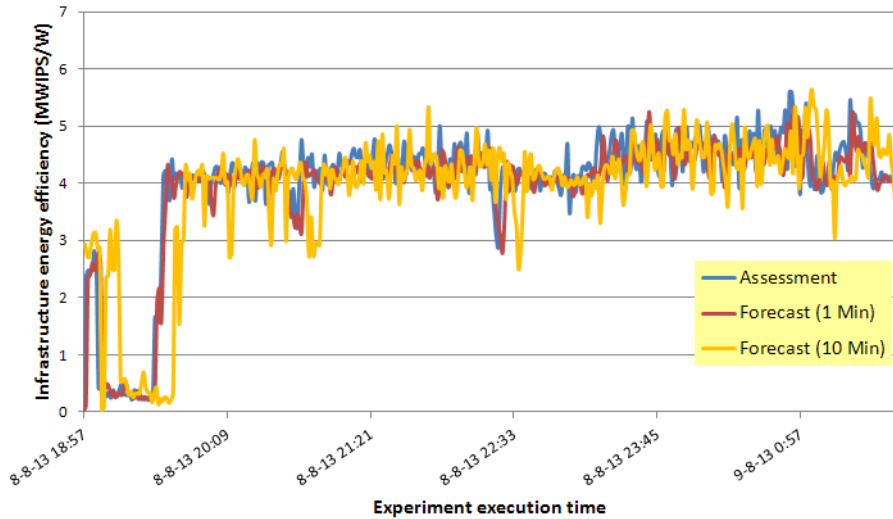


Figure 6.36: Infrastructure’s energy efficiency assessments and forecasts during Experiment 3 (first execution, Mixed)

Also, the degradation of the precision, displayed in Figure 6.38, is not very pronounced. In this case, it decreases by 10% when predictions are made 6 minutes ahead in the future. Note that these time intervals represent the experiment execution time and not the simulated time (refer to Section 6.1.3 for more details).

Even though the overall precision of the Ecoefficiency Tool improved because of the newly configured parameter values, it was suspected that such improvements were caused, partially, by the fact that the execution had been very “stable”. Because of this, the experiment was repeated one more time.

In this second execution of Experiment 3, a CCC of 87.29% was obtained, which is actually lower than the one obtained in Experiment 2. It can be observed in Figure 6.37 that in this execution, the energy efficiency has varied abruptly in

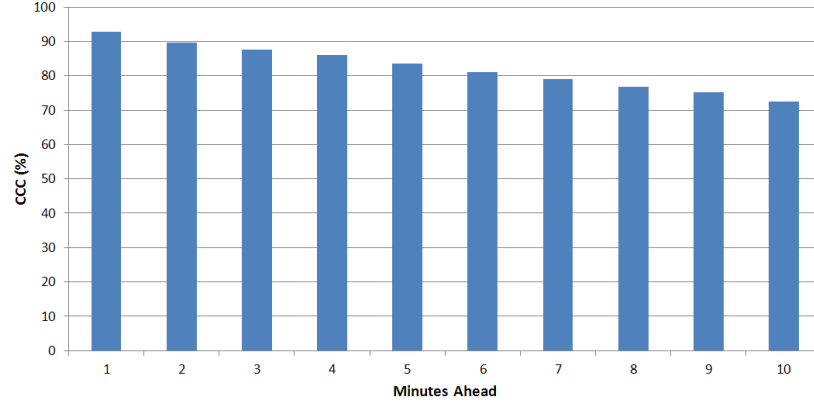


Figure 6.37: CCC degradation of the infrastructure's energy efficiency forecasts for different time interval predictions in Experiment 3 (first execution, Mixed)

multiple occasions, it has not been so “stable”. Therefore, it can be concluded that the more “CPU utilisation jumps” a *Worker VM* performs (because no new tasks are waiting in the scheduler queue), the more difficult it becomes to predict the overall infrastructure energy efficiency. In order to overcome such limitation, the CPU estimators would need to have information regarding the number of tasks waiting in the queue and the remaining execution time of those being executed.

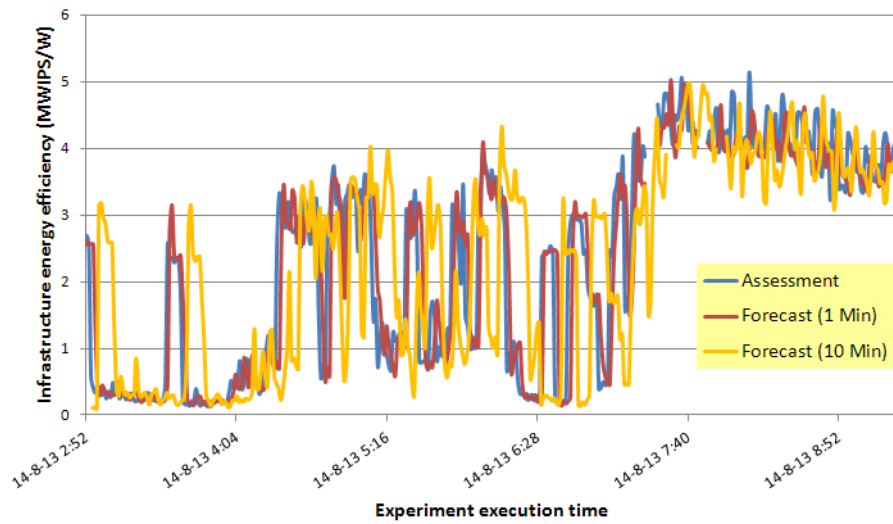


Figure 6.38: Infrastructure's energy efficiency assessments and forecasts during Experiment 3 (second execution, Mixed)

The CCC degradation as the forecasts were performed for more minutes ahead in the future was much more pronounced in this case. The degradation was of 10% for just 2 minute predictions compared to the 1 minute ones, while values of CCC close to 40% (which is considered to be very bad) were obtained for 7 minute predictions onwards. These results also reflect the influence of the “workload stability” on the obtained results.

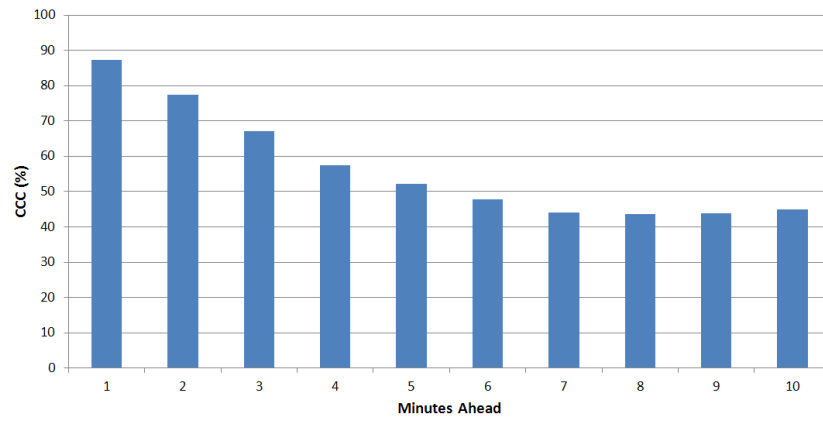


Figure 6.39: CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions in Experiment 3 (second execution, Mixed)

Chapter 7

Conclusions and Future Work

7.1 Conclusions

IaaS providers have a great interest in optimising the overall energy efficiency of their infrastructure. This requires precise knowledge of the current and future energy efficiency at different levels (VM, node, infrastructure or service) in order to take the appropriate decisions (e.g. VM replication, migration, cancellation). The current available tools do not provide such information in real time. To tackle such limitation, mathematical models to calculate and forecast it at the different aforementioned levels have been provided in Section 3.2.

In addition, the VM placement algorithms need to know what will be the impact of performing such actions (service deployment or VM deployment, migration or cancellation) in the overall infrastructure energy efficiency. This thesis has also addressed this last requirement in Section 3.3, by mathematically modelling such situations.

It has also been observed that a precise way to determine the future CPU utilisation of a VM is necessary to perform all the aforementioned forecasts. To this end, a generic variable estimator which can be used for CPU utilisation forecasting has been developed and detailed in Chapter 4.

The assessment and forecast of the whole infrastructure has been the most important model deduced in this thesis, because it is the factor that IaaS providers want to optimise and it is the basis of other models, such as the forecasts for potential actions described in Section 3.3. So, the conducted experimentation has been focused in determining the configuration parameter values of the internal CPU estimators which provide the best precision when forecasting the energy efficiency of the infrastructure for different workload types. Many conclusions have been drawn.

In general, the forecasts' precision (evaluated by means of the CCC) improves significantly when the individual predictor configuration parameters are properly

tuned (see Chapter 4). However, the way the individual CPU utilisation forecasts of the different internal predictors are combined may not be the optimal one, as in some cases an individual predictor achieved a CCC higher than the one achieved by the “best forecasts”.

Nevertheless, the obtained precision is also highly dependant on the conducted experiment execution. If the workload causes numerous spontaneous variations in the resource utilisation, this reduces the overall achieved precision because the internal predictors are not able to predict such sudden changes. This effect can be observed when a batch or a mixed workload (including a batch workload) is executed. On the one hand, if tasks are waiting to be executed in the scheduler queue, the execution is more “stable” and the overall achieved precision increases. On the other hand, if more “CPU utilisation jumps” take place (e.g. when a task starts or finishes its execution and there are no tasks waiting in the scheduler queue) the overall precision decreases.

Web workloads tend to be on average more stable than batch workloads (they don’t experience sudden CPU utilisation changes from 0% to 100% or vice versa). However, they are very noisy since their resource utilisation levels tend to oscillate around a given average value, depending on the number of simultaneous users accessing the web service. This fact causes that the obtained best configuration parameter values for the individual predictors mentioned in Chapter 4 make them less reactive (slower in the presence of changes) and more focused on noise filtering.

Batch workloads are quite the opposite as they have been found to require more reactive predictors. They allow them to overcome sudden utilisation variations in a fast manner. What is more, given that during a task execution the CPU utilisation (of each CPU) remains stable around 100% and without much noise, even though the reactors are very reactive the obtained forecasts are still very precise.

When web and batch workloads are combined in a single mixed workload, the best configuration parameter values tend to be the ones used when a pure batch workload is used. That is, the batch workload tends to predominate over the web workload, and the predictors need to be more reactive to obtain a better precision, on average. Even though the resource utilisation caused by the web workload component is still very noisy, the noise variance is much smaller than the stable resource utilisation caused by the batch workload component.

The degradation of the precision of the forecasts as they were made for a longer period of time in the future has been evaluated. It has been observed that for web workloads, the precision remains quite stable until predictions up to 7 minutes in the future, and decreases by 10% for 8 minute ones. The reason for this is that during the experimentation, each simulated hour (and hence associated load level) lasted only 8 minutes. Therefore, predictions up to that time interval were

accurate as the number of simultaneous users accessing the web service remained more or less constant. For batch and mixed workloads the degradation was much faster: 10% degradation for 3 and 2 minute predictions, respectively. This last result was also observed to be highly dependant on the execution, as it varied depending on the number of sudden resource utilisation variations during the conducted experiment.

7.2 Future Work

The currently adopted performance metric, MWIPS, evaluates the performance of integer and floating-point arithmetic. It is focused on the performance being delivered by the CPU, but not in the one which could be measured considering the system's memory, hard disk or network card. So, the way the performance is calculated could be extended by also considering such components

In [32] a very interesting mechanism for VM power metering is presented. The researcher uses a power model that based on the resource utilisation of each VM provides an estimation of its power consumption. A similar model could be used when assessing or forecasting the energy consumption of a VM, which could be then used to assess and forecast the energy efficiency at this VM level more precisely. Moreover, as discussed in [37], as servers' energy consumption becomes more proportional to their resource utilisation, the datacenters' network energy consumption becomes more significant. Therefore, the network power consumption caused by each VM because of its traffic should also be taken into account in the power calculation at the different levels (VM, node, service and infrastructure).

The way the individual forecasts are combined in the developed CPU estimator may not be the optimal one, as in some cases the CCC of an individual predictor was higher than the one achieved by the "best forecasts". So, alternative ways to combine the information provided by the different individual predictors should be studied. For instance, the individual forecasts could be averaged using weights based on the mean average error of each individual predictor.

It has been observed that sudden variations in the resource utilisation leads to an overall decrease in the forecasting precision. Therefore, a way to improve the current CPU estimator is to provide it with means to anticipate such spontaneous changes. For example, when running a batch or mixed workload, having information about the estimated remaining execution time of each task and the scheduler queue status could help in improving this aspect.

Each workload type requires different configuration parameters. A possible improvement could be to keep track, for each individual predictor, of the parameter value which would have maximized the CCC over a given number of CPU

utilisation samples in the past. Based on this result, the configuration parameter value could be changed in real-time in order to adapt the predictors to the workload type of each individual VM.

Currently, the Ecoefficiency Tool has been programmed as a centralized component which obtains the required data from the Monitoring Infrastructure database and performs the required calculations for the whole infrastructure at all the required levels. Such implementation is not scalable if the number of nodes grows. A distributed implementation would be more desirable in this sense.

Chapter 8

Publications

An initial version of the work conducted in this thesis has been published in the following journal issue:

- G. Katsaros, et al., A service framework for energy-aware monitoring and VM management in Clouds, *Future Generation Computer Systems* (2013), doi:10.1016/j.future.2012.12.006

Note that some of the models presented in that publication have been improved in this final version of the thesis, such as the models for the energy efficiency at infrastructure and service levels.

Appendix A: Detailed results for the experimentation with web workload

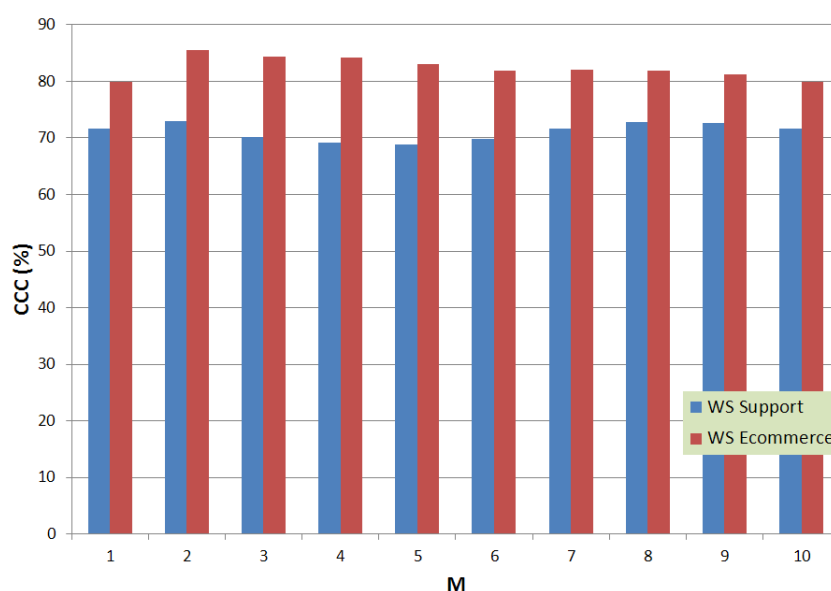


Figure A.1: Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Web)

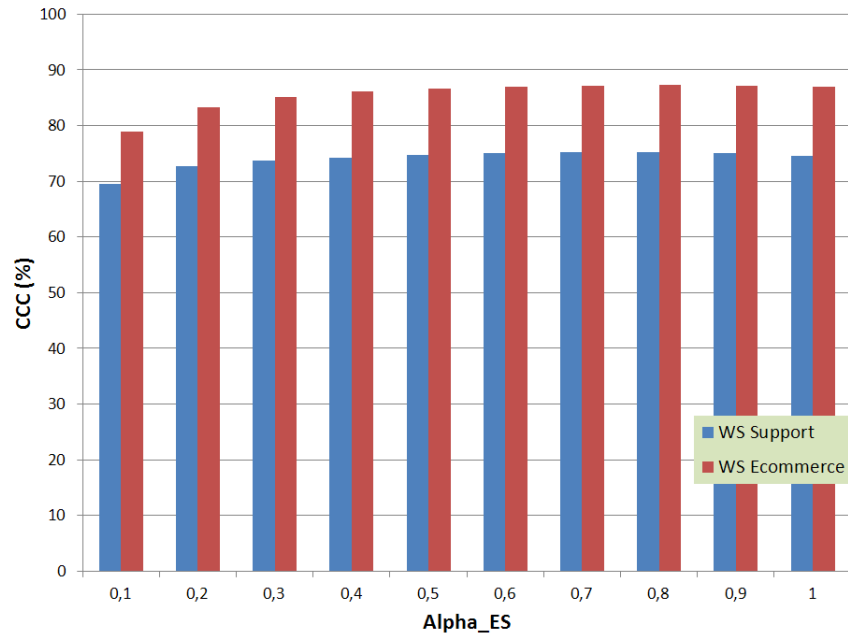


Figure A.2: Individual obtained CCC for different α_{ES} values (exponential smoothing predictor) in Experiment 1 (Web)

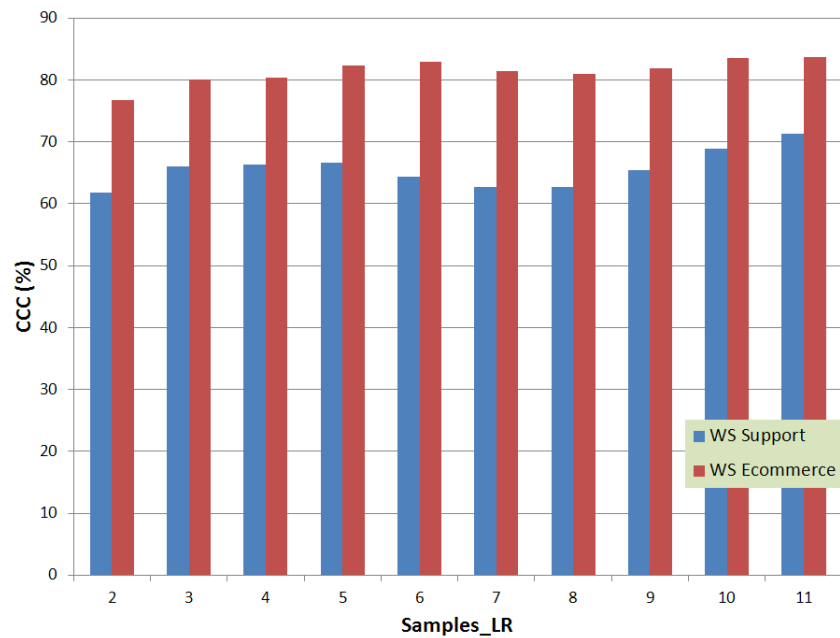


Figure A.3: Individual obtained CCC for different $Samples_{LR}$ values (linear regression predictor) in Experiment 1 (Web)

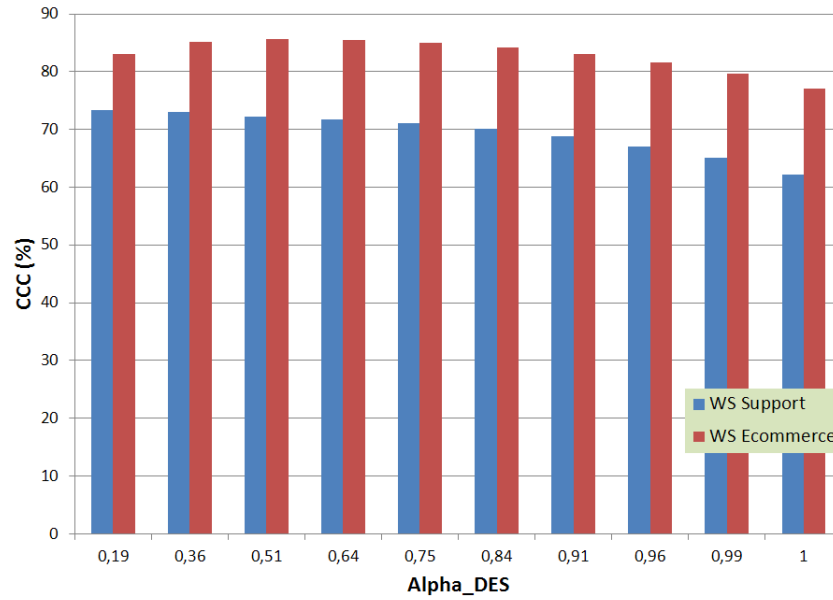


Figure A.4: Individual obtained CCC for different Alpha_DES values (double exponential smoothing predictor) in Experiment 1 (Web)

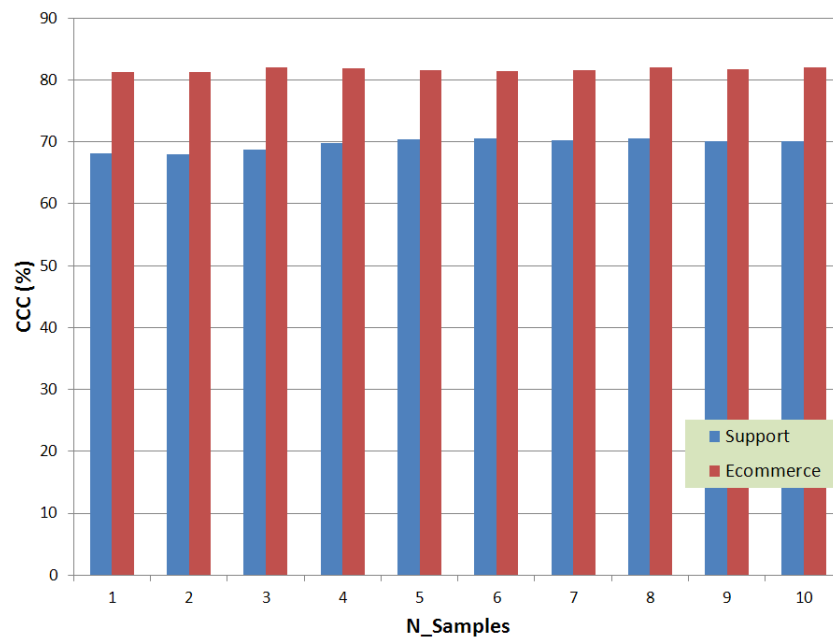


Figure A.5: Individual obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Web)

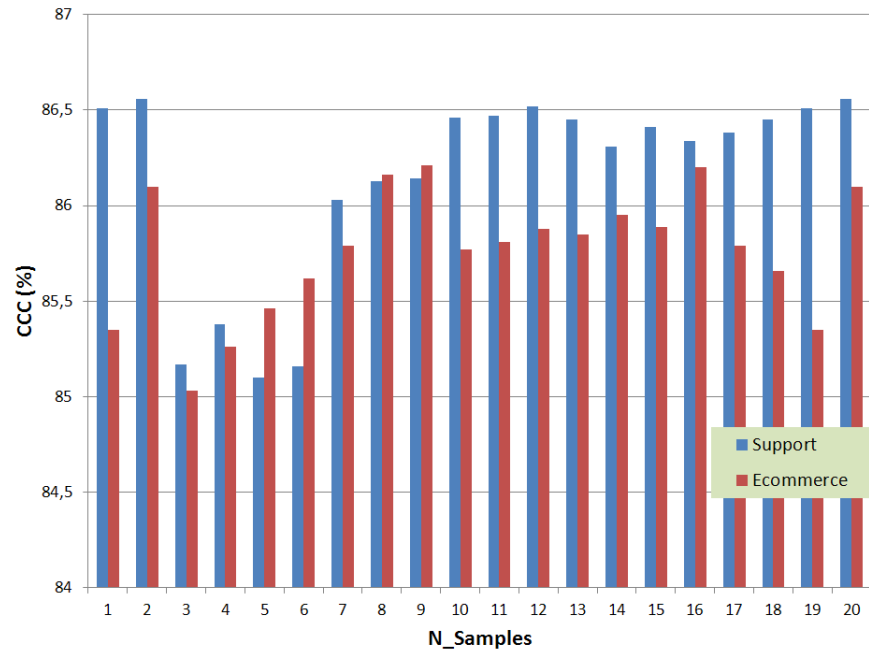


Figure A.6: Individual obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 3 (Web)

Appendix B: Detailed results for the experimentation with batch workload

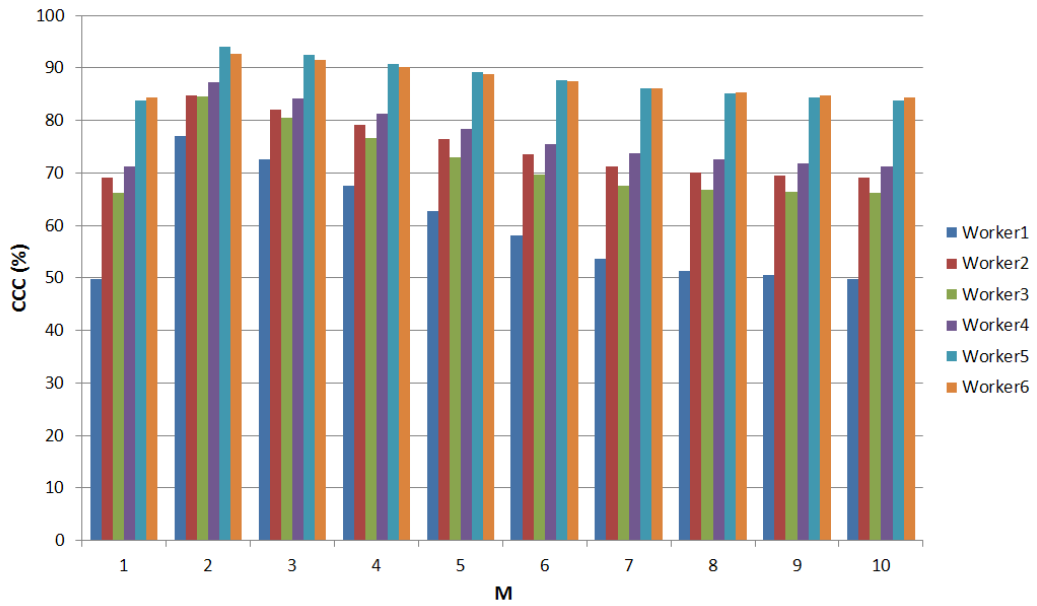


Figure B.1: Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Batch)

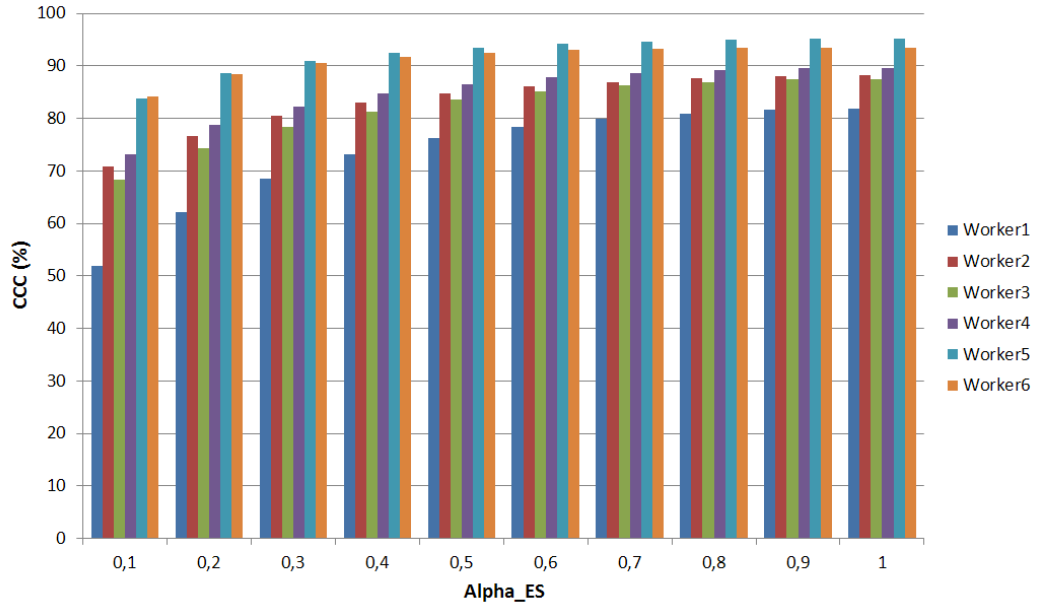


Figure B.2: Individual obtained CCC for different Alpha_ES values (exponential smoothing predictor) in Experiment 1 (Batch)

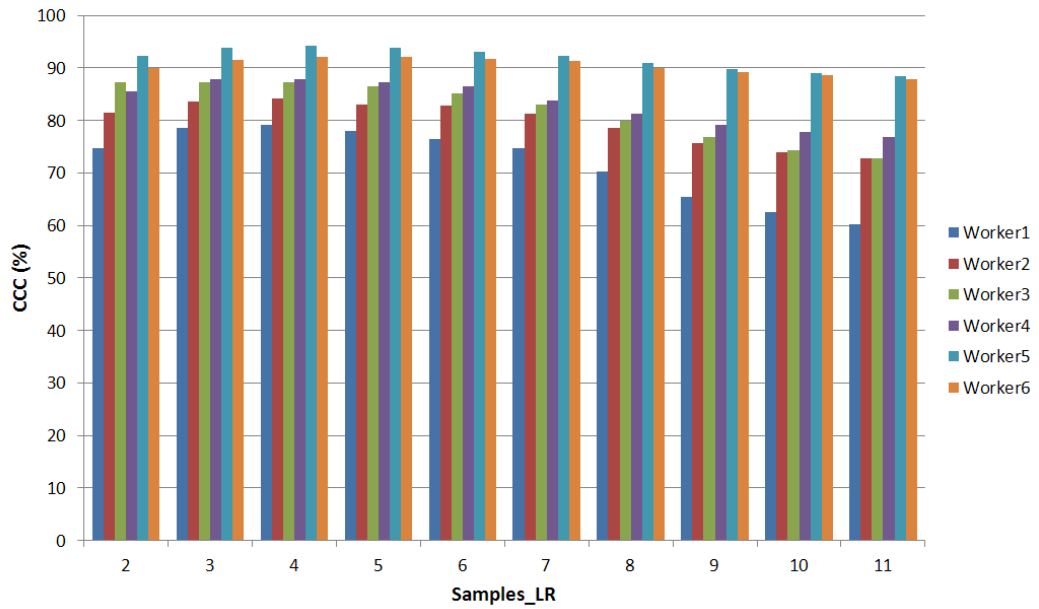


Figure B.3: Individual obtained CCC for different Samples_LR values (linear regression predictor) in Experiment 1 (Batch)

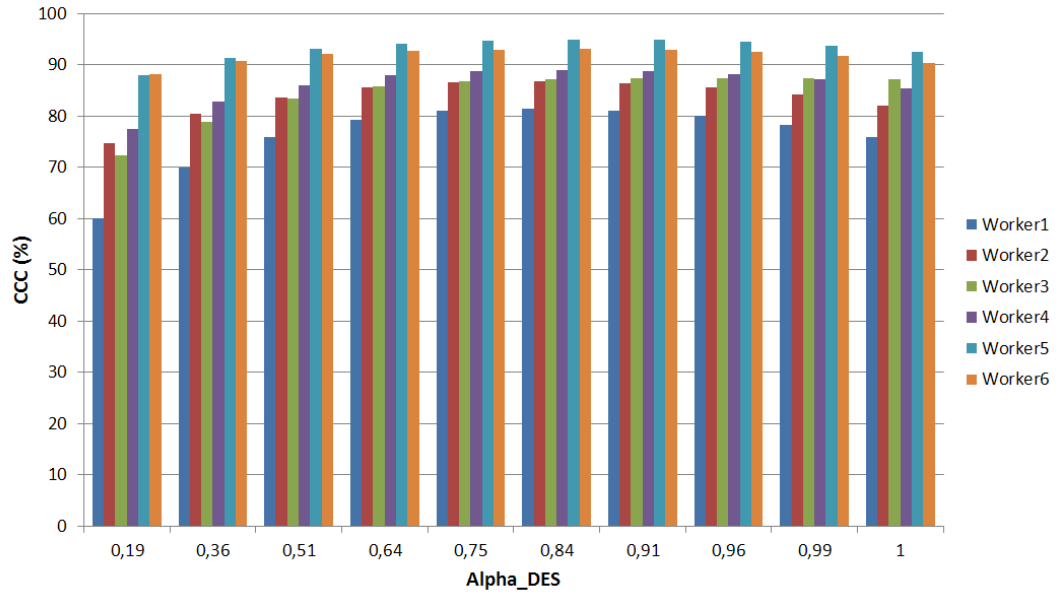


Figure B.4: Individual obtained CCC for different Alpha_DES values (double exponential smoothing predictor) in Experiment 1 (Batch)

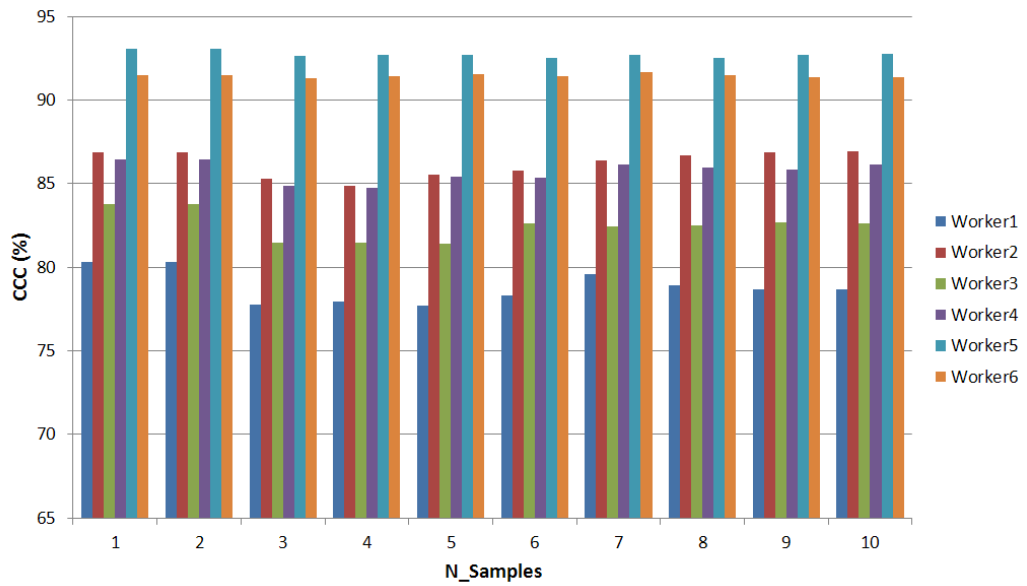


Figure B.5: Individual obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Batch)

Appendix C: Detailed results for the experimentation with mixed workload

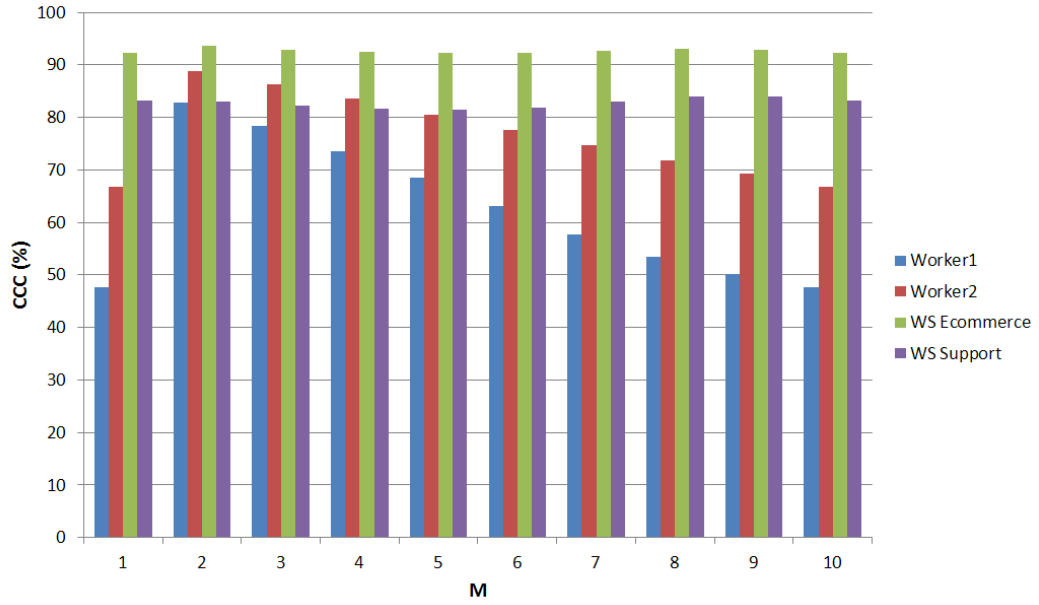


Figure C.1: Individual obtained CCC for different M values (moving average predictor) in Experiment 1 (Mixed)

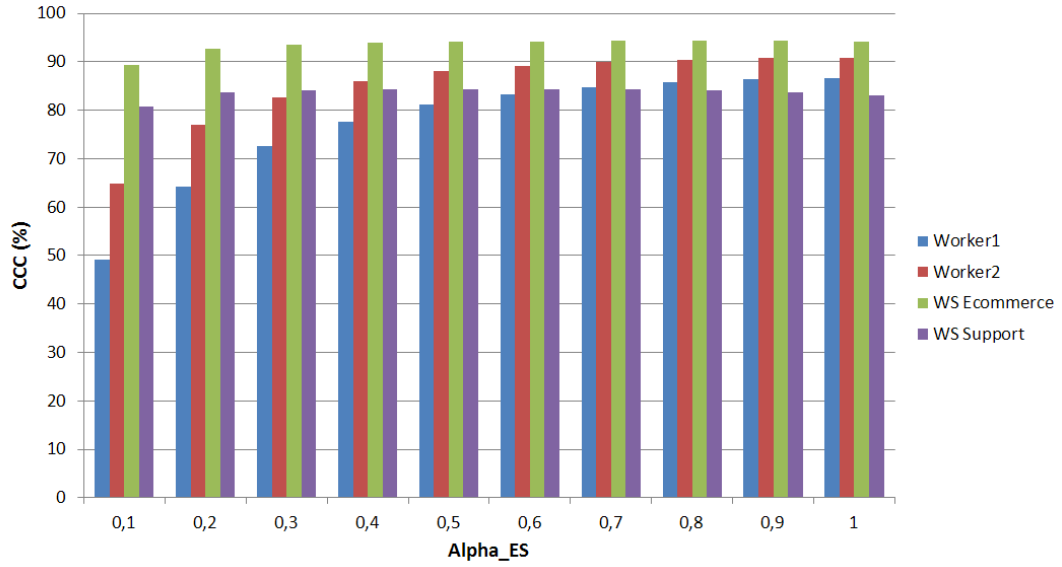


Figure C.2: Individual obtained CCC for different Alpha_ES values (exponential smoothing predictor) in Experiment 1 (Mixed)

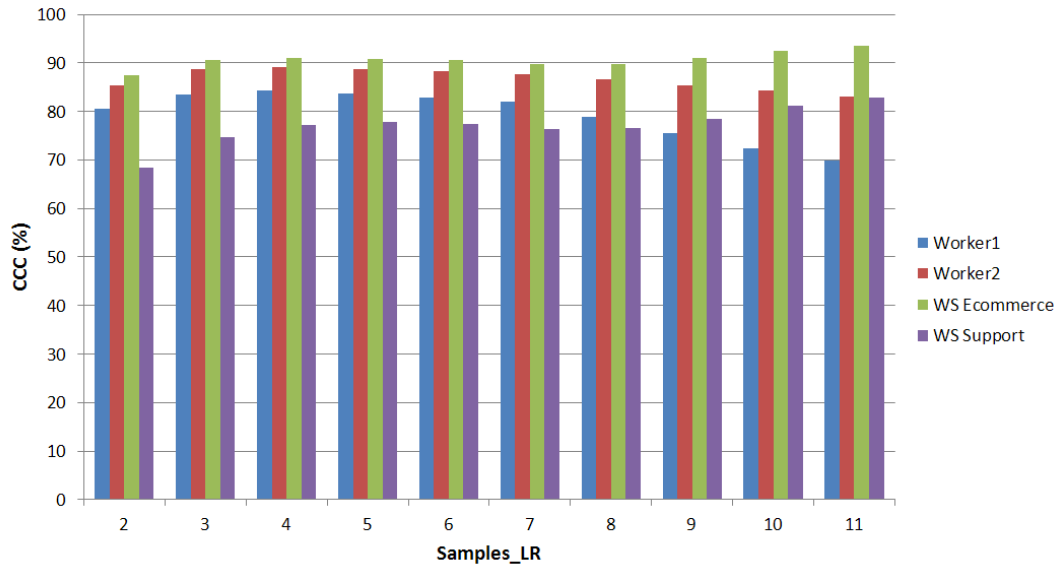


Figure C.3: Individual obtained CCC for different Samples_LR values (linear regression predictor) in Experiment 1 (Mixed)

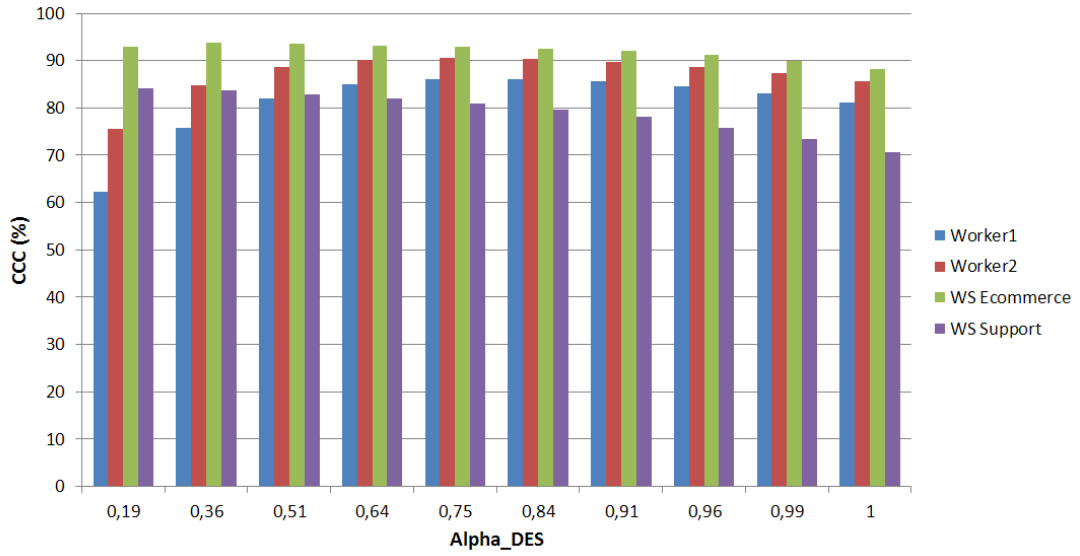


Figure C.4: Individual obtained CCC for different Alpha_DES values (double exponential smoothing predictor) in Experiment 1 (Mixed)

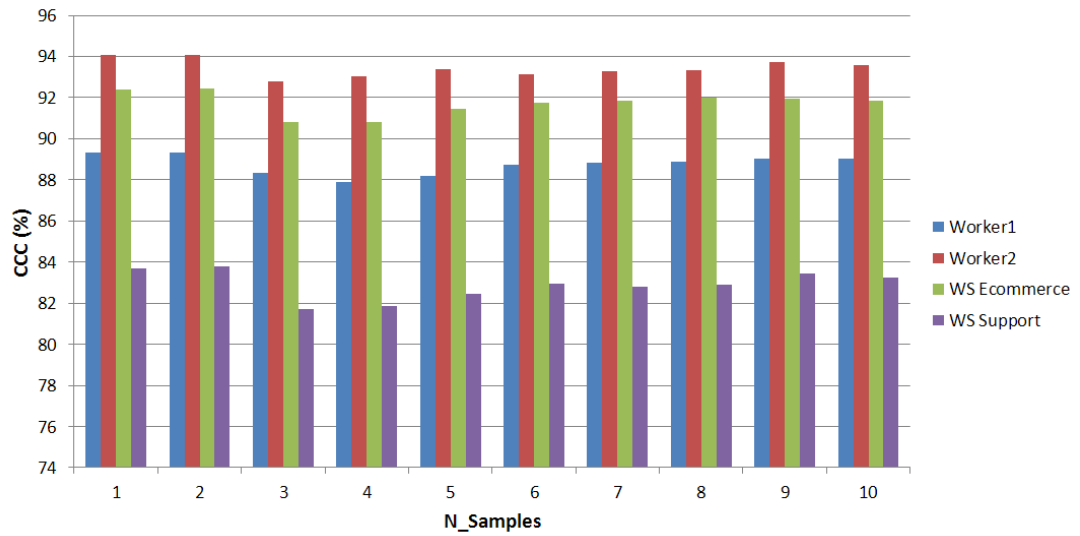


Figure C.5: Individual obtained CCC for different $N_Samples$ values (determines the “best forecast”) in Experiment 2 (Mixed)

References

- [1] OPTIMIS EU project. Optimis - optimized infrastructure services, 2012. URL <http://www.optimis-project.eu/>. ii
- [2] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. *NIST special publication*, 800(145):7, 2011. 1
- [3] Jeremy Sugerman, Ganesh Venkitachalam, and Beng-Hong Lim. Virtualizing i/o devices on vmware workstation’s hosted virtual machine monitor. In *USENIX Annual Technical Conference, General Track*, pages 1–14, 2001. 1
- [4] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. 2008. 2
- [5] Power Scorecard. Electricity from coal, 2000. 2
- [6] Íñigo Goiri, Ryan Beauchea, Kien Le, Thu D Nguyen, Md E Haque, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenslot: scheduling energy consumption in green datacenters. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 20. ACM, 2011. 2
- [7] San Murugesan. Harnessing green it: Principles and practices. *IT professional*, 10(1):24–33, 2008. 2
- [8] Michael Grubb, Christiaan Vrolijk, Duncan Brack, and Tim Forsyth. *The Kyoto Protocol: a guide and assessment*. Royal Institute of International Affairs London, 1999. 2
- [9] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA ’07, pages 13–23, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-706-3. 2, 17, 35

REFERENCES

- [10] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010. 2
- [11] Murray G Patterson. What is energy efficiency?: Concepts, indicators and methodological issues. *Energy Policy*, 24(5):377 – 390, 1996. ISSN 0301-4215. URL <http://www.sciencedirect.com/science/article/pii/S0301421596000171>. 2, 5, 13
- [12] R. Fromm, S. Perissakis, N. Cardwell, C. Kozyrakis, B. McGaughy, D. Patterson, T. Anderson, and K. Yelick. The energy efficiency of iram architectures. In *Computer Architecture, 1997. Conference Proceedings. The 24th Annual International Symposium on*, pages 327 –337, jun 1997. 5
- [13] Unixbench benchmark suite, 06 2011. URL <http://code.google.com/p/byte-unixbench/>. 5
- [14] Optimis - optimized infrastructure services, 07 2013. URL <http://optimis-project.eu>. 5
- [15] Emotive cloud, 07 2013. URL <http://emotivecloud.net>. 7
- [16] Opennebula, 07 2013. URL <http://opennebula.org>. 7
- [17] Openstack, 07 2013. URL <http://www.openstack.org>. 7
- [18] Occi: Open cloud computing interface, 07 2013. URL <http://occi-wg.org>. 7
- [19] Ovf: Open virtualization format, 07 2013. URL <http://dmtf.org/standards/ovf>. 7
- [20] Virt-top, 07 2013. URL <http://people.redhat.com/~rjones/virt-top/>. 8
- [21] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The Computer Journal*, 53(7):1045–1051, 2010. doi: 10.1093/comjnl/bxp080. URL <http://comjnl.oxfordjournals.org/content/53/7/1045.abstract>. 8
- [22] Chenguang Liu, Jianzhong Huang, Qiang Cao, Shenggang Wan, and Changsheng Xie. Evaluating energy and performance for server-class hardware configurations. In *Networking, Architecture and Storage (NAS), 2011 6th IEEE International Conference on*, pages 339 –347, july 2011. 9

REFERENCES

- [23] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M Stephens Jr, Karl Huppler, and Evan Haines. Energy benchmarks: a detailed analysis. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 131–140. ACM, 2010. 9, 10
- [24] Young Lee and Albert Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, pages 1–13, 2006. ISSN 0920-8542. 10, 11, 12
- [25] Anton Beloglazov and Rajkumar Buyya. Energy efficient resource management in virtualized cloud data centers. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 826 –831, may 2010. doi: 10.1109/CCGRID.2010.46. 11, 12
- [26] Gaurav Dhiman, Giacomo Marchetti, and Tajana Rosing. vgreen: a system for energy efficient computing in virtualized environments. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, pages 243–248. ACM, 2009. 11, 12
- [27] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. Joulesort: a balanced energy-efficiency benchmark. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 365–376, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-686-8. 16, 19
- [28] M.K. Patterson. The effect of data center temperature on energy efficiency. In *Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITherm 2008. 11th Intersociety Conference on*, pages 1167 –1174, may 2008. doi: 10.1109/ITHERM.2008.4544393. 16
- [29] Spyros Makridakis, Steven C Wheelwright, and Rob J Hyndman. *Forecasting methods and applications*. John Wiley & Sons, 2008. 32
- [30] stress Workload Generator, 01 2012. URL <http://weather.ou.edu/~apw/projects/stress/>. 35
- [31] cpulimit Tool, 07 2009. URL <https://github.com/opsengine/cpulimit>. 35
- [32] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A. Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 39–50, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0036-0. doi:

REFERENCES

- <http://doi.acm.org/10.1145/1807128.1807136>. URL <http://doi.acm.org/10.1145/1807128.1807136>. 35, 79
- [33] Scott Trent, Michiaki Tatsubori, Toyotaro Suzumura, Akihiko Tozawa, and Tamiya Onodera. Performance comparison of php and jsp as server-side scripting languages. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 164–182. Springer-Verlag New York, Inc., 2008. 41
- [34] Comp superscalar, 08 2013. URL <http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar>. 47
- [35] David G Kendall. Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354, 1953. 49
- [36] I Lawrence and Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268, 1989. 53
- [37] Dennis Abts, Michael R Marty, Philip M Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 338–347. ACM, 2010. 79