POLITECNICO DI MILANO
DEPARTMENT OF ELECTRONIC, INFORMATION, AND
BIO-ENGINEERING
DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

MEASURING AND IMPROVING ENERGY EFFICIENCY OF A DATA CENTER IN A SELF-ADAPTIVE CONTEXT

Doctoral Dissertation of:
Monica Vitali

Supervisor:
Prof. Barbara Pernici

Tutor:
Prof. Donatella Sciuto

The Chair of the Doctoral Program:
Prof. Carlo Fiorini

2013 – XXVI

Abstract

THIS thesis proposes a methodology and novel techniques to assess and improve Energy Efficiency (EE) in a data center environment, while respecting the constraints established in the Service Level Agreement (SLA) with clients.

This work has been motivated by the recent attention towards green energy and renewable resources, and by the growing demand of energy coming from the Information Technology (IT) sector. This demand is so high to be comparable with the one of the major countries and it is expected to keep growing in the next years.

In this work we try to understand the current trends towards a greener asset for data centers and how it is possible to understand the current level of efficiency of a data center while suggesting some repair actions for improving it. We propose a goal-oriented approach where the state of the system is assessed using a set of indicators. This set is composed by some classical indicators, coming from the state of the art, and some new indicators, specifically defined to better describe the efficiency of a data center. These indicators are subjected to thresholds that are used as goals of our system. We propose a self-adaptive context-aware framework, where we learn both the relations existing between the indicators and the effect of the available actions over the indicators state. The system is also able to respond to changes in the environment, keeping these relations updated to the current situation. Results have shown that the proposed methodology is able to create a network of relations between indicators and to propose an

effective repair action to contrast suboptimal states of the data center. We also propose a model for simulating the data center behavior which can be used to conduct what-if analysis, allowing testing new configurations in a safe environment and predicting future states of the system.

The proposed framework can be an important tool for assisting the system administrator in the management of a data center oriented towards EE, showing him the connections occurring between the sometimes contrasting goals of the system and suggesting the most likely successful repair action(s) to improve the system state, both in terms of EE and Quality of Service (QoS).

Riassunto

In questa tesi si propone una metodologia e alcune tecniche innovative per stimare e migliorare l'efficienza energetica nei data center, tenendo in considerazione i vincoli pattuiti con i clienti nel Service Level Agreement (SLA). Lo studio qui presentato è motivato dalla recente attenzione verso il risparmio energetico e le risorse rinnovabili, e dalla crescente richiesta di energia derivante dal settore delle tecnologie dell'informazione (IT). Tale richiesta è talmente elevata da essere comparabile con quella dei principali Paesi e continua a crescere di anno in anno.

Questo lavoro analizza le tendenze attuali nell'ambito del miglioramento energetico nei data center e le tecniche disponibili per la stima del livello di efficienza energetica dello stesso. Propone inoltre delle azioni di riparazione che possano migliorare tale livello. Si propone un approccio basato sugli obiettivi, dove lo stato del sistema è valutato usando un insieme di indicatori. Tale insieme è composto da indicatori classici, derivati dallo stato dell'arte, e da nuovi indicatori, definiti specificatamente per descrivere l'efficienza all'interno di un data center. Gli indicatori selezionati sono soggetti a delle soglie, il cui soddisfacimento rappresenta l'obiettivo del sistema. In questo lavoro si propone l'uso di un sistema adattivo e basato sul contesto, dove le relazioni tra gli indicatori e l'effetto delle azioni sugli stessi vengono appresi in modo automatico. Il sistema è inoltre in grado di reagire ai cambiamenti nell'ambiente circostante, aggiornando di conseguenza le relazioni apprese. I risultati mostrano che la metodologia proposta è efficace nel creare una rete di relazioni tra gli indicatori e nel proporre azioni di

riparazione efficaci per contrastare stati sub-ottimali del sistema. Per simulare il comportamento del data center si è inoltre implementato un modello che può essere usato per condurre analisi di tipo "what-if", permettendo la verifica di nuove configurazioni in un ambiente protetto e consentendo di predire stati futuri del sistema.

Il sistema proposto può essere un importante strumento per l'amministratore di sistema nella gestione di un data center in cui si ponga attenzione all'efficienza energetica, informandolo sulle connessioni esistenti tra i diversi obiettivi (talvolta contrastanti tra loro) e suggerendo l'azione di riparazione che con più probabilità è in grado di migliorare lo stato del sistema, sia da un punto di vista dell'efficienza energetica che da un punto di vista della qualità del servizio.

Contents

Preface

THIS thesis contains the result of research undertaken at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB) of the Politecnico di Milano, during my Ph.D. in Information Technology.

The work, under the supervision of Prof. Barbara Pernici, started in October 2010 and was developed till the end of 2013. It was focused on Energy Efficiency in data centers and on adaptation techniques for the energy consumption reduction. During the thesis work, I was involved in the GAMES (Green Active Management of Energy in IT Service Centres) and ECO_2Clouds (Experimental Awareness of CO_2 in Federated Cloud Sourcing) European projects. This gave me the opportunity of meeting and cooperating with researchers of the European community and of using partners systems for obtaining more experimental data.

During the Ph.D. I had also the opportunity of spending a period abroad at the Massachusetts Institute of Technology (MIT). I was hosted by the EVO-Design Optimization group, then renamed ALFA (Anyscale Learning For All), lead by Prof. Una-May O'Reilly. I spent there the fall term, between September 2012 and January 2013. During this period I had the opportunity to learn more about machine learning and adaptation techniques that I applied in this thesis. I also developed a model for simulating a data center described in one of the chapters of this thesis.

The research presented in this thesis was partially funded within the framework of the GAMES (http://www.green-datacenters.eu/, contract number ICT-248514) European project, from October 2010 to September 2012,

and within the framework of the ECO_2Clouds (http://eco2clouds.eu/, contract number 318048) European project from July to September 2013. Other funds came from the Italian Ministry of Education, University and Research (MIUR) from October 2012 to December 2013. The visiting period at MIT was funded by the Roberto Rocca fellowship program.

CHAPTER 1

Introduction

Indice

THE general interest about environmental issues is continuously growing. One of the main contributors to pollution and energy waste is Information Technology (IT). The growing rate of the technology diffusion has an impact for the whole life-cycle of the hardware involved (manufacturing, usage, and disposal). In everyday life, the amount of technology used is quickly increasing. The number of technological devices with which we deal everyday and the amount of time spent using these devices is considerable. The impact of technology and its growing rate is easy to understand at the everyday scale. This phenomenon is even more significant at a larger scale. Let's consider data centers. Data centers are

used for housing computing systems of different kinds. When dealing with data centers, a lot of side components have to be considered. The cooling system is in charge of keeping the temperature low enough to allow the servers in the data center to work properly. Of course, cooling consumes energy. Backup power facilities are also needed in order to ensure that the applications or services running in the data center keep working in case of a temporary lack of energy. Even these devices have an impact in terms of energy. We should also consider the storage systems, where data are usually duplicated for disaster recovery, and all the other computational resources, such as networking devices, power distribution, and so on. According to this, we can classify data center components into three categories:

- Temperature controlling devices: it includes heating and cooling systems, and ventilation. This category comprises all the devices involved in the management of the temperature inside the data center. Servers are highly sensitive to high temperatures and their performances can significantly decrease or they can get damaged if there is not enough control in this sense. For this reason, this category usually consumes a considerable amount of energy;

- Power supply devices: it consists in the power supply system, including the distribution of the power to all the devices in the data center and the storage of energy for reacting to eventual black outs, ensuring IT service continuity, availability and reliability of the hosted applications;

- IT Equipment: it includes all the IT devices, including servers, storage, and network devices. Each of these devices needs energy for running. The amount of energy consumed is usually dependent on the usage rate, even if a fixed amount of energy can be also consumed in stand-by mode.

Several solutions have been proposed for reducing the energy consumption of the first category, and some studies are trying to optimize the second one. In the present thesis, we are going to focus on the last category: the IT equipment.

The reduction in the cost of hardware has resulted in the employment of a large amount of servers, that usually run under their capacity. However, each single server consumes energy when active, and only a part of this energy depends on how much the device is used. In spite of this, it seems that the main focus for data center administrators is currently on performance

instead of efficiency, aiming at the real time retrieval and processing of a big amount of information over the internet. In order to have a response time as low as possible, data centers are spread world wide and data are significantly duplicated in a global scale. The cost of this practice in terms of energy is considerable.

Given these premises, it is evident that a lot can be done to improve efficiency in an IT system, and some efforts have been already made in this direction. Information Systems (ISs) are being influenced by the issue of reducing pollution and energy consumption and new fields are rising dealing with this topic. One of these fields is Green IT. Green IT studies methodologies and techniques to face the problem of the fast increment of energy consumption and CO_2 emissions in the field of IT. The role of IT towards Energy Efficiency (EE) is twofold:

- IT is a consumer of energy. Each component of the IT category needs energy to perform its job and, as we discussed before, energy consumption is also involved in all the phases of the life-cycle of IT;
- IT is also an effective source of solutions for monitoring and reducing energy consumption. Technology represents a necessary help for monitoring the energy consumption of an organization and for implementing solutions to reduce this consumption.

This work has been motivated by this recent attention towards Green IT and Energy Efficiency, and by the growing demand of energy coming from the IT sector. This demand is so high to be comparable with the one of the major countries and it is expected to keep growing in the next years. In this work we focus on efficiency considering the data center level. Several perspectives can be considered when dealing with efficiency at this level. For instance, the efficiency of the hardware employed has an important effect on energy consumption. Different devices of the same typology can have a significantly different efficiency and this difference should be considered in the selection of the components of the data center. Also the operating system and the software level can play their role in improving the efficiency of a system. Recently, virtualization and cloud computing have changed the way resources are handled in a data center, allowing a more dynamic redistribution of computational resources among the several applications hosted by the data center or the cloud. As a consequence, the waste of resources is significantly reduced.

A deep study of the state of the art related to the Green IT topic has highlighted that several approaches exist, covering the topic from different

perspectives. However, they usually focus on limited aspects of the problem, showing a lack of a general and comprehensive view. We think that this general view represents the only way to obtain a significant outcome.

The focus of this thesis is most of all on data centers, with a special attention to the contribution of Service Oriented Architectures (SOAs) to energy consumption and performance. We face the problem of EE by proposing a comprehensive approach. The approach includes the definition of the goals for the organization, the selection of metrics for the assessment of the system state, and the definition of techniques for the improvement of the system. In order to do so, we propose the use of a goal-oriented model.

1.1 Research Challenges

The open research challenges that are investigated in this thesis are described in this section.

1. *Which is the impact of the application level on the efficiency of an IS?*

   In this thesis we argue that the application level plays an important role in the improvement of the efficiency of an IS. Let's consider a very common scenario, where we have an already established data center running some applications. In an effort of improving the efficiency of the system, it is unlikely that the data center owner wants to invest in a new and more efficient equipment to reach this goal. It is more likely that the strategy will consist in a better utilization of what he already has. This is when applications become important in this scenario. We argue that a better management of resources is possible starting from the deployment of the application. How to improve EE from an application perspective is an open issue that we are going to investigate here.

2. *How it is possible to state when an IS is green? Which metrics can be used as a reference for assessing the efficiency of an IS? How it is possible to measure energy at different levels?*

   When dealing with efficiency, a methodology is needed for assessing the system state. The improvement of the system should be measurable in order to understand the current trends. Also some goals should be set and the system should be directed towards these goals. The issue is to select a significant set of goals for assessing the efficiency of an IS. These goals can be directly related to the energy consumption, but we argue that this is not enough. In fact, other aspects can

indirectly impact on the efficiency of the system and should be considered in the estimation of efficiency. We are going to investigate which metrics can be used to express the EE of an IS and how it is possible to set some reference levels for them. Without reference, it would be impossible to understand if we are moving in the right direction.

Since we are dealing with applications, we need a set of metrics to help us in understanding the efficiency at the application level. This is not a trivial task. In fact, most of the metrics available are not properly defined from an application perspective and retrieving some useful information at the application level can be not trivial. Since measuring energy at the application level is still an open issue, some assumptions have to be made to obtain an estimation for the energy related metrics at this level.

3. *What can be done to improve Energy Efficiency in an IS? Is it possible to suggest a general adaptive approach for improving efficiency?*

   Once the state of the system can be assessed using proper metrics, some techniques have to be proposed for dealing with inefficiencies and possible improvements. Several techniques can be used and their impact on the system state is not always clear or straightforward. In fact, it can be dependent on the context of application. Given these premises, the issue is to understand which techniques can be applied to improve the efficiency of a data center and to comprehend and predict the outcome of their application under different contexts. Moreover, we argue that an adaptive approach should be implemented for keeping the system state under control and for applying proper repair actions to fix issues related to Energy Efficiency, without neglecting the effects on other performance parameters. With an adaptive approach the system can be automatically driven towards an efficient asset. However, how it is possible to select the proper techniques and which is the impact of the context on this decision is an open issue.

4. *How it is possible to protect the system from wrong decisions? How a simulation environment can help?*

   In an adaptive approach, where the effect of improvement techniques can be uncertain, a wrong decision can damage the performance of the system. We argue that using a simulation environment, which models the principal features of the system, can help in predicting outcomes while preserving the real system from performance issues. The simulation system can be used to perform training sessions and

to test solutions giving a general idea of the outcomes of decisions related to the IS organization and management. Simulation systems available in the state of the art are not able to properly represent the features of a data center, so new solutions are needed.

1.2 Problem Statement

In this work we try to understand the current trends towards a greener asset for data centers. First of all we analyze available techniques for assessing the efficiency and performance of a data center and the several levels at which this assessment can be performed. Starting from the Green Grid Data Center Maturity Model (DCMM) we propose a set of metrics and define some constraints over their values that can be used for detecting suboptimal configurations of the system. The aim of the constraints is to bring the system towards a higher maturity level in terms of efficiency while avoiding violations of Quality of Service (QoS) constraints. In order to avoid violations, we propose an adaptive approach based on a goal-oriented model. The model is composed of two layers: the goal layer and the treatment layer. In the goal layer there are the goals of the system (namely constraints satisfaction) and the relations existing among them. The treatment layer is composed of a set of repair actions that can be used to react to suboptimal situations (namely constraints violations). Actions are connected to the goal level by annotated links indicating if the action has a positive or negative effect on the goal.

We propose a self-adaptive context-aware framework. In order to deal with a dynamic environment, we propose a technique to automatically learn relations among the goals of the system, represented using a Bayesian Network (BN). A new technique is proposed for learning the structure of a BN starting from continuous variables. We also automatically learn the links connecting actions and goals by proposing an adaptive approach called Adaptive Action Selection (AAS), inspired to the Adaptive Operator Selection (AOS) approach. The system is able to respond to changes in the environment, keeping these relations updated to the current situation. Results have shown that the proposed methodology is able to learn both the goal-to-goal connections and the action-to-goal connections. The AAS algorithm is also used to suggest the most likely effective repair action given a specific context, considering all the current violations but also the side effects of the actions over the other goals. We also propose a model for simulating the data center behavior that can be used to conduct what-if analysis,

allowing testing new configurations in a safe environment and predicting future states of the system.

The proposed framework can be an important tool for assisting the system administrator in the management of a data center oriented towards Energy Efficiency. Connections among goals can be used to predict possible outcomes of some modifications in the system and can put light over unexpected connections occurring between the sometimes contrasting goals of the system. Moreover, action-to-goal connections enable a repair mechanism able to suggest the most likely successful repair action(s) to improve the system state, both in terms of EE and QoS. The AAS algorithm is designed to dynamically update its knowledge about the action effects, fitting the features of a system that can frequently change in time.

1.3 Research Context

In this section we describe the context in which we are going to implement our goal-oriented model for EE. As described before, we are dealing with data centers. In our vision, a data center is composed of several servers, each one with different features in terms of amount and kind of components. As stated before, in our approach we focus on the application perspective for improving efficiency. For this reason we are not considering issues related to cooling or other facilities, but we look in details at applications and their IT infrastructure.

We are considering the context of SOAs, where the applications hosted in the servers are services that can be represented using a process. Each service is decomposed in a set of activities connected in a work-flow. We consider that the data center uses virtualization for hosting its applications and we consider that each single activity runs on a dedicated Virtual Machine (VM).

A monitoring system is available for monitoring the system behavior with sensors that can work at several levels:

- Data center level: sensors collect information about the general data center (e.g. energy consumption, number of active machines);
- Server level: sensors collect information about a specific server. Examples can be the percentage of usage of the resources in the server and the amount of energy consumed by it;
- Virtual Machine level: the monitoring system collects data that can be obtained by the hypervisor or directly by the Operating System (OS)

of the VM as if they were physical machines;

- Application level: at this level, sensors collect information related to the application, such as response time, throughput, and availability.

An incoming load is handled by dividing the work into the different activities according to the work-flow definition. The use of virtualization enables a dynamic allocation of the resources available in the data center.

1.4 Contributions and Results

This section highlights the main contributions of the presented thesis to the Information Systems, Green IT, and Machine Learning scientific communities.

Hierarchical classification of Green IT approaches This thesis provides a deep study of the fields of Green IT and Green IS and of the state of the art of these fields. Instead of limiting this analysis to a simple quote of available results, we propose a hierarchical classification of the different perspectives and we also propose a methodology for combining them in order to have a comprehensive approach towards EE.

Metrics selection and proposal Several metrics are analyzed in order to find a set which is representative of the QoS and EE of an IS. We also evaluated metrics that have an indirect impact on these two aspects, affecting the system state. Also new metrics have been introduced and tested, focusing attention on the application usage perspective.

A goal-oriented model for efficiency management A model is proposed and tested based on a goal-driven approach. The aim of the model is to provide an adaptive environment able to automatically adapt to suboptimal situations by enacting adaptation strategies that impact directly or indirectly over efficiency and quality.

Representation and learning of indicators relations Using a Bayesian Network representation, we express the relations occurring among the various metrics used for monitoring the system state. These relations are very important for predicting the future behavior of the system and for conducting what-if analysis reasoning. The network has been learned from monitored data, without using any kind of knowledge coming from experts. Using machine

learning techniques, we were able to learn the structure of a BN starting from continuous variables.

Adaptive strategy selection Effects of actions over EE and QoS are automatically learned providing an adaptive and dynamic environment for managing these aspects of an IS. In this way, the framework is able to suggest the best adaptation action without any intervention from experts and to automatically adapt to modifications in the environment that can modify the impact of the actions enactment.

Modeling service execution in data centers A mathematical model is provided for simulating the behavior of a service oriented data center. This model includes features typical of the considered domain (the service oriented data center), unavailable in other models. The model enables to run simulations of the data center behavior under several loads and with different configurations. This feature is particularly important during the learning phases. In fact, in this way it is possible to test new actions and new configurations in a protected environment and to observe possible outcomes on the system, avoiding to undermine the performance of the real system.

1.5 Outline

The thesis is organized into three parts. In Part 1 we analyze the state of the art. This part is divided into two chapters:

- Ch. 2 shows a wide selection of contributions related to the Green IT sector by proposing a hierarchical classification of these approaches. Three main categories are identified: assessment of the system state, measurement and estimation of EE, and techniques to improve EE;

- Ch. 3 focuses on adaptation and machine learning techniques with a special attention towards SOAs. We introduce the issue of adaptation in SOAs, discussing the application to service composition and to the management of QoS. Then we discuss Bayesian Networks and their main features and techniques available for learning the structure of a BN from data. Finally we introduce an adaptive approach called AOS for dynamically adapt a system towards a better configuration.

In Part 2 we focus on modeling and assessing Energy Efficiency:

- Ch. 4 introduces a selection of metrics that can be used for the assessment of the efficiency and QoS of a system and proposes a new category of indicators based on the resources usage from the application perspective;
- Ch. 5 proposes a model that can be used in the management of the efficiency of an adaptive system based on a goal-driven approach. Also a set of repair actions is defined for reacting to suboptimal situations.

In Part 3 we discuss some techniques for the improvement of the efficiency and quality in an IS:

- Ch. 6 proposes a methodology for learning relations and influences among indicators used to assess the system efficiency by using a BN representation;
- Ch. 7 proposes an adaptive technique to learn the impact of several repair actions on the system state and to suggest the best actions to solve a critical situation;
- Ch. 8 proposes a model for a data center environment that can be used for simulation and what-if analysis in order to safeguard the real system during testing.

In Ch. 9 we summarize the approaches presented in this thesis and the results obtained while also outlining future developments and research paths.

1.6 Scientific Production

INTERNATIONAL JOURNALS

1. M. Vitali, B. Pernici, "A Survey on Energy Efficiency in Information Systems," *International Journal of Cooperative Information Systems*, IN PRESS
2. A. Kipp, T. Jiang, J. Liu, M. Fugini, M. Vitali, B. Pernici, and I. Salomie,"Applying green metrics to optimise the energy consumption footprint of it service centres," *International Journal of Space-Based and Situated Computing*, vol. 2, no. 3, pp. 158-174, 2012

INTERNATIONAL CONFERENCE PAPERS

1. M. Vitali, U.-M. O'Reilly, and K. Veeramachaneni, "Modeling Service Execution on Data Centers for Energy Efficiency and Quality of Service Monitoring," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC2013)*, 2013

2. C. Cappiello, P. Plebani, and M. Vitali, "Energy-Aware Process Design Optimization," in *Proceedings of the EuroEcoDC workshop, within the 3rd International Conference on Cloud and Green Computing*, 2013

3. B. Pernici, C. Cappiello, M. G. Fugini, P. Plebani, M. Vitali, I. Salomie, T. Cioara, I. Anghel, E. Henis, R. Kat, et al., "Setting energy efficiency goals in data centers: the GAMES approach," in *1st International Workshop on Energy Efficient Data Centers, within the 3rd International Conference on Future Energy Systems*, Springer, 2012

4. D. Chen, C. Cappiello, A. M. Ferreira, E. Henis, T. Jiang, R. Kat, A. Kipp, J. Liu B. Pernici, D. Sotnikov, and M. Vitali, "Usage centric green performance indicators," in *ACM SIGMETRICS Performance Evaluation Review*, Volume 39 Issue 3, pp. 92-96, December 2011

5. C. Cappiello, M. G. Fugini, A. M. Ferreira, P. Plebani, and M. Vitali, "Business Process Co-Design for Energy-Aware Adaptation," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pp. 463-470, 2011

6. D. Chen, C. Cappiello, A. M. Ferreira, E. Henis, T. Jiang, R. Kat, A. Kipp, J. Liu B. Pernici, D. Sotnikov, and M. Vitali, "Usage centric green performance indicators," in *Proceedings of the Green Metrics 2011 Workshop (in conjunction with ACM SIGMETRICS 2011)*, 2011

Part I

State of the Art

CHAPTER 2

Green Information Technology: a Classification of the Existing Approaches

Indice

Concerns about energy and sustainability are growing everyday involving a wide range of fields. Even Information Systems (ISs) are being influenced by the issue of reducing pollution and energy consumption and new fields are rising dealing with this topic. One of these fields is Green Information Technology (IT), which deals with Energy Efficiency (EE) with

a focus on IT. Researchers have faced this problem according to several points of view. The purpose of this chapter is to understand the trends and the future development of Green IT by analyzing the state of the art and classifying existing approaches to understand which are the components that have an impact on EE in Information Systems and how this impact can be reduced. At first, we explore some guidelines that can help to understand the efficiency level of an organization and of an IS. Then we discuss measurement and estimation of EE and identify which are the components that mainly contribute to energy waste and how it is possible to improve EE, both at the hardware and at the software level.

2.1 Introduction

The general interest about environmental issues is growing every day in a wide range of fields. In the last years, the problem of being green and of saving energy has been faced also in Information Systems, giving birth to a new research branch known as Green Information Technology (IT). Examining any IT system, it is evident that any single component of the system has a cost in terms of energy consumption or pollution. At the same time, IT is also an effective source of solutions for monitoring and improving Energy Efficiency (EE). This twofold role of IT for EE is analyzed in this chapter.

The purpose of this survey is to analyze the maturity of the approaches available in the state of the art about assessing and improving the sustainability in Information Systems, from a computer science and engineering point of view. A similar effort has been recently published in [1], where the authors faced the problem of EE from several perspectives: hardware, operating system, virtualization, and data center levels. The focus is on the hardware component contribution to energy consumption and on tools available to improve their efficiency.

On the other hand, the focus of this chapter is most of all on IT, with a special attention to the contribution of the Business Processes (BPs) and of the setting of the organization to both energy waste and saving. A thorough analysis of the literature allows us to identify three main areas. The first area is related to the definition of general instruments and guidelines to assess the maturity level of sustainability in an organization. The second one involves measurement and estimation of energy consumption of an entire Information System (IS) or of individual components, that can be both hardware and software. Available metrics to assess and monitor greenness

are also discussed. The last area focuses on the definition of a set of techniques that can be used to reduce the waste of energy, both at the hardware and at the software level.

The research questions that we want to inspect into this survey are: *What are Green IT and Green IS?*; *How it is possible to state when an IS is green?*; *How can we measure Energy Efficiency of an IS?*; and *What can be done to improve Energy Efficiency in an IS?*.

As EE in Information Systems is a recent scientific field, the authors have performed a systematic review of journals and conference proceedings. Only representative research approaches were selected based on the quality of their contribution and their significance and relevance to the scope of this article by also considering the respective publication year and the classification of the conference or journal in which the approaches were published. Different criteria were exploited to assess the quality of a document, such as the document expressiveness, richness, formality, and innovation. Research approaches of conferences or journals with low classification were not considered, unless giving an interesting contribution or perspective not detected in other papers with a higher classification. We examined the proceedings of the main conferences related to Information Systems and, at the same time, the first special sessions and special issues dedicated to the EE topic. We also examined the work of the best known researchers in the area, selecting relevant citations from their work in order to have a comprehensive view of each topic. The main keywords in the research of documents were: *Energy Efficiency*, *Green IT*, *Green IS*, *Energy Estimation*, *Energy Efficiency Improvement*, *Energy-Aware Information System*, *Energy Efficiency Assessment*, *Energy Metrics*, and *Energy-Aware Adaptation*.

The rest of the chapter is divided into four sections. First of all, an overview about efficiency in Information Systems and Information Technology is given with some definitions (Sect. 2.2). After that, three main areas are identified for dealing with greenness in Information Systems. This division is represented through a chart in Fig. 2.1. Each of the areas represented is analyzed in detail: Sect. 2.3 introduces the problem of assessing EE of an organization, Sect. 2.4 discusses approaches to measure energy in an IS, and Sect. 2.5 shows several techniques adopted to improve EE. Finally, Sect. 2.6 summarizes the concepts emerged from the analysis and it discusses future trends in Green IT.

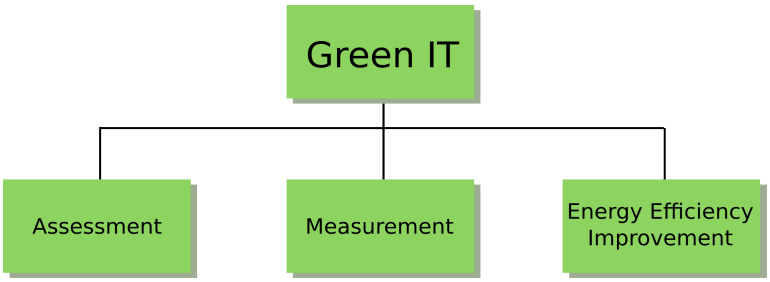


Figure 2.1: *Different approaches to Green IT*

2.2 An Overview of Green Perspectives for Information Systems

In this section we give some definitions and perspectives on Energy Efficiency both at the IT and IS level, trying to answer the research question "*What are Green IT and Green IS?*".

The most efficient driver for EE can be identified in governmental regulations. A big number of regulations exist and cannot be described in details in this work. To understand the amount of different existing regulations, a survey published in 2009 by the Organization for Economic Co-operation and Development (OECD) [2] analyzed 90 of them. Just to cite some relevant regulations, in the USA we can find Energy Star, developed by the US Environmental Protection Agency (EPA) [3], for producing energy efficient devices. In Europe, the Code of Conduct for data centers has been developed to give directions about how to improve their greenness [4].

Focusing on Energy Efficiency at the Information Technology level, the field of Green IT has been analyzed in the recent year. A definition for Green IT can be found in [5]:

> *"Green Information Technology (IT) is a systematic application of ecological-sustainability criteria (such as pollution prevention, product stewardship, use of clean technologies) to the creation, sourcing, use and disposal of the IT technical infrastructure as well as within the IT human and managerial practices."*

As it can be inferred from this definition, this field is really wide and can be faced at different levels. A detailed analysis of this problem is described in [6], where the authors examine green issues related to an IS at each level, looking at IT both as a source of pollution and as a solution to it.

According to [6], reasons of pollution due to IT are multiple. The computer pollutes because it uses energy, because it is made with toxic materials and because nowadays disposal of computers is so fast that a great amount of old machines needs to be demolished. So, in order to understand the impact of a machine on the environment, its whole life cycle has to be considered.

Therefore, the approach to Green IT should cover four areas [6]:

- *Usage* of IT: energy efficient machines are needed and good practices should be enacted. At the level of personal computers, some simple practices can dramatically reduce energy consumption. For instance, energy saving features can be enabled, or the computer could be turned off when it is not in use. Even the use of screen savers can reduce energy consumption and in an organization with a large amount of employees this saving can be considerable. It is worth noticing that a computer in power saving mode consumes about 25% of the normal mode power. At the level of data centers an efficient cooling system or the migration from old mainframes to more recent servers could enable EE improvement. Another good practice can be virtualization, which optimize resources usage;

- *Design* of energy efficient components and equipments: some techniques to improve EE could be used. For instance, switching from single to multiple core architectures improves EE. The cache could be divided in sub-segments that could be powered only when needed. About storage, using less devices with a greater capacity can contribute to reduce energy usage by the 50%;

- *Manufacturing* components with the minimal possible environmental impact: the materials chosen to build servers or personal computers have a big impact on Green IT;

- *Disposal* of IT: reusing old machines and appropriately disposing of the ones that are useless. At the end of the life cycle, reusing and recycling of old machines should be mandatory.

In addition, IT can also be used as a tool to improve EE in other fields. In fact, IT gives the ability to build models for predicting the behavior of a given system, to run simulations for predicting its energy consumption under given circumstances, and to design decision support tools that can be used in a wide range of areas for reducing energy consumption. This concept has been synthesized in [7] with the name of *energy informatics*.

Energy informatics is concerned with the analysis, design and implementation of Information Systems able to improve the EE of a system. The core idea is that enriching the system with monitored information the energy can be reduced. Actors involved in a typical system can be divided into three categories: suppliers who provide the service consuming energy, clients who ask for the services, and governments who intervene enacting some regulations to lead suppliers toward a more efficient management of the resources. It is important to find an agreement between service providers and clients. Usually a Service Level Agreement (SLA) is ratified between them that expresses constraints about the quality that the service has to respect to satisfy the client. When EE becomes part of the equation, a trade off between EE and the quality of the service might be needed and SLA should to be renegotiated.

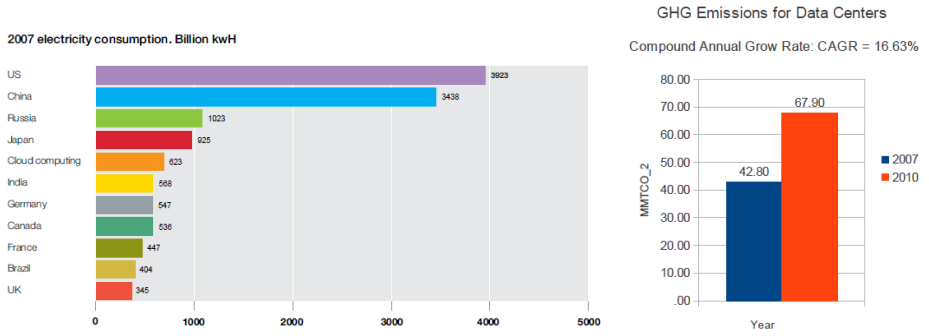
This twofold aspect of Green IT for sustainability is studied by Vom Brocke [8]. He moves the attention from IT to Information Systems, touching the area of Green IS [9]:

> *"Green Information Systems (ISs) initiatives generally focus on designing, building and operating sustainable IS in order to improve the sustainability of organizations."*

In his analysis of sustainability, he focuses on BPs, as discussed in a panel section on sustainability of Information Systems [10]. Vom Brocke claims that an improvement of EE in an organization is possible only if there are enablers for the adoption of sustainability practices. These enablers are classified in four major categories: definition of a clear *strategy for sustainability*, *organizational support* from the top management, *motivation* for applying strategies, and *traceability* in order to be able to see the results of the adopted practices. This approach shifts the focus from a technological perspective to an organizational one. A proximate area to Green IS is the topic of Green Business Process Management (BPM), defined and analyzed in details in [9]:

> *"Green Business Process Management (BPM) concerns the understanding, documenting, modeling, analyzing, simulating, executing, and continuously changing of business process with dedicated consideration paid to the environmental consequences of these business processes."*

In [11], Green BPM is seen as the area of intersection between Green IS and BPM.



(a) Electricity demand of ICT compared with countries [12]

(b) ICT GHG emissions growth

Figure 2.2: *GHG emissions and energy demand of ICT*

Given these premises, in the following sections we analyze approaches to manage EE in IT, with an Information Systems perspective.

2.3 Assessing the Energetic Maturity of an Information Systems

As discussed before, IT systems have a relevant impact on the total amount of CO_2 emissions and more and more companies are now starting to deal with this problem. As reported by Greenpeace in [12], in 2008 Information and Communications Technology (ICT) was responsible for the 2% of the global greenhouse gas (GHG) emissions. Moreover, considering the electricity demand of data centers and cloud as the one of a country, and comparing it with the one of the other countries, the former would be in the fifth position. This demand is also growing: in an era where recession made electricity consumption stable, the demand from data centers and clouds has been growing of the 56% from 2005 to 2010. In [1], authors report data from the EPA, stating that GHG have increased from 42.8 million metric tons of CO_2 ($MMTCO_2$) in 2007 to 67.9 $MMTCO_2$ in 2011 (see Fig. 2.2).

The issue of CO_2 emissions is related but separated from EE. In fact, improving EE results in a reduction of CO_2, but this is not the only aspect to be considered. According to Greenpeace, also the source of energy has to be considered, and big companies have the responsibility of pushing towards the adoption of policies for renewable energy. In the report, Greenpeace analyzes the trends in energy consumption of well known companies in the IT sector (e.g. Google, Apple, Akamai, and Amazon), reporting their

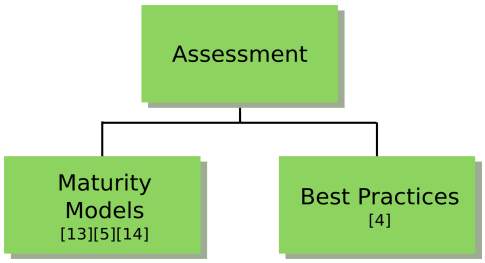


Figure 2.3: *Assessment of energy efficiency*

clean energy index, compared to the percentage of energy originated from non renewable sources. The levels of greenness of these organizations are significantly different, due to different investments that their organic has reserved to the green perspective.

The effort in greening an organization can be motivated by different goals. Some companies are constrained by new regulations about environmental impact and they need to modify their behavior to respect these regulations. Other companies are trying to become greener for reasons related to public image. Nowadays, customers are getting sensitive to environmental issues and are starting to prefer companies which demonstrate an effort in this direction. As an example, the big company Facebook, was moved to adopt a green direction for its data center energy provisions by campaigns conducted by Greenpeace and several user associations[1].

In this section we try to answer the research question "*How it is possible to state when an IS is green?*". The problem for companies that set off towards a greener asset is twofold. First, they need to be able to understand their current maturity level in sustainability. For maturity level we mean the level at which sustainability is addressed inside the organization; this level should be measurable using some comparison parameters. Second, they need some guidelines to decide what to do to improve their status. These two trends, discussed in this section, are shown in Fig. 2.3. This effort requires the participation of a large number of subjects that are involved in the process. Different approaches exist looking at different aspects of an organization: while some scholars analyze the whole organization, others focus their attention only on IT. The general approaches analyze the production process, people involved in the organization and managerial factors. Approaches focused on IT usually analyze different levels of the infrastructure such as the facilities (power, cooling, etc.) or the resources (disks, network,

[1] http://www.greenpeace.org/international/en/news/features/Victory-Facebook-friends-renewable-energy/

etc.). In the rest of the chapter we see that dealing with only one of these aspects can be not enough to improve the efficiency of the organization, so both IT and organizational factors, such as the BPs, need to be considered.

Some efforts made for understanding the sustainability maturity level of an organization have generated some capability maturity models related to sustainability. A capability maturity model can be seen as a methodology used to develop and refine an organization's aspect, in this case its sustainability level.

The Green Grid consortium[2] proposed a capability maturity model addressed to data centers. The consortium is composed of companies, government agencies, and educational global institutions that are dedicating their research to improve EE in data centers. The model, presented in [13], captures some relevant aspects related to data center EE. The maturity level of an organization can be one of six levels, from 0 to 5. Level 0 represents a company which is not enacting any effort for sustainability, between 1 and 2 characteristics of a data center actuating sustainable practices are described, with level 2 being the state of the art best practice. Levels from 3 to 5 express future sustainable data center and level 5 is expected to be reached in 2016. Moreover, under each level, a distinction between different aspects of the data center is made. Two main areas are analyzed: facility and IT. The facility area stresses the attention on cooling and power, considering also the management of the data center. The IT area stresses the importance of compute, storage, and networks components including also a more general category which describes the level of utilization and the workload distribution in the servers. At each level, each of these areas describes which should be the state of the data center in order to belong to that level of maturity. As an example, considering the general IT category, a state of the art data center should use virtualization and consolidation to handle its workload and should have a Central Processing Unit (CPU) average usage rate of at least 20%. CPU average utilization should incrementally increase to 60% to reach level 5. Mapping an organization in the model makes it simpler to understand which efforts are needed to move from one level to another. The model can be an important tool for the organization to understand which are the parts of the data center that are excelling and which parts present the largest opportunity for improvement.

Another approach to sustainability in Green IT is presented in [5]. Here, the authors define the G-readiness concept as *"an organization's capability in Greening IT (that is applying environmental criteria to its IT technical*

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

infrastructure as well as within its IT human and management practices across the key areas of IT creation, sourcing, operations, and disposal) in order to reduce IT, BP, and supply chain related emissions, waste and water use; improve EE; and generate green economic rent". So, greening IT should involve both technology and related people. Also here five categories are distinguished which are *attitude*, *policy*, *practice*, *technology*, and *governance*. The authors also divide the readiness into 7 levels and represent the G-readiness of an organization using a graph in which the seven levels and the five categories are depicted. Through this graph, the organization is able to understand which are the fields that need more attention.

Starting from this model, a new maturity model has been proposed by the Innovation Value Institute (IVI) as complementary of pre-existing approaches. The model is described in [14]. A framework is presented for increasing the sustainability in ICT executing four key actions: (i) defining the scope and goal of a sustainable ICT for the company; (ii) understanding the current capability maturity level; (iii) developing and managing sustainability in a systematic way using some capability building blocks; (iv) assessing and managing progress over time. The scope depends on the strategy of the organization that can range from satisfying customers requests about greenness and sustainability, to gaining a role of leadership in the field. Even in this case, the organization needs to understand its own maturity level that is distinguished into five levels: *initial* (where sustainable ICT is not considered as a business asset), *basic* (partially considered), *intermediate* (a strategy exists with plans and priorities), *advanced* (sustainable ICT is a core component of the business strategy), and *optimizing* (the adopted strategies are considered as an example for other organizations). As in the Green Grid Maturity Model, each level is decomposed into different categories that cover different aspects of the system. In this case, the categories highlighted by the model are *strategy and planning*, *governance*, *people and culture*, and *process management*. Each of them is decomposed into a small number of subcategories to focus on more detailed aspects. Each IT process consists of building blocks against which maturity levels are defined. The framework can be used to assess the maturity of the organization and systematically improve it in a measurable way to meet the sustainability objectives.The importance level of each category can also be considered by the organization and the comparison with its maturity level can lead the organization to focus its efforts on a category rather than an another.

Considering approaches to assessment based on best practices, the Codes of Conduct, proposed since 2007 by some researchers of the European Commission, proposes an action with the aim of improving the EE of ICT. Some specific documents give some indications related to data centers. In [4] a set of best practices is described and classified. Their role is to identify and implement measures to improve the EE of data centers. All organizations willing to obtain a *participant* status need to implement the non optional best practices. A score is associated to each best practice, indicating its impact and consequently its priority. The score is between 1 and 5. For all the practices, participants have to specify if they are already implemented or endorsed to outsource partners, or if there is a plan to implement them. Otherwise, the participant has to specify the reason why the practice is not implemented or is not applicable to its data center. Practices are grouped in categories which are identified in the document as: *Data Center Utilization, Management and Planning*, *IT Equipment and Services*, *Deployment of New IT Services*, *Cooling*, *Data Center Power Equipment*, *Other Data Center Equipment*, *Data Center Building* and *Monitoring*. The authors also define a scope for the suggested practices that can be requested only for part of the data center. Five scopes are identified: entire data center, new software, new IT equipment, new build or retrofit, and optional tasks. This last category covers all the practices which are suggested, but not required.

A comparison between the presented models is summarized in Tab. 2.1, which reports the main features of the analyzed models. The level of operation can be of two kinds: Data Center when the model is used to assess the maturity at the Data Center level; Organizational in case of a more general approach including several aspects of the whole organization. All approaches divide the assessment of the sustainability levels in different categories and subcategories, but the Green Grid maturity model seems to give more detailed and technical information to help the organization to place their system in one level or another, while the other approaches describe categories and levels in a more qualitative way.

Another issue for a company should be how much to invest in Green IT. This decision should depend from the asset of the company. This problem is discussed in [15]. The paper analyzes the factors that can influence the decision of the kind of governance for Green IT of an organization. Three levels of governance are identified: (i) *centralized*: no coordinator for Green IT but only an extension of other jobs, low importance; (ii) *federal*: a coordinator dedicated to the Green IT management, medium importance;

Table 2.1: *Comparison Between Assessment Models for Green IT*

Maturity Model/ Best Practices	Level of operation	Number of Levels	Number of Categories	Level of Detail
Green Grid [13]	Data Centers - Facility & IT	6	2 main categories + several subcategories	high, each subcategory contains detailed information
G-Readiness Framework [5]	Organization - Business Processes & IT	7	5 and subcategories	low, levels are seen as a sort of score for each sub category
IVI Maturity Model [14]	Organization - Business Processes & IT	5	4 and subcategories	low, levels are seen as a sort of score for each sub category
EU Code of Conduct [4]	Data Centers - Facility & IT	N.A.	8	high, each practice highlights a needed action to be implemented

(iii) *decentralized*: a committee as coordinator for Green IT, high importance, Green IT can lead the organization strategy. The contingency factors individualized are: competitive strategy, firm size, organization structure, performance strategy, environmental impact, environmental strategy, IT infusion, and IT diffusion. Supported by the model, companies can select the most promising Green IT governance form or compare their existent governance against the model, which can confirm or provide strategic directions for improving it.

2.4 Measuring Energy and Energy Efficiency

To assess the maturity level of an organization, the metrics proposed in the previous section need to be computed. Precise indications in the maturity models and best practices on how to compute these metrics are still work in progress. To this purpose, another branch of Green IT focuses its attention on measurements. In order to improve EE or to understand the state of the system, we need to be aware of its energy consumption. This branch can be divided into two parts, as shown in Fig. 2.4. In Sect. 2.4.1 we deal with energy consumption estimation: in some systems, measuring the amount of energy used by IT components is a non trivial problem. In Sect. 2.4.2 we

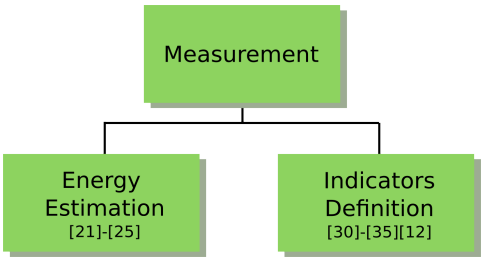


Figure 2.4: *Measuring energy and energy efficiency*

see that the state of an IT system can be measured looking at metrics other than energy itself. This section addresses the research question "*How can we measure EE of an IS?*".

2.4.1 Energy usage measurement and estimation in IT systems

The first step that has to be taken in order to improve EE in an IT system consists in measuring the energy consumption of the system itself and eventually of the individual components of the system. Measuring the real energy consumption of a system under a specific workload is usually very hard and, most of the time, the precision of the results is very poor. In fact, energy consumption of devices changes in a non linear way depending on the load. That means that complex models need to be learned to estimate the power consumption of a device given its load. Some benchmarks have been defined to compare the behaviors of different configurations [16] stressing the importance of both performance and power. As an example, the Standard Performance Evaluation Corporation (SPEC) [17] proposes a power and performance benchmark, SPECpower. It provides a means to measure power in conjunction with a performance metric, helping IT managers to consider power characteristics along with other selection criteria to increase efficiency of data centers. SPEC includes power measurements even in other benchmarks, such as SPECweb, used for evaluating web server performance, and including also a power workload and a performance/power metric. The same approach can be found in SPECvirt, addressing performance evaluation of data center servers considering consolidation in virtualized environments. Another benchmark dealing with energy has been provided by the Transaction Processing Performance Council (TPPC) with the name of TPCEnergy [18]. This benchmark enriches former ones with information about energy with the purpose of helping IT managers in choosing the best configuration taking this aspect into account.

A recent benchmark has been proposed by the GreenIT-BB network [19] for showing potential energy and cost saving. The benchmark is designed for energy and resource efficiency using environmentally sound performance parameters both at the hardware and software level. Data centers are classified in peer groups based on the number of servers and CPUs, the use area, and the operational purpose. In the end, thirty comparison values are given and a traffic light representation for each of them suggests dimensions that have to be improved to conform the average of the data centers in the same peer group.

Most of the benchmarks with attention on energy are still under development and some scholars prefer to use more consolidated benchmarks as a starting point to build a model for estimating power consumption. In [20], authors refers to TPC-C benchmark because they wanted to study the behavior of transactional systems. TPC-C is known to be the most credible of this kind of benchmarks and a big amount of data are available. Starting from the system configuration explicit in the TPC-C tests, the authors build a model for estimating power consumption taking into account the main components of the system: CPU, memory, disks, server chassis, and disk enclosures. While the energy consumption of CPU, memory and disks is estimated considering the peak power consumption reported in the manufacturer's specifications, the consumption of the server chassis has been estimated as the 30% of its component consumption plus a fixed overhead of 100 watts. The model was compared with real measured values and showed an error between 14% and 17% (see [20] for more details).

The principal utilization of benchmarks is in the comparison of several systems with the intent of choosing the best option. This decision is usually based on data describing the behavior of the system when running the benchmark. In other cases, the goal can be to improve an already existing system. In this scenario, metrics to evaluate a system need to be defined.

Measuring energy in case of hardware components means, for instance, to measure the energy consumption of a single processor or hard disk. If it is not possible to measure the energy consumption directly on the component, it can be estimated starting from the usage of the component itself and from the data sheet released by the manufacturer. Data about usage are provided by the operating system.

In addition, it is also useful to measure energy consumption of a software or middleware component. In some cases, it can be necessary to split the energy consumption of servers in the energy consumption of the Virtual Machines (VMs) deployed on them or, at a higher granularity, to estimate

the energy used by running applications. A considerable amount of work has been done in this direction, but the problem is still open. In [21], the authors try to estimate the energy consumption of a single application based on the resources used and the total amount of time, also considering the interaction among system resources. The resources considered are CPU, network, and disk. The estimation of the energy consumption is performed starting from the battery usage of a laptop. So, this software cannot be installed on a plugged in system. A similar approach is described in [22]. Here a distinction is made about the consumption of each component in activity and in idle state.

Other approaches focus on the VM level trying to produce models that approximate the energy consumption of each running VM. An example of this approach is described in [23], where the account of energy consumption is based on Performance Monitoring Counters (PMCs) that enable CPU and memory energy consumption estimation. PMCs are a hardware facility which collects data about the occurred events. The approach is based on the production of a power model collecting data for different loads of the components, which is then used to train the model. The model is created using linear regression to compute the weight in power consumption for each component. The per-VM accounting can be performed because of the abstraction provided by the operating system, able to gather per process PMCs. Another approach [24] builds a parametric model to compute energy based on resources usage. In the model only CPU, memory, and disk are considered. The contribution of each component is modeled as a linear function of its usage. The accounting of each VM can be made knowing the amount of each physical resource used by it. This information is available from the hypervisor. The parameters of the linear functions need to be learned using linear regression.

In [25], authors try to estimate energy consumption in a cloud environment in order to compare it with conventional computing. They consider three approaches using cloud: (i) Storage as a Service (STaaS), in which only storage is in the cloud; (ii) Business Process as a Service (BPaaS), in which cloud is used only for large computational tasks; (iii) Software as a Service (SaaS), in which the cloud hosts both software and storage devices. Different models for energy consumption are described to represent the transport network, the data center, and even terminals and equipments owned by customers. A distinction is made between private and public clouds, which have a different cost for accessing to the network due to the devices and distances involved. For user devices and data centers, en-

Table 2.2: *Comparison Between Energy Estimation Approaches for Applications and Virtual Machines*

ID	Level	Resources	Description
Do2009 [21]	Application	CPU, Network interface, Disk	Energy consumption is inferred monitoring the battery usage (Watt), considering the amount of resources used and time
Kansal2008 [22]	Application	CPU, Disk	Energy is estimated distinguishing between idle and active mode of components and assuming to know the power consumption in the two modes
Bertran2010 [23]	VM	CPU, Memory	PMC based power model with different weight for each component contribution. Parameters learned through linear regression
Kansal2010 [24]	VM	CPU, Memory, Disk	Energy estimation using a model based on a linear function of usage of resources. Parameters learned through linear regression
Baliga2011 [25]	Cloud	Network, CPU, Disk, Memory, GPU	Three kinds of clouds services are compared with conventional computing. Network, data center, and customer equipment are considered in energy computation

ergy consumption is estimated from hardware specifications and from the assumption that the average power use in a data center is 66% of the maximum power. The energy consumption of the network is estimated considering all the devices involved, knowing their power consumption and their capacities expressed in bits per second.

An overview of the approaches for estimating EE discussed in this section can be seen in Tab. 2.2, highlighting the components that are considered in each of the presented approach and the level at which energy is estimated (VM, Application, or Cloud). Estimation of energy is still an open issue and all the presented techniques use approximation in defining the energy consumption model.

Beyond IT resources energy consumption evaluation, other factors should also be considered. It is worth noticing that even software has a relevant im-

pact on energy consumption, as emphasized in [26], due to a different resource management. Here, an empirical measurement of energy consumption has been made to compare Management Information Systems (MIS) running similar applications from the efficiency point of view. Three kinds of benchmarks have been used: 2 Enterprise Resource Plannings (ERPs), 2 Customer Relationship Managements (CRSs), 4 Database Management Systems (DBMSs), and significant differences in energy consumption for similar applications emerged. Even if the relation is non-linear, the fastest application proved to be also the most energy efficient. The Operating System itself has an impact, and Linux proved to be 50% more efficient than Windows, on average. This demonstrates the importance of being aware of the energy consumption of an application and of being able to measuring it. In [27], authors claims that a bad software can waste efficiency obtaind with optimal hardware. They also identifies factors that influence software performance as computational efficiency (e.g. efficient algorithms and data structures, multithreading), data efficiency (e.g. data movement minimization, efficient use of cache memories), and context awareness (application that can adapt their behavior to changes in the environment).

Another aspect that is usually omitted when measuring energy and the impact of a system over the environment is related to the source of energy. Different sources of energy can not be considered as equal because some of them can be greener than others. This concept is well described in [28] where authors make a distinction between green and non green energy, arguing that the true impact in the production of energy has to be considered together with the amount of energy consumed. This difference should be taken into account when allocating tasks to servers in the cloud. A similar approach is also faced by [29] where authors highlight the importance of using green energy to reach sustainability in IT systems and propose a load balancing algorithm that takes into account if the source of energy is green or not.

2.4.2 Monitoring and assessment of Energy Efficiency

In complex systems, measuring or estimating energy is not sufficient to enact an effective strategy for sustainability. Other features of the system can be monitored and evaluated with the intent of understanding its EE. For this reason, a wide set of indicators has been defined in the state of the art.

The most common way to measure efficiency of a system is the use of the Power Usage Effectiveness (PUE) metric. It measures the total energy of the data center divided by the amount of energy consumed by the IT. The

Data Centre Metrics Coordination Taskforce was set up in 2010 to ensure clarity and consistency in data center metrics[3]. It published a white paper [30] giving details of how to compute PUE in an effective way. Other metrics are also defined to measure renewable energy technology and reuse of energy such as Green Energy Coefficient (GEC) measuring the percentage of green energy, Energy Reuse Factor (ERF) measuring the percentage of energy reused outside the data center, and Carbon Usage Effectiveness (CUE) measuring the amount of GHG emissions relative to IT energy consumption.

Even if PUE is very extensively used, it is most of the time a misleading metric in stating the EE of a data center. As discussed in [12], if the data center manager reduces energy consumption of the IT by managing resources in a better way (e.g. turning off a unused server), PUE will decrease though the performance of the data center is improved. For this reason, PUE can not be considered as a standalone metric for stating greenness of a data center.

Another category of metrics has been defined for measuring efficiency at the application and service level. A well known set of indicators, related to the performance of the system, is composed by the Key Performance Indicators (KPIs). This set of indicators includes the most relevant metrics that are able to express the ability to respond to the requests in terms of quality and success of the whole organization, of a process or of a single activity. In order to meet the green requests, other indicators have been recently defined to express the EE and not only the performance of a system or organization. These indicators can be referred in the state of the art with the names of Green Performance Indicators (GPIs) [31] or Key Ecological Indicators (KEIs) [32] with the purpose of measuring the environmental impact of a monitored system, process or activity. Authors of [31] cluster indicators in four categories:

- *Application Lifecycle KPIs*: include all the indicators defined for monitoring the performance of a system, an application or an activity. They are only indirectly related to energy;
- *Organisational GPIs*: the organizational factors that help to define guidelines for managing the organization according to laws and regulations related to energy;

[3]Members of the task-force are: Green Grid, the US Department of Energy's Advanced Manufacturing Office and Federal Energy Management Programs, the US EPA Energy Star Program, the European Data Centre Code of Conduct, Japan's Ministry of Economy, Trade and Industry (METI) and Japan's Green IT Promotion Council (GIPC)

Table 2.3: *Green Performance Indicators Clusters*

Cluster Name	Definition	Examples
Application Lifecycle KPIs	Indicators defined for monitoring the performance of a system, an application or an activity	Response Time, Throughput, Availability, Recoverability, Reliability, Process Engineering
Organizational GPIs	Indicators describing organizational factors that helps to define guidelines for managing the organization according to laws and regulations related to energy	Human Resource, Compliance, Infrastructural Cost, Carbon Credit
Energy Impact GPIs	Indicators describing the impact of a system or application on the environment considering energy related factors	Application Performance, DCIE, PUE, CO_2 Emission, IOPS/Watt, Capacity/Watt
IT resource usage GPIs	Indicators expressing the efficiency of IT resources in terms of their usage level	CPU, Memory, I/O and Storage Usage, CADE, DH-UR, DH-UE

- *Energy Impact GPIs*: this cluster describes the impact of a system or application on the environment considering energy related factors;
- *IT resource usage GPIs*: this cluster is strictly related to IT and it is composed by metrics expressing the efficiency of resources in terms of their usage level.

Tab. 2.3 summarizes these four clusters of indicators making some examples.

GPIs are usually associated with a threshold indicating the minimum and/or maximum desired value for that metric. Three levels of satisfaction are defined for GPIs: green if they are fully satisfied, yellow for partial satisfaction, and red for violation [33].

The concept of KEIs proposed in [32] refers to a set of indicators used to measure the environmental impact of a process or of part of it, considering features such as energy consumption, water consumption, CO_2 emission, recycling, or regulatory compliance. KEIs are defined as a tuple consisting of a metric and a target value representing the goal that has to be achieved. This goal is defined by the system manager.

In [34], a new perspective about which kind of metrics should be used is presented. The authors find out that consumption-based metrics reflect energy savings better than considering only resource efficiency once the performance level (expressed as the number of operations to be executed)

is fixed. In fact, while resource efficiency reflects the level of performance given the amount of energy used to do so, consumption based metrics show the energy needed to obtain a fixed performance. Using this kind of metrics, resources are easily comparable in relation to their EE. As an intuitive example, the authors use the comparison between the Miles per Gallon (MPG) metric (a resource efficiency metric) and the Gallon per 100 Miles (GPM) metric (a consumption-based metric) for comparing vehicles. They argue that the MPG metric can be misleading because it obscures the fact that small improvements in efficiency lead to important reductions in fuel consumption, highlighted by the GPM metric. The authors propose to apply this reasoning to decide whether it is recommendable to change a hardware device or not. A similar argument is defended in [35], in which "usage metrics" are introduced. Authors make a distinction between two perspectives for measuring indicators: the *manufacturer perspective* and the *usage perspective*. The former focuses on the potential EE of the single component of the system, while the latter considers the efficient usage of these components. Authors stress the usage perspective suggesting that how a resource is used in a data center configuration is as important as the potential efficiency of the resource itself. So, the optimization of the actual usage of the components of a data center is necessary for improving EE.

In [12], Greenpeace proposes a methodology for assessing the greenness of big companies using the Clean Energy Index. Companies are scored considering the following aspects:

- *Coal and nuclear intensity*: the total percentage of coal and nuclear generated electricity used to power the data center;
- *Energy transparency*: the level of detail made publicly available related to the energy consumption of the IT infrastructure;
- *Infrastructure siting*: the siting criteria and investments for allowing the usage of cleaner energy;
- *Energy efficiency and GHG mitigation*: the strength of the strategies to mitigate the demand for not renewable energy generated by the IT infrastructure;
- *Renewable energy investment*: the commitment to renewable energy investments ant to wide-scale renewable energy generation and use.

All the indicators described in this section can be used to evaluate the state of the system and to understand its efficiency level. The energy as-

sessment can lead to some repair or adaptation actions if the efficiency is not satisfying. Techniques to improve the EE are discussed in Sect. 2.5.

2.5 Methods and Techniques to Improve Energy Efficiency

Metrics introduced in Sect. 2.4 can be used to detect situations of inefficiency in the monitored system. Once an inefficiency is detected, a set of techniques can be used to deal with it.

This section addresses the last research question: "*What can be done to improve EE in an IS?*", classifying some techniques and dividing them in two categories (Fig. 2.5). Most of the work at the state of the art about EE focuses on physical resources management, however this is not the only way to deal with energy saving. Some studies face the problem with a focus at a higher level, taking into account the process itself and how it can be modified in order to obtain a greener solution. So, in Sect. 2.5.1 the problem is considered from the point of view of designing processes, while in Sect. 2.5.2 the techniques of resource management for improving EE are analyzed. Given the wide body of literature on some specific aspects, this section does not want to be comprehensive of all the possible techniques adopted to deal with EE improvement, but only to give an overview of which parts of the system can be improved and available strategies to do that.

2.5.1 Designing green processes

Physical resources management is not the only way to deal with energy saving. Some studies face the problem with a focus at a higher level, taking into account the process itself and how it can be modified in order to obtain a greener solution.

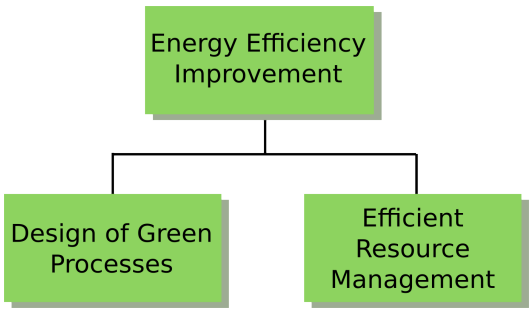


Figure 2.5: *Improving energy efficiency*

According to [36], the role of BPs to create green IS and environmental sustainable organizations has to be explored with attention. Without a design and implementation of BPs with attention to green features, it is not possible to reach sustainability. So, according to the authors, the greening of an organization has to include a process re-engineering. To this purpose, four relevant areas are identified: (1) Process Design, (2) Process Measuring, (3) Process Improvement and Change, and (4) Process Implementation, covering the life cycle of a BP. Business process design and implementation should take into consideration environmental constraints and it should be possible to measure sustainability in order to reach transparency and awareness. Through awareness, humans involved in the process are more cooperative in working for sustainability. Moreover, awareness allows the management to change some aspects of the process to improve its efficiency.

The four areas identified in [36] are explored with some variations in other research papers. In fact, several approaches can be used to reach the goal. In this section we focus on four designs of green processes, classifying the approaches in the following categories: process annotation, patterns definition, service reconfiguration, and goal-oriented representation (Fig. 2.6).

Business Process annotation

A research branch proposes the enrichment of the BP description with additional information that can be used to understand its behavior in time with respect to energy consumption.

In [32], the authors analyze the problem proposing a methodology to improve EE. It consists in re-engineering the BP considering its composing activities, enriched with information about their own EE. The re-engineering

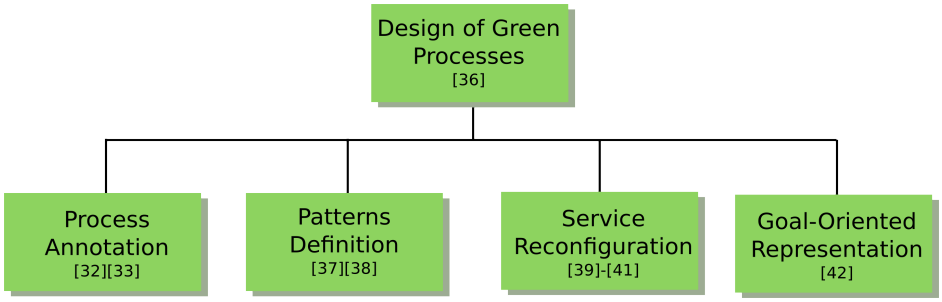


Figure 2.6: *Business Process design*

process is divided in four phases: (1) Strategy; (2) Sensing and Monitoring; (3) Analysis and Management and (4) Adaptation. In the first phase, a set of performance (KPI) and ecological (KEI) indicators are defined. These indicators are used to reflect the ecological objectives and the Quality of Service (QoS) requirements of the process. In the sensing and monitoring phase, data to compute the indicators are collected and linked to the activities that use the monitored resources. The result of this phase is a process annotated with indicators. During the Analysis and Management phase, indicators are analyzed to identify the activities with the higher negative impact on the environment. From the set of activities only the ones that are modifiable and that violate the thresholds of the indicators are considered, and a weight is associated to them. The weight depends on the amount of the violation. The last phase, the adaptation, consists in applying some actions to improve EE that can involve a restructuring of the process or the modification of the allocated resources. The impact of an adaptation strategy is evaluated computing the value that KPIs and KEIs would assume after its execution. This evaluation enables the choice of the best strategy. However, there is no automatic decision maker, but the choice is performed by an expert knowing in advance all the needed information about the outcome of an adaptation action (examples of possible adaptation actions are described below in the Pattern-based approach).

Another adaptive approach to EE improvement in BPs is described in [33]. Here, the BP is enriched with information related to past observations of the workflow, such as the probability of execution of a branch and the set of indicators (KPIs and GPIs) used to monitor the whole process or each single activity. A set of adaptation actions is also defined and linked to the violation of indicators by the definition of "rules". A rule is composed by an antecedent, which is the violation of an indicator, and a consequent which is the action that can be addressed to solve the violation. Some examples of actions are the reconfiguration of the resources allocated to the BP or to its specific activities or the reconfiguration of the BP itself by including or excluding optional activities in the workflow. Some parameters are associated to each rule to describe the reliability (*confidence*) and the impact (*importance*) of executing that action in the current context. These parameters are computed considering the past history of the system. An algorithm is proposed to decide the best strategy to be enacted taking into account a hierarchical organization of the indicators. In this case all the violations are considered in the same way, without considering the entity of the violation. Through the described algorithm, the system is able to

support the system administrator in the decision of appropriate actions to achieve goals based on KPIs and GPIs, suggesting all the available strategies that fit the actual situation and ordering them considering the confidence and importance parameters. Using this approach, the strategy can also be automatically selected, requiring a human intervention only when no improvement occurs.

Patterns to design Business Processes

One qualitative approach to improve EE of a BP is described in [37], where the authors propose some patterns to design green BPs. In this work the concept of green BP modeling as an extension of the classical BP modeling approach is introduced. While BP modeling considers four dimensions (cost, time, flexibility and quality), green BP modeling adds as a fifth dimension, the ecological one. Improving the classical four dimensions usually has an improvement over EE as a side effect. A set of patterns can be identified to lead the organization to reduce its environmental impact. Authors in [37] identify nine different patterns, which explore different ways to modify the BP toward a greener dimension. These are Green Compensation, Green Variant, Resource Change, Green Feature, Common Process Improvement, Process Automatization, Human Process Performance, Outsourcing and Insourcing, summarized in Tab. 2.4. A decision tree helps deciding which pattern/s can be activated proposing a set of questions about the process and the organization.

The approach has been further developed in [38], where the infrastructure level has been considered through the definition of a set of cloud patterns. The paper proposes a pattern-driven adaptation divided into three main phases: preparation (annotate patterns, identify ecological problems, identify green patterns), action (migration decision, migration), and follow-up analysis (monitoring and analysis). In the cloud context, BP components can be migrated inside the cloud infrastructure to reach a more efficient configuration. The application of the green BP patterns is supported by a set of cloud patterns, describing how to create and manage applications and services in a way supporting the ecological adaptation of the application. The cloud patterns considered in the approach can be grouped into three categories: *cloud environment* (private, public, hybrid), *cloud elasticity* (component removal, start and stop on demand), and resource sharing through *multi-tenancy*. Each of these patterns can be related to the BP patterns described in Tab. 2.4, as described in Tab. 2.5[4].

[4]Other cloud patterns are described at `http://cloudpatterns.org/` and `http://en.`

Table 2.4: *Patterns to reduce the environmental impact of a business process [37]*

Pattern Name	Pattern Description
Green Compensation	If the business process cannot be modified, a compensation activity can be started to compensate its environmental impact
Green Variant	An alternative green process is used achieving the same result in a greener way
Resource Change	Change the resources used by some of the activities with greener ones
Green Feature	Replace all the features for which a green alternative exists
Common Process Improvement	Some process or activities are improved using their KPIs
Process Automatization	Some of the activities within the BP are automatized
Human Process Performance	Replace activities performed by machines that pollute with human activities
Outsourcing	Outsource certain activities from the owner to another partner providing an equivalent but greener service
Insourcing	Integrate services from third parties into its own organization

Service reconfiguration

Some scholars faced the problem of sustainable BPs focusing their attention on BP reconfiguration. The general approach consists in decomposing the process in sub-tasks that can be substituted with similar but greener ones. This is the case of [39] where the authors propose a technique to re-design BPs basing the decision on qualitative and quantitative metrics, such as CO_2 emissions, air quality, and damages to fauna and flora. These measures are annotated to the BP and are propagated to obtain a single cumulative value for each possible path. The BP is considered as composed by fragments. Each fragment can be substituted using a library of similar fragments, in order to find the greener solution to the problem. In the exploration of the possible solutions, a requirement of *design proximity* is used. It means that the system tries to obtain the best solution maintaining the status quo as

`clouddesignpattern.org/index.php/Main_Page`

Table 2.5: *Cloud and BP patterns relations [38]*

BP Pattern	Cloud Patterns
Green Compensation	Cloud environment
Green Variant	Cloud environment, cloud elasticity, multi-tenancy
Resource Change	Cloud environment, cloud elasticity
Green Feature	Cloud environment, multi-tenancy
Common Process Improvement	Cloud environment
Process Automation	Cloud environment, cloud elasticity, multi-tenancy
Human Process Performance	none
Outsourcing	Cloud environment, cloud elasticity, multi-tenancy
Insourcing	Cloud environment, cloud elasticity, multi-tenancy

much as possible.

Another example is discussed in [40]. Here, the authors take into consideration both QoS and EE metrics proposing a technique to aggregate and combine them in a non linear way. The process is represented in an abstract way and it is decomposed in sub-blocks. For each of the abstract sub-blocks several implementations exist with different costs in terms of EE and quality. The problem consists in finding the best configuration using service composition techniques and optimizing a general function for the whole problem. In order to do that, authors propose to use the Constraint Satisfaction Optimization Problem (CSOP) technique, under the assumption that the probability of execution of branches is known. Solutions that do not respect the constraints for unlikely paths are considered with a penalty, so that a wider range of solutions can be found. However, this approach can present some computability issues, since the resolution algorithm is NP-Hard. For this reason, it is effective only for relatively stable compositions.

A similar approach is presented in [41]. The architecture in this case is divided into three levels. The higher one is the abstract BP composed by activities implemented through services that compose the middle level. Services are deployed over physical machines that constitute the lower level. Also in this case the problem of service composition is instantiated as a CSOP. The decision is based over a QoS value for each service, comprehensive of performance and green indicators. The main difference consists in the proposal of a model to compute energy consumption at the service

level. Energy is estimated considering the percentage of the resources used by each service weighted with parameters computed observing the components behavior under different workloads. When deciding whether to modify the actual implementation of the abstract process, the cost of this modification is considered, in order to change it only if the gain is real.

Goal-oriented representation

In [42], authors propose a process representation using Goal Requirement Language (GRL) to improve EE. In this model, EE goals are modeled as non functional requirements. That means that even if they are not mandatory, they are taken into account when selecting the best strategy. The model includes several elements, such as goals, soft-goals, tasks, resources, and beliefs. A goal is usually the objective that the organization wants to achieve and can be either a business and a system goal. Soft-goals are quality attributes representing the non-functional requirements. Tasks are different strategies that can be used to achieve a goal, and they can be decomposed in subtasks. Tasks also use resources that can be available or not. All the elements are related to each other with different kinds of links, expressing the kind and the strength of the relation. Authors suggest an analysis strategy divided in four steps to help the decision process of a strategy to satisfy the goals. A five steps process is proposed: (i) *modeling IT system design objectives*, where the goals are identified; (ii) *generating IT system design alternatives* to satisfy the detected goals, they are represented as tasks connected to the related goal; (iii) *building environmental analysis models for the IT system design*, where the most important aspects of the IT system's environmental impact are identified together with their relation to the operational task that influence them; (iv) *evaluating the environmental impact of the design alternatives*, based on the positive and negative impact that each possible path has over detected environmental goals; (v) *improving the design alternatives and making design decisions* considering the results obtained at the previous step to compare the different solutions. This approach gives also the possibility to deal with performance and energy issues at the same time, giving more importance to performance.

The four categories discussed in this section show the several possibilities available in the literature to improve EE acting over the process. A summary is shown in Tab. 2.6, where the main features of each of them are highlighted. The enrichment of the process with annotations imposes constraints over energy consumption and highlights the features of the process

Table 2.6: *Approaches to Green Process Design*

Approach ID	Category	Short Description
Nowak2011 [32]	Process Annotation	Activities annotated with information about their greenness and weighted accordingly
Cappiello2011 [33]	Process Annotation	BP and activities annotated with information about their greenness and probability of execution
Leymann2011 [37]	Patterns Definition	Definition of a set of general patterns to increase real or perceived greenness of a BP
Leymann2012 [38]	Patterns Definition	Definition of a set of general patterns to increase real or perceived greenness of a BP in a cloud environment
Ghose2009 [39]	Service Reconfiguration	Decomposition of the BP in interchangeable fragments associated with an energy efficiency parameter
Ferreira2009 [40]	Service Reconfiguration	Decomposition of the BP in blocks and composition implemented as an optimization problem
Oliveira2010 [41]	Service Reconfiguration	Decomposition of the BP in blocks and composition implemented as an optimization problem. Energy estimated at the service level
Zhang2010 [42]	Goal-Oriented Representation	Use of the GRL to represent the process and reason about the alternative solution and their energetic impact

itself that can be used to take green decisions about it. Examples of the available decisions can be represented through patterns that describe general solutions available. Service reconfiguration enables the modification of the process choosing the best solution in a pool of services, while goal-oriented representation allows the process designer to include greenness as one of the goals of the organization. All these approaches are not exclusive, and can be combined as a composed green solution.

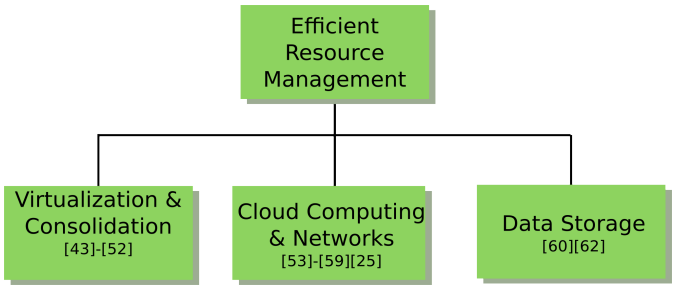


Figure 2.7: *Efficient resource management*

2.5.2 Efficient resource management

The most common and intuitive approach to improve EE is related to IT resource management. Many techniques, born to improve the performance of IT systems, have a great impact also on energy. This set of techniques are widely applicable and flexible. In fact, resources are easy to be monitored and managed and techniques about resource management can be applied to each kind of system at every moment, without the need of an a-priori design. In this section resource management techniques are discussed considering three categories: virtualization and consolidation, cloud computing and networks, and storage EE improvement (Fig. 2.7).

Virtualization and consolidation

Every system uses a set of resources that need to be allocated to it. If the allocation is oversized, unused resources are wasted, and the result is a waste of energy too. In the last years, a lot of studies have demonstrated the important contribution of efficient resource allocation in energy saving, enabled by common techniques such as virtualization and cloud computing.

Virtualization consists in allocating virtual resources instead of physical ones to a machine, allowing the system to run more than one machine on a single server and to distribute physical resources in a more efficient way. The use of VMs allows an efficient and dynamic management of the available resources that would be impossible using only physical machines. In this way, resources that are underutilized can be reallocated. At the same time, an application requesting more resources than expected can be empowered to satisfy the SLA stipulated with the user. A lot of efforts have been made to find a trade off between efficient resource allocation and performance, creating a field known as server consolidation. The aim of server consolidation consists of allocating VMs to physical devices in the

most efficient way. Trying to concentrate the allocated resources over less servers, servers that are not used can be turned off or put in stand by with a consequent energy saving. Simple solutions can be used in case of static workload [43] considering only performance. Here the reconfiguration is lead by a cost and utility function, considering only CPU and memory as devices. The algorithm is able to refine its parameters during the execution, but it is not designed to deal with dynamic loads. The gain in performance is around 25%. A similar context is described in [44]. It considers the problem of VM allocation in case of static workload, where the only factor to consider is the initial allocation of a new VM. Several algorithms are compared and three different objectives are considered: job performance maximization under power consumption constraints, power consumption minimization under job performance constraints, and multi-objective optimization of both power consumption and job performance. An heuristic is needed since all the algorithms are NP-Hard to solve. The multi-objective approach seems to perform better, even if it requires a longer elaboration time, that can be a problem in case of big data centers. The gain in energy terms for the best algorithm is around 10%.

As stated before, virtualization techniques were initially used to improve performance and not energy. Only few authors focus their attention also on energy aspects. An example is [45], where a comprehensive approach is presented to efficiently allocate resources maximizing SLAs while reducing costs. A set of possible interventions to improve EE are coupled with virtualization. These interventions are divided into two categories that we adopt in the analysis in this section: *short term* and *long term*. As "short term" the authors classify all the actions that can be actuated in a small time range such as load balancing, capacity allocation, and frequency scaling. As "long term" they identify more complex actions such as server switching to power off and resource allocation to applications in servers and reconfiguration. Short term interventions can be enacted with a higher frequency than long term ones. An autonomic computing approach is presented to allocate resources. The service center is modeled as a queue of requests that can be satisfied by one or more applications. The problem consists in deciding the best solution about which application to use and which resource to allocate. A cost model is presented consisting of utility functions in which penalties are related to the performance level and to the energy cost associated with the use of servers. A first solution is approximated using a heuristic approach.

Simpler systems concentrate their efforts in allocating VMs to available

processing units trying to keep the frequency of the processor as low as possible [46]. This is because a big amount of energy consumed by a server is due to the processor and the lower is the frequency, the lower is the energy consumption. Optimization techniques are used to distribute the load to the different VMs in an efficient way. Before being allocated, VMs are modeled in terms of required processor frequency and execution time. The allocation algorithm processes VM allocation one by one by allocating first VMs with a higher processing frequency requirement.

Other approaches tend to optimize resource allocation taking advantage of some features of the virtualization technology, such as min, max and share parameters, to set the amount of each resource requested [47]. Thorough these parameters it is possible to specify the minimum amount of a resource needed to run the VM, the maximum amount that can be allocated, and how much it is possible to share the allocated resources with other VMs. Guided by application utilities, high utility applications get most of the resources. The improvement in the overall utility of the data center is around 47%.

In [48], authors propose a two level control architecture to guarantee response time while minimizing energy. At the first level, load balancing is used to guarantee that each VM respects response time constraints. The second level controls CPU frequency to increase EE. In [49] a control loop is designed to dynamically reconfigure VMs, monitoring and evaluating the current state, and planning and executing modifications. System load and response time are used to assess the level of performance, and reconfiguration operations are defined, such as switching off servers and frequency scaling.

Artificial intelligence techniques are often used to deal with uncertain information and to predict the future behavior of the system. The approach proposed by [50] deals with resource allocation and switching off servers. The approach, focused on CPU intensive workloads, uses machine learning to predict the SLA level in a new configuration helping in deciding the goodness of the solution. The system tries to consolidate tasks into as few machines as possible, without affecting performance. Models to predict power consumption and client satisfaction of the future configuration are learned from collected data. The algorithm reduce power consumption by 10%. In [51] a simpler system is presented. Here, a reinforcement learning approach is used to apply efficient system reconfiguration to deallocate unused resources and to increase insufficient resources. The resources considered are only memory and CPU. A Markov decision process is used to

choose the best policy. The policy consists in the best reconfiguration action given a state and can be learned by enacting random actions at the initial steps and collecting information about performance of each VM in the new configuration. This approach is dynamic, in the sense that a new policy is automatically learned when the external environment changes. Two different algorithms are compared in [52] to control the CPU frequency in order to save energy. This has been done working on P-states, which are a predefined set of working states of a CPU. The number of P-states depends on the specific processor, switching from a P-state to another allows clock rate reduction. The authors propose a human immune system inspired approach and a fuzzy logic approach. The former is divided into two steps. In the initialization stage, data are collected and associations between workloads and P-states of the CPU are acquired, forming "detectors". In this phase, also "effectors" are acquired, associating actions to a given state. The second step is the self-optimization stage, following the Monitoring - Analyzing - Planning - Executing (MAPE) paradigm. Here, collected data are compared with optimal data about maximum and minimum workloads for each P-state, and a strategy is actuated if thresholds are not satisfied, which consists in changing the P-state. The second approach uses a fuzzy technique, defining three sets for each P-state: HIGH, LOW and MEDIUM. A fuzzy controller, depending from this value, is evaluated regularly and a P-state modification occurs when its value reaches 0 or 1. From tests, the immune inspired techniques seems to be more successful because it is more able to capture dynamic modification in the workload.

An overview of different approaches to an efficient allocation of the resources can be seen in Tab. 2.7, highlighting the main features of each approach and the long term and short term strategies adopted. As it can be seen in the table, the approaches differ for the combination of long and short term techniques adopted and for the decision algorithm used to decide when and how to change the configuration.

Cloud computing and networks

Similar approaches can be used to deal with cloud computing. Cloud computing consists in the dynamic allocation of shared resources to a VM. Differently from the previous approach, in this case the user of the system is not aware of the physical location of the resources allocated to its service. As before, the resource allocation problem consists in the search of the best physical machine where to place the VM. The choice is not limited to a single data center as in the previous class of approaches, but it

Table 2.7: *Resource Allocation Improvement Strategies*

ID	General Approach	Short Term Strategies	Long Term Strategies
Soror [43]	Static load balancing using utility and cost functions	none	resource allocation, only CPU and memory
Pierson [44]	multi-objective optimization of energy and performance	none	resource allocation, only CPU and memory
Ardagna [45]	Autonomic computing approach using an optimization model, a heuristic is used to find the first solution	load balancing, capacity allocation, frequency scaling	server switch off, resource allocation
Laszevski [46]	Ordered allocation of processing unit keeping frequency as low as possible	frequency scheduling	resource allocation
Cardosa [47]	Resource allocation using hypervisor features (min, max and share parameters)	none	resource allocation
Wang2008 [48]	Response time guarantee through a two layer architecture	load balancing, frequency scaling, CPU allocation	none
Petrucci [49]	Loop procedure for reconfiguring VM dynamically	frequency scaling	server switch off, resource allocation
Berral2010 [50]	Machine Learning approach to consolidate VM keeping SLA satisfaction, energy and SLA prediction in future configurations	none	server switch off, resource allocation
Rao [51]	Dynamic resource allocation using reinforcement learning with reward based on performance	none	resource allocation
Salomie [52]	Immune-inspired and fuzzy-based techniques	frequency scaling	none

is usually extended to a set of servers positioned in different data center, geographically distributed. As a consequence, issues related to network transportation costs have to be considered by the allocation algorithm. An example can be found in [53]. Here the problem of resource allocation is considered as a bin packing problem and an energy-aware heuristic al-

gorithm is proposed to find a solution. The algorithm also deals with the dynamic resource demand of applications by saving around 10% of energy. An other approach is proposed in [54], where authors propose a technique for improving EE in a cloud environment using three basic actions: VM reconfiguration, VM migration, and physical server powering off/on. In the model proposed by the authors, also the cost of each of these actions is considered, in the attempt to keep the number of SLA violations as low as possible. The reconfiguration action has the aim to optimize the resource usage level, avoiding under- and over-utilization. Several algorithms are proposed for migration, giving penalties for overloaded hosts and for number of migrations, while giving a reward for every empty host. Maximum energy saving using only reconfiguration is estimated to be around 61.6%; adding migration it can increase of a 37%. Similarly, in [55] the resource allocation takes into consideration several factors, related both to energy and QoS. In the specific, three factors are considered: total energy consumption, number of SLA violations (expressed as Million Instructions Per Second (MIPS) not allocated to a VM when needed), and number of migrations. Resource allocation is divided into two phases: new VMs placement and old VMs allocation optimization. The first problem is solved using a bin packing algorithm, while trying to keep CPU utilization of the server between a maximum and a minimum threshold. The second part consists in deciding when a VM should be migrated. The decision is again based on the thresholds over CPU utilization: when the server has low utilization, its VMs are migrated to put it in sleep mode; when it has high utilization some VMs has to be moved to ensure SLA satisfaction. The possible energy saving declared by the authors is between 53% and 77% if compared to a non energy-aware algorithm, with a performance degradation of 5.4%.

In [56], authors propose a framework for energy-efficiency awareness and adaptation. Starting from real power measurements at the server level, they propose models for assessing and predicting efficiency at several levels: server, VM, infrastructure, and service. Optimization is achieved through an algorithm considering actions as consolidation and migration of VMs, and deployment and undeployment of services. The objective function is the maximization of the total resulting efficiency of all nodes. Authors declare an improvement in EE of 14%.

As stated before, network transportation is a relevant component to be considered when dealing with clouds. Some recent studies focused their attention on that, stating that network transportation of data needs energy and under some circumstances using a cloud system can be more expensive

than using a conventional one [25]. In fact, energy consumption in transport and switching can be significant compared to the total energy consumption in cloud computing. In particular, cloud computing could not be the best solution in services in which there is an elevated access to storage devices that are placed in the cloud. That means that, when deciding whether to use cloud computing, considerations related to the nature of the workload have to be made.

Network efficiency is a very important variable in distributed systems. As discussed in [57], in order to obtain a green ICT system, a lot of data about monitoring and configuration have to be transmitted over the network. But this transmission has a cost that shouldn't overcome the benefits. Some scholars have been studying techniques to improve EE in networks working on routing control while keeping QoS level high. As an example, in [58], the routing optimization is modeled as an optimization problem, using as goal a function depending on network power consumption and delay of packets transmission at each node. The algorithm, solved using gradient descendant, can help in load balancing, distributing the traffic over the network and taking into consideration both QoS and energy.

An overview about reaching EE in a cloud environment is given in [59], where authors analyze the problem of software development life cycle in a cloud environment with a green perspective. Authors also propose a green cloud computing framework in which they identify opportunities for EE in efficient hardware and software selection, network optimization, VM scheduling and management, clean energy source selection, and efficient data center design.

Improvement of Energy Efficiency for data storage

A branch about resource management is related to storage EE. Storage devices impact on energy is really important and a good management of these devices can reduce energy waste significantly. Two main approaches can be detected in this field. The first approach reduces energy waste by working with disk speed. Disk can work with different speed modes, wasting more or less energy. An example of this approach is discussed in [60]. Here the authors provide an algorithm able to determine the optimal disk speed setting to optimize EE. In the optimization function, also the cost of changing the disk mode is considered, and a modification can be made only after a minimum time period if there is no need for performance issues, avoiding changing mode too frequently. The energy performance improvement of the approach is estimated around 29%. The other approach deals with

data migration in order to consolidate data in a smaller amount of devices, enabling turning off of unused devices or putting them in stand by mode. An example is discussed in [61] where authors use a conservation technique called Popular Data Concentration (PDC) that migrates frequently accessed data to a subset of the disks. In this way, the rest of the devices can be turned to lower-power modes. Data are organized hierarchically, based on their access rate. It is worth noticing that data in the cache will be placed in the lower level of the hierarchy. The approach can conserve up to 30-40% of the disk energy. There are also approaches that combine the two techniques already described, as in [62]. Here, the authors control disk dynamically, taking into account the actual usage of data and disk characteristics. The action performed are orchestrated by two controllers: the former controls disk modes and the latter manages data migration. The behavior of the controllers is lead by a set of policies expressed as fuzzy rules and clustered in three categories: policies for new data allocation (which is the best disk to store new data depending from the expected access rate), policies for disk modes (deciding when it is the case to switch a disk in quiet mode and to switch it back in normal mode) and policies for data migration (positioning of data in disks depending on their access frequency).

2.6 Concluding Remarks and Expected Evolution

In this chapter we have analyzed the state of the art related to the recent Green IT topic, considering it mostly from an Information Systems perspective. From this analysis three main areas emerged, linked to different research questions. The first area, answering the question "*How it is possible to state when an IS is green?*", is related to the definition of models and suggestion of practices to assess and improve the sustainability level of an organization. The second area, addressing the question "*How can we measure EE of an IS?*", faces the problem of measuring energy and of defining useful metrics for monitoring and controlling the greenness of the system. In the last area, answering the question "*What can be done to improve EE in an IS?*", techniques for improving EE are analyzed. These techniques can be distinguished into two main categories, that are high level techniques which affect the BP of the organization, and low level techniques which deal with efficient resource management. A summary of the techniques discussed and of the trends and challenges related to them can be seen in Tab. 2.8.

The three detected areas and their relative sub-categories cannot be con-

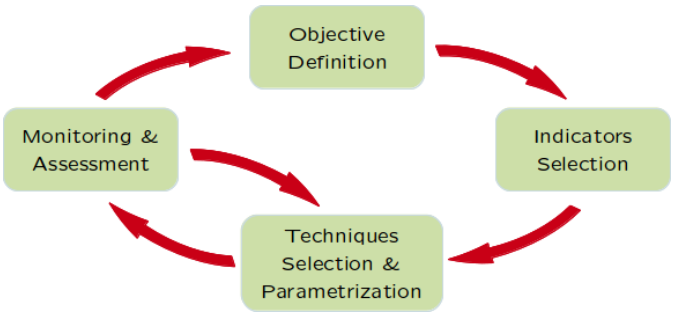


Figure 2.8: *Design process of a Green IT system*

sidered as independent, but some relations can be found. For instance, the assessment area needs metrics and requests the adoption of efficient techniques. Conversely, the monitoring of the system state requests the definition of thresholds for the measured metrics in order to detect non-green situation in the system. These thresholds could be defined explicitly or implicitly in the assessment models. Moreover, adaptation techniques require metrics selection for verifying their efficacy or for driving decisions. However, these relations are not always exploited and the three areas seem to be completely independent in most of the research work analyzed above. This can be seen as the main drawback of the approaches we discussed here. An approach taking into considerations all the identified areas is still missing.

We can imagine that future trends in research about Green IT for Information Systems will try to combine the three areas in order to obtain a comprehensive approach able to increase EE. In our opinion, the design of a Green IT system should be structured as in Fig. 2.8.

The first activity for the design of a Green IT system should be the assessment of the actual system taking as an example one or more maturity models described in Sect. 2.3. The assessment is obtained by comparing the desired state with the real one, that can be observed from the data collected through a monitoring system. Once the maturity level has been established, a set of objectives has to be defined. The objectives should depend either on the desired maturity level to reach or on a set of best practices that the organization wants to enact. The next step consists in selecting the indicators that can help the organization to assess the accomplishment of the objectives selected in the previous phase. A monitoring system is needed to support the collection of data from which the indicators values can be computed. Finally, techniques to improve EE can be selected, both related to BPs and to resources allocation. These techniques should be parametrized

according to the objectives. This can be done setting thresholds for the indicators or deciding the lower and upper bounds for resources usage. After an observation time depending on the type of techniques selected, two directions can be taken. If the objectives are still not reached, additional techniques can be applied or parameters can be tuned. Otherwise, if the goal is reached, the organization can decide to move to the next maturity level, setting new objectives and improving EE further on. The evolution and maintenance of the system are intrinsically supported by this model, since the model is independent from the specific process, and any changes in the objective definition automatically leads to the reconsideration of the other states, providing a dynamic and context-aware approach.

In any case, the current interest toward greenness and EE makes us believe that this field will continue its development and that soon all organizations will consider this aspect in their business plans.

The content of this chapter has been accepted for publication from the International Journal of Cooperative Information Systems [63].

Table 2.8: *Comparison Between Assessment Models for Green IT*

	Why Important	Sub Topics	What Directions	Challenges
Efficiency Assessment	It helps to understand the maturity level of the system and to identify future steps for improvement	Maturity Models, Best Practices	Regulations and constraints can rise the attention about EE and greenness, pushing towards a faster improvement in the field	Definition of a general set of goals for an heterogeneous set of organizations, definition of significant goals that can orient development towards efficiency
Efficiency Measurement	It allows to assess the level of maturity and to measure the effects of the enacted strategies	Energy Estimation, Indicators Definition	Definition of models for measuring energy at hardware and software level, definition of new indicators for measuring EE	Measuring energy consumption with a high level of precision and at different levels of the organization, definition of a significant set of metrics to be used in evaluating the system
Efficiency Improvement	It allows the reduction of the amount of energy consumed, it allows an adaptive behavior as a reaction to non-optimal situations	Design of Green Processes, Efficiency Resource Management	Using efficiency improvement techniques can significantly improve EE only with a coordination of techniques at different levels, considering the whole life-cycle of the system	The definition of a comprehensive approach for EE is still an open issue; efficiency is sometimes in contrast with QoS and it requires a redefinition of SLAs

CHAPTER 3

Self-Adaptive Techniques

Indice

The administration of modern Information Systems (ISs) can be not trivial due to the large number of elements which compose them and to the complexity of the services provided to users. This complexity requires a

high level human experience and a great effort in terms of money and time in a lot of phases of the system life-cycle. In order to overcome this issue, several techniques can be used to reduce the human intervention in the system using adaptation and self-adaptation. Self-adaptation enables the automatic and independent acquisition of new knowledge and the ability of reacting to unforeseen situations and to dynamically adapt in a changing environment. In this chapter we analyze a selection of adaptation techniques that can be used in IS management in order to automatically learn relations among variables of the system and to discover the most appropriate adaptation strategies to solve suboptimal situations.

3.1 Introduction

Nowadays, Information Systems (ISs) are very complicate systems composed by a large number of components and providing complex services to users. The administration of such an environment is not trivial and requires the intervention of high level human experience in a lot of phases of the system life-cycle.

Several techniques allow the reduction of the human intervention in the system using adaptive systems. Adaptation is widely used in several fields and enables a lighter management from the administrator point of view. The use of adaptation can be addressed to different goals. As an example it can be used to retrieve information about the current structure of the IS and of its components. This structure can change over time, especially in a virtualized environment. Adaptive techniques enable the automatic update of the information related to the system structure by analyzing occurred modifications and updating the information. Another useful application of adaptation is related to the enactment of repair strategies in case of failures. Using adaptive techniques, repair actions can be delegated to the system itself by making the repair process faster than in case of human intervention.

Different levels of autonomy can be addressed. Adaptation can be driven by mere if-then rules set by the system administrator or can be automatically learned from past experiences and more sophisticated learning processes. Of course, while the first case is simpler to implement, it lacks of flexibility since it is not able to respond to unforeseen situations. On the contrary, a self-adaptive system is flexible and able to respond to new situations. Self-adaptive systems learn adaptation strategies from experience. Obtaining good results with this kind of techniques can be complex and requires a big amount of data from which the system can learn, nevertheless

it has interesting upsides. First of all, it overcomes the need for domain experts to set predefined knowledge that can result to be incomplete. Moreover, unexpected knowledge can emerge that experts could not be able to provide.

In this chapter we analyze some techniques and approaches available in the state of the art in order to obtain an adaptive IS. First of all, in Sect. 3.2, we analyze adaptation in Service Oriented Architectures (SOAs) by proposing a set of techniques for responding to Quality of Service (QoS) violations and service failures. Then, in Sect. 3.3 we introduce the well known technique of Bayesian Networks (BNs) for representing relations among variables of a system and facing the problem of automatically learning these relations. Finally, in Sect. 3.4, we introduce the Adaptive Operator Selection (AOS) paradigm to automatically learn the best adaptation strategy from scratch. In Sect. 3.5 we summarize the content of this chapter while showing how all the techniques described here are used in the rest of the thesis.

3.2 Adaptive Information Systems and Services

In this work we focus on Energy Efficiency (EE) from the application perspective by adopting the SOA approach, where web services are supported and provided in terms of Business Processes (BPs). In web services, several systems cooperate to satisfy customers' requests in an integrated way. Resources used to this extent can be distributed over several systems. In this way, the service implementation is independent from the platform where it is running and can be seen as a black box providing an answer to the user through an interface, without specifying anything about how the answer is computed. In a SOA, interactions between several services and the combination in which they are used inside a process can be easily modified, by providing a flexible architecture. This allows a flexible reuse and modification of the single blocks.

In the years, SOAs have evolved moving towards more complex and distributed assets that rise new challenges in terms of service management and QoS.

In a service oriented context, the service offered to the customer is usually composed by several independent tasks that can be provided by different sources. When a contract is signed with a customer for the service provisioning, the customer is most of the time unaware of how the service is provided and which specific sources are providing it. However, a con-

tract is stipulated between the two parties in order to fix some constraints in terms of functional and non functional requirements. This contract is known as Service Level Agreement (SLA).

One of the issues that have to be faced by the system administrator is the dynamism of the environment where the availability and quality of the offered services can change due to several external and unpredictable factors. In this case, the system needs to be modified to respond to these modifications in order to respect the contract agreed with the user through the SLA. In this section we analyze some of the techniques available in the state of the art for providing an adaptive SOA, able to automatically react to service failures and to QoS violations. The selection does not aim to be exhaustive but it is addressed to provide some examples of available approaches for respecting QoS constraints and/or for improving performance.

The issue of adaptive services has been one of the main goals of the S-Cube Network[1]. The S-Cube Network has been created with the aim of "promoting a European network of excellence in software services and systems". The main goal was to promote the European scientific community in a leading role for SOAs. Adaptivity and measurement in SOAs was one of the main topics of the S-Cube community, with the aim of composing, evolving and adapting services coming from different sources. In more detail, the network developed adaptation concepts and techniques for SOAs including proactive adaptation and predictive monitoring techniques. One of the results of the S-Cube framework has been the work published in [64], where several approaches to describe and improve QoS are surveyed and classified.

3.2.1 Adaptive service composition

As stated before, a single process can be seen as a composition of a set of services interacting with each other in a sequence of executions. The performance provided by each one of these blocks can change over time. In fact, at a given point in time a service can perform poorly while other services can improve their performances or new services can become available. In this context, the problem is to select the better set of services to provide a process to the user fitting quality constraints.

In [65], authors propose a QoS based Web Service framework, called WebQ. The framework enables the selection of an appropriate service for each of the tasks in the underlying workflow, and dynamically refines existing services by keeping high the performance level. Five phases are defined

[1] http://www.s-cube-network.eu/

in the framework: (i) Workflow modeling, the general process is defined as a composition of services; (ii) QoS setting, constraints are added to the system in terms of quality; (iii) Initialization, a set of services are associated to each task in the workflow based on a ranking function; (iv) Execution and Monitoring, the load is divided across N selected services for each task and QoS is monitored; (v) Dynamic service selection, whenever a service decreases its quality the set of services for the correspondent task is reselected. The selection of the action set is performed considering a fitness function. The incoming load can be equally distributed or different weights can be used for each of the selected services. Using N services makes the process fault tolerant and increases performance. However, keeping all these services running together has a cost for the service provider that could not be justified by the performance improvement.

The problem of the optimization of web service selection is also faced in [66]. According to the authors, the problem can be faced using two different approaches. With a local approach a single service is selected at each time for executing a task. This approach enables only local optimization. Using a global approach, the complete set of services for the whole process is selected before the execution. The process is modeled at an abstract level, knowing branches probability in case of alternative paths and estimating the maximum number of loop iterations. During process execution, services are selected at runtime optimizing the execution plan by guaranteeing constraints and maximizing QoS. In the approach, performance parameters considered are execution time, availability, price, reputation, and data quality. Abstract services are matched with concrete services using Mixed Integer Linear Programming (MILP). For each service, a ranked list of suitable substitutes is kept. Negotiations occur when the optimization problem is infeasible and a solution can not be found. In this case, the system identifies the maximum number of feasible constraints and negotiates the others. Re-optimization is performed to adapt the process to a new context or to fix failures and suboptimal conditions.

In [67], authors propose a framework for flexible and adaptive execution of managed service-based processes, called PAWS (Processes with Adaptive Web Services). The framework supports the process management during both the design and the execution phases, by automatically selecting the best services for executing the process and the most appropriate QoS level. Also recovering actions in case of failures are considered. When a service has to be selected for executing a task of the BP, both functional and non functional requirements are considered. At design time, the designer sets

global and local constraints for each task, obtaining an enriched version of the process. Services are described through an abstract signature and a similarity metric is defined to compare the signature with concrete services. At run time a concrete service is selected for each task among the candidates. The selected service has to satisfy the constraints. In case of failure, a self-healing approach allows the execution of recovery actions. Possible actions are: retry, redo, substitute, and compensate.

3.2.2 Ontology based service matching

In [68], authors face the problem of providing an adaptive SOA in a multi-channel accessible context. The proposed framework, called MAIS (Multichannel Adaptive Information System), is able to automatically bind concrete services with abstract services requested by the user. This binding is performed taking into account the context of the request. This context represents explicit (provided by the user) and implicit (derived from the interaction with the user) information related to the user profile. In order to express the user profile and preferences, authors use an ontology-based model, by providing a generic domain that can be extended and enriched in a flexible way. The concepts in the ontology refer to a generic environment without any reference to the specific application. Each requirement is expressed by a pair attribute-value.

The framework is composed of three main components: (i) the repository, containing the context top ontology, (ii) the global context manager, which reasons on the context; (iii) the agent context manager, providing functionalities to manage the context and to communicate with the IS.

In the MAIS architecture, services are provided in a context-aware manner through several channels. The user selects a process, expressed in terms of abstract services where only the functionality and the interface are provided. The MAIS framework selects the best concrete service, matching the description with the context information.

3.2.3 Self-healing services

In [69], a self-healing approach for SOAs repair is proposed. Authors provide a tool for classifying the failure and automatically learn the best repair strategy according to the classification. According to the authors, failures can be classified into three categories: (i) transient, the active period of the failure is finite in time; (ii) permanent, the active period of the failure is not finite in time; (iii) intermittent, the active period of the failure is composed

of finite active periods followed by finite inactive periods.

Classification of failures is performed using a Bayesian classifier, which is able to select the correct typology starting from information obtained from a training set of failures. In the approach only two repair actions are considered: retry and substitute. The former action tries to execute the service again to obtain the desired result, the latter substitute the failing service with another compatible one. Since in most of the cases a single invocation of the action "retry" may be not enough, a retry period is identified which is dependent from the fault and which is a parameter of the action invocation.

Once the failure is classified, the system compares the current situation with similar situations occurred in the past. Similarity is evaluated by comparing faults contexts which depend from domain dependent and domain independent features. If not similar faults are retrieved, a general schema is applied depending on the fault type. After the action execution, the new information is added to the knowledge base for future similarity comparison in case of new faults.

3.2.4 Quality of service constraints relaxation

In [70], authors propose to face the problem of QoS satisfaction in a dynamic environment by relaxing constraints over quality parameters. The approach proposes a fuzzy representation of QoS parameters by allowing small violations. Parameters are transformed in fuzzy values by using a membership function. This function models the level of satisfaction of the parameters in terms of distance from the agreed quality in the SLA. The result of the application of the membership function is a value included in the 0-1 range. The use of this technique allows for partial satisfaction of the parameters, transforming the binary value (violated/satisfied) into a continue value. The satisfaction level can be classified as low, average and high and parameters satisfactions are combined using the minimum value for expressing an AND condition (more parameters have to be satisfied together) and the maximum value for expressing an OR condition (at least one of the parameters has to be satisfied).

Once the satisfaction level is evaluated, three solutions are possible. If there are no violations, no modification is required. Otherwise, if the violation of QoS parameters is limited, it is compatible with a possible renegotiation of the SLA with the customer. Finally, service replacement occurs when parameters deviation from the agreed values is too high. The decision of when a violation is compatible for renegotiation depends on a threshold set by the system administrator.

According to this, the process can be summarized as follows: (i) apply membership function to QoS parameters; (ii) obtain the membership result for each parameter; (iii) combine outputs into a single result according to AND and OR relationships; (iv) use a compatibility threshold to select the proper repair action.

3.2.5 Influential factors analysis

The framework proposed in [71] aims at avoiding Key Performance Indicators (KPIs) violation considering influential factors between metrics. The approach is based on the construction of a dependency tree learned using machine learning techniques.

The approach consists into four phases: (i) Quality modeling for analysis and adaptation: models are created for metrics (KPIs and QoS) and adaptation actions, by modeling the effect of each action on the modeled metrics; (ii) Analysis of influential quality factors: a dependency tree is generated for describing impact of metrics violation on other metrics; (iii) Selection of an adaptation strategy: several possible strategies are ranked considering their positive and negative effects on the metrics; (iv) Process adaptation: the set of selected adaptation actions is executed.

Adaptation actions are described by their positive and negative effects on metrics. To select the best strategy, the first step consists in selecting the set of KPIs for which violation has to be prevented. Once metrics are identified, influential factors are analyzed using the dependency tree, by retrieving the desired state for each of the metrics in order to avoid violations. The successive step consists in selecting the set of actions that improve each violated metric without negatively affecting other metrics that belong to the set for which violation has to be avoided. The resulted set of actions is combined using a Cartesian combination and ranked considering negative effects over other metrics (the less the better).

3.3 Bayesian Networks: Exploiting Connections Between Variables

Many domains can be described using a set of variables. Most of the time these variables are not independent, but they can be related to each other. One way to represent relations between variables is using Bayesian Networks (BNs).

A Bayesian Network is a probabilistic graph representing relations between variables in a domain. This kind of structure can be addressed with

different names, such as belief network, causal network, probabilistic network, or knowledge map. Through this representation it is possible to express the probability of an event (expressed as a value for a given variable or for a set of them) given the state of a system. The state consists in the values of all the other variables of the system. A BN also expresses direct dependencies between variables. The network is an oriented graph where the value of a child node is influenced by the values of its parents.

The term "Bayesian Network" was first used in 1985 [72]. In the following year two papers summarized the properties of these networks and established BNs as a research field [73] [74]. In the years, Bayesian Networks have been used for modeling knowledge in a very wide range of fields such as decision support systems, computational biology, bio-informatics, medicine, document classification, information retrieval, semantic search, image processing, data fusion, engineering, gaming and law.

In this section we describe the elements of a BN and how to use it to obtain predictions about the future state of the variables (Sect. 3.3.1). We also introduce how this technique can be used to learn the relations among variables (Sect. 3.3.2).

3.3.1 Bayesian Network theory

In this section we introduce the main elements of a BN and how this representation can be used to predict future states of a monitored system.

As stated before, a BN represents dependencies among variables in a very concise way. It is a Directed Acyclic Graph (DAG), where each node is annotated with probability information. A definition of BN is given in [75], where three main aspects are highlighted:

1. nodes correspond to a random variable that can be discrete or continuous;
2. nodes are connected through a set of directed edges. If the edge goes from node X to node Y, than X is a parent of Y;
3. a probability table is associated to each node, quantifying the effect of the parents on the node.

The parent-child relationship describes a causal dependency between nodes, where the parent is the cause while the child is the effect. In order to build a BN, the domain expert has to decide the direct influences in the domain, building in this way the structure of the network. Once the structure is ready, it has to be enriched with a conditional probability distribution for

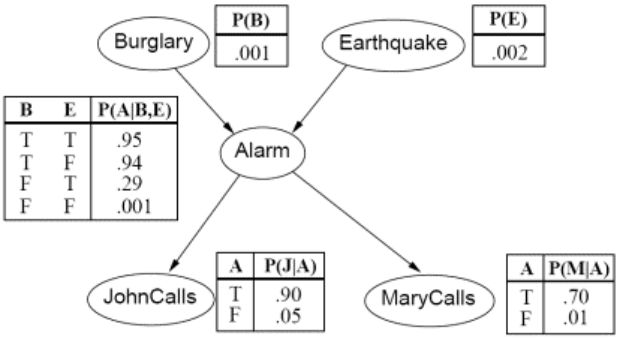


Figure 3.1: *The alarm network representation*

each node given its parents. A typical example of a complete BN is the alarm network shown in Fig. 3.1. This network describes a domain related to a burglar alarm. This alarm is reliable in detecting a burglary, but it can also be activated when a minor earthquake occurs. Two neighbors, Mary and John, have been asked to call the owner where they hear the alarm. However, John confuses the alarm with the phone, while Mary often fails to hear the alarm. Given all this information, and the evidence about who has called or not, it is possible to estimate the probability of a burglary.

Conditional distributions are represented in a Conditional Probability Table (CPT). Each row in the table is a possible combination of values for the parents. In the example, all the variables are binary, so only the probability p of a true case is represented, while the probability of false is omitted since it can be obtained from $1 - p$. For instance, we consider the CPT table for the node *JohnCalls* in Fig. 3.1. According to the first row, the probability of John calling when the alarm is ringing is $P(J|A) = 0.90$. Consequently, the probability of John not calling under the same conditions is $P(\neg J|A) = 1 - 0.90 = 0.10$. If we consider now the second row we know that $P(J|\neg A) = 0.05$, while $P(\neg J|\neg A) = 0.95$, meaning that it is very unlikely for John to call when the alarm is not ringing. If the variables are not binary, all the values have to be explicit. A node with no parents contains only the prior probability for each possible value of the variable. CPT tables can be observed in Fig. 3.1.

Given these premises, we can illustrate how to use BNs to infer knowledge about the domain they represent. To understand the semantics of a BN we have to define the joint distribution of its variables. A joint distribution is the probability of the conjunction of particular assignments to each vari-

able. It can be expressed as $P(X_1 = x_1 \wedge X_2 = x_2 \wedge \ldots X_n = x_n)$, or more concisely as $P(x_1, x_2, \ldots, x_n)$. In the equations, capital letters represents variables and lower-case letters represents the values of the variables. Since in a BN the probability at each node is expressed as $P(X_i|Parents(X_i))$, the joint probability is:

$$P(x_1, \ldots, x_n) = \prod_{i=1\ldots n} P(x_i|Parents(X_i)) \tag{3.1}$$

Using the alarm example we can compute the probability that the alarm has been activated if neither a burglary nor an earthquake has occurred and John called while Mary did not. This can be expressed as[2]:

$$\begin{aligned} P(a, \neg b, \neg e, j, \neg m) =& P(j|a)P(\neg m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ =& 0.90 \times 0.30 \times 0.001 \times 0.999 \times 0.998 = 0.00027 \end{aligned} \tag{3.2}$$

The probability has been computed considering Eq. 3.1 and the CPTs in Fig. 3.1. For each node we consider the row in the table corresponding to the evidence. Starting from the higher level, nodes *Earthquake* and *Burglary* have no parents, so the value corresponding to their evidence is selected ($P(\neg b)$ and $P(\neg e)$). The node *Alarm* has two parents, *Earthquake* and *Burglary*. Since the evidence on both parents is false, the row to be considered in the CPT for node *Alarm* is the last one ($P(a|\neg b, \neg e)$). Finally, both the nodes *JohnCalls* and *MaryCalls* have as single parent the node *Alarm*, which evidence is true. Then, the first row of the CPT has to be considered in both cases, according to the evidence on these nodes ($P(j|a)$ and $P(\neg m|a)$).

Knowing the CPTs we can discover the probabilities of all the possible states.

Bayesian Networks provide conditional independence features. This means that the probability of a node is independent from the probability of its predecessors, given its parents. It is also true that the node probability is independent from the probability of its non-descendants given their parents. In the alarm example this means that the probability of Mary calling does not depend from the probability of John calling if we know the probability of the alarm, and it does not directly depend from the probability of burglary and earthquake too. This is because, given the probabilities of all

[2] a = Alarm, b = Burglary, e = Earthquake, j = JohnCalls, m = MaryCalls

its parents and children (the *Alarm* node in the example), its value is independent from all the other nodes. On the contrary, the probability of the alarm to ring depend from all the other variables in the network, which are its parents or children. In general, conditional independence can be defined as follows:

Definition 1. *The conditional independence property between nodes in a Bayesian Network implies that a node is independent of all other nodes given its parents, children and children parents.*

The alarm network is composed of binary variables, but the same concepts can be extended to discrete variables. However, many real-world problems involve continuous variables. In these cases, it is possible to apply thresholds to divide the values into ranges obtaining discrete values. The results are sometimes good, but most of the time this artifact introduces loss of accuracy and large CPTs [75] with a number of rows depending on the number of values the variable can take. This is also the case of our domain, in which we have found an alternative solution for not applying discretization, as described in Ch. 6.

3.3.2 Learning Bayesian Networks

Sect. 3.3.1 has introduced Bayesian Networks and their properties stating that an expert is in charge of defining both the relations between variables and the Conditional Probability Tables. This can be a very complex and time consuming task. In the state of the art there are a lot of techniques to learn a BN from data. Learning a BN consists into two phases: learning the structure and learning the parameters. Both these phases are described in details in [76], that is our reference for this section.

Learning the structure of a Bayesian Network

In this paragraph we describe how to learn the structure of a BN from data, starting from the methodology described in [74]. The problem of learning structure can be seen as a model selection problem. It consists of determining and selecting the DAG with the maximum probability of fitting the data. The ability of a DAG to fit the data can be evaluated using some scoring functions that will be discussed later in this section. Generally speaking, we can have multiple DAGs with the same probability. Instead of considering all the DAGs, it is possible to consider the DAG patterns. In order to define what a DAG pattern is, we have to introduce the concept of Markov Equivalence Class:

Definition 2. *Let G_1 and G_2 be two DAGs containing the same set of variables V, then G_1 and G_2 are Markov equivalent if for every three mutually disjoint subsets $A, B, C \subseteq V$, A and B are conditional independent given C in G_1 if and only if A and B are conditionally independent given C in G_2:*

$$I_{G_1}(A, B|C) \Leftrightarrow I_{G_2}(A, B|C) \tag{3.3}$$

Starting from the definition of Markov Equivalence Class, we can define a DAG pattern as:

Definition 3. *A DAG pattern for a Markov Equivalence Class is the graph that has the same links as the DAGs in the equivalence class and has oriented all and only the edges common to all of the DAGs in the equivalence class. Other edges are not oriented.*

A DAG pattern can be seen as a generalization of the set of DAGs considered, where not oriented edges represent uncertainties about the direction of a connection. This reduces the number of DAGs to be explored. However, this approach is not feasible in most of the cases, when the number of variables is not small, since the complexity of a learning algorithm grows together with this number. The exploration of all the DAG patterns is an NP-complete problem. For this reason, this problem is usually handled by using heuristic search algorithms. The most common techniques used to approximate the search for the best DAG pattern is the K2 algorithm. The K2 algorithm, introduced by [77], is a greedy search algorithm that tries to improve the score of a BN by gradually introducing parent-children relations between nodes. The algorithm starts from an ordered set of nodes with an empty parent set. At each step it adds a parent to each node that maximizes the score. The main drawback of this procedure, is that it requires an ordering of the nodes. In fact, given a list of nodes, a node X_i can be a parent of a node X_j only if it precedes X_j in the list of nodes.

Another approximation method consists in considering more than one DAG as the best one. When we have lots of data and a small number of variables, there is an high probability that one structure is definitely more likely than another. When the number of variables increases and available data are not so much, it is very likely to find several structures with similar scores. In this cases it is possible to perform inference by averaging over models: it consists in performing inference using each DAG pattern and multiply the result by the posterior probability of the structure. This approach is also useful when the goal of selecting a structure is not inference, but the discovering of relations among variables. Considering several

structures with the same score, it is possible to discover popular relations between variables.

As stated before, structure learning is a selection problem that consists in selecting the DAG pattern that better fits data. We also look for a DAG pattern to be concise. To perform this selection, we need to define a scoring criteria to compare different DAG patterns. Def. 4 defines the properties that a score must have to be considered consistent with a model:

Definition 4. *Given a dataset d_M of M values of a set of variables, a scoring criterion s and a joint probability distribution P_M determined by the data d_M, a score is consistent for a Markow Equivalence Class $\mathcal{D}$ if:*

1. *For M sufficiently large, if $\mathcal{D}_1$ includes P_M and $\mathcal{D}_2$ does not, then*

$$s(d_M, \mathcal{D}_1) > s(d_M, \mathcal{D}_2). \tag{3.4}$$

2. *For M sufficiently large, if $\mathcal{D}_1$ and $\mathcal{D}_2$ both include P_M and $\mathcal{D}_1$ has smaller dimension than $\mathcal{D}_2$, then*

$$s(d_M, \mathcal{D}_1) > s(d_M, \mathcal{D}_2). \tag{3.5}$$

Different scoring criteria are available in the state of the art. According to [78], scoring functions can be grouped into two main classes:

- **Bayesian scoring functions** assign a score to a Bayesian network by computing the posterior probability distribution. The best DAG is the one that maximizes the conditioned probability $P(D|\mathcal{D})$, with $\mathcal{D}$ the dataset containing examples of values of the nodes in the network;
- **Information-theoretic scoring functions** assign a score dependent from the compression that can be obtained over the data $\mathcal{D}$ given a BN structure D. For this reason, this score is also called Minimum Description Length (MDL).

In Sect. 6.4 both classes are considered and tested.

Learning the parameters of a Bayesian Network

The general assumption in parameter learning is to know the structure of the BN without knowing the CPTs. The aim of parameters learning is to learn conditional probabilities from data. In this section we illustrate the method described in [76]. The first step consists in creating an augmented BN as described in Def. 5:

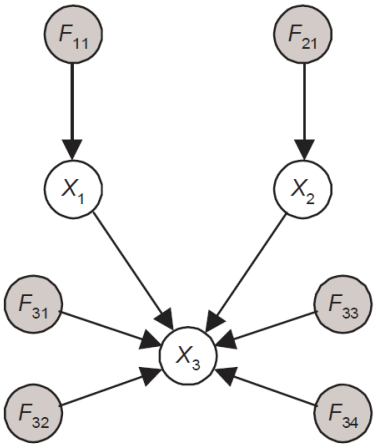


Figure 3.2: *An augmented Bayesian Network in the case of binary variables [76]*

Definition 5. *An **augmented Bayesian Network** is a Bayesian Network defined by:*

1. *a DAG $\mathbb{G}$, composed by N random variables X_i;*
2. *a set of auxiliary parent F_{ijk} for each X_i and a density function ρ_{ijk} for each F_{ijk}. Each F_{ijk} is a root node and have a single edge towards X_i. The number of auxiliary parents is equal to the number of parents J of X_i times the number of values K that each parent can take;*
3. *a probability distribution of X_i conditional on all the parent nodes values pa_i and of all the auxiliary nodes F_{ijk}.*

An example of augmented BN can be seen in Fig. 3.2 where binary variables are considered. The values F_{ijk} are beliefs about the outcome of the variable X_i. More explicitly:

Definition 6. *The probability distribution of the random variable F_{ijk} represents our belief concerning the frequency with which the variable X_i equals to k given that the parent j was equal to k*

$$P(X_i = k|f_{ijk}) = f_{ijk} \Rightarrow P(X_i = k) = E(f_{ijk}) \tag{3.6}$$

where $E(f_{ijk})$ is the expected value of f_{ijk}.

This value is usually computed by an expert that is able to specify an a-priori probability distribution for a variable. The belief over the values of

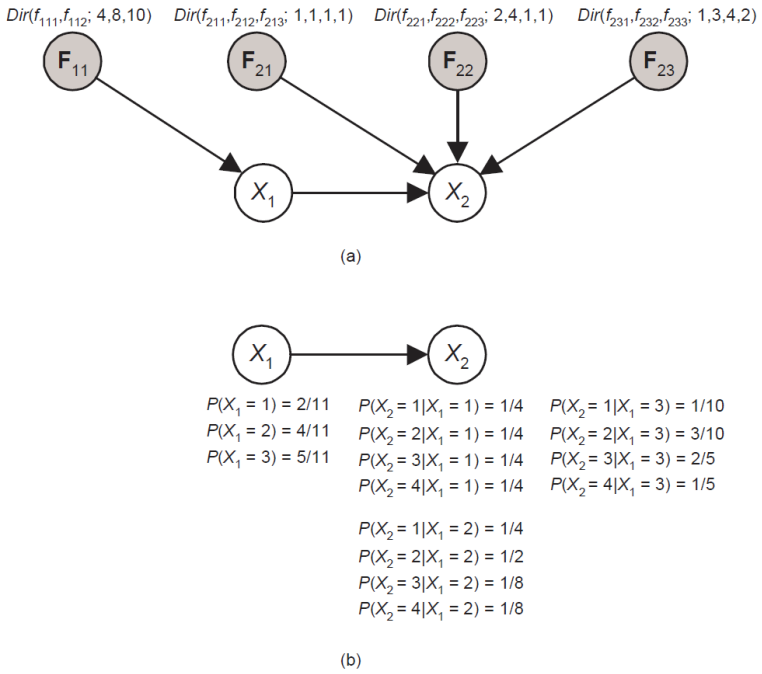


Figure 3.3: *Learning parameters from the augmented BN [76]*

the random variables F are usually represented using a **Dirichlet density function** denoted as:

$$Dir(f_1, f_2, \ldots f_{K-1}; a_1, a_2, \ldots a_K) \tag{3.7}$$

where parameters a_k are proportional to the probability of taking value k.

The initial belief can be updated from data, observing the outcomes of different experiments. Since in the network the nodes F_{ijk} are all independent, the updates can be performed independently. This learning phase enables the learning of the parameters in the BN that can be derived from them using Eq. 3.6. An example of how CPTs are computed in a simple case can be seen in Fig. 3.3.

3.4 Adaptive Operator Selection: Discovering the Strategy

The main feature of adaptive systems is their ability to recover from undesired states, applying different adaptation strategies based on the context. One of the issues to be solved is how to select the best adaptation strategy from a pool of strategies. One possible technique is the AOS, usually

applied in the context of evolutionary algorithms but general enough to be applied to different and more generic problems [79].

The scenario for applying such a paradigm is characterized as follows:

- the algorithm has to choose between different available and well defined options affecting the state of the system;
- the best option is supposed to change under different conditions or with the evolution of the system;
- a feedback can be collected after the execution of an option that can be used for evaluation.

Given these premises, the AOS is suitable in contexts in which several adaptation options are available, with an unknown but measurable effect that can change over time. Moreover, the algorithm is responsive to variations in the system, being able to automatically change its belief if the system is modified (new options are added or an option is no longer effective).

In this section we describe the general AOS algorithm, referring to the work described in [80], in which AOS algorithms are discussed in details and some variations are proposed to improve their behavior. The AOS is an example of *Exploration versus Exploitation*. The exploitation would bring to use the best known operator as much as possible to have better performances, while the exploration tries other less performing operators in order to allows the system to dynamically adapt to variations and to avoid the noise due to unlucky trials for some operators.

The general AOS algorithm can be synthesized in few steps:

1. whenever an adaptation strategy has to be enacted, the AOS module is questioned;
2. the AOS module returns the best strategy to be applied, based on recent performance of all the options;
3. the selected strategy is applied impacting the system;
4. the impact is observed and it is transformed into a *credit* associated to the adaptation option;
5. the credit is used to update the *quality* estimate of the option.

The algorithm is repeated in a loop, whenever the system needs to be adapted. According to the algorithm, two main mechanisms need to be better defined: the credit assignment and the operator selection. The former observes the impact of a selected option over the system and transforms this observation in a quality estimation of the option itself. The latter selects the best option to be applied, given quality estimations for all the options available. These two mechanisms are discussed in the rest of the section.

The choice of this algorithm has been dictated by a consolidated knowledge available in the working group in relation to general adaptation strategies and in more details to the AOS paradigm.

3.4.1 Credit assignment

The credit assignment consists into two phases: the measurement of the impact due to the application of an adaptation operator on a system, and the computation of a credit subsequent to the impact value.

Most of the approaches in the state of the art use the AOS paradigm associated with evolutionary algorithms. In this case, the best way to assess the impact of an operator is the fitness value obtained after its application compared to a reference value which can be the fitness of the parents [81] or of the previous population [82]. A simpler and more general approach consists in assigning a binary value to the impact of an operator: 1 if successful and 0 is unsuccessful [83]. Other techniques use multi-modal optimization, considering more than one dimension for assessing the success of the operators. In case of evolutionary algorithms, these two dimensions are usually the fitness and the population diversity, in an attempt to avoid local maximum. Several dimensions can be integrated into a single impact value by considering a weighted sum of the different dimensions [84], or using the Pareto dominance paradigm [85].

Once the impact has been measured, a credit has to be computed. The simpler way to compute a credit is to consider the impact value as is. However, this technique is unstable, since it considers only the most recent impact value in the credit estimation. More stable techniques take into account a fixed number of previous applications of the operator, using a sliding time window $\mathcal{W}$, and aggregating values in the considered period of time [86] [87]. When the impact is measured using a binary value, credit is simply computed as the success rate of the operator in the considered period.

3.4.2 Operator selection

The operator selection mechanism is in charge of selecting the best operator given the results of the credit assignment. Most of the approaches associate an application rate to each operator basing this rate on a quality estimation. Given the application rates for all the available operators, the operator is probabilistically selected using a wheel-like process. The most common technique for operator selection is Probability Matching, proposed by Goldberg in 1990 [88]. In Probability Matching, the probability of selecting an operator is proportional to its quality estimation. Given the number of available operators K, for each operator the algorithm keeps two values: the probability $p_{k,t}$ and the quality estimation $q_{k,t}$. The operator selection using Probability Matching is performed as follows:

1. an operator k is selected by the algorithm using a wheel-like selection scheme;
2. the operator is applied and a credit $r_{k,t}$ is computed by the Credit Assignment module;
3. the quality for operator k is updated considering the new credit as
$$q_{k,t+1} = (1 - \alpha)q_{k,t} + \alpha \cdot r_{k,t} \tag{3.8}$$
where the value α is used to tune the importance of the previous applications of the operator;
4. the probabilities of application of all operators are updated proportionally to their new quality estimation:
$$p_{j,t+1} = \frac{q_{j,t+1}}{\sum_{l=1}^{K} q_{l,t+1}} \tag{3.9}$$
Computing probabilities according to Eq. 3.9, a bad operator would easily reach a probability 0 of being selected. Future improvements of its application would be impossible to discover. In order to avoid this inconvenient, a minimum probability can be assigned to each operator in any case, modifying the probability assignment as:
$$p_{j,t+1} = p_{min} + (1 - K * p_{min})\frac{q_{j,t+1}}{\sum_{l=1}^{K} q_{l,t+1}} \tag{3.10}$$

The AOS paradigm is very general and effective for application in systems that dynamically change their behavior in time. For this reasons we use this paradigm with some modification is Ch. 7, where we describe a dynamic adaptive selection of adaptation strategies based on context.

3.5 Conclusion

In this chapter we have analyzed adaptation and learning techniques that can be applied in an Information System. First of all we have considered the problem of adaptation in a Service Oriented Architecture where adaptive techniques are useful in order to react to constraints violations and to system faults. Then, we have seen how it is possible to learn relations among elements and variables of a domain using the Bayesian Network representation. Finally, we have analyzed the Adaptive Operator Selection technique for automatically selecting an effective operator in a dynamic environment.

In this thesis we are going to use all the concepts introduced in this chapter to build the model described in Ch. 5. In the techniques introduced in Sect. 3.2, the goal of adaptation was the management of QoS violations. In our approach we are going to consider both QoS and EE as drivers of adaptation in the system. BN learning is used to automatically learn the relations among the metrics used to assess the system state, as described in Ch. 6. We also use the AOS technique to automatically learn the effect of a set of predefined adaptation actions over the metrics and to provide a technique for the selection of the best strategy based on the system context (Ch. 7).

The aim is to provide adaptation in an Information System based on service provisioning. The resulting adaptive system will be able to automatically adapt to modifications in the environment by updating its model of the world and automatically managing changes in the system and in the action effects over the metrics behavior.

Part II

Assessing and Modeling Energy Efficiency in a Data Center

CHAPTER 4

Energy Efficiency and Quality of Service Metrics

Indice

In order to be able to assess Energy Efficiency (EE) in data centers, we analyze and define a set of metrics that can be used to monitor the state of the system and to drive further improvements. Green Performance Indicators (GPIs) provide measurable means to assess the EE of a resource or system. Most of the metrics commonly used today measure the EE potential of a resource, system or application usage from a manufacturer perspective, rather than the EE of the actual usage. However, the way in which

the resources and systems are actually used by an application in a given data center configuration is at least as important as the efficiency potential of the raw resources or systems. Hence, for data center EE, we suggest to both select efficient components (as done today), as well as optimize their actual usage in the data center. To achieve the latter, optimization of application usage centric GPIs should be employed and targeted as a primary green goal.

4.1 Introduction

The assessment of the state of a system is a primary activity that needs to be executed whenever we want to move an organization towards a greener asset. This activity can be seen as composed of two main tasks: the monitoring and measurement of the system and the evaluation of the system state. These two tasks are strictly connected since what should be monitored depends on which parameters are used for the evaluation. The selection of these parameters is a crucial activity for the success of the assessment and results in the definition of metrics and indicators.

The class of systems we aim to monitor (and improve) is a coherent set of servers composing a single data center. The intended purpose of the system class is to support a Business Process (BP). A Business Process can be described as a work-flow of activities (also referred here as applications). It is deployed over these servers using virtualization. We assume that each activity runs on its own dedicated Virtual Machine (VM). The work flow has conditional branching which depends on the user interaction but which can be estimated from historical data. The system's monitoring sub-components will periodically sub-sample the system state. The number of variables to be monitored and the rate for data collection are dependent from the number of servers and VMs involved.

A representation of one example of this class of systems is shown in Fig. 4.1. It is composed of 2 servers running 2 and 3 business activities respectively, each executed in its own VM.

Before discussing the metrics selection issue in more details we need to introduce some terminology. Since the topic has been faced from different perspectives in the last decades, we can refer to the IEEE standard glossary of software engineering terminology published in 1990 [89].

First of all the term metric is defined as follows [89]:

> *"A **metric** is a quantitative measure of the degree to which a system, component, or process possesses a given attribute."*

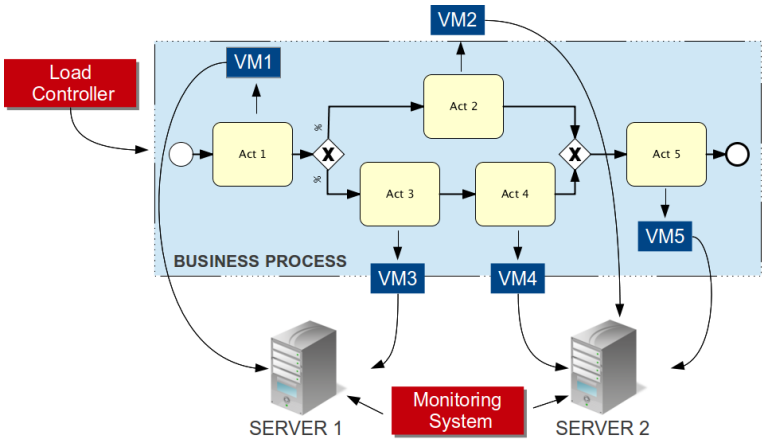


Figure 4.1: *An example of the considered class of systems*

Connected with this definition, a qualitative metric is defined [89]:

> *"A **qualitative metric** is a quantitative measure of the degree to which an item possesses a given quality attribute."*

According to this definition, the qualitative metrics are the parameters that have to be computed from measurements of the system behavior.

Finally, the term indicator is defined as follows [89]:

> *"An **indicator** is a variable that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition."*

An indicator can be seen as an instance of a metric, applied to a specific aspect of a system and associated with a comparative term used to assess if the monitored value is in line with the prescribed one or not. The terms metric and indicator are often used interchangeably due to their strict relation.

Given a set of metrics, the system administrator has to be able to choose a subset of them able to lead the system towards the goals of the organization, that can be related to different aspects. In our case they are related to Energy Efficiency (EE) and Quality of Service (QoS). Once the metrics are selected, a monitoring system has to be installed or configured accordingly to this selection, and some constraints and reference values are defined for each indicator, dependent on the pursued goals.

According to this, we define the system state as follows:

Definition 7. *The **state of the sytem** at a given time t is composed of the values of all the metrics selected to lead the system towards the predefined goals at the time t:*

$$state(t) = m_1(t) \dots m_n(t) \dots m_N(t) \tag{4.1}$$

where $m_n(t)$ is the value of metric n at time t and N is the number of selected metrics.

The aim of this chapter is to introduce the concepts of metrics and indicators and to give definitions of the most common metrics used in the state of the art that are related to EE and QoS, and their associated indicators, known respectively as Green Performance Indicators (GPIs) and Key Performance Indicators (KPIs).

After a brief description of the main contributors to the definition of EE and QoS metrics for data centers (Sect. 4.2), we give a detailed definition of the concept of indicator and we define the concepts of operational and qualitative dependency among them (Sect. 4.3). Then, we introduce the concept of application usage centric metrics in Sect. 4.4, stressing the importance of two different perspectives in the measurement of the indicators: the manufacturer perspective, aimed at the selection of the most efficient components to obtain a better performance, and the proposed application usage perspective, focused on the best utilization of the available resources. Sect. 4.4.3 gives an example of a possible selection of metrics for pursuing the goal of a green and efficient Service Oriented Architecture (SOA) (see Sect. 3.2 for more details about Service Oriented Architectures).

The content of this chapter has been partially published in [35] and [90][1].

4.2 State of the Art

As stated in Sect. 2.1, one of the most efficient drivers for EE are the governmental regulations. In this section we survey the principal drivers and their contribution to the field.

The **U.S Environmental Protection Agency (EPA)** submitted in 2007 a report to the U.S congress on server and data center EE [91], projecting that from 2006 to 2011 the energy consumed by data centers would be doubled. The EPA suggested that there is a significant potential for energy savings in data centers, identifying servers as the primary source of Information

[1] In [35] and [90] "application usage centric metrics" are simply referred as "usage centric metrics". Here we decided to rename them in order to avoid confusion with the resource metrics described later in the chapter, which focus on usage but without the application perspective.

Technology (IT) energy consumption in data centers, second only to the site infrastructure. Therefore, the EPA introduced its ENERGY STAR Data Center Energy Efficiency Initiatives: the ENERGY STAR Rating for Data Centers, which provides tools to evaluate the efficiency of the data center, and the ENERGY STAR Data Center Product Specifications, which focus on requirements for data center products. Several indications have been published in the recent years, as the ENERGY STAR Computer Server Specification program version 1.0 [92], the ENERGY STAR Data Center Storage Specification [93], and the ENERGY STAR Uninterruptible Power Supplies Specification [94].

The **Green Grid** is a global consortium dedicated to advancing EE in data centers (see also Sect. 2.3). The Green Grid publishes EE metrics that are widely used in data centers. The most known is Power Usage Effectiveness (PUE) [95], which measures the data center infrastructure energy overheads for running the IT equipment. The Green Grid set out to chart data center productivity metrics, which define the energy cost of useful work. Unfortunately, defining what is useful work is not simple. Therefore, the Green Grid defines proxies [96] for useful work productivity metrics.

The **Storage Networking Industry Association (SNIA)** is an industry association dedicated to developing standards for storage. SNIA is actively pursuing energy efficient storage through its Green Storage Initiative [97]. It has also developed the SNIA Emerald [98] program and specific benchmarks for evaluating the EE of the storage system. SNIA Emerald is based on the Storage Power Efficiency Measurement Specification [99] currently open for public review. The specification has three components: (i) an idle measurement that focuses on what SNIA defines idle-ready state of a storage system, measured in GB/Watt, (ii) a workload performance metric based on I/Os per second, measured in IOPS/Watt, and (iii) a bandwidth performance metric based on the throughput of the storage systems, and measured in MBPS/Watt.

The **Storage Performance Council (SPC)** [100] is a non-profit corporation with the goal of defining storage benchmarks and to provide an authority for verifiable performance data. SPC published two benchmark specifications: SPC-1, which focuses on online transaction processing and SPC-2, which focuses on sequential data processing. In 2009 SPC released an energy extension for SPC-1 that provides a performance/energy metric. SPC also provides an insight into the expected energy consumption of the storage by defining heavy, moderate, light and idle workloads. Using the data from the benchmark, the expected energy cost and the amount of hours

per day spent in each workload type, it provides an estimate of the annual storage energy cost.

The **Standard Performance Evaluation Corporation (SPEC)** [101] is a non-profit corporation that defines and creates a standardized set of relevant benchmarks that can be applied to the newest generation of high-performance computers. SPEC provides a set of benchmarks that address topics such as Central Processing Unit (CPU), graphics, Java, Power, Virtualization. SPEC started to work on a benchmark that evaluates both performance and power, and resulted in SPECpower_ssj2008. In the process of creating this first benchmark, SPEC, has created a power and performance benchmark methodology [102] that is now used to add power metrics to existing benchmarks, instead of creating energy specific benchmarks.

As can be observed, the main drivers described in this section make an extensive use of benchmarks in order to measure and compare the EE and the performance of the tested environment. This evaluation is performed by using a set of metrics that express how the system behaves during the benchmark execution. Each benchmark can be based on different metrics, focusing on different aspects. We believe that evaluating the behavior of a system exclusively through the application of benchmarks is not enough. Benchmarks are useful to state the system behavior under a specific kind and amount of workload, by enabling a generic classification of the system performance. However, benchmarks are not useful to provide a run time evaluation of the system performance, and to detect possible failures of the system in terms of EE and QoS when several and different concurrent applications are executed. To reach this knowledge, a composite and wide set of metrics can be monitored and analyzed during the whole lifetime of the considered system. In the rest of the chapter, these metrics are defined and discussed.

4.3 Indicators Definition, Relations and Granularity

An indicator can be defined as a measure of a relevant aspect of an organization or a system. This kind of measurement is often used to assess the success of an entire organization or of a specific department. The success assessment is lead by the definition of strategic goals or, at a lower level, of organizational goals. The selection of a relevant set of indicators is an important and not trivial activity in the life-cycle of a system, since it requires a good understanding of what is important to the organization. Usually, performance aspects are privileged, but even the economical and green as-

pects should be considered. Indicators allow the organization to assess the current state of the system and to identify the key activities. At the same time they enable the identification of potential improvements.

Each Indicator I_k can be defined in terms of a metric:

$$I_k = f(D_k) \tag{4.2}$$

that defines the formula used to evaluate it, where D is the set containing all the raw data obtained through the monitoring system and $D_k \subseteq D$ is the subset of raw data used in the assessment of indicator I_k. For example the GPI $I_1 = CPU\,usage$ can be associated with the set $D_1 = \{CPUused, CPUallocated\}$.

In general, indicators are not independent, and can be related to each other. In some cases, an indicator can be defined as a function of other indicators. The defined relationship is called *Operational Dependency* and can be expressed as:

$$OpInd_k = Op(I_k) \tag{4.3}$$

where $OpInd_k = \{I_1 \dots I_n\}$ is the set of indicators used to compute the indicator I_k. According to this, Eq. 4.2 can be generalized as:

$$I_k = f(D_k, OpInd_k) \tag{4.4}$$

In this case there is a direct relation among indicators that is expressed in the evaluation formula. In summary, indicators can be defined as aggregations of different elementary variables or indicators, which are defined at the same or at a lower level.

Relations among indicators can also be indirect. This kind of relation is called *Qualitative Dependency*, and can be expressed as:

$$QInd_k = Q(I_k) \tag{4.5}$$

where $QInd_k = \{I_1 \dots I_n\}$ is the set of indicators that are correlated with the indicator I_k. Qualitative dependencies are useful to predict the behaviors of indicators after a modification of the system state and can be found using data mining or machine learning techniques. In this thesis, we propose a machine learning based approach discussed in details in Ch. 6.

Indicators can be defined at different granularity levels. Granularity depends on the level of detail used to measure the factors composing the formula of an indicator. Typically, a system is designed to store the indicators values at the finest level of granularity, but the values can be aggregated along different dimensions in order to enable users to achieve business

goals. In order to understand the reasons why a data center system has performed in a certain way, it is often necessary to analyze data starting from a summary followed by detailed level analysis (i.e., drilling down operation), and vice versa. Possible dimensions that can be used to aggregate indicators values are time, infrastructure component, middleware component, application, geographical areas, etc. An intuitive example of analysis, in which indicators can be explored at different granularity levels, is the resource usage metrics. In this case the indicator measures the amount of a resource used relative to the amount of resource available. Each of these indicators can be measured at the server level (including all the applications running on the same server) or with a finer granularity at the application level, measuring the amount of resources actually used by a single monitored application (relative to the amount of resources reserved for it). Aggregation can also be applied at a finer level of granularity to clusters. In this case measurement can be made at each node and aggregated using a weighted sum in order to obtain a single value for the whole cluster.

Sect. 4.4 introduces a subset of the indicators available in the state of the art, by distinguishing between two perspectives for measuring indicators: the manufacturer perspective, which is focused on the efficiency of the elements of the data center, and the application usage perspective, which focuses on the efficient usage of these elements.

4.4 Usage Centric Metrics

In Sect. 4.3, we defined the concept of indicator and we highlighted the importance of the selection of a good indicator set to correctly assess the state of a system and/or organization and to lead future developments. In this section we report the definition of some indicators in the state of the art related to EE, called GPIs, and to QoS, known as KPIs. Then, we propose a new perspective for the measurement of these indicators.

In every system it is necessary to control the energy consumption while maintaining a high QoS. This is even more important in a SOA, where Service Level Agreements (SLAs) have been established with clients and have to be satisfied. For this reason, it is important to have the correct tools to evaluate the state of the system (see Def. 7) from both an EE and a performance perspective. This can be done through a monitoring system associated with the definition of a set of metrics.

The purpose of defining GPIs and KPIs is to provide measurable means to assess the EE or performance of a resource or system. If appropriate

metrics are defined, then making changes to the system which results in better metrics levels should improve EE. Hence, defining "good" metrics is a central step towards optimization. Currently, a myriad of GPIs have been defined and are used by various organizations. Many of the common metrics measure the EE of a single resource (e.g., Hard Disk Drive (HDD), or CPU), whereas others measure resources and entire systems behavior. Some metrics assess the EE of a "raw" device, and others assess the EE of their usage. All metrics commonly used have merit to some extent, and are useful tools to compare components and systems, and make informed decisions and choices based on comparative analysis.

In this work, we argue that the way that the resources and systems are actually used in a given data center configuration is at least as important as the efficiency potential of the raw resources or systems. Hence, for data center EE, we suggest to both select energy efficient components (as done today), as well as optimize the actual usage of the components and systems in the data center from an application perspective. To achieve the latter, optimization of application usage centric GPIs should be employed and targeted as a primary green goal.

Consider a simple example of capacity EE of a 300GB HDD measured in GB/Watt. Clearly, a disk with a higher value of this metric provides more efficient raw storage than one with a lower value. Note that a large percentage (60-70%) of the energy consumed by this HDD is expended whether the entire or a small portion of the space is used, due to the energy cost of spinning the disk. In one system configuration the space provided by this HDD may be fully used by applications (say 90% of the available 300 GB raw storage), whereas in another configuration only 10% of the capacity is actually used, out of the same HDD. Are we satisfied to use the raw capacity metric (which has identical value) in both cases? Had we measured the energy cost (GB/Watt) of the used capacity of the HDD rather than measuring the raw capacity, we would have reported a much higher efficiency value for the 90% usage case, and a reduced carbon footprint. The example is also shown in Fig. 4.2.

Based on the above motivation, we argue that defining and using usage centric GPIs has merit as EE metrics, and should be defined and used at all system levels as a primary means of assessing a system's EE. In this section we identify and present application usage centric metrics, which should be monitored and optimized for improving EE, and hence, reduce the data center carbon footprint. To this extent, we survey the most important metrics in the state of the art, distinguishing between two perspectives for measur-

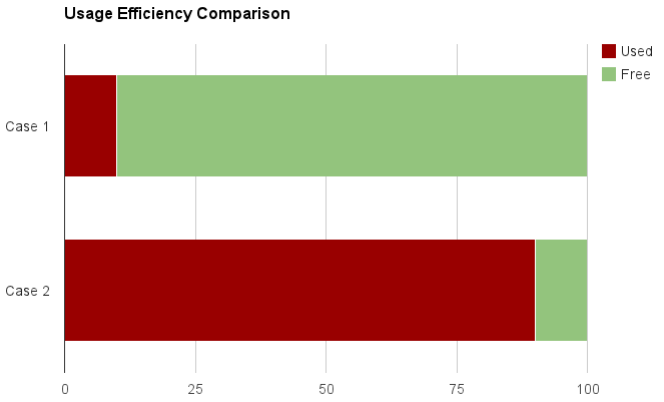


Figure 4.2: *Two HDDs with the same size but different usage level. Powering a HDD has the same cost independently from its usage. A highest usage is more efficient.*

ing indicators: the classical manufacturer perspective (Sect. 4.4.1) and the proposed application usage perspective (Sect. 4.4.2).

4.4.1 Classical metrics survey and classification

As previously discussed, indicators can be classified in two main classes: GPIs and QoS indicators or KPIs (see also Sect. 2.4.2). In the literature many indicators have been suggested to measure the performance and greenness of a system. Note, however, that some of these metrics cannot be measured easily due to the difficulty in retrieving the raw data through the monitoring system. The subset of the most relevant indicators has to be identified on the basis of the business context and the characteristics of the system itself.

Indicators can be classified on the basis of what they monitor, estimating the energy consumption of the application itself and the environment in which the application is run. Inside the framework of the GAMES project [103], we identified four clusters of metrics for monitoring the system state: (i) facility and data center, (ii) energy impact, (iii) application, and (iv) resources. The first cluster is focused on environment measures, such as air conditioning, temperature and humidity. Since in this work we focus on IT metrics, this cluster is not considered in the rest of the discussion.

Energy Impact Metrics

This cluster of metrics is strictly related to the EE of hardware and software components. Many energy related parameters can be considered, such as power supply, consumed materials and emissions. Hardware components from different vendors have different EE properties. In order to measure these properties it is possible to use performance benchmarking tools with incorporated EE measurement. These tools are used to stress the system towards high loads and to observe its performance in comparison to some sample systems. Some representative metrics included in this cluster are:

Capacity metric: measures the energy consumed within the storage facility [104] and it is computed as the ratio between *raw* space (GB) and power (Watts). Used storage space is defined as the space used by files written and stored on the storage system. Power is measured as the average power of the storage system under typical usage, measured over a representative (long enough) time period:

$$Capacity\,Metric = \frac{Capacity_{storage}(GB)}{Watt} \tag{4.6}$$

I/O throughput metric: this indicator expresses the EE of I/O operations (i.e., data read and write) [104] considering the number of operations performed in the considered benchmark, computed as the ratio between *predefined benchmark I/O rate* (IOPS) and power (Watts):

$$I/O\,throughput = \frac{Number\,of\,IOPS}{Watt} \tag{4.7}$$

Data Transfer throughput metric: expresses the EE of I/O operations considering the rate of transfer of the amount of data involved in the I/O transactions [104], measured as the ratio between *predefined benchmark data transfer rate* (MBPS) and power (Watts):

$$Data\,Transfer = \frac{MBPS}{Watt} \tag{4.8}$$

Performance per Energy indicator: measures the energetic efficiency of a *benchmark test*, calculated on the basis of the number of floating point operations per second (FLOPS) and the energy consumed in a time period [105]. It is measured as:

$$Performance\ per\ Energy = \frac{FLOPS}{kWh} \tag{4.9}$$

CO_2 Emissions: measures the amount of average carbon dioxide (CO_2) emissions for the generation of a kWh of electricity. It usually depends on the source of energy used. It can be expressed as:

$$CO_2\, emissions = \frac{CO_2}{kWh} \tag{4.10}$$

PUE: The PUE [106] is the most common metric used to assess the EE of a data center and is calculated as a ratio between the total facility power consumption and the IT equipment power consumption. PUE is calculated using the following formula:

$$PUE = \frac{Total\, Facility\, Power}{IT\, Equipment\, Power} \tag{4.11}$$

An efficient way to compute this metric has been provided in [30], as previously discussed in Sect. 2.4.2.

Data Center Infrastructure Efficiency (DCiE): it is used to determine the EE of a data center and it is referring to how much energy the IT equipments consume from the total energy consumption. This is expressed as a percentage, calculated as the inverse of PUE by dividing IT equipment power by total facility power [106]:

$$DCiE = \frac{IT\, Equipment\, Power}{Total\, Facility\, Power} \tag{4.12}$$

Application Metrics

Application metrics do not directly reflect the greenness of an application or of a data center. These indicators measure the performance of an application, and express the quality of the process and the efforts needed to design and maintain it. These metrics include QoS indicators that refer to the analysis of the non-functional properties of the application [107] [108] [109].

Response Time: it expresses the time taken by a service or an activity for handling user requests, measuring the expected delay between the moment when a request is sent and the moment when it is rendered. Response time RT is determined using the processing time $T_{process}$ and the transmission time T_{trans} as follows:

$$RT = T_{process} + T_{trans} \tag{4.13}$$

Throughput: it is defined as the average number of requests served successfully during a given time period.

$$Throughput = \frac{Served\,Requests}{Time} \tag{4.14}$$

Availability Rate: it is defined as the average rate of availability of a service or activity S to be accessed, meaning the probability that a request is correctly responded within a maximum expected time frame. A measure of the availability can be calculated from monitored values as:

$$Availability = \frac{Number\ of\ successful\ requests}{Number\ of\ requests} \tag{4.15}$$

Resource Metrics

In the attempt to measure the energy consumed by a server or an application the resources they are using within the system must be considered. Intuitively, the efficiency of the system depends on the efficiency of resources allocation among the running applications. The cluster of resource metrics measures the efficiency of resource usage relative to available resources. Consequently, their values are expressed as a percentage. Examples of indicators in this category are [110]:

CPU Usage: this GPI relates to the CPU utilization, and can be measured as the ratio between the CPU used and the total CPU available at the *server level*:

$$CPU\,Usage = \frac{Used_{CPU}}{Available_{CPU}} \tag{4.16}$$

Memory Usage: this GPI refers to the usage of the main memory (RAM) by a server or an application, and it is computed as the ratio between the memory used and the total memory available at the *server level*:

$$Memory\,Usage = \frac{Used_{Memory}}{Available_{Memory}} \tag{4.17}$$

Storage Usage: this GPI refers to the *entire storage utilization percentage* for data read and write operations on the corresponding storage device, and can be computed as as the ratio between the disk space used and the total disk space available at the *data storage level*:

$$Storage\,Usage = \frac{Used_{DiskSpace}}{Available_{DiskSpace}} \tag{4.18}$$

Deployed Hardware Utilization Ratio (DH-UR): this indicator measures the amount of active deployed servers as [111]:

$$DH - UR = \frac{Number\,of\,active\,servers}{Number\,of\,Deployed\,Servers} \tag{4.19}$$

Deployed Hardware Utilization Efficiency (DH-UE): this indicator measures the efficiency of the deployed servers taking into account the amount of servers needed to face a peak load and it is measured as [111]:

$$DH - UE = \frac{Minimum\,number\,of\,servers\,for\,peak\,load}{number\,of\,deployed\,servers} \tag{4.20}$$

For this cluster of indicators, we observe a complex situation. On one hand, it is obvious that the less a resource is used to achieve a result, the more energy efficient is the application, and hence, a lower value for these indicators is desirable. On the other hand, often it is not possible to reduce the amount of resources used by the system and their energy cost has already been paid (i.e., resources have already been configured and allocated). Hence, improving system EE means that a higher value for the indicator could be preferable, matching applications needs with allocated system resources.

4.4.2 Application usage metrics redefinition

In the following we provide examples for application usage metrics for servers and storage. This examples aim to strengthen our argument regarding the importance of usage centric EE metrics.

While the metrics definition is very similar to metrics described by the organization referenced in Sect. 4.2, the main difference is how the metrics are computed. Therefore, the metrics provide the actual usage perspective and allow applications and administrators to evaluate and improve the EE.

CPU Usage: this GPI relates to the CPU utilization, and can be measured as the ratio between the CPU used and the total CPU available at the *application level*:

$$CPU\,Usage = \frac{Used_{CPU}}{Allocated_{CPU}} \tag{4.21}$$

Memory Usage: this GPI refers to the usage of the main memory (RAM) by a server or an application, and it is computed as the ratio between the memory used and the total memory available at the *application*

level:

$$Memory\,Usage = \frac{Used_{Memory}}{Allocated_{Memory}} \tag{4.22}$$

Storage Usage: this GPI refers to the *application storage utilization percentage* for data read and write operations on the corresponding storage device, and can be computed as the ratio between the disk space used and the total disk space available at the *application level*:

$$Storage\,Usage = \frac{Used_{DiskSpace}}{Allocated_{DiskSpace}} \tag{4.23}$$

Capacity Metric: it represents the EE of storing the data of the user's applications. The metric is defined as the ratio between storage space *used by applications* (GB) and power (Watts). Used storage space is defined as the space used by files written and stored on the storage system by the application. Power is measured as the average power of the storage system that is due to the application utilization, measured over a representative (long enough) time period:

$$Capacity\,Metric = \frac{Capacity_{application}}{Watt} \tag{4.24}$$

I/O throughput metric: this indicator represents the energy cost of the storage system while running the user applications workload. The metric is defined as the ratio between *application I/O rate* (IOPS) and power (Watts). Applications I/O rate is measured as the number of I/O operations per second that an application executes. Power represents the average power consumed by the storage system while running typical workload:

$$I/O\,throughput = \frac{Number\,of\,Application\,I/O\,operations\,per\,second}{Watt} \tag{4.25}$$

Data Transfer throughput metric: it expresses represents the same metric as the above I/O throughput, but for data transfer (MBPS). The metric is the ratio between *application data transfer rate* and power. The two throughput metrics emphasize different aspects of the same efficiency. I/O throughput is commonly used for evaluating random workloads, while data transfer throughput is commonly used for evaluating sequential workloads.

$$Data\,Transfer = \frac{Application\,related\,MB\,moved\,per\,second}{Watt} \tag{4.26}$$

Table 4.1: *Comparison between manufacturer and application usage perspectives.*

Metric	Manufacturer Perspective	Application Usage Perspective
CPU Usage	the ratio between the CPU used and the total memory available at the *server level*	the ratio between the CPU used and the total memory available at the *application level*
Memory Usage	the ratio between the memory used and the total memory available at the *server level*	the ratio between the memory used and the total memory available at the *application level*
Storage Usage	the ratio between the disk space used and the total disk space available at the *data storage level*	the ratio between the disk space used and the total disk space available at the *application level*
Capacity Metric	the ratio between storage *raw* space and power	the ratio between storage space *used by applications* and power
I/O throughput	the ratio between *predefined benchmark I/O rate* (IOPS) and power	the ratio between *application I/O rate* (IOPS) and power
Data Transfer throughput	the ratio between *predefined benchmark data transfer rate* and power	the ratio between *application data transfer rate* and power
Performance per Energy	the ratio between the number of *floating point operations* and energy	the ratio between the number of *transactions* and energy

Performance per Energy indicator: it measures the energetic efficiency of an *application*, calculated on the basis of the number of transactions and the energy consumed in a time period [105]. It is measured as:

$$Performance\ per\ Energy = \frac{Number\ of\ transactions}{kWh} \tag{4.27}$$

where a transaction is an atomic unit of work performed by the application.

The above EE usage metrics describe how the user application utilizes the system resources. Hence, the above usage metrics are defined via terms that are directly linked to a user application. The metrics definitions with the two perspectives are compared in Tab. 4.1.

Table 4.2: *Green Performance Indicators and Key Performance Indicators*

Indicator	Description
CPU Usage	The amount of CPU used as a fraction of CPU allocated. It is a number between 0 and 1. It can be measured both at the VM and at the server level
Response Time	Time between the start and the completion of a specific activity instance in the service. It is measured in seconds.
Energy	The amount of energy used by a component in the system. It is measured in kilowatt-hour
Performance per Energy	The number of operations that each activity executes per energy unit, expressed in kilowatt-hour.
Memory Usage	The amount of memory used as a fraction of memory allocated. It is a number between 0 and 1. It can be measured both at the VM and at the server level
I/O throughput	The number of I/O operations that each activity executes per time unit
Storage Usage	The amount of disk space used as a fraction of total disk space allocated. It is a number between 0 and 1. It can be measured both at the application and at the server level

4.4.3 The GAMES experience: a subset of metrics

Following the experience acquired in the GAMES project [103] we propose a set of indicators to assess the system state. These indicators are summarized in Tab. 4.2. They can be evaluated from different perspectives (manufacturer and application usage perspective) and at several granularity levels (e.g. data center, server, VM, service or activity). For instance, CPU usage can be measured both at server and VM levels, as well as memory usage, storage usage, I/O throughput, and performance per energy (measured in FLOPS at the server level and in transactions at the activity level). Response time can be evaluated for the whole service or for the single activity while energy can be computed for a server, a VM or for the whole data center.

The set of indicators defined in this section is used as a reference point in the rest of this thesis.

4.5 Conclusion

Metrics related to EE and QoS provide measurable means to assess the state of a resource or system. Many of the commonly used IT metrics today

measure the potential efficiency of a resource, typically from the manufacturer's point of view. Using this approach allows comparing the potential efficiency of various alternative resources, and favors the more efficient one over others. Given the selected resources, however, we argue that the way that the resources are actually used in a given data center configuration is at least as important as the potential efficiency of the raw resources. Optimization for data center efficiency must target both the selection of energy efficient components (as done today), as well as the optimization of the actual usage of these components. For the latter, optimization of application usage centric GPIs should be employed and targeted as a primary green goal.

The application usage metrics can be evaluated continuously, as opposed to potential metrics which can be computed only once and on a reference system. Data center administrators can review the changes in the metric efficiency over time and also compare with other data centers. In practice, comparing and selecting resources based on their raw EE is relatively easy. The manufacturers test their components and advertise their efficiency results. Measuring the application usage centric metrics for a given data center configuration, and optimizing the configuration to improve the metrics levels is much more complex. It requires expertise and configuration/workload planning that many data center operators are only now beginning to practice.

In this chapter we identified and presented an approach to properly measure (via application usage centric metrics) and optimize system usage for EE and carbon footprint.

CHAPTER 5

A Goal-Oriented Model for Green Data Centers

Indice

Given a set of metrics and a monitoring system, the state of an Information System (IS) can be assessed comparing the monitored values with

some reference values contained in policies or subscribed in the Service Level Agreement (SLA) representing the best range. In this chapter we propose a goal-driven model to lead the system towards a better configuration, inspired by similar models available in the requirement engineering field. The model is composed of two layers: a goal layer and a treatment layer. The goals are identified with the indicators satisfaction and a methodology to set their satisfaction range is proposed. The treatment layer is composed of a set of repair actions that can be used to adapt the system and to repair undesired states for the goals. Based on techniques available in the state of the art, a set of repair actions is defined and classified.

5.1 Introduction

Measuring the Energy Efficiency (EE) of a system is only the first step towards greenness and efficiency. Once the current situation is clear, the second step consists in comparing the system state to a set of reference values in order to discover weaknesses.

The set of indicators described in Ch. 4 can be used to investigate the state of the system under several perspectives and levels. The evaluation of the indicators value is not significant by itself, but it has to be compared to some reference values in order to understand the level of efficiency of the system. This goal is accomplished setting thresholds for each indicator. Indicators can be enriched with thresholds indicating the range of satisfaction for a given aspect of the system. Every time an indicator is not inside the normal region, the system state is suboptimal according to that aspect. When a suboptimal situation occurs, something has to be done to improve the system state.

According to this, the system that we are considering can be seen as a goal-driven system, where the goals are represented by the indicators satisfaction. The system is at its optimum when all the goals are satisfied (i.e. all the indicators are in their normal range). However, defining thresholds is not an easy task. Thresholds definition has an important impact over the system behavior and should be done carefully. In this chapter we use indications contained in the maturity models (see Sect. 2.3) as a reference for the definition of satisfaction ranges for indicators. In particular, we refer to the Green Grid Data Center Maturity Model (DCMM) [13], where several aspects of the data center are considered in the attempt of improving EE.

After a brief summary of the Green Grid DCMM and of some goal-driven approaches in Sect. 5.2, we discuss the proposed approach in Sect. 5.3

and we analyze the two layers of the proposed model in Sect. 5.4 and Sect. 5.5. We also discuss an experiment in which we used the defined thresholds for leading EE in Sect. 5.6.

Part of the content of this chapter has been published in [112].

5.2 State of the Art

Recent years stressed the attention over the improvement of EE in data centers. One of the organizations who focused on this aspect is the Green Grid consortium[1]. Green Grid is an international consortium of companies, government agencies, and educational institutions focused on improving the overall EE in data centers by providing recommendations on best practices, metrics, and technologies. The consortium identified the need of a comprehensive model for improving sustainability and EE in data centers, developing the DCMM [13]. Indications contained in the model allow the system administrators to benchmark the current performance of their system and to identify its maturity level. They also enable the possibility of developing a short time and/or long time plan for reaching a higher maturity level. The model addresses the assessment of the maturity level of different aspects of the system that can be grouped into two main categories: facility and Information Technology (IT). This allows managers to better understand which parts of the data center are excelling and which present the largest opportunity for improvement. The two main categories are divided into subcategories, focusing on several details of the system:

- Facility: (i) Power; (ii) Cooling, as cooling contribution to Power Usage Effectiveness (PUE); (iii) Other Facilities; (iv) Management, including monitoring, and PUE;
- IT: (i) Compute, including utilization, workload management, and power management; (ii) Storage, including workload and technology; (iii) Network, including utilization, workload, and technology.

Six maturity levels are identified by the consortium for each of the subcategories, with a description of what should be implemented in the system for belonging to that level. The six levels can be summarized as follows:

- **Level 0**: it represents an organization in which efforts towards EE are minimal or completely absent;

[1] http://www.thegreengrid.org/

- **Level 1**: it describes an organization in which some efforts are made towards efficiency, but best practices are only partially implemented;
- **Level 2**: it is considered as the state of the art optimal level and it includes organizations where best practices are implemented;
- **Level 3**: it pushes the line towards a system more efficient than the state of the art;
- **Level 4**: it represents another step towards a visionary efficient system;
- **Level 5**: it is the visionary level supposed to be the future state of the art in 2016.

The same organization can belong to different levels for different aspects, allowing to focus on the main weaknesses. From the indications contained in the DCMM it is possible to extract some thresholds for the indicators considered in our system, as later discussed in Sect. 5.4.

Once the indicators are selected and their optimal values are defined, it is possible to look at them as goals for the EE of the data center. Several authors propose a goal oriented approach for requirements satisfaction. An example is proposed in [113]. The model is composed by three layers, named asset, event, and treatment, defined as follows:

- *Asset Layer*: it is composed of the goals of the system and their relationships. Relationships represent mutual effects over goals satisfaction. In traditional goal-driven models, the asset layer represents stakeholders' business goals which are taken from the application requirements model;
- *Event Layer*: events are defined as significant environmental changes that are occurring at the present moment or are expected to occur in the near future. For that reason all events are labeled as either "predicted" or "real". Events can have a positive or negative impact on goals satisfaction.
- *Treatment Layer*: looking at minimizing the negative impact of events over the goals, treatments are enacted from the treatment layer. Treatments mitigate the impact of an event over a goal, or prevent the event occurrence.

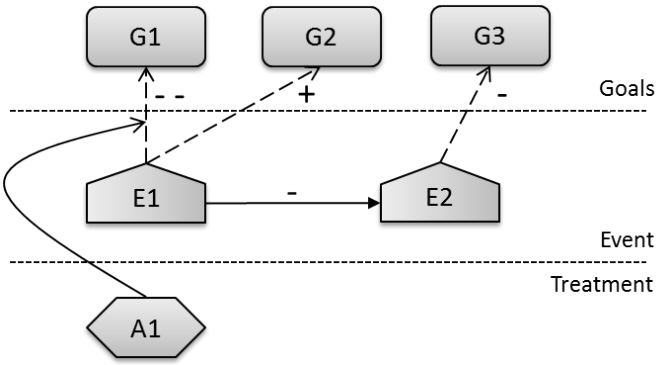


Figure 5.1: *A simplified representation of the three layers goal-driven model.*

A representation of the model is shown in Fig. 5.1. As can be observed, relationships between the elements of the system are labeled with a symbol, representing the kind and the amount of the impact of the predecessor over the ancestor. The kind can be "positive" or "negative", while the amount of impact can be "strong" or "weak". The described model properly fits the adaptive system that we aim to design for maintaining EE and Quality of Service (QoS) in a Service Oriented Architecture (SOA). Other research groups have used the same model as a reference point. An example is reported in [42], where the model has been adapted to include also EE goals. This work has been discussed in more details in Sect. 2.5.1. Another example can be seen in [114] and [115], where the authors use again the model for taking decisions about how to improve EE while maintaining the required QoS. These last contributions focus principally on the Event Layer. Goals are represented by a set of metrics related to both EE and QoS, for which the value has to be maintained inside a given interval. When a violation occurs, an event is raised. The adaptation process is composed of two phases: the event creator and the adaptation strategy selector. In the first phase, the state of the system is analyzed in order to identify the goal violations that can be considered significant. In fact, not every violation can be categorized as an event. Temporary violations can be ignored since they can automatically recover without enacting any strategy. So, in order to raise an event, the violation should lasts in time. According to this approach, events are described by: a duration window, a direction (increasing/decreasing), a priority and a severity. When one or more events are raised, actions can be enacted to lead the system towards a normal state. An action is selected based on the events and past history. Actions are described by: duration,

cost, conditions, and group (performance or resources). Actions have to be parametrized by the system administrator before the execution.

In the following section we describe a goal-oriented model which takes inspiration by the one described in [113], and by its adaptation discussed in [114].

5.3 A Goal-Driven Approach for Energy Efficiency

Indicators are the key to lead a system towards a desired goal. They give relevant information about the actual system state and the desired state, allowing focusing attention over significant aspects that need to be improved.

In this work, we propose a goal-driven approach to EE, obtained from a simplification of the model proposed in [113], following the path started in [114] and [115]. We consider the satisfaction of the indicators thresholds as a set of goals that need to be achieved in order to have the optimal situation. Relations exists between different goals and their satisfaction, since indicators are not necessarily independent from each other. That means that a goal satisfaction/violation can have effects on other goals of the system.

Whenever a goal is not satisfied, something needs to be done. We indicate these treatments as *repair actions*. Each action can have a positive or negative outcome over one or more indicators in the system. Given the relation among indicators and actions it is possible to decide which is the best repair strategy in order to bring the system in a new configuration as near as possible to the optimal one.

A representation of the approach is shown in Fig. 5.2, where two layers are represented. The first layer is composed by the indicators and the specification of their relations. The second layer is composed by the set of repair actions, and it is connected to the system goals through links between actions and goals. A link indicates an effect of the action over the indicator. This effect can be both positive or negative. Also this information is included in the model. A positive effect means that the action improves the indicator state, while a negative one means that the indicator is negatively affected by the action. An action can have opposite effects on different indicators.

The main difference with the basic approach is the removal of the event layer. However, even if this layer is not shown in the model, it is implicitly modeled in the system. As described in [115], events are used for notifying the violation of an indicator, by considering constraints about the duration of the violation and its relevance. Both these aspects are implicitly con-

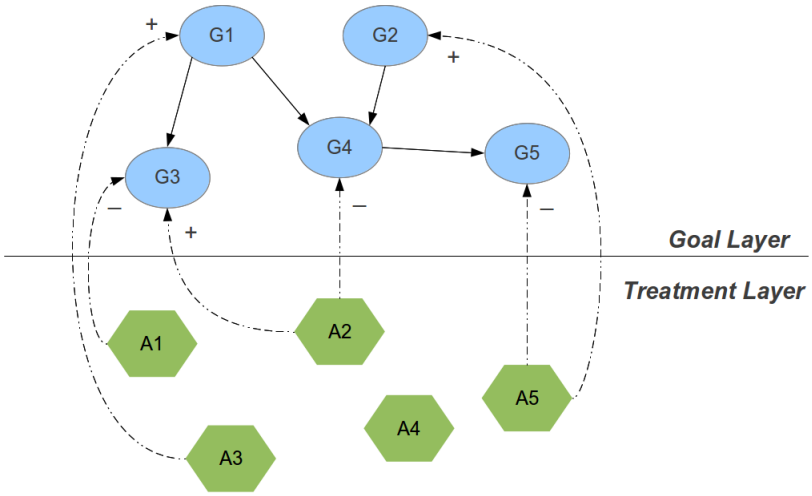


Figure 5.2: *The goal-driven approach for Energy Efficiency*

sidered in the proposed model. For simplification we are simply referring to the violation of a threshold over the indicator value for enacting a repair strategy. However, even if not explicitly shown in the system, we are referring to the event creator described in [115] to consider only relevant violations, while ignoring temporary ones. Since the vent rising policies have been deeply analyzed in [115], we are not describing this aspect in our model, but we are assuming that the violations in our system are raised only when the constraints of the event generation are satisfied.

In the rest of the chapter, we analyze the model in more details. In Sect. 5.4 the goals of the system are defined and analyzed in details. In Sect. 5.5 the treatment layer is analyzed defining a set of available actions and their connection to the top layer elements.

5.4 Goals Definition: Mapping Indicators in the Green Grid Model

Indicators defined in Ch. 4 allow the data center administrator to investigate the state of the system. However, some thresholds need to be defined for comparing the current situation with the optimal one and for understanding which aspects should be improved. The satisfaction of these thresholds can be considered as the goal of the system.

In this section we consider the subset of indicators used in the GAMES experiments [103] and introduced in Sect. 4.4.3 and we propose a method-

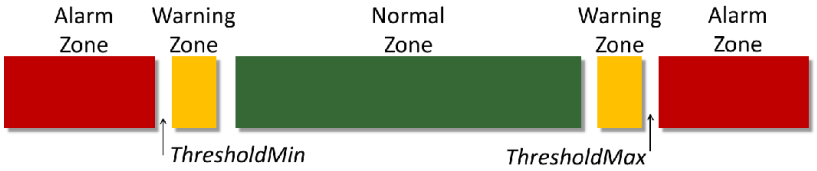


Figure 5.3: *Indicator thresholds intervals with five states*

ology for assessing the goals satisfaction.

5.4.1 Goals definition through thresholds

Once a set of metrics has been identified as relevant, they can be computed starting from row data obtained by the monitoring system. Metrics can be collected and/or aggregated at several granularity levels. For each of these levels, users can operate a restriction on the admissible range of values, according to the content of the Service Level Agreement (SLA). Setting the constraints, the users identify the values related to desired (satisfaction zone) and undesired (alarm zone) behavior for each metric [116]. Also a warning zone is defined slightly before the alarm zone, and its range of values is identified automatically on the basis of the designer requirements. In fact, such a range of values depends on the strategy that the designer wants to adopt for the system improvement, that can be proactive or reactive. Adopting a proactive approach implies that a warning is triggered before reaching a real violation, so that the system can react in advance. Proactiveness is realized setting an extended warning zone between the satisfaction and the alarm zone. On the contrary, using a reactive approach implies a warning zone very close to the alarm threshold. In this case, the system reacts only when it is significantly close to a violation. A general representation of the zones for the metrics is shown in Fig. 5.3, even if for some metrics either *ThresholdMin* or *ThresholdMax* can be defined. As an example, Central Processing Unit (CPU) usage indicators need to be described with 5 values, since both high (performance) and low (efficiency) bounds have to be defined. On the contrary, response time needs only three values, since the lower bound is not considered critical. A value from 1 to 5 can be assigned to each of the zones represented in Fig. 5.3, increasing left to right. This value is used for identifying the state of an indicator.

5.4.2 Deciding thresholds values

In order to decide the satisfaction intervals for each indicator we referred to the Green Grid DCMM. Since our approach focuses on IT, the facility category of the DCMM is not further considered, while we focus on the IT category and in particular on the "compute" and "storage" sub-categories. In Tab. 5.1, a subset of parameters of the Green Grid DCMM (related to IT factors) is mapped to the selected indicators for a wise selection of the indicators thresholds.

In this work, we map our selected set of indicators (see Tab. 4.2) in the categories and subcategories of the DCMM, and take into account the recommendations contained in the several levels for setting the thresholds of the goals for our system. From the *Compute* category perspective we consider *utilization*, *workload*, *operations*, and *power management* subcategories. Considering the *Compute:Utilization* subcategory, the DCMM suggests CPU usage set at 20% for level 2 (best practices) and greater than 60% at level 5. Similarly, the *Compute:Workload* is related to several indicators, since it concerns the correct utilization of the available resources. At the best practice level, the application of virtualization and consolidation is required, while level 5 moves toward the cloud paradigm adoption, proposing a "Follow the Moon" strategy by redirecting the workload on several distributed servers. The *Compute:Operation* sub-category is mainly related to the performance of the system and to the utilization of resources related to computation such us CPU and memory. In Level 2, a simple comparison with predefined benchmarks performance is required, while at level 5 the intent is to allow a wiser utilization of resources at the application level. From the *Compute:Power Management* subcategory perspective, indicators such as energy and Performance per Energy are involved. Best practices require the monitoring of the power consumption, while a level 5 system should optimize power usage while maintaining performance. From the *Storage* category point of view, two aspects are considered. The *Storage:Operation* subcategory relates to how the storage is used. At the best practice level, storage consolidation is required, while at level 5 more sophisticated techniques are suggested, including an improved storage utilization at the application level, and a Total Cost of Ownership (TCO) driven selection of the storage media (i.e. solid state vs disk, cloud, etc.). The other storage aspect is related to the *Storage:Technology* subcategory, where level 2 organizations just need to properly select the storage devices, while level 5 organizations are required to implement low power state utilization.

Table 5.1: *Contribution to Green Grid Data Center Maturity Model*

Parameter	Level 2 (Best Practice)	Level 5	Involved GPIs
Compute: Utilization	CPU 20%	CPU 60%	CPU Usage
Compute: Workload	Rationalize Workload (virtualization/consolidation)	Shift all of the workload across many data centers taking into account business priorities, external drivers, availability of resource and TCO - "Follow the Moon" strategy	CPU Usage, Memory Usage, Performance per Energy, IO throughput, Energy
Compute: Operations	Understand performance through the use of standard benchmarks	Improve application use of processor, memory and major power consumption components	CPU Usage, Memory Usage
Compute: Power Management	Power Monitoring	Optimization of power with no impact over Performance	Performance per Energy, Energy
Storage: Operation	Storage Consolidation	Improved application use and creation of data, media choice based on TCO model	Storage Usage, Energy
Storage: Technology	Low Power Consuming Technology	Use/enable low power states for storage	Storage Usage, IO throughput, Energy

5.5 Action Definition: Controlling the Indicators Values

Once a set of constraints has been identified for each indicator, several techniques can be applied in order to keep the indicators inside the desired interval of values. This goal can be achieved using the treatment layer introduced in Sect. 5.3 through the definition of a set of repair actions.

A repair action is a modification of the system configuration that can affect the state of one or more indicators. In this work, a repair action is described by the following set of elements:

- ID: an identifier for referring to the action;
- Short Description: a brief description of the repair action;
- Parameters: the parameters needed to apply the action;
- Execution Time: the average time needed to apply the action and to observe its outcome;

- Energy Consumption: the amount of energy needed to execute the repair action;
- Preconditions: the conditions that have to be verified for applying the action.

Several repair actions have been identified and their description is reported in the rest of this section. The selected actions have been chosen starting from the most common strategies proposed in the state of the art for dealing with EE, previously discussed in Sect. 2.5. Also the experience acquired in the GAMES project [103] has contributed to the selection of the set of strategies that better fits our context of application. Actions are classified into four categories: virtual level, server level, process level, and general. A summary of the available actions is reported in Tab. 5.2.

5.5.1 Virtual level repair actions

This set of actions concerns the virtual level of the system by managing Virtual Machines (VMs) configuration, placement, and deployment.

VM Migration This repair action moves a VM from a server to another. This can be useful when a server is overloaded or when it is underloaded and moving the VMs deployed on it enables the server switch off. Parameters are the ID of the VM that has to be migrated and the ID of the server where it has to be moved. In order to be executable the selected server must have the resources requested by the VM. The amount of time required to execute migration is usually considerable and can be approximated using models available in the state of the art, as described in [117]. Given the time required to perform migration and the computational resources involved in the process, also the cost in terms of energy has to be considered. To model this cost we refer to the model proposed in [54].

VM Reconfiguration - add This action enables to change the amount of resources allocated to a VM according to the incoming load. Parameters are the ID of the VM that has to be modified and the kind of resource that is involved (e.g. CPU and memory). Reconfiguration is executed using fixed steps that consist in the addition of a core in case of CPU or of a fixed amount of memory (e.g. 125 MB) otherwise. Precondition for this action is that additional resources are available on the server. In this work we consider only hot-plugging, so resources can be added without stopping the machine. This feature is available in the most popular hypervisors

Table 5.2: *Repair Actions*

VIRTUAL MACHINE LEVEL		
ID	**Description**	**Parameters**
A_1	VM Migration	VM ID, Server ID
A_2	VM Reconfiguration - add	VM ID, resource type
A_3	VM Reconfiguration - remove	VM ID, resource type
A_4	VM Duplication	VM ID
A_5	VM Removal	VM ID
SERVER LEVEL		
ID	**Description**	**Parameters**
A_6	Turn on Server	server ID
A_7	Turn off Server	server ID
A_8	Storage Migration	none
A_9	CPU Frequency Scaling - increase	server ID
A_{10}	CPU Frequency Scaling - decrease	server ID
A_{11}	Storage Mode Modification - performance	disk ID
A_{12}	Storage Mode Modification - energy saving	disk ID
PROCESS LEVEL		
ID	**Description**	**Parameters**
A_{13}	Process Work-flow Modification - skip	activity ID
A_{14}	Process Work-flow Modification - execute	activity ID
GENERAL		
A_{15}	Do Nothing	activity ID

(e.g. VirtualBox and VMware). According to this, both execution time and energy cost of this action are limited.

VM Reconfiguration - remove This action enables to change the amount of resources allocated to a VM according to the incoming load. Parameters are the ID of the VM that has to be modified and the kind of resource that is involved (e.g. CPU and memory). Reconfiguration is executed using fixed steps that consist in the removal of a core in case of CPU or of a fixed amount of memory (e.g. 125 MB) otherwise. Precondition for this action

is that the minimal amount of resources is guaranteed for the VM (at least one core and one block of memory). Considerations about energy cost and response time are the same as for the add resources action.

VM Duplication When a VM receives more requests than the amount that can be handled, it is useful to duplicate it in order to distribute the load between more VMs. Parameter of this action is the ID of the VM to be duplicated. Precondition is that there is a server with enough resources for hosting the VM. In this action, execution time needed to duplicate the VM and run its clone has to be considered and can be not negligible. Also the energy cost of starting a new VM has an impact on the system.

VM Removal When more than a VM handles a given service, and the incoming load does not justify all the available duplications, it is useful to remove one of the available copies. Parameter of this action is the ID of the VM to be removed. In this case, the time and the costs to be considered are the same required for turning off a VM.

5.5.2 Server level repair actions

This set of actions concerns the physical level of the system by managing the servers and their components.

Turn on Server This repair action turns on a server. It can be useful when there are not enough physical resources available for deploying a new VM. In this case, a server needs to be reactivated. The only parameter is the ID of the server. This action can require a significant, even if limited, amount of time to be executed Also the energy consumption, due to the utilization of resources while activating the server, has to be taken into account. In order to compute this cost, we refer again to the model proposed in [54].

Turn off Server This repair action turns off a server. It can be useful when a server is not used, in order to save energy. The only parameter is the ID of the server. Precondition is that there are no VMs deployed on the selected server. For energy and execution time, the considerations made for the previous action are valid.

Storage Migration This action is used to concentrate storage data on a smaller amount of disks and to optimize data placing according to their usage. No

parameters are required. In terms of time and energy, we can use models similar to the VM migration.

CPU Frequency Scaling - increase This action reduces the frequency of the processor in order to increase performance. It takes as parameters the ID of the server where the CPU is located. Precondition is that the current frequency is not the higher available. The time and the cost of this action are negligible.

CPU Frequency Scaling - decrease This action decreases the frequency of the processor in order to reduce energy usage. It takes as parameters the ID of the server where the CPU is located. Precondition is that the current frequency is not the lower available. The time and the cost of this action are negligible.

Storage Mode Modification - performance Disks can run in a more or less energy efficient modality. This action allows the modification of the disk mode into "performance" mode by taking as parameter the disk ID. Precondition is that the storage was running in energy saving mode. The time and the cost of this action are negligible.

Storage Mode Modification - energy saving Disks can run in a more or less energy efficient modality. This action allows the modification of the disk mode into "energy saving" mode by taking as parameter the disk ID. Precondition is that the storage was running in performance mode. The time and the cost of this action are negligible.

5.5.3 Process level repair actions

This set of actions concerns the application level of the system by managing the Business Processes (BPs) hosted by bye data center.

Process Work-flow Modification - skip In a BP some activities can be considered as mandatory and others can be optional. This action can be used to skip the optional activities. It takes as parameters the activity ID. Precondition is that the selected activity was not disabled. The time and the cost of this action are negligible.

Process Work-flow Modification - execute In a BP some activities can be considered as mandatory and others can be optional. This action can be used to execute the optional activities. It takes as parameters the activity ID. Precondition is that the selected activity was disabled. The time and the cost of this action are negligible.

5.5.4 General repair actions

This group of actions include a single action that is the "Do Nothing" action. This action is used when none of the other actions can successfully be applied even if some violations are present. This action has both execution time and energy cost equals to 0.

5.5.5 Complex repair actions

In some contexts, some actions are useless or even damaging by themselves. As an example, lets consider action A_1 (VM migration) in Tab. 5.2. The migration of a VM is expensive both in terms of energy and performance. Applying this action by itself is not convenient from any point of view. However, it is useful when combined with other actions, such as A_7 (Turn off Server). For this reason, defining the actions described until now as simple actions, it is useful to introduce the concept of composed action. A composed action can be defined as follows:

Definition 8. *A composed action AC consists in a set of simple actions organized in a plan, defining the sequence in which these actions are executed:*

$$AC_i = \langle A_i \rightarrow A_j \rightarrow \ldots \rightarrow A_k \rangle \tag{5.1}$$

where each action A_x belongs to the set of all the available actions A and the symbol $\rightarrow$ indicates the order of execution of the actions in the plan.

Some examples of composed actions are described in Tab. 5.3. For all the actions described in this part, the execution time and the energy cost can be obtained by summing up the costs of the simple actions composing them.

Migrate and Turn Off This action consists in migrating all the VM on a server and then turning off the empty server. The symbol $^{\circlearrowleft}$ which appears in the first row means that the operation has to be repeated several times, in the specific case for all the VMs running on the server that is going to be turned off. It is useful when enough resources are available in other

Table 5.3: *Composed actions definition*

ID	Description	Plan
AC_1	Migrate and Turn Off	$A_1^{\circlearrowleft} \rightarrow A_7$
AC_2	Turn On and Migrate	$A_6 \rightarrow A_1$
AC_3	Migrate Storage and Change Mode	$A_8 \rightarrow A_{12}$

machines and turning off a server reduces significantly the energy usage. It takes as parameter the identifier of the server to turn off and can be applied only if the preconditions for the two actions composing this complex actions are both satisfied.

Turn On and Migrate This action turns on a server and migrate a VM on it. It is useful when more resources are needed and a new server can increase the performance of the system. The action takes as parameter the identifier of a VM and of a server. As before, preconditions for both the simple actions composing it have to be satisfied.

Migrate Storage and Change Mode This action performs migration of data in the storage system and changes the mode of the disks accordingly.

The set of actions described in this section have been connected to the categories of the DCMM by identifying on which of the model parameters each action can have effect. Results are shown in Tab. 5.4 where, starting from what has been summarized in Tab. 5.1, a set of indicators and a set of actions is associated to each of the categories of the DCMM that we are considering.

5.6 Experiment

In this section we consider a scenario where some techniques are applied for satisfying the prescribed thresholds at the virtual level.

Using VMs, as recommended for the best practice level, enables the dynamic allocation of resources to each deployed application. In this experiment, the design of the configuration of VMs for business transaction running the TPC-C benchmark[2] has been analyzed. The simulated process describes a selling system composed of several activities, such as entering

[2]http://www.tpc.org/tpcc/

Table 5.4: *Association between the Green Grid maturity model and the repair actions*

Parameter	Involved GPIs	Repair Actions
Compute: Utilization	CPU Usage	VM Reconfiguration, VM Removal
Compute: Workload	CPU Usage, Memory Usage, Performance per Energy, IO throughput, Energy	VM Reconfiguration, VM Removal, VM Migration
Compute: Operations	CPU Usage, Memory Usage	VM Reconfiguration, VM Removal, VM Migration
Compute: Power Management	Performance per Energy, Energy	CPU frequency scaling, Turn on/off Server, VM Migration
Storage: Operation	Storage Usage, Energy	Storage Migration
Storage: Technology	Storage Usage, IO throughput, Energy	Storage Mode Modification

and delivering orders, recording payments, checking the status of orders, and monitoring the level of stock at the warehouses. Each activity executes a set of transactions against a database.

In order to test our approach, we used an open source implementation of the TPC-C benchmark, namely TPCC-Uva [118]. This implementation reflects all the characteristics of the benchmark. The application, together with its database, has been installed on a VM. Virtualization enables a dynamic environment where resource allocation can change as the load of an application increases/decreases responding to different request rates. The BP application has been installed and executed on an Ubuntu10.10 VM hosted in a Windows XP machine. The used hypervisor is VMWare. In detail, the set of indicators summarized in Tab. 4.2 has been evaluated with several configurations for the VM. The Performance per Energy indicator in this case is computed according to the application usage perspective, as the ratio between the number of transactions executed and the energy cost of the application. Data about resource usage has been collected directly from the operating system of the VM running the TPC-C application. Data about QoS metrics have been retrieved from the log of the benchmark while information about energy consumption of the application has been

estimated using PowerInformer[3], an open-source tool developed by Intel, able to compute power consumption of a single application.

Taking into account the information retrieved by the monitoring system, the value of the indicators can be used to assert if the system can be improved and if resources can be better allocated to avoid wastes. As an example the application can be started in a given configuration, the indicators for the machine running the application can be computed and the state of the system can be evaluated. After that, it is possible to decide if the system can be improved or not. In the experiment we simulate an adaptation process involving reconfiguration of resources at the VM level for solving thresholds violations. The monitored values at each step of the process are shown in Tab. 5.5, while Tab. 5.6 summarizes the VM resource configuration at each step. Unsatisfied indicators are highlighted in Tab. 5.5.

Starting from configuration *Conf. 1* in Tab. 5.6, the monitoring system collects data for the computations of the indicators (*Step 1* in Tab. 5.5). At this first step three indicators are violated: CPU usage, memory usage and storage usage. The violation of the thresholds suggests the administrator to change the VM configuration in order to improve the state of the system by applying the repair action A_3 (VM reconfiguration - remove), selecting for instance the CPU as parameter of the action. The machine can be reconfigured by reducing the amount of cores allocated step by step, until a good configuration is reached (*Step 3* or *Step 4*). Now that CPU usage is satisfied, other violations can be considered. The administrator can increase memory usage passing from *Conf. 4* to *Conf. 5*, using again action A_3, with memory as parameter. We can reduce the amount of memory allocated until we reach an efficient memory allocation (*Step 6*). Now we can reduce the amount of disk allocated (*Conf. 7*) increasing the storage usage indicator and obtaining a system configuration where no violations occur (*Step 7*).

The application tested here is quite static, so the load rate is almost constant. In this case, the adaptation process can be used to find the optimal initial configuration of the system. Once this optimal configuration is achieved, nothing else can be done at run time. In real world applications, however, the load rate changes periodically depending on time, season, etc.. Hence, the proposed approach can result even more useful because the system can keep on monitoring the application behavior through measurement of the indicators, and the resource allocation can be increased or decreased depending on the load. This can ensure the optimal configuration at each moment during the application execution. In the presented BP scenario, it

[3] http://software.intel.com/en-us/articles/intel-powerinformer/

Table 5.5: *Virtual Machine reconfiguration towards Energy Efficiency*

	Perf. per Energy > 10	**CPU usage > 80%**	**Storage usage > 50%**	**IOPS > 90**	**Memory usage > 75%**	**Resp. time < 1 sec.**
Step 1	13,822 trans/kWh	**47.15%**	**38%**	96.15 B/s	**47.25%**	0.034 s
Step 2	15,080 trans/kWh	**67.42%**	**32%**	191.00 B/s	**47.25%**	0.063 s
Step 3	16,250 trans/kWh	99.67%	**38%**	171.11 B/s	**47.45%**	0.048 s
Step 4	17,354 trans/kWh	99.67%	**38%**	336.61 B/s	**46.35%**	0.054 s
Step 5	14,188 trans/kWh	94.21%	**38%**	90.98 B/s	**54.61%**	0.078 s
Step 6	19,419 trans/kWh	91.08%	**48%**	252.14 B/s	97.91%	0.035 s
Step 7	16,300 trans/kWh	94.21%	76%	103.00 B/s	96.98%	0.050 s

Table 5.6: *VM configuration at each step*

	Conf. 1	*Conf. 2*	*Conf. 3*	*Conf. 4*	*Conf. 5*	*Conf. 6*	*Conf. 7*
CPU	4 cores	3 cores	2 cores	1 cores	2 cores	2 cores	2 cores
Memory	2 GB	2 GB	2 GB	2 GB	1 GB	512 MB	512 MB
Disk	10 GB	10 GB	10 GB	10 GB	10 GB	10 GB	5 GB

was possible to save about the 8% of energy between *Step 1* and *Step 7*. This experiment has been published in [119].

5.7 Conclusion

In this chapter we have proposed a goal-driven model for EE in Information Systems (ISs). The model is divided into two layers: the goal layer and the treatment layer. The goals of the model are represented by indicators thresholds satisfaction and a set of repair actions has also been introduced in order to lead the system to an acceptable configuration whenever one or more indicators are violated. Repair actions can be simple or composed. The latter are applied when the application of a plan consisting in the sequential execution of several simple actions is needed. Actions can also be classified into four categories: virtual level, server level, process level, and general. In the proposed model, connections can exist between elements of the first layer (goal to goal connections) and between actions and goals.

A goal to goal connection means that the satisfaction of an indicator influences the satisfaction of the linked one in a positive or negative manner. A connection between an action and a goal implies a positive or negative effect of the action towards the indicator satisfaction.

In this chapter we have defined the structure and the elements of the model but we have not faced the issue of how to connect them. This issue is discussed in the rest of the thesis. More in detail, Ch. 6 proposes a methodology for learning relations among indicators from data collected by the monitoring system, while Ch. 7 discusses a method for dynamically learning the effect of the actions over the indicators from experience.

Part III

Improving Energy Efficiency in a Self-Adaptive Context

CHAPTER 6

Learning Relations Between Indicators

Indice

In this chapter we propose a method for learning relations between the indicators states collected through a monitoring system. Knowing the relations between indicators has a twofold importance. First of all it enables an indirect fixing approach. The violation of an indicator can be fixed acting directly over the indicator, or intervening over other indicators from which it is influenced. Knowing relations among indicators enables this approach. Moreover, a what-if analysis approach is also possible. The relations knowledge allows us to predict the effects of a modification of an indicator state over the other indicators in the system. We decided to represent relations using a Bayesian Network (BN) structure, and to learn these

relations using techniques available in the state of the art.

6.1 Introduction

The monitoring system collects data that allow us to assess the state of the data center in terms of Energy Efficiency (EE) and Quality of Service (QoS) (see Ch. 4). Once these data have been collected, they are used as a starting point to detect and prevent emergency and warning situation. Row data obtained by the monitoring system are subject to the set of thresholds defined in the Service Level Agreement (SLA) and introduced in Sect. 5.4.1.

Knowing the assessed value we know which indicator has to be fixed to get the general state back to the normal situation. However, this knowledge is limited. In a data center, the set of metrics used to assess Energy Efficiency and QoS are not independent in many cases. In some contexts, metrics can have a negative effect on the other, but they can be also related by positive interactions. It means that improving the state of one indicator can improve the state of another one related to him.

For these reasons, it seems very important to investigate about the relations between indicators and to find a way of representing these relations. Knowing relations between indicators has multiple advantages when trying to improve the efficiency (in terms of quality and energy) of the data center. First of all it enables an indirect fixing approach. The violation of an indicator can be fixed acting directly over the indicator, or intervening over other indicators that result to be related to it. Moreover, a what-if analysis approach is also possible. The knowledge about relations allows us to predict how the variables of the system react to a modification of an indicator state, predicting possible critical outcomes.

In this thesis, we started from the goal oriented model described in Ch. 5 and tried to simplify that structure while completing the kind of information represented. In this chapter we focus on the higher level of the model: relations between goals. The proposed approach replaces the higher level of the goal oriented model with a Bayesian Network (BN). The goal is the creation of a Bayesian Network, representing the relations between the indicators violations and/or satisfaction. As described in Sect. 3.3.1, Bayesian Networks are a useful tool to express the relations and interactions between variables. The network is an oriented graph, in which dependencies are modeled. Relations between indicators are represented through edges. A Conditional Probability Table (CPT) corresponds to each edge expressing the relations between each of the states of the two indicators involved. Through this rep-

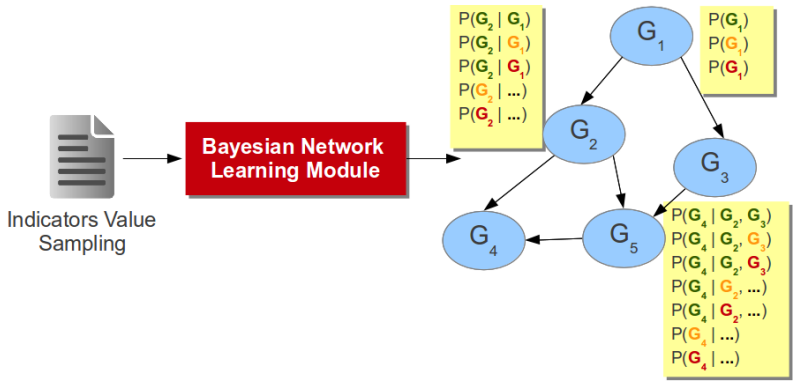


Figure 6.1: *Bayesian Network Learning Approach*

resentation we can learn relations between indicators and conduct what-if analysis and reasoning, by predicting feature outcomes for the indicators state.

The aim of this chapter is to propose a methodology for automatically learning relations between indicators starting from a set of training data, corresponding to the monitoring data collected. Even if an expert could be able to describe positive and negative relations between indicators, this task is very complex and time consuming in a very big system. Moreover, some relations can be hidden and the expert could be unable to represent them. Another reason for using automatic learning techniques is the effort requested from the expert to build the network, an effort that has to be replicated every time the system changes.

A graphical synthesis of the approach is represented in Fig. 6.1 in which the goal of the approach is highlighted. The network is used to describe the relations between the states of the indicators.

In this chapter we introduce some techniques to learn structure and parameters of a BN from data (Sect. 6.2). After that, we discuss the proposed approach for learning relations between indicators. The approach can be divided into three steps. In the first step undirected relations between indicators are discovered (Sect. 6.3); then directions are added to the BN (Sect. 6.4), and finally parameters are added to the structure (Sect. 6.5). Finally, we briefly describe the tools used for the implementation and the experiments in Sect. 6.6.

6.2 State of the Art

Learning structures and parameters of a BN is a very discussed topics in literature. This technique can be used for inferring causal relations between variables, because of the meaning of the edges in the network, but also for information retrieval, natural language processing and classification.

Principal techniques for learning a BN have been discussed in Ch. 3, but they are principally based on the use of discrete variables. When dealing with continuous variables, approximation is required in order to apply thresholds on their values. Some authors try to avoid this problem by using alternative techniques to learn the structure of the network such as correlations. An example of this approach is presented in [120] where authors use correlations in order to discover relations among variables and then try to learn causation over the network. The method consists into two steps. In the first step the algorithm converts a correlation network into a partial correlation graph, that is a graph in which variables are connected using undirected edges. After that, a partial ordering of the nodes is established and this ordering can be used to compute a directed acyclic causal network. Nodes are ordered considering a score based on their standardized partial variances. This approach proved to be effective. However its use is restricted to Gaussian variables and can not be applied into contexts in which variables are continuous but not Gaussian.

Another approach, which separates structure and directionality learning, is proposed in [121], with the Sparse Candidate Algorithm. This approach has been designed to work with a large number of variables and it consists in a search algorithm over the Directed Acyclic Graph (DAG) space, where each node is allowed having parents from a pre-determined parent set of a fixed size. The parent set is estimated using a heuristic and then refined using a hill-climbing approach, until the algorithm converge. In order to limit the number of iterations, the algorithm uses the concept of TABU list, a list containing the structures already explored, in an attempt to escape local maxima.

In [122], authors propose a modified version of the Sparse Candidate Algorithm. They use a two steps approach where in the first step, they detect the most likely parent-children relations using what they call the “Max-Min Parents and Children Algorithm (MMPC)”. The result is a list of possible parents for each node in the network. The first step reduces the search space of the second one, which computes directions in the network through what they call the “Max-Min Hill Climbing Algorithm (MMHC)”. The MMHC

algorithm identifies the parents and children set of each variable and search in the space of Bayesian Network using a greedy hill-climbing approach.

Here we propose an approach in which we combine these techniques, selecting likely parent-children relations from the correlation matrix and applying the MMHC algorithm in order to orient edges. This approach allows us to learn a BN structure from continuous data limiting the introduction of noise and without the need of specifying a node ordering, which would be difficult to provide in the general case.

6.3 Bayesian Network Structure Learning

The first step in learning the Bayesian Network describing the relations between variables in our environment is to discover the relations between indicators in order to build the structure of the network. This can be translated in learning the edges connecting variables.

As previously discussed in Sect. 3.3.2, there are plenty of techniques available in the state of the art to learn the structure of a BN from collected data. However, all these techniques request the variables to be discrete or Gaussian. In our system, the values obtained through the monitoring system are continuous values that can be transformed into discrete values applying the thresholds defined for the indicators. So, a possible approach could consist in transforming the continuous values into discrete ones and then into applying one of the available techniques. Some tests have been run in this direction. Results were disappointing. In fact, introducing thresholds reduces indicators to a value between 1 and 5 (see Sec. 5.4.1 and Fig. 5.3). This reduces the amount of information available to compute relations, generating networks that seemed to have no rational explanation. The noise introduced by thresholds application in learning BN is also discussed in [75].

Due to this reason, we decided to use correlation to discover relations between variables. The correlation value between two variables expresses the statistical relationship between them. This is usually computed using the Pearson product-moment correlation coefficient, by dividing the covariance for the product of the standard deviations:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{E[(X - E[X])(Y - E[Y])]}{\sigma_X \cdot \sigma_Y} \tag{6.1}$$

The covariance is a value expressing how two variables change together and is computed using the expected value function.

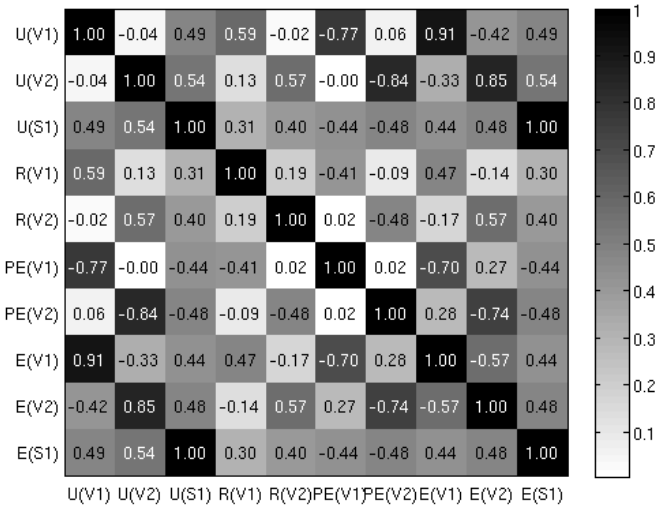
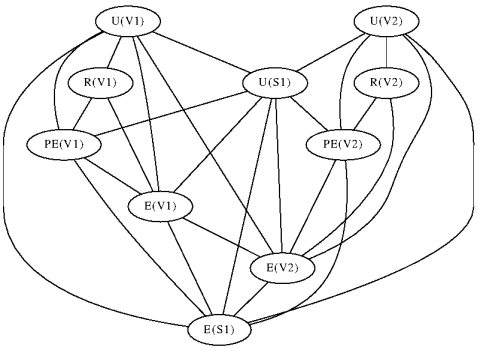


Figure 6.2: *Correlation Matrix for the **C1** configuration*

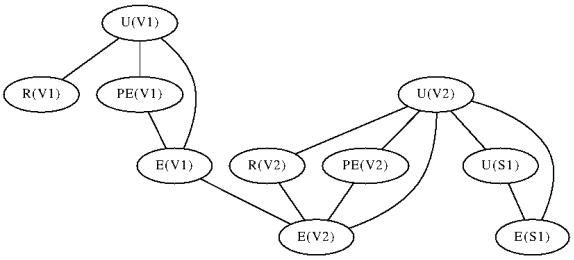
Values of the correlation can be both positive and negative, and are included in the range $[-1, 1]$. A positive value indicates an increasing linear relationship between X and Y, while a negative value indicates an anti-correlation, or a decreasing linear relationship. The nearer is the correlation value to 1 (or -1), the stronger is the relation. To build the network, only strong relations should be considered. To select the stronger relations in the correlation matrix, a threshold needs to be decided. Different thresholds enable a different number of relations between variables. A higher threshold is the cause of the loss of some relations, but a lower threshold could introduce weak relations that are not significant.

The approach has been tested with the two system configurations **C1** and **C2**. In **C1** we have two Virtual Machines (VMs), $V1$ and $V2$ running on a single server $S1$. In **C2** we have three VMs, $V1$ and $V3$ deployed on server $S1$ and $V2$ deployed on server $S2$. All the data used to evaluate the tests conducted in this chapter have been obtained using the model describe later in Ch. 8.

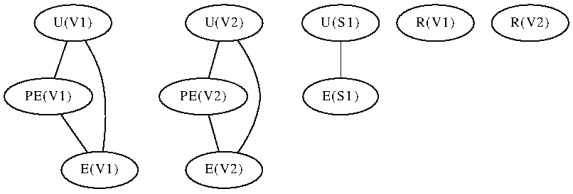
Test - Test on configuration C1 The correlation matrix has been extracted from the indicators values collected in the monitoring phase. A graphical representation is shown in Fig. 6.2[a], where dark gray indicates a stronger relation. Absolute values are represented in the matrix, since we are interested both in positive and negative relations. In Fig. 6.3 the results obtained using three different thresholds are shown.



(a) Threshold = 0.4



(b) Threshold = 0.5



(c) Threshold = 0.6

Figure 6.3: *Undirected graphs obtained with different thresholds on the correlation matrix values for the **C1** configuration*

[a] U(x) = CPU usage of x, R(x) = response time of x, PE(x) = performance per energy of x, and E(x) = energy consumption of x

Test - Test on configuration C2 The same tests have been executed with the **C2** configuration. A graphical representation of the correlation matrix is shown in Fig. 6.4[a]. In Fig. 6.5 the results obtained using three different thresholds are shown.

[a] U(x) = CPU usage of x, R(x) = response time of x, PE(x) = performance per energy of x, and E(x) = energy consumption of x

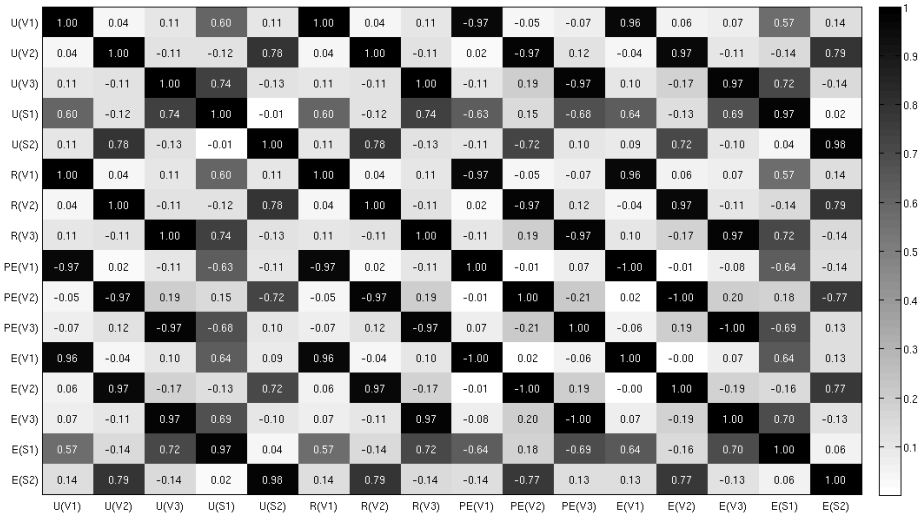


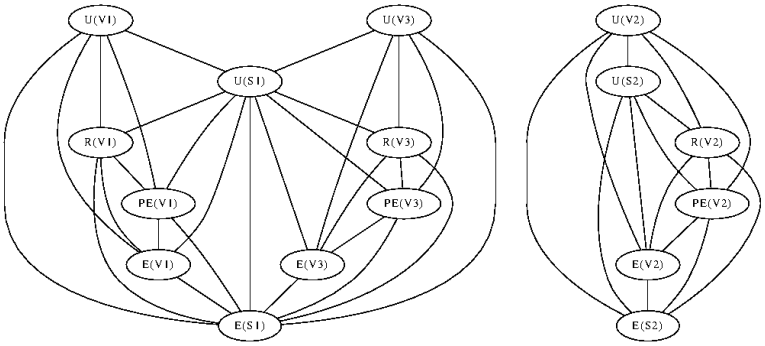
Figure 6.4: *Correlation Matrix for the **C2** configuration*

Using the correlation matrix to obtain the structure of the Bayesian Network is not straightforward. As it can be observed, correlation is a symmetric property and no information is given about the directionality of the relations between variables. On the contrary, Bayesian Networks are oriented graphs, so a direction needs to be specified for each edge in the network.

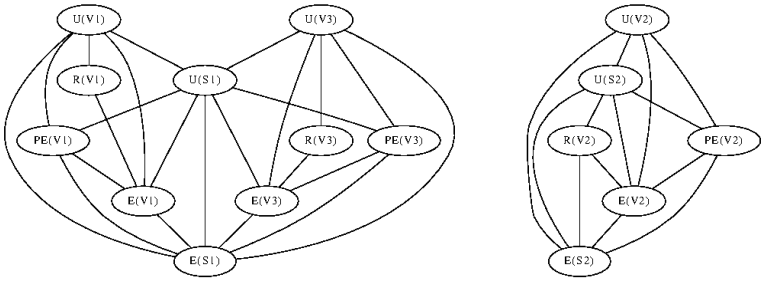
In order to find the network that describes our dataset in the best way, in the next section we apply a technique to learn directionality knowing the possible parent-children set for each node. This set can be obtained from the correlation matrix.

6.4 Bayesian Network Directionality Learning

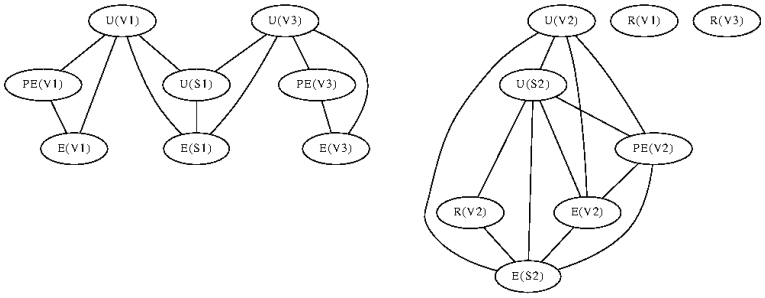
Using the correlation matrix to learn relations between variables has the main drawback of not giving any information about the direction of the relation. In Bayesian Networks, we need to know this information to be able to learn parameters and to use the conditional independence property of this representation. However, no methods are available in literature to learn the structure of a BN from continuous data. Anyway, methods are available to learn from discrete variables or Gaussian distributed variables.



(a) Threshold = 0.4



(b) Threshold = 0.5



(c) Threshold = 0.6

Figure 6.5: *Undirected graphs obtained with different thresholds on the correlation matrix values for the **C2** configuration*

In this context, we are interested in understanding the conditioning between alarm, normal and warning states of the variables. So, it is possible to apply the defined thresholds to the continuous data, in order to obtain discrete variables and to learn directionality for this variables. This approach is effective in our context only after having learned the structure of the network as discussed in Sect. 6.3. In fact, applying thresholds before this phase

introduces noise.

The approach adopted in this thesis gets inspiration from the approach described in [122]. In our approach we compute the set of possible parents and children from the correlation matrix. This set is a binary matrix obtained by putting a threshold over the correlation. Having this matrix we can apply an MMHC-like algorithm to orient the edges. The algorithm starts from an empty matrix and at each step explores all the neighbor networks, by adding, removing or reverting edges.

Definition 9. *A Directed Acyclic Graph (DAG) is considered neighbor of another DAG if it has the same structure except for one connection that can be added, removed or reverted.*

All the networks obtained are scored and the one with the highest score is chosen for the next step. Scoring functions are available in literature and they assign a score to a Bayesian Network according to their ability to describe a training set.

The MMHC algorithm is enriched using the concept of TABU list during the greedy search, following the Sparse Candidate implementation [121]. This list contains the last $\mathcal{M}$ explored structures and select the best DAG from the neighbors list that does not belong to the TABU list. Since the selected network could have a score lower than the best score detected, the algorithm stops after $\mathcal{N}$ times the score is not improved. The complete algorithm is described in Alg. 1.

In order to select the best DAG, we need to use a scoring criterion (for more information see Sect. 3.3.2 and [78]). Changing the scoring criterion can generate different results. Another variable that has to be considered is the threshold th. As discussed before, the correlation matrix gives values in the set $[-1, 1]$. Considering the absolute value of this matrix, we need to apply a threshold to limit the number of possible relations. From the application of the threshold over the correlation matrix we obtain a set of possible parent-children relations that are explored in the MMHC algorithm. Different thresholds can generate different results, limiting the number of possible connections. Examples of that have been shown in Sect. 6.3.

The direction learning algorithm has been tested for the two configurations, **C1** and **C2**, defined before. Both threshold values and scoring functions have been explored during the tests.

Test - Test on configuration C1 We implemented and tested the direction learning algorithm. Different tests have been run with different values for the threshold. Using the MMHC algorithm, a high threshold means

Algorithm 1 The Max-Min Hill Climbing Algorithm (MMHC)

procedure MMHC($\mathcal{D}$)
Input: dataset $\mathcal{D}$
Output: an oriented DAG $\mathcal{OD}$
compute correlation matrix $corr$ from $\mathcal{D}$
compute the parent-children set $\mathcal{P}$ as: $corr > th$ with th a threshold
initialize oriented graph $\mathcal{OD}$ as an empty graph
initialize counter $c = 0$ as the counter of unsuccessful steps
initialize T as the set of the last $\mathcal{M}$ already explored DAGs
while $c < \mathcal{N}$ **do**
    compute the neighbor set N of $\mathcal{OD}$ using operators add, $reverse$, and $delete$
    score all the DAGs in the set N
    select $D_N \in N$ with the highest score that is not in the T list
    add D_N to T
    if $score(D_N) <= best\_score$ **then**
        increase counter c
    end if
end while
Return the DAG with the highest score

Table 6.1: *MMHC Application Results - C1 configuration*

	th=0.25	th=0.3	th=0.35	th=0.4	th=0.45	th=0.5	th=0.55	th=0.6
Bayesian	-3975.25	-4005.19	-4130.79	-4138.44	-4381.91	-4668.29	-4807.71	-5175.97
BIC	-4818.02	-4818.02	-4855.46	-4855.47	-4939.91	-5036.56	-5169.47	-5579.35

a limitation in the parent-children set, while a low one does not necessary introduce redundancy since an edge is added only if it improves the general score. However, a low threshold increases the number of possible parent-children relations, increasing the time needed to perform the learning procedure. We compared the score values obtained using different thresholds on the correlation matrix. In the first set of tests, a Bayesian scoring function has been used to score the DAGs. The same tests have been executed using an information-theoretic scoring function, in the specific case the Bayesian Information Criterion (BIC). Results are shown in Tab. 6.1 and Figs. 6.6 and 6.7. From the data collected it seems that the lower threshold originates the best results. The values obtained with the two scoring methods can not be directly compared. In the following section the outcome of predictions performed with both methods are compared.

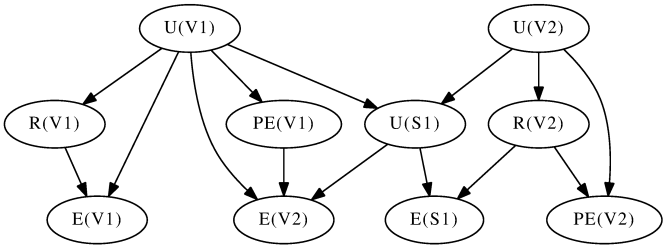


Figure 6.6: *Highest scored directed BN obtained applying the Bayesian score with the 1 server configuration*

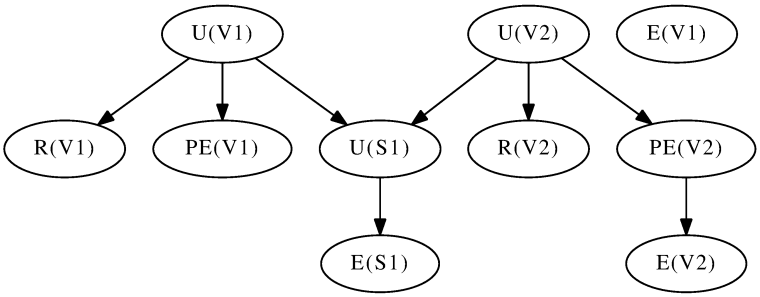


Figure 6.7: *Highest scored directed BN obtained applying the BIC score with the 1 server configuration*

Test - Test on configuration C2 Same tests have been executed for the **C2** configuration. Results are shown in Tab. 6.2. The DAGs correspondent with the best scores with the two methods are drawn in Fig. 6.8 and Fig. 6.9). As in the previous example, better results are obtained with lower thresholds. In the Bayesian score example, no differences can be detected under $th = 0.35$. In the BIC example, all the networks are the same until the threshold gets too high ($th = 0.6$).

At the end of this phase we have obtained a structure for the dependencies between data, starting from continuous values. The tests have shown that the algorithm is able to learn a structure that reflects the structure of the

Table 6.2: *MMHC Application Results - C2 configuration*

	th=0.25	th=0.3	th=0.35	th=0.4	th=0.45	th=0.5	th=0.55	th=0.6
Bayesian	-12976.3	-12976.3	-12976.3	-12994.9	-13025.4	-13025.4	-13028.3	-13455.1
BIC	-13853.7	-13853.7	-13853.7	-13853.7	-13853.7	-13853.7	-13853.7	-14177.5

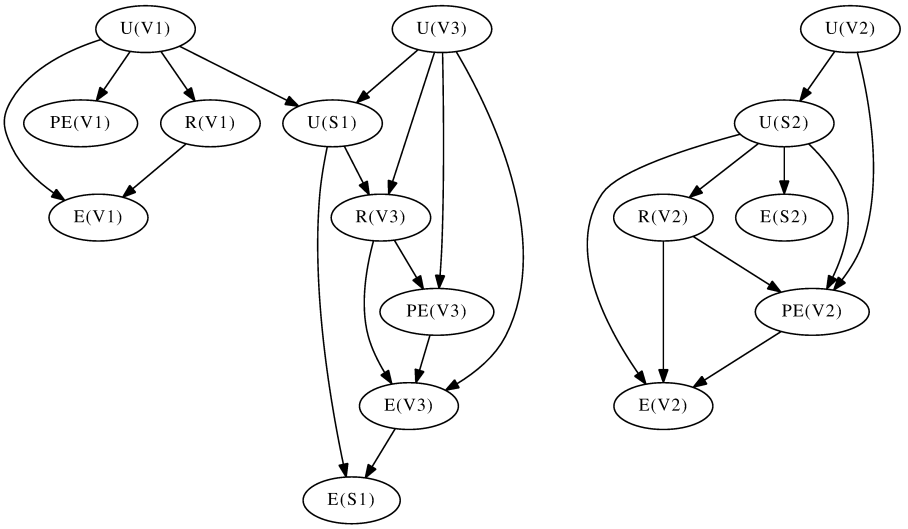


Figure 6.8: *Highest scored directed BN obtained applying the Bayesian score with the 2 servers configuration*

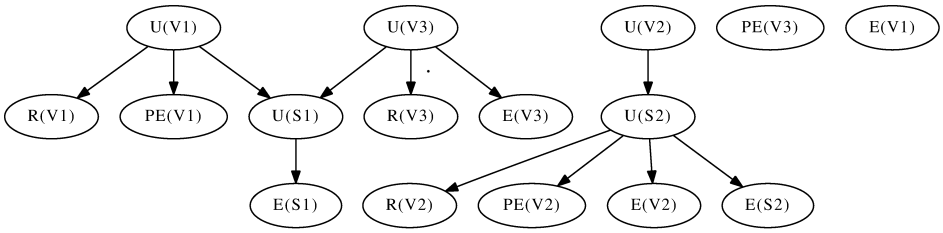


Figure 6.9: *Highest scored directed BN obtained applying the BIC score with the 2 servers configuration*

data center. For both **C1** and **C2**, the results show the ability to isolate metrics belonging to different servers while associating metrics belonging to the same servers and to the same Virtual Machines. These results make us confident about the ability of this technique to learn meaningful relations.

Now that we have extracted a directed graph for our system variables, we have to complete the model adding probability distributions to each node.

6.5 Bayesian Network Parameters Learning

Once that the structure of the Bayesian Network has been learned, the model has to be completed learning parameters. Parameters of a Bayesian Network are probability distributions associated to each node, stating the

probability for a node to assume a given value conditioned to the values of its parents. The parameters learning problem has been widely studied in literature. Here, we use the Maximum a Priori Estimation (MAP), that learns parameters given a training set of data. In this context, we are interested in being able to estimate the state of our indicators. For this reason, the training set is obtained applying the thresholds to the indicators effective values.

The complete network can be used to predict the state of the indicator, when it is unknown. In fact, the network is able to suggest the most likely values given a context (the state of the other indicators in our system). This property can be used in what-if analysis.

To test the ability of the network to predict the state of an indicator, we have run some tests. In the tests we have used a test set different from the training set used to acquire the network. We have hidden the value of a variable in all the examples and for each example we used the network to predict this value. Comparing the result with the real value we obtained the success rate of the network prediction ability. We have run the same procedure for all the variables in our system and computed an average success rate.

Prediction has been tested for the **C1** and the **C2** configurations.

Test - Prediction in the C1 configuration The prediction algorithm has been tested in the **C1** configuration. Different tests have been run using Bayesian Networks obtained with different thresholds on the correlation matrix and with both scoring techniques. Results are shown in Tab. 6.3 where average success rates for all the configurations are compared. As before, a lower threshold with the Bayesian scoring function performs better. The best configuration for both scoring methods have been compared in Fig. 6.10, where the success rate of the prediction is shown for each variable. In our context, the Bayesian scoring function generates a better result if we compare the prediction rates obtained with both methods.

Table 6.3: *Value Estimation Success Rate in C1 Configuration*

	th = 0.3	**th = 0.4**	**th = 0.5**	**th = 0.6**
Bayes	89.93	88.76	84.26	83.63
BIC	86.97	86.13	84.30	83.63

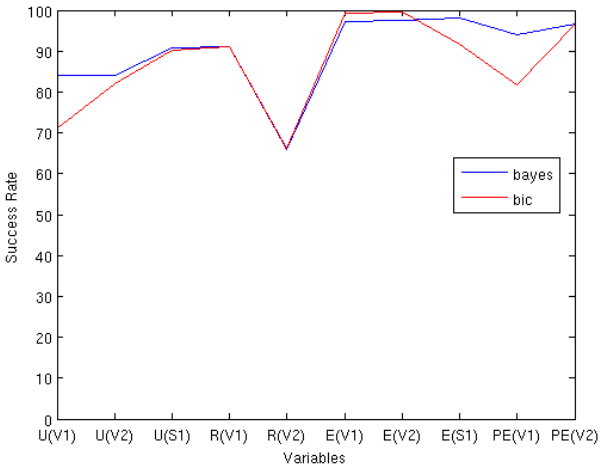


Figure 6.10: *Prediction success rate comparison for the configuration **C1***

Test - Prediction in the C2 configuration The same set of tests has been executed for the **C2** configuration. Results are shown in Tab. 6.4 where average success rates for all the configurations are compared. As before, a lower threshold with the Bayesian scoring function performs better. The best configuration for both scoring methods have been compared in Fig. 6.11, where the success rate of the prediction is shown for each variable. As before, we obtain a better result with the Bayesian scoring function using a low threshold.

Table 6.4: *Value Estimation Success Rate in C2 Configuration*

	th = 0.3	**th = 0.4**	**th = 0.5**	**th = 0.6**
Bayes	87.76	87.27	87.27	85.55
BIC	87.21	87.21	87.21	85.49

6.6 Implementation Details and Considerations

The learning module has been implemented in MATLAB [123]. For the BN scoring functions, for parameters learning, and for inference on the network, we used the Bayes Net Toolbox (BNET) [124], an open source toolbox for MATLAB that provides instruments for learning and using

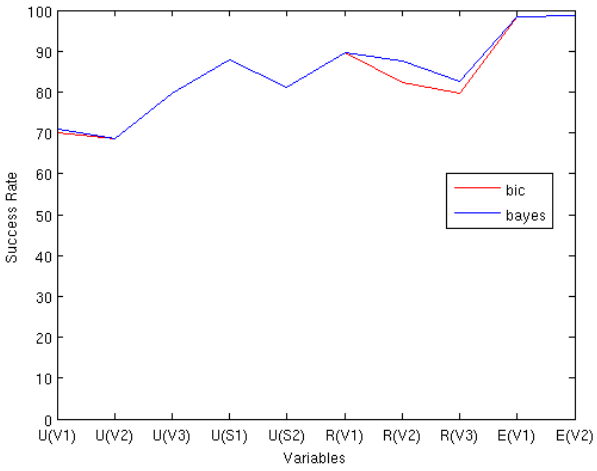


Figure 6.11: *Prediction success rate comparison for the configuration **C2***

Bayesian Networks. The code of the structure learning module is available at [125].

The main concern about this approach is related to scalability. The structure learning algorithm requires a computation time that can increase exponentially with the number of variables considered in the network. In order to analyze this issue, several tests have been run changing two parameters: the number of variables and the size of the dataset. From the tests, we observed that the size of the dataset does not significantly affect the computation time. On the contrary, the number of variables influences the computation time exponentially. Results are represented in Fig. 6.12. Even if in the tests run the computation time stays in a manageable range, this can become an issue for larger variable sets. In spite of these considerations, it has to be highlighted that the computation of the complete network needs to be performed only once, during the set up operations. Further modifications of the system (e.g. VM migrations and reconfigurations) affect only parts of the network and we can make reference to the conditional independence property of BNs to reduce the number of involved variables and to divide the computation to smaller subsets. Also the computation of the very first network could be simplified by observing recurrent relations among similar variables. This can be obtained by defining an ontology for classifying variables. This topic is discussed in more details in Sect. 9.4.

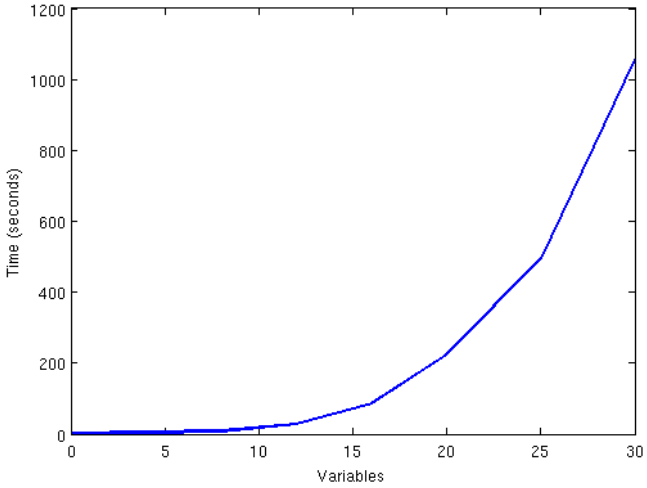


Figure 6.12: *Computation time of the learning modules with an increasing number of variables*

6.7 Conclusion

In this chapter we have investigated relations between indicators and how knowing these relations enables a set of techniques that can be used to improve Energy Efficiency in the data center.

We built a Bayesian Network representing the relations between the indicators state in the system. In order to do that, we started from the data collected by the monitoring system. Using correlation matrix to reduce the possible parent-children relations, we were able to apply well-known techniques for learning the structure and the parameters of the network.

Results have shown the reliability of the method, that was able to identify variables related to the same Virtual Machine and to the same server and to properly connect them. Also the what-if analysis tests we have executed have been successful. The success rate when estimating the state of an indicator using the network is between 87% and 89% in both the configurations we have investigated.

CHAPTER 7

Context Based Adaptive Action Selection: Exploration and Exploitation

Indice

In a dynamic Service Oriented Architecture (SOA), inefficiencies can seldom occur. Once a violation of the metrics used to monitor the system state is detected, a repair action has to be selected in order to fix the problem. In this chapter we propose the Adaptive Action Selection (AAS) algorithm, derived from the Adaptive Operator Selection (AOS) algorithm with attention to the context, which can have a significant impact over the effect of

the action. The algorithm is used both for selecting an effective adaptation strategy given a context, and for automatically learning the effect of an action over the monitored set of indicators. This information can be used to automatically build the connections among actions and indicators contained in the goal-oriented model proposed in this thesis.

7.1 Introduction

In a Service Oriented Architecture (SOA), inefficiencies and suboptimal states are very common due to the dynamism of the environment. In fact, modification in the load, in the amount of deployed applications, and other similar events, can significantly modify the performance of the system.

Using a monitoring system allows verifying the system state, and the employment of metrics and indicators enables the detection of criticalities. The metrics used for this intent have been discussed in Ch. 4. In the model introduced in Ch. 5, a set of repair actions have been defined. These actions can be used to react to suboptimal situations, with the intent of improving the state of violated indicators. As it has been discussed, actions can have a different outcome on different indicators. In fact, the same action could positively affect the state of an indicator while negatively affecting another one. A complete knowledge of the influence of the considered actions over the monitored indicators is important to avoid unexpected side effects.

A detailed mapping of action-indicator relations can be a difficult task even for a domain expert, especially in a dynamic environment where the effect of an action can be influenced by the current context. In order to overcome this issue, we propose the Adaptive Action Selection (AAS) approach as a methodology for automatically learning the action-indicator relations from experience and for suggesting the most likely successful repair action given a context. The approach associates to each action a probability of success in the indicator improvement, and keeps this probability up to date in time. In this way, the proposed model is able to automatically adapt to modifications in the dynamic environment. The algorithm implements the principal features of the Adaptive Operator Selection (AOS) algorithm introduced in Sect. 3.4, by allowing for both exploration and exploitation. In fact, the algorithm explores several possibilities by giving more importance to the ones that have a higher confidence of being effective.

A general view of the approach is shown in Fig. 7.1. Given a context and a set of repair actions, the AAS algorithm suggests the most likely successful action and observes the outcome of this choice. The evaluation

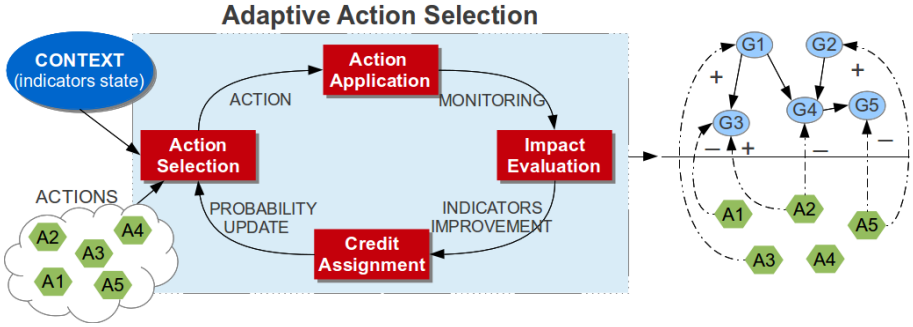


Figure 7.1: *Adaptive Action Selection approach for adaptive Information Systems*

consists in comparing the value of the indicators before and after the action execution. A credit is kept for each action considering also past executions and this value is used to keep a probability estimation of the action effectiveness. After some executions it is possible to derive information about the action-indicator relations contained in the goal-oriented model shown in Fig. 5.2, associated with a confidence value.

After a brief discussion about the techniques from which the AAS algorithm has been derived (Sect. 7.2), we describe the algorithm in details focusing on its dependency on the context in Sect. 7.3. Then, in Sect. 7.4 the technique used for learning the action-indicator relations is discussed with some experiments to show the effectiveness of the approach. Finally, in Sect. 7.5 the algorithm is evaluated in its ability of suggesting an effective repair action.

7.2 State of the Art

An adaptive system has to be able to automatically change together with the environment in which it operates. In this section, we discuss the topic of selecting the most suitable adaptation action for fixing an undesired state of the monitored system. This problem has been widely discussed in the literature and some general solutions have already been discussed in Sect. 3.4. The AOS mechanism has been discussed in both its phases: credit assignment and operator selection. For the latter, the mostly used algorithm has been presented: Probability Matching. Although simple and effective, the Probability Matching approach suffers from some drawbacks. These are mostly related to some fixed parameters in the model, that are not able to represent properly a dynamically evolving system. The first parameter is the value of p_{min}, used to ensure that each operator can be selected even if

acting very badly. Instead of being a fixed value, the value of p_{min} should be dependent on the affordability of the current best solution: the more affordable is the current solution, the less we want to explore alternatives. The other drawback is introduced by the parameter α. It was used to weight the importance of the historical values in the quality estimation of an operator. Even in this case, this value should change with the frequency of application of the operator: if the operator has not been applied for a long time, its recent value should be more important in order to make its quality estimation up-to-date quickly. On the contrary, if frequently applied, the well consolidated estimation derived from past applications should be considered.

The algorithm proposed in [80], tries to overcome these drawbacks by proposing an alternative approach based on the Multi Armed Bandit Selection (MABS) mechanism. The MABS approach has been used in different contexts. In [126], authors use it to select between different algorithm portfolios to solve decision problems, while in [127] the MABS approach is used to automatically tune the parameters of parallel computing programs, both in a static and in a dynamic scenario.

7.3 Adaptive Action Selection Algorithm

The system that we aim to model, is a self-adaptive system able to suggest an efficient repair strategy. The approach proposed is derived from the AOS approach. Unlike the classical AOS approach, in our system the probability of applying an action is strictly dependent on the context. Since in this case we are not going to suggest evolutionary operators, but repair actions, we rename the approach Adaptive Action Selection.

The context $\mathbb{C}$ can be defined as follows:

Definition 10. *The* CONTEXT $\mathbb{C}$ *is the state of the indicators in a given moment in which the system is evaluated. This state can be of three kinds: satisfied, violated or almost violated, corresponding to the indicators zones satisfaction, violation and warnings. Since there are two violations and two warning zone, the indicator state can take one of five values.*

The concept of context is very important since different adaptation strategies should be applied under different circumstances.

The introduction of the concept of context brings some modification to the AOS approach described in Sect. 3.4, and to the main components of the approach: impact, credit, quality, and probability.

7.3.1 Impact computation

After the application of an action, its impact $\mathcal{I}$ over the state of the system has to be evaluated. The system is observed after a period of time that depends from the action (see Tab. 5.2) and the new state is compared with the one before the application of the action. Impact can be defined as follows:

Definition 11. *The* IMPACT $\mathcal{I}$ *is the effect that the execution of an action has over the indicators defined for monitoring the system.*

While in the original approach a single impact value was defined for each action, the introduction of the context requires a redefinition of the impact. The impact is dependent from the context itself. First of all, an action has a different impact over different indicators, so a different value for each of them has to be used. Moreover, $\mathcal{I}$ also depends on the state that the indicator has when the action is applied. In fact, the same action can have different outcomes in different initial conditions. To make an example, let's consider action A_3 (VM Reconfiguration - remove) in Tab. 5.2. Decreasing the amount of resources allocated to a Virtual Machine (VM) has a positive impact over Central Processing Unit (CPU) utilization if resources were oversized, negative if they were already fully utilized. That means that impact can now be represented as a three dimensional matrix, of dimensions $L * N * S$ where L is the number of available actions, N is the number of indicators and S is their support (the set of values the indicator can get):

	I_1	I_2	$\cdots$	I_N
A_1	$\mathcal{I}(A_1 \mid state(I_1))$	$\mathcal{I}(A_1 \mid state(I_2))$	$\cdots$	$\mathcal{I}(A_1 \mid state(I_N))$
A_2	$\mathcal{I}(A_2 \mid state(I_1))$	$\mathcal{I}(A_2 \mid state(I_2))$	$\ldots$	$\mathcal{I}(A_2 \mid state(I_N))$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_L	$\mathcal{I}(A_L \mid state(I_1))$	$\mathcal{I}(A_L \mid state(I_2))$	$\cdots$	$\mathcal{I}(A_L \mid state(I_N))$

where the function $state(I_n)$ maps the value of the indicator I_n into its state.

In our system, the state of the indicators can get five values that correspond to the two alarm and warning zones and to the normal zone (see also Fig. 5.3). More in detail:

- if $I_n < ThresholdMin \Rightarrow state(I_n) = 1$
- else if $I_n < ThresholdMin + size(Warning(I_n)) \Rightarrow state(I_n) = 2$
- else if $I_n > ThresholdMax \Rightarrow state(I_n) = 5$

- else if $I_n > ThresholdMax - size(Warning(I_n)) \Rightarrow state(I_n) = 4$
- else $state(I_n) = 3$

where $size(Warning(I_n))$ gives the dimension of the warning interval for the indicator I_n. More intuitively, observing Fig. 5.3, each intervals get values from 1 to 5 left to right. So, each element $\mathcal{I}(A_l|state(I_n))$ represent a vector of five values, one for each of the possible states for indicator I_n.

The impact $\mathcal{I}$ of an action A_l over the indicator I_n is assigned considering the variation of the indicator after the application of the action. We modeled $\mathcal{I}$ as assuming discrete values between -1 and 1. An impact equals to zero means that the action has no effect over the indicators, an impact equals to 1 means a positive effect and an impact -1 means a negative effect. To assign the impact we compare the state of the indicator before and after the application of the action. If the indicator is in the same state, a value of 0 is assigned. If the state changes positively, a score of 0.5 is assigned for each improvement. In more details:

- from alarm to warning or from warning to normal: impact = 0.5
- from alarm to normal: impact = 1
- from normal to warning or from warning to alarm: impact = -0.5
- from normal to alarm: impact = -1

This is also described in Fig. 7.2.

The described approach has a main drawback related to scalability. As stated before, in the original approach we have a single impact value for each action, while here impact is a three dimensional matrix. Anytime an action is selected, only a small subset of the impact values for that action is updated, depending on the initial state of the indicators. To make it clearer, let's consider a simple example in which five indicators are defined with a support of three, and four actions are possible. At a given moment, the state

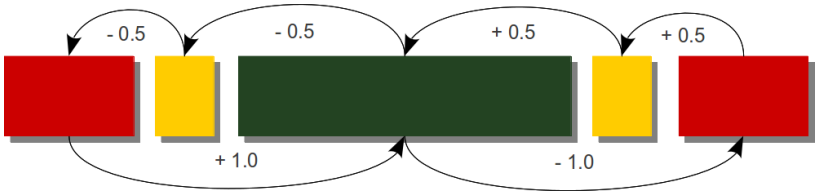


Figure 7.2: *Determining the impact of an action over the state of an indicator*

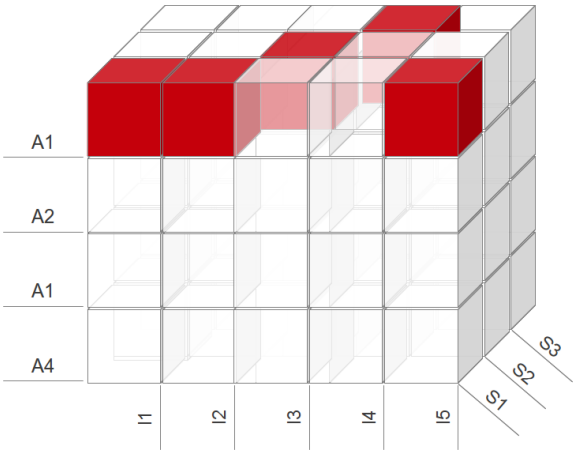


Figure 7.3: *Probability matrix update graphical representation. In red are highlighted the only values that are updated when action 1 is executed and with an original indicators state such as: I1 = 1, I2 = 1, I3 = 2, I4 = 3, I5 = 1.*

of I1, I2, and I5 is 1 (the indicators are violated), while the state of I3 is 2 (warning), and state of I4 is 3 (satisfied). If A1 is executed and evaluated, only the cells for that specific state of the indicators are updated, as shown in Fig. 7.3. From NxS values (15 in our example), only N are modified (5 in our example). A lot of trials in different conditions have to be performed to have a complete knowledge about the effect of an action, since it should be tested several time for each state of the several indicators to have a complete knowledge. Of course, this number increases together with the number of indicators and the number of states they can get. With 16 indicators and a support of 5, only 16 values are updated at each trial from a total of 80. Given the definition of action introduced in Sec. 5.5, also the parameters of the action have to be considered in this model. In fact, different parameters can bring to different results.

In order to increase scalability and to reduce the trials needed to reach a complete knowledge of the impact matrix, we consider a different point of view. Indicators are measured with continuous or discrete values and an action can affect an indicator by increasing or decreasing this value. So, instead of representing impact for each of the states of an indicator, it is possible to describe the action impact over an indicator with a single value, indicating if the action tends to increase or decrease the value of the indicator. The simplified impact matrix is a two dimensional LxN matrix, where the value $\mathcal{I}(A_l, I_n)$ represents the ability of the action A_l of increasing the value of indicator I_n. That means that impact can assume positive or nega-

tive values: positive if it increases the indicator and negative otherwise:

	I_1	I_2	$\cdots$	I_N
A_1	$\mathcal{I}(A_1, I_1)$	$\mathcal{I}(A_1, I_2)$	$\cdots$	$\mathcal{I}(A_1, I_N)$
A_2	$\mathcal{I}(A_2, I_1)$	$\mathcal{I}(A_2, I_2)$	$\ldots$	$\mathcal{I}(A_2, I_N)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
A_L	$\mathcal{I}(A_L, I_1)$	$\mathcal{I}(A_L, I_2)$	$\cdots$	$\mathcal{I}(A_L, I_N)$

The way in which the impact value is computed changes from the previous approach. We decided to consider two steps in the impact value as before. If the value of the indicator increases significantly than the assigned impact is the maximum. Otherwise, if the increase is limited but higher than a minimum value, than the impact value assigned to the action is 0.5. In order to assess which improvement is significant or not, we used as a comparison the size of the warning interval. Impact of action A_l over indicator I_n is computed as follows:

- if $(I_{n,t+1} - I_{n,t}) > size(Warning(I_n)) \Rightarrow \mathcal{I}(A_l, I_n) = 1.0$;
- if $(I_{n,t} - I_{n,t+1}) > size(Warning(I_n)) \Rightarrow \mathcal{I}(A_l, I_n) = -1.0$;
- if $size(Warning(I_n))/3 < (I_{n,t+1} - I_{n,t}) < size(Warning(I_n)) \Rightarrow \mathcal{I}(A_l, I_n) = 0.5$;
- if $size(Warning(I_n))/3 < (I_{n,t} - I_{n,t+1}) < size(Warning(I_n)) \Rightarrow \mathcal{I}(A_l, I_n) = -0.5$;

The value $size(Warning(I_n))$ has been chosen because it is representative of the impact of the action. When the increase in the value of the indicator is bigger than the warning region there is a possibility for the indicator to move from a violation to the satisfaction zone. For this reason, this interval can be taken as a reference for evaluating the effect of the actions.

Of course, even with this approach parameters of the action have to be considered when evaluating the effect of an action. However, the number of parameters is sensibly reduced in comparison with the former approach. For clarity, in the rest of the section we are not considering the parameters of the actions in the approach definition, even if they will be considered in the evaluation of the approach.

7.3.2 Credit assignment

From the impact value it is possible to assign a credit $\mathcal{C}$ to the action, in relation to the indicators. Since the system is not isolated, it is possible that the

evaluated state of the system has been influenced by other external factor. That means that some noise can be introduced in the impact evaluation. For this reason, a window $\mathcal{W}$ of impact evaluations is maintained, and a credit is assigned using an aggregation of these values. The size of the window $\mathcal{W}$ has to be chosen depending on the features of the system. If the system can be considered as stable, a large value for the window size is recommended. Otherwise, small values for $\mathcal{W}$ make the system more dynamically adaptable, enabling it to change strategy more frequently. According to these premises, the credit can be computed as:

$$\mathcal{C}_t(A_l, I_n) = \frac{\sum_{s \in \mathcal{W}} \mathcal{I}_s(A_l, I_n)}{size(\mathcal{W}_t)} \tag{7.1}$$

The credit value is used to update a quality estimation for the action.

7.3.3 Quality estimation

Quality is the estimation of the ability of the action to increase the value of an indicator. In our system, it is represented through a matrix $\mathcal{Q}$ of dimensions LxN where L is the number of available actions and N is the number of indicators. The quality matrix is updated at each iteration. While impact is computed instantly at each application, quality depends also from past values. In general, quality $\mathcal{Q}$ can be expressed as:

$$\mathcal{Q}_t = f(\mathcal{Q}_{t-1}, \mathcal{C}_t) \tag{7.2}$$

According to Eq. 7.2, quality depends on two factors:

- *Credit*: this value depends exclusively on the last application of the action;
- *History*: this value summarizes the outcome of old applications of the action.

Different methods for estimating quality by combining these two components are used in different algorithms. Here, we decided to use the Probability Matching approach introduced in Sect. 3.4.2, with slight modifications to fit our problem and to include the context.

In Probability Matching [88], quality estimation for an action can be performed as follows:

$$\mathcal{Q}_t(A_l, I_n) = (1 - \alpha)\mathcal{Q}_{t-1}(A_l, I_n) + \alpha \cdot \mathcal{C}_t(A_l, I_n) \tag{7.3}$$

where α is an adaptation rate, indicating the importance of the memory of past values. As discussed in Sect. 7.2, the definition of the parameter α can be problematic and a fixed value is not the best solution. The importance of history in the quality computation should be directly dependent on the affordability of the historical value. If the action has not been applied for a long time, its historical value has a limited importance, since it probably does not reflect anymore the current situation. In this case, more importance should be given to the credit computed from the current application of the action. On the contrary, if the action has been recently applied, the historical value reflects the current situation and it should have a higher importance. According to this, the importance of history should depend on the elapsed time since the last application of the action: if the last application is recent, history should count more. For this reason, we picked a value for α which is dependent on the freshness and affordability of the historical information about quality:

$$\alpha(A_l) = 1 - \frac{Last_{op}(A_l)}{T_{op}} \tag{7.4}$$

where $Last_{op}(A_l)$ is the timestamp of the last application of action A_l, while T_{op} is the current timestamp. Let's consider the behavior of α under different situations. If action A_l has never been applied till now, then $Last_{op}(A_l) = 0$. As a consequence, $\alpha(A_l) = 1 - \frac{0}{T_{op}} = 1$ and $1 - \alpha = 0$. So, when the action A_l is applied for the first time, history has no value and all the importance is given to the credit. On the contrary, if the action has been applied in the last execution, than $Last_{op}(A_l) = T_{op} - 1$. In this case the value of α is $\alpha(A_l) = 1 - \frac{T_{op}-1}{T_{op}} = \frac{1}{T_{op}}$ and $1 - \alpha(A_l) = 1 - \frac{1}{T_{op}}$, giving to history a greater importance.

7.3.4 Probability Computation

In this section we describe the process for computing the probability of selecting an action given a context. As for impact, in the original approach a single probability value was used for each action. Here, the goal is to be able to select the best strategy for solving indicators violation. For this reason, only violated indicators are considered in probability computation. According to this, probability consists in a vector in which each value expresses the probability of success of a given action given the current context $\mathbb{C}$.

The first step for computing probability given a context consists in eval-

uating the quality matrix according to $\mathbb{C}$. In fact, as discussed in Sect. 7.3.2 and Sect. 7.3.3, credit $\mathcal{C}$ and quality $\mathcal{Q}$ express the ability of the action to increase the indicator value. As a consequence, quality can be both positive or negative: positive if the action increases the indicator value, negative otherwise. This information has to be contextualized since some indicators can be under their desired values but other can be over it. So, in this case, we should prefer actions able to reduce the indicator value. The contextualized quality matrix $\mathcal{Q}'$ can be computed as follows:

$$\mathcal{Q}'_t(A_l, I_n) = \begin{cases} -\mathcal{Q}_t(A_l, I_n) & IF\ I_n > I_n^{max} \\ \mathcal{Q}_t(A_l, I_n) & otherwise \end{cases} \tag{7.5}$$

The value $\mathcal{Q}'_t$ is dependent from the state of the indicator. If the current value of the indicator is lower than or equal to the desired value, it coincides with $\mathcal{Q}_t$, otherwise the opposite is considered. In the equation, the value I_n^{max} represents the maximum value that the indicator can take before entering in the warning zone.

Probability $p(A_l|\mathbb{C})$ depends directly from the contextualized quality estimation matrix and can be computed as:

$$p'(A_l|\mathbb{C}) = max\left\langle 0, \frac{1}{c_{A_l}}\left(\frac{\sum_n w_n \mathcal{Q}'_t(A_l, I_n)}{\sum_n w_n} - 0.5\frac{\sum_n w'_n |\mathcal{Q}'_t(A_l, I_n)|}{\sum_n w'_n}\right)\right\rangle \tag{7.6}$$

The first part of the equation expresses the ability of the action of improving the violated indicators. The term w_n is a weight expressing the indicator violation priority, used to give different importance to the indicators in the alarm and in the warning zone. In detail, we use $w_n = 1$ if the violated indicator is in the alarm zone, $w_n = 0.5$ if it is in the warning zone, and $w_n = 0$ if it is in the normal zone. The probability to apply the action depends from the probability of the action to improve the indicators that are violated or that are near to be violated. The second part represents a penalty term, which decreases the probability value if the action has effects on other indicators that are not in the list of the violated metrics. The term w'_n is equal to 1 for each metric which is not in the violation list (for which $w_n = 0$). The penalty factor has less importance than the other one since even if the action has an effect on other metrics, this does not means that this effect can have a negative outcome. In fact, the metrics could stay inside the constraints also after the application. In the penalty computation,

the absolute value of the quality matrix is considered, since we don't want neither increasing or decreasing the values of the satisfied indicators. Finally, the cost of the action enactment is considered. This cost is expressed through the variable c_{A_l}, which is dependent both on the execution time of the action (t_{A_l}) and on its energy cost (En_{A_l}), and can be expressed as:

$$c_{A_l} = f(t_{A_l}, En_{A_l}) \tag{7.7}$$

In the Probability Matching approach, a minimum probability is assigned to each action in order to avoid that an action reaches a probability of being selected equal to 0 and is not tested anymore. This approach is useful since the system is dynamic and the utility of an action can change over time. Keeping a probability greater than 0 for each action allows the system to select the action, even if rarely, and to learn the new behavior if the system changes. According to this, probability can be defined again as follows:

$$p(A_l|\mathbb{C}) = p_{min} + (1 - L * p_{min}) \frac{p'(A_l|\mathbb{C})}{\sum_{m \in L} p'(A_m|\mathbb{C})} \tag{7.8}$$

As discussed in Sect. 7.2, the fixed value for p_{min} is one of the drawbacks of the classic algorithm. In this context it is preferable to have a dynamic value for p_{min} which depends on the affordability of the best action available in the considered context. We also want this value to be proportional to the total number of available actions to keep the amount of probability distributed to all the action under control. We propose to compute the value for p_{min} as:

$$p_{min} = \frac{1 - max(p'(A|\mathbb{C}))}{L * 10} \tag{7.9}$$

Once probabilities for all the actions are computed given the context, an action can be selected using a wheel-like process. This mechanism behaves similarly to a roulette wheel where a portion of a wheel is assigned to each possible candidate, with a different dimension proportional to its probability. The candidate is selected by randomly generating a number inside the interval composing the wheel. Even if a candidate with a low probability is less likely to be selected, there is still a chance that it may be.

Probability Matching algorithm for AAS

Every time a violation occurs, an action is suggested from the pool of available actions. The selected action depends both from the context and from

the credits previously assigned to the actions (see Sect. 7.3). In order to allow the algorithm to quickly adapt in the initial phases, where the effect of the action has still to be learned, it is important to give the priority to actions never tested before, ensuring that each action is at least tried once. Since not all the actions can be applied in every moment (due to the preconditions introduced in Sec. 5.5), this condition has to be verified at each step. The algorithm for action selection, applied every time a violation of one or more indicators occurs, behaves as follows:

1. the context $\mathbb{C}$ is evaluated, considering the state of the indicators: alarm state ($state(I_n) \in \{1, 5\}$), warning state($state(I_n) \in \{2, 4\}$), and normal state ($state(I_n) = 3$);
2. preconditions are verified for each action $A_i \in A$ and only the applicable actions are selected (A');
3. if there are actions in A' that have never been executed, the other actions are discarded and the new set A' is composed of all the actions that have never been tried before;
4. the contextualized quality matrix $\mathcal{Q}'_t$ is computed as in Eq. 7.5 for each action in A';
5. probabilities of applying each action given the context at time t is updated as in Eq. 7.7;
6. the global probability value for each action is computed as in Eq. 7.8, or equally distributed if $p'(A_l|\mathbb{C}) = 0 \; \forall \; A_l \in A'$;
7. an action A_l^* is selected via a roulette-wheel selection scheme;
8. the action A_l^* is applied, and its impact and credit is computed after the execution time is expired;
9. an updated quality value $\mathcal{Q}(A_l^*, I_n)$ is estimated as in Eq. 7.3.

The complete Probability Matching algorithm for AAS is described in Alg. 2. An implementation of the algorithm, used to test its behavior, is available at [128].

7.4 Learning Action-Indicator Relations: Exploration

The results obtained using the algorithm for AAS, shown in Alg. 2, improve in time. In fact, the confidence of the result is lower in the initial steps when

Algorithm 2 The Probability Matching algorithm for Adaptive Action Selection

procedure AAS(I, $\mathcal{Q}$, A)
Input: the values of the indicators I, the current quality matrix $\mathcal{Q}$, the action list A
compute context $\mathbb{C}$ from I
compute w_n for each indicator in I according to its context in $\mathbb{C}$
verify preconditions for each action A_i in A
create the set of actions A' which satisfy the preconditions
verify if there are actions in A' never tested and eventually remove all tested actions from A'
compute the contextualized quality matrix: $\mathcal{Q}'(A_l, I_n) = \begin{cases} -\mathcal{Q}(A_l, I_n) & IF\ I_n > I_n^{max} \\ \mathcal{Q}(A_l, I_n) & otherwise \end{cases}$ for each action in A' and each indicator in I
compute $p'(A_l|\mathbb{C}) = max\left\langle 0, \frac{1}{c_{A_l}} \left(\frac{\sum_n w_n \mathcal{Q}'_t(A_l,I_n)}{\sum_n w_n} - 0.5 \frac{\sum_n w'_n |\mathcal{Q}'_t(A_l,I_n)|}{\sum_n w'_n} \right) \right\rangle$ for each action in A'
if $\sum_l p'(A_l|\mathbb{C}) = 0$ **then**
  $p(A_l|\mathbb{C}) = 1/L$
else
  compute $p_{min} = \frac{1 - max(p'(A|\mathbb{C}))}{L * 10}$
  compute $p(A_l|\mathbb{C}) = p_{min} + (1 - L * p_{min}) \frac{p'(A_l|\mathbb{C})}{\sum_{m \in L} p'(A_m|\mathbb{C})}$
end if
select an action A_l^* via a roulette-wheel selection scheme;
wait for $t_{A_l^*}$
collect indicators values I' from the monitoring system
compute impact $\mathcal{I}$:
if $(I'_n - I_n) > size(Alarm(I_n))$ **then**
  $\mathcal{I}(A_l^*, I_n) = 1.0$
else if $(I_n - I'_n) > size(Alarm(I_n))$ **then**
  $\mathcal{I}(A_l^*, I_n) = -1.0$
else if $(I'_n - I_n) > size(Alarm(I_n))/3$ **then**
  $\mathcal{I}(A_l^*, I_n) = 0.5$;
else if $(I_n - I'_n) > size(Alarm(I_n))/3$ **then**
  $\mathcal{I}(A_k, I_n) = -0.5$;
end if
compute credit: $\mathcal{C}_t(A_l, I_n) = \frac{\sum_{s \in \mathcal{W}} \mathcal{I}_s(A_l, I_n)}{size(\mathcal{W}_t)}$
compute $\alpha(A_l^*) = 1 - \frac{Last_{op}(A_l^*)}{T_{op}}$
update quality: $\mathcal{Q}_t(A_l^*, I_n) = (1 - \alpha)\mathcal{Q}_{t-1}(A_l^*, I_n) + \alpha \cdot \mathcal{C}_t(A_l^*, I_n)$
$Last_{op}(A_l^*) = T_{op}$
$T_{op} = T_{op} + 1$
Return the updated quality matrix $\mathcal{Q}$

not all the possible strategies have been tested or, even if tested, they can be subject to noise that is compensated after a reasonable number of tests.

For this reason, before applying the algorithm to a real environment, some off line training should be executed if possible. In order to do that, we used a simulation system for data center modeling, described in details in Ch. 8. The model allows simulating the data obtained from the monitoring system and the effect of the actions by modifying the structure of the data center model.

In this section we show the result obtained by training the proposed methodology for some data center configurations. At the end of the training, the quality matrix obtained as a result, highlights the relations existing between repair actions and indicators, and their confidence of success. So, the final data can be used to build the missing part of our model, introduced in Ch. 5: the action-indicator relations.

Here, we are going to consider two system configurations and we are going to include parameters of actions in our evaluation. The tested configurations are the same used for testing the Bayesian Network (BN) learning algorithm described in Ch. 6. As a reminder, in configuration **C1** we have two VMs, $V1$ and $V2$ running on a single server $S1$. In configuration **C2** we have three VMs, $V1$ and $V3$ deployed on server $S1$ and $V2$ deployed on server $S2$. More details about the data center configuration parameters are shown in Tab. 8.4.

In both cases we considered the same set of indicators used in Ch. 6 for the evaluation of the system state: CPU usage, response time, Performance per Energy, and energy. CPU usage and energy are evaluated both at server and VM level. The repair actions considered in the learning process are: A_1 (VM Migration), A_2 (VM Reconfiguration - add), A_3 (VM Reconfiguration - remove), A_6 (Turn on Server), and A_7 (Turn off Server). In a second moment, complex actions are also considered, and in particular AC_1 (Migrate and Turn Off) and AC_2 (Turn On and Migrate).

All the results are shown using the quality matrices obtained at the end of the execution. In the matrices, the elements are shown with a different level of gray depending on the strength of the impact of the action over the indicator: the likelier the effect, the darker the color.

Test - Test on configuration C1 In this configuration, the amount of actions that can be applied is reduced. In fact, preconditions for the applications of some of the actions in the testing set will never be satisfied. For instance, with a single server it is not possible to perform migration. Even the turning on and off of servers is not possible since only one server is available in the data center, and until there are VMs running on

it, it is not possible to turn it off.

As a consequence, only actions A_2 and A_3 can be tested. Both actions take as parameter the ID of the VM to be reconfigured. We are ignoring the second parameter, the resource type, since the only resource that can be modified in our example is the CPU.

Tests have been run by simulating the behavior of the system under different loads and the resulting quality matrix is shown in Fig. 7.4. In the results we applied a threshold over the values, considering only values greater than 0.1. As can be observed, adding a core to the CPU of a VM significantly decreases the value of the CPU usage for that machine and also reduces response time. The energy consumption of the VM has also a slight probability of being increased. Since the two VMs considered in this configuration always run in the same server, a modification of one of them can have an impact on the other one. In fact, as can be seen in the first row, adding a core to $V1$ have a small probability of increasing response time of $V2$, while decreasing its energy consumption. This is due to the fact that under some conditions, increasing the CPU time for one machine can decrease the time available for the other one, affecting its behavior and consumption. The same considerations are valid for the second action, A_3, in a symmetric way. Removing a core from one VM has relevant effects on the usage and the response time of the VM that are likely to increase. As before, a slight probability of decreasing the VM energy consumption has been discovered and the other VM behavior can be affected.

Test - Test on configuration C1 with complex actions In this configuration, we added to the regular set of actions also the complex action AC_2. This action allows to turn on a new server $S2$ when the incoming load is high and more resources are needed. When this action is applied for the first time, two servers become available in the system. This means that now all the actions can be used, even migration and server turning on and off. As before, tests have been run by simulating the behavior of the system under different loads and the resulting quality matrix is shown in Fig. 7.5. Results for actions A_2 and A_3 are similar to the previous configuration. However, in this configuration the two VMs can move from a server to another, so the relations previously discovered are not valid anymore, and the application of A_2 on a specific VM has no effect on the performance of the other one. The same considerations are valid for action, A_3, in a symmetric way. Action A_1 (migration) affects the

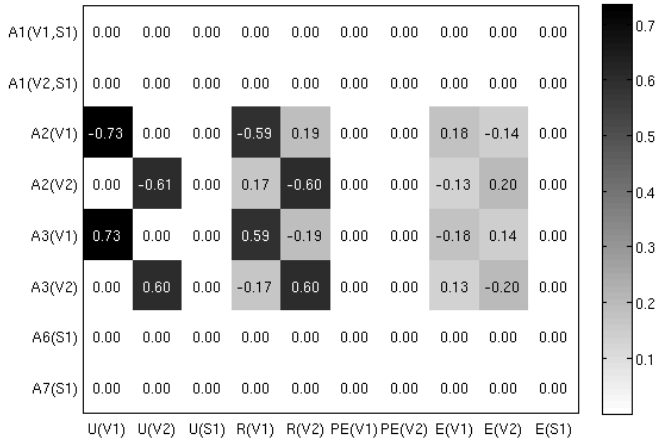


Figure 7.4: *Quality matrix for the C1 configuration*

usage of both servers, since when a machine is moved from a server to the other, the former decreases its usage and the latter increases it. It is also interesting observing that migrating in server $S1$ likely decreases the Performance per Energy and increases energy consumption of the migrated VM, while migrating on server $S2$ improves Performance per Energy while reducing energy consumption. This behavior is explained by the different features of the servers, since in the considered configuration server $S2$ is more efficient than server $S1$.

Actions A_6 and A_7 turn on or off a specific server and influence in this way the CPU usage of the server and its energy consumption. More specifically, action A_6 likely increases both the server CPU usage and energy consumption, while action A_7 decreases both these indicators.

Finally, actions AC_2 has an outcome very similar to the migration of a VM as shown in the last four lines of Fig. 7.5.

Test - Test on configuration C2 In this configuration, all the selected repair actions can be tested. The availability of two servers enables migration of VMs from a server to the other, and in some cases it could be also possible to turn off one of them, and consequently turn it on again.

Actions A_2 and A_3 can be tested taking the ID of the VM to be reconfigured as a parameter, as in the previous test. Action A_1 takes as parameters the ID of the VM to migrate and the ID of the server that is going to host it. Actions A_6 and A_7 have as parameter the ID of the server.

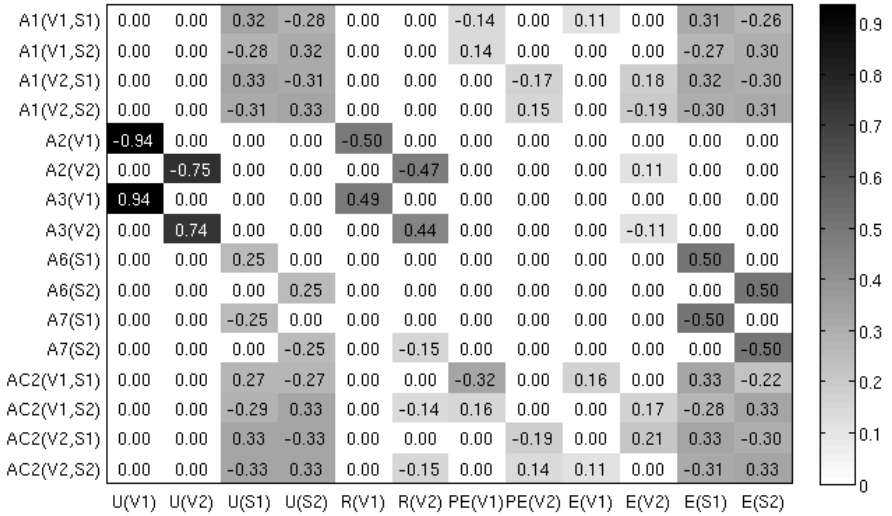


Figure 7.5: *Quality matrix for the C1 configuration with complex actions*

As before, tests have been run by simulating the behavior of the system under different loads and the resulting quality matrix is shown in Fig. 7.6. As can be observed, migrating a VM in a server, likely increases the server usage while decreasing the usage of the previous server. The same effect can be observed about the energy consumption of the servers.

Actions A_2 and A_3 confirms their behavior observed in the previous tests. A_2 likely decreases the value of the CPU usage for the VM and also reduces response time. The energy consumption of the VM has also a slight probability of being increased. In this configuration the three VMs can move from a server to another, so the application of A_2 on a specific VM has no effect on the performance of the others. The same considerations are valid for action A_3, in a symmetric way. As can be observed, both actions influence the energy of the involved VM, but this has no effect over the energy of the server. This is because, when a reconfiguration is applied to a VM, it can be deployed both on server $S1$ or on server $S2$, so a direct relation can not be discovered. Moreover, it is not said that an increase in the energy of a VM necessary results in the increase of the energy of the server where it is deployed. In fact, it can simply reduce the resources available for the other VMs deployed in the same server, keeping the total unvaried.

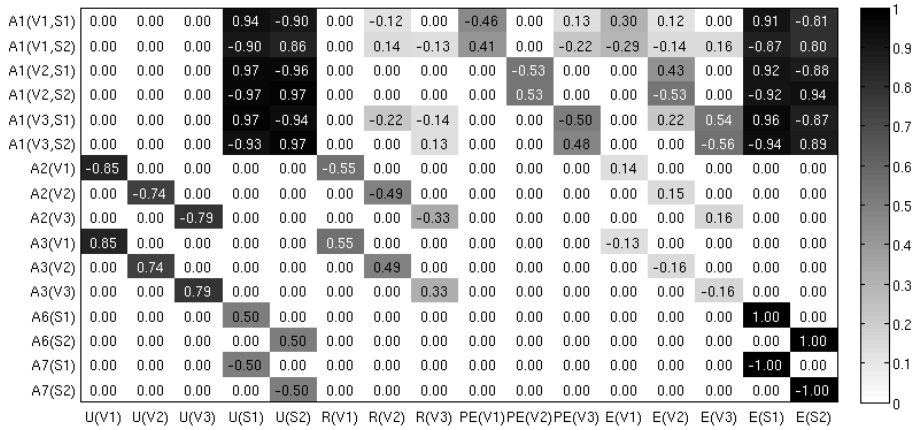


Figure 7.6: *Quality matrix for the C2 configuration*

Test - Test on configuration C2 with complex actions In this configuration, we tested also the complex actions AC_1 and AC_2 in the two servers configuration.

The outcomes obtained for simple actions is the same as the previous test. For this reason, we discuss only the results related to the application of the complex actions.

Using action AC_1, which migrates all the VMs on a server and then turn this server off, the usage of the server to be turned off is reduced. At the same time, the usage of the other server is increased, since new machines are migrated on it. This behavior reflects also on the energy usage. Action AC_1 affects also the Performance per Energy. This can be justified by the fact that migrating VMs from server $S1$ increases their performance because $S2$ is more efficient than $S1$, and the other way around. The same is valid for the energy of the VMs.

Action AC_2 behaves similarly to migration, affecting the usage of both servers, their energy, and the efficiency and energy of the VMs.

Results are shown in Fig. 7.7.

7.5 Improving Efficiency with the AAS Approach: Exploitation

After the analysis and evaluation of how the Adaptive Action Selection algorithm can be used to learn the impact of the actions over the goals of the system (Sect. 7.4), in this section we analyze the behavior of the algorithm

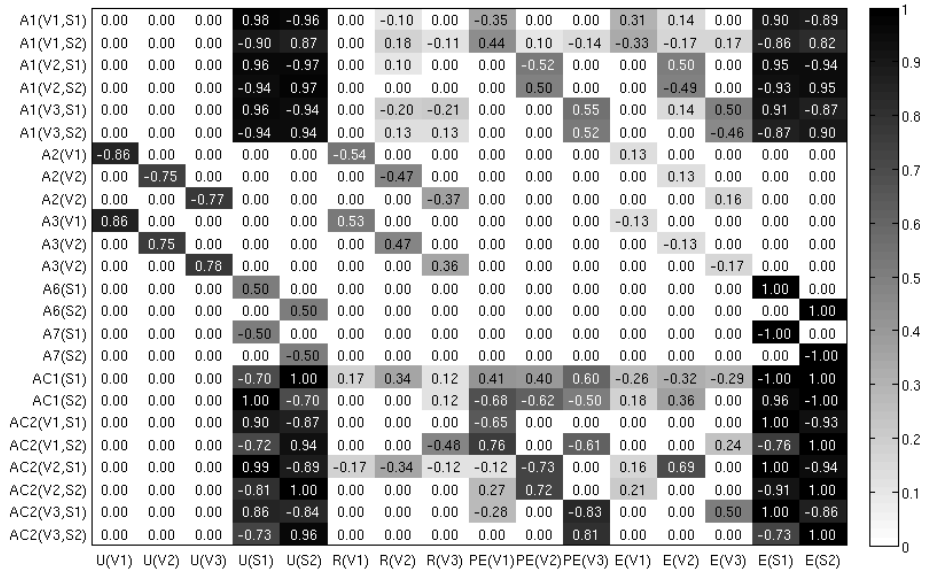


Figure 7.7: *Quality matrix for the C2 configuration with complex actions*

in suggesting the most likely efficient repair action. In the previous example, the aim of the procedure was to understand the ability of an action in increasing the value of an observed metric. The quality in the selection of the action at each step was not very important, since even a wrong choice adds information to the model. Now, we want to optimize the success of the algorithm in the selection of the best action, starting from the knowledge acquired in the previous step.

The algorithm applied is exactly the same described in Alg. 2. At each step, an action is selected considering the violations in the system, its outcome is observed and the quality matrix $\mathcal{Q}$ is updated accordingly.

In order to be able to evaluate the performance of the algorithm, we define a distance function $\mathcal{D}$ as follows:

Definition 12. *The distance $\mathcal{D}$ of a context of the system is defined as the sum of the distances of the state of each indicator from the optimal state. The state of an indicator is its value after the application of the thresholds about its satisfaction. More in details, an indicator in the normal zone has distance value $d(I_n) = 0$, an indicator in the warning zone has distance value $d(I_n) = 1$, and an indicator in the alarm zone has distance value $d(I_n) = 2$. The total distance is computed as:*

$$\mathcal{D}(\mathbb{C}) = \sum_{n} d(I_n|\mathbb{C}) \tag{7.10}$$

Starting from this definition we can define a score function $\mathcal{S}$ between two contexts as:

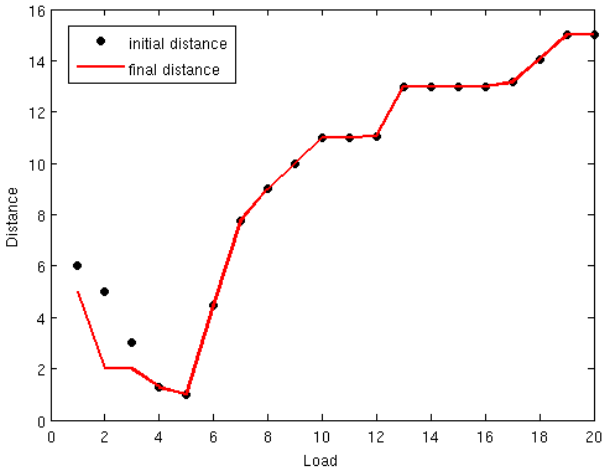
Definition 13. *The score function $\mathcal{S}$ measures the improvement obtained moving from a context $\mathbb{C}_1$ to another context $\mathbb{C}_2$. The score is computed as:*

$$\mathcal{S}(\mathbb{C}_1, \mathbb{C}_2) = \mathcal{D}(\mathbb{C}_2) - \mathcal{D}(\mathbb{C}_1) \tag{7.11}$$

Starting from the definition of distance and score, we evaluate the ability of the algorithm in selecting the best repair action using different initial system configurations. The set of actions and indicators considered are the same used in the previous section. In particular, we consider both configuration **C1** and configuration **C2**. For each configuration we compare the results obtained using the set of simple actions and the set of actions enriched with the complex actions.

In order to evaluate the algorithm, we run it using an incremental load, starting from 0 and increasing to 20. For each load, the algorithm has been executed 50 times, and the average value of each run has been computed to have a more reliable result. In the tests, we have introduced a sort of black list for VM migration. In a real system, migration is a very expensive activity, both in terms of performance and energy. Even when migration is the only action that can be executed, it should be used with moderation. For this reason we have created a mechanism that avoids the migration of a specific VM for a fixed amount of time after the execution of a migration. It means that, whenever a VM is migrated, it has to wait a fixed time to be migrated again. The system we are analyzing is quite static. Given an incoming load, it is not always possible to reach the optimal configuration, but some violations can not be solved. We consider a test step finished when the system enters into a static condition in which the "Do Nothing" action is selected for a given number of times in a row.

As before, the system has been tested using the simulation model described later in Ch. 8. For each configuration we show the average improvement in the distance function for each load rate and the average number of application of each action. The average is computed considering fifty runs in the same initial configuration and with the same load.



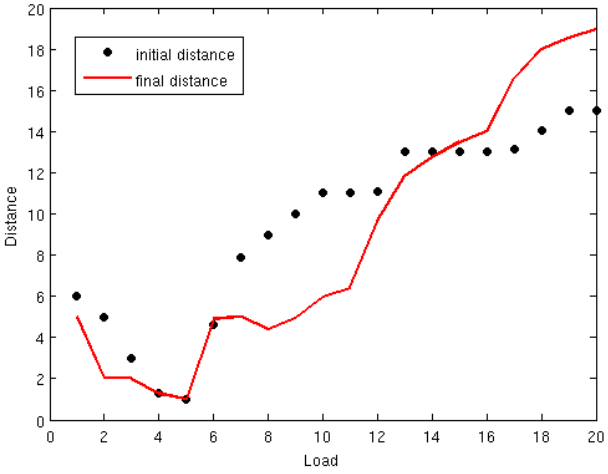
(a) Distance improvement

	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	L=11	L=12	L=13	L=14	L=15	L=16	L=17	L=18	L=19	L=20
A1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A3	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Count of actions application

Figure 7.8: *Results of the AAS algorithm at different load rates in the configuration C1*

Test - Test on configuration C1 In this configuration we have two VMs running on the same server. Only actions A_2 and A_3 can be applied, while constraints for actions A_1, A_6, and A_7 are never satisfied. Results of the application of the AAS algorithm under several workloads is shown in Fig. 7.8. As can be observed, the algorithm is able to decrease the distance of the initial configuration only when the load is low. In this case, A_3 can be applied improving the state of the CPU usage indicator for the two VMs. When the load increases, this indicator can not be improved anymore since no more resources can be added to the two VMs because all the resources of the server are already allocated. Fig. 7.8(b) shows how many times an action is applied for a given load in average. As can be observed, for $L > 3$ no action is applied. Results show that given the set of actions available, the algorithm does what it is possible to improve the state of the system.



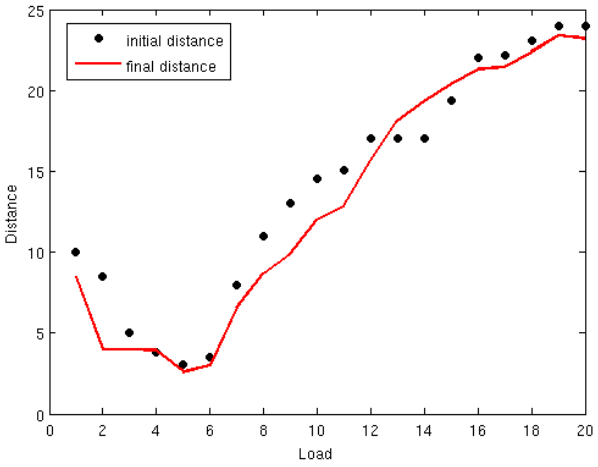
(a) Distance improvement

	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	L=11	L=12	L=13	L=14	L=15	L=16	L=17	L=18	L=19	L=20
A1	0	0	0	0	0	0.58	0.32	0.44	0.12	0.36	0.4	0.2	0.36	0.36	0.44	0.32	0.12	0	0	0
A2	0	0	0	0	0	0.98	1	1	3	3	3	3	4	4	4	4	4	4	4	4
A3	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC2	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(b) Count of actions application

Figure 7.9: *Results of the AAS algorithm at different load rates in the configuration C1 with complex actions*

Test - Test on configuration C1 with complex actions In this test we use again the configuration **C1**. This time we enrich the set of actions with the complex action AC_2 (Turn On and Migrate) which turns on a new server and migrate a VM on it. In order to be able to apply this action we added a server in the system configuration which is initially off. Results are shown in Fig. 7.9. As can be observed, the AAS algorithm behaves better than in the previous example. In the first steps, the behavior is the same and the algorithm simply applies action A_3 to increase the CPU usage indicator at the VM level. For $L = 4$ and $L = 5$ no action is applied. The system is very near to the optimum and none of the available actions is able to improve the state. For $L >= 6$, Fig. 7.9(b) shows that the algorithm starts to select AC_2 for execution. This action enables



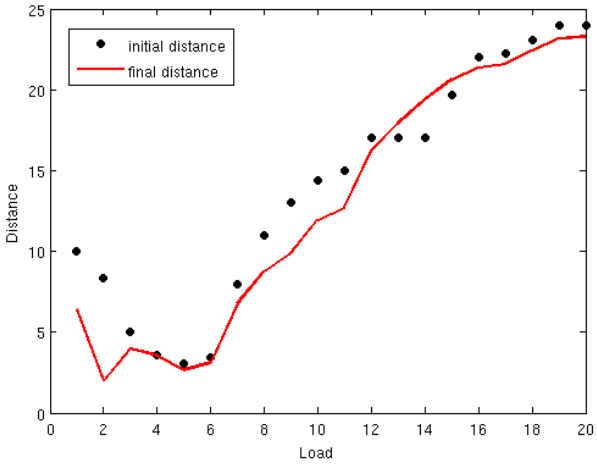
(a) Distance improvement

A1	0	0	0	3.74	3.98	3.2	3.12	2.54	2.16	1.64	2.26	0.78	0.98	0.52	0.52	0.3	0.22	0.34	0.38	0.2
A2	0	0	0	0	0	0.96	0.92	1	1.56	1.44	1.78	1.9	2	2	2	2	2	2	2	2
A3	3	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	L=11	L=12	L=13	L=14	L=15	L=16	L=17	L=18	L=19	L=20

(b) Count of actions application

Figure 7.10: *Results of the AAS algorithm at different load rates in the configuration C2*

the availability of more resources, allowing for other migrations and for resources reconfiguration. In fact, increasing the load, all the VMs need a higher amount of CPU allocated for satisfying the relative indicators. For $L > 14$ the distance after the adaptation is greater than the initial one. This is due to the fact that, for high load rates, even the second server is not enough for satisfying the requests. The number of violated indicators increases because now also a new server is on and some of the indicators related to it are violated. However, despite of the increasing in the distance, the algorithm improves the situation giving as much resources as possible, even if not enough. Even if the indicators and the distance function are not able to properly describe it, we can say that the algorithm behaves correctly.



(a) Distance improvement

	L=1	L=2	L=3	L=4	L=5	L=6	L=7	L=8	L=9	L=10	L=11	L=12	L=13	L=14	L=15	L=16	L=17	L=18	L=19	L=20
A1	0	0	0	3.9	3.76	3.16	2.48	2.56	2.18	1.64	1.84	0.58	0.9	0.42	0.3	0.36	0.24	0.28	0.18	0.3
A2	0	0	0	0	0	0.88	0.96	1	1.42	1.46	1.76	1.94	2	2	2	2	2	2	2	2
A3	3	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AC2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b) Count of actions application

Figure 7.11: *Results of the AAS algorithm at different load rates in the configuration C2 with complex actions*

Test - Test on configuration C2 This configuration consists in two servers running three VMs. The set of actions we are considering are the simple set considered since here: migration, reconfiguration, and turning off and on of a server. Results are shown in Fig. 7.10. Here we can see that the algorithm is able to improve the system state by reducing the distance from the optimal configuration in most of the cases. Fig. 7.10(b) shows the average number of times each action is applied. For low load rates, the AAS enacts resources reconfiguration by removing cores to the deployed VMs. For higher loads, migration is used to find a better configuration for the system, and action A_2 is used to increase the amount of resources allocated when spare resources are available. In this context,

nothing else can be done to improve the system state, without having additional resources.

Test - Test on configuration C2 with complex actions In this last test we observe the effect of introducing complex action in the configuration **C2**. We are considering two complex actions: AC_1 (Migrate and Turn Off) and AC_2 (Turn On and Migrate). Results are shown in Fig. 7.11. As can be observed in Fig. 7.11(a), results are very near to the previous case, shown in Fig. 7.10(a). An improvement can be observed for low load rates. In fact, as shown in Fig. 7.11(b), for $L < 3$ action AC_1 is selected, consolidating all the VMs on a single server. This behavior, significantly improves the distance for low rates when spare resources are available. On the contrary, action AC_2 is never applied, since there are no other servers available in the system.

7.6 Conclusion

In this chapter we have proposed the Adaptive Action Selection algorithm as a solution to provide a dynamic mechanism for learning the impact of a set of actions over the goals of the system and for suggesting and/or applying the best strategy in a given context. The algorithm is inspired to the Adaptive Operator Selection technique, but several modifications have been applied. First of all we added a contextualization to the algorithm, considering multiple options for adaptation at each possible context. In fact, in our system, the best repair strategy is strictly dependent on the state of the goals of the system. The introduction of the context has an impact in the definition of all the steps of the algorithm.

The algorithm has been applied in two ways. First of all, we have used it for exploration. This phase is useful for building a missing part of the goal-oriented model introduced in Ch. 5: the action-indicator relations. These relations have been learned by simulating the application of the algorithm under different loads and in different contexts, and observing the outcomes of the applied actions over the goals. We have performed tests for different configurations of the system and with different sets of actions. Results have shown that the system is able to learn the desired relations with a high confidence without introducing noise.

After that, we have applied the algorithm for exploitation, by selecting the best adaptation strategy given a context. Again, we have executed

several tests under different loads and with different configurations and actions. Results have shown that the algorithm is able to improve the system state when there are actions available for obtaining this results. Otherwise, the algorithm simply chooses the "Do Nothing" action.

The good results obtained in the tests have to face the problem of scalability. In fact, even if the algorithm performs well in a small system, it can be not the same when the amount of indicators increases and the number of actions to test is elevated. We modeled the algorithm in order to reduce as much as possible the issue of the scalability, however, other considerations can be made. Scalability principally affects the first phase of the algorithm: the action-indicator relations learning. In fact, once this information is available, the action selection is a simple task. To improve the performance of the algorithm, the relations can be learned off line, using a simulation system. This operation has to be performed only once, then the values will be kept updated during the action selection phase. However, increasing the number of indicators and the number of actions makes the learning phase significantly slower.

Another additional solution consists in performing a generalization of the relations between actions and indicators. Observing the quality matrix obtained by the application of the algorithm in the learning phase, it is possible to see that the same action behaves similarly on similar indicators, depending on the parameters used to execute the action. Generalizing the action behavior can significantly reduce the number of training steps needed for obtaining a confident knowledge of the effect of actions over indicators. This solution is proposed in Ch. 9 as a future development of this work.

CHAPTER 8

Modeling Service Execution on Data Centers

Indice

This chapter provides a system modeling approach to describe a virtualized data center environment running a Business Process (BP). This model allows the collection of simulation data at different workload rates that can be used as a dataset for machine learning in order to reason about Quality of Service (QoS) and Energy Efficiency (EE) issues related to the

process. The model overcomes the issue of obtaining real data and collecting relevant information needed for the study of the behavior of the BP in environments where a direct interaction with the real system can influence the performance of the services provided. The model is also flexible and can be used to model data centers of different dimensions and characteristics. Another use of the model is "what-if" analysis about the system configuration, predicting the outcome of a modification over EE and QoS.

8.1 Introduction

Energy Efficiency (EE) management and energy usage reduction are important topics in the data centers environment. Data centers consume a considerable percentage of the worldwide global energy and, most of the time, available resources are not properly used. In the context of Service Oriented Architectures (SOAs), Energy Efficiency is usually in conflict with Quality of Service (QoS). The tradeoff between these two aspects is difficult to manage in an efficient way.

Models are fundamental in a lot of different fields. They can be used in an offline and online mode to evaluate new configurations and solutions to a problem both at design and run time.

In the previous chapters, we have seen how it is possible to learn relations between indicators and how the best adaptation strategy can be dynamically learned interacting with the real system. Since learning on a real and running system can be dangerous for its performance, a simulation environment can be useful to avoid issues. It can be used to analyze the relations between indicators and the impact of the actions in an off-line mode. In this way, actions will be applied to solve inefficiencies in the real system only when the knowledge about their effect is reliable.

One advantage in defining a model of a data center, is the collection of data reports of energy usage and QoS reflecting a real system. This allows us to analyze and adapt its design (e.g. resource allocation, VM-server-activity mapping, etc) toward a dynamic balance between Energy Efficiency and adequate QoS. This goal presents multiple challenges. First, a predictive model of the system or some means of predicting the outcomes of a system design change is necessary. Further, online changes are required and these must be monitored. The collection of real monitoring data for energy and QoS is not trivial. The first issue is the availability of such an observable environment, with a monitoring system installed on it. Even when these conditions are satisfied, collecting data for a wide range of con-

figurations is not an easy task. In fact, not all the configurations are frequent and for some of them data can be not enough for running analysis about them. Even the interaction with a real system for testing purpose is not trivial, since modifying the system configuration can affect performances of the service delivery. Moreover, data center owners are reticent to give access to data even if the customers' data are not of a sensitive nature, and even more reticent to allow the access to their system and its modification.

Building a model of a virtualized environment is challenging. In fact, with virtualization, a system which appears to function in isolation, actually does not because of the dependencies between Virtual Machines (VMs) running on the same server. It is important to understand the behavior of the system under study while also incorporating relevant influences of the "neighboring" systems in the virtual environment.

Given these premises, in this chapter we propose a system modeling approach which helps addressing the aforementioned challenges by approximating the behavior of a real system. The model allows clear expression, calibration and prediction. Modeling allows the collection of simulated monitoring data and the experimentation of different workload levels that could be difficult to test in a real environment. The model of the data center is also flexible. It can be enriched with more components and parametrized depending on the characteristics of the Business Process considered. The structure of the system, in terms of resources allocated, number of active servers, positions of the VM, can be modified at any time. The simulation can be used to conduct a what-if analysis to evaluate the outcome of a modification over Energy Efficiency and QoS in an off-line modality. Even if approximated in comparison to a real system, the proposed approach allows testing of configurations impossible to perform in a real environment, without affecting its performance.

The class of systems we aim to model is the same introduced in Sect. 4.1 and shown in Fig. 4.1. The typical system is a data center composed by a set of servers monitored by a monitoring system upon which a service is executed. The service is represented as a work-flow of activities, each one running on a dedicated VM. The probability of executing alternative branch is known from the analysis of historical data. The system's monitoring subcomponents will periodically sub-sample the system state to enable model calibration and validation as well as support real-time analysis.

The chapter is structured as follows: Sect. 8.2 introduces other papers using models for data centers or server components. In Sect. 8.3 we describe how the system has been modeled in all its components. Sect. 8.4

explains how monitoring data have been collected and analyzed by simulating an incoming load. Conclusions and future work are discussed in Sect. 8.5.

Part of the content of this chapter has been published in [129].

8.2 State of the Art

Simulation environments are a convenient help when it is not possible to perform experiments in repeatable, dependable, and scalable environments using real-world environments. Through simulation it is possible to evaluate hypothesis while being able to reproduce the obtained results. At the same time, simulation can help to detect system bottlenecks before deploying on the real system and to test the behavior of the infrastructure under several workloads.

In this chapter we want to create a model of a data center from which to collect useful information about the system state at different workloads. Other similar approaches can be found in literature. As an example, a recent work has been proposed with the name of CloudSim [130]. CloudSim is a simulator for cloud environments which supports several kinds of workload. In the definition of the data center it is possible to define the number of servers and their resources, the VMs and their deployment, together with the kind of workload running on them. VMs are detailed with the amount of processing power assigned to them, and the same information can be specified for each single task. Even the network communication between nodes is modeled using a latency matrix representing the time needed for communicating between nodes. The model is enriched with a power consumption utility, giving estimation about power consumed both at the server and at the VM level. In the model also Service Level Agreement (SLA) violations are considered, expressed as number of Million Instructions Per Second (MIPS) not correctly allocated to the demanding VMs. The model has been used by the authors in [55], to run tests on a simulated cloud environment.

Another tool for simulating a data center environment is GreenCloud, described in [131]. Here, EE of the tested configuration is computed in terms of Power Usage Effectiveness (PUE) and Data Center Infrastructure Efficiency (DCiE) metrics, while the performance level is assessed using response time. Frequency scaling techniques are also simulated for reducing the energy impact. The simulator has been tested considering three different kind of applications: computational intensive, data intensive and

mixed.

Both the presented simulators are very detailed and functional, but they have some limitations that make them unsuitable for our work. In fact, using these tools, not all the information needed in our model can be collected. For instance, from CloudSim it is possible to obtain information about the application throughput, Central Processing Unit (CPU) utilization and energy consumption, but information about response time and Performance per Energy is missing. On the other hand, GreenCloud gives information about response time and energy, but throughput and usage of resources are not available.

Given these premises, we need a model able to provide energy consumption information. In Energy Efficiency research, power models are used to improve the design of single components and of the whole system. A collection of papers related to power models is provided in [16]. Different components can be considered in modeling power consumption, as can be seen in [132], [133] and [134], such as CPU, memory, I/O operations, and network transmissions. In [135], a model for real-time power consumption prediction is provided. Most of the approaches simplifies the problem of modeling power consumption of a server by considering only the CPU usage level [136] [137] [131] [54] [56]. This is because CPU is the component that has the main impact over energy. However, this approach is effective onlyfor CPU intensive workloads, while it is not valid in case of memory intensive applications, as discussed in [1].

Shifting energy estimation from the hardware to the software level is not simple. Some scholars try to estimate the energy consumption of an application from the knowledge of the amount of resources used by the application and the total execution time [21] [22]. Other approaches model energy consumption focusing on the Virtual Machine level, like in [23] and [24]. For more details on energy estimation and for a classification of the available techniques see Sect. 2.4.1 and Tab. 2.2.

The definition of a model for energy estimation at the application or Virtual Machine level is still an open issue and all the presented techniques use approximation in defining the energy consumption model.

The cited contributions are only a small subset of available fields in which models have been proposed to describe the behavior of a system and to allow testing and off-line reasoning on its behavior. The system modeled in this paper focus attention on virtualized data centers, incorporating application level information related to the Business Process and its parameters.

8.3 The Model of the Data Center

As introduced in Ch. 4, Energy Efficiency and QoS of a process executed over a data center are not directly measurable, but can be assessed using a set of indicators belonging to two families: Green Performance Indicators (GPIs) and Key Performance Indicators (KPIs). In this work we consider the subset of indicators collected in Tab. 4.2. Indicators values are computed from data collected through a real or simulated monitoring system.

In this section, we present a system modeling approach to simulate the collection of monitoring data. Some assumptions are made about the system that we are going to model:

- we model a Business Process (BP) in which each activity is deployed on a dedicated Virtual Machine;
- resources are not over-allocated, the sum of the resources allocated through virtualization is no more than the available physical resources;
- resources that are not used can be momentarily used by other Virtual Machines if needed.

The complete list of variables monitored and computed in the system is contained in Tab. 8.1.

The first step consists in modeling the incoming load in the system. We model the load using the variable L. L is an integer value expressing the number of requests for service execution received in a fixed amount of time. In our model we consider L as the number of requests received in a second. The number of requests that have to be handled by the single activities depends from the process structure and from the branch probabilities and it is expressed as l_i. For simplicity, we do not consider the propagation time from an activity to another, but we consider the load as in regime, since we expect the incoming load to variate slowly. As a consequence, at time t, the value l_i depends only from the incoming load L, but it is independent from the time needed to execute precedent activities in the BP.

In this section, the modeling of all the interesting components is analyzed, including CPU utilization (Sect. 8.3.1), response time (Sect. 8.3.2) and Performance per Energy (Sect. 8.3.3).

8.3.1 CPU usage

CPU usage of the Virtual Machine is strictly dependent on the workload and on the activity executed on the Virtual Machine. In [55], authors map

Table 8.1: *Variables in the system*

Symbols	Description
$i : i = 1 \dots I$	Virtual Machine i of the system
$j : j = 1 \dots J$	server j of the system
$k : k = 1 \dots K$	activity k of the system
d_k, d_i	need for CPU for activity k or Virtual Machine i
L, l_i	incoming load rate for the whole process and for Virtual Machine i
U_j, U_i	CPU usage value for server j or Virtual Machine i
V_i	CPU usage value for Virtual Machine i expressed in terms of server resources
R_i	response time of the activity k running on Virtual Machine i
PE_i	Performance per Energy unit of the activity k running on Virtual Machine i
P_j, P_i	instant power consumed by the server j or Virtual Machine i
E_j^T, E_i^T	energy consumed by the server j or Virtual Machine i in the time period T
CPU_j, CPU_i	total CPU allocated to the server j or Virtual Machine i

workload variations to CPU usage of VMs using a uniformly distributed random variable. This is justified by the unknown types of applications running on the system. We use the same approach in our system.

We consider that each activity k has an average usage need λ_k. Given this value, in each moment, the resource demand of the activity can be sampled by a distribution so that samples are around λ_k but with some small deviation, in order to simulate variations around the average usage rate. We decided to sample CPU usage from a Poisson distribution, in which the parameter λ_k represents the more probable value. The Poisson distribution has been chosen because it samples integer positive values around the more probable one. The demand for activity k can be defined as:

$$d_k \sim Pois(\lambda_k) \tag{8.1}$$

with $d_k : [0...D_k]$ an absolute value representing the amount of resources needed.

Given the instantaneous load value l_i, the estimated resource need of Virtual Machine i to execute activity k is:

$$d_i = \begin{cases} \frac{d_k}{CPU_i} & IF\ l_i \leq 1 \\ \sum_{l_i} \frac{d_k}{CPU_i} + f(l_i) & IF\ l_i > 1 \end{cases} \tag{8.2}$$

with $d_i \in [0...D_i]$ a value representing the amount of resources needed expressed as a fraction of the resources on Virtual Machine i. In Eq. 8.2, the value $f(l_i)$ is an overload due to the management of the request queue when more than 1 request is received.

From Eq. 8.2 we can derive the usage of Virtual Machine i as a direct consequence of its resource demand as follows:

$$U_i = min\left\langle d_i, 1\right\rangle \tag{8.3}$$

with $U_i \in [0...1]$ expressed as a ratio of the Virtual Machine resources.

This model derives the CPU usage so that all the available resources are used if needed, without exceeding the maximum available which is 1.

Starting from sampled values from the defined distributions for each Virtual Machine in the example system, it is possible to compute the CPU usage at the server level. In this case, we expect the CPU usage of the server to be proportional to the CPU usage of the Virtual Machines running on it, plus a constant value s representing the CPU used by the server itself. We also add a noise parameter η to the equations. We represent the CPU usage model for the server in normal conditions in Eq. 8.4:

$$U_j^{min} = min\left\langle 1, \sum_{i \in j} w_i U_i + s + \eta \right\rangle \tag{8.4}$$

with $U_j^{min} \in [0...1]$ expressed as a ratio of the server resources.

The weight w_i represents the ratio of CPU on the server allocated to the Virtual Machine and can be expressed as:

$$w_i = \frac{CPU_i}{CPU_j} \tag{8.5}$$

where $\sum_i w_i <= 1$.

The usage of each Virtual Machine is not isolated and when the demand of one of them is high, it can use more resources than the one directly assigned, by stealing unused CPU cycle if they are available or competing with other Virtual Machines for them. For this reason, when defining the CPU usage for a server, this can be bigger than the sum of the usage of

the single Virtual Machines running on it. We assume that all the available resources are allocated, if needed, and we define the server usage as:

$$U_j = min \left\langle 1, \sum_{i \in j} w_i d_i + s + \eta \right\rangle \tag{8.6}$$

with $U_j^{min} \in [0...1]$ expressed as a ratio of the server resources.

From Eq. 8.4 and Eq. 8.6 it is possible to derive the amount of resources on the server that are reallocated to one or more of the Virtual Machines running on it as:

$$U_j^* = U_j - U_j^{min} \tag{8.7}$$

with $U_j^* \in [0...1]$ expressed as a ratio of the server resources. The amount of extra resources used by a Virtual Machine is a portion of this value proportional to the unsatisfied demands of all the Virtual Machines on the server j:

$$U_i^* = min \left\langle d_i - U_i, \frac{d_i - U_i}{\sum_{i:d_i > U_i} w_i (d_i - U_i)} * U_j^* \right\rangle \tag{8.8}$$

with U_i expressed as a ratio of the Virtual Machine resources.

The effective CPU usage of the Virtual Machine derives from Eq. 8.3 and Eq. 8.8:

$$U_i' = U_i + U_i^* \tag{8.9}$$

with U_i' expressed as a ratio of the Virtual Machine resources possibly bigger than 100%.

The resources used on the server j for the Virtual Machine i can be derived from Eq. 8.9 as:

$$V_i = w_i U_i' \tag{8.10}$$

with V_i expressed as a ratio of the server resources.

8.3.2 Response time

Response time is modeled here as the time needed for executing an instance of an activity. Response time is dependent from the characteristics of the activity, but also from the CPU usage of the machine running the application. When CPU usage is low, the application can take easily all the

resources needed and complete in a small amount of time. When the CPU usage is high, the application have to wait for resources, and response time value increases.

As defined in [137], response time of an activity can be expressed by the following relation:

$$R = \frac{a * usage}{b - usage} + c \tag{8.11}$$

with a, b, and c parameters dependent from the activity.

This expression is generic and does not fit properly a virtualized environment. If the server in which the Virtual Machine is deployed is overloaded, the response time is longer because of the non isolation of the resources shared between Virtual Machines concurrently running on the same server. In our model, we want to express the relation between the average response time of an activity running on a Virtual Machine, taking into account that the real usage of the Virtual Machine can be bigger than 1 if it is using spare resources on the server. We also want to model response time so that an exponential growth can be observed when the resources available for the Virtual Machine are less than the one demanded. Hence, we modify Eq. 8.11 in order to model it properly to describe activity k deployed on Virtual Machine i as follows:

$$R_i = t_k + \frac{t_k 10^{-2} U_i}{[1 - (d_i - U_i')]^{\gamma}} \tag{8.12}$$

with $\gamma > 1$. In our experiments we used $\gamma = 2$.

According to this model, defined t_k as the minimum amount of time to complete the activity k on Virtual Machine i, response time is only slightly increased with the usage of the Virtual Machine, with an exponential increasing when not enough resources are available.

8.3.3 Performance per Energy

The Performance per Energy metric gives a measurement of the number of operations the activity can complete using an energy unit. The mathematical definition is as follows:

$$PE_i = \frac{N_i^T}{E_i^T} \tag{8.13}$$

where N_i^T is the number of operations completed in the period of time T and E_i^T is the energy used in the same period.

The numerator of Eq. 8.13 is strictly dependent on response time. We can consider the total number of operations executed for each request by the activity k as a constant value N_k. Knowing the response time of the activity, which is the time to complete all its operations, the number of operations completed in the period T can be expressed as:

$$N_i^T = \frac{N_k}{R_i} * T \tag{8.14}$$

where the activity k is the one deployed on Virtual Machine i.

To derive the second term of equation 8.13 we have to define the power consumed by the server. According to [131], power consumed by a server can be computed as the combination of the idle power consumption plus a component dependent on the CPU usage. The idle power is a portion of the power consumed in peak usage condition, expressed as P_j^{peak}. From this value we can derive a formula for modeling the server power consumption as follows:

$$P_j = \underbrace{\alpha P_j^{peak}}_{idle} + \underbrace{\beta P_j^{peak} * U_j}_{usage\ dependent} \tag{8.15}$$

where α and β are equal to 0.66 and 0.34 according to [131]. Similar values are provided in [136] where authors assign to idle server the 60% of the peak power consumption, and in [1], where the idle power consumption is indicated as the 70%.

Another model is provided in [137] similar to Eq. 8.15, and in [1] where the same linear model is proposed.

The power consumed by the whole server has to be split for all the Virtual Machines running on it. If we consider Eq. 8.15, the only portion of energy that can be ascribed to the Virtual Machines is the non idle part. Assuming the server is not doing anything except running the Virtual Machines, power can be split on the basis of CPU utilization.

$$P_i = \beta P_j^{peak} * w_i U_i' \tag{8.16}$$

where w_i is defined as in Eq. 8.5 and U_i' is defined as in Eq. 8.9. The same approach for accounting power consumed by the server on the Virtual Machines running on it is used in [56]. Here, instead of computing power consumption of the Virtual Machines, the authors take into account the computing power for estimating the efficiency of the applications using the Virtual Machines.

Table 8.2: *Parameters in the system*

Parameter	Description
s	server CPU usage that is independent from the Virtual Machines deployed on it
η	noise added to the server usage computation
γ	parameter for the exponential behavior of the response time
α, β	parameters for the computation of the power consumption knowing the peak power usage
t_k	minimum time for completing activity k
λ_k	average resource need of activity k

Energy can be computed integrating the power over the time period considered, both for the server:

$$E_j^T = \int_T P_{j,t} \, \mathrm{d}t \tag{8.17}$$

and for the Virtual Machine:

$$E_i^T = \int_T P_{i,t} \, \mathrm{d}t = \int_T \beta P_{j,t}^{peak} * w_i U'_{i,t} \, \mathrm{d}t \tag{8.18}$$

All the parameters of the model are grouped and described in Tab. 8.2. The model defined to this point must next be calibrated to express the physical equipment used, the applications running on the Virtual Machines, the structure of the data center, and the mapping between activities, Virtual Machines, and servers.

For each component we also have to define parameters that allow us to work with the model. All parameters are described in Table 8.3. They are divided into three categories: Server, Virtual Machine and Activity. Each server in the system is defined by an identifier, the amount of CPU available in the server, thresholds over the usage of the CPU, the peak power consumption of the server and thresholds over the energy usage. Similarly, VMs are defined with an id, the amount of CPU allocated and relative thresholds, energy consumption thresholds and the id of the activity running on it. The activity is defined by its id, the number of operations executed at each request, the average CPU demand of each request, the average time needed to complete a request, response time and Performance per Energy constraints, and the probability to be executed (dependent from the workflow),

Table 8.3: *Parameters for the Data Center Definition*

Server	
id	an integer value that is unique for each server
cpu	number of CPUs in the server
cpuT.trH, cpuT.tyH, cpuT.trL, cpuT.tyL	threshold set for the CPU usage when it is respectively in the high emergency, high warning, low emergency, and low warning interval
peak	power consumption of the server in the peak situation
E.trH, E.tyH	threshold set for the Energy consumption when it is respectively in the high emergency and high warning interval
Virtual Machine	
id	an integer value that is unique for each Virtual Machine
cpu	number of CPUs in the Virtual Machine
cpuT.trH, cpuT.tyH, cpuT.trL, cpuT.tyL	threshold set for the CPU usage when it is respectively in the high emergency, high warning, low emergency, and low warning interval
activity	name of the activity assigned to the Virtual Machine
E.trH, E.tyH	threshold set for the Energy consumption when it is respectively in the high emergency and high warning interval
Activity	
id	an integer value that is unique for each activity
N	number of operations typically processed by the activity
R.trH, R.tyH	threshold set for the response time when it is respectively in the high emergency and high warning interval
PE.trL, PE.tyL	threshold set for the Performance per Energy when it is respectively in the low emergency and low warning interval
lambda	average CPU demand for the activity in medium load
splitProb	probability of execution of the activity in the work flow
completionTime	average time needed in normal condition to complete an instance of the activity

8.4 Model Validation

Once the model has been defined, it can be validated for relational correctness by simulating an incoming load into the system. Validation is used to check whether the model expresses the key relational properties of the

system, reflecting how this class of systems reacts to different load regimes.

Samples values for all the metrics of the system can be obtained giving a load value as input of the model. The load represents the number of requests that has to be handled at a given time by the process. Since each activity has a probability to be executed due to the branches probabilities, the general load has to be redistributed according to these values. As an example, in the Business Process in Fig. 4.1, Activity1 and Activity5 are always executed, so they will have to handle the 100% of the incoming load, while Activity2, Activity3 and Activity4 will handle only a portion of it depending from the branch execution percentage. This value is expressed as a parameter of the activity.

Given a load value L, different activities can require different CPU resources, This feature has been modeled using the activity parameter λ_k, representing the average CPU usage of activity k. Eq. 8.2 allows us to derive the resource demand d_i of a VM as:

$$d_i = \sum_{l_i} \frac{Pois(\lambda_k)}{CPU_i} \tag{8.19}$$

where $l_i = L * splitProb_k$.

Substituting this value in the formulas described in Sect. 8.3 results in the derivation of simulated values of the monitoring system.

Tests have been run both for verifying a correct behavior of the components of the model (Sect. 8.4.1) and for proving the usefulness of the model in supporting what-if analysis (Sect. 8.4.2). The model has been implemented using MATLAB [123]. The code used for the experimental part presented in this section is available at [138]. Configurations used for the data center components are reported in Tab. 8.4.

8.4.1 Testing the model behavior

The model has been tested with two system configurations to observe the behavior of all the monitored components at the variation of the load. In both tests, an incremental load was used to test the model, starting from 0 and increasing to 20 for one Virtual Machine while keeping the others stable at a constant load of 5, and the same was repeated for each VM.

It is worth noticing that in this phase we considered each activity as independent, neglecting the structure of the process, for testing more deeply the behavior of the system.

Table 8.4: *Data Center configuration parameters for the tests*

SERVERS			VIRTUAL MACHINES	
id	***CPU***	***peak***	***id***	***CPU***
S1	4 cores	350 W	V1	2 cores
S2	4 cores	280 W	V2	2 cores
S3	4 cores	320 W	V3	2 cores

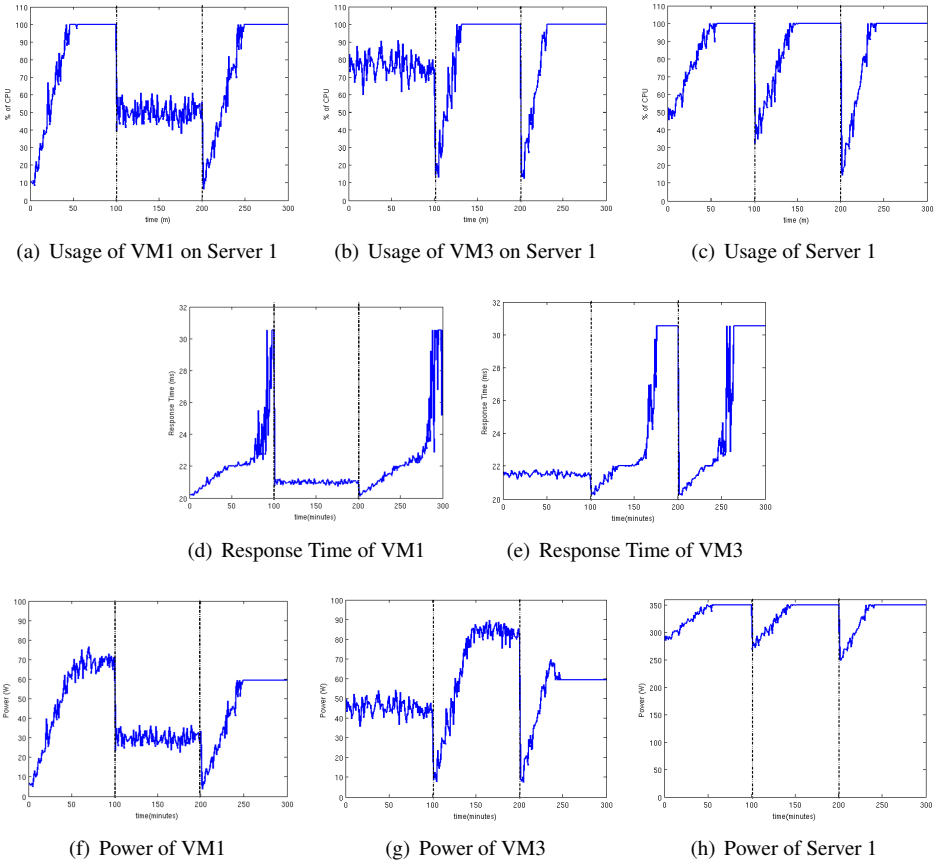
ACTIVITIES				
id	***N***	***lambda***	***splitProb***	***completionTime***
A1	1000	0.2	1	20 seconds
A2	10000	0.3	1	20 seconds
A3	5000	0.25	1	20 seconds

Test - Test on configuration C1 As a first test, the model was calibrated on a system composed by one server, $S1$, and two Virtual Machines, $V1$ and $V3$, deployed on it. Each Virtual Machine runs an activity, respectively $A1$ and $A3$ (see configuration **C1** described in 6.3).

The incoming load behaves as described before, with an additional step in the end of the experiment where the load has been incremented together for both Virtual Machines.

Values obtained for the system are represented in Fig. 8.1. Observing Fig. 8.1(d) and Fig. 8.1(e), it is possible to see that response time starts to increase exponentially when the the server hosting the two Virtual Machines reaches the full utilization. This is because the VM can not steal any more CPU cycle from the other one which is in competition. The power consumed by the two VMs at full load when both are asking for resources is less than if only one is asking for resources while the other one is stable. This is again because of the stealing of CPU cycles that is not possible when both machines need resources. This behavior can be observed in Fig. 8.1(f) and Fig. 8.1(g).

Test - Test on configuration C2 Then, the model was calibrated for a system with two servers $S1$ and $S2$ and three Virtual Machines $V1$, $V2$, and $V3$ (see configuration **C2** described in 6.3). This model is a part of the system shown in Fig. 4.1. In the system, $V1$ and $V3$ are deployed on $S1$, and $V2$ on $S2$. An activity of our process is deployed on each Virtual Machine ($A1$ in $V1$, $A2$ in $V2$ and $A3$ in $V3$). To observe the load im-



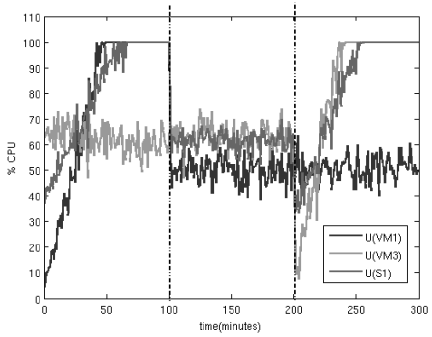
(a) Usage of VM1 on Server 1 (b) Usage of VM3 on Server 1 (c) Usage of Server 1

(d) Response Time of VM1 (e) Response Time of VM3

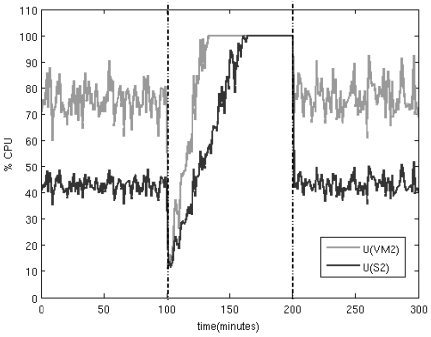
(f) Power of VM1 (g) Power of VM3 (h) Power of Server 1

Figure 8.1: *Monitoring values with configuration C1*

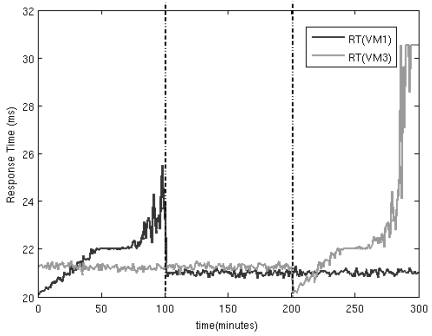
pact, an incremental load was used to test the model, starting from 0 and increasing to 20 for the first Virtual Machine while keeping the other two stable at a constant load of 5, and the same was repeated for each VM. Values obtained for the system are represented in Fig. 8.2, where the first line contains the plots of all the values monitored on server $S1$ and the second all the values monitored on server $S2$. The CPU usage values of different VMs are independent from each other, but they influences the CPU usage of the server where they are deployed. This relation is shown in Fig. 8.2(a) and Fig. 8.2(b). Even response time values are independent from each other, but they are dependent from the usage of the VM's CPU and response time values increase exponentially when CPU gets saturated (see Fig. 8.2(c) and Fig. 8.2(d)). Powers of different VMs



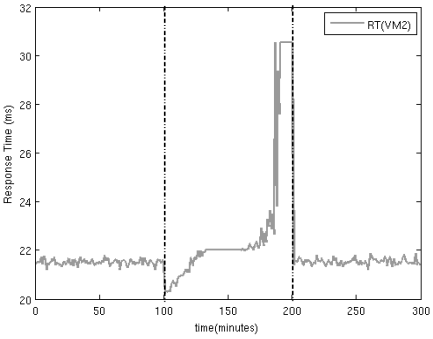
(a) Usage of VM1, VM3, and S1



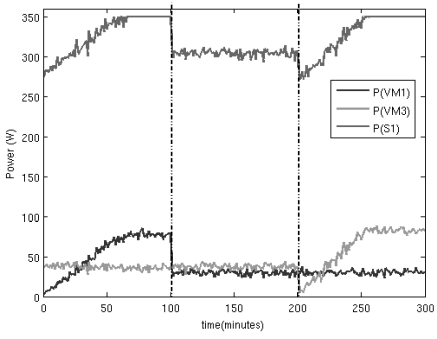
(b) Usage of VM2 and S2



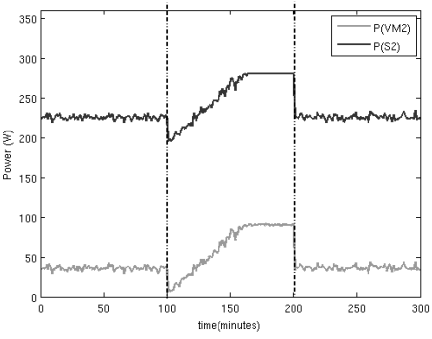
(c) Response Time of VM1 and VM3



(d) Response Time of VM2



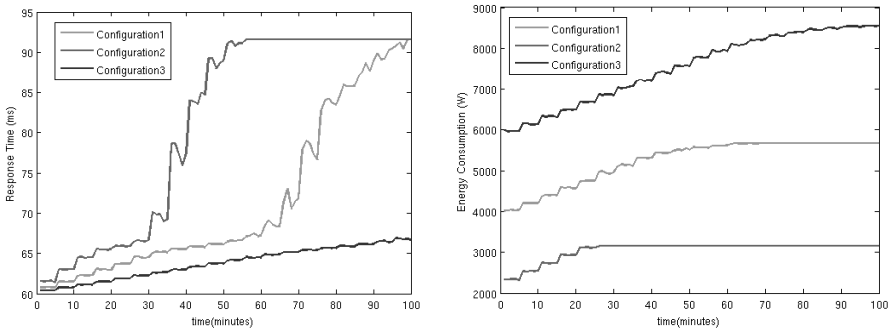
(e) Power of VM1, VM3, and S1



(f) Power of VM2 and S2

Figure 8.2: *Monitoring values with configuration C2*

are independent, but they are related to the power of the server which hosts them and they are also dependent from the CPU usage of the VM. This behavior is reflected in plots of Fig. 8.2(e) and Fig. 8.2(f).



(a) Comparison between response time values (b) Comparison between energy consumption values

Figure 8.3: *What-if analysis with three different configuration of the data center*

8.4.2 Using the model for what-if analysis

As stated before, the model can be also used for conducting what-if analysis about different server configurations. We used three configurations, evaluating QoS and Energy Efficiency for each of them. QoS has been expressed as the total response time, while for Energy Efficiency we considered the total energy consumed at the server level. In "Configuration1" the system has three Virtual Machines on two servers (as in configuration **C2**). Starting from this configuration we attempt two different modification of the system, evaluating their effects. In the first modification, we migrate VM2 on S1, and S2 is turned off. The new configuration obtained after migration is indicated as "Configuration2". In the second modification, a new server S3 is added, and VM3 is migrated on it. This new configuration is indicated as "Configuration3". Tests have been conducted by increasing the load with the same rate for all the VMs. From Fig. 8.3 we can see that changing to "Configuration2" improves the EE while response time grows more quickly than in the original configuration. On the contrary, "Configuration3" increases the energy consumption while affecting positively response time that is significantly reduced. The choice of the best configuration will depend on the goal of the system administrator.

8.5 Conclusion

This chapter presented a system modeling approach for describing the behavior of the components of a data center related to the deployment of a Business Process. Unlike other models for energy usage estimation, this ap-

proach focuses on the application level information and model a complex environment involving a whole data center using virtualization. Through this model, it is possible to collect simulated monitoring data that can be used in the study of strategies for improving Energy Efficiency and QoS of a Business Process. The model allows off line simulation of different workload levels and a what-if analysis, by changing the configuration of the data center and observing the generated data. The what-if analysis enables the evaluation of different system configurations, both in terms of Energy Efficiency and QoS, providing a tool for selecting the best solution given a load rate. Results discussed in Sect. 8.4 showed that the monitored system behaves as expected, with an increasing critical level as the load rate increases.

CHAPTER 9

Concluding Remarks

THIS chapter concludes this thesis work by firstly revisiting the problem statement and the achieved results, then by proposing a more general view of the adopted approach and summarizing the lessons learned during the development of the thesis, and finally by pointing out future directions.

9.1 Problem Statement and Results

In this work we proposed a goal-oriented model for driving adaptation in an Information System (IS) having as primary goal Energy Efficiency (EE) and Quality of Service (QoS) constraints satisfaction. Doing so, we have been able to give some answers to the research challenges that we wanted to investigate.

The first challenge was related to the analysis of the application level impact over efficiency. We have shown that the application level can have a relevant impact on the EE of a Service Oriented Architecture (SOA) by also analyzing some results available in the state of the art.

About the second challenge, we have been able to select a set of metrics

that can be used to assess the system state, both in terms of EE and QoS, and we have analyzed direct and indirect relations among these metrics. We have also considered several granularity levels for the selected metrics, from the infrastructure to the software. These metrics have shown to be able to provide useful information and have been used for deciding when the system needs to be repaired. Correlations among the selected indicators have been analyzed and resulted in the definition of a Bayesian Network (BN) model. The model is able to predict mutual influences between the variables of the system in a given context. We tested results with two different configurations of a data center and we considered several values for the parameters of the learning algorithm.

Challenge three faced the issue of discovering an adaptive approach for maintaining and improving EE. We have answered to this challenge by proposing our goal-oriented model, and by defining and testing a set of repair actions that can be automatically selected using the Adaptive Action Selection (AAS) algorithm. The algorithm has been used for both learning the effect of actions on the goals of our system and for suggesting the most likely effective action. Finally we simulated the enactment of repair actions for modeling the effect of actions over indicators and for building a probabilistic model able to choose a proper repair action from a pool. The action selection proved to be successful in improving the system state when actions are available for repairing the specific violated indicators.

Answering to the last challenge, we have proposed a simulation environment which models the main features of a data center and which can be used for simulating the outcomes of the application of repair actions in a safe environment. The simulation environment proved to be effective in modeling the system behavior under different workloads and with different configurations.

9.2 Generalization of the Approach

The proposed framework has been developed to face the problem of Green Information Technology (IT) and EE in ISs. This problem was the main motivation to our work. However, in the development of the solution, we have delivered a very general approach that can be used in different fields. The goal-oriented model proposed and the techniques employed for learning it, as well as the AAS algorithm are not strictly related to the Green IT field, but can be applied to other fields with some minor modifications. Generally speaking, the proposed approach is suitable for application in a

system showing the following features:

- goals of the system can be represented using a set of measurable variables that can be evaluated in time and compared with some reference values;
- the system can be monitored and data can be used for further analysis;
- a set of actions has to be available so that it can be used to improve the system state;
- the execution of an action should have effects that are observable and measurable in order to assess the action outcome;
- the system runs in a dynamic environment where conditions can change in time as well as the actions effect over the system variables.

9.3 Lessons Learned

This thesis work was very interesting in order to understand the complexity of the considered environment.

Monitoring and assessing EE and QoS in data centers can be a complex task. The selection of an appropriate set of indicators is determinant for obtaining significant results and the large amount of metrics available makes the choice difficult. In the considered approach even indirect contributions have been considered and understanding these contributions is not trivial. Moreover, setting thresholds over indicators is a very delicate task: a wrong threshold can raise unmotivated alarms about the behavior of the system or can cause the inability of identifying suboptimal states. The Green Grid Data Center Maturity Model (DCMM) helped in this direction but still other metrics are strictly dependent on the specific context, service, or task and a general solution is not available. Another issue consists in the computation of the metrics. Data for the computation of some of them can be hard to collect. As an example, the problem of computing the energy consumption of a Virtual Machine (VM) or an application is still discussed in the state of the art. Even if some solutions are available, usually they use a high degree of approximation.

Another interesting aspect is the difficulty of obtaining real data and of being able to test solutions in real data centers. Most of the time data centers owners are not willing to give any kind of information about their structure and even more reticent in allowing the collection of monitoring data or the testing of repair actions that can impact on the configuration of

the system. This is why simulation tools are very useful to this purpose. We have proposed our own simulation tool, but other tools are available in the state of the art, each one with different strong points and drawbacks. Building the model was a useful exercise to understand more deeply the behavior of virtualization and of a SOA in general.

Even if a lot of effort has been made towards a greener asset of data centers, still it seams that most of the companies are not ready to sacrifice performance for saving energy. This is why we tried to consider both aspects in our model, ensuring that Service Level Agreements (SLAs) are satisfied. Nevertheless, the current trends seem to go towards a higher attention for greenness even in this fields and we expect that governmental regulations will force reticent data centers administrators to take this aspect into account.

9.4 Future Directions

During the development of this thesis, some interesting open issues have been discovered and should be considered in future work.

One of the open issues consists in the selection of the set of metrics. Deciding which metrics better represent a system is a topic that should be further investigated. It is also possible that this set can evolve in time or can be dependent from the context or from the specific system we are considering. For instance, in a transactional oriented service, the monitoring of IO operations is probably more important than Central Processing Unit (CPU) usage. Other considerations could also be made and an automatic way to select the best set of indicators for a given application or service is an interesting challenge.

Also the management of data collected by the monitoring system is an interesting topic. In a big system, the amount of data generated by the monitoring activity needs to be saved. Of course, the storage of these data consumes energy and resources that could be used in another way. How long data should be saved is a question that should be answered. Also, the sampling interval should be adaptive, monitoring more frequently when the monitored feature is changing and less frequently when it is stable. Also some compression techniques could be used in order to reduce the amount of the data to be maintained in the storage system.

From the learning perspective, some interesting considerations could be done on the relations in the model. Supported by an ontology, it could possible to generalize the relations among variables of the system, by looking

for similarities in the connections between elements belonging to the same categories. The same considerations can be done for the action-goal relations: generalization can be used to generalize the effect of an action to a category, instead than to a single indicator, by considering similar patterns for several parameters. The result would be a general action, having variables instead of parameters belonging to a given category in the ontology. This mechanism could be useful especially in the first steps of the learning approach or in big systems by avoiding the need of testing each action with each possible parameter to have a comprehensive knowledge about its effect.

The same approach proposed in this thesis could be adopted in a federated cloud environment. In this case, other variables should be considered, especially when deploying new VMs or when migrating one machine from a site to another. The network cost can not be neglected in clouds, or at least further studies are needed to investigate its impact. Moreover, the position of the different sites should be considered and the cost of energy could depend on the time of the day. Also global greenhouse gas (GHG) emissions can be an interesting aspect in such a system and can be different from site to site. All these considerations should be included in the adaptive model for performing an effective choice.

List of Figures

List of Tables

List of Acronyms

A

AAS Adaptive Action Selection 8, 9, 137–140, 149, 155, 158, 159, 161, 162, 186

AOS Adaptive Operator Selection 8, 11, 59, 72–76, 137–140, 162

B

BIC Bayesian Information Criterion 129, 130

BN Bayesian Network 8, 10–12, 59, 64–72, 76, 119–123, 126, 128, 131–135, 151, 186

BNET Bayes Net Toolbox 133

BP Business Process 18, 22, 25, 26, 38–42, 44, 52, 53, 59, 61, 80, 110, 111, 114, 165–167, 169, 170, 182, 183

BPaaS Business Process as a Service 31

BPM Business Process Management 22

C

CPT Conditional Probability Table 66–68, 70, 72, 120

CPU Central Processing Unit 25, 30–32, 35, 46–50, 84, 87, 91, 92, 95, 104, 105, 107, 108, 110, 114, 141, 151–154, 158–160, 169–181, 188

CRS	Customer Relationship Management 33
CSOP	Constraint Satisfaction Optimization Problem 42
CUE	Carbon Usage Effectiveness 34

D

DAG	Directed Acyclic Graph 65, 68–71, 122, 128–130
DBMS	Database Management System 33
DCiE	Data Center Infrastructure Efficiency 90, 168
DCMM	Data Center Maturity Model 8, 98–100, 105, 112, 187

E

EE	Energy Efficiency I, II, 5–7, 9–11, 17–23, 25–29, 32–34, 36–40, 42, 43, 45–47, 50–55, 59, 76, 79–84, 86, 87, 89, 90, 92–96, 98–102, 107, 115, 120, 135, 165–170, 182, 183, 185–187
EPA	Environmental Protection Agency 20, 23, 34, 82, 83
ERF	Energy Reuse Factor 34
ERP	Enterprise Resource Planning 33

F

FLOPS	floating point operations per second 89, 95

G

GEC	Green Energy Coefficient 34
GHG	global greenhouse gas 23, 34, 36, 189
GIPC	Green IT Promotion Council 34
GPI	Green Performance Indicator 34, 35, 39, 40, 79, 80, 82, 85–88, 91–93, 96, 170
GPM	Gallon per 100 Miles 36
GRL	Goal Requirement Language 43, 44

H

HDD	Hard Disk Drive 87

I

ICT Information and Communications Technology 23, 26, 27, 51

IS Information System 5–8, 10–12, 17–20, 22–24, 29, 37, 38, 52, 53, 57–59, 62, 76, 97, 115, 185, 186

IT Information Technology I, 3–5, 9–11, 17–29, 32–34, 43, 45, 52, 53, 82, 83, 88, 90, 95, 99, 105, 186

IVI Innovation Value Institute 26

K

KEI Key Ecological Indicator 34, 35, 39

KPI Key Performance Indicator 34, 35, 39–41, 64, 82, 86, 88, 170

M

MABS Multi Armed Bandit Selection 140

MAP Maximum a Priori Estimation 132

MAPE Monitoring - Analyzing - Planning - Executing 48

METI Ministry of Economy, Trade and Industry 34

MILP Mixed Integer Linear Programming 61

MIPS Million Instructions Per Second 50, 168

MIS Management Information Systems 33

MMHC Max-Min Hill Climbing Algorithm 122, 123, 128

MMPC Max-Min Parents and Children Algorithm 122

$MMTCO_2$ million metric tons of CO_2 23

MPG Miles per Gallon 36

O

OECD Organization for Economic Co-operation and Development 20

OS Operating System 9

P

PDC	Popular Data Concentration 52
PMC	Performance Monitoring Counter 31, 32
PUE	Power Usage Effectiveness 33–35, 83, 90, 99, 168

Q

QoS	Quality of Service II, 8–12, 39, 42, 50, 51, 55, 59–61, 63, 64, 76, 81, 82, 84, 86, 88, 90, 95, 101, 113, 120, 165–167, 170, 182, 183, 185–187

S

SaaS	Software as a Service 31
SLA	Service Level Agreement I, 22, 45–47, 49, 50, 55, 60, 63, 86, 98, 104, 120, 168, 188
SNIA	Storage Networking Industry Association 83
SOA	Service Oriented Architecture 6, 9, 11, 59, 60, 62, 76, 82, 86, 101, 137, 138, 166, 185, 188
SPC	Storage Performance Council 83
SPEC	Standard Performance Evaluation Corporation 29, 84
STaaS	Storage as a Service 31

T

TCO	Total Cost of Ownership 105
TPPC	Transaction Processing Performance Council 29

V

VM	Virtual Machine 9, 10, 30–32, 45–51, 80, 95, 107–114, 124, 131, 134, 135, 141, 151–155, 157–162, 167–182, 187, 189

Bibliography

[1] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[2] OECD, "Towards Green ICT Strategies: Assessing Policies and Programmes on ICT and the Environment," tech. rep., June 2009.

[3] EPA, "ENERGY STAR® Program Requirements for Computer Servers," tech. rep., 2007.

[4] L. Newcombe, M. Acton, J. Booth, S. Flucker, P. Latham, S. Strutt, and R. Tozer, "2012 Best Practices for the EU Code of Conduct on Data Centres," tech. rep., European Commision, 2011.

[5] A. Molla, V. Cooper, and S. Pittayachawan, "The Green IT Readiness (G-Readiness) of Organizations: An Exploratory Analysis of a Construct and Instrument," *Communications of the Association for Information Systems*, vol. 29, no. 1, pp. 66–96, 2011.

[6] S. Murugesan, "Harnessing Green IT: Principles and Practices," *IEEE IT Professional*, vol. 10, no. 1, pp. 24–33, 2008.

[7] R. T. Watson, M. C. Boudreau, and A. J. Chen, "Information Systems and Environmentally Sustainable Development: Energy Informatics and New Directions for the IS Community," *Management Information Systems Quarterly*, vol. 34, no. 1, pp. 23–38, 2010.

[8] S. Seidel, J. C. Recker, C. Pimmer, and J. Vom Brocke, "Enablers and barriers to the organizational adoption of sustainable business practices," in *Proceedings of the 16th Americas Conference on Information Systems: Sustainable IT Collaboration around the Globe*, 2010.

[9] J. Vom Brocke and S. Seidel, *Green Business Process Management: Towards the Sustainable Enterprise*. Springer, 2012.

[10] B. Pernici, M. Aiello, J. Vom Brocke, B. Donnellan, E. Gelenbe, and M. Kretsis, "What IS can do for environmental sustainability: a report from CAiSE'11 panel on green and sustainable IS," *Communications of the Association for Information Systems*, vol. 30, no. 1, 2012.

[11] C. Houy, M. Reiter, P. Fettke, P. Loos, K. Hoesch-Klohe, and A. Ghose, "Advancing Business Process Technology for Humanity: Opportunities and Challenges of Green BPM for Sustainable Business Activities," in *Green Business Process Management*, pp. 75–92, Springer, 2012.

[12] G. Cook, "How clean is your cloud," tech. rep., Greenpeace International, April 2012.

[13] The Green Grid Consortium, "Data Center Maturity Model," *White Paper*, 2011.

[14] B. Donnellan, C. Sheridan, and E. Curry, "A capability maturity framework for sustainable information and communication technology," *IEEE IT Professional*, vol. 13, no. 1, pp. 33–40, 2011.

[15] N.-H. Schmidt and L. Kolbe, "Towards a contingency model for green it governance," *ECIS 2011 Proceedings*, 2011.

[16] S. Rivoire, M. A. Shah, P. Ranganatban, C. Kozyrakis, and J. Meza, "Models and metrics to enable energy-efficiency optimizations," *Computer*, vol. 40, no. 12, pp. 39–48, 2007.

[17] SPEC. http://www.spec.org/benchmarks.html.

[18] TPC, "TPC-Energy Specification - Standard Specification Version 1.2.0," tech. rep., 2010.

[19] Y. Israel and T. Leitert, "The GreenIT DC-benchmarking tool: from scientific theory to real life," in *Energy Efficient Data Centers*, pp. 47–53, Springer, 2012.

[20] M. Poess and R. O. Nambiar, "Energy cost, the key challenge of today's data centers: a power consumption analysis of TPC-C results," *Proceedings of the VLDB Endowment*, vol. 1, no. 2, pp. 1229–1240, 2008.

[21] T. Do, S. Rawshdeh, and W. Shi, "pTop: A Process-level Power Profiling Tool," in *Workshop on Power Aware Computing and Systems (HotPower'09)*, 2009.

[22] A. Kansal and F. Zhao, "Fine-grained energy profiling for power-aware application design," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 2, pp. 26–31, 2008.

[23] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzalez, X. Martorell, J. Torres, and E. Ayguade, "Accurate Energy Accounting for Shared Virtualized Environments using PMC-based Power Modeling Techniques," in *International Conference on Grid Computing*, 2010.

[24] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*, pp. 39–50, 2010.

[25] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: Balancing energy in processing, storage, and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2011.

[26] E. Capra, C. Francalanci, and S. Slaughter, "Measuring Application Software Energy Efficiency," *IEEE IT Professional*, vol. 14, no. 2, pp. 54–61, 2012.

[27] M. Sabharwal, A. Agrawal, and G. Metri, "Enabling green it through energy-aware software," *IT Professional*, vol. 15, no. 1, pp. 19–27, 2013.

[28] I. Jean-Marc Pierson *et al.*, "Green task allocation: Taking into account the ecological impact of task allocation in clusters and clouds," *Journal of Green Engineering*, vol. 1, no. 02, p. 11761, 2011.

[29] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew, "Geographical load balancing with renewables," *In Proc. of Sigmetrics 2011*, 2011.

[30] Data Centre Metrics Coordination Taskforce, "Harmonizing Global Metrics For Data Center Energy Efficiency," *Whitepaper*.

[31] A. Kipp, T. Jiang, M. Fugini, and I. Salomie, "Layered green performance indicators," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 478–489, 2012.

[32] A. Nowak, F. Leymann, D. Schumm, and B. Wetzstein, "An architecture and methodology for a four-phased approach to green business process reengineering," in *Proceedings of the 1st International Conference on ICT as Key Technology for the Fight against Global Warming, ICT-GLOW 2011, August 29 - September 2, 2011, Toulouse, France*, vol. 6868 of *Lecture Notes in Computer Science (LNCS)*, pp. 150–164, Springer-Verlag, 2011.

[33] C. Cappiello, M. G. Fugini, A. M. Ferreira, P. Plebani, and M. Vitali, "Business Process Co-Design for Energy-Aware Adaptation," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pp. 463–470, 2011.

[34] R. P. Larrick and K. W. Cameron, "Consumption-Based Metrics: From Autos to IT," *Computer*, pp. 97–99, 2011.

[35] D. Chen, E. Henis, C. Cappiello, *et al.*, "Usage centric green performance indicators," in *Proceedings of the Green Metrics 2011 Workshop (in conjunction with ACM SIGMETRICS 2011)*, 2011.

[36] S. Seidel, J. Vom Brocke, and J. C. Recker, "Call for Action: Investigating the Role of Business Process Management in Green IS," *Sprouts: Working Papers on Information Systems*, vol. 11, no. 4, 2011.

[37] A. Nowak, F. Leymann, D. Schleicher, D. Schumm, and S. Wagner, "Green Business Process Patterns," in *Proceedings of the 18th Conference on Pattern Languages of Programs*, ACM, 2011.

[38] A. Nowak, T. Binz, C. Fehling, O. Kopp, F. Leymann, and S. Wagner, "Pattern-driven green adaptation of process-based applications and their runtime infrastructure," *Computing*, vol. 94(6), pp. 463–487, 2012.

[39] K. Hoesch-Klohe and A. Ghose, "Carbon-Aware Business Process Design in Abnoba," *Service-Oriented Computing*, pp. 551–556, 2010.

[40] A. Mello Ferreira, K. Kritikos, and B. Pernici, "Energy-Aware Design of Service-Based Applications," *Service-Oriented Computing*, pp. 99–114, 2009.

[41] A. De Oliveira, G. Frederico, and T. Ledoux, "Self-optimisation of the energy footprint in service-oriented architectures," *Proceedings of the 1st Workshop on Green Computing (GCM'10)*, 2010.

[42] H. Zhang, L. Liu, and T. Li, "Designing IT systems according to environmental settings: A strategic analysis framework," *The Journal of Strategic Information Systems*, 2011.

[43] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, "Automatic virtual machine configuration for database workloads," *ACM Transactions on Database Systems (TODS)*, vol. 35, no. 1, 2010.

[44] D. Borgetto, H. Casanova, G. Da Costa, and J.-M. Pierson, "Energy-aware service allocation," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 769–779, 2012.

[45] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-aware autonomic resource allocation in multitier virtualized environments," *Services Computing, IEEE Transactions on*, vol. 5, no. 1, pp. 2–19, 2012.

[46] G. von Laszewski, L. Wang, A. J. Younge, and X. He, "Power-aware scheduling of virtual machines in dvfs-enabled clusters," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pp. 1–10, 2009.

[47] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*, pp. 327–334, 2009.

[48] Y. Wang, X. Wang, M. Chen, and X. Zhu, "Power-efficient response time guarantees for virtualized enterprise servers," in *Real-Time Systems Symposium, 2008*, pp. 303–312, 2008.

[49] V. Petrucci, O. Loques, B. Niteroi, and D. Mossé, "Dynamic configuration support for power-aware virtualized server clusters," in *WiP Session of the 21th Euromicro Conference on Real-Time Systems. Dublin, Ireland*, 2009.

[50] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pp. 215–224, 2010.

[51] J. Rao, X. Bu, C. Z. Xu, L. Wang, and G. Yin, "Vconf: a reinforcement learning approach to virtual machines auto-configuration," in *Proceedings of the 6th international conference on Autonomic computing*, pp. 137–146, 2009.

[52] I. Anghel, T. Cioara, I. Salomie, G. Copil, D. Moldovan, and C. Pop, "Dynamic frequency scaling algorithms for improving the CPU's energy efficiency," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pp. 485–491, 2011.

[53] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "Enacloud: an energy-saving application live placement approach for cloud computing environments," in *Cloud Computing, 2009. CLOUD'09. IEEE International Conference on*, pp. 17–24, 2009.

[54] D. Borgetto, M. Maurer, G. Da-Costa, J.-M. Pierson, and I. Brandic, "Energy-efficient and SLA-aware management of IaaS clouds," in *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, p. 25, 2012.

[55] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[56] G. Katsaros, J. Subirats, J. Oriol Fitó, J. Guitart, P. Gilet, and D. Espling, "A service framework for energy-aware monitoring and VM management in Clouds," *Future Generation Computer Systems*, 2012.

[57] G. Andersson, M. D. Ilic, V. Madani, and D. Novosel, "Network systems engineering for meeting the energy and environmental dream," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 7–14, 2011.

[58] E. Gelenbe and C. Morfopoulou, "A Framework for Energy-Aware Routing in Packet Networks," *The Computer Journal*, vol. 54, no. 6, pp. 850–859, 2011.

[59] N. S. Chauhan and A. Saxena, "A green software development life cycle for cloud computing," *IT Professional*, vol. 15, no. 1, pp. 28–34, 2013.

[60] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: helping disk arrays sleep through the winter," in *ACM SIGOPS Operating Systems Review*, vol. 39, pp. 177–190, 2005.

[61] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *Proceedings of the 18th annual international conference on Supercomputing, June*, 2004.

[62] C. Cappiello, A. Hinostroza, B. Pernici, M. Sami, E. Henis, R. Kat, K. Meth, and M. Mura, "Adsc: Application-driven storage control for energy efficiency," in *Information and Communication on Technology for the Fight against Global Warming* (D. Kranzlmüller and A. Toja, eds.), pp. 165–179, Springer, 2011.

[63] M. Vitali and B. Pernici, "A Survey on Energy Efficiency in Information Systems," *International Journal of Cooperative Information Systems (IJCIS), IN PRESS*, 2014.

[64] S. Benbernou, I. Brandic, C. Cappiello, M. Carro, M. Comuzzi, A. Kertész, K. Kritikos, M. Parkin, B. Pernici, and P. Plebani, "A survey on service quality description," *ACM Computing Surveys, in press*, 2014.

[65] C. Patel, K. Supekar, and Y. Lee, "A QoS oriented framework for adaptive management of web service based workflows," in *Database and Expert Systems Applications*, pp. 826–835, 2003.

[66] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *Software Engineering, IEEE Transactions on*, vol. 33, no. 6, pp. 369–384, 2007.

[67] D. Ardagna, M. Comuzzi, E. Mussi, B. Pernici, and P. Plebani, "PAWS: A framework for executing adaptive web-service processes," *IEEE software*, vol. 24, no. 6, pp. 39–46, 2007.

[68] C. Cappiello, M. Comuzzi, E. Mussi, and B. Pernici, "Context management for adaptive information systems," *Electronic Notes in Theoretical Computer Science*, vol. 146, no. 1, pp. 69–84, 2006.

[69] B. Pernici and A. M. Rosati, "Automatic learning of repair strategies for web services," in *Web Services, 2007. ECOWS'07. Fifth European Conference on*, pp. 119–128, 2007.

[70] B. Pernici and S. H. Siadat, "A fuzzy service adaptation based on qos satisfaction," in *Advanced Information Systems Engineering*, pp. 48–61, 2011.

[71] R. Kazhamiakin, B. Wetzstein, D. Karastoyanova, M. Pistore, and F. Leymann, "Adaptation of service-based applications based on process quality factor analysis," in *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops*, pp. 395–404, 2010.

[72] J. Pearl, *Bayesian Networks: a model of self-activated: memory for evidential reasoning*. Computer Science Department, University of California, 1985.

[73] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausble Inference*. Morgan Kaufmann Pub, 1988.

[74] R. E. Neapolitan, "Probabilistic reasoning in expert systems: Theory and algorithms.," *JOHN WILEY & SONS, INC., P. O. BOX 6792, SOMERSET, NJ 08875-9976(USA), 1989, 448*, 1989.

[75] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Englewood Cliffs, 2010.

[76] R. E. Neapolitan, *Learning bayesian networks*. Pearson Prentice Hall Upper Saddle River, 2004.

[77] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine learning*, vol. 9, no. 4, pp. 309–347, 1992.

[78] A. M. Carvalho, "Scoring functions for learning bayesian networks," *Inesc-id Tec. Rep*, 2009.

[79] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, 2010.

[80] A. Fialho, *Adaptive operator selection for optimization*. PhD thesis, Ecole Doctorale d'Informatique, Université Paris-Sud, Paris, 2010.

[81] Y. Y. Wong, K. H. Lee, K. S. Leung, and C. W. Ho, "A novel approach in parameter adaptation and diversity maintenance for genetic algorithms," *Soft Computing*, vol. 7, no. 8, pp. 506–515, 2003.

[82] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm," in *Evolutionary Computation, 1997., IEEE International Conference on*, pp. 121–125, 1997.

[83] J. Niehaus and W. Banzhaf, "Adaption of operator probabilities in genetic programming," in *Genetic Programming*, pp. 325–336, Springer, 2001.

[84] J. Maturana and F. Saubion, "A compass to guide genetic algorithms," in *Parallel Problem Solving from Nature–PPSN X*, pp. 256–265, Springer, 2008.

[85] J. Maturana and F. Saubion, "From parameter control to search control: Parameter control abstraction in evolutionary algorithms," *Constraint Programming Letters*, vol. 4, no. 1, pp. 39–65, 2008.

[86] J. Maturana, F. Lardeux, and F. Saubion, "Autonomous operator management for evolutionary algorithms," *Journal of Heuristics*, vol. 16, no. 6, pp. 881–909, 2010.

[87] M. Giger, D. Keller, and P. Ermanni, "Aorcea–an adaptive operator rate controlled evolutionary algorithm," *Computers & Structures*, vol. 85, no. 19, pp. 1547–1561, 2007.

[88] D. E. Goldberg, "Probability matching, the magnitude of reinforcement, and classifier system bidding," *Machine Learning*, vol. 5, no. 4, pp. 407–425, 1990.

[89] J. Radatz, A. Geraci, and F. Katki, "Ieee standard glossary of software engineering terminology," *IEEE Std*, vol. 610121990, p. 121990, 1990.

[90] D. Chen, E. Henis, R. I. Kat, D. Sotnikov, C. Cappiello, A. M. Ferreira, B. Pernici, M. Vitali, T. Jiang, J. Liu, and A. Kipp, "Usage centric green performance indicators," *SIGMETRICS Performance Evaluation Review*, vol. 39, pp. 92–96, dec 2011.

[91] EPA, "Report to Congress on Server and Data Center Energy Efficiency," 2007. `http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf`.

[92] EPA, "ENERGY STAR® Program Requirements for Computer Servers," 2009. `http://www.energystar.gov/ia/partners/product_specs/program_reqs/computer_server_prog_req.pdf`.

[93] EPA, "ENERGY STAR® Program Requirements for Data Center Storage ," tech. rep., 2010.

[94] EPA, "ENERGY STAR® Program Requirements for Uninterruptible Power Supplies (UPSs)," tech. rep., 2012.

[95] A. Rawson, J. Pflueger, and T. Cader, "The green grid data center power efficiency metrics: Pue and dcie," 2007.

[96] J. Haas, M. Monroe, *et al.*, "Proxy Proposals for Measuring Data Center Productivity," 2009.

[97] SNIA, "SNIA Green Storage Initiative (GSI)," 2011. `http://www.snia.org/forums/green/`.

[98] Storage Networking Industry Association (SNIA) Emerald. `http://sniaemerald.com/`.

[99] SNIA, "Storage Power Efficiency Measurement Specification." `http://www.snia.org/sites/default/files/Storage_Power_Efficiency_Measurement_Spec_v0.2.10_DRAFT.pdf`.

[100] Storage Performance Council (SPC). `http://www.storageperformance.org`.

[101] SPEC, "SPEC - Standard Performance Evaluation Corporation." `http://www.spec.org`.

[102] SPEC, "SPEC Power and Performance Benchmark Methodology," 2011. `http://www.spec.org/power/docs/SPECpower-Power_and_Performance_Methodology.pdf`.

[103] GAMES, "Green Active Management of Energy in IT Service Centres." http://www.green-datacenters.eu/, 2010-2012.

[104] G. Schulz, *The Green and Virtual Data Centre*. CRC/Auerbach Publications, 2009.

[105] K. Asanovic, R. Bodik, B. C. Catanzaro, *et al.*, "The landscape of parallel computing research: a view from Berkeley," Tech. Rep. UCB/EECS-2006-183, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2006. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf`.

[106] C. Belady, A. Rawson, J. Pfleuger, and T. Cader, "Green grid data center power efficiency metrics: PUE and DCiE," *the green grid*, pp. 1–9, 2008.

[107] A. Mani and A. Nagarajan, "Understanding quality of service for Web services," 2002. `http://www.ibm.com/developerworks/webservices/library/ws-quality/index.html`.

[108] G. von Bochmann, B. Kerherve, *et al.*, "Introducing QoS to Electronic Commerce Applications," in *Proceedings of the Second International Symposium on Topics in Electronic Commerce*, 2001.

[109] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q. Sheng, "Quality driven web services composition," in *Proceedings of the 12th World wide Web Conference (WWW)*, 2003.

[110] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[111] J. R. Stanley, K. G. Brill, and J. Koomey, "Four metrics define data center "greenness"," tech. rep., 2007. `http://uptimeinstitute.org/wp_pdf/(TUI3009F)FourMetricsDefineDataCenter.pdf`.

[112] B. Pernici, C. Cappiello, M. G. Fugini, P. Plebani, M. Vitali, I. Salomie, T. Cioara, I. Anghel, E. Henis, R. Kat, *et al.*, "Setting energy efficiency goals in data centers: the games approach," in *Energy Efficient Data Centers*, pp. 1–12, Springer, 2012.

[113] Y. Asnar, P. Giorgini, and J. Mylopoulos, "Goal-driven risk assessment in requirements engineering," *Requirements Engineering*, vol. 16, no. 2, pp. 101–116, 2011.

[114] A. Mello Ferreira, *Energy-aware service-based Information Systems*. PhD thesis, Politecnico di Milano, Italy, 2013.

[115] A. M. Ferreira and B. Pernici, "Using intelligent agents to discover energy saving opportunities within data centers," 2013.

[116] C. Cappiello, P. Plebani, and M. Vitali, "Energy-Aware Process Design Optimization," in *Proceedings of the 3rd International Conference on Cloud and Green Computing*, 2013.

[117] Y. Wu and M. Zhao, "Performance modeling of virtual machine live migration," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pp. 492–499, 2011.

[118] D. R. Llanos, "Tpcc-uva: an open-source tpc-c implementation for global performance measurement of computer systems," *ACM SIGMOD Record*, vol. 35, no. 4, pp. 6–15, 2006.

[119] A. Kipp, T. Jiang, J. Liu, M. Fugini, M. Vitali, B. Pernici, and I. Salomie, "Applying green metrics to optimise the energy consumption footprint of it service centres," *International Journal of Space-Based and Situated Computing*, vol. 2, no. 3, pp. 158–174, 2012.

[120] R. Opgen-Rhein and K. Strimmer, "From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data," *BMC Systems Biology*, vol. 1, no. 1, p. 37, 2007.

[121] N. Friedman, I. Nachman, and D. Peér, "Learning bayesian network structure from massive datasets: the Sparse Candidate algorithm," in *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pp. 206–215, 1999.

[122] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[123] MATLAB, *version 7.14.0 (R2012a)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[124] K. Murphy, "Bayes Net Toolbox for Matlab." https://code.google.com/p/bnt/, 2007.

[125] M. Vitali, "StructureLearning Tool." https://github.com/monicavit164/StructureLearning, 2013.

[126] M. Gagliolo and J. Schmidhuber, "Algorithm selection as a bandit problem with unbounded losses," in *Learning and Intelligent Optimization*, pp. 82–96, Springer, 2010.

[127] M. Pacula, J. Ansel, S. Amarasinghe, and U. M. O'Reilly, "Hyperparameter tuning in bandit-based adaptive operator selection," *Applications of Evolutionary Computation*, pp. 73–82, 2012.

[128] M. Vitali, "ActionSelection Tool." https://github.com/monicavit164/ActionSelection, 2013.

[129] M. Vitali, U.-M. O'Reilly, and K. Veeramachaneni, "Modeling Service Execution on Data Centers for Energy Efficiency and Quality of Service Monitoring ," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC2013)*, 2013.

[130] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[131] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, pp. 1–21, 2010.

[132] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 2, pp. 83–94, 2000.

[133] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, "The design and use of simplepower: a cycle-accurate energy estimation tool," in *Proceedings of the 37th Annual Design Automation Conference*, pp. 340–345, 2000.

[134] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, and M. Kandemir, "Using complete machine simulation for software power estimation: The softwatt approach," in *High-Performance Computer Architecture, 2002. Proceedings. Eighth International Symposium on*, pp. 141–150, 2002.

[135] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *In Proceedings of Workshop on Modeling, Benchmarking, and Simulation*, pp. 70–77, 2006.

[136] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *Proceedings of the 14th international conference on Architectural support for programming languages and operating systems*, ASPLOS XIV, (New York, NY, USA), pp. 205–216, ACM, 2009.

[137] M. Pedram and I. Hwang, "Power and Performance Modeling in a Virtualized Server System," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pp. 520–526, 2010.

[138] M. Vitali, "dcSimulation Tool." https://github.com/monicavit164/dcSimulation, 2013.