FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Cost-based Decision Making in Cloud Environments using Bayesian Networks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Dmytro Grygorenko
Matrikelnummer 1129105

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Priv.-Doz. Dr. Ivona Brandic
Mitwirkung: MSc Soodeh Farokhi

Wien, 21.08.2014

(Unterschrift Verfasserin) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Cost-based Decision Making in Cloud Environments using Bayesian Networks

MASTER THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Internet Computing

by

Dmytro Grygorenko
Registration Number 1129105

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Dr. Ivona Brandic
Assistance: MSc Soodeh Farokhi

Vienna, 21.08.2014

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Dmytro Grygorenko
Karlsplatz 13, 1040 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Wien, Datum) (Dmytro Grygorenko)

Acknowledgements

First of all I would like to express my deep gratitude to my master thesis supervisor, Priv.-Doz. Dr. Ivona Brandic. I have received a lot of useful knowledge during her lections. I appreciate her contributions of time, commitments, scientific advices, insightful discussions about the research. I am also grateful to my co-advisor MSc Soodeh Farokhi for providing useful suggestions about this thesis, insightful comments and constructive criticisms at different stages of my research. They are hard-working people and I believe their academic achievements will continue to increase.

I would like to express my special appreciation and gratitude Univ.Prof.Dr. Full Professor Schahram Dustdar, who has been a tremendous mentor for me. His expertise, knowledges, and provision of the font materials in the study of Clouds added considerably to my graduate experience. I would especially like to thank Philipp Leitner, Drazen Lucanin and whole Distributed Systems Group.

I would like to thank Renate Weiss for academic and technical support during master thesis preparation.

A very special thanks goes out to Univ.Prof. Dipl.-Ing. Dr.techn. Hannes Werthner, without whose motivation and advices I would not have considered a magistracy in Technical University of Vienna.

I owe more thanks to my friend and wife, Kateryna and my mother Svitlana Grygorenko for their support during the final stages of this master thesis and encouragement through education process. Completing this work would have been all the more difficult without their patient and faith.

Abstract

Cloud computing providers experienced fast growing trend in recent years by offering highly available and scalable services. However, increasing customer resource demand and competitive market force the providers to enlarge their data centers that leads to huge power consumption or apply more economical resource provision plans that degrades quality of service. Hence, the cloud providers need the solutions to decrease energy costs and improve resource provisioning.

In this thesis, we propose a new approach for a cost-aware cloud power management that enables effective placement of customer's virtual machines (VMs) in the cloud data centers. This approach consists of two phases. First, we create a model of the cloud infrastructure using Bayesian Networks (BNs). BNs are graphical models that represent variables of interest and probabilistic dependencies among them. They allow to apply knowledge about domain in order to find hidden and causal relationships between different parts of cloud infrastructure. Second, we make decisions about effective VM placement in order to save cloud provider costs. On this step Multi-criteria decision aid (MCDA) is applied. This technique helps to quantitatively measure the benefit of a certain decision. It implies the usage of utility function that is calculated based on ranking set of reasonable factors and criteria defined by us.

The related work that has been done in this field of study does not cover all aspects of the problem. In particular, our study focuses more concretely on the geo-distributed data centers experienced frequent power outages, operating in different time zones and in constantly changing outdoor temperatures.

Additionally, we developed a simulation toolkit and used it to validate our algorithm by conducting a performance evaluation. The results of experiments proved good performance and applicability of the proposed model and its high potential to operate in various real-world scenarios.

Kurzfassung

Die Anbieter von Cloud-Computing haben in den letzten Jahren durch das Angebot von hochverfügbaren und skalierbaren Services ein schnelles Wachstum erfahren. Dennoch zwingen die steigende Nachfrage der Kunden nach Ressourcen und der vom Wettbewerb bestimmte Markt die Anbieter dazu, ihre Datenzentren zu erweitern, was einen enormen Stromverbrauch und den Einsatz von wirtschaftlicheren Plänen für die Bereitstellung von Ressourcen zur Folge hat, worunter die Qualität der Services leidet. Daher benötigen die Cloud-Anbieter Lösungen für die Senkung von Energiekosten und eine verbesserte Bereitstellung von Ressourcen.

In dieser Arbeit stellen wir einen neuen Ansatz für ein kostenbewusstes Cloud- Energiemanagement vor, das die effektive Platzierung der virtuellen Maschinen (VMs) von Kunden in Cloud-Datenzentren ermöglicht. Dieser Ansatz besteht aus zwei Phasen: Zunächst erschaffen wir ein Model der Cloud-Infrastruktur, das Bayessche Netze (BNs) nutzt. BN sind graphische Modelle, die für relevante Variablen und deren probabilistische Abhängigkeiten stehen. Sie erlauben die Anwendung von Wissen über Domains, um versteckte und kausale Zusammenhänge zwischen unterschiedlichen Teilen der Cloud-Infrastruktur zu finden. Zweitens treffen wir Entscheidungen über eine effektive Platzierung der VMs, um Kosten des Cloud-Anbieters zu senken. Hier wenden wir Multikritierien-Entscheidungshilfen (MCDA) an. Dieses Verfahren hilft dabei, den Nutzen einer bestimmten Entscheidung quantitativ zu messen. Sie impliziert die Nutzung von Nutzenfunktionen, die auf Basis einer durch uns definierten Rangfolge von angemessenen Faktoren und Kriterien berechnet werden.

Die relevante Forschung, die auf diesem Studiengebiet bereits getätigt wurde, deckt nicht alle Aspekte des Problems ab. Unsere Studie richtet den Blick insbesondere konkreter auf geoverteilte Datenzentren, die häufig von Stromausfällen betroffen sind und in unterschiedlichen Zeitzonen bei sich ständig ändernden Außentemperaturen betrieben werden. Zusätzlich dazu haben wir ein Simulations-Kit entwickelt, das wir für die Validierung unserer Algorithmen verwendet haben, indem wir eine Leistungsbewertung durchgeführt haben. Die Ergebnisse der Experimente beweisen eine gute Leistung und Anwendbarkeit des vorgeschlagenen Models und sein hohes Potential, in verschiedenen realen Szenarien betrieben zu werden.

Contents

List of Figures

List of Tables

Listings

CHAPTER 1

Introduction

In recent years we have seen a growing trend of lasting occurrence of cloud computing in our lives. Nowadays many companies from a wide range of branches of industry and science are looking towards a cloud to help simplify their workflows, increase flexibility and adaptability to deliver new and innovative ideas and solutions [14].

Multitude of IT vendors offer computational, storage, and application services and platforms. They try to provide coverage in several continents, ensuring quality of service for performance and uptime promises with *Service-Level Agreements* (SLAs) [60].

Cloud service models are varied by exposing resources and follow an "utility"pricing model where customers pay only for the utilized storage (so-called "pay-per-use"model), computational capacities, and transferred data [80].

Usage of this sophisticated piece of technology enables shared access for multiple users to available resources that allows IT vendors to receive a much better return on their investments, to achieve high availability and to scale easily [13, 39].

Cloud computing becomes more and more popular drastically increasing its power consumption. A report of Greenpeace [16] issues, Cloud Computing is already responsible for 1-2 percent of the world electricity use. On the other hand, it doesn't consider the hidden green benefits that arises due its usage. According to a report of AT & T, a company that moves its infrastructure into the Cloud reduces its energy consumption, lowers its carbon emissions and decreases its capital expenditure on IT resources [63]. In general, cloud infrastructure is often considered nowadays as a green IT approach that addresses two critical elements, energy and resource efficiency [65]. However, we have to agree with both sides of the discussion, cloud providers' energy usage plans are mostly not still optimal and have a lot of optimization opportunities [63].

The backbone of each cloud computing platform is an *Ultra-Large-Scale* (ULS) system [51], which complex, globally distributed infrastructure consists of heterogeneous sets of software and hardware nodes. It is important for IT vendors to have a deep knowledge of the system's performance behaviour in order to ensure that a ULS system can scale to handle increasing service demand.

Additionally, a cloud provider has to conceive a big amount of external factors such as geo-distributed infrastructure, weather impact, power outages, etc. Despite advances in infrastructure robustness, many cloud customers still face database, hardware, and software downtimes that can lead to loss of application service or data. Everyone would agree that today's data centers should never go down, and applications should be always available, end-users worldwide need to be able to believe on data center availability for critical data and application responsiveness at any time [9].

1.1 Problem definition

Taking into account all factors and responsibilities described above an efficient and reliable cloud model that not only address concerns of customers, but satisfies the vendor's business needs and requirements is very important in this case.

Main research goal of this thesis is to propose such model, namely to create and evaluate efficient solution for the management of physical infrastructure of a cloud that can simultaneously reduce operational costs satisfying the needs of a cloud vendor and increase *Quality of Service* (QoS) thereby addressing the concerns of customers. It should be mentioned that designated problem introduces necessity of multi-criteria decision making involving a big variety of factors, criteria, causality, uncertainty, and interdependence among variables.

1.2 Motivation

Although, there is large amount of work that has been done on this topic so far, an effective solution still stays an open question in the cloud computing community. There are different methodologies such as Machine Learning [70], Genetic Algorithms [66], Regression models [89], etc. which applications to the mentioned problem yield promising results. The problem of previously proposed solutions is that they do not take advantage of knowledge about observable cloud domain and thereby loose information important for the building of a good model. Most of research approaches addresses just a small part of defined problem, e.g., VM placement [56, 62, 73], temperature-aware energy usage [43], performance of VM migration [32], but none of them tries to investigate combination of all these processes, interconnections and dependencies between them into one model. Exactly this reason became a motivation to investigate the problem and to create own algorithm that would consider all disadvantages and weaknesses of previous solutions.

1.3 Contributions of the Thesis

Cost-aware cloud power management approach

In this thesis, we propose a new approach for a cost-aware cloud power management that enables effective placement of customer's virtual machines (VMs) in the cloud data centers. It is able to meet high cloud resource demand in order to reduce penalty costs. We focus more concretely on the placement and consolidation of virtual machines in geo-distributed data centers experienced

frequent power outages, operating in different time zones and in constantly changing temperatures. These aspects were not yet considered during investigation of the problem of cloud power management and therefore this study should become an important contribution for current cloud computing research.

Additionally, we utilize commonly used techniques for power saving, namely switching power of physical hosts in a data center on/off and *Dynamic Voltage and Frequency Scaling* (DVFS) [50]. DVFS allows to adjust the frequency of a microprocessor considering its CPU utilization and thereby to reduce power consumption.

Our approach consists of two phases. During the first phase, we create a model of the cloud infrastructure in the form of Bayesian Network (BN) [78]. BNs are graphical models that represent observable quantities, latent variables, unknown parameters and hypotheses, and edges are conditional dependencies between them. They have two important advantages over other research methods. First, they allow to apply knowledge about domain in order to find hidden and causal relationships between different parts of cloud infrastructure. Various observations about cloud infrastructure become an input for the designed BN. The BN will then calculate a possibilities of different criteria based on input values. Second, BNs are widely used for discovering relationships in raw data that are not clearly expressed with mathematical notations. Finally, they are based on a powerful mathematical background that is able to prove the correctness of built models.

During the second phase, we make decisions about effective VM placement in order to save cloud provider costs. On this step Multi-criteria decision aid (MCDA) [38] is applied. This technique helps to quantitatively measure the benefit of a certain decision. It implies the usage of utility function that is calculated based on ranking set of reasonable factors and criteria defined by us as follows: each criteria g_i is defined as an utility weighting u_i that represents the relative importance of each attribute for the given decision problem. The overall utility $U(a)$ of an action a is then simply the weighted sum $\sum u_i g_i(a)$.

Following MCDA technique, we defined different criteria that are important for a cloud provider in the context of the VM placement problem such as energy price in the data center area, possibility of a downtime of target physical machine, etc. The importance of criteria attributes was obtained from BN designed on the previous phase. Afterwards, we used a utility function in order to define the most suitable physical host for a given VM. Finally, a VM was allocated or migrated to a physical machine with the biggest value of the utility function.

Cloud Simulation framework

Ensuring of the performance and applicability of management models is a top priority for cloud computing vendors. However, due to the complexity of ULS systems, it is usually not possible to have a system running in a lab environment for performance testing. The quality and performance of the software often depends on the runtime environment. This fact introduces a risk of unexpected problems. A cloud vendor provides a certain QoS designated in SLA and promises to keep the service running at least, for example, 99,9% of the year, i.e., the service can still encounter a downtime up to 8..7 hours per year. Unfortunately, an unexpected crash happened in April, 2011 at Amazon AWS (Amazon Web Services) [82] due to operation mistakes in network reconfiguration. Due to this failure the performance of more than 70 organizations

significantly decreased, some sites were down for a dozens of hours. Such example clearly illustrates why currently a lot of organizations perform modelling and simulation in order to improve their systems, to understand how efficiently they can scale with the increasing demand of cloud capacity [51].

The applicability of the proposed models in the real-world applications cannot be done without evaluations and experiments.

There are four classes of experimental methodology depending on real or model application and environment: benchmarking, in-situ, emulation and simulation [42]. This thesis will use simulation as the main experimental approach for the following research.

Implementation, testing and execution of an application model involves the accomplishment of several conditions. All experimental results that are obtained from the abstract model should be reproducible on other platforms. The experimental use cases have to be executed under a representative set of parameters. In the case of this thesis it will be a set of generated data that provides the realistic and accurate measuring of data center workloads. The experimental model has to be extensible that supposes modelling and analysis of future results in an environment where the number of simulation instances (e.g., CPUs or VMs) increases. The last property that should be fulfilled is revisability. Each use case and developed hypothesis have to be designed in a way to explain and improve errors or unexpected results. For such purposes a simulation approach is necessary.

Though there are several cloud simulation frameworks [97] such as CloudSim, D-Cloud, PreFail, etc., none of them allows to simulate objectives and processes, e.g., VM migration between geo-distributed data centers, mentioned above.

The framework, called Cloudnet, proposed in the thesis, was designed considering the objectives and activities described above. The following key features listed below allow to simulate big variety of use cases as well as to apply the models intended by BDA analysis:

- *Simulation of cloud infrastructure*: execution of *physical machines*, allocations and migrations of VMs, etc., communication between different parts of the cloud computing environment, support of various resource utilization models, usage of real IT vendor specifications in order to obtain results as close as possible to realistic data.

- *Geo-distributed data centers*: performing simulations for data centers located in different parts of the globe that involves different energy price models, temperature data, cooling costs.

- *Management of cooling systems*: consideration of temperature-aware models that can be used in decision-making about data center workload management in order to reduce the overall energy consumption.

- *Use of weather data*: a possibility to either generate synthetic temperature data sets or use real data sets from various Web services.

- *Power outages*: as was noticed before power outages can cause unexpected changes in behaviour of hardware and software components therefore they should be modelled and studied in order to improve a fault-tolerant design that allows a system to continue its

intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails.

- *SLA-aware simulation*: escalation penalty costs can become a big problem for a cloud provider if the duties defined in the SLAs will not be fulfilled, hence, efficient management and sophisticated provisioning of cloud resources should be performed.

- *Prediction of resource usage*: large amount of collected statistics gives an opportunity to predict future workloads, to identify increasing and decreasing demand of resources and eventually to route user requests to the most appropriate endpoints in order to balance the load of the whole system.

The usage of Cloudnet framework is very similar to other simulation frameworks, especially to widely-used CloudSim [81]: firstly a developer configures a cloud infrastructure operating with familiar objects like data center, physical machines, etc. After that a generator of customer requests should be defined according to utilized cloud service model. Chapter 2 will give an overview of different service modes of cloud services. In particular, it should be mentioned that in comparison to other frameworks Cloudnet was designed for simulation of various service models. Eventually a developer has to set a cloud manager that triggers various change actions considering current state of the cloud. The responsibility of execution rests on a simulation engine that takes cloud configuration and iterates step by step until certain condition will be fulfilled.

1.4 Structure of the thesis

The Chapter 2 will give an overview of the state of art of in the field of Bayesian Networks and Decision Analysis. It will also introduce a basement of MCDA and explain the current state in the area of cloud computing architectures and patterns.

An overview of related work in the field of the costs-based decision-making for cloud computing as well as detail comparison of cloud simulation frameworks, their advantages and disadvantages will be given in the Chapter 3.

The Chapter 4 will introduce our methodology and the process of BN designing, will formally describe the problem of cost-aware cloud power management and, finally, will present our solution based on the previously designed BN.

The architecture and details of packages presented in the Cloudnet framework will be done in the Chapter 5. Additionally, we will shorty notice the examples of its usage.

Chapter 6 will present critical evaluations of the proposed approach in comparison to widely-used heuristics, will summarize results and shortly detail advantages and disadvantages of our solution. Execution of evaluations will be performed using Cloudnet framework.

CHAPTER 2

Materials and Methods

In this Chapter, the fundamentals required for the understanding of the following chapters will be introduced. It will give a short overview of BDA concepts, introduce a basement of MCDA and explain the current state in the area of cloud computing architectures and patterns.

2.1 Bayesian Networks

Definition

Bayesian Network (BN) is a graphical model that represents variables of interest (e.g., object features, event occurrences) and probabilistic dependencies among them via *Direct Acyclic Graph* (DAG) [69]. BN is also known as Belief Network or Bayesian Belief Network. It simulates the mechanism of exploring causal relations between key factors, and facilitates such models as prediction, abduction, cognitive activity (e.g., causal reasoning) [5].

Using BNs it is possible to discover causal structures with hidden variables from empirical data, calculate the effectiveness of interventions, like alternative management decisions (e.g., it is worthwhile to run an advertisement in order to increase sales of specific product), observe system changes, and generate future scenarios (e.g., predict weather time series data). BNs are widely used for discovering relationships in raw data that are not clearly expressed with mathematical notations (e.g., representations of a probabilistic relationship between diseases and patient symptoms). BNs describe data in both qualitative (graphical representation) and quantitative aspects (a number that represents the strength of the relation) [61].

BNs were developed to solve diverse problems of different data size and complexity with possibility of using incomplete data sets with high level of uncertainty [93] when correlation between input variables is not clearly observed.

BNs describe realistic results and are particularly useful for adapting analysis and decision-making process. In addition to their intuitively simple causal graphical structure, BNs can be extended and modified; they can be used for solving both discriminative tasks (classification) and regression problems (forecasting); they can incorporate missing data through the application of

Bayes theorem and use of complex learning algorithms; they can show high predictive accuracy on small sample sizes due to possibility to "smooth"models in a way to use all availability data for training purposes [76]; they can handle models with different scales and several type variables at the same time. BNs are widely used in medicine, document classification, bioinformatics, information retrieval, image processing and can be combined with decision analytic tools [11, 33, 48].

BN models are applied in fault diagnosis (NASA [71], Microsoft [20, 44]), pattern recognition (Standford University [40]), and medical diagnosis (Microsoft [64]).

In the background Bayesian Networks apply *Bayes theorem*, a simple mathematical formula for computing conditional probabilities.

For better understanding of the theorem we will recall the definitions of different probabilities.

Marginal probability is the probability of an event, ignoring any information about other events. The marginal probability of A is written $P(A)$.

Conditional probability is the probability of some event A, assuming event B. Conditional probability is written $P(A|B)$, and is read as "the probability of A, given B".

Joint probability is the probability of two events occurring together.

Bayes theorem relates the probability $P(A|B)$ of a hypothesis conditional on a given data set to the probability $P(B|A)$ of the data conditional on the hypothesis. The network is solved when Bayes' Rule is applied to each node of the graph:

$$P(A|B) = P(B|A)P(A)P(B) \tag{2.1}$$

where $P(A)$ is the prior distribution of parameter A; $P(A|B)$ is the posterior distribution, the probability of A given new data B; and $P(B|A)$ the likelihood function, the probability of B given existing data A [69].

Firstly, this theorem was mentioned in a masterwork "An Essay Toward Solving a Problem in the Doctrine of Chances" [12] of Thomas Bayes.

Bayes theorem is expressed in different forms that can be useful for various purposes. BNs use Bayes theorem in order to update the probabilities of system states in the respect to new conditions: reference to posteriori.

		System dynamics	Bayesian networks	Meta models	Coupled complex models	Agent based models	Expert systems
Model Purpose	Prediction		XXXX	XXXX	XXXX		XXXX
	Forecasting			XXXX	XXXX		XXXX
	Decision making	XXXX	XXXX	XXXX	XXXX		XXXX
	System understanding	XXXX			XXXX	XXXX	XXXX
	Social learning	XXXX			XXXX	XXXX	XXXX
Input Data Type	Qualitative and quantitative		XXXX				XXXX
	Quantitative only	XXXX		XXXX	XXXX	XXXX	
Focal Range	Focused and indepth				XXXX		
	General and broad	XXXX					
	Compromise			XXXX			XXXX
	Both		XXXX			XXXX	
Express uncertainty	Yes		XXXX				XXXX
	No	XXXX		XXXX	XXXX	XXXX	
Model Output	Individual					XXXX	
	Aggregated	XXXX	XXXX	XXXX	XXXX		XXXX

Figure 2.1: Functionality of selected methods of Integrated Modelling [7]

If compared with other modelling techniques that are used for applications and data mining, Bayesian Networks have the following benefits (see Figure 2.1 [7]): they use probabilistic, rather than deterministic expressions to describe the relationships [59]; they are an appropriate method for dealing with systems where uncertainty is inherent because uncertainty is accounted for the model itself; they include structure and parameter learning; they facilitate learning about causal relations between variables [92], and can be adjusted to new knowledge and facts; they are able to learn a model based on observations. On the other hand BNs also have the some limitations. First of all, it can be difficult to reach an agreement on the structure of nodes, their states and relations of BN with experts. Secondly, some packages of BNs can have a limitation in work with streaming data. Another limitation is a spatial and temporal scaling within BN. The approach that can solve this challenge is the development of a BN for each time/location case and performing them separately [3].

Building a Bayesian Network

The starting point for building a BN is to define a model for the given problem, the variables of interest that will become a nodes in BN, objectives, perspectives, system and scale. The next step is to build a conceptual model of the system and parameterize it with the given data set.

The objective of a decision problem is the ultimate reason why a user is interested in solving the problem [38]. The proposed model should have clear operational meaning for experts who model the BN, the domain experts and the end users. When the model is defined poorly or incorrectly, unfocused objective will compromise the model development process [5].

The objective and particular perspective are key components for the final decision. The perspective point of BN modelling incorporates stakeholders and network experts, who are the most affected parties in the case of decision making and can have different interests [38]. It can influence different scenarios when the final model will be used for reasoning. Finally, BN can be used for arbitrary combinations of diagnostics.

Structure of a Bayesian Network

We will give a simple example of the structure of a BN using well-known problem of modelling a Sprinkler system [68] (see Figure 2.2). The node *Cloudy* affects nodes *Sprinkler* and *Rain*, which in turn may affect node *WetGrass*. In this case, node *Cloudy* is referred to as a parent of *Sprinkler* and *Rain*, with *Sprinkler* and *Rain* being referred to as the children of *Cloudy*. This model can be described as follows: grass becomes wet if either the sprinkler is switched on or it is raining outside. Also, suppose that there is an event of cloudy that has a direct effect on the use of the sprinkler or a possibility of a rain.

In a BN, the directions of arcs cannot loop back (i.e., cycle back into the model) and the form of the structure is a (DAG).

Condition probability tables

As we have already defined the structure of BN, next we will quantify the relationship between nodes that have links with each other. The relationship between child and parent nodes is de-

Figure 2.2: A structure of a simple Bayesian Network designed using Bayes Server, version 5 [83]

scribed by building of *Conditional Probability Table* (CPT). Every node should have a CPT associated with it that contains combinations of the parent node, parent state values and each cell contains calculated conditional probabilities. It can be easily observed, that a big amount of dependencies and states in parent and child node will introduce an exponential growth of the size of CPT. It is one of the weaknesses of BNs and it was described in details in the work of Cooper [17]. In the case when a node has no parents, like a root node, it can be probabilistically described by a marginal probability distribution [54].

Figure 2.3 shows CPT for node *WetGrass* used in the BN in the Section 2.1.

P (WetGrass | Sprinkler, Rain)

	Sprinkler	Rain	WetGrass = False	WetGrass = True
▶	False	False	1	0
	False	True	0,1	0,9
	True	False	0,1	0,9
	True	True	0,01	0,99

Figure 2.3: Example of a CPT for the node *WetGrass* created using Bayes Server, version 5 [83]

The next step is a calculation of the probability distributions of each node. As the BN was already queried, it is ready for evaluation (see Figure 2.2). After performing evaluation tests, the network is complete and can be used for scenario analysis.

Individual scenarios can be built as follows: user can specify a level of accuracy, recompose models and create a set of observations of the system. All these steps can be applied to simplify examination of the network. The next step is a testing of scenario. The user queries the network by setting a set of evidences in the defined nodes. After that he can easily obtain the probability distribution of rest nodes. One of the major advantages of BN is a rapid view of node and whole system changes affected by decisions and modified conditions.

Multiple-criteria decision aid

Although a BN model is effectively used to aid decisions by observing the value of uncertainty of items of interest, but in certain circumstances, it is important to make a decision based on multiple criteria. Using of poor BN theory makes it impossible to solve such problems as BNs ignore multiple criteria. In this case MCDA [95] technique can be applied to improve the effectiveness of obtained results. The main goal of MCDA approach is to rank all less or more preferred alternatives. Sometimes trade-offs or conflicts can arise among objectives, because options that are more beneficial can be more costly. Therefore the identification of the objectives and perspectives is the next step. After that it is necessary to define the decision problem and specify such parameters as set of possible actions or alternatives that user can use; set of functions defined on actions; set of criteria constraints [38].

When we speak about multi criteria, we refer to the idea that each alternative or option can be observed from several points that are usually in conflict. Such criteria should be combined simultaneously and then evaluated in making decisions.

The set of alternatives that should be estimated is finite. The main conflict criteria are cost and product quality. When purchasing a house cost, safety, attractiveness, criminal level in the district can be the key criteria. It is logical that expensive house will be built in the most quiet, friendly district. In this case in order to make a correct decision set of constraints should optimise a number of possible conflicting criteria (e.g. amount of salary). The ideal case is to minimise the set of criteria in order to get just a single action to choose. During criteria analysis any action that fails to satisfy the constraint for criteria will be deleted. Only the most narrow, exact constraint will be used. To win in one conflicting criteria we have to lose in another. MDCA is a method that concentrates on such problem and helps to choose between actions [95]. MDCA assists in the problem structuring and proposes a decision that matches better for the long-term goals of the expert and possible environmental conditions [98].

However, MCDA has a set of limitations [38]:

- Since relevant criteria are well defined, the calculation of $g(a)$ for action a and given criteria g is obvious.
- The relevant criteria are certain and the value $g(a)$ is deterministic rather than stochastic for a given action a above parameters.
- The relevant criteria are independent of each other.

Combined approach of BN and MCDA

The advantage of BNs is that they help with complex and uncertain challenges by collecting information in consistent framework. They represent the uncertainties in natural way, measure probabilities and assign them to election results [38]. This fact suggests the existence of other parameters and factors, except uncertain criteria, in a final set of nodes that can influence the result value of a criterion for a given action. These factors are also called risk factors and can be hardly controlled by a decision-maker.

The combined approach of BN and MCDA was proposed in [38]. *Goal Question Metric* (GQM) [38] is used in order to define the underlying measures for the chosen criteria, MCDA is used to provide a means of combining resulting measures and to rank the actions as a result. The majority of key criteria will depend on various factors that we have to identify to make predictions of the values of uncertain criteria for the different actions. As can be seen on the Figure 2.4, at this point we use BNs to compute values for each criterion and apply MCDA to combine the values and rank the actions.

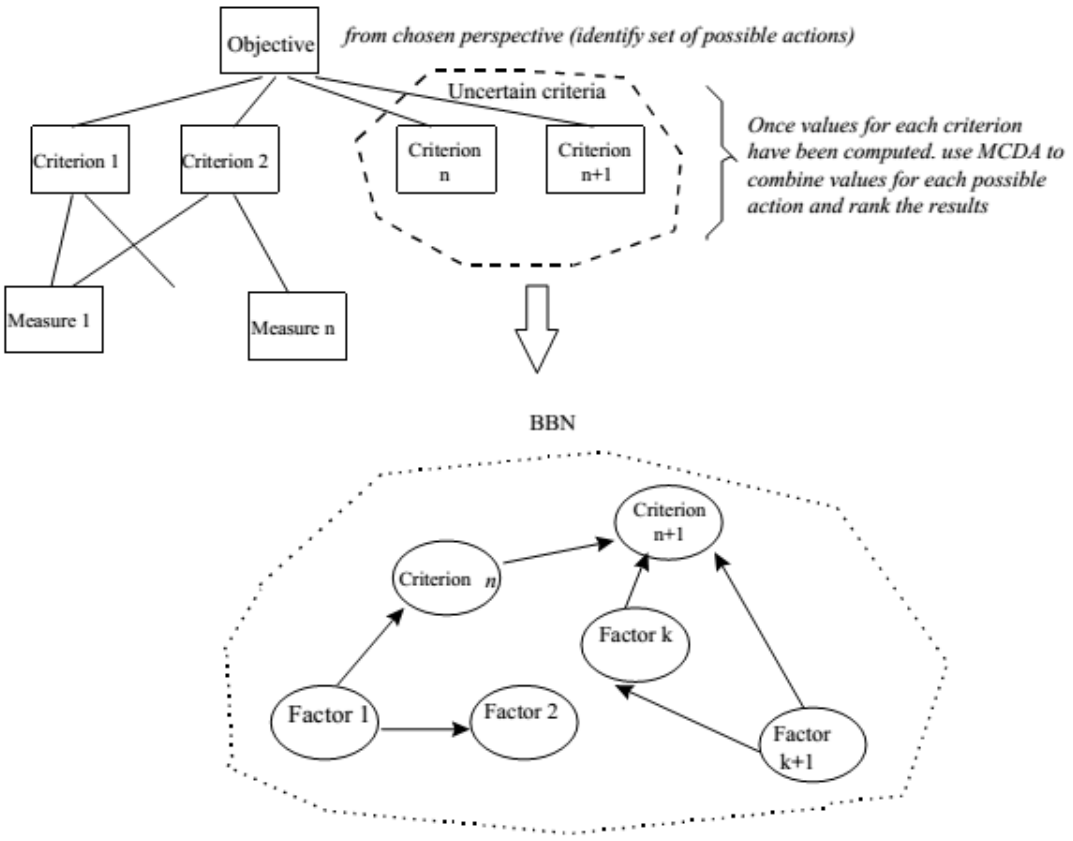


Figure 2.4: Usage of Bayesian Networks with GQM and MCDA [38]

Each criteria g_i is stated as an utility weighting u_i that represents the relative importance of each attribute for the given decision problem. The overall utility $U(a)$ of an action a is then

simply the weighted sum $\sum u_i g_i(a)$.

The method described above can be simply modified in consideration of each decision problem and therefore allows for sophisticated and flexible utilization of BNs in the decision-making analysis.

2.2 Cloud computing concepts

As was mentioned in the Chapter 1 in order to perform realistic simulations of Cloud Computing infrastructure the application tools that are used for this task have to follow acknowledged guidelines, embrace a wide range of cloud principles and make use of them extensively [14]. Such principles should be based on the knowledge obtained from practical experience and therefore become an advise how to build architectures, define the important design decisions, and cover limitations that have to be considered.

Cloud Service Models

Nowadays there is a wide of range of cloud service models. Among them are not only the most common Infrastructure as a Service(IaaS) [14], Platform as a Service(PaaS) [14] and Software as a Service(SaaS) [14], but also recently appeared Data as a Service (DaaS) [4], Backend as a Service (BaaS) [52], etc. As each service model introduces own workload issues, e.g., IaaS (shown on the Figure 2.5) requires from a provider intelligent provisioning of Virtual and Physical Hardware, where PaaS involves also management of application tasks, load balancing, etc., each of them should be simulated independently from others.

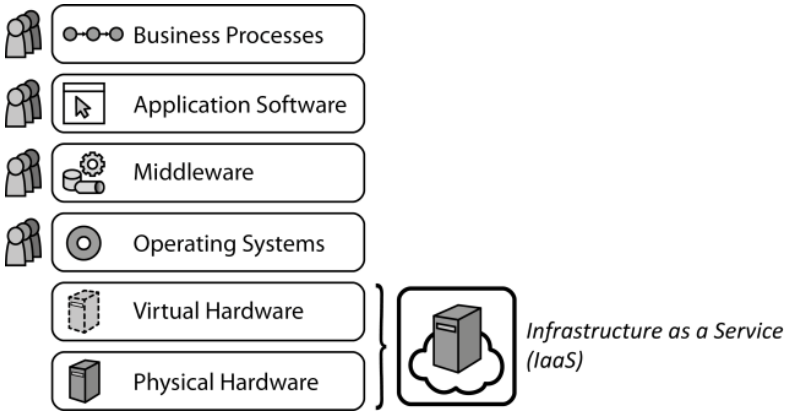


Figure 2.5: Infrastructure as a Service [14]

Application workloads

Application workload patterns capture different user behaviours on a hosting VM that results into various changing utilization models of IT resources. Learning and identifying of these workloads helps in prediction of IT resource usage and application of betters scaling techniques.

There are several most frequently occurred workload models as follows: *static workload* that only changes minimally over time, *periodic workload* that has recurring peaks, *once-in-a-lifetime workload* that has a peak once, *unpredictable workload* that changes frequently and randomly, and *continuously changing workload* that grows or shrinks over time [14].

Static workload

Static Workload [14] has a flat utilization profile over time within certain boundaries, therefore for the provisioning of such workflow there is no necessity to vary processing power level, memory size or bandwidth. Required IT resources can be provisioned for the static load with a certain over-provisioning rate to be able to solve any minimal variances in the workload. From the point of elastic scaling, in the case when experienced workload comes close to the level of completely utilizing of IT resources, minor alignments can be performed, that will be a relatively low cost overhead for the minimal overprovisioning. But benefits from such dynamic alignment are limited because of the equal utilization over time. In particular cases, costs may increase. However, in other cases the effects of IT resources and elasticity homogenization can be profitable to use in the cloud. Such kind of workload is experienced by private, small or medium size enterprises who do not fully utilize their VM [14].

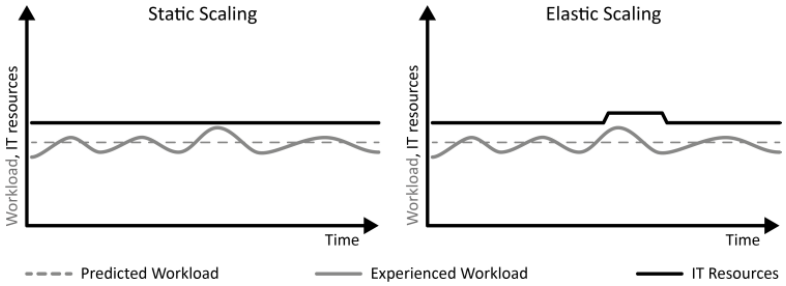


Figure 2.6: Example of Static workload [14]

Periodic Workload

One of the most common workloads is periodic, such as yearly car checkups, monthly telephone bills, daily use of public transport in rush-hours [14]. Generally they are utilized by people at the same time intervals. As far as business processes, tasks and routines associated with IT systems, periodic utilization can be observed in IT systems as well. The general issue of IT industry to handle in case of periodic workload and static scaling is the sufficiency of resources during periods of peak utilization and their inefficacy in non-peak periods. Since such overprovisioning can result in a low utilization of the allocated IT resources, can be used an elastic scaling to monitor experienced workload and if it increased during peaks of utilizing resources can be added dynamically. Such technology enables to combine resources and assign those that are not used by one customer to another one [14].

The cloud providers often use so-called resource pooling property than enables assigning of resources not used by one customer to another customer.

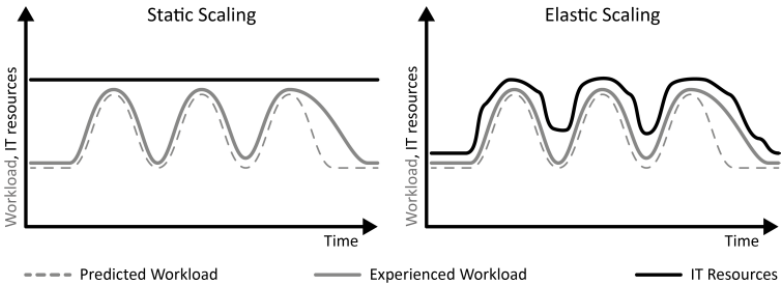


Figure 2.7: Example of Periodic workload [14]

Once-in-a-lifetime Workload

A special case of periodic workload is a once-in-a-lifetime workload [14]. It is characterized with only one peak of periodic utilization that occur in a long timeframe and expected in advance. Such peaks relate to a certain event or task. On the one hand, such peak does not happen frequently and the challenge of the enough resource provisioning cannot repeat again. But on the other hand, such peak can require even more IT resources to handle the task flexibly than during common peaks during periodic workload. For solving such challenge the same method as for periodic workload is used. Resource provisioning is accomplished for predicting workload peak via static scaling. But here can arise one issue: wrong prediction. In such case if the workload higher than it was expected, additional resources required for efficient task calculations often cannot be quickly provisioned. Such cloud behaviour will affect performance and violate results. To solve this problem elastic scaling can be used, that allows to handle increases of experienced workload even more than expected [14].

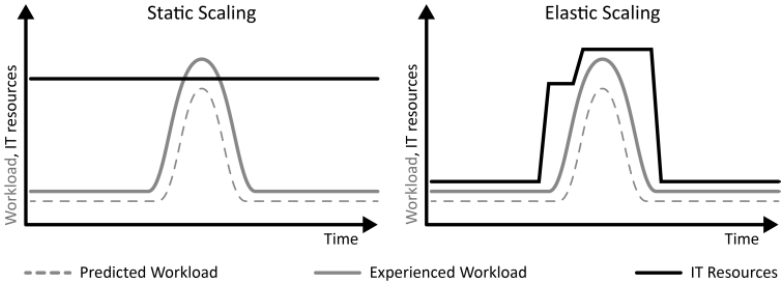


Figure 2.8: Example of Once-in-a-lifetime workload [14]

Continuously Changing Workload

Another type of workload is a continuously changing workload [14] that implies utilization of IT resources of two phases: ongoing continuous growth or decline of the utilization. Such workload occurs quite often and the rate of the workload change varies or unknown. Cloud elasticity guarantees applications same rate of IT resource provision and decomposition when the workload changes. Static scaling enables stepwise IT resource provision. Large increments

can be observed on Figure 2.9. They are presented because the physical hardware (such as clusters, servers) is more efficiently provisioned in large bulks. Using elastic scaling it is possible to provide IT resources more flexible continuously one by one [14].

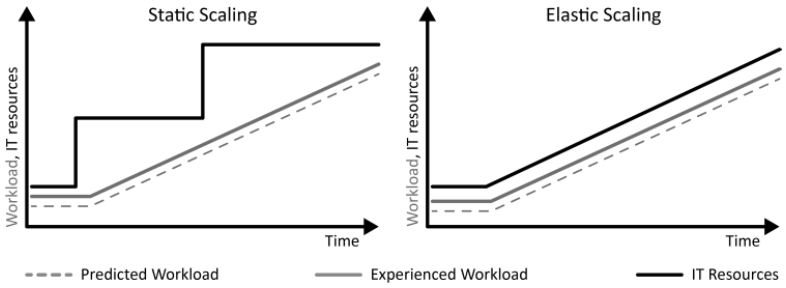


Figure 2.9: Example of Continuously Changing workload [14]

Unpredictable Workload

The last workload type is an unpredictable workload [14] that is a general case of periodic workload as it requires elasticity, but is not predictable. Such workloads are characterized by random or unforeseeable IT resource utilization and occur often in the real world. They require unplanned and unpredictable provision and decomposition of IT resources from hosts. Therefore, it is extremely hard to handle them with static scaling. Generally IT resources are provisioned to a certain average level that is economically feasible. In case of static scaling when the workload unexpectedly exceeds this level, the application will fail. The better possibility can be provided by elastic scaling. In this case workload predictions are omitted and experienced workload is just being monitored. Then the changes in application workload will be handled by newly provisioned IT resources or removal of unused once [14].

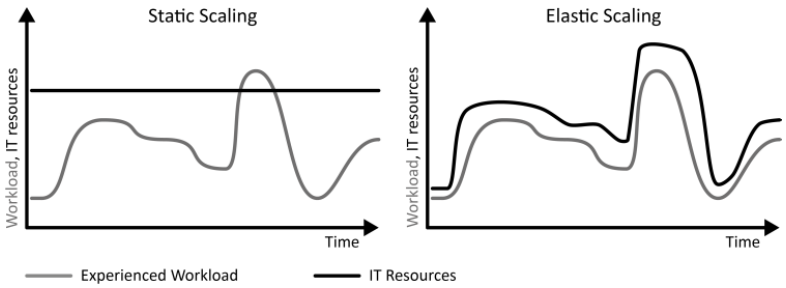


Figure 2.10: Example of Unpredictable workload [14]

Elasticity Manager

The cloud elasticity manager is an autonomic manager for the utilization of Cloud resources on which application component instances are deployed. Its main tasks are monitoring and

analysing of the Cloud environment infrastructures, prediction of future possible resource consumptions, and the execution of user given policies. Thereby elasticity manager has the possibility on-the-fly to grow and shrink the Cloud infrastructure at run-time.

Live VM Migration

Live VM migration (Figure 2.11) is a replacement of running VMs seamlessly across distinct physical servers without any impact on VM availability for the end user [67]. During migration such VM components as process memory, storage, and network connectivity are transferred from the original host machine to the destination. VM migration improve performance, manageability of system and allow data centers to serve users in flexible and efficient way [35]. Live VM migration became an appropriate tool for such scenarios like power management, VM load balancing, and fault tolerance.

Live migrations are extensively used in virtualized data centers. However, migration costs may depend on many factors like the diversity of VM configurations, workloads, energy costs, data center cooling efficiency, etc. Different decisions about those factors may result in significant differences of energy consumption, downtime, and performance. Considering all these issues cloud providers try to develop more efficient VM allocation and migration algorithms.

There are several techniques for live VM migration. The first one is a *pure stop-and-copy* [8,26,53] technique. Since VMs are stored like regular files on disk the one possibility is to copy VM entire memory to the destination using portable storage devices or network [47]. On the one hand this technique is very simple and it involves minimal total migration time. But on the other hand both downtime and total migration time are proportional to VM physical memory and since VM is suspended during the whole time when data are transferred to the destination host, the user can be suffered from high downtime. Another possibility to perform VM migration is a *pure on-demand* [100] migration that requires VM to stop and copy only the most important VM data to the destination. Although, use of this method requires a very short downtime, however, it requires high total migration time and therefore introduces degradation of host performance after migration. Both stop-and-copy and on-demand migrations have poor performance [25].

Another approach is a *pre-copy* migration, that combines bounded iterative push phase with a final and typically very short stop-and-copy phase [25]. “Iterative”phase implies the idea that pre-coping performs in multiple rounds, in which the VM memory pages that have been modified during the previous copy round will be transferred to the destination. It is supposed that the number of modified pages is small enough to halt the VM and after copying pages will be restarted on the destination host. Such design can be more efficient as it minimizes migration overhead and downtimes.

There are several key factors that have to be studied as a prerequisite for accurate modelling of live vm migrations: available migration link bandwidth, VM disk size, memory size, or page dirty rate. The most important are described below [25]:

- *Migration link bandwidth*: this is one of the most influential parameters for migration performance. The higher speed links the faster data will be a transfer and less time will be used to complete VM migration to destination.

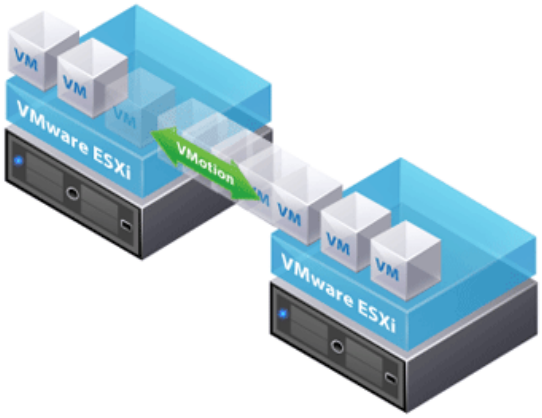


Figure 2.11: Example of Live Migration with using of vSphere Cluster [77]

- *Page dirty rate*: it defines the rate at which memory pages are modified. The value directly affects the number of pages that are transferred in each pre-copy iteration. The higher the rate the larger amount of information will be sent and the longer will be total migration time of VM. Dirty pages at a high rate will hit the iteration threshold.

CHAPTER 3

State of the Art

The following Chapter will give an overview of related work in the field of the costs-based decision-making that concern cloud computing. High attention will be paid for the techniques that aim to reduce energy consumption for cloud data centers. Additionally, it will detail the comparison of cloud simulation frameworks, describe their advantages and drawbacks.

3.1 Cloud energy management

There are large amount of prior techniques on energy management in cloud data centers. The reduction of energy consumption leads as well to the reduction of operational electricity costs. As was mentioned in the Chapter 2 the most common idea is automatic load balancing of VM resources using sophisticated scheduling and further replacement between hosts. The section will overview recently proposed state-of-the-art techniques in this field of study.

Energy-aware Green Cloud solution

One example of such energy-aware approach was proposed in [6] by Beloglazov et al. The authors emphasize on huge amount of electrical energy consumed by data centers that leads also to high operational costs and therefore propose a Green Cloud solution that allows not only to minimize operational costs but also to reduce environmental impact.

Figure 3.1 presents four following main entities defined in the paper that involved in a Green Cloud computing infrastructure supported energy-aware resource allocation: *Consumers/Brokers*, *Green Service Allocator*, *VMs*, *Physical Machines*.

Among all parts of a Physical Machine, such as CPU, memory, disk storage and network interfaces, CPU consumes the main part of energy. Additionally, on average an idle host consumes about 70% of the power consumed by the host running at full CPU speed. Therefore the main focus of the work is applied to power consumption and energy usage of physical machines.

This approach divides the problem of VM placement in two: selection and admission of newly created VMs and further optimization of their allocation. The proposed algorithm for the

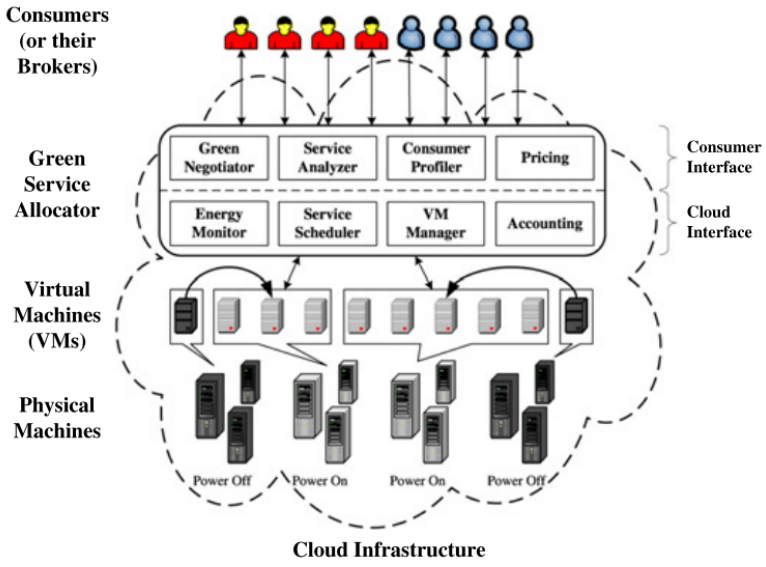


Figure 3.1: The high-level system architecture of a Cloud infrastructure supported energy-aware resource allocation [6]

first part is a modification of the Best-Fit-Decreasing (MBFD) algorithm that "sorts all VMs in decreasing order of their current CPU utilizations, and then allocates each VM to a host that provides the least increase of power consumption due to this allocation". Thereby it tries to allocate a VM to the most power-efficient nodes first.

The second step, the optimization of the current VM allocation, is performed as follows: first, the approach identifies VMs that should be migrated, second, applies MBFD algorithm to place them to the hosts.

The proposed idea of VM selection is to define lower and upper thresholds of host energy utilization and to keep total host utilization between them. For these purposes three following policies were defined [6]:

- *The minimization of migrations policy* selects the minimum number of VMs that have to be migrated in order to decrease CPU utilization below the upper threshold if it is violated.
- *The highest potential growth policy* is applied in the case if upper threshold is violated. It migrates VMs with the lowest CPU usage to reduce the potential growth of the power utilization and prevent possible SLA violation.
- *The random choice policy* defines a random selection of a number of VMS for decreasing of the CPU utilization by a certain host below the upper threshold.

The published experiment results showed that the approach allowed to reduce significantly energy consumption of Cloud data centers. However, the authors mentioned that there was still a large amount of opened challenges such as consideration of energy utilization by multiple

system resources, optimization of the allocation of communicating applications, optimization of amount of heat produced by physical machines, etc.

Consolidation of heterogeneous applications and forecast-based resource provisioning

Li et al. [29] have also considered the problem of high energy consumption and operational costs. As the result of the work they proposed a consolidation algorithm for sophisticated allocation of required resources. Their approach combines heterogeneous applications based on various correlations between them and forecast-based resource provisioning algorithm that utilizes Bayesian Networks for these purposes.

The authors made two observations. They state that all cloud applications can be grouped by finite number of types. Additionally they assume that workloads occurred in data centers have typically seasonal cycles based on the work of Chen et al. [27]. Based on these observations authors propose to consolidate a various types of workloads on a smaller number of servers. In the context of this problem an *Online Colouring Bin Packing problem* (OCBP) is formulated as follows:

Input: A sequence of items $a_1, a_2, ..., a_m$ with size $s(a_i) \leq 1$ for each $1 \leq i \leq m$, where each item a_i comes with a colour c_i and the number of items with the same colour is at most η.

Output: The minimum number of bins such each items is placed in a bin without exceeding its capacity, and the items must be placed in the same order as they arrive, and no items of the same colour are placed in the same bin.

As the OCBP is a special case of regular Bin-packing problem that is NP-hard it is also NP-Hard. It means that it can be solved just using heuristics. The work proposes an Online Colouring First-Fit (OCFF) algorithm as a modification of First-Fit (FF) approximation algorithm and proves that it provides a 1.7 performance guarantee.

After that authors introduces a Predictive Bayesian Network (PBN) model that tries to obtain estimates of the time-varying workload demands. Further these estimates are utilized in order to dynamically change the number of active servers and their frequencies as the work considers the usage of DVFS technology that scales processor clock frequencies and supplies need voltages.

The Maximum Likelihood Estimation is used for parameter learning of PBN on historical workloads. After the network is learned it can make the estimations. Further in order to minimize the operational costs the proper amount of resources are allocated using the estimations given by PBN. The allocation is performed according to the algorithm that computes mutual information between any two nodes with $O(n^2)$ operations.

The results demonstrate that the approach allows to achieve energy and operational cost savings close to near-optimal offline approach.

A Dynamic Nature of VM allocation requests

Other interesting aspects of the VM placement problem is shown in the work of N. Calcavecchia et al. [30]. This work considers the dynamic nature of the incoming stream of VM allocation requests continuously arriving to the cloud. The paper proposes a technique called Backward Speculative Placement (BSP) that projects the past demand behaviour of a VM to a candidate

target host. Similarly to the [6] the authors utilize the algorithm first for handling the stream of deployment requests, second in a periodic optimization in order to handle the dynamic aspects of the demands.

The work introduces a scoring functions called "demand risk" [30] measuring the level of demand dissatisfaction of a host in the last TM timestamps. Equation 3.1 [30] states the function U that measures the amount of unsatisfied demand for host h for given time instant t when the set of VMs V is placed on it and its capacity is reduced by $1 - \delta$ percent.

$$U(h, V, t, \delta) = \frac{\sum_{l \in V} d_l(t) - \delta \cdot C_{hCPU}}{\delta \cdot C_{hCPU}} \tag{3.1}$$

The following equation 3.2 [30] denotes the "demand risk"DR:

$$DR(h, V, T) = A \cdot \sum_{t=T-TM}^{T} U(h, V, t, 1) + B \cdot \sum_{t=T-TM}^{T} U(h, V, t, \delta) \tag{3.2}$$

where A and B are the coefficients that measure satisfied and unsatisfied VMs demand in the last TM time instants. It is assumed that $A \gg B$.

The "demand risk"is utilized during continuous deployment phase while allocating VMs. In the case when new request to deploy a VM is received the score is computed for each host and then a VM is allocated to the one with the highest score. Thereby the algorithm follows a Best-Fit-Decreasing strategy.

As actions triggered during continuous deployment phase can bring system to a sub-optimal state the placement are periodically optimized involving VM migrations. The most loaded hosts are identified and VMs are migrated towards hosts which are most likely can ensure demand satisfaction according to "demand risk". For a given VM the score is computed for the current host and then compared with each candidate target host. If the difference is positive the migration is considered.

The consequence of the second phase is a possibility that from the moment when it is triggered to the moment when it finishes a sensible amount of time might elapse and newly arrived deploy requests might not be placed correctly. The authors propose to avoid this problem by switching back to the continuous deployment phase whenever a new request is received.

The demonstrated results show that the technique allows to achieve high level of demand satisfaction.

Multivariate Probabilistic Models

The paper of Sijin et al. [28] introduces the application of probabilistic models in order to improve utilization of resource utilization in the Cloud infrastructures. The authors pay attention of the fact that the techniques described in the previous Sections tend to divide the efficient placement of VMs in two phases: Target Mapping Generation (TMG) and Migration Plan (MP). Thereby as they notice these strategies cause an increase in the total migration costs in MP with the increase of the number of physical machines. The introduced technique called Physical Machine Candidate Selection (PMCS) for improving of resource utilization uses a multivariate

probabilistic normal distribution model to select appropriate physical machines for re-allocation of VMs before reconfiguration plan is generated.

The work introduces two following metrics, i.e., imbalance and volume which are multi-dimensional characteristics of VMs and physical machines.

Three major resource types, i.e., CPU, memory and I/O are considered [28]. The resource values are normalized and all information related to them are expressed as vectors shown on the Figure 3.2 [28]. The total capacity $\vec{C}$ of a physical machine is defined with a vector from the origin of $(0, 0, 0)$ to point $(1, 1, 1)$. The vector $\vec{L}$ represents the resource utilization of a PM. The vector $\vec{F}$ shows the remaining capacity in the PM which could be for allocation of new VM.

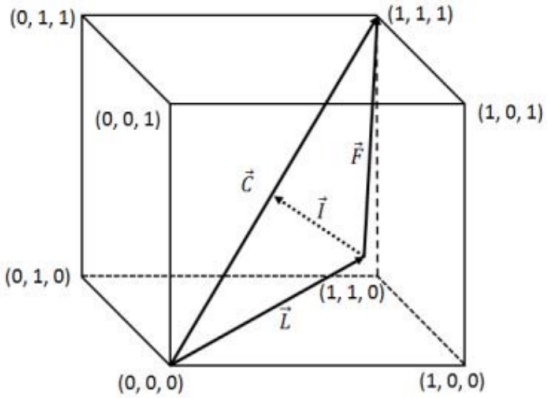


Figure 3.2: The Vector Representation of a physical machine [28]

Taking into account the vector representations defined previous the imbalance $\vec{I}$ can be denotes as the vector difference between $\vec{L}$'s projection on $\vec{C}$ and $\vec{L}$. Thereby it will indicate the degree of imbalance of resource utilisation of a PM. If $\vec{L}$ exactly aligns with $\vec{C}$ then the PM is well utilized.

Volume V is a measure that indicates the size of resource utilization of a physical machine and denoted in the Equation 3.3 [28]:

$$V(\vec{A}) = \prod_{i \in D} \vec{A}_i \tag{3.3}$$

where $\vec{A}_i$ is a resource vector.

A VM configuration can be expressed in the terms of vectors as well. If assume that there are a large number of different VM configurations arrive to a cloud it is reasonable to claim that they are in a normal distribution. Then the Probability Density Function (PDF) of the VM capacity vector x can be calculated as defined in Equation 3.4 [28]:

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)} \tag{3.4}$$

The authors divides the algorithm of VM placement into three stages.

Physical Machine Candidate Selection (PMCS) [28] defines the target physical machine for the VM allocation. It calculates remaining capacity $\vec{F}$ for every physical machine using the Equation 3.4. After that a host whose score is not in a similar size of the mean VM capacity is considered as candidate for the allocation.

Target Mapping Generation (TMG) [28] takes all candidates from the previous stage and applies one of two heuristics proposed by the authors in order to map a physical machine for each VM. One of them uses the imbalance metric, when the other one utilizes the volume metric.

Migration Plan (MP) [28] is mentioned in the work as an open research problem, though it is noticed that it must consider the costs associated with performing the migration of VMs using a certain cost function.

The evaluation presented in the paper demonstrates a minor decrease in resource utilization levels that results from reducing a number of physical machines for re-allocation. Therefore the approach tends to lower a number of re-allocated VMs, i.e., to reduce migration costs. Also the results show advantage in usage of the imbalance heuristic over the volume one in user-defined view and the opposite fact in the provider-defined view. Additionally, the evaluation shows that multidimensional heuristic is better than non-dimensional one in the used-defined view and has no significant difference in the provider view.

Exact VM allocation and migration using linear programming

In [15], the authors pay attention once more to a problem of excessive energy consumption of data centers. They propose an exact energy allocation algorithm using classical formulation of Bin-Packing problem. The algorithm is based on the linear integer programming techniques and aims to reduce the number of running hosts in a data center. They combine exact allocation and migration algorithms to reduce overall energy consumption in the data centers.

The simulation model is considered similarly to previous works: an infrastructure provider allocates VMs to physical hosts. At the same time it tries to reduce energy power consumption consolidating the placement of VMs using migration and maximizing the number of idle servers that can be put to sleep mode.

The proposed system model consists of the following modules [15]:

- *Cloud IaaS manager* controls and manages resources, handles VM scheduling, fetches and stores VM images in storage spaces.
- *Energy estimation module* behaves energy estimation tool that, e.g., can use power models to obtain power consumption of VMs or servers from their resource usage. It is an intermediate part between the cloud infrastructure manager and the energy-aware scheduler.
- *Energy-aware VM scheduler* places VM to the target physical hosts in the data centers. It consists of two modules: first performs exact VM allocations, second is responsible for further dynamic consolidation of VMs. The servers that are not used are putting into sleep mode that thereby reduce the power consumption.

The exact allocation algorithm is presented as an extended Bin-Packing approach through inclusion of valid constraints and inequalities as follows:

- Each physical host has power limit that cannot be exceed.
- All customer requests within a prescribed SLA or quota should be fulfilled. Moreover each request should be assigned just to one and only one physical host.
- The sum of provisioned resources for all VMs hosted on a physical machine should not exceed the capacity of this machine.

The migration algorithm is designed as Decreasing Best-Fit heuristics for exact and extended Bin-Packing problem. The VMs are sorted in decreasing order of power consumption and are placed to the server with the smallest remaining power consumption budget until a VM fits in it this target host. The process continues until all VMs are not placed and packed as much as possible in the most occupied servers. The unused serves is put to sleep mode.

Described migration algorithm relies on an integer linear program (ILP). The ILP algorithm introduces a number of valid inequalities described previously to reduce the span of the convex hull of the migration problem. Ideally, the algorithm has to minimize the number of running physical hosts and maximize the overall number of VMs hosted by these hosts. Additionally, it considers power consumption caused by migration and aims to minimize it.

At the end, two algorithms are combined together to achieve minimal energy consumption in data centers. "Both the exact Bin-Packing extension and the Best-Fit heuristic are utilized to achieve optimal and suboptimal placement respectively."

The results show the benefits of combining the allocation and migration algorithms and significant reduction of power consumption of exact Best-Fit algorithm in comparison to the Best-Fit heuristic.

A Two-Tiered On-Demand Resource Allocation Mechanism for VM-Based Data Centers

Y. Song et al. state in [36] that existing approaches that tend to turn on and off servers with the help of VM migrations are not efficient enough. They argue that the key improvement of resource utilization and throughput lies in the finding of optimized dynamic resource allocation method. They propose a Two-Tiered on-demand resource allocation mechanism consisting of the local and global resource allocation with feedback to provide on-demand capacities to the concurrent applications. Using these mechanism they also propose a set of on-demand allocation algorithms. It should be mentioned that although this work assume its usage more likely in PaaS and SaaS environments, the ideas proposed by authors can be as well extended to IaaS environment.

Figure 3.3 depicts the proposed mechanism. Comparing both figures it can be easily observed that the two-tired on-demand resource allocation algorithm differs from the traditional approach in adding a resource management level for VMs.

The applications ("application 1"... "application S") illustrated on the Figure 3.3 consists of multiple instance copies each of which is allocated in a VM. The VMs hosting the instances of the same application belong to the same application domain. Obviously each server hosts VMs that belong to multiple application domains. In this case each VM from a certain server

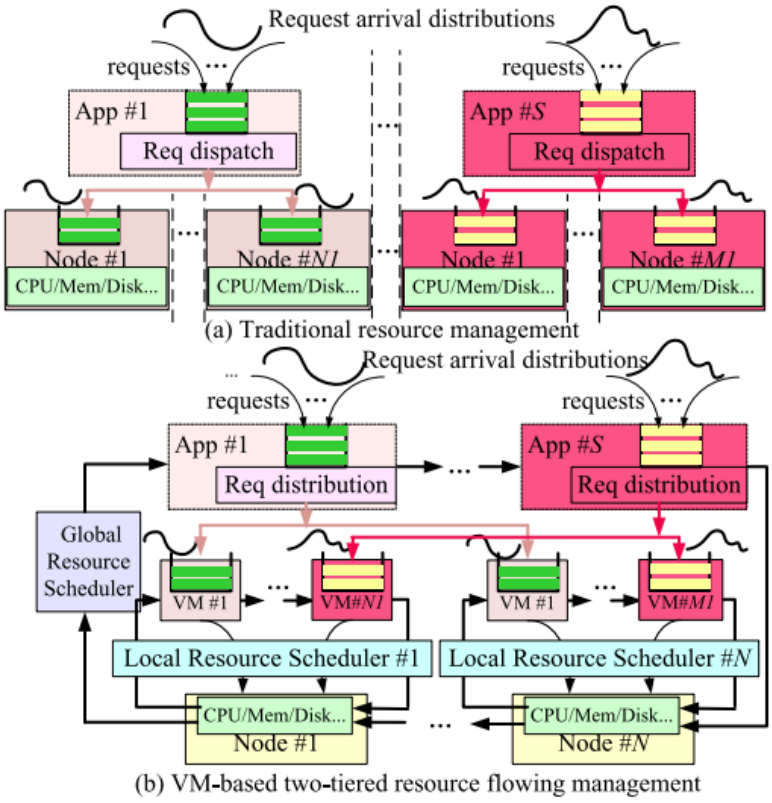


Figure 3.3: Two-tiered on-demand resource allocation mechanism versus traditional resource management [36]

encounters time varying and various from each other workloads. This fact causes the problem of dynamic resource allocation among VMs with a single physical machine. Moreover, the application workloads on different physical hosts are also time varying and of various type that results into the problem of dynamic resource allocation among applications. As there are currently no technological support on the resource allocation on a server to a VM residing in another server, the authors provide global resource optimization technique.

The work consider the control theory as the basis of modelling and designing feedback-driven closed-loop allocation algorithms. The two-level control model, illustrated on the Figure, is utilized. It is assumed that an object to be controlled is often represented as input-output system. On the Figure 3.4 Controller-L and Controller-G correspond to the local and global resource resources schedulers respectively.

Controller-L is responsible for the allocation of application to a server according to the resource utilization $U_i(t)$ of each VM, static priority SP_i and activity $A_i(t)$ of an application. Resources utilization is considered by authors as the CPU utilization. The output $C_i(t+1)$ refers to to the resource assigned to VM_i at time $t+1$.

Controller-G is responsible on the other side for the activities of applications. Resources

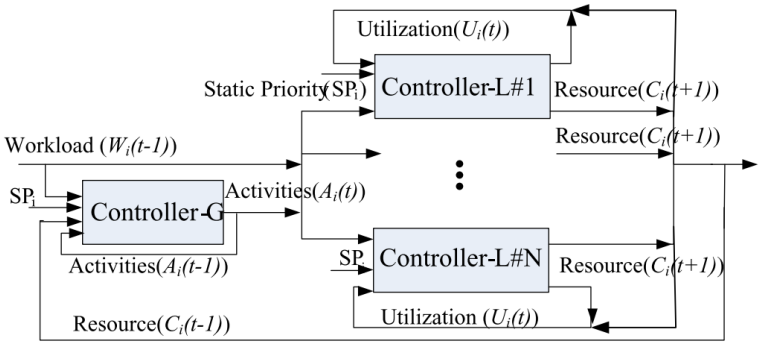


Figure 3.4: Two-level feedback control model [36]

allocated to each application $C_i(t-1)$, static priority SP_i, workloads $W_i(t-1)$, and the feedback activity $A_i(t-1)$ of each application are the inputs. The key parameter is an activity $A_i(t)$ that controls resource allocation in Controller-L. The proposed loop should have an actuator to implement the changes indicated by the control knobs. The global resource allocation algorithm proposed by authors is exactly its actuator.

The work defines two different resource allocation problems as follows (see Figure 3.5) [36]:

- K-VN-1-PM problem of allocation of applications to K VMS within a single physical host.
- K-VN-N-PM problem of allocation of applications to K VMs within multiple number of physical hosts.

The goal of the further resource allocation is the optimization of the qualities of the hosted application taking into account their priorities in the scope of limited resources. The authors propose to use optimization theory and to transform the resource allocation problem into optimization problem with limiting conditions. After they utilize Simplex Method is used in order to resolve it.

The K-VM-n-PM problem is further resolved using the same approach as applied to K-VM-1-PM problem.

The K-VM-1-PM model is used by authors to design a priority-based local and global on-demand resource allocation algorithms to optimize the resource allocation among applications. It is mentioned that the model takes a number of arrival requests during the last discrete interval as a predictor at the next interval.

The local resource allocation algorithm, called ResourceFlow-L, is a combination of two algorithms [36]:

- *CpuFlow-L* dynamically adjusts the weights of VMs according to their static priorities, resource utilizations, and activities of the hosted applications. It introduces double-threshold approach that allocates more resources for a VM which CPU resource usage exceeds the upper threshold and decreases resources for a VM if a CPU resource usage becomes lower than the lower threshold.

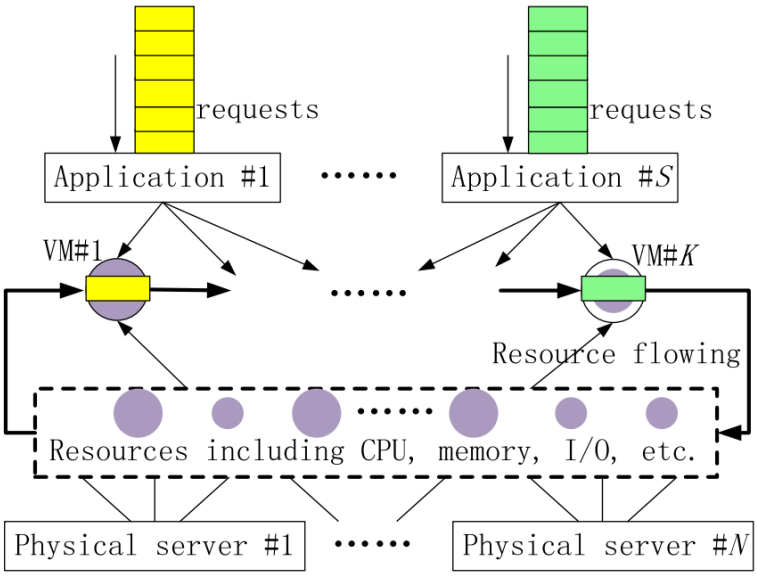


Figure 3.5: The on-demand resource allocation problems [36]

- *MemFlow-L* dynamically controls memory allocation determining the events when the memory overload in a VM occurs. The algorithm uses one threshold that defines that the memory need to be reallocated if the idle memory is higher than threshold, or nothing should be done otherwise.

The global resource allocation algorithm, called ResourceGlow-G, uses K-VM-1-PM model to optimize resource allocation among applications in the entire system. It adjusts the activity of each application according to the monitored resources R_{it} and optimized amount of resources R^r_{it} that should be assigned to each application. In most cases, they are not equal to each other. Therefore the authors introduces the threshold that actuates the activity adjustment of VMs as it avoids frequent adjustment. If R^r_{it} is more than R_{it} more than a threshold, ResourceFlow-G decreases the activity of a VM at time t. In turn, if R_{it} is more than R^r_{it} and difference exceeds the threshold the algorithm increases the activity of a VM at time t.

The evaluation demonstrates that the use of proposed method significantly improves application performance as well as in the CPU utilization in the typical enterprise environment. On the other side, it introduce degradation in the case of hosting multiple I/O-intensive applications with high workloads. Additionally, the authors show that the algorithm improves the performance of the critical applications as well as resource utilization, while just slightly degrading the performance of others.

3.2 Cloud simulation tools

Taking into the consideration that the thesis introduces a cloud simulation framework the state-of-the-art approaches in this field of study should be overviewed. The following section provides

a related work on a comparison of the cloud simulation frameworks, practices and ideas that should be considered during implementation of own simulation approach.

Cloud Testing Tools

X.Bai et al. in [97] made a broad and deep survey of existing cloud testing tools, their advantages and disadvantages, use cases to apply, etc.

There are main techniques used in cloud testing systems and their implementations as follows [97]:

- *Simulation.* An implementation of a cloud simulator allows to focus on a particular cloud component, and analyse system behaviour under various scenarios.
- *Service mocking.* External services used by the systems in the cloud should be mocked in order to at least provide a possibility to test the a system and furthermore to guarantee repeatability of results.
- *Test job parallelization.* Parallel programming is an obvious way to reduce time and cost of testing tasks by dividing them into independent jobs.
- *Environment virtualization.* Sometimes it necessary to perform tests in different testing environments, for various versions of software and platforms. The usage of virtual machines often can help to ease and accelerate the process, and to reduce test cost.

The following part of this section will overview of currently existing tools for the techniques presented previously.

Simulation

CloudSim [81], the most commonly used framework in the cloud computing research community, provides a tool kit for modelling and simulating behaviour of various cloud components such as data centers, physical machines, VMs, etc. including typical cloud features, e.g., VM allocation, cloud federations, dynamic workloads. Main purpose of its use is the evaluation of cloud resource provisioning strategies in a controlled simulated environment.

D-Cloud [34] "is a dedicated simulated test environment build upon Eucalyptus, an open-source cloud infrastructure providing similar functionalities as Amazon EC2. "

D-Cloud provides a possibility to simulate different typical faults of a cloud environment and to inject them into host operating system (OS). The XML-based workflow configurations are used for set up infrastructure and fault injection parameters. After a test plan is submitted, D-Cloud initiates VMs to simulate the execution process following the test plan.

PreFail [75] is a framework for systematic and efficient failure exploration, for validation of correctness and efficiency of cloud recovery protocols. In comparison to D-Cloud that provides simulated actual faults, PreFail inserts a failure into the target system or the OS library. Using such technique, testers can flexible set up failure scenarios.

Distributed Load Simulation is a cloud storage system that is characterized by highly parallelism and non-determinism that was implemented as a part of the Cloudy2 distributed database system. [46]

The framework contains two types of nodes: Master and Slave. Main and uniquely identified Master nodes are responsible for distribution, synchronization and management of Slave nodes. During the testing, waits for slaves connect to it, and then sends them tasks. The slaves run the tasks and store tests results locally. At the end of the test, master node collect all results, analyses them and then generates statistics.

This framework is capable to simulate a big variety of workload scenarios.

Service Mocking

iTKO [85] is a tool for analysing the constraints of enterprise systems of restricting availability and accessibility for development and delivery. It aims to provide a constraint-free cloud environment with fully virtualized services.

Test Job Parallelization

Cloud9 [55] is a software framework that provides an ability to use important testing technique named *symbolic execution* on a cloud platform. It divides the path of application exploration work into independent jobs that are assigned to different worker nodes. Further In order to get high efficiency of testing service the workload of each worker is balanced by the global load balancer. Each worker contains a runtime, a searcher and a constraint solver. It independently from other nodes explores a sub-tree of applications's execution tree.

Although the framework allows to reduce testing time significantly, it is still encounters a scalability problems with increasing application size and complexity.

HadoopUnit [90] migrates JUnit test framework to Hadoop platform [57]. The framework allows to submit execution of JUnit tests cases as independent Hadoop MapReduce jobs that can greatly improve testing execution time.

Similar possibilities to HadoopUnit provides **YETI** [74], another cloud testing tool. It also uses MapReduce to parallelize to map input testing tasks and to reduce final results.

Environment Virtualization

OCT (Open Cloud tesbed) [31] is a wide area testbed for cloud computing. The OCT was introduced for performing advance cloud computing research, enabling experimental studies involving different cloud computing architectures, providing a platform for cloud interoperability studies, and to encourage cloud computing benchmarks. The following heterogeneous platforms such as Eucalyptus, CloudStore, Hadoop, Sector/Sphere was tested using the OCT.

Summary

The novel design and methods introduced by cloud computing force the cloud testing tools to emphasize on system testing and online testing. Although there are lot of platforms proposed so far, many issues stay unresolved and need further investigation and research. For example,

fault-tolerance is one of the most important promise given by clouds can be tested just using D-Cloud and PreFail. Additionally, state-of-the-art techniques and tools lack the support of the following features [97]:

- *Cross-cloud testing.* Services has to be migrated between different application domains or even clouds. Therefore, a unified methods of testing and evaluation of services on heterogeneous cloud platforms is needed.
- *Online and adaptive testing.* An online testing tool for dynamic service composition, deployment and on-line evolution is necessary.
- *Real-time results mining.* The synthetic offline testing cannot always identify all issues encountered in real production systems, e.g., huge number of data can show bottlenecks in fault discovery, allocations, recovery techniques.

CHAPTER 4

Efficient allocation of cloud resources based on MCDA and BDA

The following chapter will formally define the problem of cloud power management mentioned in the Chapter 1 and present a desired algorithm as its solution. This algorithm involves the usage of a specific BN and further application of MCDA during decision-making process. Hence, we will additionally detail methodology and the BN design process and its usage in decision-making analysis.

4.1 Methodology

The building of a large BN is a step-wise process that starts with the creation of a certain small model that is further being enlarged and extended with new relationships. Initially, a designer of the network defines a set of the most important observations and measurements about a studied system. They are used either to create a model manually or to obtain the structure using one of the existed learning algorithms. After the model is built, the network can be used for reduction of the main problem. In the case, when the model returns feasible results and thus proves its applicability, new logic can be added into the model in order to improve further its efficiency. In the case, when existing solution does not met the requirements, it can be immediately fixed or reworked. The advantage of this method is that each following step move a designer towards desired goals.

Exactly the same method is used in the thesis for the building of desired BN that will be presented later in this Chapter.

4.2 BN for a simplified problem of costs reduction

Definition of simplified problem

The definition of the decision problem that was defined in the Chapter 1 can be simplified as follows: we will not consider any external factors and will try to reduce energy costs and SLA penalty costs paid in the case if QoS degrades and a customer is not supplied with a promised service as much as possible.

Starting from now and in the following parts of the thesis we will assume that the cloud provider offers an IaaS [14] service for its customers. It allows us to slightly reduce a diversity of possible parameters and more thoroughly focus on our problem, namely costs.

Recall that IaaS service model offers physical and virtual hardware that can be by demand of customers quickly provisioned and decommissioned through a self-service interface [14]. Using these IT resources customers are allowed to install their own operating systems, middleware, and applications software supporting their business needs and processes [14]. The main consequence of this fact is that a cloud provider should consider workloads of customers' VMs as an input and can manage only the placement of these VMs and power state (either switched on or off) of physical hardware. It is the starting point of our BN design.

Objectives and perspectives

It is already possible to define objectives, perspectives and decision problem in the scope of previously defined assumptions. The stakeholders of the problem are the provider itself as the decision maker and the customers that suppose the fulfilment of the duties defined in the SLA. In the case, if customers encounter poor service, e.g., frequent downtimes, provider will have to pay a penalty and in the worst case will lose the clients. Thereby the objectives of the cloud provider is to consume less amount of energy and at the same time try not to violate its responsibilities to the customers. As was already mentioned before, key challenge of implementing IaaS is determining of efficient placement of VMs, i.e., the assignment of each VM to a host in the provider's cloud infrastructure [30].

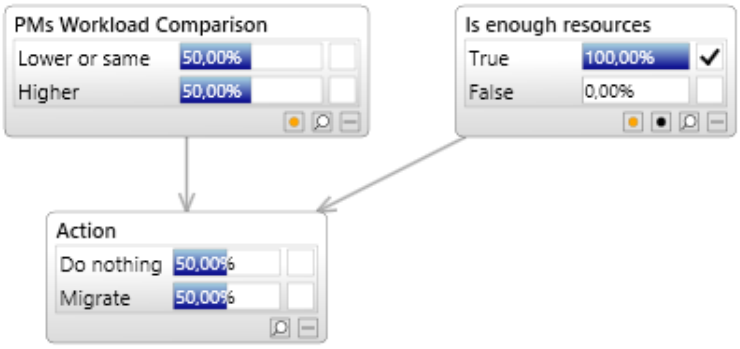
Design of a BN

We will start with a quite straightforward model that enumerates and compares all physical hosts pairwise and migrates a VM to the host that has higher workload and at the same time has enough resources to satisfy VM's needs. The Bayesian Network that is utilized in this case is depicted on the Figure 4.1. It has three nodes, two of them, *PMs workload comparison* and *Is enough resources*, are our observations from cloud infrastructure and the other one, *Action*, is a decision node that defines the action that has to be performed. Additionally, Figure 4.1 illustrates the CPT of the node *Action*.

We do not evaluate the efficiency of specially this network and the model in total, we only want to pay attention to the following facts:

- the structure of the model was created completely based just on our knowledge about studied domain and it is intuitively simple for understanding and further application

- the CPTs of the model can be either defined based on the experts' experience and used in forward direction or they can be learned using training data set in the backward direction beforehand



(a) Structure

P (Action | PMs Workload Comparison, Is enough resources)

	PMs Workload Comparison	Is enough resources	Action = Do nothing	Action = Migrate
	Lower or same	True	1	0
	Lower or same	False	1	0
	Higher	True	0	1
▶	Higher	False	1	0

(b) CPT

Figure 4.1: Structure (a) and CPT (b) of a Bayesian Network for solving VM Placement problem designed using Bayes Server, version 5 [83]

4.3 Bayesian Network for the problem of VM migration

After building a simple Bayesian Network in the Section 4.2, we will extend the model using more detailed knowledge about cloud provider infrastructure.

Problem definition and solution

A provider rarely owns just one single data center. More often there are several data centers geo-distributed all over the world and connected with each other using certain network topology. Each of them has own particularities such as specific electricity markets, different weather

conditions (e.g, Google place some of its data centers above the Arctic Circle [96]), various frequency of power outages, etc. All of them can have a big impact on the total costs spent by the provider for maintenance of its infrastructure.

We consider infrastructure of a data center composed of distinct physical hosts each of them has limited capacity of resources (e.g., CPU, bandwidth, RAM, etc.) and connected with each other via network. In order to achieve the best energy efficiency cloud provider has to balance the load of cloud resources by migrating customer requests to most appropriate hosts considering data center and host specifics. As was already previously mentioned in the Section 2.2 the main method of rebalancing workload in the virtualized data centers is a live VM migration. Moreover, we will assume that *pre-copy* migration strategy is used.

The approach of the Bayesian Network structure is based on the following considerations:

- A VM should be hosted in a data center with the cheapest energy price
- A data center already encountering high workload is considered as not suitable candidate for placement and migration of VMs, because many SLAs can be violated that leads to the payment of penalties
- Migration of VMs from one host to another induces additional overhead to a network infrastructure and can introduce malfunction in the life-time of cloud services. Migration gain depends directly on the current workload of a VM and can be presented by a *Dirty Page Rate* which value are computed from the combination of CPU and RAM workloads of the VM.

Decision problem investigated in the current case is the identification of target data center for each VM hosted in the cloud. It is assumed that the decision-making algorithm compares each data center pairwise and use certain observations from cloud entities. The set of possible actions regarding VM placement are defined as follows: *Migrate* a VM to target data center or *Do Nothing*.

Design of a BN for VM migrations

The Figure 4.2 presents desired Bayesian Network with already assigned evidences. All qualitative measurements are firstly discretized before evidence assignment. After that a certain qualitative level, e.g., *Low*, *Middle*, *High*, is assigned to each discrete interval. The network consists from the following nodes:

- *VM CPU* node represents current CPU workload of a VM considered as a candidate for migration.
- *VM RAM* node behaves similarly to the *CPU usage* node, except that it represents current RAM workload of a VM.
- *Dirty Page Rate* node expresses the value of Dirty Page Rate that is calculated based on the values of CPU and RAM workloads of a VM. Higher values of RAM and CPU workload imply higher values of Dirty Page Rate.

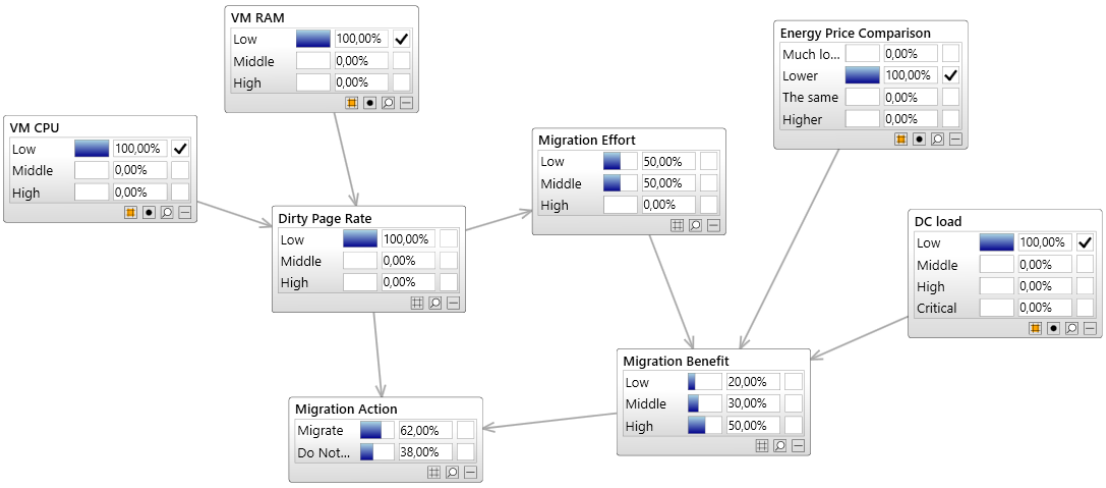


Figure 4.2: Bayesian Network Decision-Making considering VM Placements in the Cloud designed using Bayes Server, version 5 [83]

- *Migration Effort* node represents a qualitative value that defines an effort of VM migration to the target data center. Higher values of Dirty Page RateDirty Page result into higher migration effort.
- *Energy Price Comparison* node represents the comparison of energy prices between a data center where VM is currently allocated and a target data center
- *DC load* node shows the current load of target data center.
- *Migration benefit* node represents a benefit from migration of a VM to a target data center. This node has three parent nodes as follows: *Migration Effort*, *Energy Price Comparison*, and *DC load*. Higher values encountered by parent nodes will result into lower value migration benefit.
- *Migration Action* node is a decision node that calculates a quantitative assessment of each action that can be applied to a VM. Higher possibility set to a certain level of this node will identify that it is more suitable that the other one.

In the case, illustrated on Figure 4.2, we evaluate a migration of the VM with low CPU and RAM workloads to a data center with lower energy price and low overall data center load. The Network computed that the action *Migrate* is more suitable in current case and, hence, a VM should be migrated to the target data center.

Summary

Summarizing the models from the last two sections, it can be concluded that although presented models are a little bit oversimplified, but they became a good starting point for more efficient and applicable approach, presented in the following Section.

4.4 Cost-aware algorithm for allocation of cloud resources

The Section 4.2 and Section 4.3 give a short overview how simplified problem of VM placement can be examined and decided in the scope of existed knowledge about cloud service life-time.

The following Section will introduce and describe in turn in details the decision-making algorithm that was noted in the Chapter 1. It captures many criteria and factors arisen in the real-world cloud environment and uses combined method of BN and MCDA with the specified utility functions for certain decision actions.

Problem definition

The simplified definition of the problem was already done in the Section 4.2. We will formalize it and extend its scope.

We will recall the objectives of the cloud provider, namely the reduction of operational costs of running cloud infrastructure as well as minimization of SLA penalty costs. It is absolutely clear that these two criteria conflict with each other therefore we are dealing with multi-criteria decision problem. In addition, we assume that data centers are geo-distributed all over the world in order to cover local requirements of each customer and to improve responsiveness of cloud services. This fact introduce several open issues into the life-time of cloud provider .

Firstly, as was noted in previous sections, each area has its own independent electricity market that directly impacts on the energy costs. Global electricity price comparison [1] shows that sometimes the prices in one country can be ten times higher as in another one. Also some electricity markets experiences the reduction of energy prices during certain time of day, e.g., in the night.

Second issue arises due to different weather conditions in various parts of the globe. As was mentioned in the Chapter 3 temperature-aware analysis of data center workloads can greatly reduce energy costs as data center built in the cold region has smaller $pPUE$ rate [43], or broadly speaking, consumes less energy for cooling its infrastructure.

The third problem relates to power outages that as was noticed in the Chapter 1 can become a big issue for the provider. For example, statistics of electrical outages [23] states that in large number of countries it is even not possible to guarantee common availability level of 99.9 %. But if a provider decides to place a data center in such region or country with low quality of electricity services due to either peculiar properties or business requirements or simply much lower prices, the information about power outages will be useful. As example, we can consider the case when availability level of a certain customer is already almost violated. Taking into account that SLA penalties are high [1] [2], it would be a bad decision to place the VMs of this customer into the a data center with high possibility of power outage.

Problem formalization

The following section will give a formal definition of the problem of VM placement. Additionally, it will present the process of building of physical model for the cooling infrastructure that

[1] http://www.rackspace.com/information/legal/cloud/sla

[2] http://azure.microsoft.com/en-us/support/legal/sla/

will be an important part of the final algorithm presented in the following section.

VM Placement

As was noted previously the method of workoad balancing will be done using live VM migrations. At each simulation timestamp each running VM existed in the cloud can be in two states, either already allocated to a host or still without destination host assigned to it and therefore has to be allocated. We call this sets $placed(t)$ and $allocation(t)$ respectively. The set of all VMs will be called $all(t)$. It is clear that $all(t) = allocation(t) \bigcup placed(t)$. During each time point two following issues should be resolved:

- find target physical host for all VMs from the set $allocation(t)$
- find target physical host for migration of currently allocated VMs from the set $placed(t)$ if their current placement are not optimal enough.

It should be noted that the benefit from the VM allocation or migration are evaluated not only in the scope of the VM, but also in the scope of the whole cloud.

Additional set $migrated(t)$ defines a set of VMs that are being migrated to some hosts at time t. It is intuitively clear that $migrated(t) \subseteq placed(t)$.

We define a set of possible decision actions as follows: Allocate VM, Migrate VM. Additionally, we determine that the set of decision actions such as switch a physical host on are triggered automatically when at least one VM begin being allocated or migrated to it or switch off if no VMs are placed on it.

We assume that a data center consists of M distinct physical hosts. Each host is defined by a certain capacity of resources R. Each host m has a known limited capacity C_{mr} for each resource r, where $m \in \{1..M\}$ and $r \in \{1..R\}$.

We define the binary variable $x_{ij}(t)$ that indicates that VM v_i is placed at host j at time t.

Equation 4.1 states that each VM from the set $placed(t)$ will be placed exactly in one host.

$$\sum_{j=1}^{M} x_{ij} = 1, \forall v_i \in placed(t) \tag{4.1}$$

Each VM v_i has its specifications that defines upper bound of each resource $max(vr_{ir}(t))$ required by it at any point of time. During each simulation timestamp a VM requires certain amount of resources vr_{ir} that is considered during decision about placement of the VM. According to the specified strategy these resources will not be necessarily provisioned for the VM therefore we introduce additionally the amount of resources $vp_{ir}(t)$ that are provisioned for VM. This value can be less, equal or greater than $vr_{ir}(t)$. In the case when it is less VM experiences a downtime.

Usually there are 4 most common resource types considered for VM: CPU, RAM, bandwidth and size. For simplification reasons we assume that the hosts in the data center are always big enough to satisfy needs of RAM, bandwidth or size. Just one resource, CPU, is considered as limited. We will show later that the model can be easily extended for the case of limitations of other resources as well.

Equation 4.2 guarantees that amount of provisioned resources for all VMs allocated on a distinct host does not exceed the overall capacity of this host.

$$\sum_{i \in placed(t)} x_{ij}(t) \cdot vp_{ir} \leq C_{jr}, \forall j = 1..M, r = 1..R \tag{4.2}$$

Moreover Equation 4.3 states how the workload W_{jr} for a certain physical host j and resource r with allocated VMs can be computed:

$$W_{jr} = \sum_{i \in placed(t)} x_{ij}(t) \cdot vp_{ir}, \forall j = 1..M, r = 1..R \tag{4.3}$$

Further we define the binary variable $m_{ij}(t)$ that indicates that VM v_i is being migrated to host j at time t.

Equation 4.4 states that each VM from the set $placed(t)$ will be placed exactly in one host.

$$\sum_{j=1}^{M} m_{ij} = 1, \forall v_i \in migrated(t) \tag{4.4}$$

In our model we assume that the migration of the VM does not affect the resources of the target host:

$$\sum_{i \in migrated(t)} m_{ij}(t) \cdot vp_{ir} = 0, \forall j = 1..M, r = 1..R \tag{4.5}$$

The *Dirty Page Rate* of the VM is defined by variable $dpr_i(t)$. Equation 4.6 states how it depends n the requested by VM RAM:

$$dpr_i(t) = f(vr_{iram}), \forall v_i \in migrated(t) \tag{4.6}$$

where f is a custom defined functional dependency. For simplicity we assume f as a certain linear function. Consequently, the Equation 4.6 can be rewritten as Equation 4.7:

$$dpr_i(t) = C \cdot vr_{iram}, \forall v_i \in migrated(t) \tag{4.7}$$

where C is certain constant value.

If we further assume that amount of RAM and consequently *Dirty Page Rate* does not change during some period between two distinct timestamps then the amount of migrated RAM for VM i defined as $migr_i(t)$ to another physical host can be computed as follows:

$$migr_i(t) = \frac{bw(t) \cdot dt}{dpr_i(t)} \tag{4.8}$$

where $bw(t)$ is bandwidth speed rate between source and target hosts.

Additionally, each VM are assigned with an SLA. Each SLA has a priority level that defines priority of VM during its placement and resource provisioning to a target host. A VM with higher SLA level will have higher priority during resource provisioning over other VMs allocated to the same physical host.

Physical model of a cooling system

As we plan to use temperature data in our model, it is important to measure an impact of an input temperature on energy consumed for cooling a data center infrastructure. The work [43] makes an overview of impact of outside temperature to pPue rate. These results give us possibility to find a physical model cooling system, namely a function dependency between outside temperature, cooling mode and output *partial power usage effectiveness* (pPue) rate as follows:

$$pPue = f(Temperature, CoolignMode) \tag{4.9}$$

Although the results, given in this work, are related to a concrete cooling system (namely Emerson's DSE) the method used in the thesis can be easily extrapolated for each similar cooling system. Table 4.1 gives a concrete measurements of these three observed variables.

Outdoor ambient	**Cooling mode**	**pPue**
35 °C (90 °F)	Mechanical	1.30
21.1 °C (70 °F)	Mechanical	1.21
15.6 °C (60 °F)	Mixed	1.17
10 °C (50 °F)	Air	1.10
−3.9 °C (25 °F)	Air	1.05

Table 4.1: Efficiency of Emerson's DSE TM cooling system with an EconoPhase air-side economizer [43]. Return air is set at 29.4 °C

The proposed physical model will operate under two assumptions:

- dependency between pPue rate and temperature values for a distinct cooling mode is linear over a certain range of temperature values:

  $$pPue = a \cdot Temperature + b \tag{4.10}$$

  where a and b are parameters of the physical model for each cooling mode. It should be mentioned that each cooling mode has its temperature range within which it can be utilized, i.e., it is assumed that use of air cooling mode will be completely not applicable when outside temperature exceeds 20 °C.

- pPue of mixed mode is computed as an average value of pPue of mechanical and air modes:

  $$pPue_{mixed} = \frac{pPue_{air} + pPue_{mechanical}}{2} \tag{4.11}$$

- The temperature range within which cooling system utilizes mixed mode are defined manually with consideration to physical models for air and mechanical modes. According to the [43], we will define this range between $12\,^{\circ}\mathrm{C}$ and $18\,^{\circ}\mathrm{C}$.

Equation 4.12 states the expressions that operate under mechanical cooling model:

$$\begin{cases} 21.1a + b = 1.21 \\ 35a + b = 1.3 \end{cases} \tag{4.12}$$

The coefficients near a and b used in each equation are temperatures value and respectively pPue rates defined in the Table 4.1.

Equation 4.13 and Equation 4.14 show the derivation of a and b:

$$a = \frac{1.3 - 1.21}{35 - 21.1} = 0.0065 \tag{4.13}$$

$$b = 1.21 - 21.1a = 1.21 - 21.1 \cdot 0.0065 = 1.0725 \tag{4.14}$$

Thereby after solving the Equation 4.12 we can obtain the Equation 4.15 that describes mechanical cooling model:

$$pPue_{mechanical} = 0.0065 \cdot Temperature + 1.0725 \tag{4.15}$$

Similarly we can find physical model for air cooling mode. Equation 4.16 states the expressions that operate under air cooling model:

$$\begin{cases} -3.9a + b = 1.05 \\ 10a + b = 1.1 \end{cases} \tag{4.16}$$

Equation 4.17 and Equation 4.18 show the derivation of a and b:

$$a = \frac{1.1 - 1.05}{10 + 3.9} = 0.0036 \tag{4.17}$$

$$b = 1.05 + 3.9a = 1.05 + 3.9 \cdot 0.0036 = 1.064 \tag{4.18}$$

Once more after solving the Equation 4.16 we can obtain the Equation 4.19 that describes air cooling model:

$$pPue_{air} = 0.0036 \cdot Temperature + 1.064 \tag{4.19}$$

Thus we can already define the physical model of the whole cooling system:

$$pPue = \begin{cases} 0.0036 \cdot Temperature + 1.064 & Temperature \leq 12\,^{\circ}\mathrm{C} \\ 0.00505 \cdot Temperature + 1.06825 & 12\,^{\circ}\mathrm{C} < Temperature < 18\,^{\circ}\mathrm{C} \\ 0.0065 \cdot Temperature + 1.0725 & Temperature \geq 18\,^{\circ}\mathrm{C} \end{cases} \tag{4.20}$$

It should be noted that this physical model operates just for temperature values in Celsius and will have other parameters for values in Fahrenheit.

After defining the physical model of cooling system the Equation 4.21 that computes energy consumption for a given time range can be defined:

$$Energy_{overall} = Energy_{specs} \cdot pPue \cdot dt \tag{4.21}$$

where $Energy_{overall}$ is an overall data center energy consumption, $Energy_{specs}$ is an data center energy consumption without cooling costs defined by its specifications. It is assumed that temperature and therefore pPue are constants within given time range.

Equation 4.22 states the energy costs consumed by a data center within a given range:

$$Costs_{energy} = Energy_{overall} \cdot Price_{energy} \tag{4.22}$$

It is assumed that energy costs and temperature are constants within given time range.

Proposed Bayesian Network

Previous Sections introduced the formalization of different processes occurred in the cloud as well as rules applied to it. The approach that we will use to reduce costs of cloud infrastructure is the sophisticated VM placement and further power management of busy and empty physical machines.

In this Section we will present desired BN, detail the reasoning that led to its creation and criteria that in combination with MCDA are used for decision making.

The Section 4.4 introduced the issues occurred for the cloud provider such as different energy prices and weather conditions, frequent power outages, high SLA penalty, etc. Therefore, we can already define the idea of our approach for the VM placement problem: at each timestamp we will try to either allocate of currently scheduled or migrate of already hosted VM to a physical host that are running in a data center with the cheapest energy, the lowest pPue rate and at the same time with the lowest possibility of VM's SLA to be violated. Thereby, an elasticity manager that will follow this approach, will minimize both energy and penalty costs of a cloud provider.

Using this approach we will define an algorithm for resolving VM placement problem. Our algorithm consists of two phases: allocation of VMs and their periodic consolidation. The first phase can be seen as Bin-Packing problem [94] with variable bin sizes and prices. Taking into account that this problem is NP-hard, we will modify well-known heuristic Best-Fit-Decreasing that is shown to use no more that $\frac{11}{9} \cdot OPT + 1$ bins (where OPT is defined as the number of bins given by the optimal solution) [6, 99]. Our modification consists in an utility function that will be used to define the most appropriate physical host for a VM hosted in the cloud. We have established a Bayesian Network (BN) (see Figure 4.3) which nodes are built upon observations collected from the cloud infrastructure and defined factors and criteria. This BN calculates probabilities of various criteria based on the set of known facts. Further these probabilities are used in the MCDA as follows: each criteria g_i is defined as an utility weighting u_i that represents the relative importance of each attribute for the given decision problem. The overall utility $U(a)$

of an action a, namely allocation or migration of a certain VM to a target host, is then simply the weighted sum $\sum u_i g_i(a)$. A VM is allocated or migrated to a physical machine with the biggest value of the utility function.

Second phase implies the consolidation of existed VM placements based on a certain VM migration policy. After all candidates for migration are identified, we apply the algorithm from the first phase, in order to find target physical machine for a VM.

The proposed BN consists from the following set of nodes:

- *Same Data Center* node defines whether target PM is located at the same data center as the VM
- *VM Next RAM Workload* node represents predicted amount of RAM utilized by the VM on the next simulation timestamp. It is assumed that workload prediction is done by a custom prediction police. The set of prediction policies used during evaluation will be listed later in this Section.
- *VM Next CPU Workload* node represents amount of CPU of target physical machine on the next simulation timestamp that a VM would be consumed alone, e.g., if a VM will utilize 500 MIPS and the target host CPU capacity equals to 1000 MIPS, then this values will equal to 50%.
- *Other VMs Next CPU Workload* node represents amount of CPU of target physical machine on the next simulation timestamp that will be utilized by all VMs that have the same or higher level of SLA than current VM.
- *PM Overall CPU Workload* node represents current CPU workload of the target physical machine.
- *PM Next Overall CPU Workload* node represents estimation of the overall CPU workload of the target physical machine if the VM will be placed on it.
- *Country* node defines the country where data center is located.
- *VM RAM* node defines the amount of RAM of the VM defined in its specifications.
- *Bandwidth* node defines bandwidth rate between a physical machine where VM is placed and target physical machine. As was mentioned in the Chapter 2 it has direct impact on the migration time.
- *Dirty Page Rate* node represents assessment of possible Dirty Page Rate in the case if the VM will be migrated to the target host
- *Same SLA Next CPU Workload* node represents estimation of CPU workload of the target physical machine in the case if the VM will be placed to it. The node has two parents, namely *VM Next CPU Workload* and *Other VMs Next CPU Workload*.

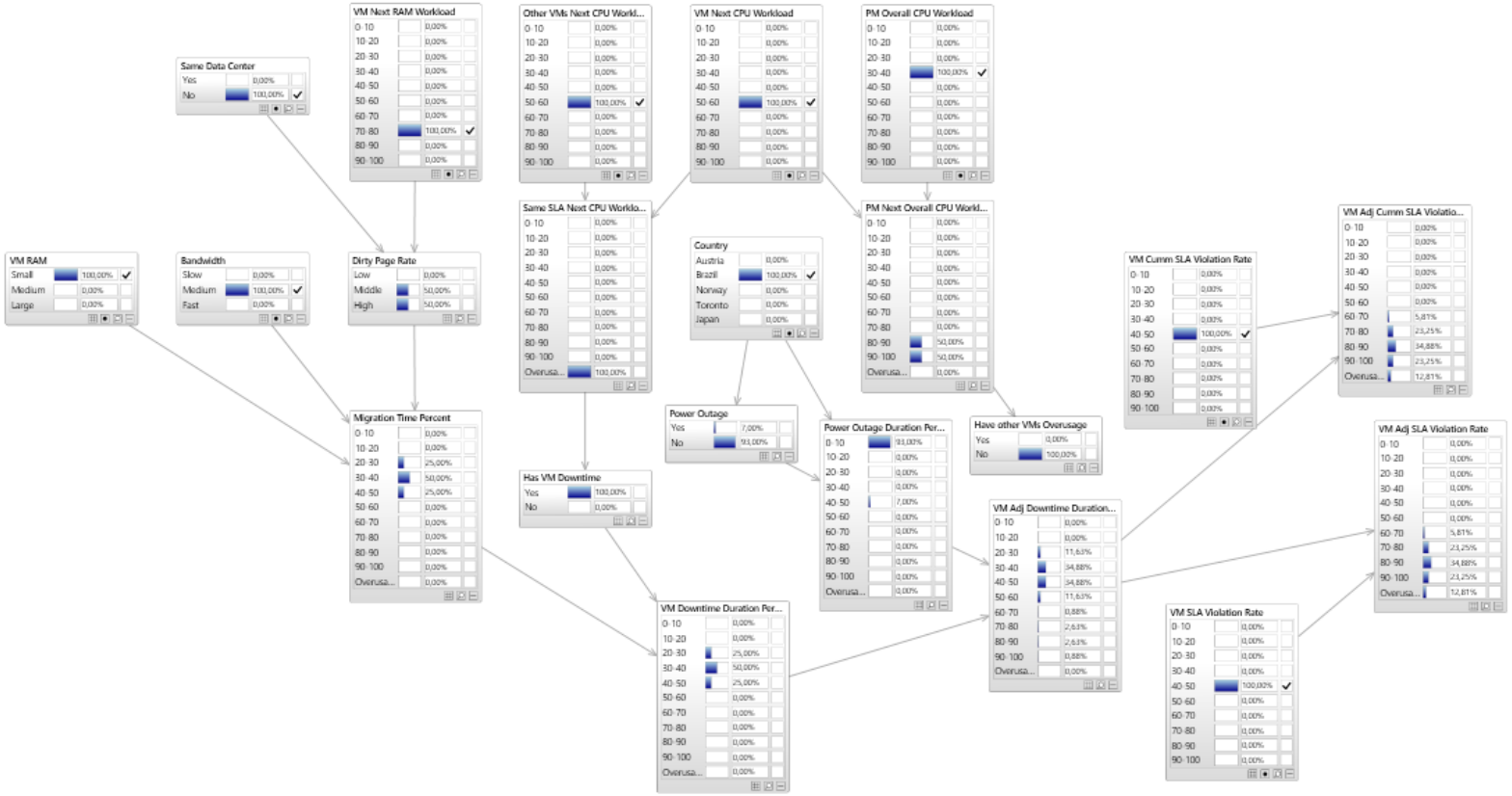


Figure 4.3: Proposed Bayesian Network with assigned evidences

- *Migration Time Percent* node represents a percent of time that takes a VM to migrate to the target host from the maximal allowed downtime for examined VM until the provider will pay a penalty, e.g., we assume that a VM SLA claims 99.9% of availability per months. It means that maximal allowed downtime equals to 43.2 minutes. Then the percentage value of *Migration Time Percent* is computed with respect of this maximal downtime. The same rules are applied for the nodes such as *VM Downtime Duration Percent*, *VM Adj Downtime Duration Percent*, *Power Outage Duration Percent*, *VM Cumm SLA Violation Rate*, *VM SLA Violation Rate*, *VM Adj Cumm SLA Violation Rate*, and *VM Cumm SLA Violation Rate*.

- *Has VM Downtime* node defines whether the VM will encounter a downtime if it will be placed to the target physical machine. The value of this node is computed depending on the values of *Same SLA Next CPU Workload*.

- *Power Outage* node represents a possibility of a downtime occurred in the target data center during following timestamp.

- *Power Outage Duration Percent* node represents percentage of downtime from the maximal allowed downtime of the VM.

- *Has other VMs overusage* node shows whether at least one VM placed on the target physical machine will not be fully provisioned with CPU resources and therefore will be in downtime.

- *VM Adj Downtime Duration Percent* node represents percentage of downtime with consideration of possible power outage time from the maximal allowed downtime of the VM.

- *VM SLA Violation Rate* node represents percentage of already encountered downtime in the current billing period from the maximal allowed downtime of the VM.

- *VM Adj SLA Violation Rate* node represents percentage of downtime considering current downtime and possible downtime after migration of VM to the target host from the maximal allowed downtime of the VM.

- *VM Cumm SLA Violation Rate* node represents percentage of duration of currently encountered downtime in the current billing period from the maximal allowed downtime of the VM, e.g., if a VM is in downtime at the moment this value will equal to the difference between current timestamp and the time of last VM availability, other it will be equal to 0.

- *VM Adj Cumm SLA Violation Rate* node similarly to *VM Adj SLA Violation Rate* represents the value of *VM Cumm SLA Violation Rate* with the consideration of possible downtime after migration of VM to the target host.

The SLA utilized in the model guarantees overall availability over defined period, e.g., monthly availability of 99.9%, and the maximal time of cumulated duration of distinct downtime, e.g., that every single downtime will not last more than 30 minutes.

The discretization of observed variables to 10 levels was done due to the fact that majority of hardware specifications benchmarks provide their data (e.g., power utilization of as physical machine) in respect of these 10 levels.

We have built the Bayesian Network around an example that assumes that a given cloud provider owns 5 data centers located in the following cities: Vienna (Austria), Oslo (Norway), Toronto (Canada), Tokyo (Japan), and Rio de Janeiro (Brazil). These countries are utilized as levels in the node *Country*. It should be noted that chosen concrete values do not affect the architecture of the network and in the case when new location should be added it can be simply done with updating of the node *Country* and distributions of its two child nodes *Power Outage* and *Power Outage Duration*.

Additionally, we have mentioned that the workloads utilized in the model are predicted using several prediction policies. In our further evaluations in the Chapter 6 we will use the following policies:

- *Last Workload Policy* supposes that next workload value will equal to current one.
- *Rate Workload Policy* supposes that next workload value differs from the current one to the static rate.
- *Trend Workload Policy* supposes that the values of workload follows a certain linear trend and tries to identify this linear dependency and to calculate appropriate prediction value.
- *Linear Regression Workload Policy* applies regression algorithms to the workload history values in order to predict possible future value of workload.

It should be noted that we use this four simple policies and not more sophisticated solutions because they are not the focus of our study in this thesis.

We will describe in details decision process about VM placement using an example illustrated on the Figure 4.3. We use the BN to calculate values of uncertain criteria. In this scenario, we evaluate the benefit of migration of VM with small amount of RAM to a host in the data center located in Brazil. The BN calculates that the probability of violation of overall availability as well as cumulative downtime duration is 0.1291 whereas their current violation levels are 40-50% from the guaranteed values.

The next step is a definition of criteria. We have defined the following criteria:

- violation rate of the overall availability ($g_1 \in [0; 1]$),
- violation rate of cumulative downtime duration ($g_2 \in [0; 1]$)
- power utilized by physical host with consideration to consumption of energy for its cooling (g_3)
- CPU workload of physical host if a VM will be migrated (g_4)
- migration time (g_5)
- energy price (g_6)

- possibility of downtime of other VMs on the host (g_7)

Table 4.2 represents the mapping of each criteria to a ratio scale $[0, 1]$ where 0 represents the worst value and 1 represents the best one.

g_1,g_2,g_5	
0-10 %	1
10-20 %	0.9
20-30 %	0.8
30-40 %	0.7
40-50 %	0.6
50-60 %	0.5
60-70 %	0.4
70-80 %	0.3
80-90 %	0.2
90-100 %	0.1
violation %	0

g_4	
0-10 %	0.125
10-20 %	0.25
20-30 %	0.375
30-40 %	0.5
40-50 %	0.625
50-60 %	0.750
60-70 %	0.875
70-80 %	1
80-90 %	0.66
90-100 %	0.33
violation %	0

g_7	
yes	0
no	1

Table 4.2: Criteria mapping to the ratio scale $[0; 1]$

Equation 4.23 states the calculation of g_3:

$$g_3 = (P - p * pPue)/P \tag{4.23}$$

where P is a constant that defines maximal power that can be utilized by each host with consideration of cooling energy consumption, p is host power consumption, $pPue$ is pPue rate for current data center where physical host is located.

Equation 4.24 states the calculation of g_6:

$$g_6 = (E - e)/E \tag{4.24}$$

where E is a constant that defines maximal energy price over all data centers, e is current energy price in the target data center.

At the end in order to utilize BN for decision analysis and evaluate triggered actions we use the utility function $U(a)$ defined previously.

Example. In the Table 4.3 we have specified utility weightings to the criteria for the scenario given above on Figure 4.3. The weight for each single utility is defined in the header of the table. Each column refers to the criteria specified in its header. The SLA violations as well as energy price have been considered as the most important criteria. In the table the rows represent the physical host's characteristics: its location, bandwidth between target host and current host where VM is allocated, host CPU workload, etc. The values of each criteria are taken considering the rules defined previously. To obtain the values for the uncertain criteria (what host is the best for VM placement) we get the mean values from the BN when we enter the specified input vectors and then transform these using the rules. The column with **total** value contains that values that are calculated as weighted sum of each criteria and its weight. In this example,

target host (location, same data center, bandwidth, host workload, vms workload)	g_1**(3)**	g_2**(3)**	g_3**(2)**	g_4**(0.5)**	g_5**(3)**	g_6**(2)**	g_7**(2)**	**total**
(Brazil, no, medium, 30-40%,50-60%)	0.2	0.7	0.3	0.7	1	0.66	0.8	**9.37**
(Brazil, no, medium, 10-20%,10-20%)	0.2	0.7	0.4	0.7	1	0.825	0.8	**9.9**
(Japan, no, high, 70-80%,70-80%)	0.6	0.7	0.15	1	0	0	0.2	**5.3**
(Japan, no, high, 0-10%,50-60%)	0.9	0.9	0.45	1	1	0.75	0.2	**10.9**
(Austria, yes, high, 20-30%,60-70%)	0.8	0.2	0.35	1	1	1	0.6	**10.0**
(Austria, yes, high, 60-70%,20-30%)	0.4	0.6	0.2	1	0	0	0.6	**5.7**
(Canada, no, medium, 80-90%,80-90%)	0.3	0.3	0.1	0.7	0	0	0.5	**3.85**
(Canada, no, medium, 0-10%,0-10%)	0.9	0.8	0.45	0.7	1	0.75	0.5	**11.35**
(Norway, no, high, 20-30%,40-50%)	0.8	0.9	0.35	1	1	1	0.3	**11.2**
(Norway, no, high, 10-20%,30-40%)	0.9	0.9	0.4	1	1	0.825	0.3	**11.25**

Table 4.3: An example of decision-making analysis using proposed approach. Criteria values are obtained from the BN illustrated on Figure 4.3

the biggest total utility value refers to the data center in Canada that supposes to migrate a VM to it.

CHAPTER 5

Architecture and Implementation of Simulation Framework

The Chapter 1 introduced the key features of the simulation framework called Cloudnet that allows its use for evaluations of cloud infrastructures among which a possibility to simulate live VM migrations between Geo-distributed data centers, usage of realistic weather data, scheduling of power outages, simulation of various cooling models and resources utilization workloads, etc.

The following Chapter will describe in detail the design, architecture and implementation of the Cloudnet framework, which has been developed with deep consideration of common Cloud Computing patterns introduced in the Chapter 2. Cloudnet solution allows a cloud provider to test its infrastructure in repeatable and controllable way, to find and avoid performance bottlenecks, evaluate different cloud management scenarios under varying geo-aware, load and pricing conditions. On the other side, it should be mentioned that although the simulation is very powerful experimental methodology, it should be utilized with great understanding of the running process and is applicable just in certain number of use cases. If necessary other methodologies such as benchmarking, in-situ, emulation [42] should be strongly considered.

Cloudnet is a reliable, flexible, fast simulation framework written in JAVA with the possibility to perform native invocations of the R language [79] using external running R environment. A part of it is hardly concentrated on the usage of BNs for the decision-making analysis utilized in this Thesis.

Cloudnet was designed and implemented on the basis of loose-coupling paradigms [14] that assumes decoupling of different simulated components from each other and their communication through the *Message-oriented middleware* (MOM) [18]. The dependent components of the system interact with each other through interfaces that allows simple extension of almost each part of the framework and highly configurable possibilities that captures many simulation use cases.

5.1 Architecture

The framework consists of the following components:

- *Simulation core*: different implementations of simulation engines as well as simulation clock that shows the actual simulation time. The actual simulation time can be polled by any object but can only be set by the simulation engine responsible for this.
- *Cloud domain*: the classes that represent all main cloud entities, models and interfaces.
- *Various physical models*: Implementation of different physical models that are utilized during runtime of cloud infrastructure.
- *MOM*: A set of classes that allows loose couple communication between different components of the simulated cloud.
- *Monitoring infrastructure*: attachable observers that monitor ans make snaphots of cloud entity states and log them into different output formats for further analysis.
- *R Wrapper*: service wrapper for execution of commands in the external R environment.
- *Bayesian Networks package*: a set of classes that provides various possibilities to work with BNs.
- *Data-collecting utilities*: a set of utilities that are responsible for downloading and transformation of various weather data from different Web services.
- *Logging package*: Highly configurable logging capabilities for tracing various information about execution of the simulations.

The core classes as well as cloud interfaces are assumed to a necessary part of the framework that denotes their irreplaceability.

Simulations

The *Simulation package* contains the components that are responsible for the simulation of the cloud infrastructure: interface of a cloud engine and its two implementations, simulation clock and interfaces for abstract scheduler that is responsible for generation of customer requests and simulated entity which can be utilized by simulation engine.

Figure (add figure) shows a UML diagram of the package.

The interface `SimEngine` allows to perform simulations of the processes in the given cloud environment. Current implementation of the framework contains two concrete implementations represented in the Table 5.1.

As was mentioned above, the class `SimClock` is utilized by a simulation engine in order to save the actual simulation time. Further this information is used by each component that implements interface `Simulated` to simulate its execution between two different time instants.

Class Name	Responsibility
`SimEngineSimple`	Simulates predetermined number of steps
`SimEngineDates`	Simulates predetermined time range between two dates

Table 5.1: Concrete implementations of the `SimEngine` interface

Listing 5.1 exemplifies one execution step of the simulation engine. Initially it schedules customer requests, then increases simulation time and after that simulates execution of the attached simulated entity, e.g, a cloud in this case.

Listing 5.1: One execution step of the simulation engine

```
1 // schedule
2 scheduler.schedule(cloud, clock);
3 // increase time
4 clock.add();
5 // simulate cloud execution
6 cloud.simulateExecution();
```

Cloud domain

The *Domain Core package* contains main cloud entities and interfaces that they use during their life-time.

`CloudEntity`, the base class for each simulated entity, implements the interface `Simulated` utilized by a simulation engine. It is implemented using *Abstract Factory pattern* [21] that provides a simple way to encapsulate a group of individual entity implementations.

Cloud entities in the Cloudnet are the *containers* [24] for various physical models of the processes running within them. For example, a `Datacenter` has a `CoolingModel` that can be utilized to get power usage effectiveness of its cooling system. Often containers are implemented as concrete classes, whereas physical models are represented by interfaces that allows simple extensibility of the framework, e.g., `CoolingModel` has two following implementations which physical models where described in details in the Section 4.4: `AirCoolingModel` and `MechanicalCoolingModel`.

Table 5.2 gives an overview of main cloud entities and models.

`Cloud` contains the list of data centers and references to `ElasticityManager` and `Monitoring System` attached to it. The simulation of its execution consists from the following stages:

- Waiting and performing of changes triggered by the `ElasticityManager`
- Simulation of execution of cloud infrastructure
- Calculation of current costs
- Monitoring

It means that `Cloud` does not perform any changes without a designation of the `ElasticityManager` that completely manages all actions applied to the cloud infrastructure.

Class Name	Responsibility
`Cloud`	Represents abstract implementation of Public Cloud [14] that can be customized for each concrete cloud service model such IaaS, PaaS, etc.
`Datacenter`	Represents a geo-aware data center
`Pm`	Represents a physical machine (host) that are located in a data center and can host VMs
`PmSpec`	Represents concrete specifications of a physical machine such as amount of resources, power utilization
`Vm`	Represents a VM
`VmSpec`	Concrete specifications of VM that contain information about its maximal amount of resource usage
`ElasticityManager`	Represents an interface for ElasticityManager described in the Chapter 2 that is used to perform management of cloud infrastructure
`MonitoringSystem`	Represents an interface for an system that can be attached to a cloud for monitoring purposes
`Sla`	Interface for an SLA which concrete implementation should be provided for each cloud service model
`SlaLevel`	SLA's priority level
`WorkloadModel`	Interface that is used to simulate various application workloads described in the Chapter 2
`CoolingModel`	Interface that is used to get power usage effectiveness of a cooling system used in a data center
`EnergyPriceModel`	Interface that is used to simulate energy price with consideration a data center location. It uses combination of `WorkloadModel` with a specific factor in order to simulate various prices
`PowerOutageModel`	Interface that is used to simulate power outages with consideration a data center location
`TemperatureModel`	Interface that is used to simulate outside temperature with consideration a data center location
`Location`	Interface that encapsulate all necessary information regarding certain geo-aware location such as city, country, timezone, different statistics, etc.
`Provisioner`	Interface that represents the strategy of resource provisioning

Table 5.2: Cloud Domain classes and interfaces

The framework emphasis on accuracy of calculations therefore almost all of them are per-

formed using the type `java.lang.Long` (or the primitive type `long` where it is possible), because it avoids any rounding issues. An example is given in the Listing 5.2 where part implementation of helper class `Size` is shown.

Listing 5.2: Implementation of different memory sizes that avoids rounding issues

```
1 public final static long Bit = 1;
2 public final static long KBit = Bit * 1024;
3 public final static long MBit = KBit * 1024;
4 public final static long GBit = MBit * 1024;
5 public final static long Byte = Bit * 8;
6 public final static long KB = Byte * 1024;
7 public final static long MB = KB * 1024;
8 public final static long GB = MB * 1024;
9 public final static long TB = GB * 1024;
```

MOM

The *MOM package* introduces a layer of distributed communication for sending and receiving messages between different parts of the cloud infrastructure and thereby allows to insulate them from details of implementation of each other.

`MessageBus` in Cloudnet is a communication broker that implements *publish–subscribe pattern* [37]. With such scheme of interaction, subscribers register their interest in certain pattern of messages, and are subsequently notified of such messages generated by publishers.

Each entity in the cloud that extends `CloudEntity` can communicate with each other as it already contains a reference to the `MessageBus`.

`Message` is a base for each concrete message in the framework. Each message implementation contains specific information about notified event. Messages are hardly utilized, for example, for the communication between `ElasticityManager` or `Scheduler` and `Cloud`.

The examples of concrete message types are presented in the Table 5.3.

Class Name	Responsibility
`PmStartMessage`	Notifies that certain physical machine should be started
`PmStopMessage`	Notifies that certain physical machine should be stopped
`VmAllocationMessage`	Notifies that certain VM should be allocated to the specified physical machine
`VmMigrationMessage`	Notifies that the live migration of the specified VM should be started
`VmSchedulingMessage`	Notifies about new VM scheduled by a customer

Table 5.3: Examples of Cloudnet message types

System Monitoring

The *Monitoring package* contains a set of classes that allows collecting and storing different states of the cloud entities. The extensibility of the framework allows different implementations of the monitoring systems such as passive that simply collects measurements from attached entities as well as active that additionally publishes different messages to a cloud if some problems occurred (also known as *Watchdog* [14]).

Currently the framework offers the `PassiveMonitoringSystem` (and its asynchronous implementation `PassiveAsyncMonitoringSystem`) that can collect states of a cloud, data center, physical machine or VM and save them into necessary output format using a concrete implementation of the interface `HistoryWriter`.

Service models

Architecture of Cloudnet is applicable for the usage of each known cloud service model. Currently only IaaS model is available.

The *IaaS package* contains concrete implementation `IaaSCloud` that simulates execution of entities with such consideratiosn to this service model as allocation and migration of VMs, power management of physical machines, etc.

Specifications

In order to provide realistic results during evaluation of simulation use cases the framework provides a set of real hardware and VM specifications.

The *PM package* contains several implementations of specification for physical machines provided by the benchmark *SPECpower ssj2008* [87]. Each specifications except of resources (CPU, RAM, etc.) available on a physical machine defines its power utilization ratio in respect of target CPU load. The example of such specifications [1] is shown for on the Figure 5.1.

The *VM package* as well as *SLA package* were implemented around VM specifications and SLAs of *Microsoft Azure* cloud service [10]. Current implementation assumes that the VM specification is stable and cannot be changed during execution of simulation.

Application workloads

Cloudnet offers a wide range of application workloads. A part of them were already described in details in the Section 2.2. Each implementation utilizes own algorithms and mathematical functions in order to simulate desired behaviour, e.g., `PeriodicWorkload` uses function $sin(x)$ with a specific factor that creates effect of periodicity. Also its workload is pseudo-randomized with defined deviation of the base value. `UnpredictableWorkload` also uses pseudo-randomization via class `Random` from the `java.util` package for generation of unpredictable workload.

[1] `http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00336.html`

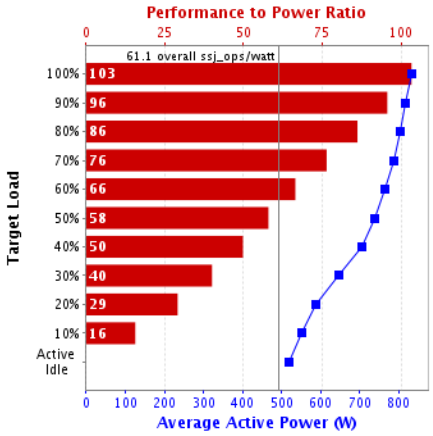


Figure 5.1: Example of power utilization under different CPU load for HP ProLiant DL580 G3 according to the benchmark *SPECpower ssj2008* [87]

Except of standard workloads the framework contains a time zone specific workload container `DayNightWorkloadModel`. It is possible to specify two different workload models, one of them operates daily and another one nightly. It is very useful for simulation of application workloads in different locations of the globe.

Additionally, the *Workload package* contains the *Prediction sub-package* that provides implementations for various prediction strategies. All of them use `WorkloadHistory`, an object that stores history of the workload's observations. Prediction of workloads can be utilized for more accurate and sophisticated management of the cloud by elasticity manager. Due to the fact that time-series forecasting is not a focus of current research just the most obvious strategies, listed in the Table 5.4 , were implemented.

Class Name	**Responsibility**
`LastWorkload`	Takes the last encountered workload and simply returns it
`RateWorkload`	Takes the last encountered workload, multiplies it by the specified rate and returns resulting value
`Trend`	Takes two last workloads and builds a linear function of dependency between workload and time; afterwards it uses this function to return a result substituting the input future time stamp as an argument of the function
`Simple Regression`	Observes the workload history provided by a VM, builds a simple regression model based on these history observations and returns the value predicted by the model

Table 5.4: Overview of workload prediction strategies

Temperature models

There are two implementations of the abstract `TemperatureModel` available in the framework.

First of them is `GeneratedTemperatureModel`. It generates synthetic temperature data using values of the attached `WorkloadModel`. The output of this model can be used if the simulations should be performed under defined initially known temperature or for short time intervals within a part of the year where average temperature values do not change so much day by day.

`RealTemperatureModel` is the model that reads temperature data preloaded in advance into text files with specific format as follows:

```
// format: Date,Timestamp,Temperature
2012-12-30 23:00:00,1356904800,18.89
```

Obtaining of real temperature time series can be done using each web service suitable for this. Cloudnet framework contains utility classes that allows to get data from the service Forect.io [2]. It is a free weather service which API provides data for most locations on the globe for a specific time, past or future [3].

`WeatherFether` is used to fetch data from the Forecast.io in JSON format [49] for a specified location, start and end dates. Listing 5.3 shows its usage for obtaining temperature data for Vienna for the whole 2013^{th} year. First, we specify start and end dates (lines $1-4$). After that we define a geo-coordinates that refers to a city of Vienna (line 5) and, finally, we execute `WeatherFether` by calling it with necessary parameters (line 6).

Listing 5.3: Fetching of the temperature in Vienna for the whole 2013^{th} year

```
1 Calendar start = new GregorianCalendar(TimeZone.getTimeZone("GMT"));
2 start.set(2013, Calendar.JANUARY, 1, 0, 0, 0);
3 Calendar end = new GregorianCalendar(TimeZone.getTimeZone("GMT"));
4 end.set(2014, Calendar.JANUARY, 1, 0, 0, 0);
5 String vienna = "48.2000,16.3667";
6 WeatherFether.fetchData(vienna, start, end);
```

In order to convert obtained data into compatible for `RealTemperatureModel` format, `FormatConverter` should be further utilized.

Cooling models

The *Cooling package* contains set of implementations of the `CoolingModel`. Physical models for a cooling infrastructure were described in detail in the Section 4.4. There were two models namely Air and Mechanical that are reflected in the classes `AirCoolingModel` and

[2] `http://forecast.io/#/f/35.6832,139.8089`

[3] `https://developer.forecast.io/docs/v2`

`MechanicalCoolingModel` respectively. `MixedCoolingModel` implements general approach that introduces mixed mode where two models can be put together.

Power outage models

The *Power outage package* contains a set of models that introduces simulation of power outages in data centers.

`ProbabilityPowerOutageModel` utilizes function $sin(x)$ for simulation of power outage events with initially defined overall duration and number of occurrences over the year. Such implementation is very convenient as it provides the usage of commonly published statistical indices *CAIDI* and *SAIFI*.

R Wrapper and Bayesian Networks

Cloudnet framework provides the possibility to invoke different scripts in *R Language* from JAVA code using external R environment.

`R` is a *wrapper* [21] that encapsulates execution of commands and obtaining of results sent from R environment via callbacks.

The opportunity to use *R Language* opens access to wide range of mathematical models, algorithms and packages supported by it.

The object of interest was a package *gRain* [41] for probability propagation in Bayesian Networks.

Listing 5.4 shows the usage of this package for instantiation of the famous BN *Asia* [58] using R language.

Listing 5.4: R script illustrating instantiation and quering of the BN *Asia* using gRain package

```
1  yn <- c("yes", "no")
2  asia <- cptable(~ asia, values = c(1, 99), levels = yn)
3  t.a <- cptable(~ tub + asia, values = c(5.0, 95, 1, 99), levels = yn)
4  s <- cptable(~ smoke, values = c(5,5), levels = yn)
5  l.s <- cptable(~ lung + smoke, values = c(1, 9, 1, 99), levels = yn)
6  b.s <- cptable(~ bronc + smoke, values = c(6, 4, 3, 7), levels = yn)
7  x.e <- cptable(~ xray + either, values = c(98, 2, 5, 95), levels = yn)
8  d.be <- cptable(~ dysp + bronc + either, values = c(9, 1, 7, 3, 8, 2, 1, 9),
9   levels = yn)
10 e.lt <- ortable(~ either + lung + tub, levels = yn)
11 bn.asia <- grain(compileCPT(list(asia, t.a, s, l.s, b.s, e.lt, x.e, d.be)))
12 # evidence example, conditional
13 bn1 <- setEvidence(bn.asia, c("asia","either"), c("yes", "yes"))
14 querygrain(bn.asia, nodes=c("lung", "bronc"), type="joint")
15 # marginal probability example
16 querygrain(bn1, nodes=c("asia", "tub"), type="marginal")
```

The listing starts with the definition of the levels that will be used in all nodes further (line 1). After that we define each node of the network separately and assign a variable for each of them (lines $2-10$). Next we instantiate a network object by compiling all nodes into one single network using function `grain` (line 11). After that the listing provides two example of network querying: one for the computation of conditional probability for the nodes *lung* and *bronc* (lines $13-14$), and the other one for the computation of marginal probability for the nodes *asia* and *tub* (line 16).

Cloudnet provides a *BN package* that contains a set of classes for design and usage of BNs. They are formally wrappers over R expressions used in the *gRain* package to instantiate and perform querying of nodes. Table 5.5 gives a short overview of these classes.

Class Name	**Responsibility**
`Network`	Represents a BN that contains a set of nodes with links between them. It is responsible for preparing and sending of expressions into *R Engine* using R Wrapper
`AsyncNetwork`	An asynchronous implementation of `Network`
`Node`	Represents a regular node in the BN
`Link`	Represents a link between two nodes
`Distribution`	An abstract base for node distributions
`DiscreteDistribution`	Concrete implementation of discrete distribution
`NodelinkType`	Specifies link types of a node such as *NONE*, *AND*, *OR*
`ProbabilityType`	Specifies probability during querying of node's inferences

Table 5.5: Overview of classes for BNs usage

Listing 5.5 shows the usage of the *BN package* for creation of the BN presented in the Section 4.2.

Listing 5.5: Creation of the BN from Section 4.2 using the *BN package*

```
1  String[] levels = new String[]{Levels.YES, Levels.NO};
2  Node res = new Node(Nodes.RESOURCES, levels);
3  Node workload = new Node(Nodes.WORKLOAD, levels);
4  Node migrate = new Node(Nodes.MIGRATE, levels,
5  new Integer[]{9, 1, 2, 8, 0, 1, 0, 1});
6  Network network = new Network("allocation");
7  network.setR(r);
8  network.addLink(res, migrate);
9  network.addLink(workload, migrate);
10 network.init();
```

First, we define levels for the nodes of the network (line 1). After that we create three following nodes:

- *Resources* using boolean levels (line 2)
- *Workload* using boolean levels (line 3)
- *Migrate* using boolean levels, but already with definition of appropriate CPT (lines $3-4$)

Further we create the network (line 5) and set the R Wrapper adapter object to it (line 7) allowing communication with R environment. After that we specify structure of the network by setting links from the *Resources* node to the *Migrate* node (line 8) and the *Workload* node to the *Migrate* node (line 9). Finally, we initialize the network (line 10) that also establishes it in the R environment workspace for further usage.

Elasticity

The *Elasticity package* was intended for the various implementations of elasticity managers.

The framework provides implementation for the BN described in the Chapter 4 called `ElasiticityManagerMCDA`. In addition, three elasticity managers, presented in the Table 5.6, that use well-known *First-Fit Decreasing* heuristic algorithm [94] were added into the solution.

Class Name	Responsibility
`FirstFitUnefficient`	Elasticity manager that represents the worst case of power consumption as it keeps all physical machines always in running state. Also it uses a First-Fit Decreasing heuristic algorithm for the placement of VMs. Additionally, it does not consider the requested resources by a VM and assumes that it will utilize all resources defined by its specification.
`FirstFitPessimistic`	Elasticity manager that behaves the same as `FirstFitUnefficient`, but it switches off unused physical machines
`FirstFitOptimistic`	Elasticity manager that behaves the same as `FirstFitPessimistic`, but it try to optimize the placement of VMs considering amount of resources requested by them

Table 5.6: Elasticity managers using First-Fit Decreasing heuristic

Each elasticity manager performs three main tasks: power management of physical machines, allocation of scheduled VMs and their further consolidation based on the defined `VmMigrationPolicy`. The `VmMigrationPolicy` defines the conditions under which a VM should become a candidate for a VM migration. We pay attention on the fact that it just detects the candidates for a possible VM migration, but not force to migrate a certain VM to another physical host. Table 5.7 gives an overview implementations of `VmMigrationPolicy` presented in the Cloudnet.

Class Name	Responsibility
`Always`	It always consider each VM that are not currently migrated to another physical machine as a candidate for VM migration.
`Downtime`	It refers a VM to a candidate in the case if it encountered a downtime on a previous simulation step and currently are not migrated
`ShortDowntime`	It behaves the same as `Downtime`, but also consider possible short-time downtimes that caused, e.g., by power outages

Table 5.7: Overview of implementation of VM migration policies

5.2 Implementation

Cloudnet solution was implemented using Java language, version 8 (Java SE8). The implementation is not compatible with previous versions of Java, but this choice in favour of version 8 was done due to its new very powerful features such as lambda expressions, new possibilities of concurrent parallel execution and performing different useful operations with collections [45]. Thereby the framework becomes more compact, stable and faster.

Additionally, the framework depends on certain third-party libraries. Their usage is applied just whenever it makes sense to improve the implementation performance and to keep development effort low. Table 5.9 contains the list of third-party libraries that are used in the implementation. All of them are published under open source license.

Library	Version
JRI Engine	0.9.7
Apache Commons Math	3.3
Apache Commons IO	1.3.2
Apache Commons CLI	1.3
Log4J	1.2.17
JSON.simple	1.1.1
Google Guava	17.0

Table 5.8: Third-party software used in Cloudnet

The package JRI Engine provides a possibility to run commands written in R Language inside Java applications that allows to design and use BNs inside of Cloudnet. The library Apache Commons Math is used instead of regular Java Math package whenever it is possible as it much faster performs mathematical and statistical computations [72] that, for example, hardly utilized in the implementation of application workloads. Apache Commons IO package provides convenient opportunities of file processing that are used, e.g., in the `WeatherFether`. Another package that was utilized in the same class is JSON.simple. Temperature data obtained from the Web service Forecast.io are available just in JSON format. The JSON.simple library helps

to perform error-prone process of parsing of JSON documents stable and fast. Apache Commons CLI was used during implementation of highly configurable command line applications used for evaluations described in the Chapter 6. Log4J is commonly used library for logging that provides a lot of convenient features and possibilities to write logs into almost any output source.

The *BN package* depend also on the R that should be additionally installed on the machine where simulations are performed. The used version of R is 3.0.2 or higher. Table denotes the R packages used in the implementation.

R Package	Version
gRbase	1.7-0.1
gRain	1.2-3
Rgraphviz	2.8.1

Table 5.9: Third-party software used in Cloudnet

It should be mentioned that the package Rgraphviz cannot be installed from the official *CRAN* repository, instead of that a repository called *Bioconductor* should be used as shown in the Listing 5.6. In this listing we, first, add new package repository to R environment (line 1) and then install new package named *Rgraphviz*.

Listing 5.6: Installation of the R package Rgraphviz

```
1 source("http://bioconductor.org/biocLite.R")
2 biocLite("Rgraphviz")
```

Implementation issues

Mostly the development process of Cloudnet solution was held as expected. Issues that revealed themselves were related to the usage of R commands from Java code, configuration of their correct invocations and the single-threaded nature of the library JRI Engine.

Limitations of R

R has a lot of crucial advantages that became fundamentals of choosing it as the algorithmic platform for the Thesis. It is the most comprehensive statistical analysis package available nowadays. It includes a lot of statistical models, tests, algorithms, and, additionally, provides simple and understandable syntax for managing and manipulating data. Besides of that it is free, open source and has huge growing community [2]. But there are also some disadvantages become apparent during the usage of R. They concerns performance aspect, which is manifested in the inability to quickly handle big amount of complex requests as well as perform them efficiently in parallel [91]. The performance of R is the known issue, but it is often neutralized by the opportunities of this language.

Another issue that concerns the overall performance of the framework was a fact that JRI Engine library allows to run R inside of Java application only as a single thread. It means that

if the second instance of the class `Rengine` from this package will be created it cannot be instantiated. Therefore the `Rengine` can be considered as a *Singleton* object [21] and it is generally complex issue for the use in the cases of parallel programming.

First implementation of the Cloudnet prototype performed all calls into R in a single-threaded sequential manner. Considering all problems described above the execution of the simulation was obviously slow and tedious and therefore required urgent improvements. As was identified most of the time consumed for the invocation was consumed not by R, but for sending calls as well as receiving results via callbacks.

Recall the algorithm of the usage of the BN described in the Section 4.4. According the algorithm `ElasiticityManagerMCDAViolations` performs assessment of all physical machines for each VM hosted in the cloud during each simulation step. After all rates are calculated the host with the higher score wins and the VM decided to be migrated to it in the case if it has not been placed there yet. Formally speaking, if there are n VMs and m physical hosts the algorithm requires nxm steps in order to calculate all ratings. Besides that the placement of certain VM reserves a part of space on the target host that impacts the following decisions regarding placement of other VMs to the same host. This implies the theoretical impossibility to perform seeking of the target host for each VM in parallel, but it does not means that the evaluation of the physical machines for a separate VM cannot be done so.

The enhancement was accomplished via introduction of MapReduce [19] paradigm. The "Map"step consists from the evaluation of each physical hosts for a single VM. The "Reduce"step collects all results and choose the most appropriate target host.

The usage of MapReduce is applied in the implementation of BN called `AsyncNetwork` that is able to be queried in parallel. For the querying purposes `AsyncNetwork` creates a copy of the base BN in the R environment and performs all calls already to it. It should be also mentioned that after the calculations are complete, the copy is released from the R environment in order not to imply memory leaks.

The described improvement allows to perform simulation much faster.

CHAPTER 6

Evaluation

In this Chapter, we will discuss an evaluation of the approach for sophisticated VM placement based on BN and MCDA proposed in the Section 4.4. We define a set of reasonable scenarios that tend to identify strengths and weaknesses of the proposed algorithm.

The evaluation will be built around comparison of proposed method with two near-optimal First-Fit-Decreasing [94] heuristics for Bin-Packing problem and the approach when no power management is in place as the worst-case performance baseline.

During evaluation the most attention will be paid to the following aspects:

- *Power consumption and cost savings:* How much power energy are used? How does it affect on the costs?
- *Quality of Service:* What level of service availability and reliability can be guaranteed?
- *Migration overhead:* Although we do not consider any migration costs except the migration time between two hosts, we will count the number of overall migrations to understand possible future improvements of proposed algorithm.

The evaluation implies step-wise simulation of a cloud environment with the running VMs scheduled to it. The Cloudnet framework is used for the simulation purposes. Also it provides an ability to collect detail information about simulation execution as well as to obtain final results. The following sections will describe in detail the process of preparation of simulation environment and further execution and analysis of the scenarios.

6.1 Preparation of simulation environment

Initially we defined several evaluation use cases that tend to test different aspects of the proposed algorithm. Each use case was configured and evaluated using the Cloudnet framework. The detail overview and analysis of these use cases will be done in the following Section. This Section describes preparation of a simulation environment for the experiments.

Simulation environment

Although the implementation of Cloudnet guarantees full repeatability of results, they still may depend on environment where experiments are performed. Hence, we didn't change the runtime environment and used just one machine for all experiments:

- Our test machine was an Intel Core i5 U430 desktop PC with 4 GB of RAM. The machine was running on Microsoft Windows 8 Pro.
- Oracle JRE (Java Runtime Environment), version x64 1.8.05 for Windows, was installed on the machine. All code was compiled using Oracle Java compiler `javac`, version 1.8.05.
- R, version 3.1.1 x64 for Windows, was installed on the machine. Additionally, JRIEngine, version 0.9.7, was utilized for the interaction between Cloudnet and R runtime environment. A set of installed R packages was described in details in the Section 5.2.

Input parameters

One of the important goals of each experiment is an applicability of algorithm in the real world [42]. Furthermore, our evaluation should prove or refute the ability of the proposed approach to use additional information about cloud domain such as temperature data, or information about energy prices in different locations, in order to make more efficient decisions concerning placement of VMs in the cloud. Due to this reason real-world data were used during experiments as much as possible. Though the set of possible input parameters and conditions can be potentially infinite, we tried to define realistic and representative set of them:

- *Geo-distributed data centers:* We defined 5 locations on the globe for the location and simulation of data centers: Rio de Janeiro (Brazil), Toronto (Canada), Oslo (Norway), Vienna (Austria), Tokyo (Japan). Selected locations allow to test model considering combination of various input parameters as temperature, energy price, time zone, power outage statistics. For example, although most of the time it is hot in Rio de Janeiro that on the one hand will cause higher cooling costs, but on the other this location offers competitive energy prices that can be a good reason for placement VM there.
- *Electricity prices:* Electricity prices for the locations selected previously were defined using statistics in [1]. Additionally, we considered that Oslo, Tokyo and Vienna offer the reduction of the electricity price at night in average from 11 PM to 8 AM. Additionally, information about electricity prices in Austria was taken considered prices for business customers provided on the Web site of Wien Energie [86].
- *Power outage statistics:* We obtained a lot of power outage statistics for recent years for the selected locations using [22, 23, 88]. For example, Figure 6.1 shows a measure of total electric grid outage duration, called *SAIDI* (System Average Interruption Duration Index), across a sample of countries in the world in 2013. SAIDI represents the total number of

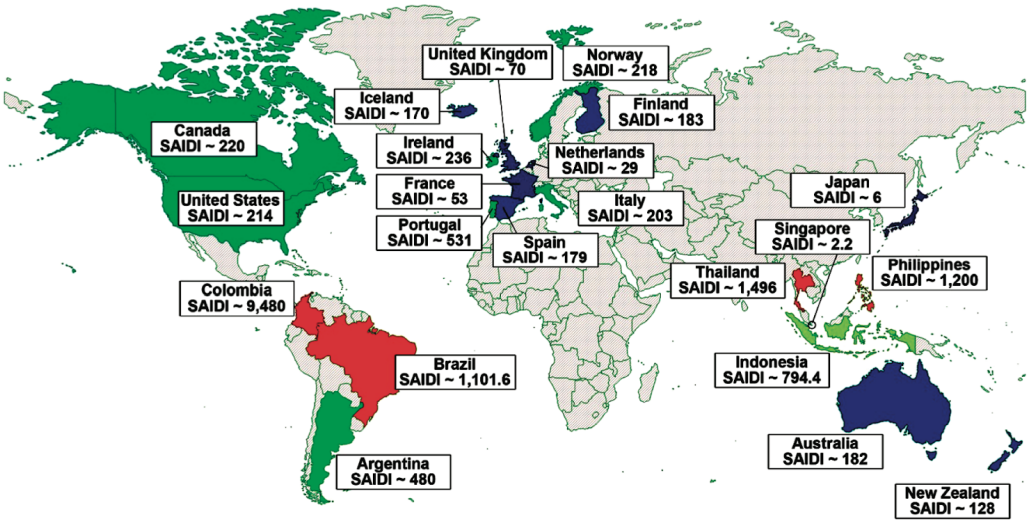


Figure 6.1: International Power Outages Comparison [88]

minutes per year of outage experienced by an average customer [88]. Further these statistics with combination of *CAIDI* (Customer Average Interruption Duration Index) were utilized in the `ProbabilityPowerOutageModel`, one of the power outage simulation models implemented in the Cloudnet framework.

- *Temperature data:* We obtained real temperature data from the Web service Forecast.IO [84] for the whole experiment interval with the granularity of one hour for all previously defined locations.
- *Cooling models:* There are three cooling models used in the experiments: Air, Mechanical and Mixed. Their physical models were described in details in the Chapter 4.
- *Physical machine power specifications:* We used data of *SPECpower ssj2008* [1] to define the power specifications of a physical machines in data centers.
- *VM resource utilization:* We simulated the resource utilization of VMs using various application workloads described in the Section 2.2. Moreover, we took technical specification of VMs proposed by Azure Cloud Service [10]. All our further evaluation utilizes the same type of VMs, called *A0*. It is characterized by 1 Virtual CPU and 768 RAM. We use the same seed value for all VMs therefore all VMs request always the same amount of resources.
- *VM migration network overhead:* The Chapter 2 defines the Page Dirty Rate that is measure of network data overhead during migration of a VM. We considered it as a multiplier of amount of sending data through a network bandwidth.

[1] `http://www.spec.org/power_ssj2008/results/res2011q1/`

- *SLAs:* Each VM has an SLA that composes from two availability measures: time of an overall downtime per billing period (one month in our case) and maximal duration of a single downtime. Moreover, each SLA has a priority level that forces to allocate resources to the VMs with higher priority level firstly.

Performance metrics

In order to compare the efficiency of algorithms we use several metrics.

The first metric is total energy costs spent by all data centers in the cloud caused by VM workloads.

The second metric is an SLA violation costs paid by a cloud provider. We define that an SLA violation occurs either if a VM unavailability exceeds a certain time threshold over the whole billing period or during single downtime taken separately. We separate downtimes to regular downtimes and short downtimes. Regular downtime for a given VM can occur either if it is not provided by a requested amount of *Million Instructinos per Second* (MIPS) or if a data center where its hosted server is located encounters a power outage. In the case, if a power outage lasts less then the duration of the simulation step such downtime is considered as the short downtime. Regular downtime due to lack of resources occurs in the case if a VM shares the same host with other VMs. The metric shows a level of degradation of a cloud provider QoS caused by consolidations in the cloud in order to save costs. It is assumed that the provider pays a penalty in the case of an SLA violation.

The third metric is a number of regular and short downtimes that represents the time where a VM was not provided by a necessary resources that leads to a performance degradation.

The last metric is the number of VM migrations scheduled by an elasticity manager.

Experiment setup

For evaluation purposes two helper classes were additionally implemented: `Scenario` and `ScenarioExecutor`.

Scenario

`Scenario` represents an evaluation scenario and contains a set of initial configurations for an experiment as follows:

- An elasticity manager algorithm: defines an elasticity manager. Just one and only one elasticity manager can be used during evaluation.
- A VM migration policy: identifies migration policy for the elasticity manager.
- A workload prediction strategy: defines the strategy that is used for prediction of future VM resource workloads.
- A simulation period: start and end date and time of simulation.

- A number of data centers: defines the number of data centers taking participation in a simulation. In the case of usage of just one data center the data center in Tokyo was chosen.
- A number of physical machines per data center: defines amount of hosts located in each data center.
- A number of simulated VMs: defines a number of VMs scheduled during the first simulation step to the cloud.
- Existence of power outages: defines whether a simulation engine schedules a power outages to a cloud infrastructure.
- VM CPU workload model: defines a workload generated by a VM. Used workloads refer to the application workloads defined in the Section 2.2.

We used the same data centers characteristics in all our experiments. Table 6.1 provides overview of them of each data center selected in this Section previously.

Parameter \ Data center	Brazil	Toronto	Oslo	Vienna	Tokyo
Time Zone	Brazil/ East	America/ Toronto	Europe/ Oslo	Europe/ Vienna	Asia/ Tokyo
Energy price (USD/kWh)	0.162	0.117	0.159/ 0.1113	0.2484/ 0.1678	0.24/ 0.20
Day/Night switch hours	8 AM/ 11 PM	8 AM/ 11 PM	8 AM/ 11 PM	6 AM/ 10 PM	8 AM/ 11 PM
SAIDI (min/year)	1101.6	220	218	39	6
CAIDI (min/time)	25	5	5	3	1

Table 6.1: Overview of the data center input characteristics

We used also just one type of physical machine during all simulations, namely, HP ProLiant ML110 G3 1 CPU Pentium D930 3000 MHz 2 cores and 4GB RAM, which power specifications are provided by *SPECpower ssj2008* [87]. We assume in our experiments that 3000 MHz refers to 3000 MIPS.

The granularity of all simulations was set to one hour.

ScenarioExecutor

`ScenarioExecutor` is responsible for a configuration of a cloud simulation engine, starting an evaluation and obtaining final results in convenient readable format. It should be mentioned that it does not perform monitoring of the evaluation execution. On the contrary, monitoring is performed by an attached to a cloud `PassiveMonitoringSystem` described in the Section 5.1. After each simulation step it makes snapshots of current state of each cloud entity

including the cloud itself and outputs monitoring results into defined by `ScenarioExecutor` files. Further, these data are used during analysis of final results. Such detailed log of simulation execution allows to achieve high revisability [42] of the experiment. In the case, if an experimental hypothesis is not met, it should help in identifying the reasons: whether these reasons are caused by the modelling, the algorithm, the design, its implementation or the experimental environment.

6.2 Execution and analysis of scenarios

The following section describes the execution and further analysis of various evaluation scenarios. At the end it will give a summary of all executed experiments.

Each scenario implies the simulation of one month execution of 10 VMs, between January 1, 2013 and February 1, 2013, that equals to one billing month for the SLAs defined in the previous Section.

All First-Fit-Decreasing heuristics used `ShortDowntimeVmMigrationPolicy` for discovering the candidates for a migration. It means that they start to migrate a VM just in the case of it encounters short or regular downtime. Our approach described in the Chapter 4 used `AlwaysVmMigrationPolicy` for all scenarios, because it is assumed that the used multi-criteria decision-making algorithm already implies the discovering of VM migration candidates. Additionally, we evaluate the proposed algorithm using different workload forecasting strategies, described in the Chapter 4, such as *LastWorkload*, *RateWorkload*, *Trend* and *SimpleRegression*, in order to identify an impact of the prediction strategy to the efficiency of the proposed model.

Results for each scenario will present with a set of plots as follows:

- *Energy costs* plot shows total energy costs.
- *SLA penalty costs* plot shows total penalties paid due to SLA violations.
- *Number of downtimes* plot represents the sum of downtime events occurred for each VM. We consider a downtime for a VM if it was not provisioned with requested amount of CPU resources.
- *Number of downtimes* plot represents the sum of short downtime events occurred for each VM. We consider a short downtime for a VM if it was not provisioned with requested amount of CPU resources or was shortly unavailable between two simulation timestamps due to power outage. This plot is used just in the Scenario 4 instead of the plot *Number of downtimes*.
- *Number of migrations* plot represents number of all triggered migrations.

Each plot shows a comparison of different strategies evaluated in this scenario. On the Y-axis that value of the parameter is defined, on the X-axis the evaluated algorithms are shown.

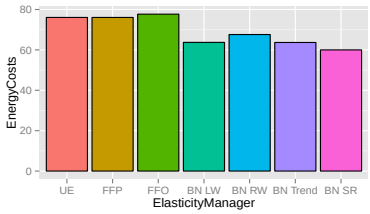
Table 6.2 gives and overview of evaluated algorithms and abbreviations which they use on the plots respectively. The baseline separates foreign approach (above the line) from ours (under the line).

We will follow this notation for all figures between Figure 6.2 and Figure 6.8.

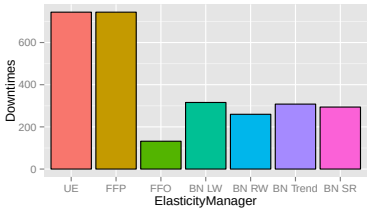
Algorithm	VM Migration policy	Workload Prediction policy	Abbreviation
`Unefficient`	`ShortDowntime`	No strategy used	UE
`FirstFitPesimistic`	`ShortDowntime`	No strategy used	FFP
`FirstFitOptimistic`	`ShortDowntime`	No strategy used	FFO
`BayesianNetwork`	`Always`	`LastWorkload`	BN LW
`BayesianNetwork`	`Always`	`RateWorkload`	BN RW
`BayesianNetwork`	`Always`	`Trend`	BN Trend
`BayesianNetwork`	`Always`	`SimpleRegression`	BN SR

Table 6.2: Overview of evaluated algorithms and a set abbreviations referred to them.

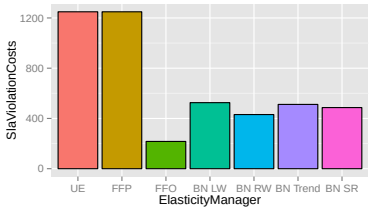
Scenario 1: An efficient power management in a highly-concurrent environment



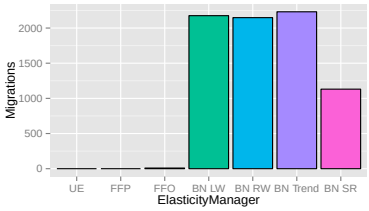
(a) Energy costs



(c) Number of downtimes



(b) SLA penalty costs



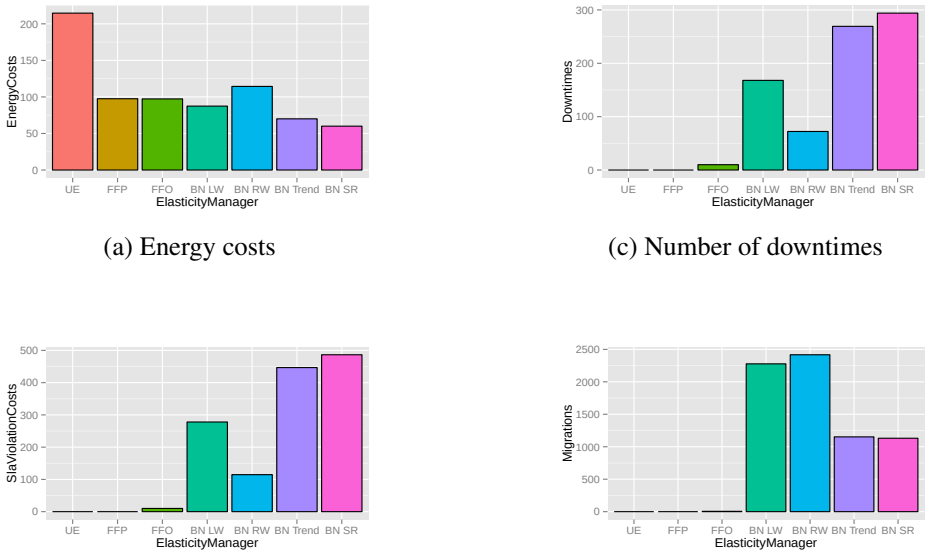
(d) Number of migrations

Figure 6.2: The comparison of performance metrics of evaluated elasticity managers for the scenario 1

The first scenario evaluates the ability of an algorithm to perform correct placement of VMs with periodic CPU workload to the hosts in a single data center, namely in Tokyo. This scenario does not schedule any power outages. In order to force periodic downtimes of at least some part of VMs we set the number of physical hosts to 3 and the range of periodic workload of a VM between 10% and 80%. As all VMs generate the same workload it is easy to observe that at least one VM should encounter a downtime at the moment of peak resource usage.

The experimental results of the evaluation are shown on the Figure 6.2. The usage of our algorithm allows to reduce energy cost to the most 22% in comparison to the worst case, but on the other hand introduces more downtimes in comparison to the optimistic First-Fit algorithm.

Scenario 2: An efficient power management in a non-concurrent environment



(a) Energy costs

(c) Number of downtimes

(b) SLA penalty costs

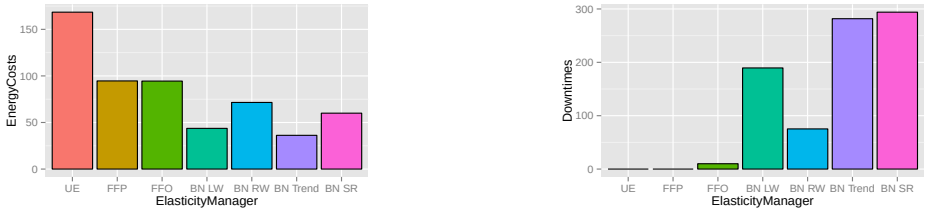
(d) Number of migrations

Figure 6.3: The comparison of performance metrics of evaluated elasticity managers for the scenario 2

The following scenario operates under the same conditions as Scenario 1 except the number of physical machines available in a data center. We set the number of hosts to 10 that guarantees non-concurrent resource supply for the hosted VMs. The goal of this scenario is to test how efficiently our approach can place VMs to these physical machines in non-concurrent environment.

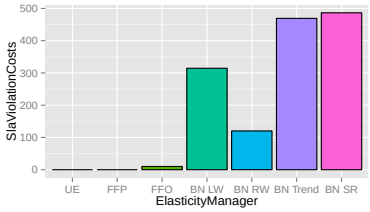
The experimental results of the evaluation are shown on the Figure 6.3. The usage of our algorithm provides improvements of costs usage to the most 72% in comparison to the worst case and to 40% to First-Fit algorithm. Once more, as in the previous scenario, a big amount of VM migrations cause an increase of downtimes and SLA penalty costs.

Scenario 3: The benefits of geo-distibuted data centers

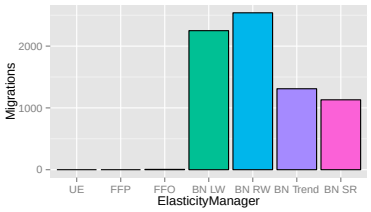


(a) Energy costs

(c) Number of downtimes



(b) SLA penalty costs

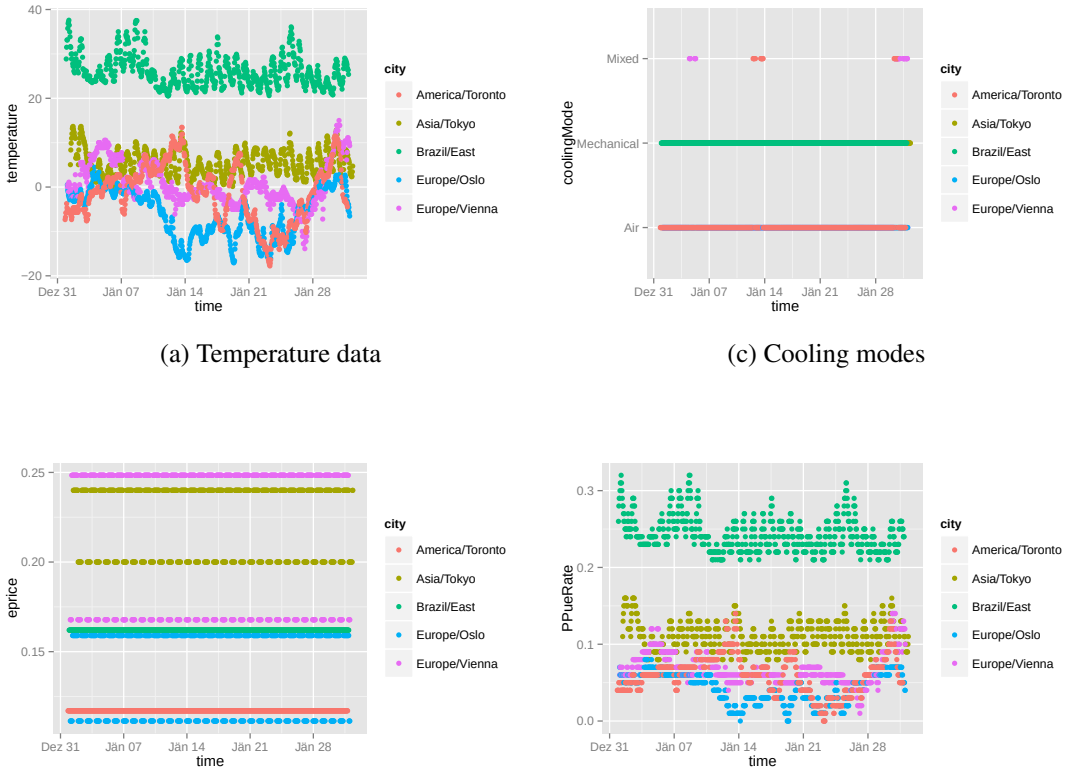


(d) Number of migrations

Figure 6.4: The comparison of performance metrics of evaluated elasticity managers for the scenario 3

This scenario increases the number of data centers in comparison to the previous one. Thereby it introduces new information and conditions that can be used in our algorithm. Each location hat its own electricity price and temperature that differ depending on the time of day. As was mentioned in the Chapter 2 the temperature involves the usage of different cooling modes and also directly impacts on the PPue rate. Figure 6.5 shows the values of various input parameters such as energy price, temperature, used cooling mode, PPue rate during the whole simulation period at each data center location. Moreover, each data center locates in its own time zone. These two facts imply more or less competitive energy price at a certain data center in comparison to other data centers as well as well as ability to use the most efficient cooling mode depending on temperature and thereby consume less energy. Our algorithm should consider all these objectives and migrate VMs between data centers during the day in order to save costs.

The experimental results of the evaluation are shown on the Figure 6.4. The usage of our algorithm provides improvements of costs usage to the most 79% in comparison to the worst case and to 62% to First-Fit algorithm. The best results in perspective of downtimes for our algorithm was achieved with *RateWorkload* prediction strategy.



(a) Temperature data

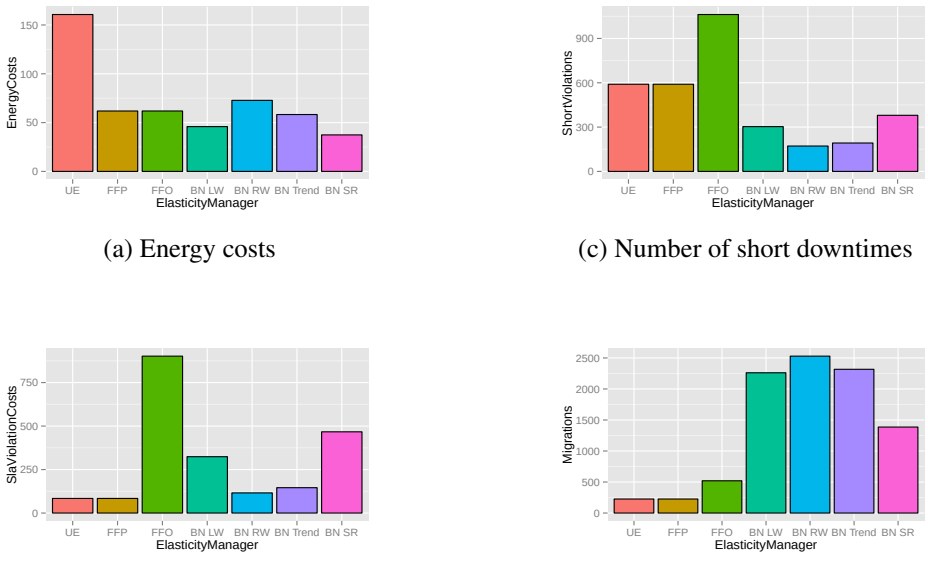
(c) Cooling modes

(b) Energy prices

(d) PPue rates

Figure 6.5: Temperature data (a) (provided by Forecast.IO), energy prices (b), cooling modes (c) and respected PPue rates (d) between January 1, 2013 and February 1, 2013 of 2013 at each data center location

Scenario 4: High reliability during frequent power outages



(a) Energy costs

(c) Number of short downtimes

(b) SLA penalty costs

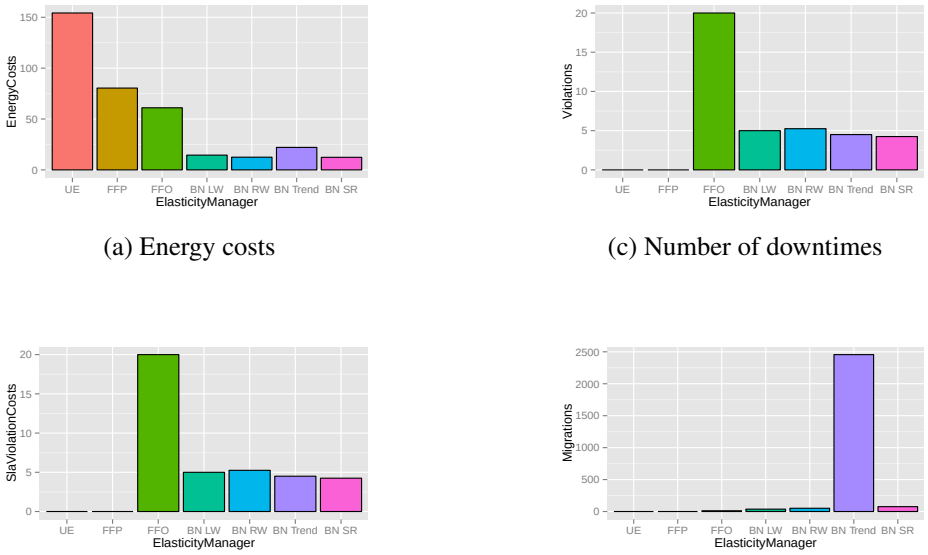
(d) Number of migrations

Figure 6.6: The comparison of performance metrics of evaluated elasticity managers for the scenario 4

The following scenario introduces power outages into cloud management. It should illustrate the impact of power outages on cloud reliability and to analyse the benefits that brings the usage of power outage statistics data.

The experimental results of the evaluation are shown on the Figure 6.6. The usage of our algorithm provides improvements of costs usage to the most 77% in comparison to the worst case and to 40% to First-Fit algorithm. The best results in perspective of downtimes for our algorithm was achieved with *RateWorkload* prediction strategy: number of short downtimes decreased to the most 71% in comparison to the worst case and to 84% to First-Fit algorithm.

Scenario 5: Maintenance of rare unscheduled peak workloads



(a) Energy costs

(c) Number of downtimes

(b) SLA penalty costs

(d) Number of migrations

Figure 6.7: The comparison of performance metrics of evaluated elasticity managers for the scenario 5

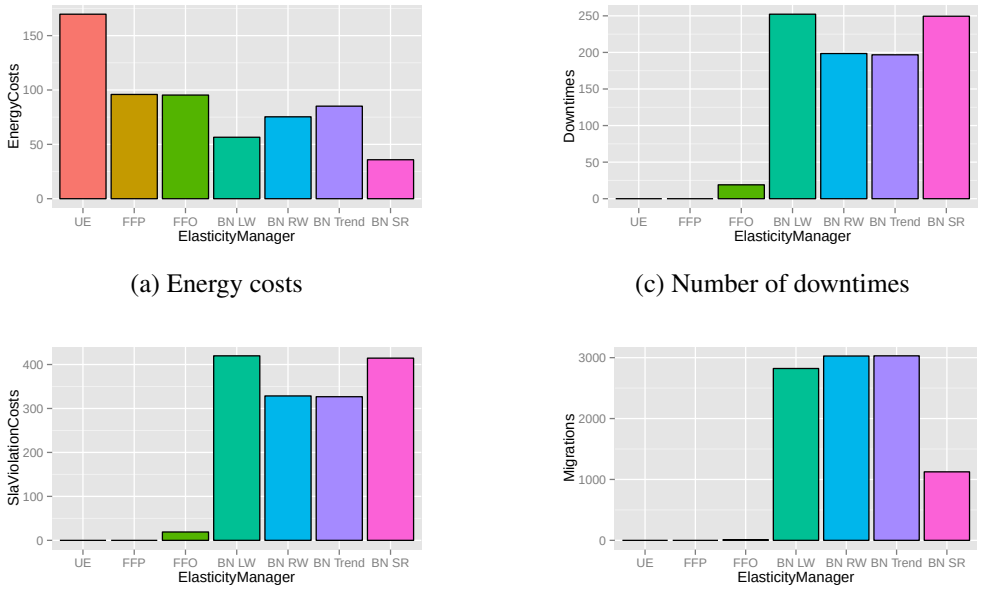
This scenario operates under same input parameters as Scenario 3 except the usage of Once-in-a-lifetime Workload for simulation of VM CPU resource usage. As was mentioned in the Chapter 2 this type of workload refers to a certain event or task occurred once during an observation period.

We set the starting time of workload increase at 12 AM on the 1^{st} of January with the duration of 12 hours.

The experimental results of the evaluation are shown on the Figure 6.7. The usage of our algorithm provides improvements of costs usage to the most 92% in comparison to the worst

case and to 79% to First-Fit algorithm. The best results in perspective of downtimes between strategies used our algorithm was achieved with *SimpleRegression* prediction strategy.

Scenario 6: Efficient consolidation of unpredictable workloads



(a) Energy costs

(c) Number of downtimes

(b) SLA penalty costs

(d) Number of migration

Figure 6.8: The comparison of performance metrics of evaluated elasticity managers for the scenario 6

The last scenario tests the performance of proposed algorithm in the case when a cloud provider operates in environment with unpredictable workloads generated by the VMs. The scenario has the same input parameters as previous one except the fact that Unpredictable Workload is used for generation of CPU resource demand.

The experimental results of the evaluation are shown on the Figure 6.8. The usage of our algorithm provides improvements of costs usage to the most 79% in comparison to the worst case and to 63% to First-Fit algorithm. Hovewer, it should mentioned that despite efficient energy power consumption, all elasticity manager using our algorithm faces high SLA penalty costs.

6.3 Evaluation Summary

The Section 6.2 presented the evaluation scenarios and obtained results. We simulated the performance and behaviour of proposed algorithm in comparison to commonly used heuristics in this field of study. The experimental results indicate that our algorithm significantly improves

energy costs savings almost in all simulation cases. Furthermore, it can be easily observed that the more additional information is available for the algorithm the better results it shows. A good example is the Scenario 4 that provides all possible existed information for an elasticity manager. In this case, our approach reduced energy costs to 40% in comparison to the best First-Fit approach.

Also based on the results we can conclude that the best performance was achieved when an algorithm used a linear regression model for the prediction of future workloads. It proves the fact that more sophisticated and accurate prediction algorithm could further more improve results of our experiments. We should mention that good time-series forecasting model is not a focus of current Thesis, therefore these improvements should be done in the future work.

For the described advantages the proposed approach also has a set of weaknesses:

- A big amount of triggered migrations introduces a lot of downtimes that results in higher SLA penalty costs.
- Sometimes a set of migrations induces more other migrations that can bring the system to the unstable state.
- The utility function defined in the Chapter 4 that is used in decision-making process while selecting target host for scheduled VMs still can be improved as it was defined based on our subjective view of the importance of its parameters.
- A necessity to discretize input values for the BN used in the `ElasticityManagerMCDA` leads to a degradation of model precision and effectiveness.

CHAPTER 7

Conclusions and Future Work

Modern cloud computing technologies experienced fast growing trend in recent years by offering highly available and scalable services. They are allowed to store a big amount of data, perform large scale computations, ensure high reliability, and supply huge customer demands. Increasing resource demand forces the providers to either expand production capacities that increases power consumption or apply more economical resource provision plans that degrades quality of service. Such situation established in the cloud computing world gives a good change for the scientific community to supply the world's leading cloud vendors with effective algorithms for cloud infrastructure management.

In this thesis, we propose a novel approach for cost-aware cloud power management that allows to significantly reduce operational and penalty costs paid by a cloud provider. The key technique that is utilized to reduce the costs and to improve the resource provisioning is an effective placement of customer's VMs in the cloud data centers.

There is big amount of related work in this field of study. However, modern approaches focus mostly just on concrete problems or parts of cloud systems loosing the whole picture of complex interdependencies of internal and external factors and parameters. In this work we try to fill this scientific gap by involving a big amount of cloud processes, parameters into proposed algorithm. We introduce the consideration of geo-distributed data centers experienced frequent power outages, operating in different time zones and in constantly changing temperatures.

Our solution for the problem of VM placement consists of two steps. First, we model a cloud infrastructure using the theory of BNs in order to find dependencies between its different parts and processes. For these purposes we designed a suitable BN that allows to calculate a probability of various factors such a SLA violation, power consumption level, etc. with some level of confidence.

On the second step, we utilize the MCDA method to assess physical machines by a various number of criteria. We define our custom utility function that utilizes values of the factors calculated by the BN on the previous step. This utility allows us to find the most suitable host for allocation or migration of VM to it.

Ensuring of the performance and applicability of our approach is one more important step of our study. We used simulation experimental methodology to perform assessments of the model. As there was no simulation toolkit that allowed simulation of objectives and processes defined by us, e.g., VM migration between geo-distributed data centers, or scheduling of power outages, mentioned above, we implemented our own Java-based simulation framework, called Cloudnet. This framework provides a rich set of cloud simulation opportunities. One of its key features is a possibility to execute code in R language that contains a wide range of mathematical algorithms and methods. In particular, we used an R package *gRain* for creation and querying of BNs.

For conducting a critical performance evaluation of our approach we defined a set of reasonable real-world scenarios and evaluated our model in comparison to commonly used algorithm First-Fit-Decreasing defined in the context of the Bin-Packing problem.

The results of the experiments proved high performance and applicability of our approach. They showed its ability to significantly decrease energy usage and to improve resource provision in the cloud infrastructure. Moreover, the results clearly demonstrated that consideration of more input factors and application of domain knowledge greatly improved the efficiency of the proposed model.

In summary we can conclude that the proposed model for sophisticated cloud power management based on the BNs and MCDA shows promising results in solving cost-based issues in cloud computing environments.

7.1 Future Work

However, there are still some open issues and challenges that should be investigated in the future work.

First, the model triggers sometimes too much migration actions generating a huge migration overhead that bring a system to unstable state and leads to its downtime. This problem can be related to the utility function selected during MCDA based just on the own subjective view of the importance of its input variables. The correct parameters of the utility function is another optimization problem that is left for the future work. Another reason of such model behaviour can be caused by incorrectly select VM migration policy that by its nature constantly forces to perform a consolidation of the VMs placed in the cloud.

Second, the model utilizes a time-series forecasting models for prediction of future resource demands. The evaluation has shown that more sophisticated workload prediction strategy can significantly improve performance of the model.

Finally, the current design of the BN utilized by the model implies the usage of only discrete nodes. Hence, all input values supplied to the model are initially discretized and thereby loose their precision. This fact can lead to further rough assessment of probabilities in the BN and wrong conclusions. Hence, the application of Hybrid Bayesian Networks that allow usage of analogous data can improve precision of calculated parameters of the model and result into more efficient decisions.

APPENDIX A

List of Abbreviations

AWS	Amazon Web Services
BaaS	Backend as a Service
BN	Bayesian Network
BSP	Backward Speculative Placement
CAIDI	Customer Average Interruption Duration Index
CPT	Conditional Probability Table
DaaS	Data as a Service
DAG	Directed Acyclic Graph
DVFS	Dynamic Voltage and Frequency Scaling
GQM	Goal Question Metric
IaaS	Infrastructure as a Service
ILP	Integer linear program
MBFD	Modified Best-Fit-Decreasing
MCDA	Multi-criteria decision aid
MIPS	Million Instructinos per Second
MOM	Message-oriented middleware
MP	Migration Plan
OCBP	Online Colouring Bin Packing problem
OCFF	Online Colouring First-Fit
OS	Operating system
PaaS	Platform as a Service
PBN	Predictive Bayesian Network
PMCS	Physical Machine Candidate Selection
pPue	Partial power usage effectiveness
QoS	Quality of Service
SaaS	Software as a Service
SAIDI	System Average Interruption Duration Index

Continued on th Next page...

SLA Service-Level Agreement
TMG Target Mapping Generation
ULS Ultra-Large-Scale
VM Virtual machine

Bibliography

[1] Electricity pricing. Accessed: 2014-04-07.

[2] http://analyticstrainings.com/?p=101. Accessed: 2014-03-02.

[3] A beginners guide to bayesian network modelling for integrated catchment management. Technical report, Landscape Logic, July 2009.

[4] The äbcsöf daas. Technical report, Delphix, December 2011.

[5] Bayesian networks: A guide for their application in natural resource management and policy. Technical report, Landscape Logic, March 2011.

[6] J. Abawajy A. Beloglazov and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28:755–768, 2012.

[7] R. Letcher A. Jakeman and S. Chen. Integrated assessment of impacts of policy and water allocation change across social, economic and environmental dimensions. *Managing Water for Australia: the social and institutional challenges*, pages 97–112, 2007.

[8] M. Shaw A. Whitaker, R. Cox and S. Grible. Constructing services with interposable virtual hardware. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'04), Berkeley, CA, USA*, pages 169–182, 2004.

[9] Evolven's IT Operations Analytics. Downtime, outages and failures - understanding their true costs, September 2013. Accessed: 2014-06-05.

[10] Microsoft Azure. http://azure.microsoft.com/. Accessed: 2014-05-11.

[11] M. Raphael M. Rowland B. Marcot, R. Holthausen and M. Wisdom. Using bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. *Forest Ecology and Management*, 153:29–42, 2001.

[12] T. Bayes. An essay toward solving a problem in the doctrine of chances. 1764.

[13] Softlayer blog. Virtual magic the cloud, April 2014. Accessed: 2014-03-18.

[14] F. Leymann C. Fehling and R. Retter et al. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, 2014.

[15] M. Hadji C. Ghribi and D. Zeghlache. Energy efficient vm scheduling for cloud data centers: Exact allocation and migration algorithms. In *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 671–678. IEEE, 2013.

[16] G. Cook and J. Van Horn. How dirty is your data? Technical report, Greanpeace Internation, April 2011.

[17] G. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[18] E. Curry. Message-Oriented Middleware. *In Middleware for Communications*, pages 1–28, 2004.

[19] J. Dean and S. Ghemawat. MapReduce: Simplied Data Processing on Large Clusters. *MapReduce: Simplied Data Processing on Large Clusters*, 2004.

[20] O. Doguc and J. Ramirez-Marquez. An Efficient Fault Diagnosis Method for Complex System Reliability. In *Proceedings of the Seventh Annual Conference on Systems Engineering Research 2009 (CSER 2009)*, 2009.

[21] R. Johnson E. Gamma, R. Helm and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1994.

[22] Austrian electrical outages statistics. http://www.e-control.at/en/businesses/electricity/security-of-supply/outage-and-disturbance-statistics. Accessed: 2014-06-12.

[23] World electrical outages statistics. http://www.nationmaster.com/country-info/stats/energy/electrical-outages/days. Accessed: 2014-06-12.

[24] A. Schatten et al. *Best Practice Software-Engineering, Eine praxiserprobte Zusammenstellung von komponentenorientierten Konzepten, Methoden und Werkzeugen*. Spektrum Akademischer Verlag (Springer), 2010.

[25] C. Clark et al. Live Migration of Virtual Machines. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'05), Berkeley, CA, USA*, pages 273–286, 2005.

[26] C. P. Sapuntzakis et al. Optimizing the migration of virtual computers. In *Proceedings of the 5th symposium on Operating systems design and implementation*, volume 36, pages 377–390, 2002.

[27] G. Chen et al. Energyaware server provisioning and load dispatching for connection-intensive internet services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 337–350. USENIX Association, 2008.

[28] H. Sijin et al. Improving resource utilisation in the cloud environment using multivariate probabilistic models. In *Proceedings of the Fifth International Conference on Cloud Computing*, pages 574–581. IEEE, 2012.

[29] L. Jian et al. Reducing operational costs through consolidation with resource prediction in the cloud. In *Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 793–798. IEEE, 2012.

[30] N. Calcavecchia et al. VM Placement Strategies for Cloud Scenarios. In *Proceedings of the Fifth International Conference on Cloud Computing*, pages 852–859. IEEE, 2012.

[31] R. L. Grossman et al. The Open Cloud Testbed: A Wide Area Testbed for Cloud Computing Utilizing High Performance Network Services. *CoRR*, abs/0907.4810, 2009.

[32] S. Akoush et al. Predicting the Performance of Virtual Machine Migration. In *Proceedings of the Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 37–46. IEEE, 2010.

[33] S. Kuikka et al. Modeling environmentally driven uncertainties in baltic cod (gadus morhua) by bayesian influence diagrams. *Canadian Journal of Fisheries and Aquatic Sciences*, 56(4):629–641, 1999.

[34] T. Banzai et al. D-cloud: Design of a software testing environment for reliable distributed systems using cloud computing technology. In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 631–636, 2010.

[35] W. Voorsluys et al. Cost of virtual machine live migration in clouds: A performance evaluation. 2009.

[36] Y.Song et al. A Two-Tiered On-Demand Resource Allocation Mechanism for VM-Based Data Centers. *IEEE TRANSACTIONS ON SERVICES COMPUTING*, 6(1):116–4129, 2013.

[37] P. Eugster. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)*, 35(2):114–131, 2003.

[38] N. Fenton and M. Neil. Making Decisions: Using Bayesian Nets and MCDA. *Knowledge-Based Systems*, pages 307–325, 2001.

[39] B. Furht and A. Escalante. *Handbook of Cloud Computing*. Springer, 2010.

[40] D. George and J. Hawkins. Invariant Pattern Recognition using Bayesian Inference on Hierarchical Sequences.

[41] gRain: Graphical Independence Networks. http://cran.r-project.org/web/packages/grain/index.html. Accessed: 2014-03-08.

[42] J. Gustedt and E. Jeannot. Experimental Validation in Large-Scale Systems: a Survey of Methodologies. In *Parallel Processing Letters 19*, pages 399–418. IEEE, 2009.

[43] C. Feng H. Xu and B. Li. Temperature aware workload management in geo-distributed datacenters. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, pages 373–374, 2013.

[44] D. Heckerman and J. Breese. Causal independence for probability assessmentand inference using bayesian networks. Technical report, Microsoft Research, Advanced Technology Division, 1995.

[45] What's New in JDK 8. http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html. Accessed: 2014-07-28.

[46] T. Kraska J. Moreno, D. Kossmann and S. Loesing. A Testing Framework for Cloud Storage Systems. Master's thesis, Swiss Federal Institute of Technology Zurich, 2010.

[47] E. Cooke J. Oberheide and F. Jahanian. Empirical exploitation of live virtual machine migration. 2008.

[48] F. Jensen and T. Nielsen. *Bayesian Networks and Decision Graphs*. New-York: Springer Verlag, 2007.

[49] Introducing JSON. http://json.org/. Accessed: 2014-01-17.

[50] W. Gu-Yeon K. Wonyoung, M.S. Gupta and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *Proceedings of the IEEE 14th International Symposium on High Performance Computer Architecture, 2008*, pages 123–134. IEEE, 2008.

[51] Z.M. Jiang K.C. Foo and B. Adams et al. Modeling the performance of Ultra-Large-Scale systems using layered simulations. In *Proceedings of the International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*. IEEE, 2011.

[52] W. Kohl. Backend as a service using the example of enginio a cloud service for qt. 2013.

[53] M. Kozuch and M. Satyanarayanan. Internet suspend/resume. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 40–46. IEEE, 2002.

[54] M. Krieg. *A Tutorial on Bayesian Belief Networks*. DSTO Electronics and Surveillance Research Laboratory, 2001.

[55] S. Bucur-V. Chipounov L. Ciortea, C. Zamfir and G. Candea. Cloud9: A Software Testing Service. *ACM SIGOPS Operating Systems Review*, 43(4):5–10, 2010.

[56] Z. Huanyang L. Kangkang and W. Jie. Migration-based Virtual Machine Placement in Cloud Systems. 2013.

[57] C. Lam. *Hadoop in Action*. Manning Publications Co. Greenwich, 2010.

[58] S. Lauritzen and D. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological) 50(2)*, pages 157–224, 1988.

[59] C. Stow M. Borsuk and K. Reckhow. A bayesian network of eutrophication models for synthesis, prediction, and uncertainty analysis. *Ecological Modelling*, 173:219–239, 2004.

[60] V. Emeakaroha M. Maurer and I. Brandic. Cost and Benefit of the SLA Mapping Approach for Defining Standardized Goods in Cloud Computing Markets. *Future Generation Computer Systems*, 28:39–47, 2012.

[61] D. Marquez M. Neil and N. Fenton. Using Bayesian networks to model the operational risk to information technology infrastructure in financial institutions. *The capco institute, Journal of financial transformation*, pages 131–138.

[62] S. Masoumzadeh and H. Hlavacs. Integrating VM Selection Criteria in Distributed Dynamic VM Consolidation Using Fuzzy Q-Learning. In *Proceedings of the 9th CNSM 2013: Workshop SVM 2013*, pages 332–338, 2013.

[63] J. McKendrick. Cloud computing's hidden green benefits, March 2011. Accessed: 2014-03-18.

[64] I. Milho and A. Fred. A user-friendly development development tool for medical diagnosis based on Bayesian Networks.

[65] C. Mines. 4 reasons why cloud computing is also a green solution, July 2011. Accessed: 2014-03-18.

[66] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[67] MSDN. Virtual machine live migration overview, 2013. Accessed: 2014-06-30.

[68] K. Murphy. An introduction to graphical models. 2001.

[69] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[70] N. Nilsson. *Introduction into Machine Learning*. Stanford University, November 1998.

[71] S. Poll O. Mengshoel and T. Kurtoglu. Developing Large-Scale Bayesian Networks by Composition: Fault Diagnosis of Electrical Power Systems in Aircraft and Spacecraft.

[72] Performance of FastMath from Commons Math. http://blog.juma.me.uk/2011/02/23/performance-of-fastmath-from-commons-math/. Accessed: 2014-05-02.

[73] S. Ohta. Virtual Machine Placement Algorithms to Minimize Physical Machine Count. 2013.

[74] M. Oriol and F. Ullah. Yeti on the cloud. In *Proceedings of the Third International Conference on Software Testing, Verification, and Validation Workshops*, pages 434—437, 2010.

[75] H. S. Gunawi P. Joshi and K. Sen. Prefail: a programmable tool for multiple-failure injection. In *Proceedings of the 2011 ACM international conference on Object oriented programming systems languages and applications*, pages 171–188, 2011.

[76] T. Silander P. Kontkanen, P. Myllmäki and H. Tirri. Comparing predictive inference methods for discrete domains. In *Proceedings of the Sixth Internation Workshop On Artificial Intelligence and Statistics (Ft. Lauderdale, USA, January 1997)*, 1997.

[77] A. Pogosyan. vmotion virtual machines between clusters with different hosts, 2013. Accessed: 2014-04-02.

[78] W. Premchaiswadi. *Bayesian Networks*. InTech, 2012.

[79] R Project. http://www.r-project.org/. Accessed: 2014-03-08.

[80] J. Broberg R. Buyya and A. Goscinski. *Cloud Computing. Principles and Paradigms*. John Wiley, 2011.

[81] A. Beloglazov-C. De Rose R. Calheiros, R. Ranjan and R. Buyya. CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experienc*, 41(1):23–50, 2011.

[82] Amazon Web Service. http://aws.amazon.com/. Accessed: 2014-02-12.

[83] Bayes Server Web Site. http://www.bayesserver.com/. Accessed: 2014-01-12.

[84] Forecast.IO Weather Web Service Web Site. http://www.forecast.io/. Accessed: 2014-03-11.

[85] ITKO Web Site. http://www.itko.com/. Accessed: 2014-07-28.

[86] Wien Energie Web Site. http://www.wienenergie.at/. Accessed: 2014-07-11.

[87] First Quarter 2011 SPECpower ssj2008 Results. http://www.spec.org/. Accessed: 2014-03-12.

[88] Internation Power Outage Statistics. http://earlywarn.blogspot.co.at/2013/05/international-power-outage-comparisons.html. Accessed: 2014-06-12.

[89] A. Sykes. An Introduction to Regression Analysis. *Chicago Working Paper in Law & Economics*, 1993.

[90] N. Daley T. Parveen, S. Tilley and P. Morales. Towards a distributed execution framework for junit test cases. In *Proceedings of the IEEE International Conference on Software Maintenance 2009*, pages 425—428, 2009.

[91] Taking R to the Limit: Parallelism and Big Data. http://blog.revolutionanalytics.com/2010/08/taking-r-to-the-limit-parallelism-and-big-data.html. Accessed: 2014-06-18.

[92] L. Uusitalo. Advantages and challenges of bayesian networks in environmental modelling. *Ecological Modelling*, 203:312–318, 2007.

[93] O. Varis. Bayesian decision analysis for environmental and resource management. *Environmental Modelling and Software*, 12:177–185, 1997.

[94] V. Vazirani. *Approximation Algorithms*. 2003.

[95] Ph. Vincke. Multicriteria decision-aid. *Journal of Multi-Criteria Decision Analysis*, 3(2):131, 1994.

[96] Finland Web Page of a Google's data center located in Hamina. http://www.google.com/about/datacenters/inside/locations/hamina/index.html. Accessed: 2014-04-05.

[97] B.T.W.-T. Chen J. Gao X. Bai, M. Li. Cloud testing tools. In *Proceedings of the 6th International Symposium on Service Oriented System (SOSE 2011)*, pages 373–374. IEEE, 2011.

[98] I. Yevseyeva. *Solving Classification Problems with Multicriteria Decision Aiding Approaches*. Jyväskylä University Printing House, 2007.

[99] M. Yue. A simple proof of the inequality FFD (L)<11/9 OPT (L)+ 1,for all l for the FFD bin-packing algorithm. *Acta Mathematicae Applicatae Sinica (English Series)*, 7(4):321–331, 1991.

[100] E. Zayas. Attacking the process migration bottleneck. In *Proceedings of the eleventh ACM Symposium on Operating systems principles*, volume 21, pages 13–24, 1987.