Application Performance Optimization in Multicloud Environment

Martin Bobák*
Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9, SK–845 07 Bratislava, Slovakia
martin.bobak@savba.sk

Abstract

Nowadays, cloud computing is one of the most used information technologies. It offers an infrastructure, a platform or a software as a service over the network to the huge number of remote users. Currently, this is a very promising and popular way in which they are offered. Cloud computing has a potential to change the using of information technologies.

There are two main open problems in cloud computing – interoperability and privacy. Our dissertation thesis is focused on interoperability. Nowadays it is difficult to migrate an application between clouds offered by different providers. The thesis deals with this problem in multicloud environment. Specifically, it is focused on an application performance optimization in a multicloud environment. A new method is suggested based on the state of the art. The method is divided into three parts: a multicloud architecture, a method of a horizontal scalability and a taxonomy for multicriteria optimization. The principles of the method have been applied in a design of a multicriteria optimization architecture, which we verified experimentally. The aim of the experiment is carried on a portal offered a platform according to the users' requirements.

Categories and Subject Descriptors

K.6.2 [**Management of computing and information systems**]: Installation Management—*computer selection, computing equipment management, performance and usage measurement*;
K.6.3 [**Management of computing and information systems**]: Software Management—*software selection software maintenance*

*Recommended by thesis supervisor:
Assoc. Prof. Ladislav Hluchý
and thesis consultant: Dr. Viet Tran
Defended at Institute of Informatics, Slovak Academy of Sciences on TBA.

Keywords

cloud computing, multicloud computing, architecture, multicriteria optimization

1. Introduction

Cloud computing is in the top 10 strategic technology trends for 2015 [11]. It is used by an academic field as well as by an industrial field. One of the main reasons for the expansion of cloud computing is the need for high-performance computing in almost every modern area. Users are requiring two main properties from clouds:

1. interoperability
2. privacy

Dissertation thesis is focused on interoperability. An intercloud migration of an application still represents a serious issue. Almost every cloud provider offers their cloud resources through specific API. Such an approach causes vendor lock in. The users have to adapt to provider's API. Nowadays there is not a standard API or a reference cloud implementation.

The thesis is an extension of Minh Binh Nguyen's research [15]. He created a generic development and deployment framework for cloud computing. The model solves interoperability on an operating system level. This thesis improves the model, in order to carry out an application performance optimization in a multicloud environment.

1.1 Motivation

Multicloud computing is advantageous for users as well as providers. The main advantages are: unified cloud-based architecture, avoiding of vendor lock-in, interoperability, scalability, improved accessibility, reduced latency and reduced operating cost.

There exist four fundamental cloud-based collaboration use cases [21]:

- hybrid cloud
- multicloud computing
- federated cloud
- Inter-cloud

Users of a hybrid cloud utilize a public cloud partly (e.g. resources of private cloud are insufficient). Alternatively, the users offer their unused resources to cloud providers. A hybrid cloud represents a resources' extension of a private cloud.

Users of a multicloud computing work in a third party environment. The provider of the environment offers interoperability. That means the provider translates messages from different end-users to cloud providers and vice versa. However the management of resources is left to the end-users in many cases.

The concept of an inter-cloud was introduced by Cisco [2]. The paper describes the inter-cloud as a cloud made up of different clouds. This is a parallel to the Internet – the network of networks. Inter-cloud is characterized as a group of clouds which communicates through uniform standards. The standards provide cloud interoperability to users. This model has been inspired by Internet and telephone networks. It is typical for them that, in spite of infrastructure offered by different providers, all elements are interconnected.

Federated cloud is an aggregation of multiple cloud providers which offers its resources to improve services of individual cloud providers [13]. The providers made their idle resources available to each other. This policy improves scalability and flexibility across the federation. Thus, users work with resources that come from different providers. This property does not increase the load of the system. It is also irrelevant from the point of view of the system.

Federated cloud, Inter-cloud and multicloud computing are very new concepts. Computer scientists use them often interchangeably. We look at Inter-cloud as a global federated cloud. So the federated cloud is a first step towards Inter-cloud. Multicloud computing is an abstract concept that is an expression of the fundamentals of Inter-cloud and federated cloud.

2. State of the art

There is a very good chance that the next evolution step of cloud computing will be distributed cloud computing – multicloud computing. Authors of paper [7] identified three evolutionary steps: monolithic, vertical supply chain and horizontal federation. Cloud services are offered by a single provider in the monolithic stage. Next step is to offer a pipelined cloud service. The whole service is offered by multiple providers. The providers create a vertical supply chain at the end of which is the service offered to a user. The last step is horizontal federation. Providers create a federation in which they share resources among themselves. This approach makes their services more reliable and improves scalability of their services. On the other hand, it is the interaction of the providers, which represents the greatest challenge.

Currently there dominate monolithic cloud services (Amazon, Google, Microsoft ...). Each provider has their own API which makes service administration and maintenance complicated. Consequently, a user is restricted by one provider in many cases. It is very difficult to collaborate with many providers and cloud services combining is not supported. The heterogeneity causes vendor lock-in [19, 16]. That means, a user becomes dependent on the provider. Users are unable to replace their provider without effort. This situation suits cloud providers who do not engage in it at all [17].

There are two main ways to achieve multicloud computing [21]:

- collaborative service (middleware)
- standardization (e.g. interfaces)

A collaborative service is an intermediary between a user interface and cloud. Its task is to ensure communication between cloud interfaces (e.g. by messages translating). This approach allows customers to use clouds from different providers. It ensures cloud interoperability. Standardization offers another way to ensure interoperability (e.g. this approach was successful at network communication – it creates TCP/IP protocols). Currently, it is very difficult to obtain standardization, so we decided to create a collaborative service.

2.1 Standards supporting a multicloud computing

The following table of standards has been based on the survey [21]. It implies two important facts. The facts influenced us in our research.

The first fact is that a design of a standard for multicloud computing deals with three main factors: interoperability, security and legality (of the service in a particular country). Other obstacles are following. Cloud providers do not want to agree on a uniform standard. They rather prefer vendor lock-in [16].

The second fact is that there have been produced many standards, but neither of them has become a globally recognized for multicloud computing [16, 21] (see table 1). It is a consequence that the providers are not able to agree on a single standard and this situation suits them also. We decided to focus on collaborative services only, based on these facts.

2.2 Collaborative services

Various collaborative services are discussed in this subsection. This is the best way to achieve multicloud computing, taking into account the current situation. The services can be performed either on a user's side or can be offered as a third party service. Following analyse of collaborative services is based on the survey [21].

Reservoir [19] is an architecture for a cloud federation. The authors distinguish between two operations in the model. That also means the model has two types Reservoir sites – infrastructure providers' sites and service providers' sites. Service providers offer their customers the applications that exploits an infrastructure provided by infrastructure providers. Service providers do not own a computing resources, network or storage. Those are owned by infrastructure providers who offer them to service providers in the form of isolated virtual environments – virtual execution environment (VEE).

Architecture Reservoir has three main components: service manager, virtual execution environment manager and virtual execution environment host. The

Table 1: The table compares current standards according to the focused problem – Provisioning (Pr), Portability (Po), Service Level Agreement (SLA), Security (S), Monitoring (M), Economy (E), Network (N), Autonomics (A). (Adapted from [21].)

Standard	Pr	Po	SLA	S	M	E	N	A
DMTF Distributed Management Task Force	✓	✓	✓	✓	✓			✓
OGF Open Grid Forum	✓		✓	✓	✓			✓
CSA Cloud Security Alliance				✓	✓			
OCM Open Cloud Manifesto		✓	✓					
NIST National Institute of Standards and Technologies	✓	✓		✓				
CCIF Cloud Computing Interoperability Forum	✓			✓	✓			
OCC Open Cloud Consortium							✓	
OASIS Organization for the Advancement of Structured Information Standards	✓		✓	✓				
GICTF The European Telecommunications Standards Institute							✓	✓
ETSI Inter-Cloud Technology Forum		✓					✓	✓
CWG The Open Group Cloud Computing Work Group						✓		
OMG Object Management Group			✓	✓		✓		
ODCA Open Data Center Alliance	✓		✓	✓				
IEEE P2302 IEEE P2302 Working Group	✓	✓		✓	✓	✓	✓	
SNIA CSI Storage Networking Industry Association (SNIA) Cloud Storage Initiative	✓	✓						
ISO JTC 1/SC 38 ISO JTC 1/SC 38	✓	✓	✓		✓	✓	✓	
ITU-T FG ITU-T Focus Group on Cloud Computing	✓	✓	✓	✓	✓		✓	
SIENA Standards and Interoperability for eInfrastructure implemeNtation initiAtive	✓	✓		✓				✓

service manager is on the top layer. It interacts with service providers – it receives their service manifests (e.g. deploy virtual environments, SLA monitoring ...). Middle layer is virtual execution environment manager. Its role is to manage a virtual environment, to interact with virtual execution environment manager of other pages (this property creates a federative cloud) and to place virtual environments into virtual execution environment hosts. Bottom layer is virtual execution environment host. It is responsible for supporting various types of virtualization, virtual environment monitoring and provide migration of a virtual environment within a federated cloud.

Contrail [6] enables the utilization of cloud resources offered by different cloud providers (horizontal integration) and also it gives an opportunity to work with IaaS and PaaS (vertical integration). Contrail is based on agents who act as intermediaries between cloud users and cloud providers. Contrail consists of three layers: interface, core and adapters. The interface layer has two forms – command line and web interface. Its role is to collect requirements from users and provide services of a federated cloud as a REST service. Core layer contains modules for security, application management and SLA management. The federation runtime manager looks after the application management which exploits services of the image manager and provider watcher. Adapters layer is a bottom layer. Adapters are subdivided by type of resources provided into external and internal modules. Internal modules make available virtual infrastructure network, global autonomous file system and virtual execution platform. External modules translate messages from federated cloud to cloud providers

OPTIMIS The main components of OPTIMIS are as follows. Services are created by the service builder. It has integrated development environment that allows a development (via programming model) and a configuration (via configuration manager). SaaS cloud is gotten at the end (a service is offered over the Internet and it is executed in multicloud environment). Basic toolkit supply a functionality for service management (e.g. monitoring, security ...) to other components. Once an assembling of a service is finished, deploying of the service is started. Deploying is carried out with deployment engine. It looks for the best infrastructure provider for the service. If the search is successful, deployment engine sends a request to admission controller. The admission controller decides if the service exploits the resources of the infrastructure. Subsequently, cloud optimizer allocates the resources for the new service. Successful deployment of the service is reported to service optimizer. Afterwards, the service is observed by service optimizer which is able to make its subsequent migration to the other infrastructure, if it is necessary and the service level agreement is not broken.

Comparison of the collaborative services. Three significant research projects related to our research have been identified during the analysis of the state of the art: Reservoir, Contrail, and OPTIMIS. The comparison is summarized in the table 2. Several important observations have been made on the basis of the table. OPTIMIS focusses on an optimization and is the extension of an (multi)cloud architecture. It is the set of software tools, while the other two projects focus on interoperability and multicloud architectures. Reservoir achieves interoperability through standardization. This is the reason for the installation on a provider side as well as on a client side. Contrail and OPTIMIS do not rely on standardization and therefore are installed only on client side. They achieve interoperability through agent approach.

Table 2: The table compares the multicloud collaborative services – focus of the project (F), type of the project(T), interoperability achieving (I), provider side installation (P), client side installation (C).

Project	F	T	I	P	C
Reservoir	interoperability	architecture	standardization	✓	✓
Contrail	interoperability	architecture	agent approach		✓
OPTIMIS	efficient use of resources	software tools	agent approach		✓

3. Main objectivities of dissertation thesis

Building on these ideas, our architecture is created. From state of the art, we made several observations:

1. Nowadays, there are many cloud providers offering a cloud service through a proprietary API (e.g. Amazon, Microsoft...). This situation creates vendor lock-in [19, 16].
2. Cloud's customers mostly focus on two types of clouds – SaaS and IaaS [19]. This is due to that a platform (in the current form) is suitable for small group of users. The situation is a result of two factors. If the user is interested in a particular product, they choose a SaaS. On the other hand, if the user wants to be more variable (or they are interested in a virtualized environment), they are forced to use IaaS currently. PaaS appears to be only suitable at first glance (see section 2). That is the reason why the dissertation thesis is focused on IaaS at multicloud computing environment.
3. There are several ways to achieve multicloud computing [17]. The most common ways are a standardization of interfaces and collaborative services. The state of the art shows that the standardization is very difficult to enforce. This is due to the fact that today's cloud providers refusing to accept a common interface [21]. That is the reason why the article presents a collaborative service.
4. Collaborative services offering infrastructure is a young field of cloud computing. There exist only several projects focusing on the services. The article has presented three FP7 – Reservoir [19], Contrail [6] and OPTIMIS [9]. The main difference between these projects is the way an interoperability is achieved. Reservoir has to be carried out on multicloud provider's side. This approach has the same problem as standardization – the providers have to agree on it. Contrail and OPTIMIS take reverse approach – the collaborative service is executed on user's side (alternatively it can be provided as a service). This approach is independent of the providers because the services are located between the providers and the users. Contrail applies an agent approach to achieve multicloud services. However the approach is redundant (it is too high load on a network). In this case, the agents are used for a discovery and selection of resources. OPTIMIS is not a an architecture, it is a software kit allowing some kind of an optimization.

The main ideas that influenced the design of the architecture are as follows:

- allow scalability
- allow interoperability
- avoid vendor lock-in
- provide infrastructure primary (in order to be extensible, the design has to flexible sufficiently)
- create collaborative service
- be independent of providers
- allow flexible provider administration (new provider changes the architecture minimally)

4. Multicloud computing architecture

Let us start with a characterization of a multicloud problematic (see problem 1). This approach enables us to make a method for this problem.

PROBLEM 1. ***Multicloud computing***
Input: *virtual resources VR from different providers*
Problem: *to merge virtual resources VR in order to be accessible by an uniform way*
Output: *multicloud virtual resources MVR*

The description of proposed architecture employs the model of abstract resources created by Minh Binh Nguyen(see paper [15]). The formalism of multicloud computing was derived from problem 1. A whole multicloud with k virtual machines is labelled as C_k. Detailed definition looks as follows [3].

DEFINITION 1. ***Multicloud computing***
Multicloud architecture is represented by a (k+1)–tuple $C_k = (VM_1, \ldots, VM_k, M)$, where VM_i is an abstract virtual machine and M is a cloud middleware of a multicloud environment.

Software $SW_i = (BaseSW, App_1, \ldots, App_n)$ of an abstract virtual machine VM_i from C_k has to satisfy the following condition:

$$\forall 1 \leq i \leq n \; \exists 1 \leq j_1, \ldots, j_m \leq n : check(App_i, App_{j_1}, \ldots, App_{j_m}).$$

It is crystal clear that a tool for an application compatibility is needed. A predicate approach is supposed in this paper. The paper present a predicate $check(x, y_1, \ldots, y_m)$. The predicate ensures that applications from a virtual machine are compatible.

Definition 2. ***Abstract resources checking***
$check(x, y_1, \ldots, y_m)$ is a predicate ($check : \{x, y_1, \ldots, y_m\} \rightarrow \{True, False\}$) which on input gets an abstract application x and abstract applications $y_1, \ldots, y_m$ running on the same virtual machine as x.

$$check(x, y_1, \ldots, y_m) = \begin{cases} True & \leftarrow req(x) \subseteq \cup_{i=1}^{m} prov(y_i) \\ False & \leftarrow otherwise \end{cases}$$

The predicate verifies if there is a conflict among applications running on the same virtual machine. The basic view is that if an application requires a property then another application running on the same virtual machine provides the property.

Definition 3. ***Abstract application interface***
$prov(x) = \{v1, \ldots, v_l\}$, where v_i is a property provided by an application x.
$req(x) = \{v1, \ldots, v_l\}$, where v_i is a property required by an application x.

5. Methodology of a horizontal scalability

Let us start with a characterization of a multicriteria optimization as before (see problem 1).

Problem 2. ***Multicriterial optimization***
Input: *internal virtual resources VR[1], set of optimization criteria OC.*
Problem: *to reallocate internal virtual resources VR based on the requirements OC with respect to running applications in the cloud.*
Output: *internal virtual resources VR'*

The methodology works in the multicloud architecture. The main advantage of the architecture is that it operates with applications as independent components. This approach allows to run the methodology on a virtual machine's level. It supports a creation of a virtual machine, a deletion of a virtual machine or a migration of an application among virtual machines as necessary. So a formal definition looks like follows [14].

Definition 4. ***Intercloud multicriteria optimization***
Multicloud $C_k = (VM_1, \ldots, VM_k, M)$ is balanced if and only if:

$$\forall 1 \leq i \leq k : balanced(VM_i).$$

A concept of balanced multicloud expresses that a multicloud is balanced if and only if its every virtual machine is balanced.

[1]For example it could be virtual machines of IaaS cloud

Definition 5. ***Virtual machine optimization***
$balanced(vm)$ is predicate ($balanced : \{vm\} \rightarrow \{True, False\}$), which gets a virtual machine vm on input. Let us have a set of predicates T, which describes nonoptimality. The predicate is defined as follows:

$$balanced(vm) = \begin{cases} True & \leftarrow \forall p \in T : p(vm) = False \\ False & \leftarrow otherwise \end{cases}$$

System nonoptimality characterizes a set of rules (formally predicates), which we marked by letter T (there are rules such as: the usage of a virtual machine processor is less than 30%, the usage of a virtual machine processor is more than 80% ...).

6. Taxonomy for a multicriteria optimization in a multicloud environment

A taxonomy (see figure 1) describes a functional abstraction of a multicriteria optimization in a multicloud environment. The taxonomy is also composed from properties of entities. The basic entity is a multicloud. The multicloud offers an optimization (as defined in the definition 1). The optimization relates to a monitoring of criteria. The multicloud has to detect nonoptimal virtual machines. If a virtual machine is nonoptimal then an intercloud migration is triggered. The intercloud migration is a one of a basic functionalities offered by the multicloud. However, an election of a suitable configuration of a virtual machine is not always straightforward. This explains why, a hierarchy of virtual machines is needed. The hierarchy organized configurations of virtual machines respect to the criteria of the optimization. The virtual machine's hierarchy has to be an oriented acyclic graph. It is obvious that the hierarchy is a partially ordered set. The partially ordered set is visualised as a Hasse diagram. The Hasse diagram is an abstraction of oriented acyclic graph [12].

The multicloud entity is composed of two entities – a cloud and a criterion. The criterion entity is modelled with the following properties: a condition of a nonsatisfiability, a consequence of a nonsatisfiability and a type of an optimization. The model of the criterion is a derivation of a predicate definition of a balance (see definition 5). The condition describes a state when a virtual machine is unbalanced. The consequence describes the transition from a nonoptimal state to an optimal state. The last property is a type of optimization. As for any (mathematical) optimization, it is essential to know whether the aim is to find a minimum or maximum.

The cloud entity makes an administration of its virtual machine. That means, it has to start a virtual machine and shut down a virtual machine. It makes an actual mirror of a virtual machine (making a snapshot) on the fly. This ability is the reason why is needed backup of a virtual machine and restoration of a virtual machine. Last but not least, the cloud entity has to provide a way to access to a virtual machine. That means, users are able to download and upload data on a virtual machine and they have an access to some type of a command line.

The cloud entity is composed of two entities – a virtual machine and data. Data entity encapsulates other entities

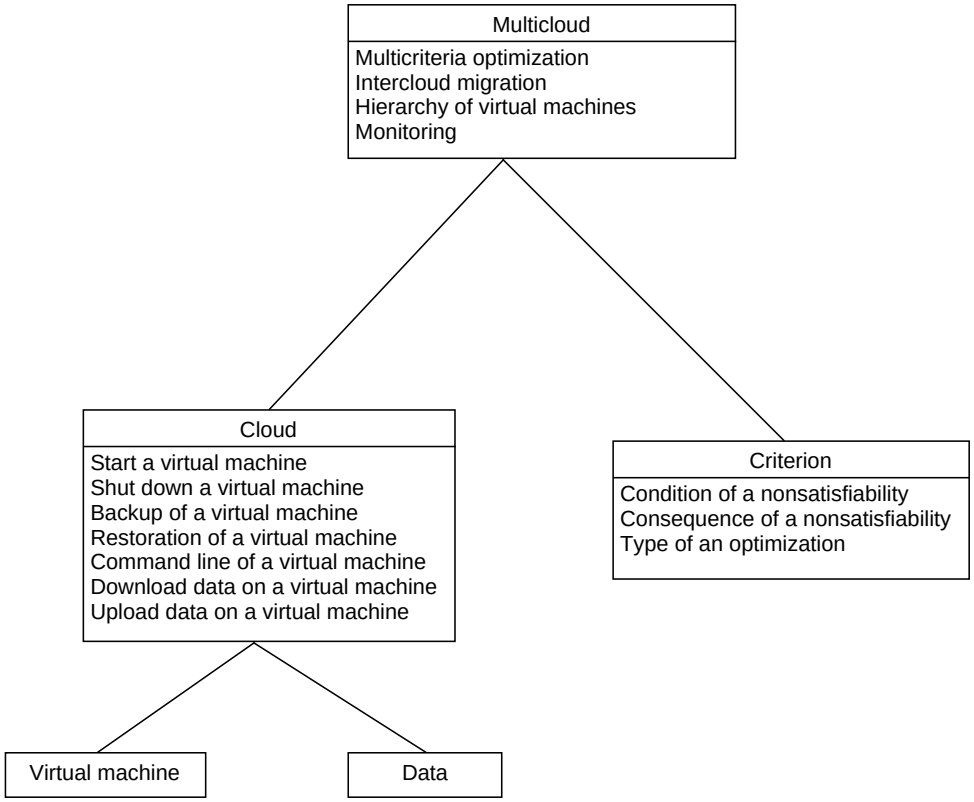

Figure 1: The schema of the taxonomy for a multicriteria optimization

often. However, it is pointlessly from the view of the taxonomy. In the same way, the virtual machine entity is also too low-level from the view of the taxonomy.

7. Application of the multicriteria optimization method in the architecture design

The architecture is based on the formal model. It's main aim is to provide a balanced system. The system is represented as multicloud C_k, whose virtual machines VM_i are an input of the algorithm 1. The algorithm 1 ensures the balance of the system.

Let us explain the application of the multicriteria optimization method in the architecture design on an example. Let us have a multicloud that is composed of two clouds. Eucalyptus is installed on the first cloud and OpenStack is installed on the second cloud. The multicloud architecture has to have drivers for both environments. The driver functionality is represented by element M in the definition of multicloud computing (see definition 1). Let us have four virtual machines – $C_k = (VM_1, VM_2, VM_3, VM_4, M_{Eucalyptus,OpenStack})$. The virtual machines have installed Unix-based operating system (e.g. Linux Ubuntu – $\forall i \in \{1,2,3,4\}\ SW_i : BaseSW = Ubuntu14.04$). Compliance of software interfaces is simplified by the installation from repositories. In case the repositories are unavailable, this functionality has to be ensured within the architecture (as stipulated in the formal model of the multicloud architecture).

The algorithm 1 also gets the set of predicates T. The set expresses the imbalance of the system. The predicates are divided into two parts in the algorithm. The first part consists of the conditions of imbalance (it is represented as tuple C). The second part is contains events, that allow a transition from an imbalanced state c_i to a balanced state r_i (it is represented as tuple R).

The multicriteria optimization is divided into several sub-optimizations which are represented by the set T. Each predicate corresponds to one partial optimization. Thus the state of the system is not defined by one criterion, but by n criteria where $n = |T|$[2]

The aim of our optimization approach is to search events from R, which enable a transition to a more balanced system. The event can be imagined as a virtual machine configuration change (e.g. performance of processor, size of RAM and so on). Virtual machines are monitored regularly, and if imbalanced is occurred, the relevant event is performed.

Let us have a user who does not want to waste of processor performance and RAM – C = (usage of CPU $< 30\%$, usage of RAM $< 30\%$) and R = (reduction of processor frequency, reduce of RAM). Monitoring tools check the actual status of CPU and RAM of every virtual machine in the multicloud system. If there is waste of resources, then relevant event is carried out.

This approach is inspired by the hill climbing search algo-

[2]The tuples C and R has the number of elements equal to the cardinality of the set T. It is a direct consequence of the definitions of the tuples.

rithm [20]. The algorithm is one of the basic approaches which searches the space of states (locally). The actual state is replaced by better neighbouring state in each iteration of the algorithm. It belongs to the greedy algorithms [8].

The main contribution of our method is the automatic configuration of virtual machines due to actual usage. If we would like to compare a nonoptimized system with an optimized system, the main benefit is in an effective allocation of multicloud resources. A provider gets a more balanced load of clouds. A user's virtual machines work only with the resources of the multicloud system which they are using actively. This is one of the main characteristics of cloud computing - pay as you go. This property is not ensured within infrastructures offered as a (cloud) service.

8. Implementation

The implementation is based on the results of previous sections. It is composed of three components (see figure 2): multicloud interface, multicloud manager and cloud driver. Whole implementation is made in Python.

Multicloud interface obtains requests from users and it offers multicloud services in form of REST services. Multicloud manager carries out a integrity of cloud resources provided by different providers and it processes requests from users. The last part of the implementation is cloud driver. Cloud driver transforms a requests which were sent between a cloud manager and a cloud provider.

For a multicriteria optimization, multicloud manager has been extended by monitoring tools and optimization algorithm. In order to know the actual state of a virtual machine load, monitoring tools have been implemented into the multicloud manager. Monitoring is focused on an infrastructure. Virtual machines are monitored (e.g. processor, RAM, storage and so on) and this information is mediated to other subsystems. Nowadays, there are many monitoring tools for big distributed systems. However, the approaches are not usable straightforwardly in a multicloud environment. The main complication is a heterogeneous aspect of the environment. Cloud providers offer cloud services through a different API which makes monitoring difficult. The implementation is able to monitor cloud resources independent of cloud providers.

Monitoring information is important for an optimization algorithm (see algorithm 1). The algorithm controls whether all virtual machines satisfied every optimization condition. The algorithm is testing a balance of a virtual machine. If a criterion is not satisfied then the operations (defined by a user) addressing the situation are carried out. Multicloud interface is expanded with the possibility to enter an optimization criterion.

9. Experiment

Presented approach had been tested in an environment of tailored platforms – platforms created to users' needs. The current situation implies that a future (cloud) services will be more plastic. This means that the services will be adapted to users not vice versa. However it cannot be achieved immediately. The technology has to grow up. The starting point could be to think about multicloud environments – platforms created according to users' requirements. Nowadays platforms are preconstructed. Users need just such a platform that offers a provider or have to move to another type of cloud (it is often a cloud offering an infrastructure). This is problematic in many cases, especially for noninformatics users. Another important problems are caused by an integration of a software executing on the platform and lack of knowledge about an operating system. The characterization of tailored platforms is described by following problem (see problem 3).

Algorithm 1 Multicriteria optimization algorithm

Require: $VR = \{VM_1, \dots VM_k\}$,
$C = \{C_1, \dots C_n\}$,
$R = \{R_1, \dots R_n\}$
Ensure: VR_{opt}
1: $VR_{opt} = \emptyset$
2: **for** $i := 1$ **to** k **step** 1 **do**
3: **if** VM_i is optimal **then**
4: $VR_{opt} := VR_{opt} \cup \{VM_i\}$
5: **else**
6: **for** $j := 1$ **to** n **step** 1 **do**
7: **if** $condition_j(VM_i)$ **then**
8: Make VR_{opt} satisfying rule R_j of $condition_j(VM_i)$.

PROBLEM 3. ***Tailored platform***
Input: *internal virtual resources* VR[3]*, a set of requests on a platform* PR.
Problem: *to reallocate internal virtual resources* VR *into the tailored platform* TP *which satisfies the demands* PR.
Output: *tailored platform* TP.

A portal (see figure 3) supporting tailored platforms is designed according to the problem 3. The portal consists of three parts: authentication, tailored platform provider and virtual machine manager [4]. The portal consists of multicloud resources. As has been mentioned, authentication is not taken to account. It is left to cloud providers.

Platforms have been represented as images. The user chooses an image and virtual machine configuration. The portal creates a platform based on these demands. Such simple approach allows us to give users some form of freedom in the platform context. The images (of platforms) could be prepared by portal provider or other users.

The runtime platform model (see figure 4) describes deployment the elements of the portal. The experiment was carried out on FedCloud's clusters (on infrastructure offered by the Institute of Informatics of Slovak academy of sciences concretely). A cluster has 176 cores with 3 GB RAM and its storage is 50 TB. The biggest offered virtual machine has 8 cores with 16 GB RAM and its storage is 160 GB. Application server of the portal is Django since that portal is implemented at Python.

This work does not focus on security as has been mentioned before. Therefore user connects directly to the portal in a network model (see figure 5). Security issues are left on cloud providers.

[3]For example, it could be virtual machines of IaaS

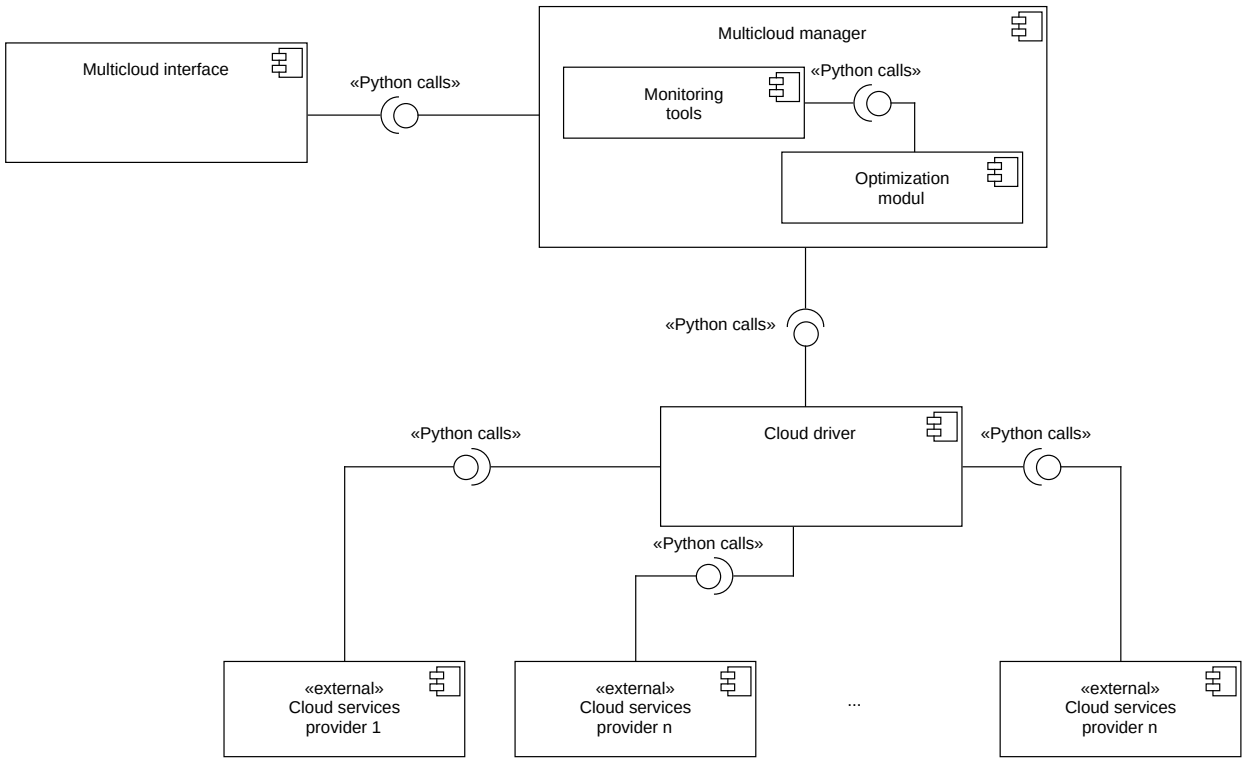

Figure 2: The schema of multicriteria optimization modules

9.1 Prototype

The experimental environment for the portal have been made. The achieved results have been verified through the prototype. That means, the architecture of the portal was multicloud with multicriteria optimization. The form of the prototype was dynamic web application. The application has two basic environment – the virtual machine manager and the manager of the optimization conditions.

Several properties are provided by the virtual machine manager (see figure **??**). The active virtual machines are shown to the user. A virtual machine can be backed up or deleted. The virtual machine manager is responsible for a creation of a new virtual machine and a restoration of a virtual machine (from a snapshot). The user of the portal is not tied with one provider. The provider can be changed with a minimal effort (a provider name is one of the parameters of a creation or a restoration of a virtual machine). The remaining operations of the virtual machine manager are the commands for a virtual machine. The user is able to download or upload a data of a virtual machine and make a connection with a command line of the virtual machine. It is very important to note that the applications could not be very tied with the operating system. Portable applications are very suitable. The reason is that the application has to be able a intercloud migration. The manager of the optimization conditions provides basic functionality for the conditions – overview, creation and deletion of the conditions.

The main contribution of our approach is to make the sharing and allocating of multicloud resources as much effective as possible. An application should work with resources that are needed at the moment. It was achieved through the monitoring of a virtual machine load at regular intervals. Through the monitoring, unbalanced virtual machines had been detected, and they were transformed into balanced.

The balanced multicloud is also much more loaded as an unbalanced multicloud. This is due to the fact, an user makes an estimation of the resources amount needed by an application. However, the resources amount is too much larger. This phenomenon causes that the resources are artificially allocated, even if they could be used by other applications.

9.2 Description of the experiment

The experiment simulated the load of the multicloud system and we observed as our approach increased effectiveness of the multicloud resources. Consequently, the optimized cloud was compared with unoptimized cloud.

The parameters of the experiment. The experiment was carried out on the infrastructure of the Institute of Informatics of the Slovak Academy of Sciences. We worked with three virtual machines at OpenStack environment. A user's behaviour has been simulated in the environment. Optimization focused on RAM and CPU. The optimization rules were as follows: if CPU usage is greater than 80% increase the processor frequency; if CPU usage is less than 20% decrease the processor frequency; if RAM usage is greater than 80% increase size of RAM; if RAM

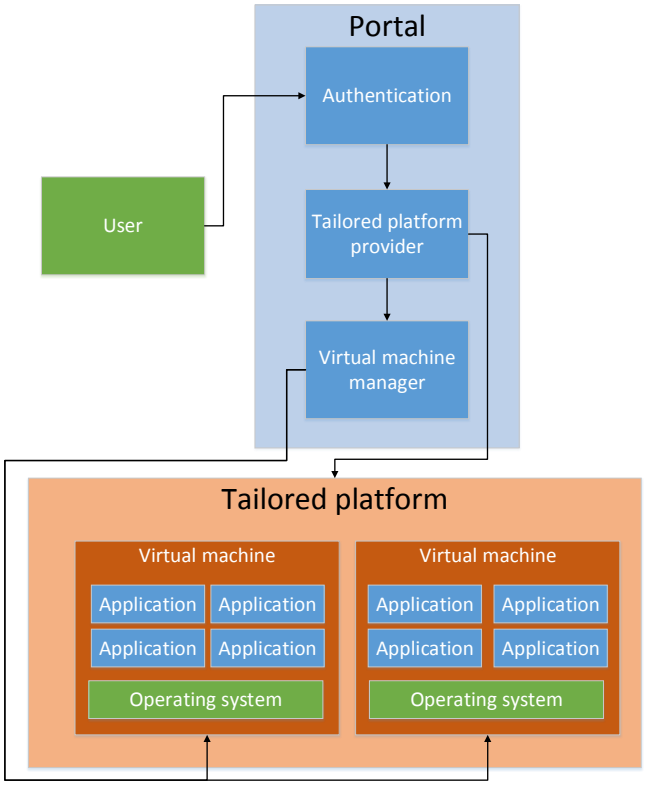

Figure 3: The schema of a portal offering tailored platforms

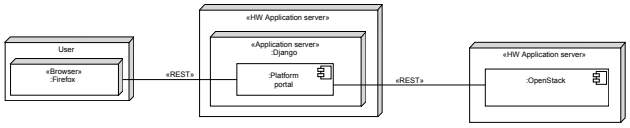

Figure 4: The runtime platform model of a portal offering tailored platforms.

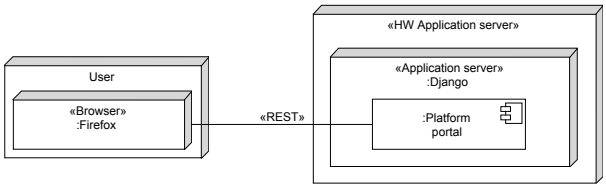

Figure 5: The network model of a portal offering tailored platforms.

usage is less than 20% decrease size of RAM[4]. The monitoring of the virtual machines was carried out every 15 minutes.

The load simulation was performed by programs written in the programming language Python. The cause of load is irrelevant from the point of view of the system behaviour. CPU load was simulated by the infinite loop, multicore load was carried out by parallel execution of the infinite loop and RAM load was performed by saving a large array into RAM.

The environment of OpenStack was not customized. It causes an issue with preconfigured virtual machines (see table 3), namely when we wanted to change one of the parameters of the virtual machine. This is not possible in OpenStack (other cloud environments behave similarly). If users want to change a parameter, they need to choose a different virtual machine configuration – so all parameters are changed. However, this may not be demanded. This unwanted effect can be solved by defining more preconfigured virtual machines.

Table 3: The table shows preconfigured virtual machines in OpenStack.

	CPU cores	Hard disk	RAM
tiny	1	1 GB	512 MB
small	1	20 GB	2 048 MB
medium	2	40 GB	4 096 MB
large	4	80 GB	8 192 MB
xlarge	8	160 GB	16 384 MB

Design of the experiment. The initial configuration of all virtual machines was medium with Ubuntu 14.04. Two virtual machines had CPU load at 15.3%. Another virtual machine had CPU load at 97.6%. All virtual machine had RAM load on 6.4%. This is a common situation. A user has to estimate the amount of resources that is needed in

[4]The OpenStack does not allow to change the processor frequency, but merely the number of processor cores. Therefore, the CPU usage has been represented as the sum of the load of all CPU cores and so the CPU frequency changing operation has been replaced by changing of the CPU core number.

average. A low amount of resources slows down the virtual machine. A high amount of resources is uneconomical from user's point of view. The result of the phenomenon is a time interval when the virtual machine has an inadequate amount of the resources.

Our optimization method adjusted the amount of allocated resources with regard to the actual need. Two virtual machines were running Ubuntu 14.04 only so they had been transformed into configuration small (configuration tiny does not satisfy the minimal hardware requirements of Ubuntu 14.04). The last virtual machine had been transformed into configuration large. Subsequently, the loads of virtual machines were changed and we observed the reaction of our approach on the actual load. The experiment is summarized in the table 4.

Table 4: The table shows the configurations of the virtual machines during the experiment.

	15 min.	30 min.	45 min.	60 min.
VM 1	small	medium	small	small
VM 2	small	medium	large	xlarge
VM 3	large	medium	small	medium

9.3 Evaluation of the experiment

Resources of unoptimized (multi)cloud system remain unchanged. The allocation is independent of the actual load of the virtual machines. As a consequence, the system wastes the resources. Resources are insufficient – the virtual machine is able to work with more resources and so the computation would be accelerated. Or, on the contrary, the amount of the resources is high – the virtual machine is not able to use the amount of the resources and so the computation is uneconomical.

The experiment shows that our approach is able to solve this issue. It relieves the user from the manual managing of virtual machines. However, it was also problems caused by the current implementation of cloud environments. The biggest problem was the small offer of virtual machine configurations and so the user is not able to change one parameter of the virtual machine configuration. They can only increase or decrease all parameters of the virtual machine configuration. This situation can lead into a loop. Improving of a parameter leads into a damage of another parameter and vice versa. A poor offer of configurations can cause that a virtual machine can only be increased (e.g. the computation produces a data).

The frequency of monitoring was important parameter of the experiment. It turned out that the parameter is very specifics for the multicloud system. It depends on the applications that are performed on the virtual machines. It is affected by frequency of resource requirements. The experiment shows that if application often asks for a large amount of resources and the newly allocated resources releases immediately, then the application is not suitable for (multi) cloud. The reason is that, the migration is a time consuming operation.

We realized that the verification should be more rigorous. We plan to deploy the method on real projects and replace simulated data with real data. On the other hand, although the verification was academic, it demonstrates the benefits of our approach clearly.

9.4 Conclusions of the experiment

Several conclusions have been made based on the experiment:

vendor lock in can be dealt with interoperability – the thesis shows that interoperability deals with vendor lock in adequately. A user is able without stronger efforts to change a provider of (multi)cloud services.

support of multicloud environments and multicriteria optimization is insufficient – it turns out that cloud computing will pass through similar developments as computer networks. The result of the evolution will be multicloud computing. However, it is not very widespread for obvious reasons. Thus the resource optimization is not sufficiently investigated. However, if multicloud computing is going to be the next step of cloud computing, it has to respect cloud computing – users pay as they go. This is maintained only if the resources are not wasted.

collaborative service is more accepted then standard – a form of (multi)cloud computing is a very important question. Nowadays, there are two forms: collaborative service and standard. Our research confirmed that there are certain advantages in the development of a collaborative service. The main reason is the model is independent from providers.

application have to be portable – if the migration is performed at the application level, then the application has to be as much independent of operating system as possible. The applications have to behave on the abstract level like containers with clearly defined interfaces.

services have to be adaptable – our experiment shows that flexible services represent a one way to improve the current status of PaaS. The user have not to adapt to services, but the services have to be able to adapt to the user.

10. Approach contribution

Considering the current situation, there does not exist a method providing a multicriterial in multicloud environment, we decided to compare our approach with current cloud providers and scientific multicloud projects (see table 5). There does not exist commercial multicloud environment. One of the biggest problems of cloud computing (in common form) is vendor lock-in. We propose to deal with the problem through interoperability, as other research groups.

Efficiency of virtual resources utilization is another problem which is equally important as previous problem. This aspect is important particularly for a cloud offering an infrastructure as a service (IaaS). Other cloud models have some mechanisms solving the problem by definition. Only IaaS' users have to deal with it.

Amazon offers one of the most used IaaS' cloud. It offers extensive monitoring tools as a CloudWatch. This tool allows some type of an optimization. A user is able to set up some alerts (they are similar with our optimization criteria) checking a load of cloud resources. However they are not resolved automatically. It is left to the user. Another

Table 5: The table compares our approach with approaches presented before.

	Infrastructure as a service	Multicriteria optimization	Multicloud environment	Monitoring tools
Amazon EC2	Yes	Manual	No	Yes
Microsoft Azure	No	–	–	–
Google App Engine	No	–	–	–
Ruby on Rails	No	–	–	–
Saleforce	No	–	–	–
Eucalyptus	Yes	Manual	No	Yes
OpenStack	Yes	Manual	No	Yes
Reservoir	Yes	Manual	Clumsy	Yes
Contrail	Yes	Manual	Yes	Yes
Our approach	Yes	Yes	Yes	Yes

significant cloud provider is Microsoft and its cloud Azure. However Microsoft is relying on that its products are used by every big corporation and so Microsoft decided to offer its products in virtual form over the Internet. Microsoft relieved its users of administration, however on the other hand it locked its users to a greater extent than Amazon. Similar way also chooses Google App Engine, Ruby on Rails and Saleforce.

The situation is quite similar in the cloud open-source projects. Other important characteristics of the projects are that they are "cloud operating systems" only. That means they do not own resources. This may be a problem for many users. Eucalyptus and OpenStack contain, as well as Amazon, monitoring tools. Nevertheless, virtual machines utilization is not optimized in any way.

The situation is more interesting among research projects. Reservoir is an architecture for cloud federation. However development has shown that the architectural design does not take root. The reason is the same as with cloud standards. In order to care out of the architecture, it has to be installed on provider side also. Nowadays, the leading cloud providers do not want to merge with other providers. Therefore, an architecture has to be independent of providers. Reservoir, as well as Contrail, does not support optimization. OPTIMIS is the only project that supports both multicloud computing and optimization. However OPTIMIS focuses on SLA, while presented approach focuses on the user's requirements.

The architecture is based on two aspects. First aspect, that lead us to create a new architecture is that the projects, mentioned above, do not offer a sufficient freedom to new providers of (multi)cloud services. We suggested also the new multicloud taxonomy for multicriteria optimization. The taxonomy clearly specifies a cloud functionality. Thus, if new providers want to be a part of a federation, then they just create a driver characterizing a dictionary of the taxonomy. The second aspect is to provide an optimization tailored by users' requirements. This issue is solved by using a multicriteria optimization.

11. Conclusion

The dissertation thesis addresses the issue in the field of cloud computing. Nowadays, cloud computing is becoming one of the leading computing paradigm of parallel and distributed computing which has a potential to process huge amount of data [1, 18]. Significant IT companies such as Google, Amazon or Microsoft decided to build huge data centres providing cloud computing. Because of virtualization, physical nodes are transformed into virtual nodes which creates the illusion of infinite resources. This service is provided to a huge mass of customers through the Internet.

The thesis is focused on application performance optimization in multicloud environment. The issue is modelled as abstraction problem which the aim is to make optimization based on user's requirement. It is a multicriteria optimization therefore. The user may, for example, require that their application completes the calculation in a particular time or use CPU and RAM optimally...

Based on state of the art, the thesis proposes the method which addresses the optimization task. The method is composed of the model of multicloud environment and taxonomy for multicriteria optimization. The optimization is carried out by a group of logical rules that allows the assessment of individual plans for the allocation of cloud resources upon which the most efficient plan is chosen. This approach allows effective handling of applications running in virtual machines.

Based on the model, the architectural design for the multicriterial optimization in multicloud environment was created. The approach has been subsequently verified and compared with similar approaches. The aim of the experiment was to provide a portal that offers a platform based on users' needs.

Acknowledgement

The author would like to thank his supervisor Assoc. Prof. Ladislav Hluchý and his consultant Dr. Viet Tran for their valuable advices and comments.

References

[1] Danilo Ardagna and Barbara Pernici. "Adaptive service composition in flexible processes". In: *Software Engineering, IEEE Transactions on* 33.6 (2007), pp. 369–384.

[2] David Bernstein et al. "Blueprint for the intercloud-protocols and formats for cloud computing interoperability". In: *Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services*. Venice, Italy, May 2009.

[3] Martin Bobák, Ladislav Hluchý, and Viet Tran. "Abstract Model of k–Cloud Computing". In: *Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2014*. Xiamen, China, Aug. 2014.

[4] Martin Bobák, Ladislav Hluchý, and Viet Tran. "Tailored Platforms as a Cloud Service". In: *Proceedings of 13th International Symposium on Intelligent Systems and Informatics, SISY 2015*. Subotica, Serbia, Sept. 2015.

[5] Martin BobÃąk, Ladislav HluchÃ¡, and Viet Tran. "General Regression in Cloud Computing". In: *Proceedings of the 18th International Conference on Intelligent Engineering Systems, INES 2014*. (SCOPUS, WoS), category B. Tyhani, Hungaria, July 2014.

[6] Emanuele Carlini et al. "Cloud Federations in Contrail". In: *Euro-Par 2011: Parallel Processing Workshops*. Ed. by Michael Alexander et al. Vol. 7155. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2012, pp. 159–168. ISBN: 978-3-642-29736-6. DOI: 10.1007/978-3-642-29737-3_19.

[7] Antonio Celesti et al. "How to enhance cloud architectures to enable cross-federation". In: *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. Miami, USA, July 2010.

[8] Thomas H. Cormen et al. *Introduction to Algorithms*. MIT press Cambridge, 2009.

[9] Ana Juan Ferrer et al. "OPTIMIS: A holistic approach to cloud service provisioning". In: *Future Generation Computer Systems* 28.1 (Jan. 2012), pp. 66 –77.

[10] Borko Furht and Armando Escalante. *Handbook of Cloud Computing*. Springer Publishing Company, Incorporated, 2010.

[11] Gartner. *Top 10 Strategic Technology Trends For 2015*. http://www.forbes.com/sites/gartnergroup/2014/10/21/gartners-top-10-strategic-technology-trends-for-2015/. 2014.

[12] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics; An Applied Introduction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2004.

[13] Tobias Kurze et al. "Cloud Federation". In: *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*. Rome, Italy, Sept. 2011.

[14] M.Bobák, Ladislav Hluchý, and Viet Tran. "Methodology for Intercloud Multicriteria Optimization". In: *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*. Zhangjiajie, China, Aug. 2015.

[15] Minh Binh Nguyen, Viet Tran, and Ladislav Hluchy. "A generic development and deployment framework for cloud computing and distributed applications". In: *Computing and Informatics* 32.3 (2013), pp. 461–485.

[16] Dana Petcu. "Portability and Interoperability Between Clouds: Challenges and Case Study". In: *Proceedings of the 4th European Conference on Towards a Service-based Internet*. Poznan, Poland, Oct. 2011.

[17] Dana Petcu et al. "Portable Cloud applications-From Theory to Practice". In: *Future Gener. Comput. Syst.* 29.6 (2013), pp. 1417–1430.

[18] Barath Raghavan et al. "Cloud control with distributed rate limiting". In: *Proceedings of the ACM SIGCOMM Computer Communication Review*. Kyoto, Japan, Aug. 2007.

[19] B. Rochwerger et al. "The Reservoir Model and Architecture for Open Federated Cloud Computing". In: *IBM J. Res. Dev.* 53.4 (2009), pp. 535–545.

[20] Peter Norvig Stuart Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[21] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. "Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey". In: *ACM Comput. Surv.* 47.1 (2014), 7:1–7:47. ISSN: 0360-0300. DOI: 10.1145/2593512.

[22] Jinzy Zhu. "Cloud computing technologies and applications". In: *Handbook of Cloud Computing*. Springer, 2010, pp. 21–45.

Selected Papers by the Author

Martin Bobák, Ladislav Hluchý, Viet Tran Application Performance Optimization in Multicloud Environment. *Computing and Informatics*, 35(7), 2016 category A, SUBMITTED.

Martin Bobák, Ladislav Hluchý, Viet Tran Methodology for intercloud multicriteria optimization. In *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2015*, Zhangjiajie, China, August 2015, category B.

Martin Bobák, Ladislav Hluchý, Viet Tran Tailored platforms as a cloud service. In *Proceedings of 13th International Symposium on Intelligent Systems and Informatics, SISY 2015*, Subotica, Serbia, September 2015, category B.

Martin Bobák, Viet Tran, Ladislav Hluchý Optimalizácia výkonu aplikácií v multicloudovom prostredí. In *Proceedings of the 10th Workshop on Intelligent and Knowledge oriented Technologies, WIKT 2015*, Košice, Slovakia, November 2015, category C.

Martin Bobák, Ladislav Hluchý, Viet Tran Abstract model of kâĂŞcloud computing. In *Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2014*, Xiamen, China, August 2014, category B.

Martin Bobák, Ladislav Hluchý, Viet Tran General regression in cloud computing. In *Proceedings of the 18th International Conference on Intelligent Engineering Systems, INES 2014*, Tyhani, Hungaria, July 2014, category B.

Martin Bobák, Viet Tran, Ladislav Hluchý Openstack: softvérová platforma pre cloudové počítanie. In *Proceedings of the 9th Workshop on Intelligent and Knowledge oriented Technologies, WIKT 2014*, Smolenice, Slovakia, November 2014, category C.