Computing and Informatics, Vol. 35, 2016, 1359–1385

APPLICATION PERFORMANCE OPTIMIZATION IN MULTICLOUD ENVIRONMENT

Martin Bobák, Ladislav Hluchý, Viet Tran

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
SK–845 07 Bratislava, Slovakia
e-mail: {martin.bobak, ladislav.hluchy, viet.tran}@savba.sk

Abstract. Through the development and accessibility of the Internet, nowadays the cloud computing has become a very popular. This concept has the potential to change the use of information technologies. Cloud computing is the technology that provides infrastructure, platform or software as a service via the network to a huge number of remote users. The main benefit of cloud computing is the utilization of elastic resources and virtualization. Two main properties are required from clouds by users: interoperability and privacy. This article focuses on interoperability. Nowadays it is difficult to migrate an application between clouds offered by different providers. The article deals with that problem in multicloud environment. Specifically, it focuses on the 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: multicloud architecture, method of a horizontal scalability, and taxonomy for multicriteria optimization. The principles of the method were applied in a design of multicriteria optimization architecture, which we verified experimentally. The aim of our experiment is carried on a portal offering a platform according to the users' requirements.

Keywords: Cloud computing, multicloud computing, architecture, multicriteria optimization

1 INTRODUCTION

Cloud computing is one of the buzzwords of computer science. It was among the top 10 strategic technology trends for 2015 [10]. Cloud computing is useful in both the academic and industrial fields. One of the main reasons for the expansion of cloud computing is the need for high-performance computing in almost every modern area. Users require two main properties from clouds:

1. interoperability,
2. privacy.

This paper focuses on the first property – interoperability. Nowadays, it is still difficult to migrate an application between different cloud providers. Almost every provider has a tendency to create specific API. Such approach causes vendor lock in. Users have to adapt to provider's API. Nowadays there is no standard API or a reference cloud implementation.

Results of this paper are based on Minh Binh Nguyen's research [17]. He created a generic development and deployment framework for cloud computing. His model describes interoperability on the operating system level. This paper extends the model, in order to carry out the application performance optimization in multicloud environment.

1.1 Motivation

Multicloud computing offers benefits not only to users, but also to providers. The main advantages are: unified cloud-based architecture, avoiding vendor lock-in, interoperability, scalability, improved accessibility, reduced latency and reduced operating cost.

There exist four fundamental cloud-based collaboration use cases [22]:

- hybrid cloud,
- multicloud computing,
- federated cloud,
- inter-cloud.

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

Users of 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 inter-cloud was introduced by Cisco [3]. The paper describes the inter-cloud as a cloud made up of different clouds. This is in parallel with 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 the 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 [15]. Providers made their idle resources available to each other. Such policy improves scalability and flexibility across the federation, and 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 as a whole.

Federated cloud, inter-cloud and multicloud computing are very new concepts. Computer scientists often use their complete interchangeability. We view inter-cloud as a global federated cloud. So, the federated cloud is the first step towards an inter-cloud. Multicloud computing is the abstract concept as an expression of the fundamentals of the inter-cloud and federated cloud.

The paper is organized into nine sections. Following the introduction, there is a state of the art analysed in Section 2. Section 3 illustrates the architecture of multicloud computing. Section 4 explains the methodology of a horizontal scalability. Section 5 presents the taxonomy for a multicriteria optimization in the multicloud environment. The implementation presented method is described in Section 6. An experiment verifying the approach is demonstrated in Section 7. The results are evaluated in Section 8 and summarized in Section 9.

2 STATE OF THE ART

There is a very good chance that the next evolution step of cloud computing will be a distributed cloud computing – multicloud computing. Authors of this paper [7] identified three evolutionary steps: monolithic stage, 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 the service is 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, the monolithic cloud services (Amazon, Google, Microsoft...) are dominant. Each provider has its own API which makes service administration and maintenance complicated. Consequently, in many cases, a user is restricted by one provider. It is very difficult to work with many providers when cloud services combining is not supported. The heterogeneity causes vendor lock-in [21, 18]. That means, a user becomes dependent on his/her provider. Users are unable to replace the provider without a considerable effort. This situation suits cloud providers who do not engage in it at all [19].

There are two main ways to achieve multicloud computing [22]:

- collaborative service (e.g., middleware),
- standardization (e.g., interfaces or API abstraction).

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 Multicloud Computing

The following table of standards is based on the survey [22]. It implies two important facts. Those facts influenced our research.

The first fact is that the 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 follow: Cloud providers do not want to agree on a uniform standard. They rather prefer the vendor lock-in [18].

The second fact is that many standards have been produced, but none of them has become globally recognized for multicloud computing [18, 22] (see Table 1). It is the consequence of the situation that providers are not able to agree on a single standard and this situation is good for them. Based on these facts, we decided to focus only on collaborative services.

2.2 Cloud API Abstraction Libraries

Another way of achieving multicloud computing is a cloud API abstraction. There are several cloud API abstraction libraries. The main benefit of the abstraction is an independence from the cloud providers. However, the libraries are facing similar issues as the standardization, because it is difficult to develop a common API supported by every cloud provider. Another trait of the libraries is that the degree of a support of the provider's features varies among libraries. Some features are not even supported by the libraries and so working with clouds is not easier with them. Several cloud API abstraction projects have been studied.

Jclouds[1] is an open-source library for managing cloud resources from 30 different cloud providers (such as Amazon, Azure, GoGrid, OpenStack, Rackspace, and so on). The library is written in Java.

[1] `http://jclouds.apache.org/`

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	✓	✓		✓				✓

Table 1. Comparison of current standards according to focused problem – Provisioning (Pr), Portability (Po), Service Level Agreement (SLA), Security (S), Monitoring (M), Economy (E), Network (N), Autonomics (A). (Adapted from [22].)

Fog[2] is an open-source library for managing cloud resources from 43 different cloud providers (such as Amazon, DigitalOcean, IBM Softlayer, Joyent, OpenStack, Rackspace, vCloud, XenServer, and so on). The library is written in Ruby.

Boto[3] is an open-source library for managing cloud resources provided by Amazon. However, it is also applicable to cloud that uses the same interface (like Eucalyptus or OpenStack). The library is written in Python.

δ-cloud[4] is an Apache project, that is currently retired. The main focus of the project is managing cloud resources. 15 different cloud providers (such as Amazon, GoGrid, OpenNebula, Eucalyptus, Rackspace, OpenStack, and so on) have been supported by the project. δ-cloud offers cloud resources via a REST API, so it is independent from programming languages (although δ-cloud provides client libraries for Ruby, Java, C, and Python). The project is written in Ruby.

Libcloud[5] is an open-source library for managing cloud resources from 55 different cloud providers (such as Elastic Hosts, RackSpace, Eucalyptus, AWS, Joyent, OpenNebula, GoGrid, Enomaly, SliceHost, vCloud, and so on). The library is written in Python.

	Supported Languages	**Number of Supported Providers**
Jclouds	Java, Clojure	30
Fog	Ruby	43
Boto	Python	1
δ-cloud	REST-capable	15
Libcloud	Python	55

Table 2. Comparison of current cloud API abstraction libraries

2.3 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 service can be performed either on a user's side or it can be offered as a third party service. The following analyses of collaborative services is based on the survey [22].

2.3.1 Reservoir

Reservoir model [21] is the architecture for cloud federation. The authors distinguish between two operations in the model. That also means that the model has

[2] `http://fog.io/`
[3] `http://boto.cloudhackers.com/`
[4] `http://deltacloud.apache.org/`
[5] `http://libcloud.apache.org/`

two types of Reservoir sites – infrastructure providers sites, and service providers sites. Customers are offered by their service providers the applications that exploit an infrastructure provided by a particular infrastructure provider. Service providers do not own 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).

Reservoir architecture has three main components (see Figure 1): service manager, virtual execution environment manager and virtual execution environment host. The service manager is on the top layer. It interacts with service providers – it receives their service manifests (e.g., deployment of 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 Reservoir sites (this property creates the federative cloud) and places virtual environments into virtual execution environment hosts. The bottom layer is virtual execution environment host. It is responsible for supporting various types of virtualization, virtual environment monitoring and it provides migration of a virtual environment within the federated cloud.

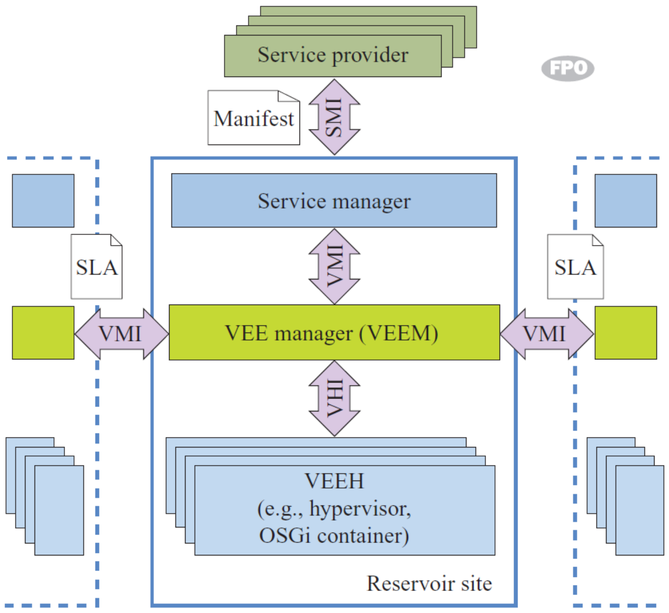


Figure 1. Basic components of Reservoir and their interfaces (OSGi: Open Services Gateway initiative; SMI: service management interface; VMI: VEE management interface; VHI: virtual host interface). Taken from [21].

2.3.2 Contrail

Contrail project [6] enables the utilization of cloud resources offered by different cloud providers (horizontal integration) and it also gives an opportunity to work with IaaS and PaaS (vertical integration). Contrail is based on agents who act as intermediary between cloud users and cloud providers. Contrail consists of three layers (see Figure 2): 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 the 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 the provider watcher. The adapters layer is the bottom layer. Adapters are subdivided by type of resources provided into external and internal modules. Internal modules make virtual infrastructure network, global autonomous file system and virtual execution platform available. External modules translate messages from federated cloud to cloud providers.

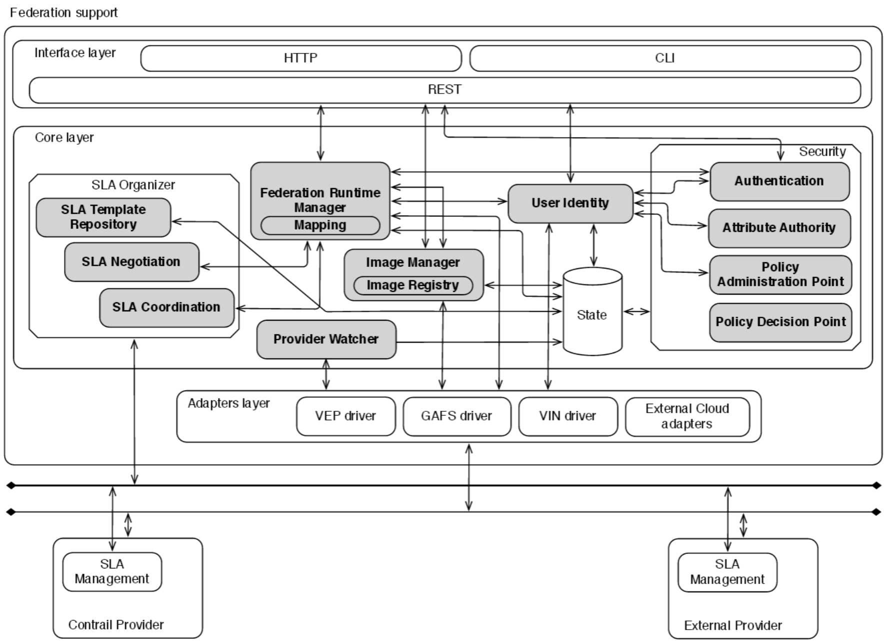


Figure 2. Basic components of Contrail. Taken from [6].

2.3.3 OPTIMIS

The main components of OPTIMIS (see Figure 3) are as follows. Services are created by the service builder. It has integrated development environment that allows a development (via the programming model) and a configuration (via the configuration manager). SaaS cloud is received at the end (a service is offered over the Internet and it is executed in multicloud environment). The basic toolkit supplies the functionality for service management (e.g. monitoring, security...) to other components. Once the service set up is finished, deploying of the service is started. Deploying is carried out by its deployment engine. It is checking for the best infrastructure provider for the service. If the search is successful, deployment engine sends a request to the 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 monitored by service optimizer which is able to make its subsequent migration to the other infrastructure if necessary, and the service level agreement is not broken.

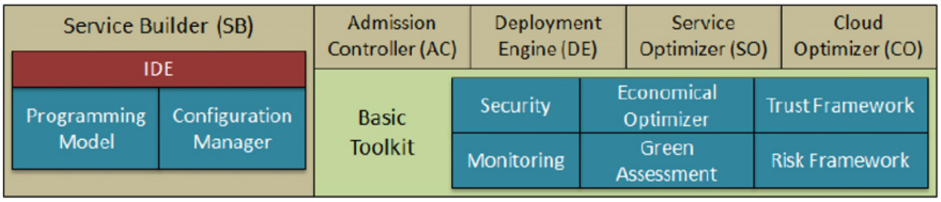


Figure 3. Components of OPTIMIS. Taken from [8].

2.3.4 PaaSage

PaaSage project[6] is a model-based cloud platform for deployment and optimization of applications among different clouds. The applications are automatically scaled according to their demands. Multicloud computing is achieved through its Cloud Application Modelling and Execution Language (CAMEL). CAMEL (see Figure 4) comprises four main parts – Cloud Modelling Language (CloudML) [11], Saloon [20], CERIF [13], and Scalability Rule Language (SRL) [14]. The project is accessible through Cloud Modelling Framework (CloudFM) [9]. CloudFM has two main components: modelling environment (CAMEL) and Models@run-time environment.

2.3.5 ModaClouds

ModaClouds project [2] is a MOdel-Driven Approach for design and execution of applications on multiple clouds. The project is built on a Model-based Software

[6] `http://www.paasage.eu/`

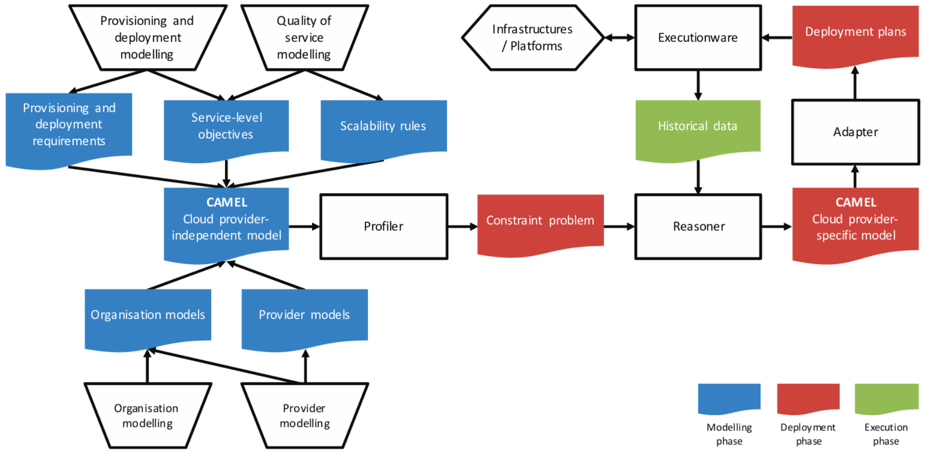


Figure 4. PaaSage workflow. Taken from [1].

Engineering. It aims to support application developers and operations in exploiting (multi)clouds. ModaClouds is composed of (see Figure 5 a)): a Decision Support System (DSS) to provide the selection of cloud providers, an Integrated Development Environment (IDE) to support a migration of applications between clouds, a design of software systems and a partial code generation, and a run-time environment enable a monitoring and a self-adaptation. The user models a cloud application at different levels of abstraction through IDE (see Figure 5 b)): a Computation Independent Model (CIM) in which non-functional requirements and constraints are designed, a Cloud-Provider Independent Model (CPIM) in which CIM artefacts are semi-automatically processed and cloud concepts are incorporated into the model but kept abstract from any cloud specific platform, and a Cloud-Provider Specific Model (CPSM) in which CPIM artefacts are semi-automatically translated and deployment artefacts are created.

3 MULTICLOUD COMPUTING ARCHITECTURE

Building on the above mentioned ideas, our architecture was 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 the vendor lock-in [21, 18].
2. Cloud customers mostly focus on two types of clouds – SaaS and IaaS [21]. This is due to the fact that a platform (in the current form) is suitable for small group of users. The situation has resulted from two factors. If users are interested in a particular product, they choose SaaS. On the other hand, if users

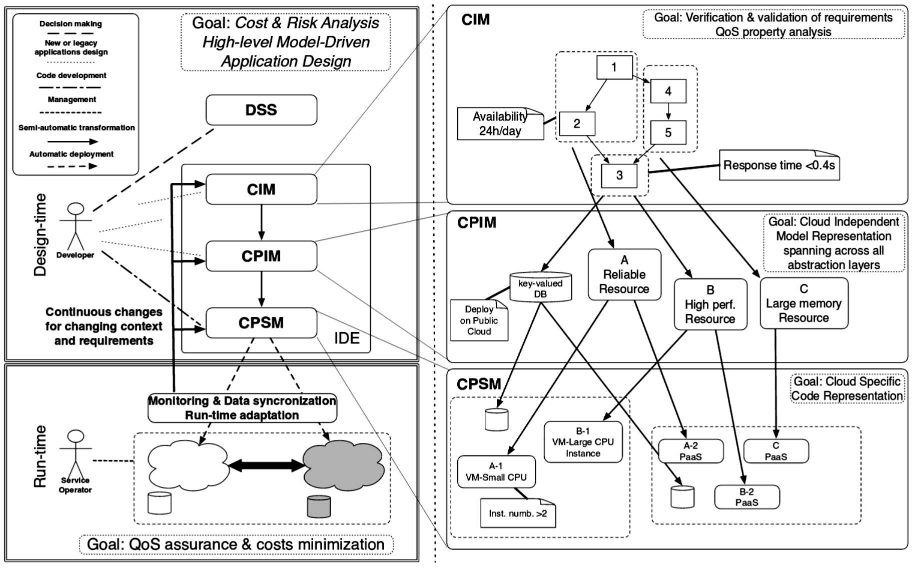

Decision making
New or legacy applications design
Code development
Management
Semi-automatic transformation
Automatic deployment
Goal: Cost & Risk Analysis High-level Model-Driven Application Design
DSS
CIM
CPIM
CPSM
IDE
Design-time
Developer
Continuous changes for changing context and requirements
Monitoring & Data syncronization Run-time adaptation
Run-time
Service Operator
Goal: QoS assurance & costs minimization
CIM
Goal: Verification & validation of requirements QoS property analysis
Availability 24h/day
Response time <0.4s
CPIM
Goal: Cloud Independent Model Representation spanning across all abstraction layers
key-valued DB
Deploy on Public Cloud
A Reliable Resource
B High perf. Resource
C Large memory Resource
CPSM
Goal: Cloud Specific Code Representation
B-1 VM-Large CPU Instance
A-1 VM-Small CPU
Inst. numb. >2
A-2 PaaS
C PaaS
B-2 PaaS
a)
b)

Figure 5. ModaClouds approach. Taken from [2].

want to be more variable (or they are interested in virtualized environment), they are currently forced to use IaaS. PaaS appears to be the only suitable for this problem, at the first glance (see Section 2). That is the reason why the article focuses on IaaS at multicloud computing environment.

3. There are several ways to achieve multicloud computing [19]. The most common ways are 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 cloud providers refuse to accept a common interface [22]. That is the reason why the article presents the collaborative service.

4. Collaborative service, offering infrastructure, is a young field of cloud computing. There exist only several projects focusing on such services. Our article has presented five FP7 – Reservoir [21], Contrail [6], OPTIMIS [8], ModaClouds [2], and PaaSage[7]. The main difference between these projects is the way the 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 a 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 the agent approach to achieve multicloud services. However, the approach is redundant (it

[7] `http://www.paasage.eu/`

is too heavy load on a network). In this case, agents are used for discovering and selection of resources. OPTIMIS is not the architecture, it is a software kit allowing some kind of optimization. ModaClouds and PaaSage are projects based on Model-based Software Engineering. They offer modelling as a service. This approach has many arguments pro and con. The biggest advantages of the approach are interoperability and vendor lock-in avoiding. However, a user of the projects needs to learn a new (modelling) language. This is non-trivial and problematic for non-IT users. One of our aims is the user-friendliness. The proposed approach is dealing with this issue without creating a specific language. It is designed for a common user.

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

- allow scalability,
- allow interoperability on a virtual machine level (of an infrastructure offered as a (multi)cloud),
- avoid vendor lock-in,
- create collaborative service,
- be independent of providers,
- allow flexible provider administration (new provider changes the architecture minimally),
- user-friendly approach.

Let us start with a characterization of the multicloud issue. Multicloud computing is a technology which obtains virtual resources from different providers and returns back uniform multicloud virtual resources. The aim is to merge obtained resources in order to be accessible in the uniformed way. The formalism of multicloud computing was derived from this characterization. A whole multicloud with k virtual machines is labelled as C_k. Detailed definition looks as follows [4].

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 the application compatibility is needed. A predicate approach is supposed in this paper. The paper presents 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.

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

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.

4 METHODOLOGY OF A HORIZONTAL SCALABILITY

Let us start with a characterization of multicriteria optimization, as known before. A method receives internal virtual resources[8] and set of optimization criteria. The method reallocates internal virtual resources based on the requirements with respect to running applications in the cloud.

The methodology works in the multicloud architecture. The main advantage of the architecture is that it operates with applications as independent components. 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 [16].

Definition 4 (Inter-cloud 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 multicloud is balanced if and only if its every virtual machine is balanced.

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

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

a set of predicates T, which describes non-optimality. The predicate is defined as follows:

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

The methodology deals with infrastructure metrics of a multicloud system, which means that the monitoring and optimization criteria are aimed at virtual machines. The proposed approach is able to optimize every key performance indicator (KPI) that is capable of being modelled through a predicate. The KPI has to be detectable via a Boolean condition and improvable via a finite set of actions (e.g., CPU utilization, RAM utilization, resource cost, and so on).

Non-optimality of the system is characterized by 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 %...).

5 TAXONOMY FOR A MULTICRITERIA OPTIMIZATION IN A MULTICLOUD ENVIRONMENT

A taxonomy (see Figure 6) 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 the criteria monitoring. The multicloud has to detect non-optimal virtual machines. If a virtual machine is non-optimal then an inter-cloud migration is triggered. The inter-cloud migration is one of the basic functionalities offered by multicloud. However, the selection of a suitable configuration of the virtual machine is not always straightforward. This explains why a hierarchy of virtual machines is needed. The hierarchy organized configurations of virtual machines respects the criteria of the optimization. The virtual machine's hierarchy has to be in the form of the directed acyclic graph. It is obvious that the hierarchy is a partially ordered set. The partially ordered set is visualized as a Hasse diagram. The Hasse diagram is an abstraction of directed acyclic graph [12].

The multicloud entity is composed of two entities – cloud and criterion. The criterion entity is modelled with the following properties: a condition of a non-satisfiability, a consequence of non-satisfiability and a type of optimization. The model of the criterion is a derivation of the predicate definition of the balance (see Definition 5). The condition describes the state when a virtual machine is unbalanced. The consequence describes the transition from a non-optimal state to the 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 value.

The cloud entity makes the administration of its virtual machine. That means, it has to start the virtual machine and shut down the virtual machine. It makes

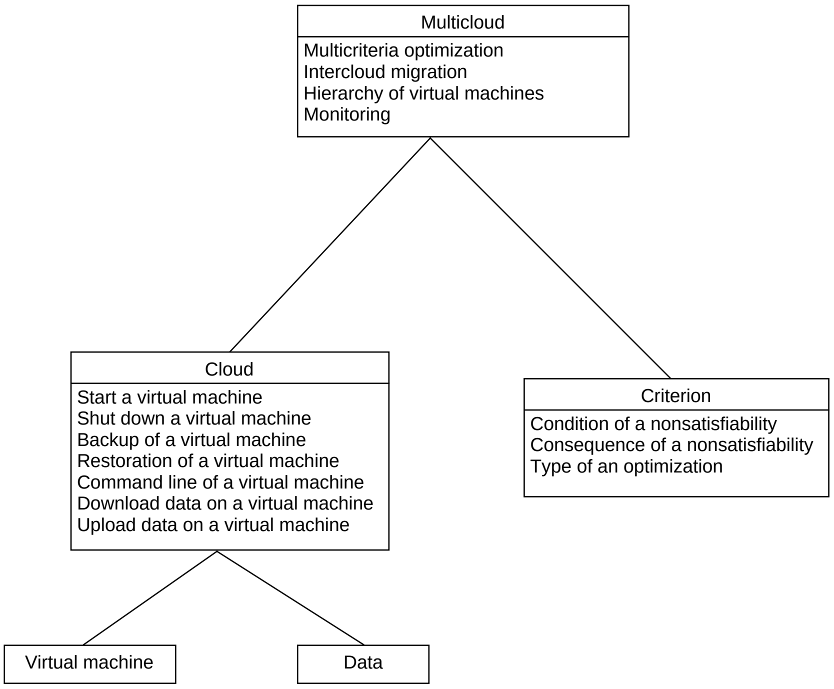

Multicloud
Multicriteria optimization
Intercloud migration
Hierarchy of virtual machines
Monitoring
Cloud
Start a virtual machine
Shut down a virtual machine
Backup of a virtual machine
Restoration of a virtual machine
Command line of a virtual machine
Download data on a virtual machine
Upload data on a virtual machine
Criterion
Condition of a nonsatisfiability
Consequence of a nonsatisfiability
Type of an optimization
Virtual machine
Data

Figure 6. Schema of taxonomy for multicriteria optimization

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

The cloud entity is composed of two entities – the virtual machine and data. Data entity often encapsulates other entities. However, it is pointless from the taxonomy view. In the same way, the virtual machine entity is also too low-level from the taxonomy point of view.

6 IMPLEMENTATION

The implementation is based on the results of the previous sections. It is composed of three components (see Figure 7): multicloud interface, multicloud manager and cloud driver. Whole implementation is made in Python.

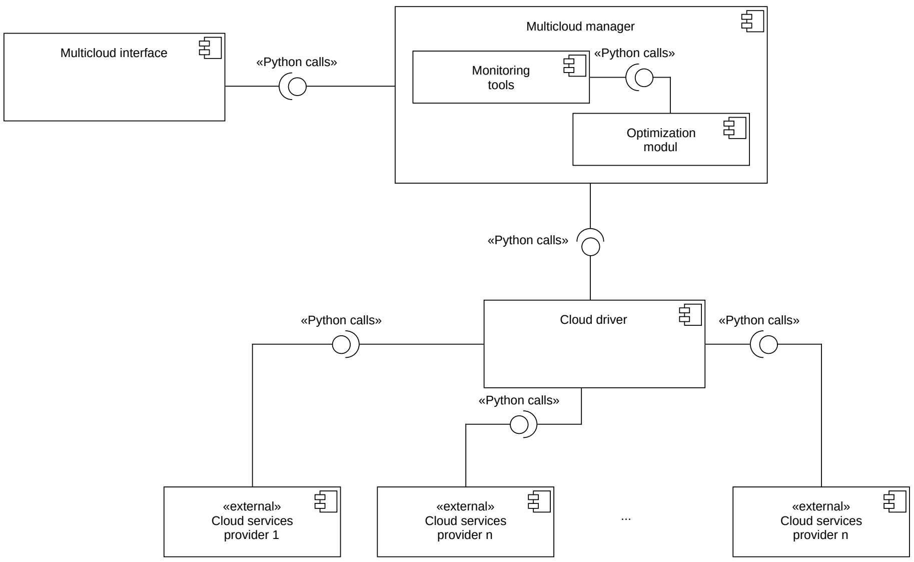


Figure 7. Schema of multicriteria optimization modules

Multicloud interface obtains requests from users and it offers multicloud services in form of REST services. Multicloud manager carries out the 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 requests which were sent between the particular cloud manager and the particular cloud provider.

For the purpose of the multicriteria optimization, multicloud manager has been extended by monitoring tools and optimization algorithm. In order to know the actual state of the virtual machine load, monitoring tools have been implemented into the multicloud manager. Monitoring is focused on the infrastructure. Virtual machines (maintained by particular cloud providers) are monitored (e.g., processor, RAM, storage, etc.) 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 the multicloud environment. The main complication is a heterogeneous aspect of the environment. Cloud providers offer cloud services through different API which makes monitoring difficult. The implementation is able to monitor cloud resources independent of cloud providers.

Monitoring information is important for the optimization algorithm (see Algorithm 1). The algorithm controls whether all virtual machines satisfied every optimization condition. The algorithm is testing the 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 the optimization criterion.

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 $condition_j(VM_i)$.
```

The approach supports multiple criteria. This should lead to a state when the objectivities are conflicting. The situation often does not have a single solution and such approach tries to find a trade off solution.

Each condition has a rule describing how to move from unbalanced to balanced virtual machine and a priority defining the importance of the condition. For example, a user wants to optimize a CPU utilization and so one of the conditions is: if CPU utilization is greater than 85 % (then a virtual machine is unbalanced). The rule of the condition should be: to increase a frequency of the virtual machine. The proposed approach uses the action of the infringed condition until the objective is not satisfied or in a contradiction to a rule with higher priority.

7 EXPERIMENT

Presented approach had been tested in the environment of tailored platforms – platforms created to comply the needs of the users. Tailored platform is offered as a (multi)cloud service in the form of a flexible and dynamically generated platform. The system is able to receive the users demands and to offer a suitable platform according to the demands.

For example, a user focuses on a machine learning, s/he is a Linux user and wants to program in language C. So a configuration of the tailored platform is as follows: an operating system is Linux (e.g., Ubuntu) and it has preconfigured tools for machine learning that support programming in language C (that should be specified by the user). Another user is a bioinformatics, s/he is an MS Windows user and wants tools for a genome analysis. So a configuration of the tailored platform is as follows: an operating system is MS Windows and it has preconfigured tools for the genome analysis.

We assume, that this is a future of PaaS clouds, according to the current situation. The PaaS clouds need to be independent of a community that uses them. Otherwise, the PaaS clouds will depend on a small group of suitable users and loose more bigger group of potential users which do not need an infrastructure however the offered platform does not match their needs. This implies that the services will be adapted to users, and-not vice versa. However, this cannot be achieved immediately. The technology has to grow up. The starting point could be to think about multicloud environments – the platforms created according to users' requirements. Nowadays, the platforms are pre-constructed. Users, however, need a platform that offers a provider or where they have the ability to move to another type of cloud (it is often the cloud offering an infrastructure). This is problematical in many cases, especially for non-informatics users. Other important problems are caused by the integration of a software executing on the platform and the insufficient knowledge about the operating system. The characterization of tailored platforms is as follows. The portal obtains users requirements and internal virtual resources[9]. The aim of the portal is to reallocate internal virtual resources into the tailored platform which satisfies the users' demands.

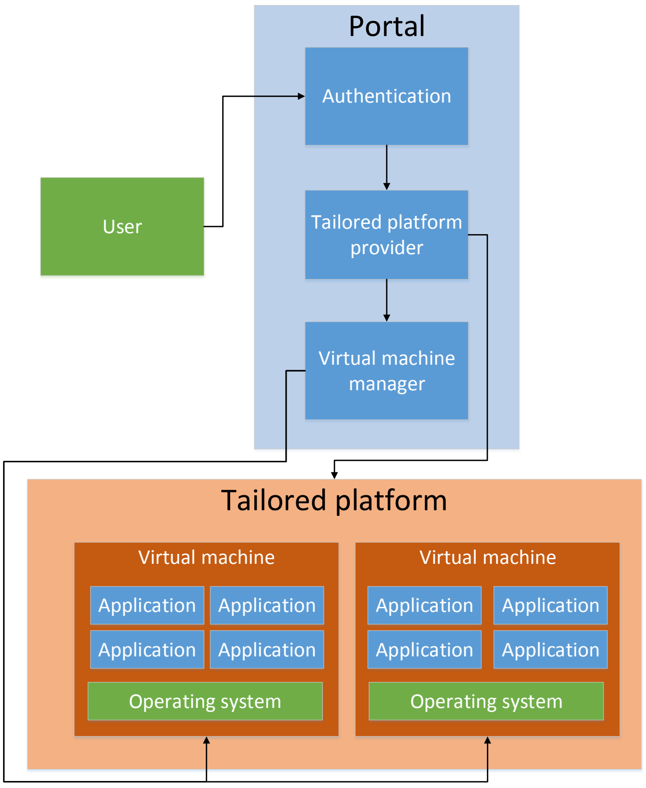


Figure 8. Schema of a portal offering tailored platforms

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

A portal (see Figure 8) supporting tailored platforms is designed according to the characterization of the portal. The portal consists of three parts: authentication, tailored platform provider and virtual machine manager [5]. The portal consists of multicloud resources. As it has already been mentioned, authentication is not taken into 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 freedom in the platform context. The images (of platforms) could be prepared by a portal provider or by other users.

The runtime platform model (see Figure 9) describes deployment of the elements of the portal. The experiment was carried out on FedCloud's clusters (on the infrastructure offered by the Institute of Informatics of the Slovak Academy of Sciences). 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 because that portal is implemented at Python.

A creation of a portal prototype is a part of the experiment. The prototype carries out applications running on Linux OS (a portable application is an alternative on Windows OS). The reasons for this decision are:

1. the proposed system should not require learning new skills,
2. there does not exist standard for applications,
3. an application migration demands of a realizable manipulation with the applications (it requests a sufficient independence of an operation system).

The proposed system is in the form of a dynamic web application and is controlled through its GUI. The GUI supports working with virtual machines (to create/restore/backup/delete/get information about a virtual machine, download/upload a data from/to a virtual machine, run a command line of a virtual machine), applications (to backup of an application, and migrate an application between virtual machines) and optimization criteria (to create/delete/get information about an optimization criterion). The proposed system is connected to a SQLite database where the information about optimization criteria, virtual machines, and a hierarchy of the virtual machines is stored.

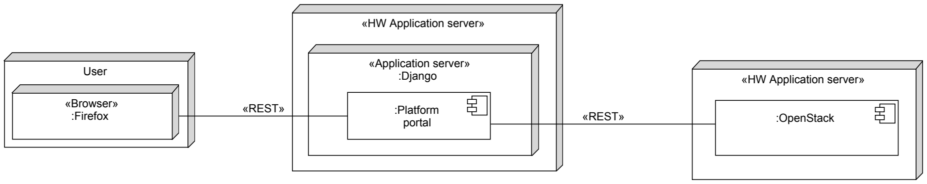


Figure 9. Runtime platform model of a portal offering tailored platforms

This work does not focus on security, as mentioned before. Therefore the user connects directly to the portal in a network model (see Figure 10). Security issues are left on cloud providers.

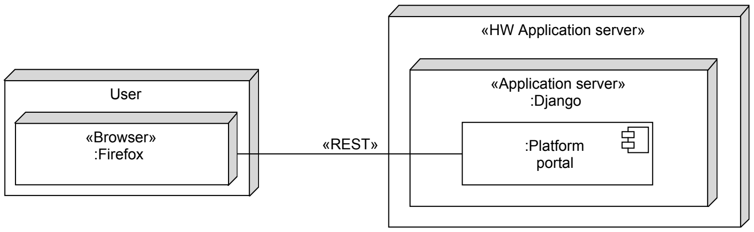


Figure 10. Network model of a portal offering tailored platforms

The approach was also evaluated through a synthetic scenario. The experiment was performed on an infrastructure of the Institute of Informatics of the Slovak Academy of Sciences. The resources were offered as a cloud services in the form of the infrastructure through an OpenStack environment (a survey of preconfigured virtual machines is shown in Table 3).

	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

Table 3. Preconfigured virtual machines in OpenStack

The aim was to balance the resources of the portal offering tailored platforms according to a resource utilization and optimization criteria. The resource utilization has been generated through simple python scripts (infinite loop and saving a large array into RAM). The cause of the utilization is irrelevant. The important thing is how the proposed architecture reacts to imbalanced states.

Imbalanced states (predicates) with their associated actions are described in Table 4. The monitoring was carried out every 15 minutes and the number of virtual machines was three. An operating system of each virtual machine was Ubuntu 14.04.

The experiment has a minor issue caused by the OpenStack environment. The environment does not support changing of one parameter of the virtual machine. It is necessary to change the whole configuration of the virtual machine (other cloud environments behave similarly). This action may not be demanded and can be solved by defining more configurations of a virtual machine.

The experiment is summarized in the Table 5. Our multicriteria optimization approach was able to customize the resources of the multicloud system according to

Imbalanced State	**Set of Actions**
CPU usage is greater than 80 %	increase the number of cores of the processor
CPU usage is less than 20 %	decrease he number of cores of the processor
RAM usage is greater than 80 %	increase size of RAM
RAM usage is less than 20 %	decrease size of RAM

Table 4. Optimization criteria used in the experiment

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

Table 5. Configurations of the virtual machines during the experiment

the actual load. The unoptimal situation is when the allocation of the resources is static. The resources are unused or needed in another virtual machine and so the whole multicloud is working non-optimally.

We realized that the verification should be more rigorous. We plan to deploy the approach into real projects and thus replace the simulated data with real data. On the other hand, although the verification is synthetic, it demonstrates the benefits of presented approach.

8 EVALUATION

Considering the current situation, when a method providing multicriteria in multicloud environment does not exist, we decided to compare our approach with the current cloud providers and scientific multicloud projects (see Table 6). No commercial multicloud environment exists nowadays. One of the biggest problems of cloud computing (in a 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 the previous problem. This aspect is important particularly for the cloud offering the infrastructure as a service (IaaS). Other cloud models have some mechanisms solving that problem by means of a definition. Only IaaS' users have to deal with it.

Amazon offers one of the most used IaaS' cloud. It offers extensive monitoring tools as the CloudWatch. This tool allows some type of 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. Solution is left to the user. Another significant cloud provider is Microsoft and its cloud Azure. However, Microsoft relies on the fact that its products are used by every big corporation, and thus Microsoft decided to offer its products in virtual form over the Internet. Microsoft makes it easier to users, and void their administration steps, on the other hand, it however, locked the users much more than

Amazon did. Google App Engine, and Saleforce have chosen a similar approach as well.

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

The situation is more interesting among research projects. Reservoir is the architecture for cloud federation. However, the development has shown that the architectural design does not take a root. The reason is the same as with cloud standards. In order to take care of the architecture, it has to be also installed on the provider's side. Nowadays, the leading cloud providers do not want to merge with other providers. Therefore, the architecture has to be provider independent. 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 user's requirements. PaaSage and ModaClouds are Model-driven approaches. Given that the users have to learn modelling language that could be the cause of several problems. Both projects offer modelling (of cloud applications) as a service, that means the users are not able to apply for a pure infrastructure. PaaSage is the most similar to our methodology. Many objectivities addressed by PaaSage are covered by proposed approach. However, they focus on different type of users. PaaSage provides modelling of a cloud application. On the other hand, our approach is aimed on offering of an infrastructure which resources are optimized. So the realization of overlapped objectivities is different.

Several autoscaling projects exist. The proposed approach is compared with two of them – OpenShift and ScalR. Those projects offer environment (platform) for developing and deploying an application. The focus of the projects is similar to OPTIMIS project. They are aimed on a software that is offered as a cloud service while our approach is focused on a virtual machine of the infrastructure offered as a cloud service. The other important feature of the proposed method is the customization of optimization, an ability to govern a multicloud system in real time, and an easy integration of a new multicloud participant into the system.

The architecture is based on two aspects. The first aspect, that leads 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. In addition, we have suggested the new multicloud taxonomy for multicriteria optimization. The taxonomy clearly specifies the cloud functionality. Thus, if new providers want to be a part of a federation, they just create a driver characterizing a dictionary of the taxonomy. The second aspect is to provide an optimization tailored to users' requirements. Multicriteria optimization is able to solve the issue.

	Infrastructure as a Service	**Multicriteria Optimization**	**Multicloud Environment**	**Monitoring Tools**
Amazon EC2	Yes	Manual	No	Yes
Microsoft Azure	No	–	–	–
Google App Engine	No	–	–	–
Saleforce	No	–	–	–
Eucalyptus	Yes	Manual	No	Yes
OpenStack	Yes	Manual	No	Yes
Reservoir	Yes	Manual	Clumsy	Yes
Contrail	Yes	Manual	Yes	Yes
OPTIMIS	Yes	focused on SLA	Yes	Yes
ModaClouds	No	Manual	Yes	Yes
PaaSage	No	Yes	Yes	Yes
OpenShift	No	Uncustomizable	No	Yes
ScalR	No	Uncustomizable	No	Yes
Our approach	Yes	Yes	Yes	Yes

Table 6. Comparison of our approach with approaches presented so far. Our architecture is oriented on an infrastructure offered as a cloud service. Multicriteria optimization at multicloud environment is the main aim of the architecture. The comparison is divided into four parts – commercial projects, open-source projects, research projects, and autoscaling projects.

9 CONCLUSION

The proposed approach has focused on cloud computing, concretely on an issue of a resource wasting in a multicloud environment. Nowadays, a user is responsible for virtual machine management of an infrastructure offered as a (multi)cloud service.

In order to solve the issue, several abstract models has been created. The environment (of a multicloud computing) is expressed as the multicloud computing model (see Definition 1). The methodology is described through the multicriteria model that is based on the multicloud computing model (see Definition 4). The core of the abstract model is a balanced virtual machine. That means the amount of resources assigned to the virtual machine has to match running applications (and their demands). Finally, the paper presented a taxonomy for a multicloud system supporting a multicriteria optimization. It is a form of abstraction of every system of this kind.

The theoretical results were applied at the prototype the implementation of which is described in Section 6. Subsequently, the prototype was used in the experiment (see Section 7). The aim of the experiment was to provide the portal that offers a platform based on the specific requirements of users. After that, the approach was also evaluated through the synthetic scenario and compared with the other similar approaches.

Acknowledgement

This work is supported by the project VEGA 2/0167/16, H2020 project EGI-Engage EU H2020-654142, and ERDF – ITMS 26240220060.

REFERENCES

[1] Achilleos, A. P.—Kapitsaki, G. M.—Constantinou, E.—Horn, G.—Papadopoulos, G. A.: Business-Oriented Evaluation of the PaaSage Platform. 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC), Limassol, Cyprus, December 2015, pp. 322–326.

[2] Ardagna, D.—Di Nitto, E.—Casale, G.—Petcu, D.—Mohagheghi, P.—Mosser, S.—Matthews, P.—Gericke, A.—Ballagny, C.—D'Andria, F. et al.: MODAClouds: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds. Proceedings of the 4th International Workshop on Modeling in Software Engineering, Zurich, Switzerland, June 2012, pp. 50–56.

[3] Bernstein, D.—Ludvigson, E.—Sankar, K.—Diamond, S.—Morrow, M.: Blueprint for the Intercloud-Protocols and Formats for Cloud Computing Interoperability. Proceedings of the 2009 Fourth International Conference on Internet and Web Applications and Services, Venice, Italy, May 2009.

[4] Bobák, M.—Hluchý, L.—Tran, V.: Abstract Model of k-Cloud Computing. Proceedings of the 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2014), Xiamen, China, August 2014.

[5] Bobák, M.—Hluchý, L.—Tran, V.: Tailored Platforms as a Cloud Service. Proceedings of 13th International Symposium on Intelligent Systems and Informatics (SISY 2015), Subotica, Serbia, September 2015.

[6] Carlini, E.—Coppola, M.—Dazzi, P.—Ricci, L.—Righetti, G.: Cloud Federations in Contrail. In: Alexander, M. et al. (Eds.): Euro-Par 2011: Parallel Processing Workshops. Springer Berlin Heidelberg, Lecture Notes in Computer Science, Vol. 7155, 2012, pp. 159–168.

[7] Celesti, A.—Tusa, F.—Villari, M.—Puliafito, A.: How to Enhance Cloud Architectures to Enable Cross-Federation. 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), Miami, USA, July 2010.

[8] Ferrer, A. J.—Hernández, F.—Tordsson, J.—Elmroth, E.—Ali-Eldin, A.—Zsigri, C.—Sirvent, R.—Guitart, J.—Badia, R. M.—Djemame, K.—Ziegler, W.—Dimitrakos, T.—Nair, S. K.—Kousiouris, G.—Konstanteli, K.—Varvarigou, T.—Hudzia, B.—Kipp, A.—Wesner, S.—Corrales, M.—Forgó, N.—Sharif, T.—Sheridan, C.: Optimis: A Holistic Approach to Cloud Service Provisioning. Future Generation Computer Systems, Vol. 28, 2012, No. 1, pp. 66–77.

[9] Ferry, N.—Chauvel, F.—Rossini, A.—Morin, B.—Solberg, A.: Managing Multi-Cloud Systems with CloudMF. Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies, Oslo, Norway, September 2013, pp. 38–45.

[10] 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.

[11] Gonçalves, G.—Endo, P.—Santos, M.—Sadok, D.—Kelner, J.—Melander, B.—Mangs, J.-E.: CloudML: An Integrated Language for Resource, Service and Request Description for d-Clouds. Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference, Athens, Greece, December 2011, pp. 399–406.

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

[13] Jeffery, K.—Houssos, N.—Jörg, B.—Asserson, A.: Research Information Management: The CERIF Approach. International Journal of Metadata, Semantics and Ontologies, Vol. 9, 2014, No. 1, pp. 5–14.

[14] Kritikos, K.—Domaschka, J.—Rossini, A.: SRL: A Scalability Rule Language for Multi-Cloud Environments. 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), Singapore, December 2014, pp. 1–9.

[15] Kurze, T.—Klems, M.—Bermbach, D.—Lenk, A.—Tai, S.—Kunze, M.: Cloud Federation. Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2011), Rome, Italy, September 2011.

[16] Bobák, M.—Hluchý, L.—Tran, V.: Methodology for Intercloud Multicriteria Optimization. Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2015), Zhangjiajie, China, August 2015.

[17] Nguyen, M. B.—Tran, V.—Hluchý, L.: A Generic Development and Deployment Framework for Cloud Computing and Distributed Applications. Computing and Informatics, Vol. 32, 2013, No. 3, pp. 461–485.

[18] Petcu, D.: Portability and Interoperability Between Clouds: Challenges and Case Study. Proceedings of the 4th European Conference on Towards a Service-Based Internet, Poznan, Poland, October 2011.

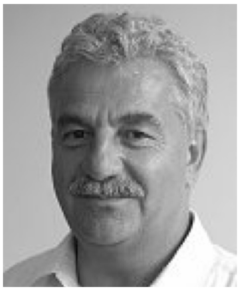
[19] Petcu, D.—Macariu, G.—Panica, S.—Crciun, C.: Portable Cloud Applications – From Theory to Practice. Future Generation Computer Systems, Vol. 29, 2013, No. 6, pp. 1417–1430.

[20] Quinton, C.—Romero, D.—Duchien, L.: Cardinality-Based Feature Models with Constraints: A Pragmatic Approach. Proceedings of the 17th International Software Product Line Conference, Tokyo, Japan, August 2013, pp. 162–166.

[21] Rochwerger, B.—Breitgand, D.—Levy, E.—Galis, A.—Nagin, K.—Llorente, I. M.—Montero, R.—Wolfsthal, Y.—Elmroth, E.—Cáceres, J.—Ben-Yehuda, M.—Emmerich, W.—Galán, F.: The Reservoir Model and Architecture for Open Federated Cloud Computing. IBM Journal of Research and Development, Vol. 53, 2009, No. 4, pp. 535–545.

[22] Toosi, A. N.—Calheiros, R. N.—Buyya, R.: Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey. ACM Computing Surveys (CSUR), Vol. 47, 2014, No. 1, Article No. 7.

Martin Bobák is a researcher and a Ph.D. candidate in applied informatics at the Institute of Informatics of the Slovak Academy of Sciences. He received his M.Sc. degree in computer science from the Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava (Slovakia) in 2013. His research interests are cloud computing, algorithms and data structures. He is (co-)author of several scientific papers and has participated in international and national research projects. He is a reviewer for international scientific conferences and journals, and a teaching assistant at the Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava (Slovakia).



Ladislav Hluchý (Associated Professor, M.Sc., Ph.D.) is Head of the Parallel and Distributed Information Processing Department, and former Director of the Institute of Informatics, Slovak Academy of Sciences (IISAS) for more than 20 years. He received his M.Sc. and Ph.D. degrees, both in computer science. He is R & D Project Manager, Work-Package Leader and Coordinator in a number of 4th, 5th, 6th, 7th and H2020 EU IST RTD projects as well as Slovak R & D projects (VEGA, APVV, SPVV). His research topics are focused on parallel and distributed computing, large scale applications, cluster/grid/cloud computing, service oriented computing and knowledge oriented technology. His highlighted research works are within EU IST RTD projects EGI-Engage H2020-654142 Engaging the Research Community towards an Open Science Commons, EGI-InSPIRE FP7-261323, EGEE III FP7-222667, EGEE II FP6 RI-031688, EGEE FP6 INFSO-RI-508833, REDIRNET FP7-607768, VENIS FP7-284984, SeCriCom FP7-218123, Commius FP7-213876, ADMIRE FP7-215024, DEGREE FP6-034619, INTAS FP6 06-1000024-9154, int.eu.grid FP6 RI-031857, K-Wf Grid FP6-511385, MEDIGRID FP6 GOCE-CT-2003-004044, CROSSGRID FP5 IST-2001-32243, PELLUCID FP5 IST-2001-34519, ANFAS FP5 IST-1999-11676, SEIHPC, SEPP and HPCTI as well as in international and Slovak national research projects. He is a member of IEEE, e-IRG, EGI Council, the Editor-in-Chief of the current contents (CC) journal Computing and Informatics (CAI). He is also (co-)author of scientific books and numerous scientific papers (more than 300), contributions and invited lectures at international scientific conferences and workshops. He is a supervisor and consultant for Ph.D. study at the Slovak University of Technology (STU) in Bratislava.

Viet Tran is a senior scientific researcher at the Institute of Informatics, Slovak Academy of Sciences (IISAS) with research focused on high-performance distributed computing and cloud computing. He received his M.Sc. degree in informatics and information technology and his Ph.D. degree in applied informatics from the Slovak University of Technology (STU) in Bratislava. He has participated in a number of EU RTD FP projects as well as international and Slovak national research projects as a work-package leader, key person and scientific coordinator. He is (co-)author of scientific books and scientific papers, member of program committees, reviewer for international scientific conferences, journals and projects.