Behaviour-inspired Data Management in the Cloud

Dariusz Król, Renata Słota, Włodzimierz Funika
Institute of Computer Science
AGH - University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
{dkrol, rena, funika}@agh.edu.pl

***Abstract* — Open source cloud computing solutions are still not mature enough to handle data-intensive applications, e.g., scientific simulations. Thus, it is crucial to propose appropriate algorithms and approaches to the data management problem in order to adjust cloud-based infrastructures to scientific community requirements. This paper presents an approach inspired by observation of the cloud user behaviour: the intensity of data access operations, their nature, etc. We also describe how the proposed approach influences the architecture of a typical cloud solution and how it can be implemented based on the Eucalyptus system which is a successful open source cloud solution.**

Keywords- cloud computing; data management; monitoring.

I. INTRODUCTION

Cloud computing is arguably the most popular buzzword in the tech world today. It promises to reduce the total cost of maintenance of an IT infrastructure with providing better scalability and reliability at the same time. Apart from "unlimited" computational power, the Cloud provides also "unlimited" storage capacity which can be accessed from every device which is connected to the Internet. Therefore, this is not a surprise that many commercial companies along with academic facilities are very interested in this paradigm. As with other paradigms, various research centers test it in a variety of ways in order to unveil its strengths and weaknesses. What makes the Cloud computing special in comparison to other, more academic-related approaches to distributed computing, e.g., Grid computing, is the support and investment made by the largest IT companies, e.g., Google, IBM, Microsoft and many others. These million dollars investments can be treated as a good omen that Cloud computing can be widely adopted and will not disappear after few years.

Today, cloud computing solutions, especially the open source ones, are not mature enough in terms of storage capabilities to handle data-intensive applications which would like to store results in the cloud-based storage. One of the issue is lack of adaptation of data management strategy, e.g., to changing user requirements or location from where the user access the cloud storage. Each of these aspects can imply the access time to data especially when considering geographically distributed resources which constitute a single cloud installation. Therefore, we propose a novel approach, based on autonomic systems (similar to situation-aware systems - [1]) and behaviour observation whose main goal is to adapt data location to the user needs which will result in decreasing access time and higher utilization factor of resources. We introduce a "Usage profile" concept which describes a piece of data stored in the cloud storage. The usage profile contains information how the described data is used by cloud clients. To create such a profile, operations related to data storage performed by cloud users are monitored and analyzed. The approach is designed to be an additional element of the cloud installation rather than being mandatory, which is invisible from the user point of view but can positively influence the storage performance.

The commercial clouds, e.g., Amazon Elastic Compute Cloud (EC2) [2] which it is an Infrastructure-as-a-Service system, which means it allows to manage a computational environment consisted of virtual machines, cannot be easily studied due to closed source code, thus in this paper they will not be taken into consideration. The rest of the paper is organized as follows:

- in Section II, a number of the existing cloud solutions are presented,
- Section III describes the data management algorithm which is based on the behaviour analysis,
- parameters of the usage profile along with behaviour which is analyzed in order to create the usage profiles are described in Section IV,
- a prototype implementation of the algorithm is presented in Section V,
- conclusions along with the future directions of the research are provided in Section VI.

II. EXISTING OPEN-SOURCE CLOUD ENVIRONMENT

Cloud computing has been already widely adopted by various commercial companies and academic centers. While many commercial companies develop their own solutions, e.g., Amazon EC2, Microsoft Azure [3] or Google AppEngine [4], others use and invest in open source solutions which are especially well suited for situations where the environment has to be adapted to some specific requirements. This feature is very important for scientific community which would like to implement new concepts and approaches to optimize access time or other parameters. Thus in the presented research only open source solutions were taken into consideration. In this section, we describe three well known "Infrastructure as a Service" environments: Eucalyptus, Nimbus and OpenNebula.

 ISBN: 978-1-61208-106-9

A. *Eucalyptus*

Eucalyptus system [5] is an example of an open source project which became very popular outside the scientific community and is exploited by many commercial companies to create their own private clouds. It was started as a research project in the Computer Science Department at the University of California, Santa Barbara in 2007 and today is often treated as a model solution for providing infrastructure as a service. Eucalyptus aims at providing an open source counterpart of the Amazon EC2 cloud in terms of interface and available functionality.

Each Eucalyptus installation consists of a few loosely coupled components each of which can run on a separate physical machine to increase scalability. The frontend of such a cloud is "Cloud controller" element which is an access point to the virtual machines related features. While "Cloud controller" is responsible for computation, the "Walrus" component is responsible for data storage. It allows storing virtual machine images along with any other files which are divided into a hierarchy of *buckets* and can be treated as the Amazon Simple Storage Service (S3) counterpart in the Eucalyptus system. Amazon S3 is a Cloud storage service which allows storing any type of data in form of files in a number of buckets (each with a unique name within a bucket) using a simple API, i.e., *put, get, list, del* operations are supported. Each virtual machine is run on a physical host which is controlled by the "Node controller" element. A group of nodes can be gathered into a cluster which exposes a single access point, namely "Cluster controller" from the virtual machine management side and "Storage controller" from the virtual machine images repository side.

Eucalyptus is based on the Java technology stack and its source code is freely accessible and can be modified as necessary. To mention a few, the current implementation uses web services (Apache Axis [6]) to expose the provided functionality to the external clients, and exposes a web-based user interface developed with the Google Web Toolkit (GWT) [7]. It also supports the Xen [8] and KVM [9] hypervisors to run virtual machines on the supervised resources.

The open version of the Eucalyptus system stores data into a single directory on the host on which the Walrus component is installed. Therefore, the only way to distribute the data is to exploit a distributed file system, e.g., Lustre [10], which will be mounted to the directory used by the Eucalyptus installation. This can be unfortunately not enough especially when the application is running.

B. *Nimbus*

Nimbus [11] is a toolkit for turning a cluster into an IaaS cloud computing solution. It is developed by the Globus Alliance [12]. A Nimbus client can lease remote resources by deploying virtual machines on these resources and configure them to fulfil the user requirements. What makes it attractive is the support of a communication interface known from the Grid computing, namely Web Services Resource Framework [13]. As in other popular solutions, Nimbus provides an Amazon EC2 compatible interface for cloud clients, which is de facto a standard of IaaS environment due to its wide adoption in a number of solutions.

A Nimbus installation consists of a number of loosely coupled elements. The center point of the Nimbus architecture is the "Workspace service" component which is a coordinator of the whole installation. It is invoked through different remote protocol frontends, e.g., WSRF or EC2 – compatible services. Another important component is "Workspace resource manager" which runs on each host within the Cloud and is responsible for controlling a hypervisor on the host machine. The current version fully supports Xen hypervisor and most of the operations on the KVM hypervisor. It is also worth of mentioning that Nimbus installation can be easily connected to a public commercial cloud, e.g., Amazon EC2 in order to achieve even greater computer power when the in-house infrastructure is not enough.

In terms of data management, the Nimbus project is limited to the virtual machine image repository. There is no component which would provide functionality similar to the Amazon S3. The user can only upload virtual machine images to the Nimbus cloud and store data stemming from computation on storage devices connected directly to a virtual machine.

The Nimbus project is based on open source tools and frameworks, e.g., Apache Axis, the Spring framework [14] or JavaDB [15]. Therefore, everyone can download its sources from a public repository and modify its functionality as desired.

C. *OpenNebula*

OpenNebula [16] is a Virtual Infrastructure Manager for building cloud infrastructures based on Xen, KVM and VMWare virtualization platforms [17]. It was designed and developed as part of the EU project RESERVOIR [18], whose main goal is to provide open source technologies to enable deployment and management of complete IT services across different administrative domains. OpenNebula aims to overcome shortcomings of existing virtual infrastructure solutions, e.g., inability to scale to external clouds, a limited choice of interfaces with the existing storage and network management solutions, few preconfigured placement policies or lack of support for scheduling, deploying and configuring groups of virtual machines (apart from the VMWare vApp solution [19]). Like other of the presented solutions, OpenNebula is fully open source and its source code can freely be checkout from a public repository.

OpenNebula architecture was designed with modularity feature in mind. Therefore, it can be extended to seamlessly support a new virtualization platform e.g., in terms of virtual image or service managers. For instance, a procedure of setting up a VM disk image consists of well-defined hooks whose implementation can be easily replaced to interface with the third-party software. To manage an OpenNebula installation, the user can use a simple, dedicated command line interface or Amazon EC2 query interface. Therefore, it can be accessed with the tools originally developed to work with the Amazon EC2 cloud.

 ISBN: 978-1-61208-106-9

In terms of storage mechanisms, it is limited to repository of VM images only. The repository can be shared between available nodes with the Network File System (NFS) [20]. It is also possible to take advantage of block devices, e.g., LVM to create snapshots of images in order to decrease time needed to run a new instance of image.

III. BEHAVIOUR-INSPIRED APPROACH TO DATA MANAGEMENT

The most important aspect of the presented approach is its orientation towards each user requirements rather than some global optimization such as equal data distribution among available resources. Our approach treats each user separately by monitoring his/her behaviour related to data storage. The monitoring process is necessary in order to discover the nature of the data automatically, e.g., whether it is read-only or often modified data. With this knowledge, the data can be managed appropriately, i.e., with requirements such as high availability taken into account. Another important feature of the approach which can be deducted from the previous one is its transparency from the user point of view. Thus, it can be applied to any existing solution without any modification required to the user-side code.

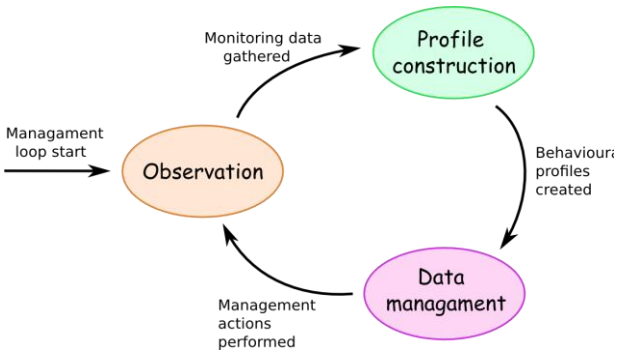


Figure 1: Data management loop.

The structure of the described algorithm is depicted in Figure 1. There are 3 phases included:

- *The observation phase* where the information about the user behaviours are aggregated. It is a start point of the management iteration. Each operation related to the storage, e.g., uploading or downloading files are recorded along with information about the user who performed the operation and a time-stamp.
- *The profile construction phase* is the one where the gathered information about behaviour is analyzed. For each user, a profile which describes how the user accesses each piece of data is created, thus the profile also contains information how each piece of data should be treated.
- *The data management phase* is responsible for modifying the data storage, i.e., applying a dedicated strategy which corresponds to a user profile. Such a strategy can e.g., create many replicas of a piece of data which is read by many users but hardly any one modifies it or it can move the data closer to the user to decrease its access time.

An important feature of the algorithm is the fact that it never ends. There is no stop condition because such a management process may last as long as the cloud is running. Each iteration of the loop results in tuning the storage strategy to the observed user behaviour. However, the historical data is taken into account as well and can influence the storage strategy rather than just be omitted. In fact, its importance to the new strategy is one of the parameters of the algorithm.

Another important aspect of the approach is its influence on the architecture of a cloud solution. The overview of architecture is schematically depicted in Figure 2. To underline the most important components, some simplifications were introduced, e.g., the cloud solution is represented only by "Cloud manager" which is an access point to the cloud infrastructure. "Storage elements" represent physical resources where the data is actually stored. New components are as follows:

- *Monitoring system* is responsible for gathering information about user actions. The most important operations are those related to data storage, e.g., uploading a file or accessing a file by the user. Information about these actions have to be remotely accessible by an external cloud client in a programming language independent way.
- *Behaviour data manager* is the main element of this new approach. It performs the analysis of the user behaviours and creates their profiles. Then, it performs all the necessary actions to adjust the storage strategy to the actual profile. In most cases, these actions will be related to either moving data between storage elements with different physical parameters or managing data replication, e.g., creating new replicas. By combining these two types of operations, we can improve the Quality of Service (QoS) of the cloud storage, e.g., decrease the data access time. It is also possible to apply more sophisticated algorithms for data management as the ones described in [21] and [22]. The communication between “Behaviour data manager” and “Storage elements” is optional. If “Cloud manager” exposes an interface to manage the actual data location, there is no need for “Behaviour data manager” to interact with “Storage elements” directly.
- *Profile knowledge base* is a repository where the historical profile for each piece of data is stored along with record of each performed action. Thus, it can be used by the "Behaviour data manager" to take into account not only the most recent information but also the previous actions.

As we can see the approach can be easily integrated with any cloud solution which can be monitored, i.e., each performed operation related to the storage is registered, and the stored data can be moved between available physical resources, either indirectly with an exposed programming

interface or directly with accessing storage elements and moving raw data. These requirements are rather easy to meet and in the next section we are presenting an example implementation based on a popular open-source cloud solution. It is also worth mentioning that the original source code of such a cloud solution stays untouched.

IV. USAGE PROFILE

The presented approach highly exploits usage profiles which are based on the observation of the user actions. From the performed actions, only those related to the data storage (e.g., uploading or downloading files) are important. In the system a set of profile types is defined along with the actions which are recommended for each of these types. A profile type and an actual profile (which will be described later in this section) correspond to a piece of data, e.g., a single file. Thus, it can be treated as a metadata for the actual data stored in a cloud storage.

Such a profile type contains information which describes how the described data is used. So far a few parameters were defined:

- Category of access frequency which describes whether the object is frequently accessed
- Percentage of the queries relating to read/write operations
- Category number of clients for read/write operations
- Category connection points (e.g., IP addresses) for read/write operations

Based on these parameters a set of a predefined profile types can be created. They should correspond to the well known storage management patterns, e.g., if an object is always read, it should be replicated to multiple physical locations in order to lower the access time.

On the other side, we observed user actions which can be grouped by an object they are related to. As a result, we have got a behaviour profile which contains the following information:

- Frequency of access to an object, e.g., per hour or per iteration of the algorithm loop.
- Number of read/write operations
- Number of different users accessing to the object (separately for read/write operations)
- Number of different places (e.g., IP addresses) from which the object was accessed (separately for read/write operations)

Such a profile is created for each piece of data in the cloud storage (e.g., file) after the first iteration of the algorithm and updated on each subsequent iteration. In each "Data management" phase, for each profile a similarity function to each defined profile type is calculated. Then, the actions related to the most similar profile type are performed.

Therefore, the approach can be easily extended in terms of recognized behaviour, simply by defining new profile types along with related actions.

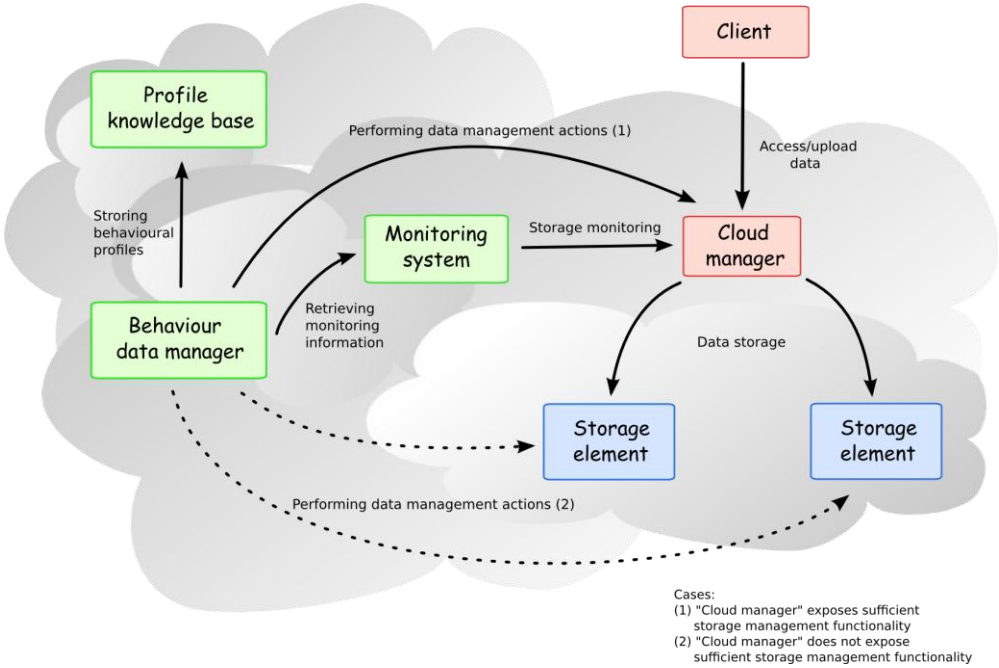


Figure 2: Architecture overview of a cloud solution with "Behaviour data manager" applied.

V. IMPLEMENTATION NOTES

To present a sample implementation of the approach, we chose the Eucalyptus system as a basis. This choice was motivated by a large popularity of Eucalyptus and its

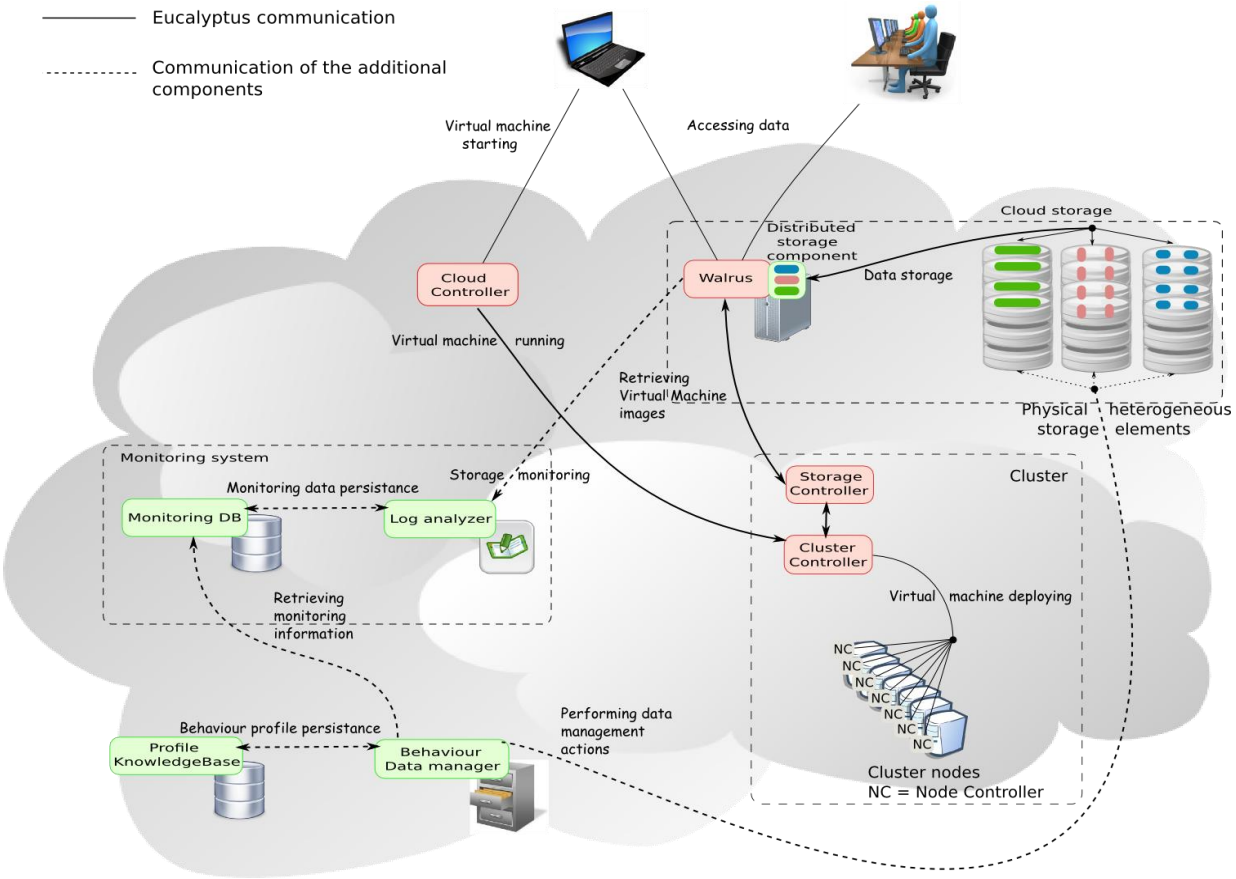
functionality in terms of storage which is very similar to the Amazon S3 offer, a de facto standard in the cloud industry.

The Eucalyptus architecture contains a component called "Walrus" which is responsible for the -storage-related functionality. "Walrus" exposes an API (Application Programming Interface) which consists of methods which create, update, and remove objects and buckets from the cloud storage.

The open version of the Eucalyptus implements the cloud storage as a designated directory on the local filesystem. It is rather a minimalistic solution of the cloud storage, due to very limited ways of data distribution. The possible way is to mount a distributed file system at the designated directory which will transparently distribute data among a number of storage elements. Unfortunately such a solution does not allow to control data manipulation which cannot be accepted in our situation. Therefore, we extended the storage system in Eucalyptus with an ability to store data in several directories instead of one only, each of which can point to a different physical location, e.g., via Network File System (NFS). With this extension, the location of the data can be easily controlled, simply by moving files between directories.

exposes an API to external users for storing data in the cloud. Apart from storing custom data, e.g., results coming from a running simulation, "Walrus" stores two other types of objects. The first one is a VM image which is uploaded by the user and then is run on the Eucalyptus infrastructure. Although, the user communicates with other component, called "Cloud controller", the images are actually stored with "Walrus". The second type of objects is the "Block storage" object. It is used as a mountable partition to store data during a VM run, similarly to a local file system. Moreover, a "Block storage" object can store data between two subsequent runs of a VM and in opposite to a VM virtual disk, the data is not erased after a VM shutdown. Such a partition is stored within the cloud storage with "Walrus". All these three types of objects are stored with "Walrus" on the same rules and thus can be uniformly managed with our system.

The first of the additional elements added to the architecture is a dedicated “Monitoring system”. It consists of “Log analyzer” which periodically reads the Walrus log where each operation related to the storage is recorded and a relational database where information who and which



As mentioned above there are a few components to add to the Eucalyptus architecture in order to implement the approach under discussion. Such an extended architecture is depicted in Figure 3. There is the "Walrus" component which

Figure 3: Eucalyptus system architecture extended with “Behaviour Data Manager”, “Profile KnowledgeBase”, “Monitoring system”, and “Distributed storage component”.

operation performed. Thus all the necessary data for profile creation is prepared in a technology neutral form. The second element is "Behaviour Data manager" which is responsible for analysing data written to “Monitoring Database (DB)”. The behaviour of each user related to each stored object is categorized to one of the defined groups, which describes how the object should be treated, e.g., if it is written mostly often by a single user or if it is read by multiple users who are geographically distributed but which is not modified often. This information in form of a “Usage profile” is then stored in "Profile KnowledgeBase" along with its update timestamp. After the categorization phase is over, "Behaviour Data manager" performs actions which are suitable for an assigned usage profile. An action can be as simple as moving an object between directories on the Walrus machine, each of which represents a different type of storage elements, or as complicated as creating many replicas and moving them as close to users as possible. The historical data is taken into account with a weight set in the system configuration.

The whole algorithm is performed periodically in order to adapt the storage strategy to the dynamic environment. The presented implementation is currently in the prototype phase. The monitoring system is finished and tested but the implementation of "Data manager" is still in progress.

VI. CONCLUSIONS AND FUTURE WORK

Although, there are several open-source cloud solutions available today, none of them provides a storage system which would be able to adapt to the user needs automatically. Instead, only basic functionality, e.g., storing VM images or custom objects is supported. The presented approach aims at providing a sophisticated data management which would be flexible enough to be applicable to different cloud solutions and which would manage data according to the user behaviour. The article describes the main assumptions of the approach along with phases of the management process. As an example of its implementation a prototype version of the system based on Eucalyptus is described. Due to the limited functionality of the Eucalyptus system, an extension which provides a real distributed data storage to multiple locations has been implemented.

Since the implementation of the approach is in progress, there is work to be done. The "Data manager" and "Profile database" components are not yet finished. From the conceptual point of view, the approach lacks a well-defined set of behaviour categories along with related actions. However, these categories will be crystallized during real-life tests when the behaviour of the real users will be observed and analyzed. Until then, some preliminary categories will be be defined.

ACKNOWLEDGMENTS

The authors are grateful to prof. Jacek Kitowski for valuable discussions. D. Król thanks to UDA-POKL.04.01.01.01-00-367/08 project for support, R. Słota and W. Funika to the projects A-0938-RT-GC EDA EUSAS Project and POIG.02.03.00-00-007/08-00 “PL-Grid”.

REFERENCES

[1] Han, J.H., Lee, D.H., Kim, H., In, H. P., Chae, H.S., and Eom Y.I., “A situation-aware cross-platform architecture for ubiquitous game”, Computing and Informatics, vol. 28, nr 5, 2009, pp. 619-633.

[2] Amazon Elastic Compute Cloud (Amazon EC2). Amazon Inc., 2008. [on-line: http://aws.amazon.com/ec2, as of June 13, 2010]

[3] Microsoft Windows Azure Platform (Windows Azure). Microsoft, 2010. [on-line: http://www.microsoft.com/windowsazure/, as of June 13, 2010]

[4] Google AppEngine. Google Inc., 2008. [on-line: http://code.google.com/intl/pl-PL/appengine/, as of June 13, 2010]

[5] Eucalyptus Systems Inc.: 2010, [on-line: http://www.eucalyptus.com/, as of July 24, 2010]

[6] Apache Axis website. [on-line: http://ws.apache.org/axis/, as of June 13, 2010]

[7] Google Web Toolkit website. [on-line: http://code.google.com/intl/pl-PL/webtoolkit/, as of June 13, 2010]

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, and A. Warfield, "Xen and the art of virtualization," in SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, NY, USA: ACM, 2003, pp. 164-177, [on-line: http://dx.doi.org/10.1145/945445.945462 as of June 13, 2010]

[9] Kernel-based Virtual Machine project wiki. [on-line: http://www.linux-kvm.org/page/Main_Page, as of June 13, 2010]

[10] Lustre filesystem wiki. [on-line: http://wiki.lustre.org/index.php/Main_Page, as of June 13, 2010]

[11] Kielmann, T., “Cloud computing with Nimbus”, March 2009, EGEE User Forum/OGF25 & OGF Europe's 2nd International Event.

[12] Globus Alliance website. [on-line: http://www.globus.org/, as of June 13, 2010]

[13] Foster, I., Frey, J., Graham, S., Tuecke, S., Czajkowski, K., and Weerawarana, S., “Modeling Stateful Resources with Web Services”, 2004. [on-line: http://www.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf, as of June 13, 2010]

[14] Spring Framework website. [on-line: http://www.springsource.org/, as of June 13, 2010]

[15] Java database website. [on-line: http://developers.sun.com/javadb/, as of June 13, 2010]

[16] B. Sotomayor, R. S. Montero, I. M. Llorente, and I. Foster, "Capacity Leasing in Cloud Systems using the OpenNebula Engine." Cloud Computing and Applications 2008 (CCA08), 2009.

[17] VMware website. [on-line: http://www.vmware.com, as of June 13, 2010]

[18] RESERVOIR project website. [on-line: http://62.149.240.97/, as of June 13, 2010]

[19] VMware Virtual Appliances website, [on-line: http://www.vmware.com/appliances/getting-started/learn/, as of June 13, 2010]

[20] Network File System version 4 protocol specification, [on-line: http://tools.ietf.org/html/rfc3530, as of June 13, 2010]

[21] Slota, R., Nikolow, D., Kuta, M., Kapanowski, M., Skalkowski, K., and Kitowski, J., “Replica Management for National Data Storage”, Proceedings PPAM09, LNCS6068, Springer, 2010, in print.

[22] Slota, R., Nikolow D., Polak, S., Kuta, M., Kapanowski, M. , and Kitowski, J., "Prediction and Load Balancing System for Distributed Storage", Scalable Computing Practice and Experience, 2010, in print.

 ISBN: 978-1-61208-106-9