SHORT ANALYSIS OF IMPLEMENTATION AND RESOURCE UTILIZATION FOR THE OPENSTACK CLOUD COMPUTING PLATFORM

Daniel Grzonka*, Michał Szczygieł, Artur Bernasiewicz,
Andrzej Wilczyński and Marek Liszka
Institute of Computer Science
Faculty of Physics, Mathematics and Computer Science
Cracow University of Technology
Warszawska st 24, 31-155 Cracow, Poland
E-mail: grzonka.daniel@gmail.com
*Corresponding author

KEYWORDS
Cloud Computing, OpenStack, Virtualization, High Performance Computing, Parallel Environments, Resource Utilization Analysis.

ABSTRACT

The problem of efficient use of computer resources is the actual challenge for many years. Huge progress in the hardware development has left behind the development of software techniques. One of the most popular solutions for this problem is the idea of virtualization, which a natural continuation is cloud computing. Cloud computing is an innovative concept, where the resources are virtualized, dynamically extended and provided as highly personalized services. In this paper, we present a short analysis of open-source cloud technology - OpenStack. We described OpenStack architecture, requirements, setup process, and related problems. We also conducted a thorough analysis of resource utility - both at full load and without. In our experiments we have analyzed the performance depending on the allocation of virtual resources. Through our work we pointed out aspects which deserve attention by choosing an OpenStack platform. Additionally, we draw attention to the burden on the use of technologies such as OpenStack cloud.

INTRODUCTION

Over the decades, the rapid development of broadly defined computer technologies is observed. In the 60s of the twentieth century, when the first integrated circuit (IC) has been developed the revolution in the computer hardware started. The advancement of microelectronic technology resulted in increase of number of transistors placed on microprocessors. Regularity in the development of digital technology was discovered in 1965 by co-founder of Intel Corporation - Gordon Moore. In 1975, Moore formulated statement, named the Moore's law, concerned the rate of density doubling. He observed that "circuit density-doubling would occur over 24 month" [1]. The statement has been generalized for many IT areas like size of RAM and HDD or bandwidth of computer networks. The most popular example is the increase of the number of transistors on integrated circuits.

With the increase in available resources the problem of efficiency occurred. In extreme cases, an increase in the available resources can have opposite effects to what was expected. The reason for this may be overhead associated with more complex communication or misapplied parallelization technologies [2]. Therefore, despite the fact that the development of computer hardware and programming languages has enabled the creation of a much more complex systems, the development of software techniques doesn't keep up with the technological development. This phenomenon is called "software crisis", and is continued to this day. E. Dijkstra summed up this situation as follows [3]:

"The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem."

The main cause of this state is that increase in computing power had outperformed the ability of programmers to effectively utilize those capabilities. Despite the development of many methods and methodologies supporting the use of available resources (like new paradigms, programming approaches or hardware solutions), the problem is still actual.

One of the proposed solutions is virtualization technology. Virtualization is wide issue refers to abstraction in many computer areas (like computer hardware, storage devices, operating systems, computer networks, operating memory). It allows effective utilization of available physical resources by any adaptations of the virtual features to the user's needs. Computing power of modern computers is so huge, that one physical server can held a few operating systems adapted for many purposes. The main advantage of

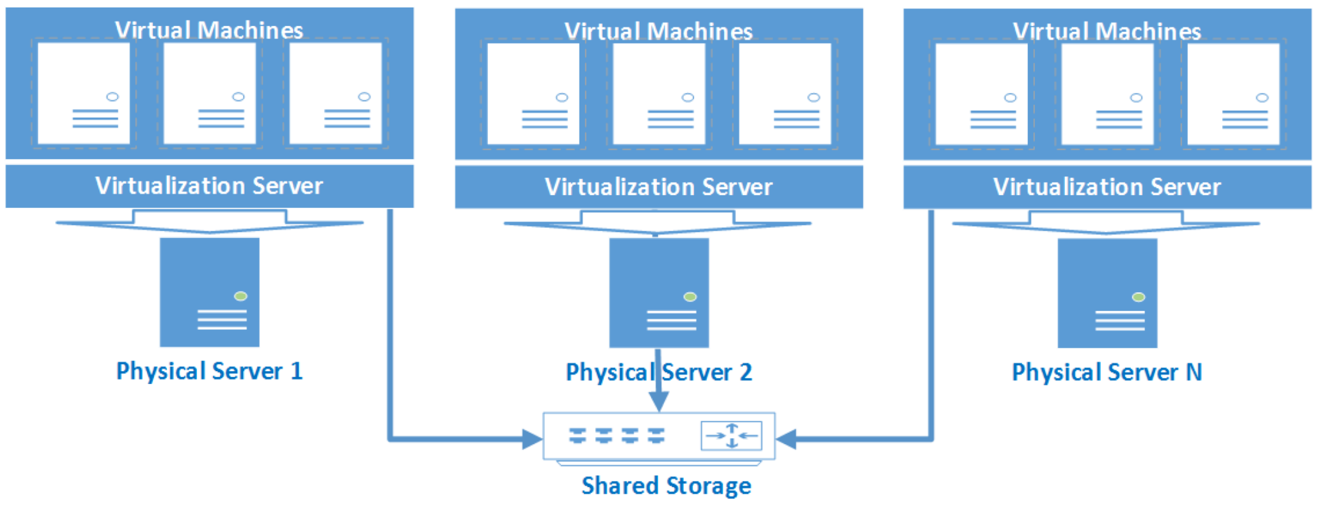

Fig. 1. Model of full virtualization.

using virtualization is better utilization of resources which implies significant savings resulting from the purchase of equipment, power consumption, heat generation and space requirements [4, 5].

Additionally with virtualization, we gain much more flexibility and efficiency in resource configuration. Without interfering with the physical equipment, we are able to change the parameters of our servers. This increases both the availability and reliability of systems. In the case of breakdown it allows for rapid reconstruction of infrastructure. Virtualization is also solution for a problem with application of compatibility across delivery required environments (operating systems). Actually, there is variety of applications generating virtual machines, e.g. VMware, KVM, Xen, Microsoft Hyper-V, VirtualBox, OpenVZ [4, 6].

As a result of the development and growing popularity of virtualization techniques the new idea called cloud computing was established. It is natural evolution of traditional grids, clusters and data centres [7]. The National Institute of Standards and Technology defined cloud computing as "*a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*" [8].

The model is based on the idea of resource virtualization which allows to offer highly personalized service. The idea is to move the obligation to provide IT services (access to data, software or computing power) to servers and enabling continued access by client devices. Fee is charged from the customer for the resources used (pay per use) like computational power, software, storage, data transfer etc [9]. Cloud computing providers offer their services in a transparent way according to several fundamental models and the three main are listed below [10]:

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS),
- Software as a Service (SaaS).

One of the innovative and more and more popular software platform providing services in the IaaS model is the OpenStack platform that is the subject of our article.

Experiments described in the article were performed under the Development Workshop classes by the Cracow University of Technology students supervised by Daniel Grzonka. The aim of the course was to familiarize students with the implementation and the basic issues related to performance in distributed environments such as cloud computing.

The paper is organized as follow. In Section 2 we described features and architecture of OpenStack platform. We present all the modules available in the latest version. The hardware requirements, installation tools and related problems are described in Section 3. Finally, in Section 4 we present results of our experiments. The paper is summarized and concluded in Section 5.

LEAD-IN OPENSTACK

OpenStack is a free and open-source platform that possesses a set of tools for the creation and management of private, public and hybrid cloud computing. Services provided by the OpenStack software control a wide range of compute, storage, and networking resources. OpenStack provides an architecture that gives the flexibility in the clouds design. It can be integrated with existing systems and third-party technologies (e.g. Amazon EC2) [11].

OpenStack has been deployed by global enterprise customers whose process and stores data are measured in petabytes. Data are not stored in a traditional file

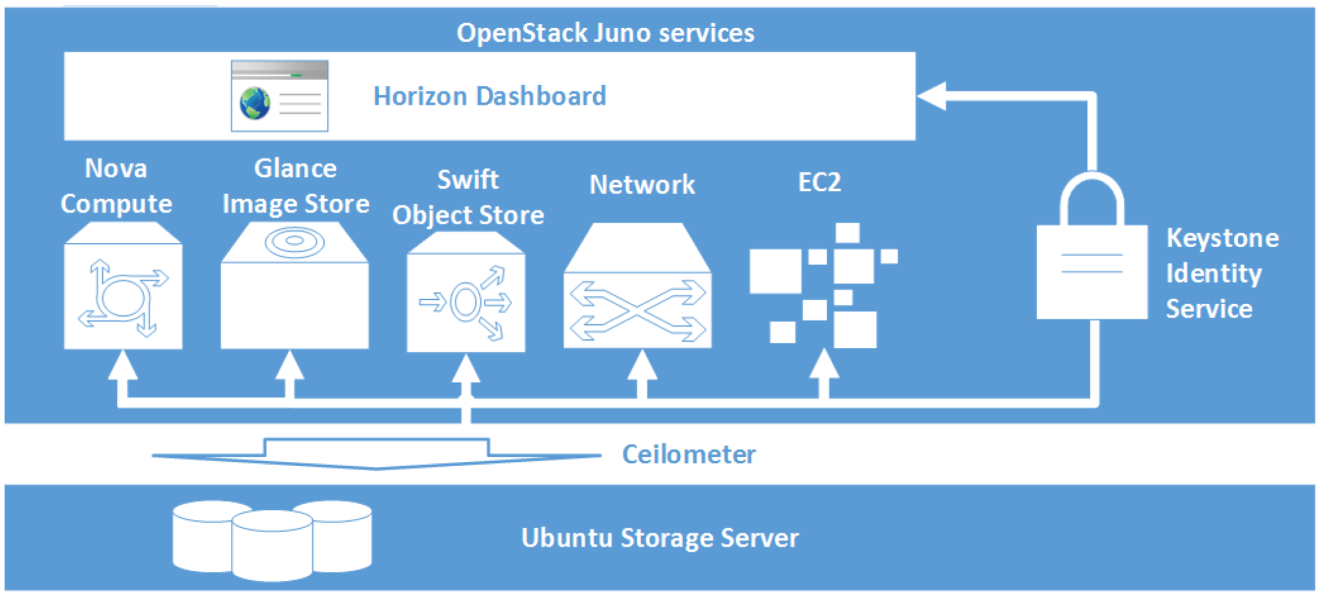

Fig. 2. The OpenStack Juno modular architecture (selected services).

system, but in a distributed storage static data systems such as images of virtual machines, files, backups and archives. In addition, OpenStack provides greater scalability, redundancy and durability. Objects and files are stored on multiple disks scattered in the data centre, providing replication and data integrity. [12, 13]

The platform provides also flexible network models for various applications or groups of users. The ability to manage network addressing allows dedicating static and dynamic IP addresses. It also allows the users to dynamically redirect traffic to all computing resources, which is very useful during maintenance or in case of failure. Users can create their own networks, control traffic and connect machines to one or more virtual networks. The project provides mechanisms for associating instances with external IP addresses and allows for remote access and control.

OpenStack is developed by a non-profit corporate entity - OpenStack Foundation, established by Rackspace and NASA in September 2012. An important advantage is the support and participation in the design of many major companies and IT organizations. At the moment, the foundation has over 18,000 members from over 150 countries around the world. The purpose of the foundation is to promote the OpenStack software and its community, which develop project. The platform is open source and can be modified and adapted as needed [11, 14].

OpenStack can be treated as an IaaS (Infrastructure as a Service) model. The IaaS provider offers resources and enables users to create own virtual infrastructure. The software is built modular and consists of many components working together that have open APIs so it is possible to manage resources from a single web interface - dashboard, or by custom solutions developed by programmers. Services communicate with each other using the API after the authentication. In our research we use the newest OpenStack version - Juno - which may (but does not need) consist of the following modules [11]:

- Nova - basic OpenStack engine; it is the controller for managing and implementing a large number of virtual machines and other items,
- Swift - storage system for objects and files; creates individual identifiers relating to the files and then decides where to store data,
- Cinder - storage component executing direct access to the data on disk,
- Neutron – set of communication functions; enables communication for individual components of OpenStack by networking,
- Glance - disk imaging services and a repository of OpenStack,
- Ceilometer - provides tariffing and reporting services for individual users clouds,
- Heat - defines the configuration files with the requirements of specific applications clouds and specifies the resources necessary for the application,
- Horizon - graphical user interface for system administrators enabling easy way for managing OpenStack,
- Keystone - OpenStack Identity service, with a list of users and permissions; allows for authorization and authentication for individual cloud services,
- Trove - managing relational database services,
- Sahara – Big Data processing service for OpenStack; provide users with simple means to data processing (Hadoop, Spark) in MapReduce model.

Fig. 2 presents OpenStack Juno modular architecture (only enabled services from our test environment).

HARDWARE REQUIREMENTS AND INSTALLATION

In our research we based on the newest OpenStack 2014.2 release (Juno). Our first machine was Pinokio server (specification in Tab. 1); according to the documentation [11] it meets the minimum hardware requirements needed for installation.

Table 1: Specification of Pinokio server.

Processor	Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz
Cores	4
Threads	4
Architecture	x86_64
Virtualization	Intel VT-x
Memory	8GiB DIMM DDR2 Synchronous 800 MHz (1,2 ns)
Disk	160GB SAMSUNG HD160JJ
Operating System	Ubuntu Server 14.10 (x64)
Swap Volume	4GiB

There are many solutions to the OpenStack platform installation, including [11, 12, 15]:

1. directly from GitHub repository,
2. Ubuntu Juju (simple client/server application that bootstraps an instance to create and control environment),
3. Vagrant OpenStack Provider (popular tool to manage virtual machines and set up development environments),
4. DevStack (set of tools used for the installation of the central OpenStack services from source repository).

We have tested three first options. The first attempt of installation we made from GitHub repository. The installation process was conducted without any problems. After the server restarts, there was an unsolvable problem with Python libraries dependencies. This problem made impossible starting the Nova service. Trying to re-install the platform ended with an error: *ImportError: cannot import name cfgfilter*. After uninstalling and reinstalling the problem appeared again.

Our second attempt was to install OpenStack using Ubuntu Juju. Juju require installation MaaS (Metal as a Service – model which role is to deploying services fast, reliable, repeatable and scalable) server – this step was performed without any problems. Next, we tried to install OpenStack platform, but during Juju bootstrap configuration the process hanged. Further installation became impossible.

The last attempt we used was Vagrant OpenStack Provider. Description of the installation steps is presented in [15], and is limited to: VirtualBox and Vagrant installation, clone OpenStackCookbook git repository and *vagrant up* command execution. OpenStackCookbook repository is extended with additional scripts to configure the environment. This approach was successful. Unfortunately, the minimum requirements does not allow to create a stable and smoothly running environment. A virtual machines absorb such a large resources (mainly RAM) making impossible to work on the platform. So we decided to buy new machine (specification in Tab. 2), calling it Pinokio-v2. Specifications of the new server allowed performing simple tests.

Table 2: Specification of Pinokio-v2 server.

Processor	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz
Cores	4
Threads	8
Architecture	x86_64
Virtualization	Intel VT-x
Memory	16GiB SODIMM DDR3 Synchronous 1600 MHz (0.6 ns)
Disk	256GB ADATA SP600
Operating System	Ubuntu Server 14.10 (x64)
Swap Volume	15GiB

The default configuration of OpenStackCoockbook creates 7 virtual machines (one for each: controller, network and cinder services; two for each: compute and swift services), which absorb about 14,5 GB RAM. Taking into account other system services, remained around 3% of free memory. Such configuration made much slower the whole OpenStack platform.

EXPERIMENTAL EVALUATION

In this section we present the results of resource utilization analysis. First the impact of the existence of running instances was examined. The CPU and RAM usage parameters were measured. For that purpose we prepared an exemplary instance of the system. OpenStack provide two images by default:

- CirrOS (9.3 MB) - a minimal Linux distribution that was designed for use as a test image on clouds,
- Trusty Tahr (244.4 MB) – distribution of Ubuntu 14.04,

and 5 flavours (sets of the virtual resources for instance) presented in Tab. 3.

Table 3: Available sets of the virtual resources.

Flavour	VCPUs	Disk (in GB)	RAM (in MB)
m1.tiny	1	1	512
m1.small	1	20	2048
m1.medium	2	40	4096
m1.large	4	80	8192
m1.xlarge	8	160	16384

With the available physical resources, single project OpenStack default allows to create up to 10 instances and assign the following virtual resources: 20 VCPUs, 50GB of RAM, 50 IPs, 10 security groups, 10 volumes and 1000GB of storage.

At the beginning, we have created project and simple network topology that would connect our instances. Then the image of operating system was chosen. Because of our limited hardware resources, we have assigned only m1.tiny flavour for each instance. To observe an increase in the use of resources caused by running subsequent instances we have been deploying the 3 instances simultaneously every time. The results of our observations where measured by *Process viewer and system resource monitor for GNOME* application. Tab. 4 presents the process of resources consuming. Each measurement was performed after 5 minutes idle time.

Table 4: Resources consumption caused by running instances.

Instances	CPU usage	RAM usage
0	5%	97,1%
3	12%	97,1%
6	18%	97,1%
9	23%	97,1%

As we can see the operating memory usage is constant, but CPU usage is growing. Each instance of CirrOS irreclaimable get and use about 2% of computing power (about 5-7% per 3 running instances).

The second task of the experiment was to measure resources usage by performing instances calculations. In order to increase the CPU load each instance of calculations was performed. For this purpose we have implemented one of the most popular benchmark - matrix multiplication.

Firstly we have prepared an instance of CirrOS based on m1.tiny flavour, and implemented matrix multiplication algorithm with matrices of 1024x1024 size. Then we have created a snapshot of the instance, which was named Archimedes. It allows us to create the same instances with our application. The results of our measurement are presented in Tab. 5.

Table 5: Resources consumption during calculations.

Instances	CPU usage	RAM usage
0	5%	97,1%
1	19%	97,2%
2	32%	97,3%
3	41%	97,3%
4	51%	97,4%
5	55%	97,5%
6	56%	97,5%

It should be noted, that instances number 1, 3 and 5 were assigned to nova-compute-1 node; the rest to nova-compute-2. As we can see, we have not reached full use of available resources. This is due to the amount of virtual computing machines. According to OpenStack documentation, the hardware resources we possess do not allow creating four virtual compute machines.

In Fig. 3 the CPU usage for each thread in scenario with four running instances on two VMs are presented. As we can see, some of threads (but not more than four in the same time) are using about 100% of computing power.

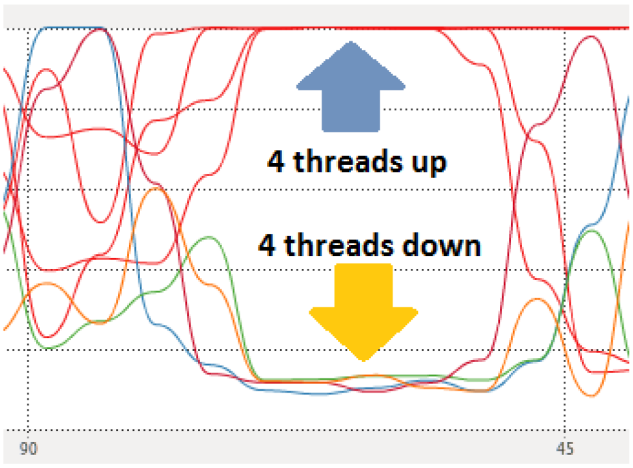

Fig. 3. The CPU usage in scenario with four full loaded running instances.

We also noticed a slight increase in consumption of operating memory. It is caused by simple calculations performed by the implemented program.

Next, we extended our research for performance analysis of task execution depend on selected OpenStack flavour. For this purpose we prepared master-slave model based on client-server idea. Our system assigns the task to a specific instance and waiting for the result. Implementation of communication is based on sockets. Specifications of tested flavours are presented in Tab. 3. As in previous experiments, in order to full-load the computing resources, we used the matrix multiplication benchmark with matrices of 1024x1024 size. The results of our measurements are presented in Tab. 6.

Table 6: Time of matrix multiplication for each flavour.

Flavour	Time [s]
m1.tiny	30,912
m1.small	30,336
m1.medium	27,520
m1.large	32,955
m1.xlarge	31,890

Despite the fact that even the smallest flavour meets the task's demands for computing resources, we received different values. The top performance was received for m1.medium flavour. The results may seem surprising, but keep in mind that, especially in the case of virtualized resources, it can be a situation in which resources increase will reduce the performance. In our case, we have to deal with delays resulting from switching between threads, or fragmented memory. The selected flavour (m1.medium) seems to be the most optimal because of sufficient resources for the system and our test task.

SUMMARY AND CONCLUSIONS

The main aim of this research was to present overview and perform resources analysis of the OpenStack cloud computing software platform. In our paper, we put emphasis on the problems that are meet by a new user and familiarized the installation method of the environment.

In our tests, we have observed a very high demand on operating memory resulting from the allocation of these resources by VMs (mainly controller and compute nodes). In order to set up a test environment at least 16GB of RAM (8GB given in the documentation does not provide a stable environment installation and operation) and processor with hardware virtualization technology (Intel VT-x or AMD-V) are needed [11, 15]. Our recommended method of installation is one of the tools that automate this process. In our study, we used Vagrant OpenStack Provider. Installation directly from the git repository ended in failure associated with the Python libraries dependencies.

OpenStack is multi-platform software, but the biggest support is provided from Ubuntu Server community. Ubuntu Server is recommended operating system for OpenStack.

In the second part of our paper, we evaluate platform demands for hardware resources and we measured resources utilization by compute instances. Creation of two compute VMs and the rest of needed services (a total of 7 machines) absorb the most of available RAM (14,5 of 16 GB). Only a small portion remains for system processes. According to our measurements, the creation and running each of instances with the lightest system irreclaimable absorb about 2% of CPU power. Memory consumption remained unchanged.

Next, we measured the use of resources during fully loaded instances. Two compute nodes allow utilizing up to 50% of computing power. Probably four compute would allow for the use up to 100%, but holed the hardware resources did not allow for the creation of additional compute nodes. The results of our experiments show efficient allocation of computing resources by the OpenStack platform.

Performance analysis of task execution depend on selected OpenStack flavour has shown that increasing the virtual resources is not always involves an increase efficiency. A lot depends on the type and distribution of resources.

According to our observations, OpenStack can be a good option for extensive computer infrastructure. In the case of a single machine losses of performance associated with the implementation of solutions far outweigh the benefits obtained. Nevertheless, having a powerful machine, you can use a platform for simple testing and educational purposes.

ACKNOWLEDGMENT

Daniel Grzonka’s research presented in the paper was partially supported by the European Commission FP7 through the project ParaPhrase: Parallel Patterns for Adaptive Heterogeneous Multicore Systems, under contract no. 288570 (http://paraphrase-ict.eu).

REFERENCES

[1] Moore, G. E. 1998. "Cramming more components onto integrated circuits", Proceedings of the IEEE, Volume 86, Issue 1.

[2] Brooks Jr., F. P. 1995. "The mythical man-month" (anniversary ed.), Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

[3] Dijkstra, E. W. 1972. "The Humble Programmer", Communications of the ACM, Volume 15, Issue 10.

[4] Serafin, M. 2012. "Wirtualizacja w praktyce" (in Polish), Wydawnictwo Helion, Poland.

[5] Popek, G. J. and R. P. Goldberg. 1974. "Formal requirements for virtualizable third generation architectures", Communications of the ACM, Volume 17, Issue 7.

[6] Gagniuc, M. B. 2008. "Virtual Machines Technologies", Proceedings of the 9th International Conference on Development and Application Systems, Suceava, Romania.

[7] Marks, M. and E. Niewiadomska-Szynkiewicz. 2014. "Hybrid CPU/GPU Platform for High Performance Computing", Proceedings of the 28th European Conference on Modelling and Simulation, Brescia, Italy.

[8] Mell, P. and T. Grance. 2011. "The NIST Definition of Cloud Computing", Recommendations of the National Institute of Standards and Technology, USA.

[9] Mhedheb, Y; F. Jrad; J. Tao; J. Zhao; J. Kołodziej and A. Streit. 2013. "Load and Thermal-Aware VM Scheduling on the Cloud", Algorithms and Architectures for Parallel Processing, Lecture Notes in Computer Science, Volume 8285.

[10] Gregg, B. 2013. "Systems Performance: Enterprise and the Cloud", Prentice Hall, USA.

[11] OpenStack Website and Documentation (http://www.openstack.org/).

[12] Pepple, K. 2011. "Deploying OpenStack", O'Reilly Media, USA.

[13] Sefraoui, O.; M. Aissaoui and M. Eleuldj. 2012. "OpenStack: Toward an Open-Source Solution for Cloud Computing", International Journal of Computer Applications, Volume 55, No. 03.

[14] Corradi, A.; M. Fanelli and L. Foschini. 2014. "VM consolidation: A real case based on OpenStack Cloud", Future Generation Computer Systems, Volume 32.

[15] Jackson, K. and C. Bunch. 2013. "OpenStack Cloud Computing Cookbook" (Second Edition), Packt Publishing Ltd, UK.

AUTHOR BIOGRAPHIES

DANIEL GRZONKA received his B.Sc. and M.Sc. degrees with distinctions in Computer Science at Cracow University of Technology, Poland, in 2012 and 2013, respectively. Actually, he is Research and Teaching Assistant at Cracow University of Technology and Ph.D. student at Jagiellonian University in cooperation with Polish Academy of Sciences. He is also a member of Polish Information Processing Society and IPC member of several international conferences. For more information, please visit: http://www.grzonka.eu/.

MICHAŁ SZCZYGIEŁ graduated in Computer Science at the Finnish JAMK University of Applied Sciences and Cracow University of Technology. He is a member of a Cloud team at Finnish Open Source project called FreeNest. Currently, he is working on the aPaaS project called Dev Cloud related with CC1 infrastructure. For more information, please visit portfolio websites: http://www.szczygiel.dl.pl/ and https://github.com/M4GiK/.

ARTUR BERNASIEWICZ is a Computer Science student at the Cracow University of Technology. He is interested in the Internet of Things. He is passionate of technical innovations like new surveillance systems or home automation. His interests include playing mini Quadcopter and watching the ISS flights.

ANDRZEJ WILCZYŃSKI graduated in Computer Science at Cracow University of Technology, Poland, in 2013. He also studied at Fontys University of Applied Sciences, Netherlands, in 2012. Currently, he works as PHP Developer at Grupa Unity. For more information, please visit: www.andrzejwilczynski.com. His e-mail address is: and.wilczynski@gmail.com.

MAREK LISZKA received his bachelor degree at Faculty of Physics, Mathematics and Computer Science at Cracow University of Technology, Poland. Currently he is attending a master's degree program at same department. He is working as PHP Developer, interested in web Technologies such as ZF2, SF2, Spring MVC and jQuery.