Daniel SAWICKI

OPERATING SYSTEMS EFFICIENCY IN CLOUD ENVIRONMENTS

ABSTRACT *This paper presents performance comparison of operating systems running on different cloud environments. The main focus is to compare the performance of the cloud systems: Eucalyptus, Nimbus and virtual system. The performance, of the encountered at the present time solutions of the cloud and virtual systems, is considerably closer to the operating system installed locally. The results of the carried out tests show that in some areas the system installed locally performs better, but in the high throughput computing the cloud environment points out.*

Keywords: *virtualization, cloud computing, Eucalyptus, Nimbus*

1. INTRODUCTION

There is little consensus on how to define the Cloud [8]. From other hand, cloud computing has become nowadays a buzzword among scientists and IT engineers. The cloud phenomenon is quickly growing towards becoming the de facto standard of Internet computing, storage and hosting, both in industry and academia.

Daniel SAWICKI, M.Sc.
e-mail: sawi@politechnika.lublin.pl

Institute of Electronics and Information Technology
Lublin University of Technology

PROCEEDINGS OF ELECTROTECHNICAL INSTITUTE, Issue 260, 2012

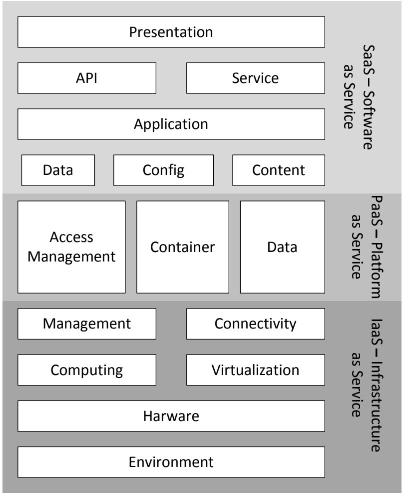


Fig. 1. Cloud building blocks

The cloud building blocks is shown on Figure 1.

The article covers the bottom level of performance cloud structure, the IaaS level, virtualization and local system. Author concentrate on the way, testing operating system efficiency in different environment.

2. CLOUD COMPUTING SERVICE STYLES AND SOLUTION

Cloud computing is typically divided into three levels of service styles, often referred to as the “SPI Model”, where ‘SPI’ refers to Software, Platform or Infrastructure (as a Service), respectively. These levels support virtualization and management of differing levels of the solution stack. The meaning of cloud service levels are [1, 3, 5, 6]:

- IaaS style clouds provide access to collections of virtualized computer hardware resources, including machines, network, and storage. With IaaS, users assemble their own virtual cluster on which they are responsible for installing, maintaining, and executing their own software stack;
- PaaS style clouds provide access to a programming or runtime environment with scalable compute and data structures embedded in it. With PaaS, users develop and execute their own applications within an environment offered by the service provider;
- SaaS style clouds deliver access to collections of software application programs. SaaS providers offer users access to specific application programs controlled The performance, of the encountered at the present time solutions of the virtual systems, is considerably closer to the operating system installed locally;

- The results of the carried out tests show that in some areas the system installed locally performs better, but in the high throughput computing the cloud environment points out;
- In cloud environment we recorded better performance in large images (more resources allocated) but it is not refer to RAM operations;
- Within an organization's infrastructure. Users self-provision and scale collections of resources drawn from the private cloud, typically via web service interface, just as with a public cloud. However, because it is deployed within the organization's existing data center, and in most cases behind the organization's firewall, a private cloud is subject to the organization's security regulations and thus offers a higher degree of security over sensitive code and data;
- A hybrid cloud combines computing resources (e.g. machines, network, storage, etc.) drawn from one or more public clouds and one or more private clouds at the behest of its users.

Before any testbed structure can be proposed, it is necessary to distinguish the characteristics of current IaaS approaches. Private cloud as any other computer system needs models of computation, storage and communication, respectively. Elasticity and the illusion of infinite capacity, the cloud computing is famous for, come from, generally speaking, the statistical multiplexing of possessed and required resources, according to service demands. That is a reason why these resources need, in most cases, to be virtualized. That concerns all mentioned above models, so the classes of IaaS implementation can be distinguished based on the level of management of the resources [7].

Article concentrate on the most popular today cloud computing IaaS platforms. The analysis of the above solution has led to conclude that for purposes of building a IaaS test environment, the most promising is Amazon-like approach. The other vendors are concentrated on providing solution closer to SaaS. Amazon approach is then good start point but its not open source. Good solution for testbed are open source platform: Eucalyptus, Opennebula, Nimbus, Openstack.

2.1. Design of Eucalyptus cloud platform

Eucalyptus is an open source software based on Linux system. Architecture of this system can implements scalable, efficiency-enhancing private and hybrid clouds. In this system users can start, control, access, and terminate entire virtual machines. Users of Eucalyptus interact with the system using the same tools and interfaces that they use to interact with Amazon EC2. Eucalyptus used Xen and KVM/QEMU hypervisor to run virtual machines [2, 6].

There are four components in Eucalyptus installation: Node Controller (NC), Cluster Controller (CC), Storage Controller (Walrus) and Cloud Controller (CLC). The relationship between each Eucalyptus component is shown in Figure 2.

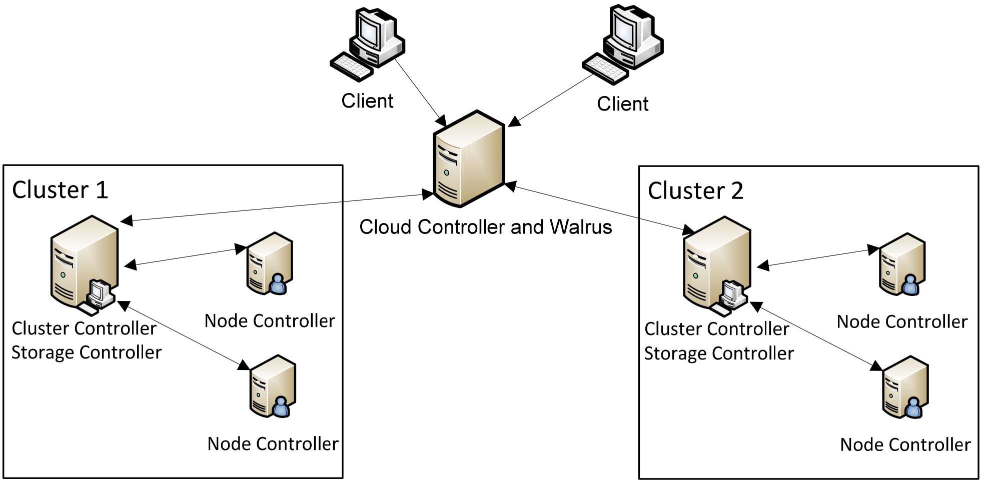


Fig. 2. Hierarchical structure of Eucalyptus

Cloud Controller (CLC) – the cloud controller provides high-level management of the cloud resources. Clients wishing to instantiate or terminate a virtual machine instance interact with the cloud controller through either a web interface or SOAP-based APIs that are compatible with AWS.

Cluster Controller (CC) – the cluster controller acts as a gateway between the CLC and individual nodes in the datacenter. It is responsible for controlling specific virtual machine instances and managing the virtualized network. The CC must be in the same Ethernet broadcast domain as the nodes it manages.

Node Controller (NC) – the cluster contains a pool of physical computers that provide generic computation resources to the cluster. Each of these machines contains a node controller service that is responsible for fetching virtual machine images, starting and terminating their execution, managing the virtual network endpoint, and configuring the hypervisor and host OS as directed by the CC. The node controller executes in the host domain (in KVM) or driver domain (in Xen).

Storage Controller – the storage controller provides persistent virtual hard drives to applications executing in the cloud environment. To clients, these storage resources appear as raw block-based devices and can be formatted and used like any physical disk. But, in actuality, the disk is not in the local

machine, but is instead located across the network. To accomplish this, virtual machines access storage through block-based disk I/O provided by the hypervisor. This bridges the guest domain with the host domain. In the host domain, a driver converts block-based access into network packets that travel across the private network and reach the remote disk. In Eucalyptus, the non-routable ATA over Ethernet protocol is used for networked disk access, which requires that the virtual machine and cluster controller be on the same Ethernet segment. EBS data is stored in pre-allocated files on disk, but the same protocol could be used to export entire drives directly across the network.
Walrus Storage Controller – Walrus provides an API-compatible implementation of the Amazon S3 (Simple Storage Service) service. This service is used to store virtual machine images and application data in a file, not block, oriented format.

2.2. Design of Nimbus cloud platform

Nimbus is an open source cloud computing Infrastructure as Service platform. In this system users can managed virtual machines and build the required computing environment. Figure 3 shows that nimbus cloud computing platform includes many different components [4].

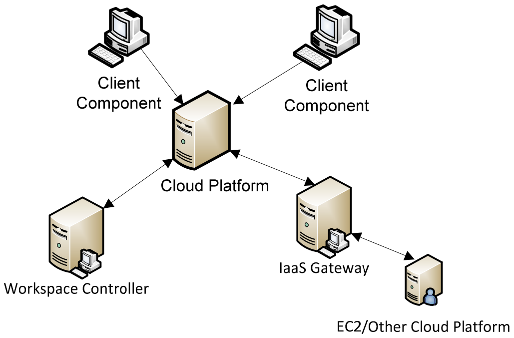


Fig. 3. Structure of nimbus cloud platform

These many functional components can be classified as three elements. First is client modules (context client, cloud client, reference client and EC2 client) are used to support all cloud clients. The second is service modules (context agent, web service resource framework, EC2 WSDL and remote interface), providing all kinds of cloud services. The third is the resource management modules (work service management, IaaS gateway, EC2 and other cloud platform support, workspace pilot, workspace resource management and workspace controller) which are used to manage physical resources on the cloud computing platform. The most important components functions are presented. Workspace service module providing access to virtual machines in different kinds of remote protocol. The current supported protocols are Web Services based or HTTP based. They all run in Apache Axis based Java container. Cloud client module permit user run the requirement resources in simple user interface. The reference client exposes all features of the WSRF frontend as a command line client. It is relatively complex to use and thus typically wrapped by task-specific scripts. EC2 WSDL is an implementation of two of the Amazon Elastic Compute Cloud (EC2) interfaces that allow to used the real EC2 system by Nimbus clients. There is supported for both EC2 interfaces: SOAP and Query. Workspace-control is a program installed on each VMM (support Xen and KVM) node used to to start, stop and pause VMs, implement VM image reconstruction and management, connect the VMs to the network [4].

3. TESTBED STRUCTURE

The testbed was a three system structure: Eucalyptus, Nimbus. Each system individual components have been implemented based on the PCs. These computers were equipped with dual-core Intel processor with support hardware virtualization, 2 GB RAM, disk storage capacity of 250 GB and 2 network interfaces. The basis for the implementation of a virtualization is KVM hypervisors.

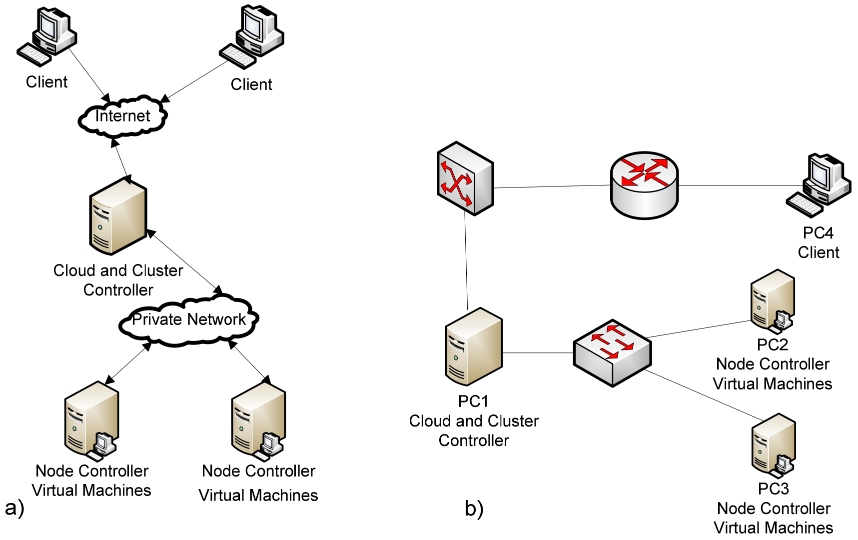


Fig. 4. Testbed architecture: a) logical, b) physical

The individual elements of the structure of the system test were assigned to that shown in Figure 6b, computer equipment, as follows:

- cloud controller, cluster controller – PC1 (2 cores, 4 logical processors, shared), 2 GB RAM, 250 GB HDD SATA, Fedora 12 (x86_64);
- node controller – PC2 and PC3 (2 cores, 4 logical processors, shared), 2 GB RAM, 250 GB HDD SATA, Fedora 12 (x86_64);
- client – PC4 (1 core, 2 logical processors, shared), 512 MB RAM, 8 GB HDD SATA, Fedora 12 (x86_32).

Testbed architecture is show in Figure 4.

3.1 Test results

Built on such a platform testing was done to investigate the performance of systems using virtualization. Their performance was tested on the following devices:

- PC computer (2 cores), 2 GB RAM, 250 GB HDD SATA, Fedora 12 (x86_64) – Testbed for testing identified later in the article as "Fedora";
- Ubuntu 9.04 system running in a KVM virtual machine environment node Eucalyptus Cloud, VM (1 core), 0.12 GB RAM, 5 GB HDD SATA, Fedora host system 12 (x86_64) – Testbed for tests marked "VM";
- Virtual instance running on Eucalyptus cloud (small image), 1 core, 0.12 GB RAM, 5 GB SATA HDD, Ubuntu 9.04 (x86_64) – Testbed for testing indicated later in the article as "Eucalyptus";
- Virtual instance running on Nimbus cloud (small image), 1 core, 0.12 GB RAM, 5 GB SATA HDD, Ubuntu 9.04 (x86_64) – Testbed for testing indicated later in the article as "Nimbus";
- Virtual instance running on Eucalyptus cloud (large image), 2 core, 2 GB RAM, 60 GB SATA HDD, Ubuntu 9.04 (x86_64) – Testbed for testing indicated later in the article as "Eucalyptus_L";
- Virtual instance running on Nimbus cloud (large image), 2 core, 2 GB RAM, 60 GB SATA HDD, Ubuntu 9.04 (x86_64) – Testbed for testing indicated later in the article as "Nimbus_L".

During the test positions were not running additional processes. Were selected several types of tests in order to obtain meaningful performance results of individual operating systems. Some tests are:

- Packing a file size of 311 MB using tar and gzip commands,
- Compile ffmpeg package,
- Compile lame package,
- Linpack Benchmark,
- Ramtest.

Packing results file using the *tar* command shows Figure 5. Packing time was determined using the *time* command, and the graph presents the actual period of time in which the program was running (real time).

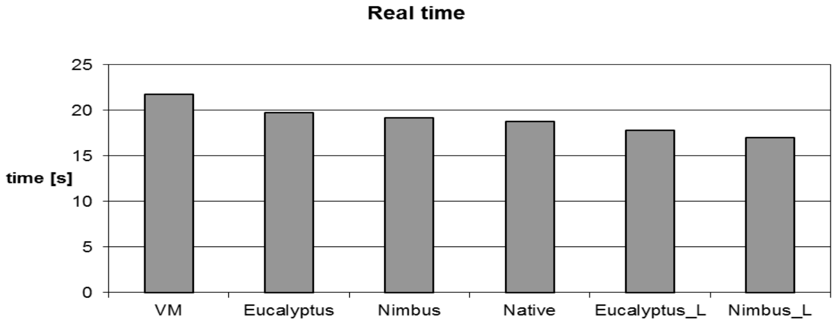


Fig. 5. Packing time of the file with tar command

File I/O operations in the virtual machine and cloud system (small image) take more time than in native systems. The same operations take less time in cloud systems (large image) than in native systems. The confirmation of this fact is to present a so-called system overhead. System overhead is defined as a percentage value, as compared CPU time in system mode, the processor until they actually devoted to the processing program in a process running mode where the processor was working in user mode (sys/user · 100%). Result can be seen in Figure 6 that for native systems overhead is the smallest, about 1%. Although real time packing gives better results for large cloud images the system overhead is two times larger.

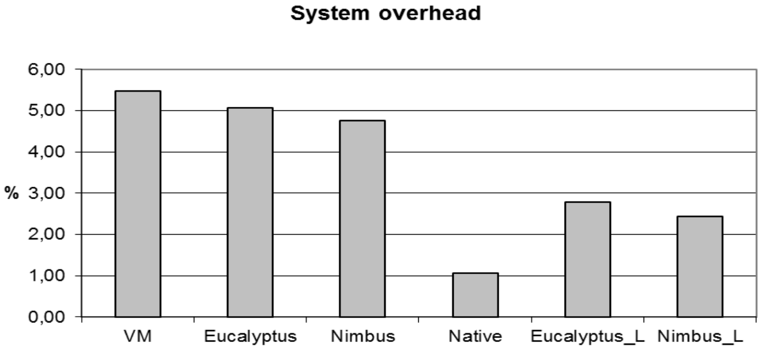


Fig. 6. Overhead of the system for packing the file

Another test conducted was the compilation of the package ffmpeg. Compilation was carried out in the hardware configuration described in section 3.1. The results of the total compilation time is shown in Figure 7.

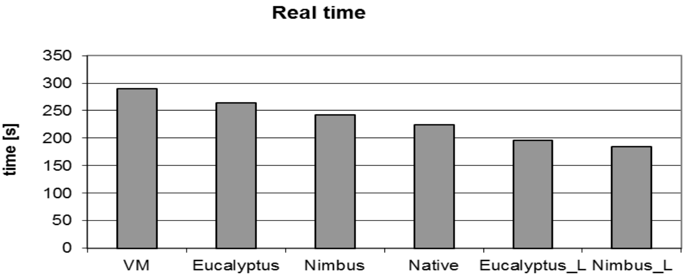


Fig. 7. Ffmpeg compilation time

Compile-time analysis of the results can be seen that virtual systems are about 20% slower compared to the native operating system installed. Nimbus large image are 15% faster than native system.

Among the virtual systems in the process of compiling a cloud architecture also has an advantage in relation to the system running in a virtual machine. In Figure 8 you can see a clear difference between Eucalyptus and VM systems, if the clouds system overhead is about 40% lower compared to the virtual machine.

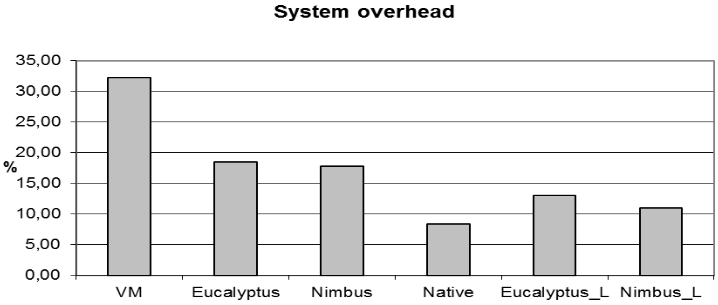


Fig. 8. Overhead of the system for ffmpeg compilation

The results of lame compilation is similar to the ffmpeg compilation which is shown on Figures 9 and 10.

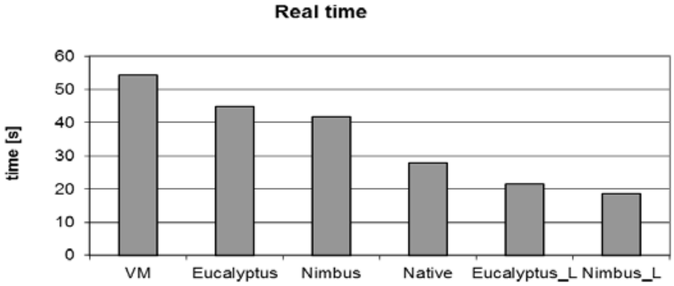


Fig. 9. Lame compilation time

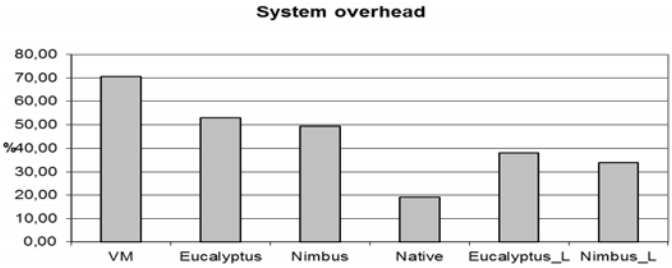


Fig. 10. Overhead of the system for lame compilation

Another test performed on such a test platform was compiled Linpack Benchmark. Figure 11 presents a summary of results for this test.

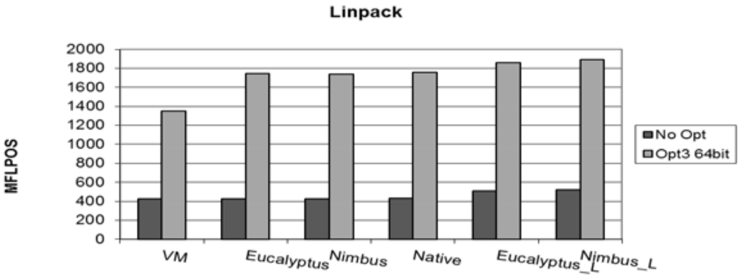


Fig. 11. Results of Linpack benchmark

The results of the benchmark without the optimization are almost identic (a difference of one MFLOPS in Eucalyptus, Nimbus, Native), while in the case of optimization is clearly worse than the system running in a virtual machine. Large cloud images gives better results of the benchmark about 100 MFLOPS.

The last test was performed ramtest. The measurement results are shown in Figures 12-15. The best results were obtained for reading cloud systems (Eucalyptus_L, Nimbus_L), they were about 40% better than the native system. This value increased to 60% for reading small blocks of 1024 KB.

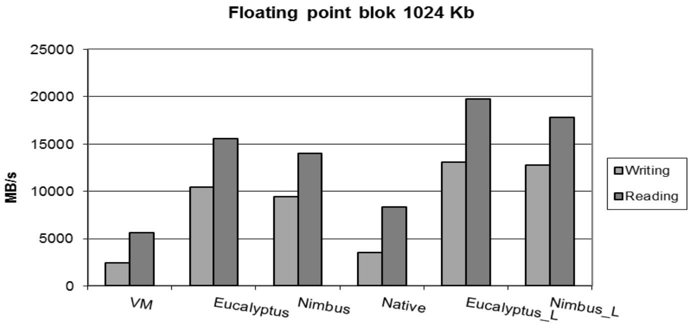


Fig. 12. Results of RAMtest floating point block 1024 Kb

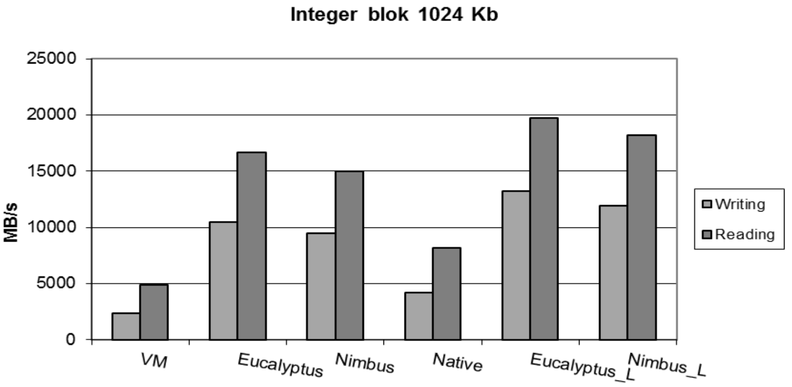


Fig. 13. Results of RAMtest integer block 1024 Kb

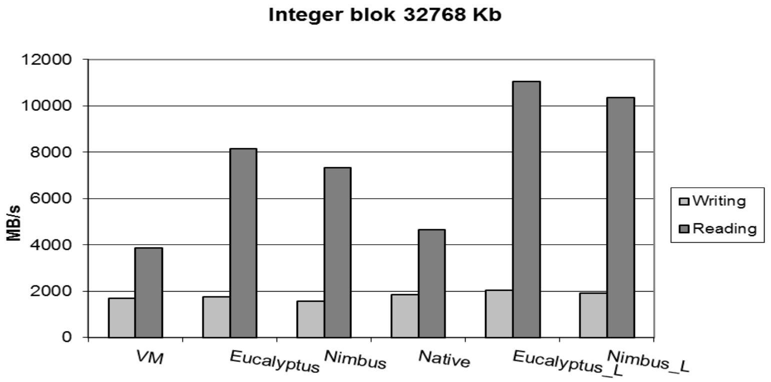


Fig. 14. Results of RAMtest integer block 1024 Kb

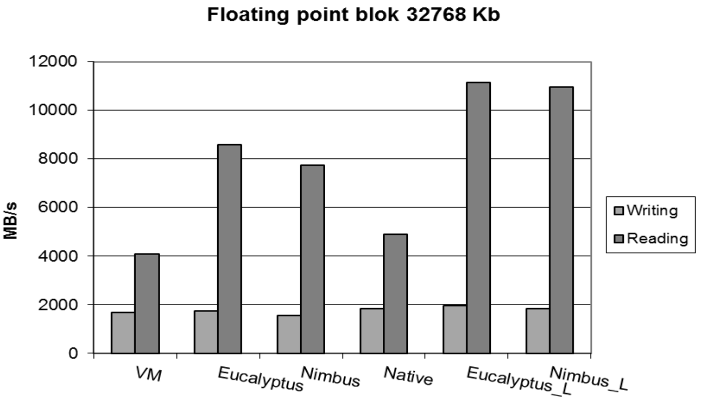


Fig. 15. Results of RAMtest floating point block 32768 Kb

3.1 Conclusions

The article compared the performance of the cloud systems, virtual system, and native system. This performance compares with locally installed operating system. After testing, following conclusions were drawn.

- The results shows that cloud systems (only large images) are better performance than native system;
- The performance, of the encountered at the present time solutions of the virtual systems, is considerably closer to the operating system installed locally;
- The results of the carried out tests show that in some areas the system installed locally performs better, but in the high throughput computing the cloud environment points out;
- In cloud environment we recorded better performance in Nimbus than Eucalyptus system.

LITERATURE

1. Armbrust M., Fox A., Griffith R., Joseph A.D., Katz R., Konwinski A., Lee G., Patterson D., Rabkin A., Stoica I., Zaharia M.: Above the Clouds: A Berkeley View of Cloud Computing, Berkeley Reliable Adaptive Distributed Systems Laboratory, September 2010.
2. Eucalyptus Administrator's Guide (2.0): http://open.eucalyptus.com/wiki/, March 2011.
3. Foster I., Zhao Y., Raicu I., Lu S.: Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop, 2008, pp.1-10.
4. Lei Z., Zhang B., Zhang W., Li Q., Zhang X., Peng J.: Comparison of Several Cloud Computing Platforms. Second International Symposium on Information Science and Engineering, pp. 23-27, 2009.
5. Johnson I., Lakhani A., Matthews B., Yang E., Morin C.: Cloud Computing and Grid Computing 360-Degree Compared, Grid Computing Environments Workshop, 2008, pp.1-10.
6. Nurmi D., Wolski R., Grzegorczyk Ch., Obertelli G., Soman S., Youseff L., Zagorodnov D.: The Eucalyptus Open-source Cloud-computing System, 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), Vol. 0, 2009, pp. 124-131.
7. Przyłucki S., Sawicki D.: Cloud computing testbed, Studia Informatica, Vol. 32, No. 3A, 2011, pp. 83-92.
8. Vaquero L.M., Rodero-Merino L., Caceres J., Linder M.: A Break in the Clouds: Towards a Cloud Definition, ACM SIGCOMM Computer Communication Review, Vol. 39, No. 1, 2009, pp. 50-55.

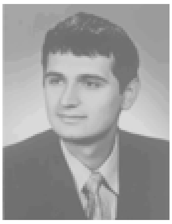
Manuscript submitted 06.08.2012

WYDAJNOŚĆ SYSTEMÓW OPERACYJNYCH W ŚRODOWISKACH CHMUROWYCH

Daniel SAWICKI

STRESZCZENIE *Artykuł prezentuje porównanie wydajności systemów operacyjnych uruchamianych w różnych środowiskach chmurowych. Główny nacisk położony jest na porównanie wydajności systemów chmurowych: Eucalyptus Nimbus wraz z system wirtualnym. Wydajność obecnie dostępnych rozwiązań systemów chmurowych i wirtualizacji jest zbliżona do lokalnie zainstalowanych systemów operacyjnych. Wyniki przeprowadzonych badań wskazują, że w niektórych obszarach lokalnie zainstalowany system działa lepiej, ale w czasie, gdy wymagana jest duża przepustowość, środowisko chmurowe jest wydajniejsze.*

Słowa kluczowe: *wirtualizacja, sieci chmurowe, Eucalyptus, Nimbus*



M.Sc. Daniel SAWICKI – graduated from the Faculty of Electrical Engineering and Computer Science University of Lublin. In 2007, he started PhD studies. Currently working as an assistant at the Technical University of Lublin.