International Journal of Cloud Computing and Services Science (IJ-CLOSER)
Vol.2, No.4, August 2013, pp. 288~295
ISSN: 2089-3337

Scalability Analysis of KVM-Based Private Cloud For IaaS

Fayruz Rahma*, Teguh Bharata Adji*, Widyawan*
* Department of Electrical Engineering and Information Technology, Gadjah Mada University

Article Info

Article history:

Received Jun 15th, 2013
Accepted Jul 30th, 2013

Keyword:

Virtual machine
Scalability
Kernel-based virtual machine
Cloud computing

ABSTRACT

One of the cloud technology cores is virtualization. Virtual Machine Manager (VMM) is said to have good scalability if it provides services to many virtual machines with a fair management of resources to maintain optimal performance of virtual machines'. Scalability evaluation of virtualization technology needs to be done so that cloud developers can choose the appropriate VMM according to the scenario of cloud usage. This study was conducted to determine the scalability of KVM in a cloud with OpenStack platform. Three scalability metrics were used (overhead, linearity, and isolation) to measure the scalability of different machine resources: CPU, network, and disk. The results showed that KVM exhibits good scalability in CPU and network. KVM is suitable for a scenario in which isolation between its CPU and harddisk is needed. KVM is suggested not to be used in a scenario where harddisk is accessed by many VMs intensively.

Copyright © 2013 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Fayruz Rahma,
Departement of Electrical Engineering and Information Technology,
Gadjah Mada University,
Jalan Grafika 2, Yogyakarta 55281, Indonesia.
Email: fayruzrahma@te.gadjahmada.edu

1. INTRODUCTION

Cloud computing research area is growing explosively. Cloud computing is the expansion of parallel computing, distributed computing, and grid computing. Cloud is able to make efficient use of machine resources. Private cloud is composed of a collection of virtual machines running on hosted infrastructure (including processors, servers, network, and storage) owned by one stakeholder group who provides shared access to many customer in other stakeholder groups [1]. The capacity of private cloud can be, adjusted accordingly with the internal needs, extended or shrunk.

Virtualization is a technology to make a virtual version of physical machine/abstraction, such as operating system, storage device, or network resources. There are three types of virtualization known as Full-Virtualization, Para-Virtualization, and Isolation [2].The advantages of virtualization are the reduction of the hardware cost, simplicity of backup and recovery, reduction of the system's heat, cost reduction of physical maintenance, and the easiness of changing or upgrading the system. The disadvantages of virtualization are the central problem, the high specification of hardware, and one central attack.

Virtualization is the core of cloud technology. Infrastructure as a Service (IaaS), an instance of service from cloud, is made from virtualization. IaaS is a service which rent the basic of information technology resources such as storage device, processing power, memory, network capacity, and other hardware resources. Virtual machines (VMs) co-located on the same physical host share both memory and CPU resources. VM consolidation to reduce cloud power consumption has been studied in [3] and the results showed that to prevent excessive performance degradation, VM consolidation has to be carefully guided.

Scalability aspect is an important point in the implementation of virtual technology. Scalability performances of cloud infrastructure can be measured by its parameters such as overhead, linearity, and performance isolation [4]. Overhead is the processing time required by a VM prior to the execution of a

command. Linearity is a quantitative assessment of how strongly related a set of data is. There are different definitions of performance isolation and no consensus yet about how VM should behave with respect to this parameter. In principle, performance isolation ensures that in a situation of load imbalance between VMs, all VMs will still get an equal access to the machine resources [4].

Cloud scalability is determined by its virtualization technology, called hypervisor, which provides VMs to the cloud's user. Kernel-based Virtual Machine (KVM) is a full-virtualization solution for Linux host based with hardware-assisted virtualization feature (Intel VT or AMD-V). KVM is implemented as a loadable kernel module which changes Linux kernel into bare metal hypervisor.

There are two principles of KVM [5]. Firstly, as KVM is designed after the emerging of hardware assisted virtualization, it did not have to implement features that were provided by hardware. The KVM hypervisor requires Intel VT-X or AMD-V enabled CPUs and leverages those features to virtualize the CPU. Secondly the KVM team applied a tried and true adage: "don't reinvent the wheel". There are many components that a hypervisor requires in addition to the ability to virtualize the CPU and memory, for example: a memory manager, a process scheduler, an I/O stack, device drivers, a security manager, a network stack, etc. In fact, a hypervisor is really a specialized operating system, differing only from its general purpose peers in that it runs virtual machines rather than applications. Since the Linux kernel already includes the core features required by a hypervisor and has been hardened into a mature and stable enterprise platform by over 15 years of support and development, it is more efficient to build on that base rather than writing all the required components such as a memory manager, scheduler, etc from the ground up.

Virtualization method and the usability of KVM have been studied together with Xen, VirtualBox, and VMWare in [6]. This paper stated that KVM is the best choice for use within HPC Cloud environments. The performance of KVM and Xen has been also compared in [7]. The results showed that Xen is superior in CPU intensive test and KVM has better performance in write and read operation because of its ability of disk caching. CPU virtualization overhead of Xen has been studied in [8]. It is stated that in realistic configurations the component approach does not give significant overhead. KVM scalability analysis in the researched cloud has been done in [9], but only the overhead of database access and web server was studied. The scalability of this cloud hasn't been thoroughly researched. Hence, we conducted an investigation of KVM scalability with three parameters (overhead, linearity, and isolation performance) for three resources (CPU, network, and harddisk).

2. RESEARCH METHOD

To measure KVM scalability, microbenchmark has been done in an OpenStack cloud (version 2011.3) with KVM hypervisor. The operating system of the host is Ubuntu server (version 10.04) and the guest operating system is Ubuntu Desktop 10.04 (Lucid Lynx). Five servers are used in these tests. Four servers (server1, server2, server3, and server4) are used for the cloud system with its total resources are 20 cores processor @1 GHz, 32 GB DDR3 memory, and storage up to 2,5 TB. The fifth server serves as the native machine with 8 cores processor, 8 GB memory, and storage up to 1 TB. The topology of researched private cloud is illustrated in Figure 1. The virtual machines are assigned to the hosts manually since OpenStack does not provide dynamic service provisioning and rescaling [3].

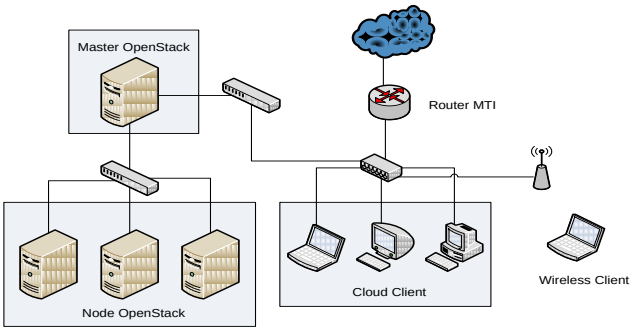

Figure 1. The topology of the researched private cloud

The tested resources are: CPU, network, and harddisk. Some simple microbenchmarks are used to emphasize the scalability characteristics of single resource. Their simplicity permits us to exhibit the scalability limitation of virtualization tools, per resource basis. The microbenchmarks are chosen based on the review of Quetier, etc [4]. Each test is conducted ten times and the mean of each test is taken to further analysis.

CPU tests use Maxima to iteratively calculate an estimate of √2 until 1.000.000 digits. This test was chosen because it minimizes the memory usage. Network measurements use netperf benchmark. A TCP stream is sent from netperf client to netserver for ten seconds and the throughput is recorded. Disk measurements use a disk duplication tool (dd in read only). A 2Gb file is read and the time needed is recorded. The memory and computation consumption of netperf and dd are negligible compared with the activity they impose to the network and disk.

Three scalability parameters are measured: overhead, linearity, and performance isolation. Overheads (O_v) are measured by comparing the execution time in native machine (T_a) with the execution time in virtual machine (T_{av}). T_a is also compared with T_{anv} (when n VMs run concurrently). In this test, only single VM actually runs the application. The other n-1 VMs are free of application. Reference virtualization overhead is measured by (1)[4]. Virtualization overhead for n VMs (O_{vn}) is calculated by (2)[4]. Performance degradation (D) is calculated by (3).

$$O_v = T_{av} - T_a \quad (1)$$

$$O_{vn} = T_{anv} - T_a \quad (2)$$

$$D = O_v/T_a \quad (3)$$

To evaluate the linearity of cloud scalability, the execution time of the same application running at the same time on several virtual machines is measured. Crontab in Linux is used to make the applications start at the same time. Each VM is set to have 1 core CPU (1GHz), memory of 512 MB, and 10 GB virtual harddisk.

If the virtualization overhead is stable at certain value (independent from the amount of virtual machines), the maximum of application running times should be an affine function of the amount of virtual machines running the application. If T_{av} is the application running time on a virtual machine, then, if n VMs run this application concurrently, the maximum of the application running time (T_{max}) can be calculated using formula (4)[4], where O_v is the overhead of one virtualization. Some virtualization approaches may have a more complex behavior, showing execution times according to the number of running VMs greater than the one driven by the virtualization overhead only. In such case, we will consider that there is at least a second component in the overhead, in addition to the one imposed by the virtualization mechanism [4].

$$T_{max} = O_v + T_{av} \times n \quad (4)$$

To understand the characteristic of performance isolation, two VMs are run at the same time (VM_1 and VM_2), executing one application on VM_1 and two copies of the same application on VM_2. If the kernel schedules the resources by process, all executions will finish at the same time. If there is performance isolation between VMs, the kernel schedules the resources by VM and the application of VM_1 will terminate the execution before VM_2.

3. RESULTS AND ANALYSIS

In this part, the discussed results are related to the virtualization overhead, linearity, and performance isolation for three resources of the physical machines.

3.1. Overhead

As explained in the previous section, the execution time of a single application run on the host is compared with the execution of the same application running on a VM while the number of VMs is increased from 1 to 10. The theoretical value of the overhead corresponds to the one where only a single VM is running. Table 1 shows the mean of the test in both native and single virtual machine. The overhead of virtualization and the degradation performance in Table 1 are calculated using (1) and (3).

Table 1. The Comparison of Execution Time on Native And Virtual Machine

Resource	Native	Virtual Machine	Overhead	Degradation
CPU (seconds)	18.314	24.733	6.455	35,05%
Network (Mbps)	94.15	94.142	0.01	0.0085%
Harddisk (seconds)	3.107722	7.382656	4.274934	137.56%

The results in Table 1 exhibit a very low overhead virtualization on network. The network virtualization overhead is insignificant compared to the execution on the native machine. CPU virtualization shows a fairly high overhead, and harddisk virtualization exhibits a very high overhead with its over 100% degradation. This huge difference in the overhead of harddisk virtualization is, perhaps, occurred because of

the usage of different physical harddisk. The harddisk access time is greatly affected by its rotational latency and its data transfer rate which is affected by the physical condition of the harddisk.

Figure 2 presents the CPU overhead while the number of VMs is increasing from 1 to 10. The graph shows that when the number of concurrent VMs is increasing, KVM exhibit near optimal scalability concerning the overhead parameter (the theoretical curve).

Figure 3 presents the network overhead. For this test, the application throughput reduction is measured while using 1 to 10 VMs. The chart in Figure 3 shows that the network overhead test result is almost the same with theoretical throughput. It means that the increasing of idle VMs did not affect the network performance in the tested VM.

Figure 4 presents the harddisk overhead. This chart shows that the increasing of idle VMs did not affect the harddisk overhead in the tested VM.

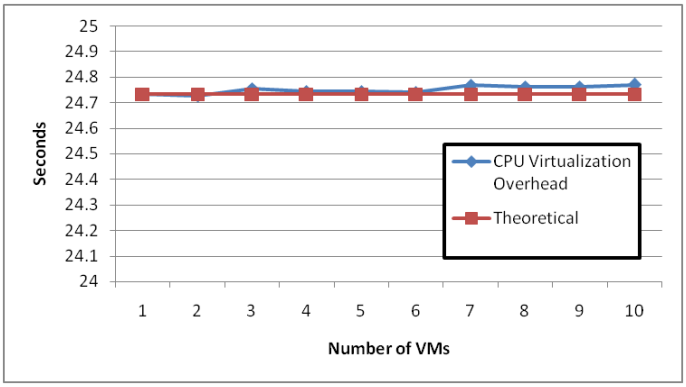

Figure 2. CPU overhead according to the number of VMs

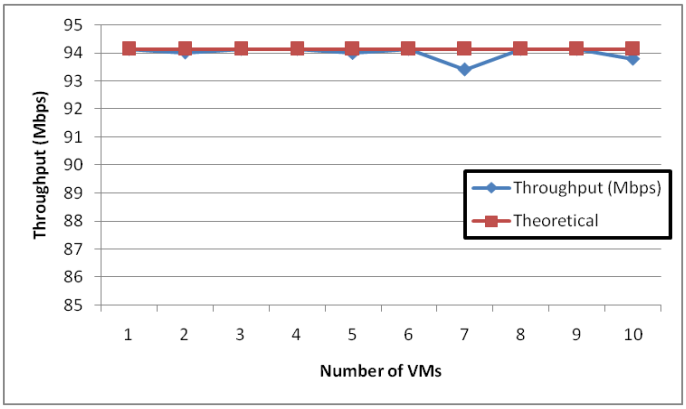

Figure 3. Network overhead according to the number of VMs

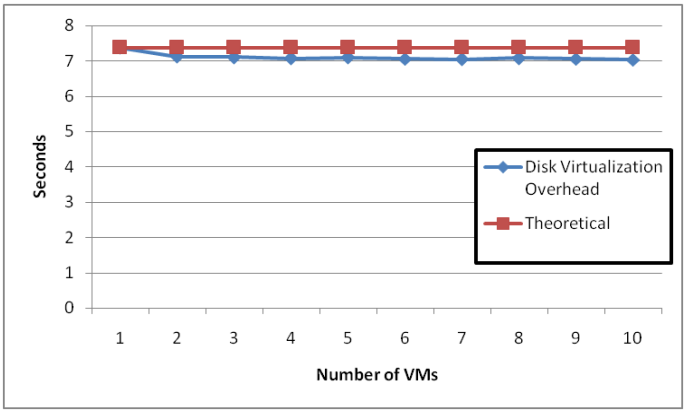

Figure 4. Disk overhead according to the number of VMs

3.2. Linearity

In linearity test, the number of VMs executing the same application is scaled. For a normal linearity, the execution time should increase linearly as a function of the number of concurrent running VMs. All presented results are means of the execution time measured on all running VMs in the same physical machine. The standard deviation is also presented to check how the resource is allocated to the concurrent VMs.

Figure 5 presents the CPU linearity according to the number of concurrent VMs running the application. KVM exhibits an unstable linearity in CPU virtualization. The gradient of the curve is getting bigger when 4 VMs is running together. The physical server, which is used for this research, has 4 cores; one of them is already used by cloud components (nova-compute). When 4 VMs are running concurrently, 5 tasks are executed while only 4 cores are available in the host. OpenStack does not provide dynamic allocation to manage VMs to get fair resources from cloud's physical resources even though there are idle processors in other physical servers. Thus, the execution time is getting slower.

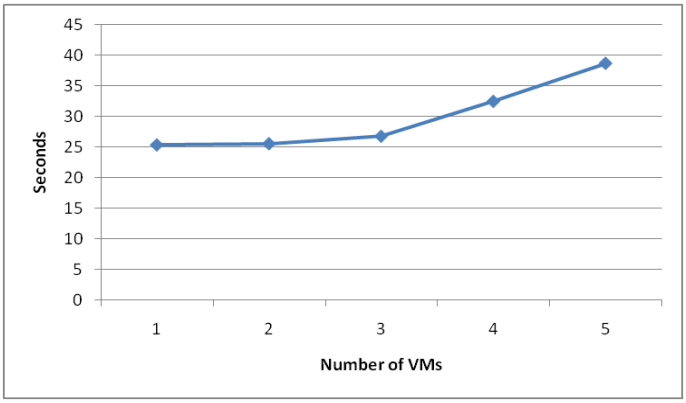

Figure 5. CPU linearity according to the number of VMs

Figure 6 presents the network linearity according to the number of concurrent VMs running in the application. The graph shows that the reduction of throughput is getting smaller even though the number of VMs is growing. This result shows that KVM provide good scalability in network virtualization.

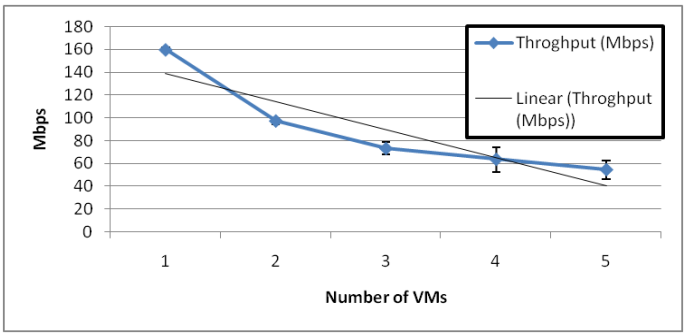

Figure 6. Network linearity according to the number of VMs

Figure 7 presents the harddisk linearity according to the number of concurrent VMs running in the application. The graph shows that the rise of execution time is too high, even much higher than the overhead on a single VM. This indicates that KVM exhibits poor linearity and poor scalability in harddisk virtualization. The high harddisk virtualization overhead of KVM (Table 1) explains this bad results. In addition, there are other overheads of harddisk (such as accessing time, seek time, data transfer rate, etc.) which affect the harddisk performance. Thus, KVM is suggested not to be used in a scenario where harddisk is accessed by many VMs intensively.

3.3. Isolation Performance

This test is conducted to know the characteristic of performance isolation on the cloud resources. A simple scenario is used where two VMs are running three microbenchmarks (two on one VM and one on the other VM).

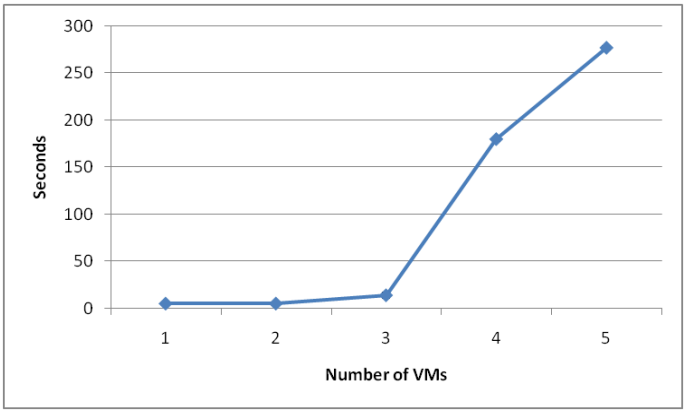

Figure 7. Harddisk linearity according to the number of VMs

Figure 8 shows that the execution time on VM_2 is twice of the execution time on VM_1. The execution time on VM_1 is similar to the execution time where two VMs running two processes concurrently which is approximately 26 seconds (Figure 5). This result shows that KVM provides processor isolation between its virtual machines.

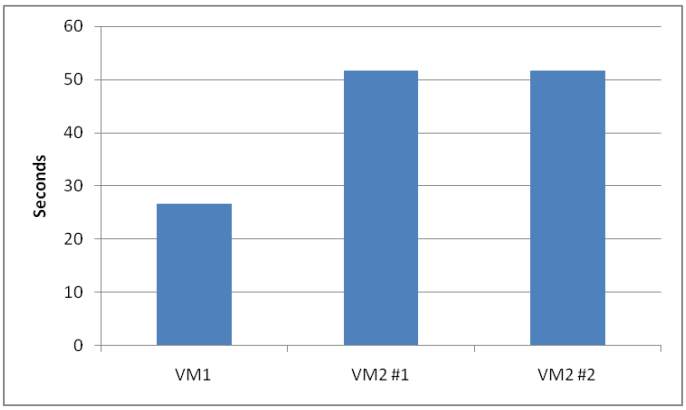

Figure 8. CPU performance isolation

Figure 9 exhibits that there is no network isolation between VMs. The throughput of each process is approximately 105 Mbps. KVM exhibits no performance isolation between VMs so that the network resource is divided to three process (the isolation is implemented between each process). Hence, each process gets one third of the physical network resource.

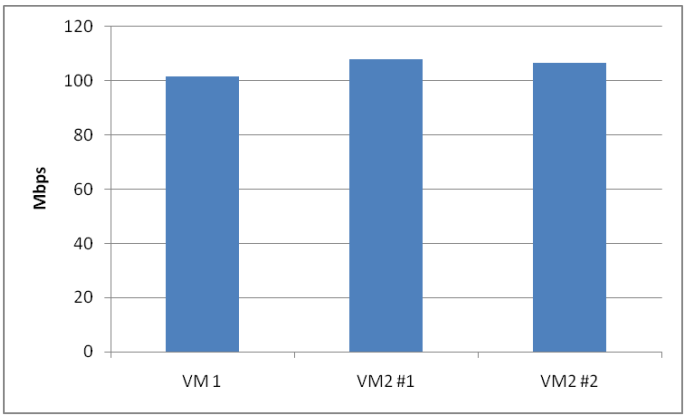

Figure 9. Network performance isolation

Figure 10 presents the harddisk performance isolation. The chart shows that the execution time on VM_2 is double of the execution time on VM_1. The execution time on VM_1 is similar to the execution time of

two VMs running two processes concurrently which is, approximately, 4.9 seconds (Figure 7). This result shows that KVM provides virtual harddisk isolation between its virtual machines.

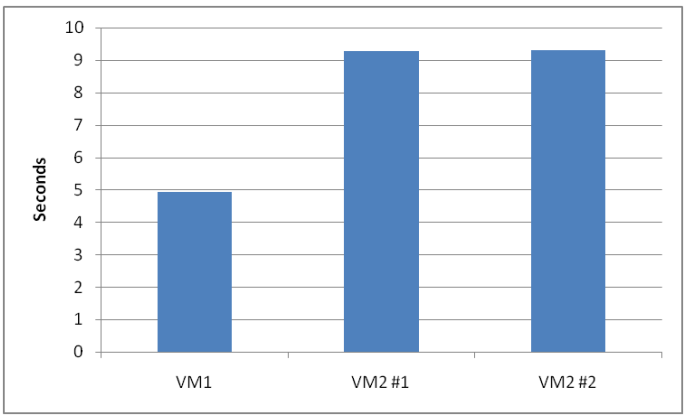

Figure 10. Harddisk performance isolation

4. CONCLUSION

KVM exhibits very low network virtualization overhead, medium CPU virtualization overhead, and very high harddisk virtualization overhead. KVM shows acceptable linearity for CPU and network virtualization, but poor linearity for harddisk virtualization. KVM exhibits performance isolation between VMs for CPU and harddisk virtualization, but no performance isolation between VMs for the network. Thus, KVM will match scenarios requiring CPU performance isolation between VMs. KVM is not recommended to be used in a scenario where many virtual harddisks will be accessed frequently in the same time because of the high harddisk overhead and its poor linearity. We are interested to study the cloud scalability of harddisk if OpenStack Swift (a component to manage virtual harddisks) is used.

REFERENCES

[1] Chung L, Hill T, Legunsen O, Sun Z, Dsouza A, Supakkul S. "A Goal-oriented Simulation Approach For Obtaining Good Private Cloud-based System Architectures". *The Journal of Systems and Software*, vol. 86, pp. 2242-62, 2013.

[2] Carlin S, Curran K. "Cloud Computing Technologies". *International Journal of Cloud Computing and Service Science (IJ-CLOSER)*, vol. 1, no. 2, pp. 59-65, 2012. http://iaesjournal.com/online/index.php/IJ-CLOSER/article/view/Northern%20Ireland%2C%20UK/pdf.

[3] Corradi A, Fanelli M, Foschini L. "VM Consolidation: A Real Case Based On OpenStack Cloud". *Future Generation Computer Systems*, pp. 1-10, 2012.

[4] Quétier B, Neri V, Cappello F. "Scalability Comparison of Four Host Virtualization Tools". *J Grid Computing*, vol. 5, pp. 83-98, 2007.

[5] Red Hat I. "KVM - Kernel-based Virtual Machine". 2008. Available from: http://www.redhat.com/resourcelibrary/whitepapers/doc-kvm.

[6] Younge AJ, Henschel R, Brown JT, Laszewski Gv, Qiu J, Fox GC. "Analysis of Virtualization Technologies for High Performance Computing Environments". *2011 IEEE 4th International Conference on Cloud Computing*: IEEE. pp. 9-16, 2011.

[7] Deshane T, Shepherd Z, Matthews JN, Ben-Yahuda M, Shah A, Rao B. "Quantitative Comparison of Xen and KVM". *Xen Summit*; June 23-24, 2008; Boston, MA, USA., 2008.

[8] Malawski M, Meizner J, Bubak M, Gepner P. "Component Approach to Computational Applications on Clouds". *Procedia Computer Science*, vol. 4, pp. 432-41, 2011.

[9] Nggilu F. "Analisis Overhead Sebagai Salah Satu Faktor Skalabilitas Private Cloud Computing untuk Layanan IaaS": Gadjah Mada University; 2012.

BIOGRAPHY OF AUTHORS

Fayruz Rahma currently is a Master student in Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei. She received her Bachelor's Degree from Department of Electrical Engineering and Information Technology, Gadjah Mada University, in 2012. Her research interests include Cloud Computing and Computer Networks. Her recent projects include Scalability Analysis of Cloud Computing and Network Mobility (NEMO).
E-mail: fayruzrahma@gmail.com

Teguh Bharata Adji received his M.Eng from Doshisha University, Kyoto in 2001 and his Ph.D from Universiti Teknologi Petronas, Malaysia in 2010. He joined Gadjah Mada University as a lecturer from 1995. His interests include AI (Genetic and DNA algorithm), Data Mining, and Natural Language Processing. His recent projects include Path Planning for UAV, animation technology using Kinect / Asus Xtion, parallel and cloud computing, Question-Answering Algorithm, and Information Extraction.
E-mail: adji@ugm.ac.id

Widyawan received his M.Sc from Erasmus University Rotterdam, in 2003 and his Ph.D from Cork Institute of Technology, Ireland, in 2009. He joined Gadjah Mada University as a lecturer in 2009. His research interests include Pervasive & Mobile Computing, Wireless Sensor Network, and Indoor Localization. His recent projects include Home Surveillance IP Camera System, Mobile Technology for Tourism Industry, Wireless Sensor Network for Landslide Detection, and Smart Building for Energy Saving.
E-mail: widyawan@ugm.ac.id