*ICIC Express Letters*
Volume **11**, Number **1**, January **2017**

ICIC International ©2017 ISSN 1881-803X
pp. **37–43**

# PERFORMANCE IMPROVEMENT CONSIDERATIONS OF CLOUD LOGGING SYSTEMS

Pakorn Chan-in[1] and Winai Wongthai[1,2,*]

[1]Department of Computer Science and Information Technology
[2]Center of Excellence in Nonlinear Analysis and Optimization
Faculty of Science
Naresuan University
Phitsanulok 65000, Thailand
*Corresponding author: winaiw@nu.ac.th

Received June 2016; accepted September 2016

Abstract. *Low cost IT investment and flexibility are the main benefits of the cloud. It is one of the preferred IT solutions of many organizations. However, security of the cloud is still a big issue from the cloud customer's perspective. To deal with the issue, demand for accountability in the cloud has increased. Accountability needs an audit log and a logging system is a key requirement for producing the log. However, previous work has provided such logging systems without considering performance improvement for the system. This paper addresses these issues and discusses performance improvement of logging systems in the cloud. Improving the comprehensiveness and accuracy of the logging systems is the focus of this research. To the best of our knowledge, these matters have not yet been described in the literature. The findings reported in this paper can assist logging system developers in designing, implementing, and deploying their systems, and ensuring that the systems appropriately and effectively meet security and performance requirements.*

**Keywords:** Cloud, IaaS, PaaS, Cloud security, Accountability, Logging system, Performance measurement, Performance improvement

1. **Introduction.** Cheap IT investment costs and flexibility are the main benefits of the cloud, and the cloud is one of the preferred IT solutions of many organizations [1]. However, security of the cloud remains a significant issue affecting both cloud providers and customers [2]. Confidentiality and integrity of customers' data is essential, and providing secure remote computation is a major issue [3]. Secure remote computation acknowledges the risks inherent in the customer's data residing on, and being processed on a distant computer somewhere in the cloud which is owned and controlled by an untrusted organization. These issues have been published in a series of reports by Cloud Security Alliance (CSA) including [4, 5, 6].

To deal reliably with the issue, demand for accountability in the cloud is essential [7]. Accountability needs an accurate and complete audit log [8] that is produced by the logging system [9]. The logging system is therefore a significant mechanism in accountability solutions for mitigating risks associated with the issue [2, 10, 11, 12]. However, to date logging systems are not available that provide the necessary level of accountability, [2, 10, 12, 13, 14, 15] nor have such logging system solutions addressed the issue of system performance improvement. This paper provides these considerations.

In software development, performance testing is an important aspect [16], as is performance of logging systems in the cloud [11, 12]. However, in the complex visualization infrastructure of the cloud, approaches to measurement and improvement of logging system performance are uniquely challenging, yet essential. Poor performance in any major

aspect of an application is unacceptable, and detracts from the reputation of any organization wishing to be considered as a trustworthy organization, and of their software assets [16].

This paper reports on our work following our previous work in the same area [12]. Experiments to improve logging system performance were performed to test increasing virtual CPU core sizes on a cloud hosted virtual machine. We refer in our work to these hosted virtual machines that host the logging systems, as dom0, each of which has a virtual CPU core that can be dynamically created, destroyed and resized. Multi-core processors are essentially operating system controlled virtual processors or computational engines running on a single physical device [17].

This paper refers to each multi-core processor as a CPU core. Performance refers particularly to the accuracy of the logging system in logging and recording file accesses. To improve the accuracy, one of the future research directions suggested in our previous paper [12] was the doubling of the number of CPU cores in dom0. This paper continues with that suggestion to test if, and by how much, increasing the number of CPU cores in dom0, that hosts a logging system, can increase the performance of this system in capturing more, or all, of the required audit information.

**Summary of contributions:** Firstly, we designed and performed iterative testing on file access processing with the aim to improve logging system accuracy, based on and extending our previous work [12]. The outcomes of this testing included proving that, when the target information is in a cloud virtual machine (VM) with a logging system capturing this information, increasing the number of CPU cores (in dom0) increases the accuracy of the logging system. As well, we identified that the accuracy of the logging system could be stated in various ways, which we discuss at some length under Section 4.2 below. We are confident that our results contribute to the development of logging systems that are able to accurately and comprehensively capture the required logging information, ensuring greater confidence and trust in the security of the application system and file storage in the cloud.

## 2. Performance Concerns of Logging Systems in the Cloud and the Experimental Environment.

### 2.1. Performance concerns of logging systems in the cloud.

Performance measurement of logging systems in complicated cloud environments is an important factor in supporting confidence and trust in the security of user data and processing in the cloud environment [10, 11, 12]. These concerns were discussed in our previous paper [12]. Briefly, one of four application system key performance indicators, or KPIs, is response time, which is the amount of time it takes for the application to respond to a user request [16]. Response time is the KPI which is the focus of this paper.

Response time is the representative KPI of performance measurement and improvement discussed for the purpose of encouraging stakeholders to be concerned with all the KPIs. Decisions on matters of security, trust and ways to create confidence in the cloud should now be a matter of concern to the larger group of stakeholders than just the cloud facilities provider. Technically, the paper focuses on the accuracy of logging systems that are located in dom0. Accuracy here means the accuracy of the logging system in accessing and capturing the target logged information from volatile memory in a target monitored VM (we call it domU), which is dependent, *inter alia*, on response time.

### 2.2. The experimental environment.

We configured both domU and dom0 first with 1 then with 4 CPU cores to test each configuration for valid response, and analyzed the results.

2.2.1. *In the target and monitored domU.* For the purposes of the experiments, the architecture in Figure 1 was used. This is based on the experimental architecture of our previous work [12], with some minor modifications. As Infrastructure as a Service (IaaS) cloud models can be a base to build Platform as a Service (PaaS) cloud models [18, 19], we designed the architecture to be a PaaS-like environment. Figure 1 shows the context of domU (the top right box in the figure) that can store the cloud customer's business files. A customer can upload a critical file (f.php in the figure) into diskU, which is domU's virtual disk. The file f.php represents a web application code file to be launched as a web application; the webApp in domU user level, to serve the customer's business purposes.

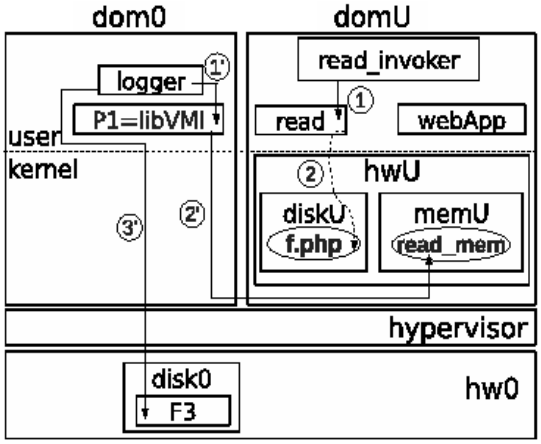


FIGURE 1. An experimental environment in domU and dom0

In the IaaS concept, [2] critical files are defined as files in diskU and are owned by the customer (domU's owners). These files can be of any file type such as text, graphic, executable, database files. They are the customer's asset and are valuable for the customer's business [2]. The customers require secure access and prevention of unauthorised access, as well as trust in the secure storage of the files, ensuring against loss or leakage of the files [2]. In PaaS, although customers do not own this domU, which is owned by the service provider, the customer still needs to upload their critical files to be stored in diskU to run their webApp for their businesses. The provider must provide a trusted secure environment for the customer's files.

Further, in Figure 1, 'read' (shown as a rectangle labeled 'read' in domU) represents any application and process that runs inside the domU user level. Without protection, attackers could run this application to read files; f.php for instance, for malicious purposes, although this file should only be legally read or accessed by webApp. The memory space allocated to the read process, shown as an ellipse titled read_mem, in a virtual main memory of domU/memU, is the memory space of the read process, which holds all target information we need to record for logging purposes. [2] calls this information 'history of a critical file' which includes file name 'f.php', a process Id and the process name of the read application, and the owner Id of the read process.

The history file can be used as evidence to mitigate risks associated with cloud security issues [2, 12]. One experiment set, indicated as 1 in the figure, tested the read_invoker which called the read application 1000 times. It was necessary to include a sleeping time in the read application to enable the logger to capture all information. Without this processing suspension the logger would not have time to record all of the information. The suspension time intervals ranged from 0 (i.e., without suspension) to 100ms. With the suspension time of 0ms, only the logger seems to be halted. At 65ms and above, all information was correctly captured.

2.2.2. *In dom0: the host of the logging system.* Dom0 is a VM which is owned and controlled by the cloud provider. The logger calls P1/libVMI (step 1' in Figure 1), which is a C language library used to obtain the process name and ID, and process owner Id of the read application in read_mem of the monitored domU described in Section 2.2.1 above (see step 2'). The logger then stores the captured information into F3/log files in disk0 (dom0's disk) in hw0 (dom0's physical machine). The goal is for the logger to capture all of this necessary information for each read of, in our example, f.php. For our testing purposes, we ran the read application 1000 times and demonstrated that the logger was able to collect the information each of the 1000 times.

3. **Experiment Description.** This is based on the experimental architecture of our previous work [12]. The aim of our experiments was to provide an environment in which a logging system could accurately and correctly capture the necessary information 100% of the time. We included in our processing a sleeping time period varied from 100ms down to approaching 0ms. Following our previous experiments [12], we defined a hit as occurring when a logger captures the file name of f.php as a string; 'f.php', and a miss is when the capture fails.

We extend the meaning of miss to include when the logging system generates an error. The accuracy of the system in each experiment set varied between 0% and 100%. Accuracy is defined as achieving a hit, so if the read operation is invoked 100 times, and in 80 of those times a hit is achieved, then we can state that the system has 80% accuracy. In the previous work, we found that as the sleeping time duration increased, the accuracy increased, until 100% accuracy was achieved at 65ms sleeping time, and above.

4. **Experimental Results and Discussion.**

4.1. **Increasing the number of CPU cores of dom0.** Increasing the number of CPU cores of dom0 that hosts a logging system increases the accuracy of the system. This is illustrated in Figure 2 and Figure 3.

From Figure 2, the thick line (d04c) and dotted line (d01c) represent the accuracy of the logging system in dom0 with 4 (d04c) and 1 (d01c) core(s) respectively. The accuracy of the system in d04c was always higher than the system in d01c. As can be seen in Figure 2, at 50ms the accuracy of the system with d04c (99.95%) is higher than the system in d01c

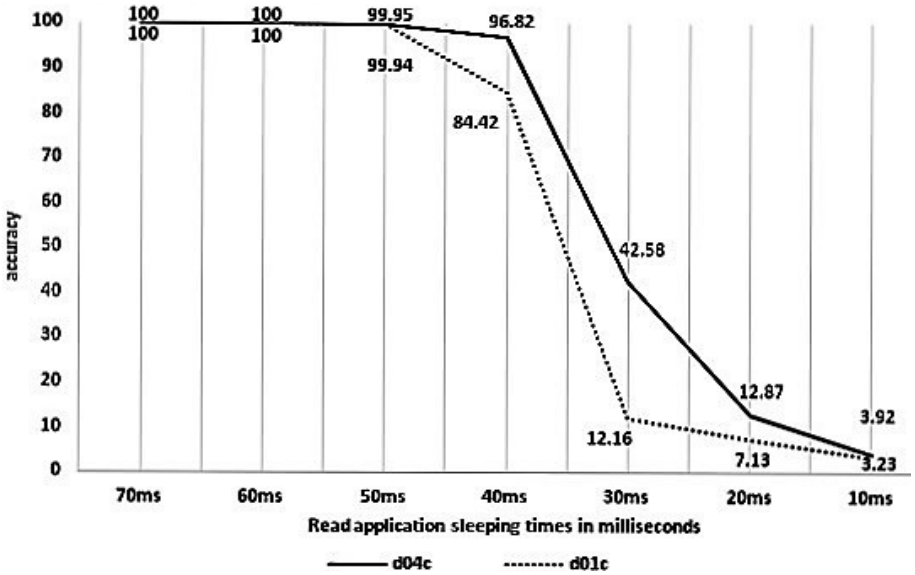


FIGURE 2. The accuracy of a logging system in dom0 with 1 core (d01c) and dom0 with 4 cores (d04c) in capturing *domU with 1 core (u1c)*

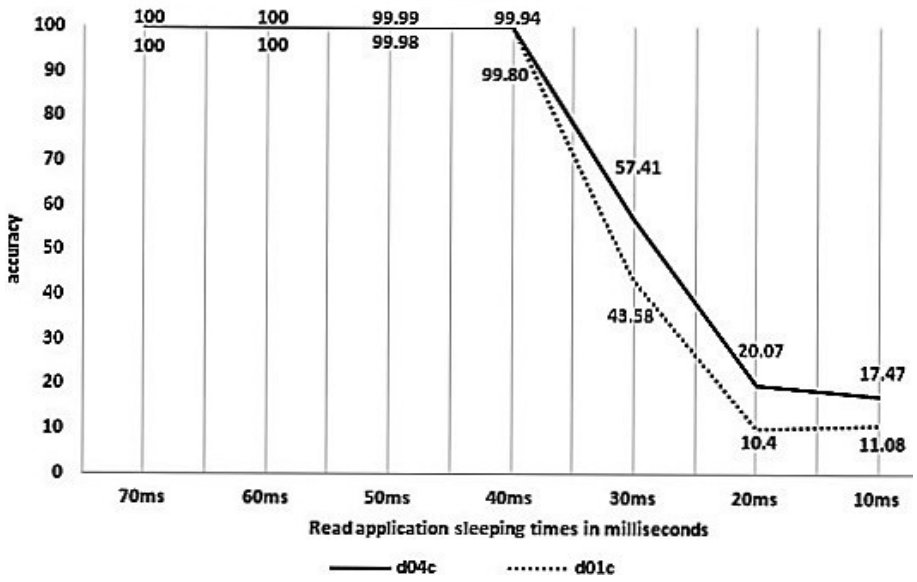


FIGURE 3. The accuracy of a logging system in dom0 with 1 core (d01c) and 4 cores (d04c) in capturing domU with 4 core (u4c)

(99.94%). At 40ms the accuracy of both systems begins to drop sharply until, at 10ms of sleeping time, both systems register less than 4% accuracy. At all times the system in d04c is higher than the system in d01c.

The data shown in Figure 3 further supports the fact that the accuracy of the system in d04c is always higher than the system in d01c. Comparing the accuracy of the systems in d04c and d01c, it can be seen that at 50ms the relative percentages are 99.99% and 99.98%, at 40ms are 99.94% and 99.80%, at 30ms are 57.41% and 43.58%, down to 10ms are 17.47% and 11.08%. Again it can be seen that increasing the number of CPU cores of dom0 from 1 to 4 increases the accuracy of a logging system. This implies that, to meet the appropriate or 100% accuracy of a logging system, designers of the system need to consider the number of dom0's CPU cores when designing the system.

4.2. **Increasing accuracy with more domU cores.** As shown in Figures 2, 3, and 4, increasing the number of domU cores increases the accuracy of the system. The increasing is explained in the following two aspects.

Firstly, an increasing number of domU's cores from 1 to 4 increases accuracy of a logging system, when the system is in d01c and in d04c and when it is capturing the target information from these domU. This is because the reasons in a) and b).

a) When the system is located in d01c, and is capturing the target information from this domU, an increasing number of domU's cores from 1 to 4 increases accuracy of a logging system. This is shown in Figure 2 and Figure 3, the dotted lines (d01c) represent the accuracy of a logging system that is in d01c and the system is capturing the target information from u1c (in Figure 2) and u4c (in Figure 3). The percentage of the accuracy of the system that is capturing u4c is always higher than when the system is capturing u1c. It can be seen that at 50ms the relative percentages are 99.98% (the dotted line in Figure 3) and 99.94% (the dotted line in Figure 2), at 40ms are 99.80% and 84.42%, at 30ms are 43.58% and 12.16%, down to 10ms are 11.08% and 3.23%.

b) In the situation where the system is located in d04c, and is capturing the target information from this domU, an increasing number of domU's cores from 1 to 4 also increases accuracy of a logging system. From Figure 2 and Figure 3, the thick lines (d04c) represent the accuracy of a logging system that is in dom0 with 4 cores (d04c), and the system is capturing the target information from u1c (in Figure 2) and u4c (in Figure 3). The percentage of the accuracy of the system that is capturing u4c is also

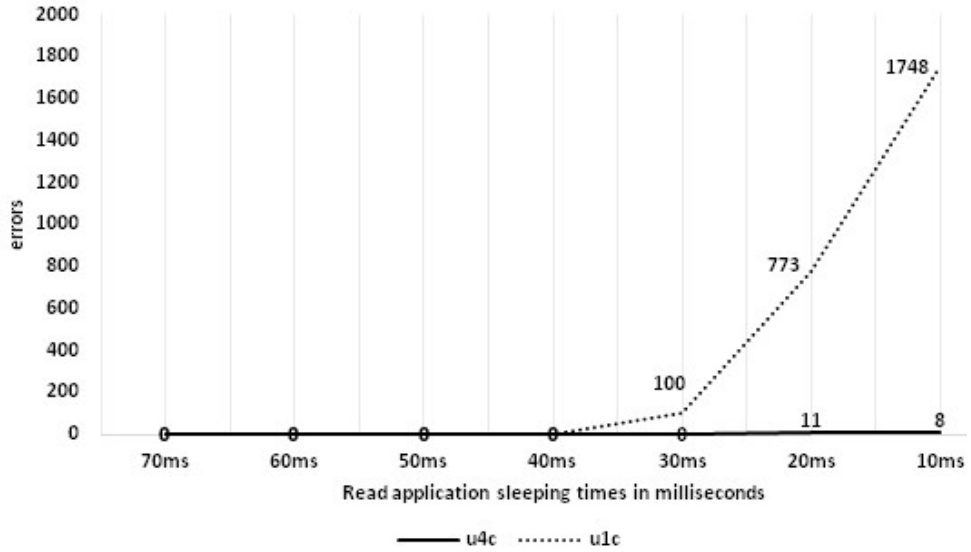


FIGURE 4. The accuracy of a logging system in d04c in capturing domU with 4 core (u4c) compared to when capturing u1c

always higher than when the system is capturing u1c. It can be seen that at 50ms the relative percentages are 99.99% (the thick line in Figure 3) and 99.95% (the thick line in Figure 2), at 40ms are 99.94% and 96.82%, at 30ms are 57.41% and 42.58%, down to 10ms are 17.47% and 3.92%.

Secondly, ideally, 100% accuracy is the target. The converse is 0 errors. Our tests have shown that accuracy increased, and error rate decreased when 4 cores were implemented. However, the aspect of speed of performance of the logging system must also be taken into consideration. In both cases of 1 core and 4 cores, 100% accuracy was achieved, but at the expense of processing performance time. Our results imply that by adding more cores, more than 4, that is, accuracy can still be achieved, error rates can still be reduced or eradicated, with greater performance speed. We did not test this thesis, however, due to the unavailability of appropriate configurations of multi-core processors.

We are able to show the results achieved with both 1 core and 4 core configurations. The number of errors that are generated by a logging system must be 0. An increasing number of domU cores from 1 to 4 reduces errors generated from a logging system that is capturing the target information from this domU. The reduction of error numbers is illustrated by Figure 4.

From the figure, the thick and dotted lines represent the error numbers of a logging system that is in d04c, and that is capturing the target information from u4c and u1c respectively. From the figure, the error number of a logging system capturing u4c is always lower than one capturing u1c. See at 30ms, the error number of the system capturing u4c (the thick line in the Figure 4) is 0 and always lower than one capturing from u1c (100). This is the same at 20ms back to 10ms as the following: $11 < 773$ and $8 < 1748$.

To sum up, from these two aspects, an increasing number of domU's CPU from 1 to 4 cores increases accuracy of a logging system. This implies that, to meet the appropriate accuracy of a logging system, the designers of the system need to consider domU's CPU cores when designing it, not only considerations of dom0's cores.

5. **Conclusions.** Based on a future work direction suggested in our previous paper and to improve the performance of cloud logging systems in terms of the systems accuracy, the objective of this current study was to design and test the effect on the performance of a logging system in a multiple core system, by testing an increased number of cores (from 1 to 4 cores). We were able to demonstrate that increasing the number of CPU cores in

a host virtual machine of a logging systems and of a target monitored VM increases the systems accuracy. Good accuracy was achieved partly by allowing brief system sleeping time to enable full logging to occur. An appropriately short sleeping time also depends on having an increased number of CPU cores.

Our testing was limited to a maximum of 4 cores. The first future research direction is warranted into the impact of multiple cores in excess of 4 which could achieve the optimum situation of 100% accuracy with 0ms sleeping time. The second one is to describe and achieve a situation of automatic balancing, customizing and optimizing dom0s cores and domUs cores. Finally, it is to undertake an investigation into the performance of a working logging system in a real world cloud application system environment.

**Acknowledgment.** We wish to thank Mr. Roy Morien of the Naresuan University Language Centre for his editing assistance and advice on English expression in this document.

## REFERENCES

[1] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka and J. Molina, Controlling data in the cloud: Outsourcing computation without outsourcing control, *Proc. of the ACM Workshop on Cloud Computing Security*, 2009.

[2] W. Wongthai, F. Rocha and A. van Moorsel, Logging solutions to mitigate risks associated with threats in infrastructure as a service cloud, *International Conference on Cloud Computing and Big Data*, 2013.

[3] V. Costan and S. Devadas, *Intel SGX Explained*, Tech. Rep., 2016.

[4] CSA, *The Treacherous 12: Cloud Computing Top Threats in 2016*, The Cloud Security Alliance (CSA), Tech. Rep., 2016.

[5] CSA, *The Notorious Nine: Cloud Computing Top Threats in 2013*, The Cloud Security Alliance (CSA), Tech. Rep., 2013.

[6] J. Archer, A. Boehme, D. Cullinane, P. Kurtz, N. Puhlmann and J. Reavis, *Top Threats to Cloud Computing, Version 1.0*, Tech. Rep., 2010.

[7] M. G. H. Niezen and W. M. P. Steijn, *Cloud's Social Implications and the Need for Accountability by Individual Cloud Users*, 2014.

[8] M. Felici, Cloud accountability: Glossary of terms and definitions, in *Accountability and Security in the Cloud*, pp.291-306, 2015.

[9] R. N. Heames and P. Sudhakar, Notice of violation of IEEE publication principles data accountability in cloud using reliable log files forwarding, *International Conference on Information Communication and Embedded Systems*, 2013.

[10] W. Wongthai, F. L. Rocha and A. van Moorsel, A generic logging template for infrastructure as a service cloud, *The 27th International Conference on Advanced Information Networking and Applications Workshops*, 2013.

[11] W. Wongthai and A. van Moorsel, Quality analysis of logging system components in the cloud, in *Lecture Notes in Electrical Engineering*, vol.376, pp.651-662, 2016.

[12] W. Wongthai and A. Van Moorsel, Performance measurement of logging systems in infrastructure as a service cloud, *ICIC Express Letters*, vol.10, no.2, pp.347-354, 2016.

[13] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang and B. S. Lee, Trustcloud: A framework for accountability and trust in cloud computing, *Proc. of the IEEE World Congress on Services*, pp.584-588, 2011.

[14] P. Macko, M. Chiarini and M. Seltzer, Collecting provenance via the Xen hypervisor, *The 3rd USENIX Workshop on the Theory and Practice of Provenance*, 2011.

[15] B. Payne, M. de Carbone and W. Lee, Secure and flexible monitoring of virtual machines, *The 23rd Annual Conference on Computer Security Applications*, 2007.

[16] I. Molyneaux, *The Art of Application Performance Testing: From Strategy to Tools*, 2014.

[17] T. W. Burger, *Intel Multi-core Processors: Quick Reference Guide*, http://www.devx.com/assets/intel/13623.pdf, 2005.

[18] CSA, *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0*, The Cloud Security Alliance (CSA), Tech. Rep., 2011.

[19] W. Dawoud, I. Takouna and C. Meinel, Infrastructure as a service security: Challenges and solutions, *International Conference on Informatics and Systems*, 2010.