

PERFORMANCE MEASUREMENT OF LOGGING SYSTEMS IN INFRASTRUCTURE AS A SERVICE CLOUD

WINAI WONGTHAI^{1,2} AND AAD VAN MOORSEL³

¹Department of Computer Science and Information Technology

²Research Center for Academic Excellence in Nonlinear Analysis and Optimization

Faculty of Science
Naresuan University
Phitsanulok 65000, Thailand
winaiw@nu.ac.th

³School of Computing Science

Newcastle University
NE1 7RU, United Kingdom
aad.vanmoorsel@ncl.ac.uk

Received July 2015; accepted September 2015

ABSTRACT. *The cloud offers computational resources to customers, such as networking, processing, and storage. Its flexibility and benefits of reducing IT costs are attractive to many companies. However, the cloud also brings with it security concerns which affect both cloud customers and providers. Accountability is one of the keys to mitigating risks associated with cloud security. A logging system is an important feature in accountability solutions to help in mitigating threats in the cloud. However, previous accountability approaches with logging system solutions have not provided system performance measurements. This paper provides these performance measurements for logging systems in the cloud. This can be a basis for clarifying the ability of the systems in capturing logging information. This clarification may be used as a guideline to efficiently and appropriately design, implement, and deploy logging systems. We argue that more research is needed on the topic of performance measurement of logging systems in complex and abstract cloud environments, and encourage other researchers to participate in providing solutions to meet these concerns. The result can be to truly enable logging systems to work in real world cloud production systems.*

Keywords: Cloud, IaaS, Accountability, Logging system, Performance measurement

1. Introduction. The cloud offers computational resources to customers, such as networking, processing, and storage [1]. Its flexibility and benefit of reducing IT costs are attractive to many companies [2]. It is arguably the future of computing and can potentially transform the IT industry in a wide variety of application areas [3]. This paper discusses only Infrastructure as a Service (IaaS) cloud model. IaaS is increasingly deployed in many areas such as in the remote provision of medical services [4]. Usually, customers need to upload their related files to IaaS such as in [4].

IaaS provides a base on which to build other cloud models including Platform as a Service (PaaS) or Software as a Service (SaaS) [5]. However, cloud security is one of five of the most major cybersecurity market trends which will define the investment of firms' cybersecurity budgets for 2015 [6]. The National Security Agency or NSA [7] also states that the complexity of security is much higher in a cloud environment where the data is distributed over a larger area and a greater number of devices. The Top Threats to Cloud Computing Report [8] by the Cloud Security Alliance or CSA illustrates seven examples of cloud security threats.

Accountability can be keys to mitigating the risks associated with the CSA top threats. Regarding accountability in the cloud, [3] argues that cloud behaviours can be inspected

by any party. Wongthai et al. [9] argues that logging systems can be an important feature in accountability solutions to mitigate the top threats. They also state that a logging system is composed of logging processes and log files. A logging process focuses on logging-related tasks, and log files store contents produced by these processes. However, previous accountability with logging system solutions [1, 9, 10, 11, 12] are provided without system's performance measurement or testing.

This paper provides performance measurement of logging systems. [13] argue that at the top level a key to dealing with one or more risks related to expense, opportunity costs, continuity, and/or corporate reputation is performance testing. Molyneaux [14] states that performance testing is a critical factor of the software development life cycle (SDLC). He also argues that non-performant applications normally do not deliver their intended profit to a company organization, thus, these applications make a net amount of time and capital, and a fall of credit from the application users. Then, they cannot be counted as reliable assets [14].

[15] states that software measurement requires a lot of practical advices to build upon experiences and to stop repeating failure. Ideally, effort and time commitments required for the deployment of monitoring and evaluation software (e.g., logging systems) for Internet applications should be minimal, even in the ever changing, dynamically growing, and continuously evolving behavior of Internet-based services [16]. The performance measurement can be one of the practical guidances that may minimize the effort and time commitments. Minimizing the commitments benefits the development of rapid logging systems which is an important aspect of the development of logging systems in the cloud, as discussed by [9]. Cloud architectures are more complex and abstract than a traditional client server model [7].

We argue that more research is needed on the topic of performance measurement of logging systems in complex and abstract cloud environments as also argued by [9]. Application performance measurement needs key performance indicators or KPIs that include availability, response time, throughput, and capacity [14]. The response time KPI is the amount of time it takes for the application to respond to a user request [14]. We discuss only the response time KPI as an example of performance measurement KPIs to encourage logging system development participants be concerned with all the system's KPIs.

Summary of contributions: The first contribution is the identification, classification and discussions of concerns in performance measurement of logging systems in the cloud. To the best of our knowledge, these discussions have not yet been described in the literature. The second contribution is the design and implementation of the performance measurement of a logging system in complex IaaS environments. The third and main contribution encompasses the results from the performance measurement activity and analysis and discussions of these results.

All these contributions can be a basis for clarifying the ability of logging systems in capturing the logged information. This clarification may be used as a guideline to efficiently and appropriately design, implement, and deploy logging systems. For example, when building a logging system, the measurement results in this paper could assist developers to achieve the correct and appropriate design of the logging system with minimal effort and time commitments. As a result, this can truly enable the logging systems to work in real world cloud production systems.

The remainder of this paper is as follows. Section 2 discusses logging systems to mitigate the risks associated with the threats and the performance concerns of the systems in the cloud. Then, Section 3 states the aim of the performance measurement and the experiment sets. Section 4 describes performance measurement environment. Then, Section 5 explains experiment design and running in dom0 and domU. Section 6 gives the result and its discussion. Finally, Section 7 provides a summary and future research directions.

2. Background.

2.1. **The proposed logging system to mitigate risks associated with the CSA threats in IaaS.** This system is in our previous work [1]. Its goal is to mitigate risks associated with CSA threat number one (malicious activities performed in cloud customer’s virtual machines/VMs or domUs) that can affect the security of both cloud customers and providers. Figure 1 shows the context of a domU for this system’s implementation. In the figure, Alice rents a Linux domU and has a critical file (s.txt) in diskU (domU’s virtual disk). Critical files are files in diskU and owned by customers (domU’s owners) [1]. These files can be any file type, such as text, executable, or database files. They are the customers’ asset and valuable for their businesses [1]. Thus, the customers do not want anyone to access these files apart from themselves and their authenticated users, and do not want loss or leakage of the files [1]. In the figure, read application (in the rectangle in user level) is an appU or an application that runs inside domU. The name ‘read’ is the appU’s name and also the process name of the appU. Alice can run this application to read s.txt (the dot-arrow line with a number 2). *Read_mem* (the ellipse in memU or a virtual main memory of domU) is the memory space of this read appU/process. This memory space holds all information we need to record. We call it history of a critical file which includes file name of s.txt, a process Id and a process name of read appU, and an owner Id of read process.

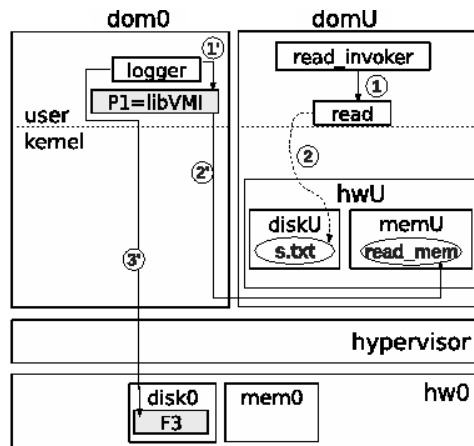


FIGURE 1. The system architecture of the proposed logging system

The aim of the proposed logging system in [1] is to record this history information, and then store it in a log file. The information can be evidence to enhance accountability in IaaS [1]. Main components of the proposed system in Figure 1 are logger, logging process (P1)/libVMI [17], and F3 as a log file. LibVMI is a C library that can read the memory space (*read_mem*) in memU from domU. From the figure, step 1’: the logger (in dom0 user level) is an app0 that calls libVMI to access memU (step 2’) to get the information in *read_mem* such as a file name of s.txt, as discussed above. Step 3’: the logger writes the information into F3.

2.2. **Performance concerns of logging systems in the cloud.** Measuring the performance of logging systems, we consider two main performance concerns of logging systems in the cloud: performance concerns of dom0 and of domUs. Firstly, we consider three factors of the performance concerns of dom0. The first factor is accuracy of the logging tasks. This is the accuracy of the logging tasks in dom0 in accessing the volatile memory in domU. In the context of logging system permanence measurement in the cloud, we consider this factor as a response time KPI. This paper presents only the measurement of this factor. Section 3 to Section 6 provide details of the accuracy of our proposed

logging system (Figure 1), how to measure it, the results, and discussions. The second one is log file size. [18] states that log files can grow at a relatively higher rate. They also argue that this issue may be mitigated by tiered storage and archival, de-duplication and summarization techniques. [19] also states that present hard disk capacities are measured in terabytes; thus, this should be a solution to the log file size explosion problem.

The last factor is the performance impact that is measured as a reduction in performance of dom0 caused by the logging systems. For example, this measurement can be achieved by measuring how much CPU of a machine is consumed by the logger process to achieve logging tasks, compared to a machine that does not deploy the logging systems. This measurement should be done when the full implementation of a logging system is ready, or after the accuracy of the logging processes in a particular logging system satisfies the accuracy requirement of the system. We experiment on only the measurement of the accuracy of the logging process in this paper. Secondly, performance concerns of domUs, a system need to access the main memory of domUs or memU. This may lead to a reduction in performance of domUs caused by the logging systems. Our proposed logging system reuses libVMI [17]. This tool imposes a minimal performance overhead to the target domU memory [12]. Thus, we do not measure this reduction.

3. The Aim of the Performance Measurement and the Experiment Sets. The aim of the measurement is to guarantee the proposed logging system (Figure 1) will yield 100% accuracy in capturing the information from memU in domU; how many milliseconds (ms) the read application or appU needs to be in memU after finishing reading a file such as s.txt before closing the file. The answer can be interpreted as that the proposed system in dom0 yields 100% accuracy in capturing the log information if an application in domU accesses a file for at least x ms. We consider this accuracy as a response time KPI because this KPI is the amount of time it takes for an application (the proposed system) to response to a user request. For experiment sets to calculate the accuracy, one set of experiments is as follows. (i) We set a particular sleeping time such as 80 ms for a read application after it finishes reading a file and before it closes the file. (ii) The read application will be run 1000 times to perform step (i). Then, (iii) the proposed system needs to capture log information (a file name string of a file that is read by read application) from all 1000 runs of the read application.

During a particular run of the read application to read s.txt, if the proposed system captures (from memU) the correct file name or the string 's.txt' that has been read by this application in this particular run, this capture is called a 'hit', otherwise a 'miss'. When a read application is run for 1000 times and the proposed system needs to capture all these 1000 times, then we can count a number of hits between 0 and 1000. For each experiment set or steps (i) to (iii), the accuracy (in percentage) is calculated by the number of hits divided by 10. Thus, the accuracy ranges from 0% to 100%. We run each experiment set for 10 times, then calculate the average of the accuracy of this experiment set. Thus, the average is calculated by the summation of the accuracy of each experiment set (in percentage) from the first time to the tenth time divided by 10.

4. Performance Measurement Environment. Firstly, application components, we use the same set of experiment environments as ones in the implementation of the proposed logging systems or Figure 1. In the figure, libVMI can obtain the names and IDs of all running processes in a *task_struct* circular list by accessing memU of a target domU. Each *task_struct* contains numerous variables to keep track of a process in the memory when it is executed by the CPU [20]. Then, the logger repetitively calls libVMI in a loop to check whether the list contains a process name called 'read' or not. However, this paper assumes that Alice runs read appU 1000 times separately, and each time the application reads s.txt. Ideally, the goal of the logger is that it has to capture the

information of these 1000 times of read appU activities. Secondly, hardware components, the experimental environment comprised a single physical machine as hw0. Its hardware configuration includes an Intel Core 2 Quad CPU Q9400 @ 2.66GHz x 4 (64-bit) CPU and 2.7 GiB of main memory. Hw0 consisted of hypervisor that used Xen 4.1.4 with a Fedora 16 dom0 running a 64-bit Linux kernel 3.6.11-4. Alice's domU is running a 64-bit Linux kernel 3.6.11-4 for Fedora 16. It consisted of 989.7 MiB of main memory. Networking of both dom0 and domU is disconnected.

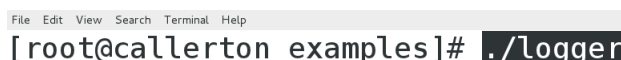
5. Experiment Design and Running.

5.1. Experiment design for domU. Firstly, setting up, Section 2 discussed the context of a domU in the implementation, see Figure 1 and the aim of the proposed system. DomU context and the aim of the logger in this measurement are exactly the same as in Section 2. The section shows the proposed system's architecture or Figure 1 which illustrates the experiment set-up for the accuracy measurement of the proposed system. It also provides an overview for both the logger and read application for this experiment. Secondly, read application routines, from the figure, read application usual routines are: 1) open s.txt file, 2) read the file and print the file's contents to a screen, 3) close the file, and 4) be terminated.

Lastly, read application routines with suspension and running the experiment, the routines above are modified to add suspending execution for microsecond intervals. The suspension is performed by C programming *usleep* function. The time intervals range from 0 to 100 ms. Read application routines with suspension are to: 1) open s.txt file, 2) read the file and print the file's contents to the screen, 3) **suspend or sleep for x ms such as x = 60**, 4) close the file, and 5) be terminated. This is one run of a read application. To run read application, all 1000 rounds of the read application with suspension run will be triggered by read_invoker which will be started only once, see Step 1 in the figure. This is one experiment set and we run 10 experiment sets for each sleeping time.

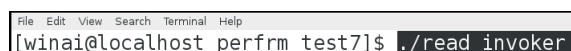
5.2. Experiment design for dom0. Firstly, setting up, the logger in this experiment has the same system architecture as the proposed system or Figure 1. In Figure 1, the main components of the proposed system in this experiment are described in Section 2, see Figure 1. The logger finds a read process in memU of the target domU by checking (using a loop) from the beginning to the end of the linked list of all running processes in the domU. Secondly, for this measurement we modified the logger in Figure 1 to be able to capture the log information of each run of read application that will be run for 1000 times. The logger in dom0 is run before read_invoker in domU as a monitoring tool. It is run only once to capture the log information from each one of all 1000 read application runs.

5.3. Running the experiment to collect accuracy of the logger. Section 5.1 to 5.2 have already discussed the design and set-up of the logger, and read and read_invoker applications to perform the accuracy measurement. Section 5.3 then explains how to collect hits and misses. To perform the experiment, the logger (Figure 2, the black



```
File Edit View Search Terminal Help
[root@callerton examples]# ./logger
```

FIGURE 2. To run the logger in dom0



```
File Edit View Search Terminal Help
[winai@localhost perfrm_test7]$ ./read_invoker
```

FIGURE 3. To run the read_invoker application in domU

highlight) will be started before read_invoker (Figure 3, the black highlight), and this is one experiment set.

To collect the average accuracy of a particular sleep time such as 100 ms, we run each experiment set for 10 times for each sleeping time setting, and then the average accuracy from these ten times is calculated and collected. The sleeping time intervals are set to vary from 100 to 0 to find the least time (in milliseconds) that the read application needs to be slept after finishing reading a file and before closing the file. This will be discussed in Section 6, along with the results.

6. Results for Accuracy of the Proposed System and Discussions.

6.1. The results. The results show that the optimum sleep time is 65 ms for the read application. This implies that the accuracy of the proposed system is 100% when any application in domU accesses (opening a file until closing the file) a file for at least 65 ms, see the graph in Figure 4. From the graph, when the sleeping times are from 80 to 65 ms (x-axis), the accuracy is 100% (y-axis). However, the first time that the accuracy is less than 100% (99.98%) is when the sleeping time is 64 ms. Thus, the minimum least sleeping time for the accuracy to be 100% is 65 ms, see the dotted line. The accuracy decreases as the sleeping time moves from 64 until 59 ms. The accuracy in this interval is only marginally different. It decreases from 99.99%, to 99.95%. To improve the accuracy, the logger process can be run in dedicated CPUs. If we double the CPU number, the sleeping time needed to get 100% accuracy should be decreased by half such as from 65 ms to 32.5 ms.

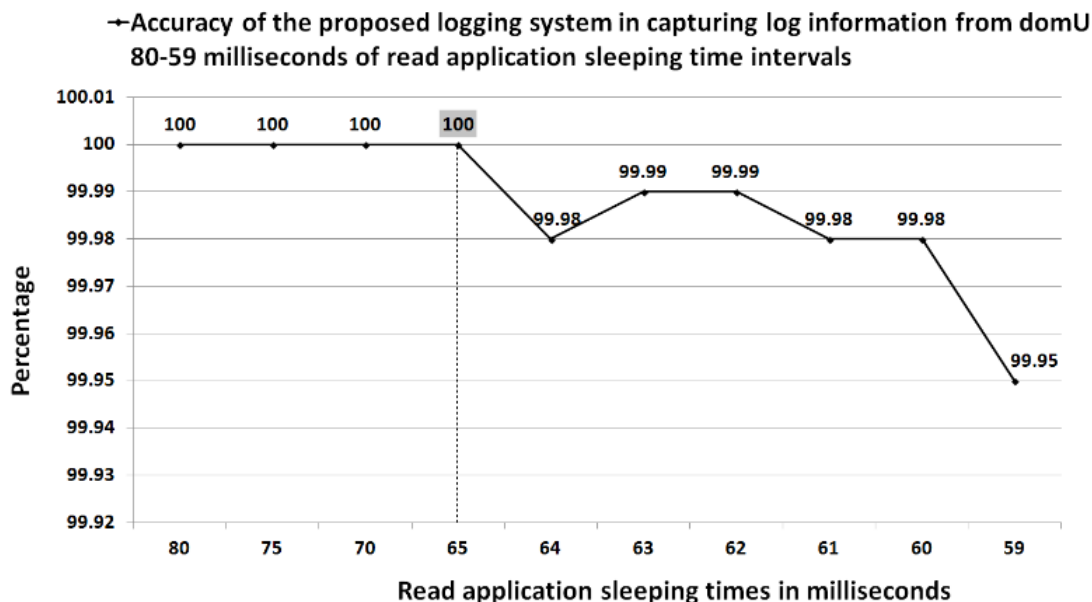


FIGURE 4. A graph showing the accuracy of the proposed logging system in capturing log information from domU

6.2. Decreasing trends of the accuracy of the logger. The section presents the decreasing trends of the accuracy when the sleeping time is set from 100 to 0 ms with the reduction of 10 ms each time. These trends can be interpreted as how the accuracy is dropped after 100% and at what sleeping time the logger does not work and is halted. In order to present the decreasing trends of the accuracy, we set up the sleeping time intervals from 100 to 0 ms with the reduction of 10 ms for each experiment set. Figure 5 presents the decreasing trends of the accuracy for the proposed system in capturing log information from domU.

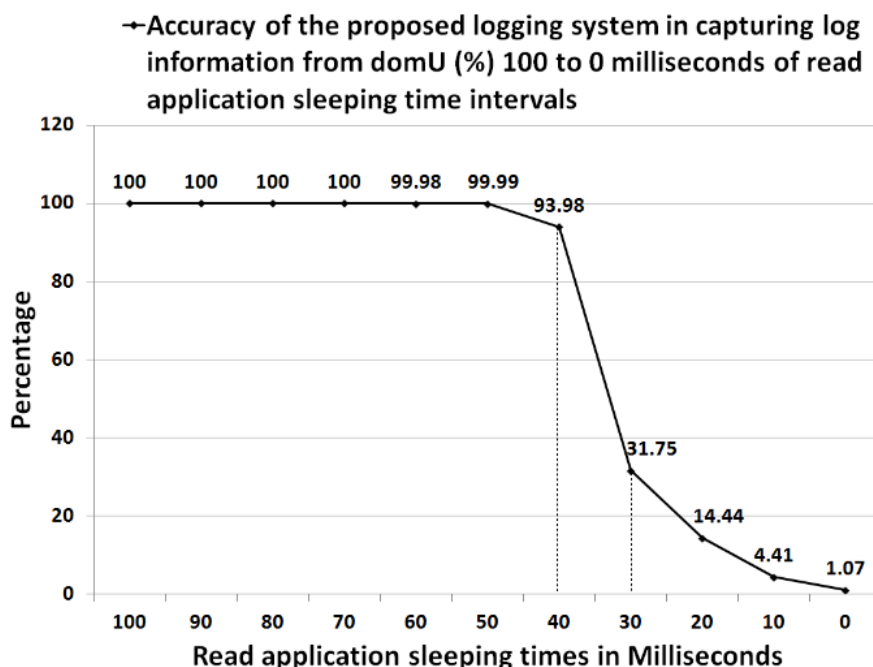


FIGURE 5. A graph showing the decreasing trend of the accuracy for the proposed logging system in capturing log information from domU

From the graph, the logger yields 100% of the accuracy when the read application sleeping time is from 100 to 70 ms. The accuracy is 99.98%, 99.99%, and, 93.98% when the sleeping times are 60, 50, and 40 ms respectively. The accuracy sharply decreases from 93.98% to 31.75% (see the dotted lines) when the sleeping time is changed from 40 to 30 ms. From the experiment, starting from the 30 ms until 0 millisecond, the logger seems to be halted. This may be because that the loop in the logger, used to traverse the processes list, is not fast enough. Thus, it should be possible to increase the speed of the loop by running the logger on dedicated CPUs in a multi-core system.

7. Summary. This paper provides the performance measurements of our proposed logging system in terms of its accuracy in capturing logging information from a target virtual machine or domU. The system has 100% accuracy when an application in a target domU accesses a file for at least 65 ms. To improve the accuracy, the logger process can be run via dedicated CPUs. A benchmark of accessing times to access a file can be how many ms that an application spends to access, open, read, write, and close its database file. This benchmark can be difficult to measure. This may be because the benchmark measurement can be heavily related to the hardware specifics of the domUs which can be located in different machines. Hardware specific qualities of these machines can differ from machine to machine. This specification can be the storage medium of the s.txt files; a solid state drive or SSD, or hard disk drive or HDD; the amount of main memory; the system load; size of the file; the file system in use (e.g., Linux third extended file system or ext3, or Microsoft Windows NTFS), CPU speed and so on.

We did not find such benchmark to compare with the minimum sleeping time for the measurement result which is 65 ms. Thus, we cannot compare the results to a benchmark of the average time taken for an application to access a file. However, the results in this paper can be a basis to clarify the ability of the logger in capturing the log information. This clarification can be used as a guideline to efficiently and appropriately design, implement, and deploy logging systems in the cloud. As a result, this can truly enable the logging systems to work in real world production systems. Lastly future research directions can be i) to implement the approaches of the dedicated CPUs and doubling the CPU

number and ii) performance measurement of logging systems in the cloud environments regarding the other KPIs including availability, throughput, and capacity.

Acknowledgment. Many thanks go to Mr. Roy Morien of the Naresuan University Language Center for his editing assistance and advice on English expression in this document.

REFERENCES

- [1] 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 (CloudCom-Asia)*, pp.163-170, 2013.
- [2] 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 2009 ACM Workshop on Cloud Computing Security*, pp.85-90, 2009.
- [3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, A view of cloud computing, *Commun. ACM*, vol.53, pp.50-58, 2010.
- [4] R. de Paris, *FReMI – A Middleware to Handle Molecular Docking Simulations of Fully-Flexible Receptor Model in HPC Environment*, Master Thesis, 2012.
- [5] W. Dawoud, I. Takouna and C. Meinel, Infrastructure as a service security: Challenges and solutions, *International Conference on Informatics and Systems*, pp.1-8, 2010.
- [6] Y. Leitersdorf and O. Schreiber, Cybersecurity hindsight and a look ahead at 2015, *TechCrunch*, 2014.
- [7] The National Security Agency (NSA), Cloud security considerations, *Tech. Rep.*, 2013.
- [8] CSA, Top threats to cloud computing, version 1.0, *Tech. Rep.*, 2010.
- [9] W. Wongthai, F. L. Rocha and A. van Moorsel, A generic logging template for infrastructure as a service cloud, *Proc. of the 27th International Conference on Advanced Information Networking and Applications Workshops*, pp.1153-1160, 2013.
- [10] R. K. L. 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, *Tech. Rep.*, pp.584-588, 2011.
- [11] 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.
- [12] B. Payne, M. de Carbone and W. Lee, Secure and flexible monitoring of virtual machines, *Proc. of the Annual Computer Security Applications Conference*, pp.385-397, 2007.
- [13] J. Meier, C. Farre, P. Bansode, S. Barber and D. Rea, *Performance Testing Guidance for Web Applications: Patterns & Practices*, Microsoft Press, 2007.
- [14] I. Molyneaux, *The Art of Application Performance Testing: From Strategy to Tools*, 2nd Edition, O'Reilly Media, 2014.
- [15] C. Ebert, R. Dumke, M. Bundschuh and A. Schmietendorf, *Best Practices in Software Measurement: How to Use Metrics to Improve Project and Process Performance*, 2005.
- [16] S. E. Parkin and G. Morgan, Toward reusable SLA monitoring capabilities, *Software Practice and Experience*, vol.42, no.3, pp.261-280, 2012.
- [17] B. Payne, About the VMI tools project, *Google Project Hosting*, 2013.
- [18] R. K. L. Ko, P. Jagadpramana and B. S. Lee, Flogger: A file-centric logger for monitoring file access and transfers within cloud computing environments, *Proc. of the 2011 IEEE the 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp.765-771, 2011.
- [19] A. Haeberlen, P. Aditya, R. Rodrigues and P. Druschel, Accountable virtual machines, *Proc. of the USENIX Conference on Operating Systems Design and Implementation*, 2010.
- [20] R. Love, *Linux Kernel Development*, 3rd Edition, Addison-Wesley Professional, 2010.