Leveraging Mobile Agents for Log Analysis in Data Center Networks

Zhifeng Xiao[1]* and Yun Wang[2]
[1]Penn State Erie, The Behrend College, Erie, Pennsylvania, USA
zux2@psu.edu
[2]Bradley University, Peoria, Illinois, USA
ywang2@bradley.edu

Abstract

Recent advances have witnessed the success of Data Center Network (DCN). In this paper, we propose a novel approach for distributed log analysis in DCN. Our method leverages Mobile Agents (MAs) to perform log analysis. MAs are able to travel across the network, fulfill their tasks on the host machines, and report to the system operator. Compared to the traditional centralized method, MA-based log analysis takes advantage of the processing ability of distributed machines, offloading log analysis from the central node. We evaluate both centralized method and MA-based log analysis in Emulab to process log files of a cluster of Apache HTTP servers, the result shows that the MA-based method outperforms the centralized method in terms of efficiency, scalability, and bandwidth consumption.

Keywords: Mobile Agent, Log Analysis, Data Center Network

1 Introduction

A modern data center consists of thousands of machines connecting to each other to form a DCN. This network structure requires significant bandwidth to perform data intensive tasks. Logging is an important supporting technique in DCN. The log files are the raw data to record system events, which can be combined together to learn the underlying behavior of the entire system. However, for a DCN, it is by no means a simple job to perform log analysis for the purpose of network forensics. Some typical problems within a DCN may include: (1) a software bug leads to the failure of job execution; (2) servers return bad results to the clients; (3) malicious virtual machines launch a DOS attack attempting to take down a web server; (4) hosts behave arbitrarily (i.e., the Byzantine Fault [17]).

There are two basic approaches to process log files for traditional distributed systems: the centralized method and the distributed method, which are described in Fig. 1. With centralized method, each machine sends local log files to a central node which performs log analysis. With distributed method, each machine processes log files locally and then sends the result to the central node for aggregation or further analysis. We argue that both approaches hardly meet the requirements of log analysis for DCN because of the following challenges:

- **Large log data volume** - Some applications running on top of a DCN generate or process tremendous amount of data. Thus, the size of log files grows drastically. For instance, a Hadoop cluster of 2000 nodes will generate 6 MB of log data per second, which makes around 520 GB of log data per day.

*Corresponding author: Penn State Erie, The behrend College, School of Engineering, 242 Burke Center, 5101 Jordan Road, Erie, PA 16563, USA, Tel: +1-814-898-6252, Web: `http://cs.bd.psu.edu/~zux2/`

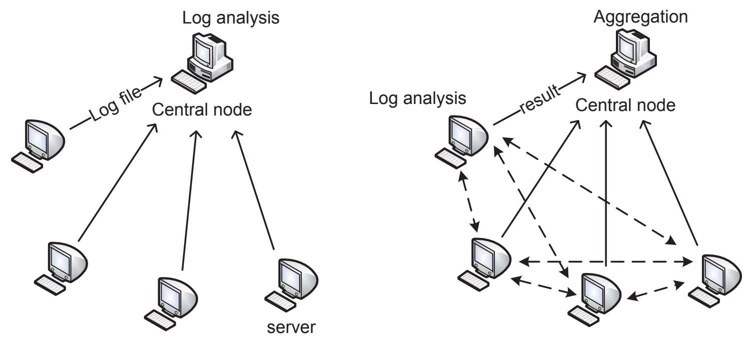


Figure 1: Two Log Analysis Approaches

- **Bandwidth** - Another side effect caused by data intensive applications is bandwidth consumption. For example, MapReduce [9] consumes a lot of bandwidth due to the nature of parallel computation. As for a DCN, bandwidth is even more precious due to the existence of the under-provision problem [2]. Therefore, it is not affordable to perform log analysis in a bandwidth-consuming pattern.

- **Efficiency** - Given the large number of machines and tremendous amount of log data, it is challenging to provide responsive feedback when a time-critical task is issued.

- **Scalability** - Log analysis should be scalable enough for potential expansion of a DCN.

Table 1: Comparison of Two Approaches

	Centralized method	Distributed method
Pros	Global knowledge	1. Efficient 2. Scalable
Cons	1. Bandwidth-consuming 2. Inefficient 3. Not scalable 4. Performance degradation	1. No global information 2. Log processing program is installed on each host, i.e., inflexible.

Table 1 compares the two methods in the DCN environment. It indicates that the centralized log analysis approach is no longer applicable for DCN because, 1) DCN applications need sufficient bandwidth to perform data-intensive tasks; 2) the log file size can be very large, hence it will be bandwidth-consuming to just move them from one machine to another; 3) the central node is overloaded in performing log analysis, which keeps it from efficiently completing time-critical tasks; 4) it does not scale well. Compared to the centralized method, the distributed log analysis is more suitable due to its efficiency and scalability. However, log processing program is installed on each machine. If a new log analyzer is in need, it will be expensive to have it deployed in thousands of machines for another application. Therefore, the distributed approach is not customizable.

To address the above shortcomings, we propose a MA-based log analysis technique for DCN. MAs carrying certain tasks will be sent to a group of machines to process and analyze log files. With the assistance of MAs, the system overhead will be greatly reduced since major processing work is done locally, and bandwidth can be reserved for DCN applications. Meanwhile, the processing speed can be improved due to the benefit of parallel computation. In addition, MAs do not work alone; instead, they exchange information between each other or obtain global knowledge from the coordinator. When MAs discover any interesting events, they notify the coordinator (i.e., the system operator), who will decide to terminate the mission or keep searching for something else.

This paper makes the following contributions:

- First, we propose a MA-based approach for distributed log analysis, and apply this technique to the data center network. To our best knowledge, this is the first time the MA technique is applied to log analysis. The proposed method well bridges the gap between bandwidth limitation and flexible maintenance in DCN.
- Second, we present a detailed case study to show how MAs can deal with log analysis for the server farm of Wikipedia.
- Third, we build a prototype of the system, and conduct a series of experiments on the Emulab platform [4] . The evaluation results show the effectiveness of our approach in terms of mission running time and aggregate traffic volume.

The rest of this paper is organized as follows. We present the related work in section 2, and describe the system design in section 3. Section 4 presents a case study. At last, we report our evaluation results in section 5, and draw conclusion in section 6.

2 Related Work

As our study covers multiple techniques, we present the related works from four aspects, including mobile agent techniques, logging, log analysis, and DCN.

2.1 Mobile Agent

A MA is a composition of computer software and data which is able to migrate from one computer to another autonomously and continue its execution on the destination computer [3]. The MA technique has been studied for over a decade. Researchers have developed successful MA frameworks such as Java Agent Development Platform (JADE) [23] [8] [21]. In addition, a reference model for MAs is also studied [22].

MA techniques have been applied to distributed network forensics [23] [18], real time IP traceback [5], intrusion detection [19], and DDoS attack traceback [6]. The prior studies show that the MA technique is suitable for a variety of applications due to its flexibility, mobility, and efficiency. By far, no prior works employ MAs for distributed log analysis. We argue that MA-based log analysis is able to show its maximal strength in the DCN context, because DCN is in need of a bandwidth efficient, protocol independent, scalable scheme for log analysis, while MAs can meet all of these requirements.

2.2 Logging

Logging is a crucial technique for system management and diagnostic, as log files provide a rich source of event history for a system's past activities. For example, UNIX provides logging service by running a general-purpose daemon process, e.g., syslog, which is configurable to achieve various levels of logging. Recent advances further demonstrate the importance of logging. Dunlap et al. propose to do logging below the virtual machine, allowing to replay system's execution before, during, and after an intruder compromises the system [10]. Kathiresshan et al. present a novel logging framework called EagleEye [20] for accountable distributed systems; instead of generating log files from host applications, EagleEye selectively captures interesting events from incoming and outgoing traffic to produce log entries. In a distributed computing environment, multiple approaches are developed for generating log files. In [13], the authors suggest to generate logs in a separate host which receives messages from other hosts within the cluster; this method is not applicable to DCN since it is bandwidth-consuming. Feild et al. propose CrowdLogging [11], a distributed search log collection, storage, and mining framework that preserves user privacy.

For large scale distributed systems, log files can be massive and can pose a huge barrier for efficient log analysis such as real-time analysis of performance problem [14, 16, 15]. In [14], the authors introduce a lightweight approach called NetLogger that generates log files only consisting of key information of events or interesting events. NetLogger reduces the volume of log files by a few orders of magnitude. In [7], the authors propose Chukwa, a monitoring system for large scale distributed systems; Chukwa enables logging in a data-intensive computing environment, thus it facilitates the logging needs for Hadoop. We argue that reducing the size of log files may not be feasible in some contexts, especially for data intensive tasks in a DCN. To this end, new log analysis techniques are desired.

2.3 Log Analysis

Visualization is an effective technique for log analysis to demonstrate system behavior in a intuitive way. Picviz [27] is a tool for spotting network attacks based on log analysis. Picviz enables network administrators to visualize the information in log files and exposes attacks that have been hidden. Tan et al. present a centralized approach named SALSA [26] which collects log files from all nodes in a distributed system. SALSA is capable of producing the state-machine views of the history of systems' executions along with related statistics, from the log files.

Machine learning and time series approaches are also used for log analysis, typically for the purpose of predicting system failure events. Standard machine learning techniques are applied, including Bayes networks, Hidden Markov models, and Partially Observable Markov Decision Process [24] [28]. Time series approach overcomes the insufficiency of involving a single message in a log file in system failure prediction. In [13], an approach using Support Vector Machines is introduced for system failure prediction based on system log files. Machine learning techniques can also be applied in performance analysis. For instance, CLUEBOX [25] is able to distill log files and troubleshoot system problems in an automated way.

Approaches have emerged to fit log analysis in large-scale distributed computing systems such as large clusters [16], in which the volume of log files can be incredibly huge. In [15], the authors propose a distributed approach for troubleshooting system failures; the proposed approach consists of two components, i.e., a NetLogger log summarizer that processes log files collected in the local machine, and an anomaly detector which collects, and then processes the results of the log summarizer and identifies the cause of a failure. We argue that the proposed approach in [15] is essentially a distributed log analysis method, hence it still has the weaknesses discussed before.

3 System Design

Before presenting the system design, we first specify the assumptions.

- We assume that the log file has already existed in the host machine. We do not focus on how to do logging in this paper.
- We assume the log file itself is trustworthy, meaning that it will not be modified after it is generated and stored in the file system. This is a reasonable assumption since there are tamper-evident logging techniques [17] that can be employed to ensure a trustworthy log file.

3.1 System Architecture

Fig. 2 depicts the architecture of the MA-based log analysis framework. There are five roles involved.

- A client uses a service provided by the DCN (e.g., a cloud service).
- A DCN operator manages the entire DCN. Both clients and the DCN operator can request the coordinator to send MAs for the purpose of log analysis. A client can not ask for sending MAs to hosts that are not assigned to her. A DCN operator, on the other hand, can request to send MAs to the entire DCN.
- The coordinator deals with MA management, including sending MAs, interacting with MAs, and revoking MAs.
- A MA is issued by the coordinator who also assigns tasks to a MA. Depending on the task, the coordinator can choose the number of MAs to be sent. Meanwhile, a MA can travel across the network, moving from one host to another to do investigation.
- A host is a node in the network. For a DCN, there may be thousands of hosts.

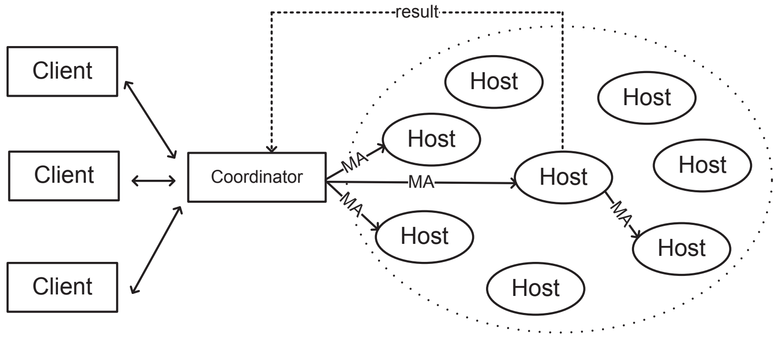


Figure 2: System Architecture

3.2 Building Blocks

3.2.1 Coordinator

The coordinator is responsible for the following jobs: (1) creating MAs with tasks, (2) sending MAs to the DCN, (3) collecting results, (4) processing results, and (5) revoking MAs.

3.2.2 Mobile Agent

A MA is a composite of software and data. It is able to migrate from one computer to another and continue its execution. The life cycle of a MA can be described in Fig. 3. When the coordinator receives a mission, it creates a MA (or multiple MAs). A MA without task and data is called an idle MA, which is only able to perform the basic functionality including migration and communication. Once assigned with a task, a MA is ready to start off. A MA will reach the first host. There will be an authentication process due to security concerns. After successfully authenticated, a MA can start its investigation by inspecting the local log files. Meanwhile, a MA will communicate with the coordinator for information exchange. For instance, MA will report any suspicious events to the coordinator; also, the coordinator may require the MA to update the current task (i.e., evolving). Once a MA completes the task, it will either move to the next host or be revoked depending on the mission needs.

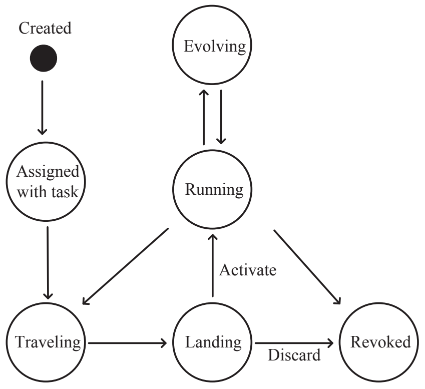


Figure 3: MA Life Cycle

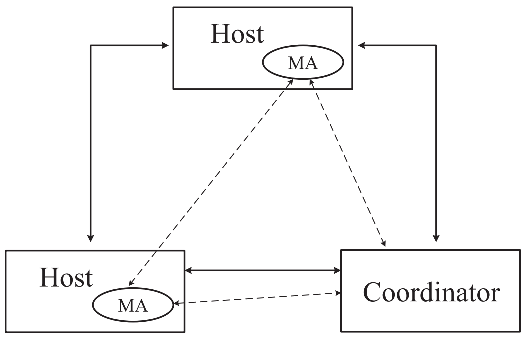


Figure 4: System Communication

3.2.3 Task

A task is the mission performed by a MA in the destination host. A task specifies the following information.

- The log file(s) that will be processed - For security purpose, a MA is enabled with least privilege, meaning that it can only read the specified log file(s).
- The action that a MA will perform - There are simple actions and complex actions. A simple action means pattern detection. In other words, the MA searches the log file for certain patterns that are predefined. These patterns reflect the events that are interested by the coordinator. A complex action means that a MA will perform more complicated tasks such as machine learning tasks. To this end, a MA will get trained at the coordinator, carrying the feature vector and predictive model. Once landing on the host, the MA feeds log files as input to the predictive model.
- The result - Depending on the purpose of the task, the output may be in different forms. For example, a compromised host may be detected since the MA found malicious behavior from the log file. Once the result is obtained, the agent will report to the coordinator.

3.2.4 Host

A host keeps a daemon process that is responsible for communicating with the coordinator, receiving the incoming agents, and activating them after successful validation.

3.3 Communication

Communication exists between the following entities (see Fig. 4).

- Coordinator and Host - Before sending a MA to a host, a hand-shaking process will be done first to establish a secure channel.
- MA and coordinator - Needless to say, MAs need to report to the coordinator and receive command from the coordinator.
- MA and MA - For some tasks, MAs share information to work cooperatively. For example, to trace how a malicious program spread from a machine to another, MAs will exchange information they obtain from log files to recover the path of the malicious program.

- Host and Host - When a MA needs to migrate from one host to another, the two hosts will be connected first.

3.4 Fault Tolerance

MAs travel on a network that may be unreliable. During its life cycle, unpredictable failure may happen on the host or in communication channels. The longer a MA travels, the higher possibility it is for the MA to be in trouble. We consider two kinds of failure.

3.4.1 Failure I: MAs cannot reach the target host

There are two possible reasons leading to this failure. First, when a routing path is down, the coordinator and the target host are unable to establish a valid connection. A possible workaround is to setup a different path through another hub/router. The second reason would be host crash or irresponsive host; in either case, the coordinator will detect anomaly. If the target host crashes, it will be rebooted. If the host is totally irresponsive, then it needs to be reconfigured.

3.4.2 Failure II: Host crashes when a MA is running

Generally, MAs may travel across several hosts during its life cycle. If a host happens to crash with a MA working on it, all work that has been done by the MA may be lost. To prevent this, a MA will report its state to the coordinator periodically, including the current checkpoint and inspection result. Hence, even the host crashes later, the coordinator will be kept informative all the time. In addition, a new MA will be sent to continue the mission when the host restarts.

3.5 Security Considerations

Security is a fundamental concern for MA-based systems. Since a MA contains executable code, it is important to constrain its capability by applying the least privilege principle. According to the system architecture, we identify four types of attack patterns.

- MA-to-host: in this category, a MA is turned to be malicious and attempts to exploit the vulnerabilities in host or launch attacks against a host. For example, a malicious MA may conduct DoS attacks by exhausting the host system's resources such as bandwidth or CPU. Also, a malicious MA may disrupt services provided by the host, or even extract information for which it is not authorized to access.
- MA-to-MA: in this category, a malicious MA may pretend to be a benign one, and request sensitive data from other benign MAs. In addition, a malicious MA can launch a DoS attack against other MAs by sending spam.
- Host-to-MA: if a host becomes malicious, A host can pretend as a honest destination to a mobile agent and extract the sensitive log analysis results from the agents or give the agents some fake log files. The malicious host can also ignore agent service requests and introduce unacceptable delays for log analysis tasks. The malicious host can also perform eavesdropping. For instance, a malicious host not only monitors communications, but also monitors every instruction executed by the agent, all the unencrypted or public data it brings to the host, and all the subsequent data generated on the host.
- MA-to-coordinator: A malicious MA can send fake or inaccurate report to the coordinator to cover malicious behavior in the system.

- Compromised coordinator: if a coordinator is compromised, the entire log analyzer system will collapse.

4 A Case Study

In this case study, we demonstrate that how MAs can identify problems in Wikipedia servers cluster, which is comprised of hundreds of Linux servers. As of December 2009, there were 300 servers in Florida and 44 servers in Amsterdam. A Wikipedia server cluster consists of a variety types of servers, including Apache web servers, Squid cache servers, load balancers, tool servers and database servers, and so forth.

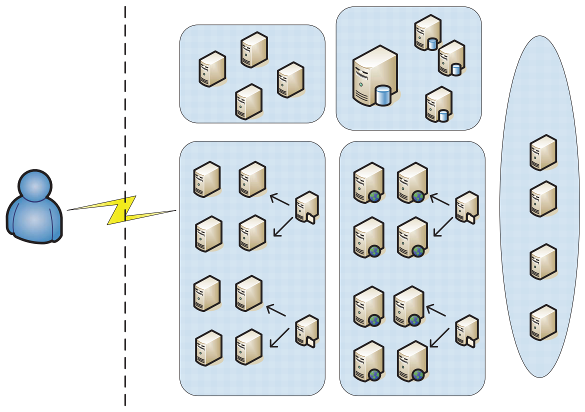


Figure 5: Wikipedia server farm

We describe a common problem a regular Wiki user may encounter: a user updates a page on Wikipedia, but fails to see her updates upon reloading the page. The possible cause is that one of the cache servers returned a stale page. However, it is not trivial to detect and fix this bug, since it is challenging to identify that which cache at which level is malfunctioning. Researchers have proposed a tracing system called X-trace [12] to detect the faulty cache. However, X-trace requires the client to install X-trace module (e.g., a Firefox add-on), which might be unrealistic for some users. Our proposed approach does not require any configuration on client side.

Now, we present how MAs work for this case. Assume that server A is an Apache web server, server B is a squid cache, server C is a database server, server D is a load balancing server. A maintains all web page resources (HTML file, images, flash, etc.). B stores the frequently requested pages in order to improve browsing speed and reduce the burden of other back-end servers. C manages a database management system. D forwards client requests to proper server for load balancing. To address the stale page problem, the coordinator sends several MAs to the cache server cluster. The attached task will include IP address, time stamp, and web page URL, all of which will be used to identify the proper log entries. According to the task setup, a MA manages to find a sequence of log entries whose pattern is (access $\rightarrow$ modify $\rightarrow$ access) other than (access $\rightarrow$ modify $\rightarrow$ update $\rightarrow$ access) on host A. The detected pattern can be used as evidence of the incident. Therefore, the MA can send a report to the coordinator to close the case.

5 Evaluation

In this section, we present the experiment results of our prototype by deploying the MA-based log analysis framework in a cluster of Apache HTTP servers on Emulab [4]. We focus on two performance metrics, i.e., job running time and network traffic load, to compare the centralized method to the MA-based method. In addition, since the two metrics vary with the change of system parameters, we have defined three factors, including log file size, task complexity, and node number, to obtain some insightful conclusion.

5.1 Experiment Setup

We consider an Apache server cluster, which handles HTTP requests from clients and records events in a local log file. There are two types log files defined in the raw Apache log files: error log and access log. The definition of each field can be found in Apache log file document [1].

5.1.1 Error Log

Error log is the place where Apache stores diagnosis information as well as any errors that it encounters in processing requests. It is the first place to look when a problem occurs with starting the server or with the operation of the server, since it will often contain details of what went wrong and how to fix it. An example of error message is shown as follows: *[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1] client denied by server configuration: /export/home/live/ap/htdocs/test.*

5.1.2 Access Log

The server access log records all requests processed by the server. Here is an example entry of access log: *127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache pb.gif HTTP/1.0" 200 2326.*

5.1.3 Experiment Parameters

We set up the parameters in Table 2 for our experiment. We design two tasks: Task 1 gives statistical results of the number of HTTP requests from each client. Task 2 simply duplicates the workload of task 1 by performing task 1 multiple times in order to increase the task complexity and consume more processing time.

Table 2: Experiment Parameters

Parameter	Value
Log file size (MB)	[0.5 - 2]
Task	Task 1 and Task 2
Cluster size	2, 4, 6, 8

5.2 Experiment Results

In the following section, we present the evaluation results. Our goal is to compare the performance of the centralized log analysis approach and the one of the MA-based approach in terms of two performance metrics, i.e., running time and aggregate traffic.

Fig. 6 compares the two approaches in terms of running time over log file size. For each test, the cluster will analyze the same log file(s) using the two different approaches. The log file sizes for the analysis tasks increase from 0.5 to 2.0 megabytes with an increment of 0.5 megabytes each time. As we can see in the figure, with the log file size increasing, the MA-based approach is more efficient than the centralized method. The reason is that MAs take advantage of the computational capabilities of working machines. Also, the MA-based method scales well, because the number of MAs is adjustable. In our test, we send a MA to each machine to minimize the running time. Fig. 7 shows how running time changes with different tasks. We conducted two tasks, in which task 2 requires more workload than task 1 in a cluster of three machines. From the result, we observe that the MA-based method is faster than the centralized method despite of the task. Fig. 8 shows the comparison of the two approaches over the cluster size. In our test, we gathered data from a cluster of size 2 to a cluster of size 8, with an increment of 2 nodes each time. We observe that the running time of centralized method increases almost linearly with the cluster size, while the speed of the MA-based approach remains stable when the cluster expands. This observation implies that the proposed approach is scalable and applicable to large scale network like DCN.

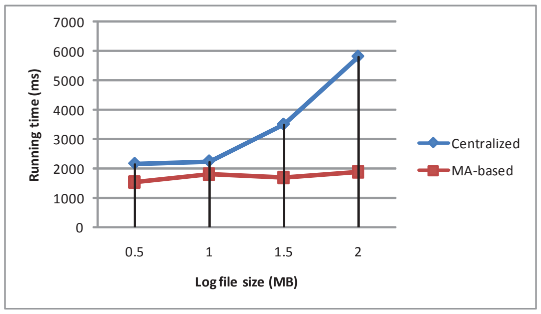


Figure 6: Running Time Vs. Log File Size

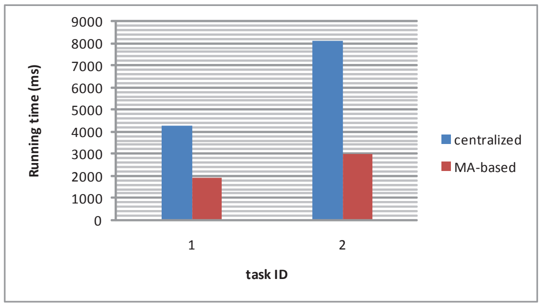


Figure 7: Running Time Vs. Different Tasks

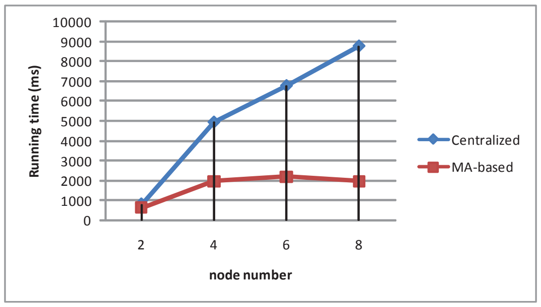


Figure 8: Running Time Vs. Cluster Size

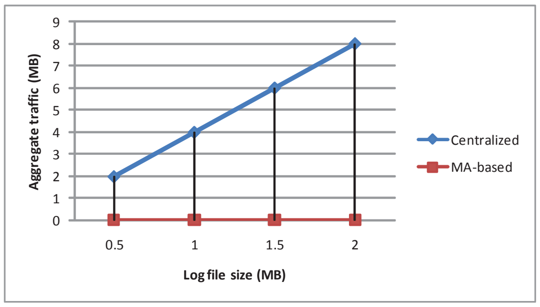


Figure 9: Aggregate Traffic Vs. Log File Size

In Fig. 9, we compare the two approaches in terms of aggregate traffic over log file size. The difference is obvious: the centralized method incurs a linear growth of traffic load with the increase of log file size, while the MA-based approach does not generate much of extra traffic when the log file size increases. This result matches the previous analysis, and further confirms the efficiency of our proposed approach. Fig. 10 compares the two approaches in terms of aggregate traffic with two different tasks. Despite of the complexity of the task, the MA-based approach generates far more less traffic than the centralized approach. Fig. 11 shows how aggregate traffic changes with the number of nodes (i.e., the cluster size). We can observe that the resulting aggregate traffic using the MA-base approach is always largely less than the one using the centralized approach. This result proves that the MA-based approach saves bandwidth, and has better scalability.

Finally, we can conclude that the MA-based approach is advantageous than the centralized method

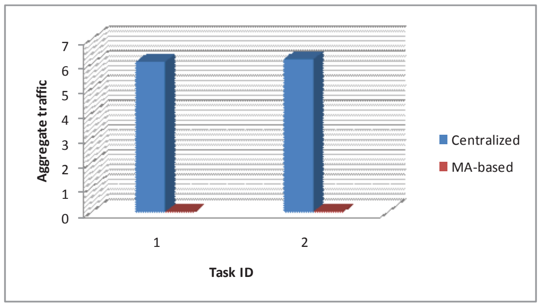


Figure 10: Aggregate Traffic Vs. Tasks

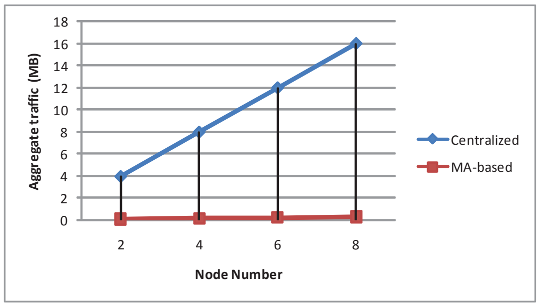


Figure 11: Aggregate Traffic Vs. Cluster Size

in terms of efficiency, traffic overhead, and scalability.

6 Conclusion and Future Work

Log analysis in DCN faces challenges. We propose to use a MA-based approach to replace traditional centralized log processing. MA is highly customizable, which brings flexibility when new log analysis tasks need to be deployed. With MAs, the workload of log processing can be distributed to host machines. Therefore, it requires less bandwidth and improves system scalability. We plan to extend this work in the following directions. First, despite of the merits of MAs, security of MAs is also an important concern; hence our next goal is to establish a security framework to ensure confidentiality, integrity, and availability of the log analysis service. Second, we plan to investigate the performance of the proposed approach on a larger network with real service co-hosted in the cluster; our goal would be to test the performance degradation when log analysis is applied.

References

[1] Apache log file documentation. `http://httpd.apache.org/docs/1.3/logs.html`. Accessed: 2012-05-06.

[2] Cisco data center infrastructure 2.5 design guide. `http://www.cisco.com/univercd/cc/td/doc/solu-tion/dcidg21.pdf`. Accessed: 2012-05-06.

[3] Definition of mobile agent. `http://en.wikipedia.org/wiki/Mobile_agent`. Accessed: 2012-05-06.

[4] The utah emulab network testbed. `http://www.emulab.net/`. Accessed: 2012-05-06.

[5] S. Armoogum and N. Mohamudally. Mobile agent based real time IP traceback. In *Proc. of the 3rd IEEE International Conference on Digital Information Management (ICDIM'08), London, UK*, pages 69–74. IEEE, November 2008.

[6] S. Armoogum and N. Mohamudally. Efficiency of using mobile agents to trace multiple sources of attack. In *Proc. of the 1st International Conference on Networked Digital Technologies (NDT'09), Ostrava, Czech Republic*, pages 464–468. IEEE, July 2009.

[7] J. Boulon, A. Konwinski, R. Qi, A. Rabkin, E. Yang, and M. Yang. Chukwa, a large-scale monitoring system. In *Proc. of the 24th international conference on Large installation system administration (LISA'10), San Jose, California, USA*. USENIX Association, November 2008.

[8] B. Chen, H. H. Cheng, and J. Palen. Mobile-C: a mobile agent platform for mobile C/C++ agents. *Software: Practice and Experience*, 36(15):1711–1733, 2006.

[9] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. *ACM SIGOPS Operating Systems Review*, 36(SI):211–224, 2002.

[11] H. A. Feild, J. Allan, and J. Glatt. Crowdlogging: distributed, private, and anonymous search logging. In *Proc. of the 34th international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR '11), Beijing, China*, pages 375–384. ACM, July 2011.

[12] R. Fonseca, G. Porter, R. H. Katz, S. Shenker, and I. Stoica. X-trace: A pervasive network tracing framework. In *Proc. of the 4th USENIX conference on Networked systems design & implementation (NSDI'07), Cambridge, Massachusetts, USA*, pages 20–20. USENIX Association, April 2007.

[13] E. W. Fulp, G. A. Fink, and J. N. Haack. Predicting computer system failures using support vector machines. In *Proc. of the 1st Workshop on the Analysis of System Logs (WASL'08), San Diego, California, USA*. USENIX Association, December 2008.

[14] D. Gunter and B. Tierney. NetLogger: A toolkit for distributed system performance tuning and debugging. In *Proc. of the 8th IFIP/IEEE 8th International Symposium on Integrated Network Management (IM'03), Colorado Springs, USA*, pages 97–100. IEEE, March 2003.

[15] D. Gunter, B. L. Tierney, A. Brown, M. Swany, J. Bresnahan, and J. M. Schopf. Log summarization and anomaly detection for troubleshooting distributed systems. In *Proc. of the 8th IEEE/ACM International Conference on Grid Computing, Austin, Texas*, pages 226–234. IEEE, September 2007.

[16] D. K. Gunter, B. L. Tierney, and S. J. Bailey. Scalable analysis of distributed workflow traces. In *Proc. of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'05), Las Vegas, Nevada, USA,*, volume 3, pages 849–855. CSREA Press, June 2005.

[17] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: Practical accountability for distributed systems. In *Proc. of the 21 ACM SIGOPS symposium on Operating systems principles (SOSP'07), Skamania Lodge, Stevenson, WA, USA*, pages 175–188. ACM, October 2007.

[18] B. W. Hoelz, C. G. Ralha, R. Geeverghese, and H. C. Junior. A cooperative multi-agent approach to computer forensics. In *Proc. of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'08), Sydney, NSW, Australia*, volume 2, pages 477–483. IEEE, December 2008.

[19] W. A. Jansen. Intrusion detection with mobile agents. *Computer Communications*, 25(15):1392–1401, 2002.

[20] N. Kathiresshan, Z. Xiao, and Y. Xiao. EagleEye: a logging framework for accountable distributed and networked systems. In *Proc. of the 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Shanghai, China*, pages 958–963. IEEE, April 2011.

[21] D. B. Lange and O. Mitsuru. *Programming and Deploying Java Mobile Agents Aglets*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[22] P. J. Modi, S. Mancoridis, W. M. Mongan, W. Regli, and I. Mayk. Towards a reference model for agent-based systems. In *Proc. of the 5th ACM international joint conference on Autonomous agents and multiagent systems (AAMAS'06), Hakodate, Japan*, pages 1475–1482. ACM, May 2006.

[23] A. Nagesh. Distributed network forensics using jade mobile agent framework. *Student Trade Show and Project Reports, Arizona State University*, pages 5–6, 2006.

[24] F. Salfner and S. Tschirpke. Error log processing for accurate failure prediction. In *Proc. of the 1st Workshop on the Analysis of System Logs (WASL'08), San Diego, California, USA*. USENIX Association, December 2008.

[25] S. R. Sandeep, M. Swapna, T. Niranjan, S. Susarla, and S. Nandi. Cluebox: A performance log analyzer for automated troubleshooting. In *Proc. of the 1st Workshop on the Analysis of System Logs (WASL'08), San Diego, California, USA*. USENIX Association, December 2008.

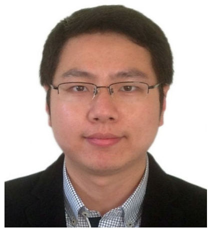
[26] J. Tan, X. Pan, S. Kavulya, R. Gandhi, and P. Narasimhan. SALSA: Analyzing logs as state machines. In *Proc. of the 1st Workshop on the Analysis of System Logs (WASL'08), San Diego, California, USA*. USENIX Association, December 2008.

[27] S. Tricaud. Picviz: Finding a needle in a haystack. In *Proc. of the 1st Workshop on the Analysis of System Logs (WASL'08), San Diego, California, USA*. USENIX Association, December 2008.

[28] Z. Xue, X. Dong, S. Ma, and W. Dong. A survey on failure prediction of large-scale server clusters. In *Proc. of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'07), Qingdao, China*, volume 2, pages 733–738. IEEE, July-August

2007.

Author Biography



Zhifeng Xiao is currently an Assistant Professor of Department Computer Science and Software Engineering at Penn State Erie, the Behrend College. Prior to that, he obtained the Ph.D. degree in Computer Science at the University of Alabama. He is broadly interested in cyber security. In particular, his research interests span the areas of computer and network accountability, cloud security, smart grid security, web security, and digital forensics. His publications have appeared in Journals such as IEEE Transactions on Smart Grid, IEEE Communications Magazine, International Journal of Security and Networks, IEEE Communications Surveys and Tutorials, etc. He is an IEEE member.

Yun Wang is currently an Assistant Professor in the Department of Computer Science and Information Systems at Bradley University, IL, USA (2011 – present). She was an Assistant Professor at the Southern Illinois University Edwardsville (2008 – 2011). She received the Ph.D. degree in Computer Science and Engineering at University of Cincinnati, Ohio, USA in 2008. Her research interests include Network Modeling and Analysis, Performance Evaluation and Optimization, Wireless ad hoc and Sensor Networks, Wireless Mesh Networks, Mobile Computing, and Security. Her publications have appeared in related international conference proceedings and journals such as IEEE Transactions on Mobile Computing and IEEE Transactions on Parallel and Distributed Systems.