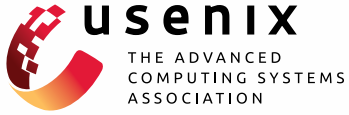


Towards Combining Online & Offline Management for Big Data Applications

Brian Laub, Chengwei Wang, Karsten Schwan, and Chad Huneycutt,
Georgia Institute of Technology

https://www.usenix.org/conference/icac14/technical-sessions/presentation/laub

This paper is included in the Proceedings of the 11th International Conference on Autonomic Computing (ICAC '14).

June 18–20, 2014 • Philadelphia, PA

ISBN 978-1-931971-11-9

Open access to the Proceedings of the 11th International Conference on Autonomic Computing (ICAC '14) is sponsored by USENIX.

Towards Combining Online & Offline Management for Big Data Applications

Brian Laub, Chengwei Wang, Karsten Schwan, Chad Huneycutt
College of Computing, Georgia Institute of Technology, Atlanta, GA 30318, USA
{*brian.laub, flinter, karsten.schwan, chadh*}*@cc.gatech.edu*

Abstract

Traditional data center monitoring systems focus on collecting basic metrics such as CPU and memory usage, in a centralized location, giving administrators a summary of global system health via a database of observations. Conversely, emerging research systems are focusing on scalable, distributed monitoring capable of quickly detecting and alerting administrators to anomalies. This paper outlines VStore, a system that seeks to combine fast online anomaly detection with offline storage and analysis of monitoring data. VStore can be used as a historical reference to help guide administrators towards quickly classifying and fixing anomalous behavior once a problem has been detected. We demonstrate this idea with a distributed big streaming data application, and explore three common fault scenarios in this application. We show that each scenario exhibits a slightly different monitoring history, which may be undetectable by online algorithms that are resource-constrained. We also offer a discussion of how historical data captured by VStore can be combined with online monitoring tools to improve troubleshooting efforts in the data center.

1 Introduction

Today's data center applications are increasingly concerned with fast and efficient processing of huge amounts of data. Web companies use data analytics to track and analyze user actions and provide real-time business decisions to improve user experience. Applications that store and process such vast amounts of data are no longer composed of single codes running on a few machines, but rather a complex set of many interconnected distributed systems, often across hundreds of machines in a data center. With cost savings and flexibility afforded by services like Amazon's EC2 and Elastic MapReduce [1, 2], these applications are now being deployed in virtualized environments as well.

As data scale increases and the push towards real-time data analytics continues, efficient monitoring and troubleshooting of these systems has become more important. Each component of a system may have different performance characteristics and failure scenarios. In traditional data centers, the scale of machines required to process data fast enough means hardware failures are common [10]. In cloud environments, performance can often be erratic due to virtualized resource contention with other cloud tenants [9]. To combat these issues, monitoring is often used as a tool for administrators to detect, diagnose, and fix failures quickly.

This paper outlines a method for combining fast and scalable monitoring of distributed applications with a scalable database backend for offline storage and analysis of monitoring data. Traditional monitoring applications use polling interfaces and centralized collection, providing a snapshot of overall system health via metrics collected over some period of time. However, these systems can have scalability issues, and lack fast anomaly detection required for real-time streaming data applications that are deeply ingrained in business logic. In our previous work, VScope [17], we developed a system that scales to thousands of machines and allows dynamic, real-time monitoring using online analytics for fast anomaly detection. In this paper we outline an extension to VScope, which we call *VStore*, that enables offline archiving of monitoring data gathered by VScope for historical analysis. This approach allows for online algorithms to be deployed dynamically to quickly detect problems, and to compare current observations to a rich set of historical data for targeted debugging.

The remainder of this paper is organized as follows: section 2 discusses problems with current solutions; section 3 outlines our system design; section 4 explores use cases from three common anomaly scenarios in a distributed weblog application, and discusses how VStore can aid troubleshooting efforts; and in section 5 we conclude and discuss future work.

2 Problem Description

Traditional data center monitoring systems such as such as Ganglia [5], Nagios [6], and OpenTSDB [8] provide for basic collection of simple metrics like CPU, memory, and network utilization from machines in a datacenter. These metrics offer a summary of global system health, but sometimes this view can be limited. In the case of Ganglia, for instance, a round-robin database limits the amount of data stored in a centralized place. OpenTSDB provides a scalable data storage backend, but still relies on simple polling of metrics. Although Nagios supports alerting, methods are typically based on heuristics such as pre-defined thresholds for each monitored metric.

Conversely, emerging methods for monitoring systems in the research literature such as Monalytics [15] and VScope [17] use a more targeted approach by aggregating data over a very large scale of machines, and performing online analysis for rapid detection of anomalies. These systems provide an opportunity to dynamically adjust the monitoring and analysis functions within the datacenter, but lack support for a rich history of monitoring data to compare against.

Each of these monitoring systems provides an invaluable set of information to data center administrators. System-wide monitoring applications provide a global overview of application and system health over longer time windows, whereas online algorithms provide extremely fast detection of anomalies and targeted debugging when problems arise. However, we lack a coherent bridge between these two paradigms. An administrator working on troubleshooting an issue in a complex distributed system would like to not only know *when* an anomaly is detected, but what *type* of anomaly the application is experiencing. Consider a distributed system like HBase, which is composed of other complex components (namely Hadoop and ZooKeeper). As noted in [14, 4], a number of different fault types can lead to irregular behavior in an HBase cluster. These range from a failing disk causing excessive I/O wait times at HDFS nodes, imbalanced key spaces causing ”hot spots” at region servers, or a failing NIC causing network requests to be delayed. Each scenario may exhibit subtly different anomalies at each tier, making classification difficult if data has been aggregated or summarized.

Bridging this gap will allow for faster detection and correction of faults in large, distributed systems. Anomalous behavior can first be detected with an online algorithm, then classified by gathering enough data from a set of monitored nodes and comparing it with historical data. This comparison guides the administrator down the right path for further debugging.

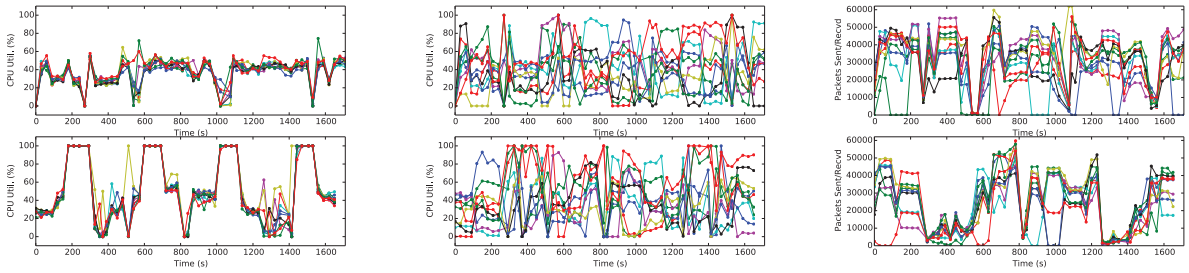
3 System Overview

We are working on integrating both offline storage and analysis of monitoring data with fast, scalable, online monitoring. We have built an extension to VScope [17] which adds a scalable database backend built on OpenTSDB [8], which we call *VStore*. This database allows us to bridge the gap between VScope's targeted, online analytics over short time windows, and the extended view of long-term interactions between application tiers provided by a database. An administrator can use VStore to draw upon a rich history of monitoring data to fine-tune online algorithms, and uncover anomalies that otherwise might be missed by VScope.

VStore integrates with VScope by hooking in to the online monitoring and analysis performed by Distributed Processing Graphs (DPG). First outlined in our previous works [19, 17], DPGs are essentially scalable overlay networks capable of being dynamically deployed and reconfigured with customized monitoring functions. For example, a ”watch” DPG may be deployed across all nodes to coarsely track low level metrics such as CPU and memory utilization, while a ”query” DPG might deploy on a subset of nodes to perform fine-grained application-level monitoring. To scale monitoring functions, DPGs support arbitrary aggregations, such as trees that aggregate values observed from child nodes in the monitoring graph. Additionally, VScope can run fast analytics on data aggregated in a DPG, which can be used to detect anomalies over a short time window. For instance, a DPG may gather 5 minutes worth of data and perform an entropy calculation (see [16, 20, 21]) to quickly identify CPU spikes or other anomalies, and alert an administrator of an impending fault.

To complement this, VStore can be deployed at the root of a DPG, where values are aggregated and analyzed. The root uses VStore to archive each individual observation before aggregation. Whereas an online process will throw away data after aggregating and analyzing over a short time window, VStore saves this history completely for later analysis. As a simple example, consider a DPG that aggregates CPU utilization at a number of nodes. The root of the DPG collects observations (perhaps once per second) from each node and buckets them over a sliding time window to perform an online analysis. At the same time, when each observation is received at the root, it sends it to VStore for archiving.

VStore acts as an OpenTSDB client, and provides a communications layer between custom monitoring code running inside DPGs and a scalable key-value store using HBase, the backbone of OpenTSDB. Monitoring code executing within a DPG that uses VStore can store any type of numeric metric that is being collected. To scale archiving at different points in the DPG, VStore can be



(a) CPU usage at HBase Region Servers **(b)** CPU usage at HDFS nodes **(c)** Network traffic at HDFS nodes

Figure 1: A periodic, high-CPU background task running at the HBase tier. The top graph shows a normal workload, while the bottom shows a periodic process starting and stopping every 5 minutes. A periodic high-CPU load at the HBase tier has relatively low impact on HDFS, making this type of anomaly more difficult to detect using a global online algorithm.

integrated at multiple levels. Each aggregation point in a DPG can thus use VStore to archive observations before either forwarding them or collapsing into a single value. Because it is backed by HBase, VStore can scale well to concurrent clients archiving a large amount of monitored data. This is critical to integrating with VScope's goal of providing low-overhead and scalable troubleshooting for over 1000 nodes.

4 Use Cases

To demonstrate how collection and offline analysis can be combined with online monitoring systems, we experimented with three typical scenarios seen in modern, multi-tiered data center applications: a periodic background process that interferes with normal processing, a misconfiguration in the software application itself, and a fault or misconfiguration at the network level. Our experiments are designed around a big data application, similar to that of [17], to collect and analyze web logs for micro-marketing purposes. We constructed a distributed log collection workload using Hadoop, HBase, and Apache Flume [3, 18].

Our experimental testbed is deployed on a local research cloud at Georgia Tech using the OpenStack platform [7], with virtual machines running Ubuntu Linux 12.10. We run an HDFS tier containing a Hadoop Master server, plus 10 VMs running data and task tracker processes. The HBase tier contains a Master server, plus 10 VMs acting as region servers, and is not colocated with HDFS nodes. 3 VMs are dedicated to a ZooKeeper quorum. Finally, 70 VMs serve as workload generators and aggregators for Flume. A workload generator runs a simple log generation process simulating a web server processing requests. These log records are sent via a Flume pipeline to sinks that connect directly to HBase to archive log records.

Each of the graphs in our results show two plots: the top plot shows data collected by VStore under normal conditions, while the bottom shows data for the same experiment with an anomaly introduced.

4.1 Periodic Processes

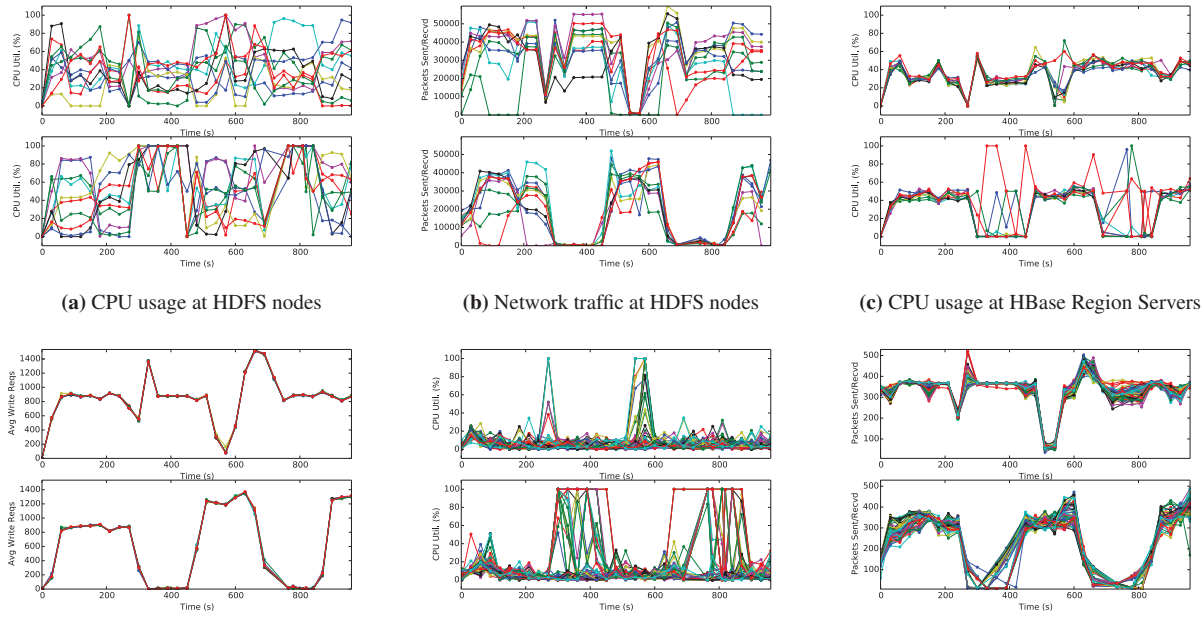
A common task for multi-functional servers in a data center is to periodically perform some short-lived operation. For example, a cron job may be used to clean up and archive system logs or application data once per day. In the case of our weblog application, HBase may be manually configured to perform major compactions on a periodic basis. Such tasks can be resource-intensive, and are typically scheduled at off-peak hours so as not to disturb other running applications. However, a misconfiguration or other fault may lead to such processes running at incorrect times.

Unfortunately, online monitoring algorithms may not detect and easily classify such cycles. As mentioned before, an online algorithm might only observe a short period of time during which a periodic process is not running. Furthermore, the observed window may not fully classify periodic behavior, for instance if only the beginning or end of a spike is observed.

To simulate a periodic process, we used the Linux "stress" utility to impose a load on certain servers, and measured the impact on metrics collected during this time. Each such stress test process runs every 5 minutes, for a duration of 2 minutes. We run 2 concurrent "stress" processes for a single resource, such as CPU or disk I/O.

4.1.1 CPU-intensive Periods

The first experiment simulates a periodic, high-CPU workload at the HBase tier. This is potentially common for major compactions which require frequent communication with HDFS nodes. Figure 1a shows the CPU utilization for each HBase region server. Each period of high-CPU activity is clearly identified, but only over a



(a) CPU usage at HDFS nodes **(b)** Network traffic at HDFS nodes **(c)** CPU usage at HBase Region Servers

(d) Avg write reqs at HBase Region Servers **(e)** CPU usage at Workload Generators **(f)** Network traffic at Workload Generators

Figure 2: A periodic, I/O intensive background task running on HDFS nodes. High disk I/O on HDFS nodes tends to starve many resources, and leads to anomalous behavior at all tiers

brief period of time. To properly correlate this behavior, an online algorithm would need an extended time window to distinguish between periodic spikes (such as those under normal load), and a deterministic process. Additionally, we can see from figures 1b and 1c that CPU and network utilization at other tiers remains relatively unaffected. Thus, an online algorithm that aggregates monitoring data across tiers may not efficiently detect the problem at the HBase tier alone.

4.1.2 Intense Disk I/O

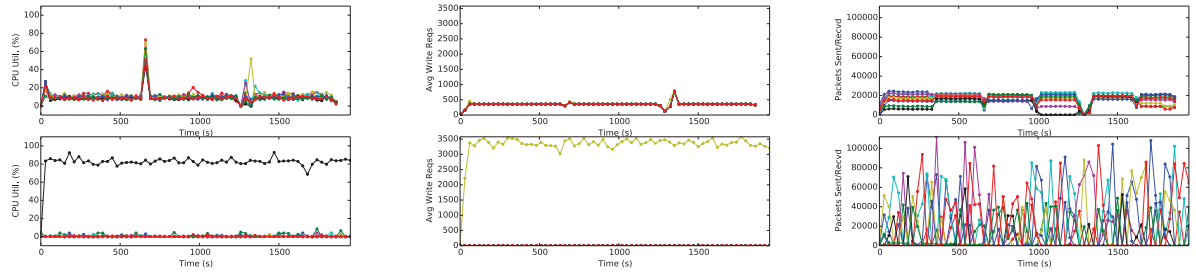
We also experimented with periods of high disk I/O at the Hadoop tier. This may be triggered by an external process (such as a periodic MapReduce Job), or an internal process (such as log cleanup). Additionally, as suggested in [13], anomalies in disk I/O could be caused by ”limpware,” failing or degraded hardware that causes excessive I/O latencies or a higher amount of disk reads compared to normal operation. For this test, we find that high disk I/O on HDFS nodes has a more drastic effect across tiers on different metrics.

Figures 2a and 2b compare CPU utilization and network flows at HDFS nodes. A spike in CPU utilization is visible during the period of high disk usage, but is difficult to distinguish from HDFS's normal CPU workload. However, we see a significant drop in packets sent and received during this time, indicating other tiers are being affected by the spike. Figures 2c and 2d show a significant change in CPU utilization and requests fielded by each region server, respectively, while figures 2e and 2f show a corresponding change at workload servers themselves. An online algorithm would need to observe both an extended time window and multiple metrics across tiers to detect this type of anomaly.

4.2 Software Misconfigurations

Another typical scenario involves a misconfigured or buggy software application which leads to load imbalance, crashes, or other anomalous behavior. In the case of HBase, one source of load imbalance stems from region splitting. By default, a client request is sent to the region server hosting the region that falls within a key range for the requested row. Tables are ”split” based on contiguous ranges of row keys. Although HBase will do this automatically, it is sometimes advantageous to pre-split tables to maintain a uniform distribution of requests across the cluster. For instance, if the row key is based on a hash function, each region server can be assigned portions of the hash space for balanced load distribution.

To emulate a misconfiguration under this scenario, we constructed a pre-split table in HBase using the `HexStringSplit` algorithm, which assumes that row keys are a uniformly distributed hexadecimal string. Flume agents write log records to HBase using a random UUID as the row key, providing even row key distribu-



(a) CPU Utilization at HBase Region Servers **(b)** Avg write reqs at HBase Region Servers **(c)** Network traffic at HDFS nodes

Figure 3: A software misconfiguration at the Flume tier. The HBase table is pre-split using `HexStringSplit`, and agents use a random hexadecimal string for row keys. Misconfigured agents use a timestamp instead, causing unbalanced regions.

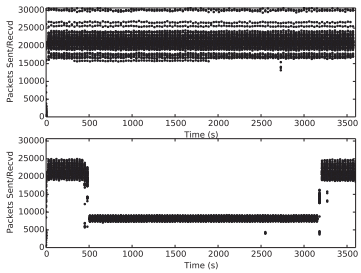
tion. An anomalous configuration uses a timestamp instead, leading to load imbalance as keys increase monotonically.

Figures 3a and 3b compare CPU utilization and write requests at HBase region servers, respectively. These figures show a clear imbalance among region server load when timestamps are used as the row key. From figure 3c, we also see how this can have a drastic effect on HDFS communications. This type of anomaly is distinct from periodic process interruptions, but could be missed by an algorithm that aggregates observations (for instance, calculating an average). However, the individual data captured by VStore clearly distinguishes the two scenarios.

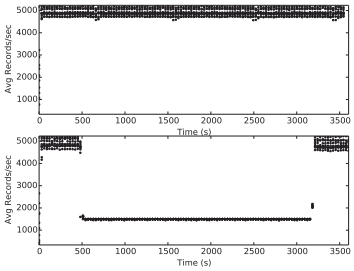
4.3 Network Faults

A faulty or misconfigured network interface is a common problem in data centers. Such a fault may stem from a variety of issues, such as a physical NIC incorrectly negotiating a link speed with a switch (see [4]), or a software-defined networking layer misconfiguring the bandwidth provisioned to a VM. As noted in [13], network hardware can also suffer ”limpware” anomalies - a faulty adapter may drop or corrupt packets periodically, forcing error correction mechanisms to be used and reducing overall bandwidth on the link. Sustained network bandwidth is important for a variety of applications. For instance, many web applications run realtime analytics on incoming requests at a high speed (such as calculating ”trending” topics). Faults at the network layer can thus directly affect an application’s business logic.

To simulate this, we configured 60 flume agents to generate an 8KB JSON object 5000 times per second (simulating web server requests). These requests are forwarded to 10 aggregator Flume agents, which count the number of records processed. Each server has a 1Gbps LAN connection; we simulate an anomaly by limiting bandwidth to 100Mbps at the 10 aggregation nodes.



(a) Network traffic at Flume agents



(b) Avg rate of records processed by Flume aggregation agents

Figure 4: Simulating a network misconfiguration. A real-time streaming aggregation application counts requests collected from a front-end tier. A VLAN misconfiguration or other network fault may limit bandwidth to a critical set of nodes, leading to reduced throughput throughout the application.

Figure 4a shows the network traffic for each server. With a faulty network link, there is a precipitous drop-off in network traffic across the cluster, as other agents become limited in the amount of work they can send to the aggregation tier. This situation also highlights how monitoring application metrics can provide useful anomaly classification information. Figure 4b shows the rate of records processed per second at each aggregation Flume agent. When bandwidth is scarce, agents are severely limited in the amount of records they can process.

4.4 Discussion

The data collected in our experiments indicate that there are uses for combined online and offline monitoring in a number of real-world debugging scenarios. Ideally, an operator troubleshooting a problem should be able to leverage both current and past data efficiently to rapidly identify and fix problems when the occur. Some opportunities for VStore to assist in these efforts include:

- *Targeted debugging with full history*: A strength of VScope is the ability to guide troubleshooting and monitoring efforts to the right servers when an anomaly is detected. VStore can integrate with VScope to offer automatic snapshots of historical data to aid in the troubleshooting process. A *baseline watch DPG* runs continuously in the datacenter and uses VStore to capture basic performance metrics, such as CPU and network utilization. When a problem occurs, the operator may use a combination of VScope's *watch*, *scope*, and *query* operations to locate the affected machines (refer to section 4.1 of [17] for an example). A new watch DPG is deployed on these machines to capture data about the current scenario. The *VShell* running this new watch presents the operator with current data, and automatically pulls snapshots of historical data from OpenTSDB for comparison - for instance data from the same time period for the past 10 days. The end result is that the operator has quickly gained knowledge about how a current problem relates to past events, which can help guide debugging efforts.

- *Detecting "limplock"*: The authors of [13] suggest that efficient detection through monitoring is one method of mitigating "limplock" - a situation where failing or degraded hardware causes systems to slow considerably, but does not trigger normal fail-stop mechanisms. VStore and VScope could be used as tools to detect limplock by providing a historical reference to help distinguish between overload-induced slowdown and failing hardware. VScope DPGs can be used to target monitoring on nodes and metrics known to cause limplock (such as HDFS write performance at Hadoop reducers). The online monitoring is then compared with historical data collected by VStore to determine if the system is limplocked, or simply experiencing a higher-than-usual load.

- *Classification of anomalies*: The rich set of historical data collected by VStore could be analyzed to help automatically classify anomalies, for instance using support vector machines [12] or statistical approaches like that of [11]. When combined with the targeted debugging approach outlined above, VScope could then query this data to attempt to automatically identify the type of anomaly currently taking place. If the current fault scenario can be classified using historical data, this helps the operator identify a debugging approach much more rapidly.

5 Conclusion and Future Work

In this paper we described VStore, an extension to our previous work VScope, which aims to bridge the gap between fast and flexible online monitoring systems with large-scale data collection for historical analysis. Our experiments show that VStore's capabilities in archiving fine-grained monitoring data across a large cluster of systems can help pinpoint hard-to-find anomalies that an online anomaly detection algorithm might miss. We have evaluated three common fault scenarios, and discussed how VStore's data archiving capabilities can be combined with VScope's online monitoring to complement debugging efforts for real-world scenarios.

A main goal of our future work involves a more complete integration of VStore with VScope's DPGs and online algorithms, to allow historical data to be captured dynamically at different points in the graph. This integration will allow us to begin exploring some of the use cases for fully integrated online and offline monitoring outlined in the discussion in section 4.4. In addition to monitoring and detecting application-level performance anomalies, we are also exploring ways to use VStore to detect infrastructure faults in cloud systems, such as a misconfigured SDN causing performance anomalies across cloud tenant applications.

A thorough performance evaluation of VStore is also needed. Results from [17] indicate that VScope can scale well to thousands of nodes, and we believe VStore also scales well as it is backed by HBase. Our experience with the experiments in this paper suggest the overhead of our system is small, but a detailed evaluation of VStore's scalability and perturbation when run along side real-time streaming data applications is required to quantify this.

Acknowledgements

We thank the anonymous reviewers for their insightful comments and feedback helping to improve this work.

References

[1] Amazon Elastic Compute Cloud (Amazon EC2). `http://aws.amazon.com/ec2`. Accessed: December 9, 2013.

[2] Amazon Elastic MapReduce (Amazon EMR). http://aws.amazon.com/elasticmapreduce. Accessed: December 9, 2013.

[3] Apache Flume. http://flume.apache.org. Accessed: December 7, 2013.

[4] The apache hbase reference guide. http://hbase.apache.org/book/book.html. Accessed: December 9, 2013.

[5] Ganglia Monitoring System. http://ganglia.sourceforge.net. Accessed: December 7, 2013.

[6] Nagios. http://www.nagios.org. Accessed: December 7, 2013.

[7] OpenStack Cloud Software. http://www.openstack.org. Accessed: December 7, 2013.

[8] OpenTSDB. http://opentsdb.net. Accessed: December 7, 2013.

[9] 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*, 53(4):50–58, Apr. 2010.

[10] L. A. Barroso and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009.

[11] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen. Fingerprinting the datacenter: Automated classification of performance crises. In *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, pages 111–124, New York, NY, USA, 2010. ACM.

[12] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[13] T. Do, M. Hao, T. Leesatapornwongsa, T. Patana-anake, and H. S. Gunawi. Limplock: Understanding the impact of limpware on scale-out cloud systems. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 14:1–14:14, New York, NY, USA, 2013. ACM.

[14] L. George. *HBase: the definitive guide*. O'Reilly Media, Inc., 2011.

[15] M. Kutare, G. Eisenhauer, C. Wang, K. Schwan, V. Talwar, and M. Wolf. Monalytics: Online monitoring and analytics for managing large scale data centers. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 141–150, New York, NY, USA, 2010. ACM.

[16] C. Wang. Ebat: Online methods for detecting utility cloud anomalies. In *Proceedings of the 6th Middleware Doctoral Symposium*, MDS '09, pages 4:1–4:6, New York, NY, USA, 2009. ACM.

[17] C. Wang, I. A. Rayan, G. Eisenhauer, K. Schwan, V. Talwar, M. Wolf, and C. Huneycutt. VScope: Middleware for Troubleshooting Time-Sensitive Data Center Applications. In *ACM/IFIP/USENIX International Conference on Middleware (Middleware)*, 2012.

[18] C. Wang, I. A. Rayan, and K. Schwan. Faster, Larger, Easier: Reining Real-Time Big Data Processing in Cloud. In *ACM/IFIP/USENIX International Conference on Middleware (Middleware)*, 2012.

[19] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. In *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, pages 141–150, New York, NY, USA, 2011. ACM.

[20] C. Wang, V. Talwar, K. Schwan, and P. Ranganathan. Online Detection of Utility Cloud Anomalies Using Metric Distributions. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2010.

[21] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan. Statistical Techniques for Online Anomaly Detection in Data Centers. In *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2011.