Exploring Graph Analytics for Cloud Troubleshooting

Chengwei Wang, Karsten Schwan, Brian Laub, Mukil Kesavan, and Ada Gavrilovska, *Georgia Institute of Technology*

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

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

Exploring Graph Analytics for Cloud Troubleshooting

Chengwei Wang, Karsten Schwan, Brian Laub, Mukil Kesavan, Ada Gavrilovska
College of Computing, Georgia Institute of Technology
{flinter, karsten.schwan, brian.laub, mukilk, ada}@cc.gatech.edu

Abstract

We propose VFocus, a platform which uses streaming graph analytics to narrow down the search space for troubleshooting and management in large scale data centers. This paper describes useful guidance operations which are realized with graph analytics and validated with representative use cases. The first case is based on real data center traces to measure the performance of troubleshooting operations supported by VFocus. In the second use case, the utility of VFocus is demonstrated by detecting data hotspots in a big data stream processing application. Experimental results show that VFocus guidance operations can troubleshoot Virtual Machine (VM) migration failures with accuracy of 83% and with delays of only hundreds of milliseconds when tracking migrations on 256 servers hosing 1024 VMs. Such successes are achieved with negligible runtime overheads and low perturbation for applications, in comparison to brute-force approaches.

1 Introduction

Troubleshooting large scale distributed systems/applications in data centers is important and challenging. There can be millions of entities (e.g., cores) running a large variety of applications across complex software stacks (e.g., hypervisors, guest VMs, middleware). With such complexity, variety, and large numbers, brute-force approaches logging all possible performance-relevant events, at all levels of abstraction, and for all entities, do not scale.

An emerging research area is to build systems that combine online monitoring with online data analytics – Monalytics [12]. As a representative solution, the VScope system developed in our previous work offers useful approaches to capture the 'most relevant' performance data about performance issues observed in large-scale data center applications [17]. Specifically, VScope uses lightweight, continuous, and global monitoring to detect performance anomalies, then 'zooms in' on those anomalies, by dynamically deploying more detailed methods for data capture and online data analysis. Results obtained from representative data center applications demonstrate clearly the advantages of VScope, compared in performance and accuracy to logging approaches capturing all performance-relevant events. Lacking from VScope, however, were the techniques needed to 'guide' the analyses being performed on captured monitoring information. VScope was not able to capture important relationships among the entities being monitored, nor did it provide structured ways to then analyze those relationships. Such a 'guidance' framework for monitoring data analysis is the key contribution of the VFocus system presented in this paper, which offers the following novel functionalities.

1. Interaction snapshot as a general representation for interactions among the software/hardware entities that present in data centers,

2. Streaming graph analytics as the online methods used to evaluate the continually evolving interactions represented in dynamically constructed snapshots.

With VFocus, data center administrators can create diverse interaction tracking methods, and the resulting 'guidance' methods can efficiently troubleshoot performance problems, by 'zooming-in' with both the data collection being performed and the analyses applied to such data. VFocus makes following novel contributions:

1. Graph-based analytics framework: a platform on which interaction graphs are constructed via online monitoring and analyzed using graph analytics, with the choice of graphs and analytics depending on the interactions that users wish to track.

2. Guidance operations using graph analytics: using VFocus guidance operations including *sort*, *group*, and *explore*, users can build various troubleshooting methods to reduce the search space for troubleshooting.

3. Validation with data center traces and use cases:

VFocus and its functionalities are evaluated with use cases to identify VM migration failures and to diagnose 'data hotspots' in the HBase key value store. Experimental results show that VFocus can troubleshoot VM migration failures with an accuracy of 83%, and with low overheads and interference with the applications.

2 VFocus Design and Implementation

2.1 System Overview

The VFocus system realizes the two-phase guidance mechanism illustrated in Figure 1. In the *snapshot construction* phase, VFocus collects metric data from monitored entities and builds interaction snapshots, realtime graphs in which vertices are software/hardware components and edges are interactions among the components. Interaction snapshots are updated continuously, to reflect the latest activities in the system. The second phase is *snapshot analysis*, for which VFocus exposes three primitive operations: *sort, group, and explore*, explained in more detail in Section 2.2. Data center operators use those operations to track and analyze interactions in the system, and to identify the entities to the performance issue being observed.

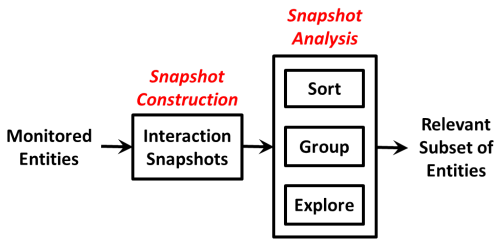


Figure 1: Guidance Framework

VFocus leverages the *DPG (Distributed Processing Graphs)* introduced in [17]. A *DPG* is an overlay network consisting of processing nodes called *VNodes* that each collects and analyzes monitoring data at realtime, in a streaming manner. *DPG* topologies can vary, to match monitoring needs (e.g., scale [18], and they can be deployed at runtime and on-demand, as described in more detail in [17]). On this basis, the system architecture of VFocus depicted in Figure 2 enables users to interact with it via two interfaces, an *analysis console* and an *archive console*, both of which are integrated in the *VMaster*, a controller process for *DPG* manipulations. The *analysis console* takes users' input as operation commands to analyze interaction snapshots and provides users with guidance results. The *archive console* is used to manage snapshots persisted in the VFocus backend store. In some troubleshooting scenarios, e.g., when analyzing snapshots spanning a time duration too long to fit all snapshots in memory, the realtime analysis may query the archiving system for history data.

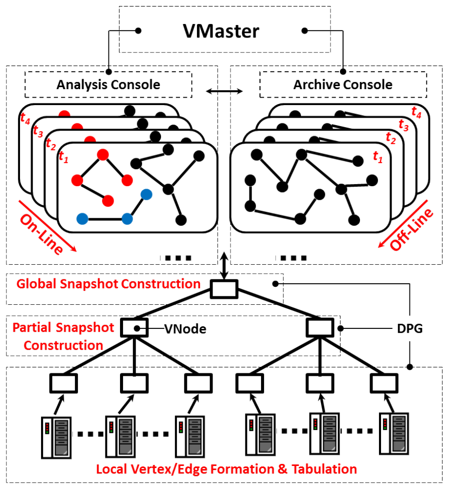

VMaster
Analysis Console
Archive Console
t_4
t_3
t_2
t_1
On-Line
Off-Line
Global Snapshot Construction
Partial Snapshot Construction
VNode
DPG
Local Vertex/Edge Formation & Tabulation

Figure 2: VFocus Architecture

The system operates as follows. The VMaster starts the guidance process by deploying a hierarchical *DPG* on the monitored machines[1]. Raw monitoring data collected periodically by the DPG's leaves are processed to extract vertices and edges for forwarding to parent nodes. Parents aggregate vertices and edges into edge lists that represent partial snapshots of the machines monitored by those parents' leaves. Partial snapshots are passed up the tree to the root for aggregation into a global snapshot.

2.2 Guidance Operations

The primitive graph analysis functions provided by VFocus are listed in Table 1. There are functions to calculate the basic properties of a graph, such as the degree of some specific vertex or the number of vertices and edges in the snapshot. For instance, the degree of a vertex is used in Section 3.1 to find the candidate servers that are most likely to have VM migration failures. There are also functions for tracking the relationships among vertices, e.g., *Search Neighbor* and *Clique Analysis*. Section 3.2 outlines a use case in which neighbor analysis is used to determine the HBase region server with a data hotspot. As stated earlier, all such analysis functions are executed on *VNodes* at runtime, at the same time as monitoring metrics are being collected. This results in repeated incremental snapshots suitable for rapid analysis and subsequent problem troubleshooting. Finally, some analytics are run concurrently and in a distributed manner. For example, an aggregation tree is used to implement the

[1]The *DPG* runtime is installed on those machines.

Table 1: Functions.CC(Connected Components), C(Centralized), H(Hierarchical)

Graph Function	Description	Mode
Search Vertex/Edge	Search vertex/edge	C
Search Neighbor	(In)direct neighbors	C
Count Vertex	Get number of vertices	H
Count Edge	Get number of edges	H
Get Degree	Get degree of a vertex	H
Get Attribute	Get vertex/edge attr.	C
Clustering/Clique/CC	Grouping vertices	C

Table 2: Operations.CC(Connected Components)

Operation	Basic Option	Supporting Fun.
Sort	Vertices by degree	Get Degree
	Edges/Vertices by attr.	Get Vertex/Edge
Explore	(In)direct neighbors	Search Neighbors
	Vertices/Edges by attr.	Search Vertex/Edge
Group	Vertices by attr.	Clustering
	Vertices by connection	Clique/CC

Count Degree function. Others operate in a centralized manner, e.g. the *Search Neighbor* function.

Analysis results are used by guidance operations to narrow down the search space of vertices and edges. Toward this end, VFocus provides three primitive guidance operations listed in Table 2: *sort*, *group*, and *explore*. They are implemented with different snapshot analysis functions listed in Table 1. Data center operators access and execute the operations in analysis console as command lines.

The *Sort* operation, supported by the *Get Degree* and *Get Vertex/Edge* functions, orders vertices or edges by their attributes (or by other aggregated properties like 'degree') and returns the top entities in the sorted list.

The *Explore* operation, supported by the *Search Vertex/Edge* and the *Search Neighbors* functions, can search the neighbors of some specified vertex at different distances (measured by the number of hops) or it can search the vertices/edges with some specified attribute. An example of its use appears in Section 3.2, where the *explore* operation is used to track the data connectivities between the HBase region servers and the HBase clients, in order to troubleshoot the data hotspot issue.

The *Group* operation places closely related vertices into a shared group, returning the group as the entity subsequently manipulated. 'Grouping' style analysis has also been used in previous troubleshooting research [10], e.g., where VMs are clustered by their similarity in terms of their computation behavior. One such similarity is a group of VMs are interacting frequently, a fact that can give rise to optimizations in which the migration of one such VM triggers the migrations of others, to avoid unnecessarily high levels of machine cross-talk in the data center. Finally, the *Group* operation has two options: grouping by attributes using clustering algorithm, or grouping by the connections between vertices, using clique analysis and connected components, etc..

3 Use Cases

3.1 VM Migration Analysis

In virtualized data centers, VM migration moves a VM running on a source host to a different destination host. Performed for consolidation and resource management purposes, migration is a 'heavyweight' management operation both in terms of its effect on the VM itself, the network and machine resources consumed. Migration failures, therefore, have substantial performance implications. A cause for migration failure is the inappropriate choice of destination machines, one reason being an *overloaded host* with insufficient capacity to rapidly complete the migration. Migration methods, therefore, will not select targets with high internal workloads, as indicated by their CPU or memory utilization. A remaining issue, however, is the external workload surrounding target hosts, an example being insufficient current network capacity to move the VM's substantial internal states quickly to the target machine.

VFocus can assist in VM migration management (1) via online tracking of host interactions, scaling to hundreds of server systems, and then (2) by using guidance operations to identify *overloaded hosts* not only based on their internal workloads but also based on their interactions with other hosts. We demonstrate the utility of VFocus for identifying overloaded hosts by 'replaying' a VM management operation trace recorded from a virtualized data center. The trace is collected from 256 servers in the Techway virtualized data center at Georgia Tech, a 'green computing' facility run jointly by the CERCS and CEETHERM research centers. The servers host a total of 1024 VMs running enterprise workloads, including **(1) Nutch:** a data analytics benchmark, **(2) Voldemort:** a key-value store [16] with *YCSB* [9] as load generator, **(3) Cloudstone:** a cloud web-serving benchmark, **(4) Linpack:** a high performance computing workload. The VM migration traces are collected from October 05, 2012 8:23:15 AM to October 12, 2012 12:04:20 PM. These 172 hours of data document 31790 successful and 2845 failed migrations.

VFocus is used as follows. A centralized *DPG* with one master *VNode* and 8 slave *VNodes* is driven by attaching to each slave log record generators. Each generator runs 1/8th of the log describing successful migrations, injecting log records into the slave *VNodes*. These leaf VNodes, then, periodically process log records to generate local snapshots and perform local snapshot analysis, the results of which are then sent to the master *VNode* to create global snapshots for guidance operations. We use an adjustable sliding window storing 1000 migrations in these experimental evaluations, in order to make statistically sound predictions, and slide the window after every new migration. In each snapshot, the vertices are the hosts, and the edges are migrations from source to destination host. There are approximately 256 vertices and 1000 edges in every snapshot, and there can be multiple edges between two vertices due to multiple migrations between two hosts in the same time period.

The troubleshooting process has following steps. In the first step, we use *group* operation to divide the global interaction snapshot, which consist of partial snapshots, into sub-graphs in each of which there is at least one path between two vertices. The hypothesis behind using this operation is that the load of a host does not come from the other host which it does not interact with.

When there is a new migration observed on a host, VFocus first checks which group this host belongs to, and then uses the *sort* operation on degrees of all the vertices (hosts) in that group and ranks the hosts in a descending order. The reason behind this operation is that the top hosts on the list are having/or had most migrations, and hence are more likely to have resource scarcity issue when new migration requests take place. In the trace, any two migrations happen at different time, therefore there is only one group chosen when the *sort* operation is executed. The *sort* operation will provide the top *n* hosts as candidates which may potentially have VM migration failures in the future. *n* is tunable and in our experiment, we find that *n*=45 gives us a good performance.

After replaying the trace in VFocus, it produces a series of candidate lists. To validate the guidance methodology, we match the list to the respective error log record at each failure point. If the actual host with migration failure is in the list, then we consider VFocus as a 'hit' as the predicted list contains the actual failure node, otherwise we consider it as a 'miss'. In this paper we only study the failures due to overloaded hosts which are indicated as 'operation timed out' in the failure log.

Table 3 shows the performance of VFocus guidance. We can see in the table that, the overloaded host errors have a significant percentage of all the errors (over 50%). As a guidance approach, VFocus reduces the search space from 256 servers to 45 servers, with an overall hit rate of over 83%. We further study the 'position' of a hit which is essentially the actual number of candidates needed to be checked before reaching the real failure node. The distribution of hit position is shown in Figure 3. About 70% of hits happen within 30 hosts and nearly 40% is within 15 hosts, which means for the majority of time, size of search space can be further reduced from 45 to 30.

Table 3: VFocus Guidance Accuracy

# Overload Errors	# Overload Found	Hit Rate
1441 (51% of all errors)	1195	83%

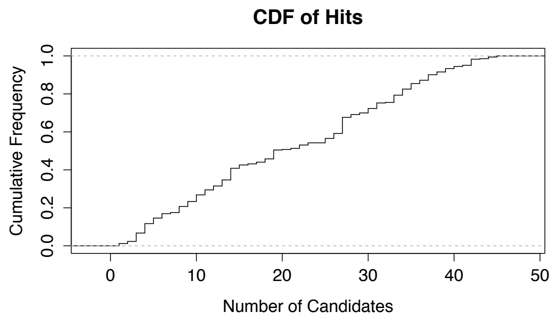


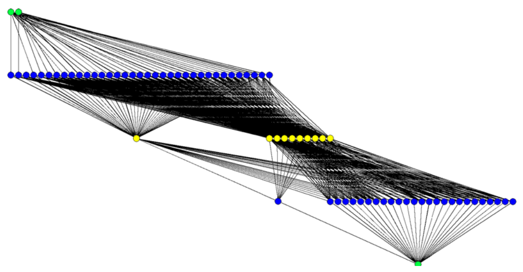
Figure 3: Distribution of Hit Positions

3.2 On-line Data Hotspot Analysis

HBase, a distributed key-value store system, is widely used as an infrastructure to support big data applications. HBase consists of multiple region servers which serve the read/write requests (mostly through memory cache) concurrently to reduce the latency and increase the throughput. One of the annoying performance issues in this platform is that some region servers in HBase become data hotspots which receive significantly more read/write requests than other region servers. As a majority portion of requests go to the hotspot region servers, their cache in main memory is filled up quickly. Those region servers then become the bottleneck and bring down the overall read/write performance of the HBase. One of the common reasons for the hotspot issue is the imbalanced accessing pattern on the row key space. Which region server each read/write request goes to is decided by the row key value in the request, and in HBase the total row key space is evenly divided, as equal regions managed by region servers. Therefore, if the row keys in the requests, either from one application or from multiple applications, concentrate on some of the key regions, the according region servers become hotspots.

VFocus addresses this challenge in two aspects. First the interaction snapshots can track the runtime communications among HBase region servers online. Secondly, the hotspots can then be easily spotted by using guidance operations.

We validate VFocus' effectiveness in our virtualized data center supported by OpenStack. We started over 100 VMs on 30 physical servers running Xen hypervisor. We


(a) Normal Interactions

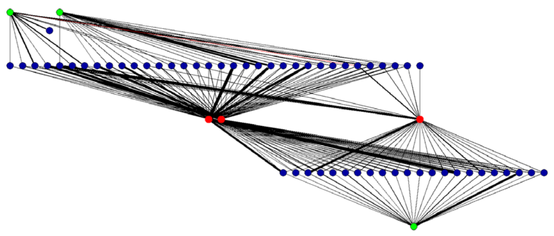

(b) Interactions with Hotspots

Figure 4: Interaction Snapshots in Hotspot Analysis

deployed two applications (1) GTStream [2], a big data streaming processing platform using FlumeNG [1], (2) traditional read/write clients using YCSB [9], a widely used cloud benchmark. The two applications share the same HBase infrastructure though they access different tables in the HBase. In GTStream application, each Flume Agent writes at a rate of 100 requests per second to HBase accessing a table named 'test'. We use a mix of 50/50 reads and writes workload in YCSB clients and each client send 100 requests per second to HBase accessing a table named 'usertable'. Both tables' key regions are divided evenly across all the region servers. We deploy VFocus on all the VMs and create a hierarchical *DPG* with 10 parent *VNodes* on 10 dedicated VMs. We use *Libpcap* [3] to sniff the network traffic on every VM.

We showcase VFocus' guidance functionalities in two scenarios. The first scenario represents a normal workload of HBase where there is no hotspot. In this scenario, both the YCSB clients and Flume agents assign row key value in a balanced way. For every request, YCSB creates a key value as a combination of a string and a hash value randomly selected from the key value space. In Flume, a key value is assigned as a combination of an arbitrary web url which the flume agent is processing and a timestamp. Figure 4(a) illustrates one of the snapshots constructed by VFocus online. Yellow dots are all the region servers. Green dots represent zookeeper servers. Blue dots represent HBase clients including Flume agent and YCSB clients. The edges among vertices are the network communications and the thickness of an edge indicates the traffic intensity between the two VMs. The snapshot tracks the network interactions among zookeepers, HBase clients (YCSB clients and Flume agents), and region servers. By using *explore* operation on vertices representing region servers to find its neighbors, we can easily find that all the region servers are being accessed by all the YCSB clients and Flume agents with approximately equal traffic between any two edges.

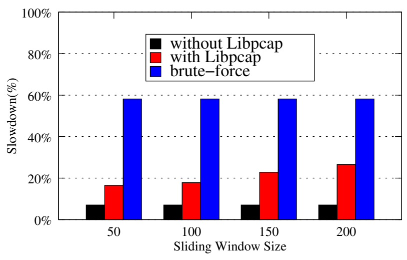
In the second scenario, both the YCSB clients and Flume agents assign row key value as a combination of a fixed string and current timestamp, which monotonically increases instead of randomly distributed in the key space. Therefore, the key values will condense in a limited range and the requests from both YCSB and Flume will go to a subset of region servers, making them hotspots. Figure 4(b) illustrates one of the snapshots constructed in this hotspot scenario, where red dots are hotspot region servers. Besides tracking the interactions among VMs according to their roles in the applications, running *explore* operation on all the region servers figures that only three region servers are communicating with HBase clients, indicating the three hotspots.

4 Performance Evaluation

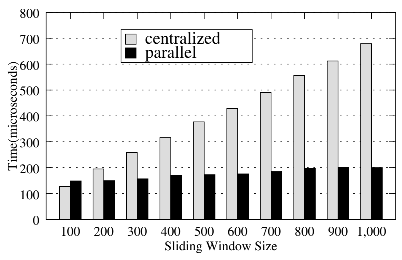
4.1 VFocus Overhead

To test the overhead of VFocus on a VM, we run the network interaction tracking used in Section 3.2 on VFocus and change the durations of sniffing. We measure the VM's CPU and Memory utilization increase as the overheads.The overhead for VFocus is within 2% in CPU utilization while the memory consumption is no more than 0.2%. The CPU utilizations increase slightly as the duration increases as the network sniffing consumes more CPU cycles as it continuously runs for a longer time.

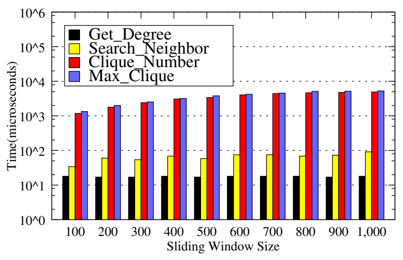
The interferences of VFocus and brute-force approach to the application are shown in Figure 5(a). It shows that as the monitoring duration increases, VFocus' interference to the application increases, while the interferences are within 30% when the duration increases from 50 seconds to 200 seconds. By contrast, the always-on brute-force approach has a considerably higher interference at around 58%. To further study the breakdown of the overheads, we turned off the *Libpcap-based* data collection function on all the Flume agents, and instead stored network connection metrics, which are pre-recorded in file using Libpcap, in memory. VFocus reads the memory and processes data in the same way as we did when using Libpcap. The black bar in Figure 5(a) indicates that the VFocus itself does not play the major role in the total overheads because it only incurs 7.06% slowdown,



(a) Interference to Application



(b) Graph Construction Time



(c) Graph Analysis Performance

Figure 5: VFocus Performance Results

compared to over 20% slowdown when *Libpcap-based* monitoring is enabled.

4.2 Snapshot Construction Performance

We test the snapshot construction performance by generating VM migration snapshots in VM migration use case with different sliding window sizes, and compare two construction strategies, centralized and parallel. In the centralized strategy, we send all the migration data to a central node and generate a global snapshot; while in the parallel strategy, we create a snapshot in a distributed way by which 8 *VNodes* create partial snapshots which are aggregated at the root *VNode*. Each snapshot has up to 256 vertices and 1000 edges. As shown in Figure 5(b), parallel construction outperforms centralized strategy and as the size of sliding window increases the construction time increases as well. However the maximum construction time is within 700 milliseconds, and the parallel construction strategy increases much slower than the centralized strategy does. Validated by real data center monitoring traces, the results show than the snapshot construction in VFocus is fast, hence VFocus is capable of tracking online, runtime interactions responsively. Parallel construction strategy used by VFocus is more scalable and faster than the centralized strategy.

4.3 Graph Analysis Performance

We evaluate the VFocus' performance on snapshot analysis by measuring the computation time of analysis listed in Table 1. Clique analysis has two functions (1) *Clique_Number* function that counts the number of cliques in the snapshot and (2) *Max_Clique* function that yields the largest clique in the snapshot. The monitoring data is the trace used in Section 3.1. We measure computation time at different sliding window sizes.

As shown in Figure 5(c), the computation time is within 10000 microseconds when the sliding window size changes from 100 to 1000, which means for VM migration use case, the graph analysis can be conducted in a timely manner. As the sliding window enlarges, the time just increases slightly because all the computation are processed in memory, which make it suitable for real-time graph analysis. Among different analysis functions, the two clique analyses have longer computation time because they have higher computation complexities than degree analysis and neighbor analysis.

5 Related Work

VFocus is similar to previous research on *dependency inference*.[8, 5, 4] use network traffic and signal processing methods to infer dependencies. [7] leverages application-level knowledge along with network traffic information, to infer dependencies. The fundamental difference between VFocus and dependency inference is that the latter does not provide a general framework for graph abstraction and analysis on dependencies. In *request path diagnosis* research, [13, 6] highlight performance differences between application activities by comparing their request execution paths. VFocus is different because it focuses on creating continuous snapshots of the interactions on-line while request path diagnosis are off-line approaches. VScope [17] is a flexible middleware which can dynamically deploy monitoring and analysis functions on any monitored entities at any time. VFocus leverages VScope's flexible architecture but it is a graph analysis system aiming to track and analyze the interactions among entities in data centers on-line at real-time. VScope has built-in guidance mechanism using ad-hoc approaches, which lacks extensibility and generality while VFocus provides a guidance framework and primitive guidance operations by which different troubleshooting approaches can be implemented.

References

[1] Apache flume. `http://flume.apache.org/`.

[2] Gtstream. `https://github.com/chengweiwang/GTStream`.

[3] Libpcap. `http://www.tcpdump.org/`.

[4] S. Agarwala, F. Alegre, K. Schwan, and J. Mehalingham. E2EProf: Automated End-to-End Performance Management for Enterprise Systems. In *IEEE Conference on Dependable Systems and Networks (DSN)*, 2007.

[5] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed system of black boxes. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.

[6] M. Attariyan, M. Chow, and J. Flinn. X-ray: Automating root-cause diagnosis of performance anomalies in production software. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012.

[7] P. Bahl, R. Chandra, A. G. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2007.

[8] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl. Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2008.

[9] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, 2010.

[10] L. Hu, K. Schwan, A. Gulati, J. Zhang, and C. Wang. Net-Cohort: Detecting and Managing VM Ensembles in Virtualized Data Centers. In *ACM International Conference on Automatic Computing (ICAC)*, 2012.

[11] M. Kesavan, A. Gavrilovska, and K. Schwan. Elastic Resource Allocation in Datacenters: Gremlins in the Management Plane. In *VMware Technical Journal*, 2012.

[12] 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 *ACM International Conference on Automatic Computing (ICAC)*, 2010.

[13] R. R. Sambasivan, A. X. Zheng, M. De Rosa, E. Krevat, S. Whitman, M. Stroucken, W. Wang, L. Xu, and G. R. Ganger. Diagnosing Performance Changes by Comparing Request Flows. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[14] B. C. Tak, C. Tang, C. Zhang, S. Govindan, B. Urgaonkar, and R. N. Chang. vpath: precise discovery of request processing paths from black-box observations of thread and network activities. In *USENIX Annual Technical Conference (ATC)*, 2009.

[15] R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 2003.

[16] Voldemort. Project voldemort - a distributed database. http://project-voldemort.com/.

[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, 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 *ACM International Conference on Automatic Computing (ICAC)*, 2011.

[19] P. Yalagandula and M. Dahlin. A Scalable Distributed Information Management System. In *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, 2004.

[20] Z. Zhang, J. Zhan, Y. Li, L. Wang, D. Meng, and B. Sang. Precise Request Tracing and Performance Debugging for Multi-Tier Services of Black Boxes. In *IEEE Conference on Dependable Systems and Networks (DSN)*, 2009.