ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Clouddiag, An Efficient, Unsupervised Diagnosis For Locating Fine-Grained Performance Anomalies

Mandalaneni Vijaykumar
M. Tech Student,
Department Of CSE,
Guntur Engineering College, Yanamadala.

N.Bhagya Lakshmi, M.Tech
Assistant Professor,
Department Of CSE,
Guntur Engineering College, Yanamadala.

Abstract:

Typical productions in existing cloud systems are service-oriented in nature. The response time of user requests directly reflects the system performance. In this regard, tracing user requests is a viable means to exposing performance data, so as to help performance diagnosis.

Recent work has shown that it is promising to pinpoint performance anomalies with end-to-end request tracing data. However, an efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies is still lacking.

Performance diagnosis is labor intensive in production cloud computing systems. Such systems typically face many real world challenges, which the existing diagnosis techniques for such distributed systems cannot effectively solve.

An efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies is still lacking in production cloud computing systems. This paper proposes CloudDiag to bridge this gap.

Combining a statistical technique and a fast matrix recovery algorithm, CloudDiag can efficiently pinpoint fine-grained causes of the performance problems, which does not require any domain-specific knowledge to the target system. CloudDiag has been applied in a practical production cloud computing systems to diagnose performance problems. We demonstrate the effectiveness of CloudDiag in three real-world case studies.

Keywords:

Fine-Grained, Unsupervised, Scalable, Cloud computing, performance diagnosis, request tracing.

INTRODUCTION:

PERFORMANCE diagnosis is labor intensive, especially for typical production cloud computing systems. In such systems, a lot of software components bear a large number of replicas (component instances) distributed in different physical nodes in the cloud. They can be assembled into multiple types of services, serving large amounts of user requests. The services provisioned by the cloud are often prone to various performance anomalies (e.g., SLA violations [1]) caused by software faults, unexpected workload, or hardware failures.

Such defects may, however, be manifested only in a small part of component replicas, hiding themselves in a large number of normal component replicas. Our experiences in performance diagnosis for Alibaba Cloud Computing1 show that troubleshooting performance anomalies in practical production cloud computing systems faces many real-world challenges. It is very difficult to apply existing diagnosis techniques for such distributed systems. We summarize the new design challenges as follows:

1.Performance diagnosis in fine granularity. A component typically hasa lot of replicas in aproduction cloudsystem. Within one component, there are many performance-related private methods (i.e., those invoked inside the component) and public methods (i.e., the interfaces invoked by other components). It is very challenging to localize anomalous methods as well as their corresponding physical replicas. Current approaches generally focus on locating anomalous physical nodes (e.g., [2]) or logical components (e.g., [3]). Such coarse-grained results are not enough. In the former case, given an anomalous physical node, a system operator has to identify the faulty component amongmanycomponents typically hosted in the same node.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

In the latter case, given an anomalous logical component, the operator has to identify which one among their numerous replicas distributed in the cloud is faulty. Consequently, huge human efforts are still required to further pinpoint the subtle primary cause. Performance diagnosis in a fine granularity is of high concern to reduce manual efforts.

2. Unsupervised performance diagnosis. Many existing performance diagnosis techniques resort to system behavior models in identifying anomalies [4], [5]. Unfortunately, it is hard to manually build such models in production cloud systems, given their complexity in system scale. In addition, cloud services are generally composed of many components developed by different teams, which are independently updated online. It is extremely difficult to maintain the behavior models for such evolutional systems. Hence, a performance diagnosis tool for production cloud systems should be completely unsupervised, without assuming that any prior knowledge about the service should be input.

3. Performance diagnosis with high efficiency. Coping with large runtime data generated by a production cloud system efficiently is a challenging task in performance diagnosis. Many defects only manifest themselves in an online production cloud that involves a large number of component replicas. Unlike a small-scale in-house debugging system, a production cloud system can generate massive performance logs during its runtime, which are recorded in a distributed manner across a large number of cloud nodes. It is, therefore, critical to design a fast, scalable performance diagnosis tool chain that can assemble the relevant logs on demand when performance anomalies occur, and quickly pinpoint the primary cause accordingly. Yet, this is not the focus of the current approaches (e.g., [6]).

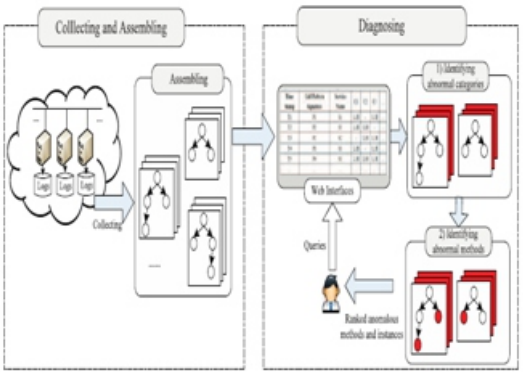
RELATED WORK:

Typical production cloud systems are service-oriented in nature. The response time of user requests directly reflects the system performance. In this regard, tracing user requests is a viable means to exposing performance data, so as to help performance diagnosis. Recent work [7], [8],[9], [10] has shown that it is promising to pinpoint performance anomalies with end-to-end request tracing data.

However, an efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies is still lacking. This paper bridges this gap by proposing CloudDiag. CloudDiag periodically collects the end-to-end tracing data (In particular, execution time of method invocations) from each physical node in the cloud. It then employs a customized Map-Reduce algorithm to proactively analyze the tracing data. Specifically, it assembles the tracing data of each user request, and classifies the tracing data into different categories according to call trees of the requests. When the cloud system is suffering performance degradation (e.g., average response time of user requests is larger than a threshold), a cloud operator can access CloudDiag with its web interfaces to conduct a performance diagnosis.

With the request tracing data, CloudDiag will perform a fast customized matrix recovery algorithm to instantly identify the method invocations (together with the replicas they locate) which contribute the most to the performance anomaly. The whole process requires no domain-specific knowledge to the target service. CloudDiag has been successfully launched in diagnosing performance problems for the production cloud systems in Alibaba Cloud Computing. We report three case studies in our real-world performance diagnosis experiences to demonstrate the effectiveness of CloudDiag in helping the operators localize the primary causes of performance problems.

SYSTEM ARCHITECTURE:



FRAMEWORK OF CLOUDDIAG:

Performance anomalies in cloud systems will manifest themselves as anomalous response time of user requests. Since a service is composed of a lot of components, a service with anomalous performance must have involved some components with performance anomalies.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

A component typically has a lot of replicas in a production cloud system; however, the performance anomaly of a component may be manifested only in a small part of its replicas. This will cause the performance degradation of the involving service, which is frequently observed in the cloud computing systems of Alibaba Inc. Fig. 1 shows the execution time of a component method in different replicas in a 100-node cloud system. We can instantly see that only a small part of replicas (e.g., nodes 8 and 12) are anomalous when executing the method. Such performance problems are the most difficult to locate, because the anomalous methods hide themselves in numerous well-functioning replicas. Therefore, to reduce human efforts in pinpointing performance anomaly, a performance diagnosis tool must first identify which component methods contribute to the performance.

CloudDiag is composed of three major parts, i.e.,

1) collecting the performance data;

2) assembling the performance data; and

3) identifying the primary causes of the anomalies.

We briefly overview each part as follows:

* Collect performance data. CloudDiag traces user requests at a given sampling rate to expose performance data. For the sampled requests, each component replica records the performance data andsaves them in its local storage. An important consideration is what kind of performance dataCloudDiag should collect and how. CloudDiagadopts an instrumentation-based approach that collects the execution time of each component method. Details are discussed in Section 3.

* Assemble performance data. CloudDiag should first assemble the performance data distributed in numerous component replicas in a request-oriented way. In other words, the performance data belonging to the same requests are correlated together. CloudDiag will then analyze such request-oriented performance data and infer the call tree of each sampled request. A customized map-reduce processis utilized to group requests into different categories based on their call trees. Requests within one category share the same call tree.

* Identify the primary causes of anomalies. CloudDiag then identifies the anomalous categories according to their latency distribution. Then, for each anomalous category, a fast customized matrix recoveryalgorithm (i.e., robust principal component analysis (RPCA) [11]) is employed to identify the anomalous method invocations together with the replicas they are located. Details are discussed in Section

Typical productions in existing cloud systems are service-oriented in nature. The response time of user requests directly reflects the system performance. In this regard, tracing user requests is a viable means to exposing performance data, so as to help performance diagnosis. Recent work has shown that it is promising to pinpoint performance anomalies with end-to-end request tracing data. However, an efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies is still lacking.

* It is very challenging to localize anomalous methods as well as their corresponding physical replicas. Consequently, huge human efforts are still required to further pinpoint the subtle primary cause.

* It is extremely difficult to maintain the behavior models for such evolutional systems. Hence, a performance diagnosis tool for production cloud systems should be completely unsupervised, without assuming that any prior knowledge about the service should be input.

* Coping with large runtime data generated by a production cloud system efficiently is a challenging task in performance diagnosis. Many defects only manifest themselves in an online production cloud that involves a large number of component replicas.

This paper bridges this gap by proposing CloudDiag. CloudDiag periodically collects the end-to-end tracing data (In particular, execution time of method invocations) from each physical node in the cloud.

It then employs a customized Map-Reduce algorithm to proactively analyze the tracing data. Specifically, it assembles the tracing data of each user request, and classifies the tracing data into different categories according to call trees of the requests.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

* CloudDiag has been successfully launched in diagnosing performance problems for the production cloud systems in Alibaba Cloud Computing.

* We report three case studies in our real-world performance diagnosis experiences to demonstrate the effectiveness of CloudDiag in helping the operators localize the primary causes of performance problems.

Cloud Server:

we design a Cloud server in local host by having the functionalities of a Cloud Storage system. Where the data owners can upload their files securely and also use it. The Cloud Server allows the access of authorized users to access the files of the data owners too.

Data Owner :

we designed a data owner module by having a unique registration for each data owner such that a new data owner should register in cloud server and then get their access to log in. After that the data owner has the facility to upload their data in the cloud server.

User :

first the user has to be get access by registering themselves when they are new to the system. Once after registration they can login the system and can find the details of the files uploaded by the data owner. The authorized users can make their request to download the file. RequestID should be unique for every request. It is assigned when a request arrives the system.

Data Collection and Assembling :

we develop the system to have data collection and assembling them into it. It is named as CloudDiag. It traces user requests at a given sampling rate to expose performance data. For the sampled requests, each component replica records the performance data and saves them in its local storage. An important consideration is what kind of performance data CloudDiag should collect and how. CloudDiag adopts an instrumentation-based approach that collects the execution time of each component method.

CloudDiag should first assemble the performance data distributed in numerous component replicas in a request-oriented way.

In other words, the performance data belonging to the same requests are correlated together. CloudDiag will then analyze such request-oriented performance data and infer the call tree of each sampled request. A customized map-reduce process is utilized to group requests into different categories based on their call trees.

Diagnosing:

From the data collected by the previous module we design diagnosing module. Where, CloudDiag then identifies the anomalous categories according to their latency distribution.

We can identify the anomalous method by measuring the execution time deviation of each method one by one. We design an unsupervised machine learning algorithm to automatically learn the characteristics of the invoked methods and identify which methods are anomalous together with on which replicas they are executed.

PERFORMANCE DATA COLLECTION:

In this section, we introduce what kind of performance data that CloudDiag should collect and how to collect them.

Our instrumentation-based tracing approach will produce performance data when a sampled request is being processed in each component replica. Specifically, each component method, when being invoked or returning, will generate a log entry.

The data structure of a tracing log entry is shown in Fig. 3a, which contains five items. Host indicates the machine where the component replica locates.

Time stamp records the time of the event occurrence (i.e., a method invocation or a method return). RequestID is the global identifier of a request. MID is a unique identifier for request.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Host	Timestamp	RequestID	MID	Method	Flag
Host1	2012-01-01 16:31:31.690272	169	739	AliStorge.ReadFile	Start
Host1	2012-01-01 16:31:32.991376	169	739	AliStorge.ReadFile	End

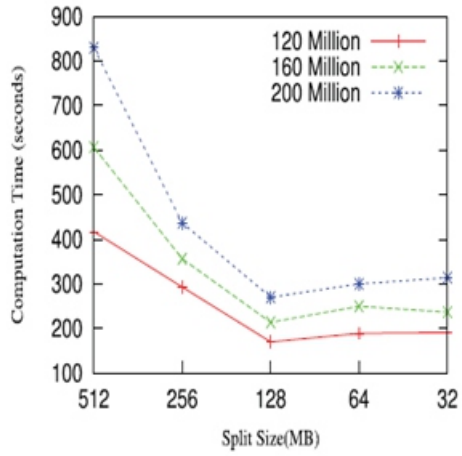
(a) Log entry for method execution time

Host	Timestamp	RequestID	Caller MID	Callee MID
Host1	2012-01-01 16:31:31.690724	169	739	991

(b) Log entry for method invocation relationship

EVALUATION:

CloudDiag has been launched in Alibaba Cloud Computing Company to perform anomaly diagnosis in itsproduction cloud computing systems. This section reports three case studies during our experiences in usingCloudDiag in Alibaba. Our target cloud system is a cloud facility for Aliyun Mail, a production e-mail system that provides free e-mail service to the public.2 ListMail, ReadMail, and SendMail are three services that are utilized to handle requests of listing mail titles, reading mail contents, and sending mails, respectively. They are the typical services of our target system, and are the focus of our experimental studies.Services are composed of a series of components (e.g.,storage and communication). Each component has many homogeneous replicas that are deployed on different hosts.



Computation time under different volumes of split sizes.

Currently, more than 20 million user requests are handled per day. On average, a request will typically go through over 10 hosts, invoking over 100 instrumented methods. By default, requests are sampled with the ratio of 1/200.

Generally, the target cloud system would produce about 30-50 gigabytes (around 120-200 million lines) oftracing logs per hour. CloudDiag is deployed in a small cluster with 10 nodes. Each node is a typical low-end computer running Linux RHEL 5.4.

CloudDiag proactively pulls the tracing data from the target cloud in a periodical manner (once every hour in our experiments). It then runs a customized mapreduceprocess to assemble and classify tracing data. First, map tasks assign correlated tracing logs that belong to the same requests to corresponding reduce tasks.

Second, reduce tasks generate and classify requests into categories. For the map-reduce cluster, one important parameter is the split size, i.e., the volume of data assigned to each Maptask. The split size determines the number of Map tasks.

A smaller split size indicates that more Map tasks are required to process a given data set. We vary the split sizes from 32 to 512 MB for three data sets (with trace entries sizesbeing 120, 160, and 200 million lines of trace logs).

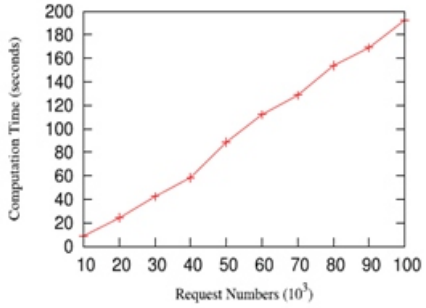
The computational time of the map-reduce procedure is shown in Fig. 8. We can see the cluster performs the best when the split size is 128 MB. Hence, in the rest of our experiments, we set the split size 128 MB.Finally,

CloudDiag adopts a state-of-the-art RPCA implementation, called inexact ALM algorithm [21] in its performance anomaly detection approach. It is a mature open-source implementation, and can be used in a blackboxway for CloudDiag.

SCALABILITY EVALUATION:

For efficiency consideration, CloudDiag is required to be scalable to the massive performance data. Since CloudDiag conducts the tracing data collection and assembly proactively,the anomaly diagnosing step is the only issue that will influence the scalability of CloudDiag. We study the efficiency of the RPCA-based anomaly detection approach. The inputs are the performance data of a typical category of requests to the SendMail service, which bears

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal



Scalability of the RPCA based anomaly detection.

a critical call tree that contains 117 methods. There are about 4 million of requests following this call tree each day. Fig. 9 plots the computation time of the anomaly detection approach under different request numbers.

It shows that the computational time of the approach also scales almost linearly with the performance data volumesof up to 100 thousand requests. This demonstrates the high scalability of the RPCA-based anomaly detection algorithm.The process of computing a 100;000 _ 117 matrix takes lessthan 200 seconds.

DISCUSSIONS:

In the above three case studies, we compare CloudDiag with the PCA-based approach in terms of precision and recall. From Table 2, we can see that CloudDiag outperforms the PCA-based approach in both measures, especially in term of precision. Hence, CloudDiag can savemore effort in troubleshooting the primary cause of the performance anomalies. This shows that the PCA-based approach cannot well handle the performance data with gross errors.

It consequently generates more false positives and false negatives. The RPCA-based approach CloudDiag, on the other hand, can work well for such non-Gaussian performance data.

There are two thresholds in our approach. _ and _ are, respectively, utilized to detect whether categories or invoked methods are anomalous or not. _ is conventionallyset to 1 [15]. In our experimental study, we have set _ in range [0.4, 0.8]. Although anomalous methods can be successfully identified by CloudDiag in our three casestudies, we observe that the value of beta can slightly influence the precision. Empirically, the value 0.5 is a best choice for _.

CONCLUSION AND FUTURE WORK:

Request tracing technologies have been proven effective in performance debugging. Here, CloudDiag resorts to a white-box instrumentation mechanism to trace service requests, because the source codes of services are generally available in typical production cloud systems. Note that such a white-box performance data acquisition component of CloudDiag can also be substituted with another tracing mechanism if it can obtain the latency data of method invocations.

Another way to trace requests is via black-box mechanisms. Black-box tracing mechanisms assume no knowledge of the source codes. But, existing approaches generally cannot directly obtain the latency data of method invocations. In this regard, a black-box tracing mechanism can be deemed as a tracing mechanism with the large granularity (e.g., in node level).

There is a tradeoff between tracing granularity and debugging effort. As a result, more effort will be increased in troubleshooting the performance anomalies if a black-box tracing mechanism is applied. To incorporate black-box tracing mechanisms with CloudDiag, a future direction is to explore black-box tracing mechanisms so that a fine granularity (i.e., in method invocation level) can be achieved. To this end, the runtime instrumentation can be a promising technique.

In conclusion, the work proposes CloudDiag, an efficient, unsupervised diagnosis tool for locating fine-grained performance anomalies. The experimental results demonstrate that our approach scales well to massive tracing data. In future, the work implements that CloudDiag can effectively and conveniently help operators diagnose three real-world performance problems with high precision and recall.

REFERENCES:

1]. Haibo Mi, Huaimin Wang, Yangfan Zhou, Michael Rung-Tsong Lyu and Hua Cai, "Toward Fine-Grained, Unsupervised, Scalable Performance Diagnosis for Production Cloud Computing Systems."IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp 1245-1254, June-2013.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

[2]. B. Sigelman, L. Barroso, M. Burrows, P. Stephenson, M. Plakal, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Technical Report Dapper-2010-1, Google, 2010.

[3]. F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, "Bigtable: A Distributed Storage System for Structured Data," ACM Transaction Computer Systems, vol. 26, no. 2, pp. 1-26, 2008.

[4]. H. Mi, H. Wang, Y. Zhou, M.R. Lyu, and H. Cai, "P-tracer: Path-Base Performance Profiling in Cloud Computing Systems," proc. IEEE 36th Ann. Computer Software Applications Conference (COMPSAC), pp. 509-514, 2012.

[5]. M. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem Determination in Large, Dynamic Internet Services," Proc. IEEE International Conference Dependable Systems and Networks (DSN), pp. 595-604, 2002.

[6]. P. Reynolds, C. Killian, J. Wiener, J. Mogul, M. Shah, and A. Vahdat, "Pip: Detecting the Unexpected in Distributed Systems," Proc. USENIX 3rd Symposium Networked Systems Design and Implementation (NSDI), pp. 115-128, 2006.

[7]. Jens-Matthias Bohli, Nils Gruschka, Meiko Jensen, Luigi Lo Iacono and Ninja Marnau, "Security and Privacy-Enhancing Multicloud Architectures", IEEE Transactions on Dependable and Secure Computing, Vol. 10, No. 4, pp. 212-224, July/August 2013.

[8]. M. V. Mahoney and P. K. Chan. "Learning Rules for Anomaly Detection of Hostile Network Traffic", 3rd IEEE International Conference on Data Mining, pp. 601-604,2003.

[9]. Varun Chandola, Arindam Banerjee and Vipin Kumar "Model-based Thermal Anomaly Detection in Cloud Datacenters", ACM Computing Surveys,pp. 1-72, Sept.-2009.

[10]. Kaustav Das and Jeff Schneider "Detecting Anomalous Records in Categorical Datasets", 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 220-229, 2007.

[11]. Chengwei Wang, Vanish Talwar, Karsten Schwan and Parthasarathy Ranganathan "Online Detection of Utility Cloud Anomalies using Metric Distributions", IEEE Network Operations and Management Symposium (NOMS), pp. 96-103, april 2010.

[12]. Lena Tenenboim-Chekina, Lior Rokach and Brach Shapira "Ensemble of Feature Chains for Anomaly Detection", © Springer-Verlag Berlin Heidelberg, 2011.

[13]. Husanbir S. Pannu, Jianguo Liu and Song Fu "AAD: Adaptive Anomaly Detection System for Cloud Computing Infrastructures", 31st International Symposium on Reliable Distributed Systems, pp. 396-397, 2012.

[14]. Matthias Gander, Basel Katt, Michael Felderer, Adrian Tolbaru, Ruth Breu, and Alessandro Moschitti "Anomaly Detection in the Cloud: Detecting Security Incidents via Machine Learning",©Springer Verlag Berlin Heidelberg, pp. 103-116, july 2013.

[15]. Haroon Malik,Bram Adams, Ahmed E. Hassan" Automatic Detection of Performance Deviations in the Load Testing of Large Scale Systems" IEEE 35th International Conference on Software Engineering (ICSE), pp. 1012-1021, may-2013.

[16] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online System Problem Detection by Mining Patterns of Console Logs," Proc. IEEE Int'l Conf. Data Mining (ICDM), pp. 588-597, 2009.

[17] A. Oliner and A. Aiken, "Online Detection of Multi-Component Interactions in Production Systems," Proc. IEEE/IFIP 41st Int'l Conf. Dependable Systems and Networks (DSN), pp. 49-60, 2011.

[18] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for Traffic Anomaly Detection," ACM SIGMETRICS Performance Evaluation Rev., vol. 35, no. 1, pp. 109-120, 2007.

[19] H. Mi, H. Wang, G. Yin, H. Cai, Q. Zhou, and T. Sun, "Performance Problems Diagnosis in Cloud Computing Systems by Mining Request Trace Logs," Proc. IEEE Network Operations and Management Symp. (NOMS), pp. 893-899, 2012.

[20] I. Jolliffe, Principal Component Analysis. Springer, 2002.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

[21] Z. Lin, M. Chen, L. Wu, and Y. Ma, "The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices," Arxiv preprint arXiv:1009.5055, 2010.

[22] K. Nagaraja, F. Oliveira, R. Bianchini, R. Martin, and T. Nguyen, "Understanding and Dealing with Operator Mistakes in Internet Services," Proc. USENIX Sixth Conf. Symp. Operating Systems Design and Implementation (OSDI), pp. 5-20, 2004.

[23] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier, "Using Magpie for Request Extraction and Workload Modelling," Proc. USENIX Sixth Conf. Symp. Operating Systems Design and Implementation (OSDI), pp. 259-272, 2004.

[24] E. Thereska and G. Ganger, "Ironmodel: Robust Performance Models in the Wild," ACM SIGMETRICS Performance Evaluation Rev., vol. 36, no. 1, pp. 253-264, 2008.

[25] B. Sigelman, L. Barroso, M. Burrows, P. Stephenson, M. Plaka l, D. Beaver, S. Jaspan, and C. Shanbhag, "Dapper, a Large-Scale Distributed Systems Tracing Infrastructure," Technical Report dapper-2010-1, Google, 2010.

[26] H. Mi, H. Wang, Y. Zhou, M.R. Lyu, and H. Cai, "P-tracer: Path- Base Performance Profiling in Cloud Computing Systems," Proc.