A Lightweight Tool for Anomaly Detection in Cloud Data Centres

Sakil Barbhuiya, Zafeirios Papazachos, Peter Kilpatrick and Dimitrios S. Nikolopoulos
School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, BT7 1NN, Belfast, U.K.

Keywords: Anomaly Detection, Cloud Computing, Data Centres, Monitoring, Correlation.

Abstract: Cloud data centres are critical business infrastructures and the fastest growing service providers. Detecting anomalies in Cloud data centre operation is vital. Given the vast complexity of the data centre system software stack, applications and workloads, anomaly detection is a challenging endeavour. Current tools for detecting anomalies often use machine learning techniques, application instance behaviours or system metrics distribution, which are complex to implement in Cloud computing environments as they require training, access to application-level data and complex processing. This paper presents LADT, a lightweight anomaly detection tool for Cloud data centres that uses rigorous correlation of system metrics, implemented by an efficient correlation algorithm without need for training or complex infrastructure set up. LADT is based on the hypothesis that, in an anomaly-free system, metrics from data centre host nodes and virtual machines (VMs) are strongly correlated. An anomaly is detected whenever correlation drops below a threshold value. We demonstrate and evaluate LADT using a Cloud environment, where it shows that the hosting node I/O operations per second (IOPS) are strongly correlated with the aggregated virtual machine IOPS, but this correlation vanishes when an application stresses the disk, indicating a node-level anomaly.

1 INTRODUCTION

Data centres that host Cloud computing services are catalysts for the economy and for society. Despite the success and proliferation of the Cloud computing paradigm, hosting data centres face immense challenges in terms of managing the scale and complexity of their hardware infrastructure and their system software stack. The hosted workloads also become increasingly complex and diverse. The complexity of the data centre ecosystem gives rise to frequent disruption of service, which manifests as lack of responsiveness, lower than expected performance, or complete disruption of service. Anomaly detection in data centres is a recently emerged field of research which explores methods to automate the detection and diagnosis of service disruption in data centres via the monitoring of system generated logs and metrics.

Log analytic tools (Lou et al., 2010; Xu et al., 2009) extract features from logs and use statistical learning techniques to automatically build models that detect and diagnose system anomalies in data centres. These models are not easy to understand by human operators as they may detect problems in a high dimensional feature space without providing meaningful explanations for the detected problems. Besides, learning-based tools require a log parser for mining the console logs in order to create the features of the model. Log parsers require the source code of hosted applications in order to recover the inherited syntax of logs, and source code may not always be accessible for legacy applications. As an alternative to using console logs, a number of anomaly detection tools use system metrics (Tan et al., 2012; Wang, 2009; Ward and Barker, 2013; Kang et al., 2010; Jiang et al., 2009). Such metrics can be collected with minimum overhead and without requiring access to the source code of hosted applications.

Some tools (Wang, 2009; Kang et al., 2010) use system metrics for detecting anomalies in Cloud data centres and are developed to take the elasticity of Cloud computing environments into consideration. These tools provide higher accuracy and effectiveness of detection. However, implementing them in a large-scale Cloud system is complex as the tools are configured in multiple layers of data monitoring and analysis. EbAT (Wang, 2009) implements metrics distribution, entropy time series construction, and entropy time series processing across multiple monitoring layers, before deploying an online tool for detecting anomalies. Peerwatch (Kang et al., 2010) extracts correlated behaviours between multiple instances of individual applications and then applies those extracted correlations to identify anomalies. The use of

Barbhuiya S., Papazachos Z., Kilpatrick P. and S. Nikolopoulos D..
A Lightweight Tool for Anomaly Detection in Cloud Data Centres.
DOI: 10.5220/0005453403430351
In *Proceedings of the 5th International Conference on Cloud Computing and Services Science* (CLOSER-2015), pages 343-351
ISBN: 978-989-758-104-5
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

application-level metrics in anomaly detection algorithms may raise enormously the volume of data to process in a large-scale system.

This paper introduces LADT (Lightweight Anomaly Detection Tool), a lightweight tool which monitors system-level and virtual machine (VM)-level metrics in Cloud data centres to detect node-level anomalies using simple metrics and correlation analysis. LADT addresses the complexity of implementing efficient monitoring and analysis tools in large-scale Cloud data centres, by collecting and storing the metrics generated by nodes and VMs using Apache Chukwa (Rabkin and Katz, 2010). The LADT algorithm performs correlation analysis on the collected data using Apache Pig (Olston et al., 2008) and MapReduce jobs in order to complete the analysis in a timely manner.

As a use case, we deploy LADT to add a new dimension in anomaly detection for Cloud data centres, namely correlation analysis between node-level disk I/O metrics and VM-level disk I/O metrics. LADT implements a simple algorithm, which uses Pearson's correlation coefficient. When the average correlation coefficient value for a number of consecutive time intervals drops below a threshold, an alarm is raised to indicate an anomaly. The metrics data required in this use case are limited to disk I/O activity data and the algorithm to detect the anomalies is lightweight. The major contribution of LADT over other tools in this context is the ability to efficiently monitor and detect anomalies in a Cloud data centre without requiring complex algorithms, application source code availability, or complex infrastructure set up.

We evaluate LADT in a lab-based Cloud ecosystem, where it continuously collects and stores system metrics from all nodes and VMs. The tool shows that the disk I/O metrics in each hosted node are strongly correlated with the aggregate VM disk I/O metrics, but this correlation vanishes when a disk-stressing application is introduced in a node, indicating a node-level anomaly.

The remainder of the paper is organised as follows. Section 2 presents background and related work in anomaly detection. Section 3 and Section 4 provide detail of the LADT architecture and algorithm, respectively. Experimental results are presented and discussed in Section 5. Section 6 concludes the paper and discusses future work.

2 BACKGROUND AND RELATED WORK

In this section we describe the challenges of detecting anomalies in Cloud data centres. A number of different tools and methods are considered and placed into separate categories based on the input they analyse. We consider two cases for monitoring and detecting anomalies: console log based anomaly detection and anomaly detection based on system metrics.

2.1 Anomaly Detection Challenges

Cloud data centres are implemented as large-scale clusters with demanding requirements for service performance, availability and cost of operation. As a result of scale and complexity, data centres exhibit large numbers of system anomalies caused by operator error (Oppenheimer et al., 2003), resource over/under provisioning (Kumar et al., 2007), and hardware or software failures (Pertet and Narasimhan, 2005). These anomalies are inherently difficult to identify and resolve promptly via human inspection (Kephart and Chess, 2003). Thus, automatic system monitoring that captures system state, behaviour and performance becomes vital. Computer system logs are the main source of information for anomaly detection. Logs can be of two types: structured or unstructured. Unstructured logs are free-form text strings, such as console logs, which record events or states of interest and capture the intent of service developers (Lou et al., 2010), whereas structured logs are numerical logs, such as logs of system metrics, which capture workload and system performance attributes, such as CPU utilisation, memory usage, network traffic and I/O.

2.2 Console Log based Anomaly Detection

Analytic tools for anomaly detection based on console logs, such as SEC (Rouillard, 2004), Logsurfer (The and Prewett, 2003) and Swatch (Hansen and Atkins, 1993) check logs against a set of rules which define normal system behaviour. These rules are manually set by developers based on their knowledge of system design and implementation. However, rule-based log analysis is complex and expensive because it requires significant effort from system developers to manually set and tune the rules. Moreover, modern systems consisting of multiple components developed by different vendors and the frequent upgrades of those components make it difficult for a single expert to have complete knowledge of the total system and to set the rules effectively. This complexity has given rise to statistical learning based log analytic tools such as the works of Lou et al. (Lou et al., 2010) and Xu et al. (Xu et al., 2009), which extract features from console logs and then use statistical techniques to automatically build models for system anomaly identification.

Lou et al. (Lou et al., 2010) propose a statistical learning technique which consists of a learning process and a detection process. The learning process groups the log message parameters and then discovers the invariants among the different parameters within the groups. For new input logs, the detection process matches their invariants among the parameters with learned invariants from the learning process. Each mismatch in the invariants is considered to be anomalous. Xu et al. (Xu et al., 2009) propose a new methodology to mine console logs to automatically detect system problems. This first creates feature vectors from the logs and then applies the PCA (Principal Component Analysis) algorithm on the feature vectors to detect anomalies. However, the learning based tools require a custom log parser for mining the console logs in order to create the features for the learned model. The log parsers require source code of the hosted applications to recover the inherent structure of the logs.

2.3 System Metric based Anomaly Detection

A number of anomaly detection tools use system metrics (Tan et al., 2012; Wang, 2009; Ward and Barker, 2013; Kang et al., 2010; Jiang et al., 2009), which can be collected with minimum overhead and without requiring any access to the source code of hosted applications. Using system metrics for detecting anomalies has advantages over traditional log-based anomaly detection tools due to consideration of elasticity and workload evolution in Cloud computing, but also due to provisioning, scaling, and termination of services in short periods of time. Some of these tools are based on feature selection and machine learning outlier detection to flag anomalies (Azmandian et al., 2011).

EbAT (Wang, 2009) is a tool that uses entropy based anomaly detection. EbAT analyses metric distributions and measures the dispersal or concentration of the distributions. The metrics are aggregated by entropy distributions across the Cloud stack in order to form entropy time-series. EbAT uses online tools like spike detection, signal processing and subspace methods to detect anomalies in the entropy time-series. The tool incurs the complexity of analysing the metric distributions and also requires third party tools to detect anomalies.

PeerWatch (Kang et al., 2010) uses canonical correlation analysis (CCA) to extract the correlations between multiple application instances, where attributes of the instances are system resource metrics such as CPU utilisation, memory utilisation, network traffic etc. PeerWatch raises an alarm for an anomaly whenever some correlations drop significantly. As a result of analysing the application instance behaviours and correlating them, this tool is capable of detecting application-level or VM-level anomalies. However, this approach requires statistical metrics analysis and knowledge of the hosted applications, which is a limitation in large-scale Clouds, where hundreds of different types of applications run on the VMs.

Varanus (Ward and Barker, 2013) uses a gossip protocol, which is layered into Clouds, groups and VMs in order to collect system metrics from the VMs and analyse them for anomalies. This approach allows in-situ collection and analysis of metrics data without requiring any dedicated monitoring servers to store the data. However, setting up a dedicated gossip protocol across thousands of VMs in a large-scale Cloud environment and maintaining the gossip based overlay network over each of the VMs is a challenging task.

The metric-based black box detection technique presented in (Tan et al., 2012) uses the LFD (Light-Weight Failure Detection) algorithm to detect system anomalies. LFD raises an alarm when there is a lack of correlation between two specific system metrics. The anomaly indicates a system problem and each such problem is associated with a specific system metrics pair. LFD is a lightweight algorithm with lower complexity than EbAT, PeerWatch and Varanus. Furthermore, LFD does not require any training or source code and understanding of hosted applications. The LFD follows a decentralised detection approach, where each node analyses its own system metrics in order to achieve higher scalability. However, this may also limit LFD in large-scale Cloud data centres, where it may not be feasible to implement LFD on each node individually, due to overhead.

In this paper we address the limitations of existing system anomaly detection tools by introducing LADT. LADT uses Apache Chukwa (Rabkin and Katz, 2010) for collecting metrics data from all nodes and VMs in a data centre, and HBase (Vora, 2011) for storing the data in servers to allow centralised monitoring of Cloud systems. LADT implements a new correlation algorithm to perform the correlation analysis on the centrally stored metrics data. The LADT algorithm correlates node-level and VM-level metrics, which is a new approach to correlation analysis in detecting Cloud system anomalies. Furthermore, the LADT algorithm deals with the synchronisation problem between the node and VM generated metrics timestamp. This problem arises due to latency in storing the VM-level metrics in the monitoring server and results in poor correlation analysis. LADT is lightweight as it uses a simplified infrastructure set-up for metrics data monitoring and the LADT algorithm uses the simple Pearson correlation coefficient for analysing the metrics data. We program the algorithm using Apache Pig (Olston et al., 2008) to leverage MapReduce jobs in order to achieve higher throughput. We use disk I/O metrics from both nodes and VMs in an actual Cloud set-up to detect I/O performance anomalies.

3 LADT ARCHITECTURE

The following sub-sections describe the architecture of the LADT tool and its functionality.

3.1 Metrics Data Monitoring

LADT utilises an agent-based monitoring architecture to retrieve system metrics from the hosting nodes and VMs in a Cloud data centre. The monitoring agent extracts system metrics from the nodes and VMs at regular time intervals. The collector gathers the data extracted by the agents. LADT uses the agents and collectors provided by Apache Chukwa's (Rabkin and Katz, 2010) runtime monitoring. The Chukwa agent collects CPU, memory, disk, and network information from the hosting nodes using sigar (Sigar, 2014) and from the VMs using Virt-Top (Virt-Top, 2014). The Chukwa collector then collects the output generated by the agents. The collector processes the data and registers the input to HBase, which is installed in the monitoring node.

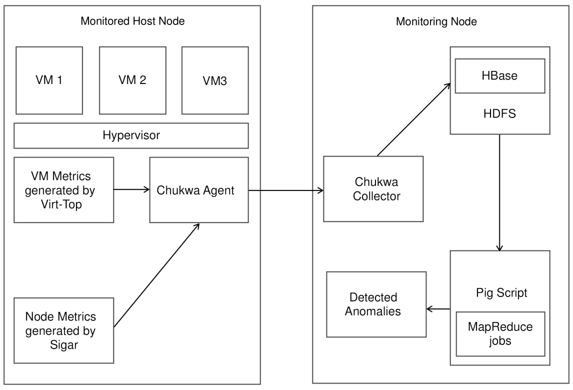

Figure 1: LADT Architecture.

Figure 1 illustrates the architectural overview of LADT. The tool installs one Chukwa agent for collecting both node-level and VM-level metrics on each monitored node in the data centre. LADT uses Chukwa collectors running on data analysis servers, for collecting node-level and VM-level metrics into HBase. The correlation analysis on the stored metrics is performed with Pig Script. Each Chukwa agent consists of adaptors which are dynamically loadable modules that run inside the agent process. LADT sends the metrics from the agents to the Chukwa collectors via HTTP. The primary task of the collector is to parse the collected data from the agent and store the extracted information in an HBase database. HBase runs on top of the Hadoop Distributed File System (HDFS) (Shvachko et al., 2010).

3.2 Metrics Data Analysis

LADT runs the metrics correlation analysis on the stored metrics using Apache Pig (Olston et al., 2008), which in turn executes MapReduce jobs. A Pig Script written in Pig Latin implements the LADT algorithm to explore the correlation between the node-level and VM-level metrics and to detect anomalies. The Pig Script program first loads both the node-level and VM-level system metrics from HBase. It then takes the mean values of the metric samples in 15-second windows. The program groups the mean values into 5-minute windows and calculates the Pearson Correlation Coefficient between the node-level and VM-level metrics in each group. Finally, the program compares the correlation coefficient value with the threshold value and generates an anomaly alarm if it finds the coefficient value is below an adjustable threshold level.

4 LADT Hypothesis and Algorithm

The underlying approach of LADT is based on the premise of the LFD technique, which identifies two metrics that are correlated during normal operation but diverge in the presence of an anomaly.

4.1 LFD: The Baseline Method

LFD is a lightweight technique for anomaly detection proposed by Tan et al. (Tan et al., 2012). The hypothesis of LFD is that in an anomaly-free system, whenever an application requests service, the processing alternates between two phases: the communication phase and the compute phase. In the communication phase, the application responds to requests received from the user, reads data from the disk, or writes data to the disk. In the compute phase, the application operates on the received inputs or requests. At the operational level, the compute phase is characterised by user-space CPU activity, whereas the communication phase is characterised by the behaviour of one or more types of system-level resource consumption, including kernel-space CPU time, disk I/O or network I/O. Behavioural change between the two phases results in a correlation between user-space CPU utilisation and system resource consumption. The LFD also hypothesises that in the case of an anomaly, there will be a significant change in the relationship between the compute phase and the communication phase. Hence, there will be lack of correlation between user-space CPU utilisation and system resource consumption, based on which anomalies can be detected in the system.

4.2 LADT Hypothesis

LADT formulates a new hypothesis, according to which there is strong correlation between the node-level and VM-level metrics in a Cloud system. Also, that this correlation will drop significantly in the event of any performance anomaly at the node-level and a continuous drop in the correlation can indicate the presence of a true anomaly in the node.

4.3 LADT Algorithm

We propose a new algorithm based on LFD, to correlate node-level and VM-level metrics. We use disk I/O metrics as a running example. The algorithm correlates disk I/O between the hosting node and the VMs in order to detect anomalies in IaaS Cloud environments. LADT computes the Pearson correlation coefficient (ρ) between the hosting node disk IOPS and the aggregated VM disk IOPS. Pearson's correlation coefficient is the ratio of the covariance between the two metrics to the product of their standard deviations as described in Equation 1 and ranges between -1.0 and 1.0.

$$\rho_{N,V} = \frac{covariance(N,V)}{\sigma_N \sigma_V} \quad (1)$$

where N = time-series of node disk IOPS
V = time-series of VM disk IOPS

Similar to the LFD algorithm, there are five tunable parameters in the LADT algorithm, which are summarised in Table 1. LADT collects raw disk I/O metrics with a 3 second period from both the nodes and VMs. The metrics collection period is set to more than a second in order to mitigate the timestamp synchronisation problem between the node-level and VM-level metrics, which arises as a result of latency in storing VM-level metrics in the LADT monitoring server. Before calculating the correlation coefficient between the metrics, their mean values are taken in small windows ($LW = 5$) in order to reduce noise. An anomaly alarm is raised when the average of D consecutive values of the correlation coefficient drops below the threshold T in order to detect the true anomalies by considering a larger range of system behaviour.

We keep the same parameter values as the LFD algorithm for four parameters ($LW = 5, LS = 5, W = 60, D = 10$) (Tan et al., 2012), which we experimentally found to achieve best performance. However, the correlation window slide, S is changed from 12 to 60 as this amortises better the overhead of a Pig (Hadoop) program, which is used to execute the LADT algorithm. Therefore, each correlation coefficient value considers the last $LW \times W = 300$ seconds (5 minutes) of system behaviour.

5 EXPERIMENTAL EVALUATION

This section describes the workload and the experimental set-up used to evaluate LADT. The section also analyses our experimental results and the functionality of the LADT tool.

5.1 Experimental Set-up

For evaluating LADT we have established a Cloud testbed with three compute nodes: Host1, Host2, and Host3 on three Dell PowerEdge R420 servers. Each node is running CentOS 6.5 with the 2.6.32 Linux kernel and has 6 cores, 2-way hyper-threaded, clocked at 2.20 GHz with 12GB DRAM clocked at 1600 MHz. Each node includes a 7.2K RPM hard drive with 1TB of SATA and an onBoard Broadcom 5720 Dual Port 1GBE network card. We run three VMs on each compute node using KVM. Each VM runs the disk I/O intensive Data Serving benchmark from CloudSuite (Ferdman et al., 2012). Disk read/write metrics are generated every three seconds in the nodes using sigar (Sigar, 2014) and in the VMs using Virt-Top (Virt-Top, 2014).

The experiment runs over a time period of 60 minutes, where we inject an anomaly in Host1 at the end of the first 30 minutes. We use a disk stressing application which increases the disk read/write operations and runs for two minutes with a two minute interval between runs for the remaining 30 minutes. This anomaly reflects a Blue Pill or a Virtual Machine Based Rootkit (VMBR) attack on a Cloud system, where the attacker introduces fake VMs via a hidden hypervisor on the victim hosting node to get access to the hardware resources such as CPU, memory, network or disk (Dahbur et al., 2011). This anomaly may

Table 1: Tunable parameters in LADT Algorithm.

LW, low-pass window width	raw samples to take mean
LS, low-pass window slide	raw samples to slide
W, correlation window width	samples to correlate
S, correlation window slide	samples to slide detection window
D, diagnosis window	correlation coefficients to consider

also represent a distributed denial-of-service (DDoS) attack, which forces the cloud service to consume system resources such as processor power, memory, disk space, or network bandwidth to an intolerable level (Rajasekar and Imafidon, 2010).

Earlier research (Antunes et al., 2008) has already used system resource-level anomaly analysis to deal with such attacks. The work of Antunes et al. (Antunes et al., 2008) analyses system resource utilisation to explore the normal system behaviour and builds a model, based on which it detects the abnormal behaviours in the system, and subsequently, the attacks. However, this approach to detecting attacks requires a large amount of historical data and use of machine learning techniques. In contrast, LADT can detect the attacks using correlation analysis between the node-level and the VM-level metrics.

LADT is implemented in the testbed, which uses Chukwa agents in each of the hosting nodes to collect both the hosting node and VM disk read/write metrics. The tool stores the collected metrics in the HBase running in the monitoring node, which is an Intel Xeon E5-2650 server. Finally, in the monitoring node, LADT analyses the stored metrics by running the algorithm programmed in Pig Latin, which calculates the correlation between the individual hosting node disk I/O operations and the aggregated disk I/O operations of all the VMs in that node, to detect the anomaly injected in Host1.

5.2 Results and Discussion

We provide experimental results for each host in time-series of total disk I/O operations per second (IOPS) and correlation coefficient values between the node disk IOPS and aggregated disk IOPS of all the VMs in the node. We present the normalised values of the IOPS, which are the mean values in small windows of 15 seconds, including 5 samples of metrics data (the frequency of metrics collection is 3 seconds). The correlation coefficient values are calculated in correlation windows of 5 minutes, covering 5 minutes of metrics data. Hence, there are 12 correlation intervals in the 60 minutes of experiment.

The results presented in Figure 2 shows that Host1's disk IOPS remains around 400 for the first 30 minutes and starts fluctuating in the next 30 minutes as a result of the injected anomaly, whereas the aggregated disk IOPS of the VMs remains below 200 throughout the experiment. The reason for the lower values of the aggregated disk IOPS of the VMs is the use of the extra software layer of the hypervisor, which is interposed between guest operating systems and hardware (Li et al., 2013). The hypervisor multiplexes I/O devices by requiring guest operating systems to access the real hardware indirectly and hence induces an overhead in the I/O context.

The correlation analysis for Host1 in Figure 2 clearly shows that there is a strong correlation between the hosting node IOPS and the aggregated IOPS of the VMs for the first half of the experiment, where the correlation coefficient values are often above 0.5 with an average value of 0.6. However the correlation value drops below 0.0 suddenly at the sixth interval when the disk stress starts. The coefficient value remains very low throughout the second half with an average value of -0.02. This is a clear reflection of the injected anomaly in Host1, which distorts the correlation between the hosting node IOPS and aggregated IOPS of the VMs.

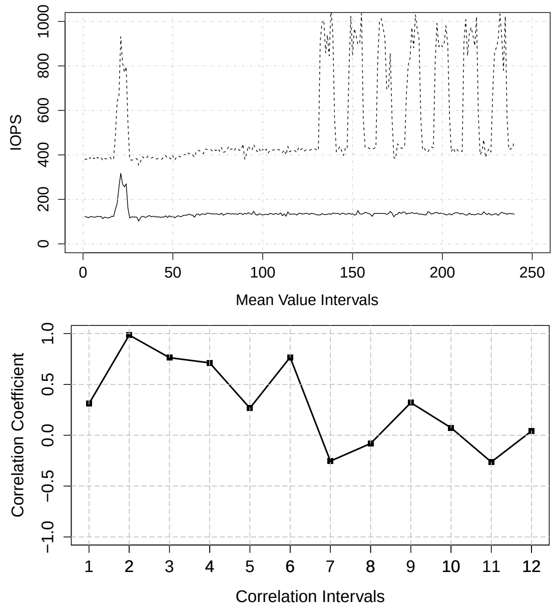

Figure 2: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host1. (Bottom) Correlation coefficients between the two time-series measurements.

Figure 3 shows that for Host2, the node IOPS and aggregated IOPS of the VMs are steady and their correlation coefficient average value is 0.4. The low average is due to the temporary drops, which happen because of the fluctuation in the overhead of I/O operations in the VMs (Li et al., 2013). Similar behaviour is exhibited by Host3 (Figure 4), where in fact there are more correlation coefficient values above 0.5 and a better average coefficient value of 0.5.

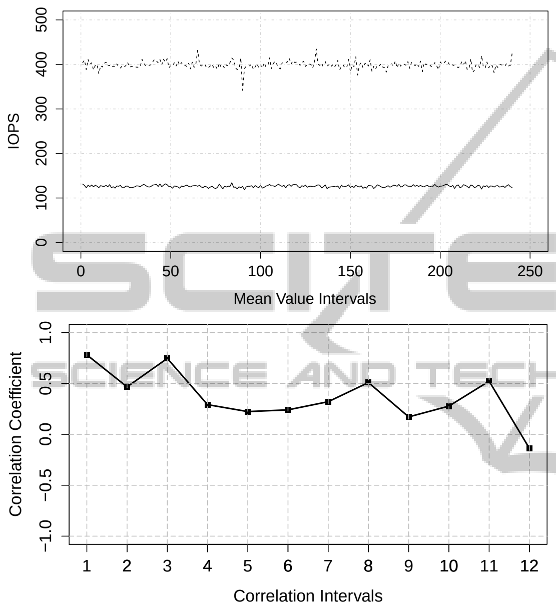

Figure 3: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host2. (Bottom) Correlation coefficients between the two time-series measurements.

Experimental results also reveal that when using LADT for disk I/O metric data collection and correlation analysis, a latency of 10 minutes is observed, where a 5 minute latency is required to hold a historical metrics data set of a 5 minute window and 5 additional minutes are used to analyse the metrics data. This latency is comparable to the latency of online data centre service troubleshooting tools (Wang, 2009) and, as yet, benefits from no optimisation of the analysis workflow at the server, which we are exploring in ongoing work.

5.3 LADT Overhead

Our next experiment assesses LADT in terms of the overhead that it introduces on hosted application VMs, because of the LADT agents that collect metrics simultaneously with the execution of application workloads. To investigate this we executed a test where a single VM runs a data serving benchmark for

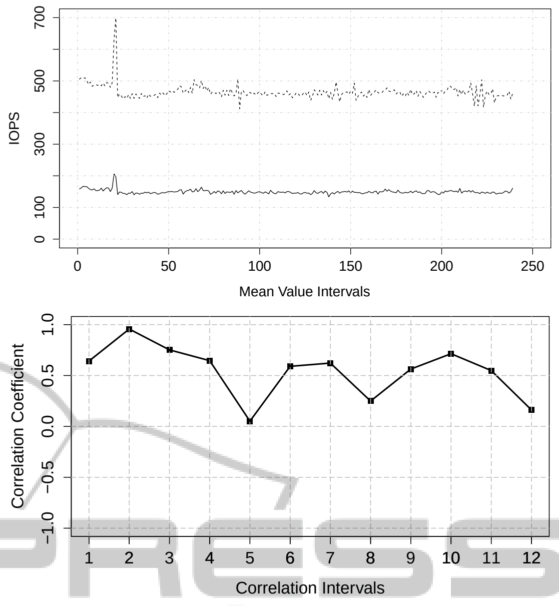

Figure 4: (Top) Time-series of node disk IOPS (dashed) and aggregated VM disk IOPS (solid) in Host3. (Bottom) Correlation coefficients between the two time-series measurements.

the duration of 10000 data operations with a rate of 200 operations/sec. With an agent running concurrently with the data serving benchmark, and during an execution interval of 50 seconds, the average update and read latencies of the benchmark were 0.21 ms and 8.28 ms, respectively. With no agent running and during the same 50 second execution interval, the average update and read latencies were 0.21 ms and 6.97 ms, respectively. In both cases we observed the expected response time from the benchmark and any differences observed in the average latencies of the update and read operations were justified by the variability introduced by the storage medium.

5.4 Further Analysis

We observe that during normal operation there is a strong correlation between the node disk IOPS and aggregated disk IOPS of the VMs in all three hosting nodes. However, the correlation becomes weaker during some intervals for short periods. This happens because in these intervals the overhead on the VM I/O operations resulting from accessing the disk indirectly via the hypervisor (Li et al., 2013) rises unpredictably and degrades the VM IOPS with respect to the corresponding node IOPS. Although the correlation coefficient value drops in some intervals even when the hosts are in a stable expected state, this drop is not as significant as it is in the case of an anomaly. Moreover, the correlation coefficient average drops below

0 when an anomaly occurs, whereas the average coefficient value ranges between 0.4 and 0.6 when the hosts are anomaly-free. We also observe that, even though all three hosts are running identical workloads in their VMs, both the IOPS and the correlation coefficient averages vary across the hosts. This again happens because of the varying overhead in the IOPS due to the hypervisor layer between the hosts and the VMs.

We conclude that the correlation coefficient values require normalisation in order to avoid false alarms for anomaly, which could arise because of a fluctuation in the overhead on VM IOPS. We detect the true anomalies by considering a larger period of system behaviour and this is done by taking the average of D consecutive coefficient values and checking if it is below the threshold value, T. The values of T and D depend on how often the user wishes to get an alarm for the anomalies. From the results, we observe the anomaly as the security attack in Host1 when the correlation coefficient value drops significantly and stays low for a longer period of time.

6 CONCLUSION AND FUTURE WORK

We presented LADT, a lightweight anomaly detection tool for Cloud data centres. LADT is based on the hypothesis that, in an anomaly-free Cloud data centre, there is a strong correlation between the node-level and VM-level performance metrics and that this correlation diminishes significantly in the case of abnormal behaviour at the node-level. The LADT algorithm raises an anomaly alarm when the correlation coefficient value between the node-level and VM-level metrics drops below a threshold level. We have demonstrated a lightweight distributed implementation of LADT using Chukwa and also demonstrated that the tool can detect node-level disk performance anomalies by correlating the hosting node IOPS with the aggregated hosted VM IOPS. Such anomalies may arise as a result of security attacks such as distributed denial-of-service (DDoS). We also demonstrated that LADT introduces acceptably low overhead, while recognizing that the implementation is amenable to optimisation along the entire path of metrics collection, data aggregation and analysis.

We intend to conduct a detailed analysis of possible attack models of the system. LADT can also detect CPU/memory/network related performance anomalies, due to the performance implications of virtualisation and resource management software stacks. We wish to explore these anomalies in more detail, using both controlled and uncontrolled set-ups (i.e. production-level set-ups with unseen anomalies) in our Cloud testbed. We plan to conduct a more thorough analysis of LADT performance, scalability and intrusion minimisation with respect to the hosted VMs. We are particularly interested in co-executing VMs with diverse characteristics (e.g. CPU-intensive, I/O-intensive), and latency sensitivity. Our aim is to understand whether adapting parameters such as the number of agent adaptors in the hosts, the frequency of data collection per VM in the hosts and the number of data aggregation tasks and cores used by collectors is necessary to keep the monitoring overhead low.

ACKNOWLEDGEMENTS

This work has been supported by the European Commission and the Seventh Framework Programme under grant agreement FP7-610711 (CACTOS).

REFERENCES

Antunes, J., Neves, N., and Verissimo, P. (2008). Detection and prediction of resource-exhaustion vulnerabilities. In *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*, pages 87–96.

Azmandian, F., Moffie, M., Alshawabkeh, M., Dy, J., Aslam, J., and Kaeli, D. (2011). Virtual machine monitor-based lightweight intrusion detection. *ACM SIGOPS Operating Systems Review*, 45(2):38–53.

Dahbur, K., Mohammad, B., and Tarakji, A. B. (2011). A survey of risks, threats and vulnerabilities in cloud computing. In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, ISWSA '11, pages 12:1–12:6, New York, NY, USA. ACM.

Ferdman, M., Adileh, A., Kocberber, O., Volos, S., Alisafaee, M., Jevdjic, D., Kaynak, C., Popescu, A. D., Ailamaki, A., and Falsafi, B. (2012). Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '12, pages 37–48, New York, NY, USA. ACM.

Hansen, S. E. and Atkins, E. T. (1993). Automated system monitoring and notification with swatch. In *Proceedings of the 7th USENIX Conference on System Administration*, LISA '93, pages 145–152, Berkeley, CA, USA. USENIX Association.

Jiang, M., Munawar, M. A., Reidemeister, T., and Ward, P. A. (2009). System monitoring with metric-correlation models: Problems and solutions. In *Pro-*

ceedings of the 6th International Conference on Autonomic Computing, ICAC '09, pages 13–22, New York, NY, USA. ACM.

Kang, H., Chen, H., and Jiang, G. (2010). Peerwatch: A fault detection and diagnosis tool for virtualized consolidation systems. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 119–128, New York, NY, USA. ACM.

Kephart, J. O. and Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1):41–50.

Kumar, V., Cooper, B. F., Eisenhauer, G., and Schwan, K. (2007). imanage: Policy-driven self-management for enterprise-scale systems. In *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, Middleware '07, pages 287–307, New York, NY, USA. Springer-Verlag New York, Inc.

Li, D., Jin, H., Liao, X., Zhang, Y., and Zhou, B. (2013). Improving disk i/o performance in a virtualized system. *J. Comput. Syst. Sci.*, 79(2):187–200.

Lou, J.-G., Fu, Q., Yang, S., Xu, Y., and Li, J. (2010). Mining invariants from console logs for system problem detection. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'10, pages 24–24, Berkeley, CA, USA. USENIX Association.

Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: A not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1099–1110, New York, NY, USA. ACM.

Oppenheimer, D., Ganapathi, A., and Patterson, D. A. (2003). Why do internet services fail, and what can be done about it? In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems - Volume 4*, USITS'03, pages 1–1, Berkeley, CA, USA. USENIX Association.

Pertet, S. and Narasimhan, P. (2005). Causes of failure in web applications. Technical report, CMU-PDL-05-109.

Rabkin, A. and Katz, R. (2010). Chukwa: A system for reliable large-scale log collection. In *Proceedings of the 24th International Conference on Large Installation System Administration*, LISA'10, pages 1–15, Berkeley, CA, USA. USENIX Association.

Rajasekar, N. C. and Imafidon, C. (2010). Exploitation of vulnerabilities in cloud storage. In *Proceedings of the First International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 122–127.

Rouillard, J. P. (2004). Refereed papers: Real-time log file analysis using the simple event correlator (sec). In *Proceedings of the 18th USENIX Conference on System Administration*, LISA '04, pages 133–150, Berkeley, CA, USA. USENIX Association.

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Sigar (2014). https://support.hyperic.com/display/sigar/home.

Tan, J., Kavulya, S., Gandhi, R., and Narasimhan, P. (2012). Light-weight black-box failure detection for distributed systems. In *Proceedings of the 2012 Workshop on Management of Big Data Systems*, MBDS '12, pages 13–18, New York, NY, USA. ACM.

The, J. P. and Prewett, J. E. (2003). Analyzing cluster log files using logsurfer. In *in Proceedings of the 4th Annual Conference on Linux Clusters*.

Virt-Top (2014). http://virt-tools.org/about/.

Vora, M. (2011). Hadoop-hbase for large-scale data. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 1, pages 601–605.

Wang, C. (2009). 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. ACM.

Ward, J. S. and Barker, A. (2013). Varanus: In situ monitoring for large scale cloud systems. In *Proceedings of the 2013 IEEE International Conference on Cloud Computing Technology and Science - Volume 02*, CLOUDCOM '13, pages 341–344, Washington, DC, USA. IEEE Computer Society.

Xu, W., Huang, L., Fox, A., Patterson, D., and Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, SOSP '09, pages 117–132, New York, NY, USA. ACM.