International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 4 | Issue 2 | ISSN : 2456-3307

ATOM : Efficient Tracking, Monitoring, and Orchestration of Cloud Resources

K Sandhya Rani *[1], Kunta Srinu[2]

[1*]sandnyakaviti@gmail.com,srinukunta78gmail.com

ABSTRACT

The emergence of Infrastructure as a Service framework brings new opportunities, that conjointly accompanies with new challenges in auto-scaling, resource allocation, and security. A elementary challenge underpinning these issues is that the continuous tracking and monitoring of resource usage within the system. during this paper, we tend to present ATOM, AN efficient and effective framework to automatically track, monitor, ANd orchestrate resource usage in an Infrastructure as a Service (IaaS) system that's wide employed in cloud infrastructure. we tend to use novel trailing methodology to ceaselessly track vital system usage metrics with low overhead, and develop a Principal part Analysis (PCA) primarily based approach to ceaselessly monitor and automatically notice anomalies supported the approximated trailing results. we tend to show a way to dynamically set the trailing threshold supported the detection results, and more, a way to regulate trailing rule to confirm its optimality beneath dynamic workloads. Lastly, once potential anomalies square measure known, we tend to use introspection tools to perform memory forensics on VMs guided by analyzed results from trailing and monitoring to spot malicious behavior within a VM. we tend to demonstrate the extensibility of ATOM through virtual machine (VM) bunch. The performance of our framework is evaluated in AN open supply IaaS system.

Keywords : Infrastructure as a Service, cloud, tracking, monitoring, anomaly detection, virtual machine introspection

INTRODUCTION

Atom is a free and open-source[4][5] text and source code editor for macOS, Linux, and Microsoft Windows[6] with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub. Atom is a desktop application built using web technologies.[7] Most of the extending packages have free software licenses and are community-built and maintained.[8] Atom is based on Electron (formerly known as Atom Shell),[9] a framework that enables cross-platform desktop applications using Chromium and Node.js.[10][11] It is written in CoffeeScript and Less.[12] It can also be used as an integrated development environment (IDE).[13][14][15][16] Atom was released from beta, as version 1.0, on 25 June 2015.[17] Its developers call it a "hackable text editor for the 21st Century".[18]

Security is another paramount system. For example, it was reported saries attacked Amazon cloud by service (DDoS) bots on user VMs by in Elasticsearch [2]. Resource usage insights to address security concerns. to constantly monitor resource usage not only for resource allocation, but in the system. Until now, the best practices for mitigating DDoS and other attacks in AWS include using CloudWatch to create simple threshold alarms on monitored metrics and alert users for potential attacks [3]. In our work we show how to detect the anomalies automatically while saving users the trouble on setting magic threshold values.

These observations illustrate that a fundamental challenge underpinning several important problems in an IaaS system is the continuous tracking and monitoring of resource usage in the system. Furthermore, several applications (e.g., security) also need intelligent and automated orchestration of system resources, by going beyond passive tracking and monitoring, and introducing auto-detection of abnormal behavior in the system, and active introspection and correction once anomaly has been identified and confirmed. This motivates us to design and implement ATOM, an efficient and effective framework to automatically track, orchestrate, and monitor resource usage in an IaaS system.

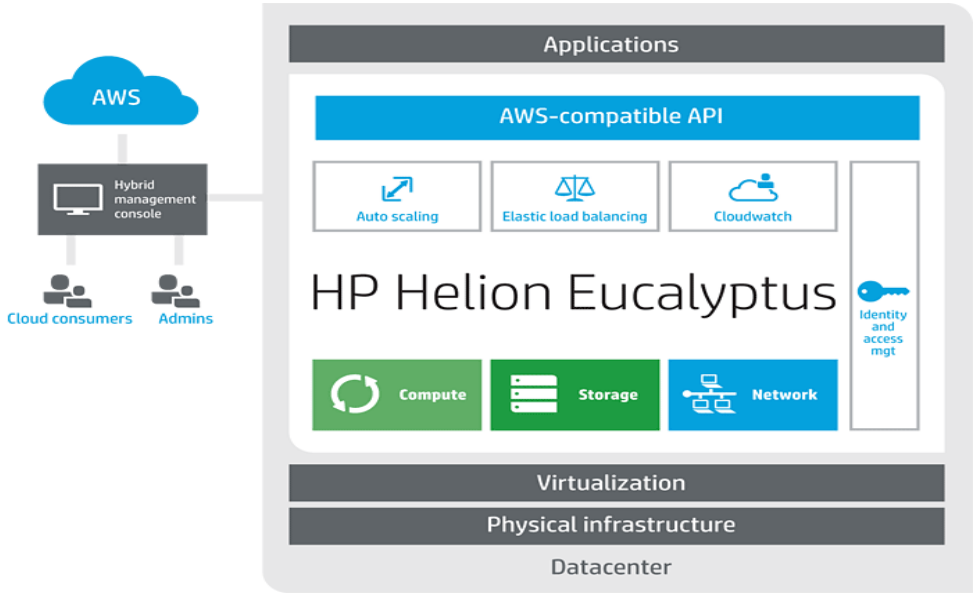

Fig. 1. A simplified architecture of Eucalyptus.

A motivating example Eucalyptus is a paid and open-source computer software for building Amazon Web Services (AWS)-compatible private and hybrid cloud computing environments, originally developed by the company Eucalyptus Systems. Eucalyptus is an acronym for Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems.[2] Eucalyptus enables pooling compute, storage, and network resources that can be dynamically scaled up or down as application workloads change.[3] Mårten Mickos was the CEO of Eucalyptus.[4] In September 2014, Eucalyptus was acquired by Hewlett-

Packard and then maintained by DXC Technology.

Eucalyptus provides an AWS-like service called CloudWatch.CloudWatch is able to monitor resource usage of each VM. To reduce overhead, such data are only collected from each VM at every minute, and then reported to the CLC through a CC. Clearly,gathering resource usage in real time introduces overhead in the system (e.g., communication overhead from a NC to the CLC). When there are plenty of VMs to monitor, the problem becomes even worse and will bring significant overhead to the system.CloudWatch addresses this problem by collecting measurements only once every minute, but this provides only a discrete, sampled view of the system status and is not sufficient to providing continuous understanding and protection of the system.

Another limitation in existing approaches like CloudWatch is that they only do passive monitoring. No active online resource orchestration is in place towards detecting system anomalies, potential threats and attacks. We observe that, e.g., in the aforementioned DDoS attack to Amazon cloud, alarming signals can be learned automatically from resource usage data, which are readily to analyze without any pre-processing like system logs [6]. Active online resource monitoring and orchestration is very useful in achieving a more secure and reliable system. Active online resource monitoring gives us the opportunities to trigger VM introspection to debug the system and figure out what has possibly gone wrong. The introspection into VMs then allows to orchestrate resource usage and allocation in the IaaS system to achieve a more secure system and/or better performance. Note that VM introspection is expensive. Without continuous tracking and online monitoring and orchestration, it is almost impossible to figure out when to do VM introspection and what specific target to introspect in a host VM. Our goal is to automate this process and trigger VM introspection only when needed. We refer to this process as resource orchestration.

ATOM introduces an online tracking module that runs at NC and continuously tracks various performance metrics and resource usage values of all VMs. The CLC is denoted as the tracker, and the NCs are denoted as the observers. The goal is to replace the sampled view at the CLC with a continuous understanding of system status, with minimum overhead.

ATOM then uses an automated monitoring module that continuously monitors the resource usage data reported by the online tracking module. The goal is to detect anomaly by mining the resource usage data. This is especially helpful for detecting attacks that could cause changes in resource usage, for example, one VM consumes all available resources and starves all other VMs running on the same physical computer [7]. The baseline for online monitoring is to simply define a threshold value for any metric of interest. Clearly, this approach is not very effective against dynamic and complex attacks and anomalies. ATOM uses a dynamic online monitoring method that is developed based on PCA. We design a PCA-based method that continuously analyzes the dominant subspace

defined by the measurements from the tracking module, and automatically raises an alarm whenever a shift in the dominant subspace has been detected. Even though PCA-based methods have been used for anomaly detection in various contexts, a new challenge in our setting is to cope with approximate measurements produced by online tracking, and design methods that are able to automatically adapting to and adjusting the tracking errors.

Lastly, virtual machine introspection (VMI) is used to detect and identify malicious behavior inside a VM. VMI techniques such as analyzing VM memory space tends to be of great cost. If we don't know where and when an attack might have happened,we will need to go through the entire memory constantly, which is clearly expensive, especially if VMs to be analyzed are so many. ATOM provides two options here. The first option is to set a threshold for each resource usage measure (the baseline as discussed above), and we consider there may be an anomaly if the reported value is beyond (or below) the threshold for that measure and trigger a VMI. This is the method that existing systems like AWS and Eucalyptus have adopted for auto scaling tasks. The second option is to use the online monitoring method in the monitoring module to automatically detect anomaly and trigger a VMI, as well as guiding the introspection to specific regions in the VM memory space based on the data from online monitoring and tracking. We denote the second method as orchestration.

That said, note that ATOM is an end-to-end framework that integrates online tracking, online monitoring, and orchestration (for VM introspection) into one framework, whereas UBL focuses on anomaly detection in performance data without the integration of tracking and orchestration. Hence, UBL is "equivalent " to the monitoring component in ATOM.

RELATED WORK

To the best of our knowledge, none of existing IaaS platforms is able to provide continuous tracking, monitoring, and orchestration of system resource usage. Furthermore, none of them is able to do intelligent, automated monitoring for a large number of VMs and carry out orchestration inside a VM.

Cloud monitoring. Most existing IaaS systems follow the general, hierarchical architecture as shown in Figure 1. Inside these sys-tems, there are imperative needs for the controller to continuously collect resource usage data and monitor system health. AWS [1] and Eucalyptus [4], [5] use CloudWatch [27] service to monitor VMs and other components in some fixed intervals, e.g., every minute. This provides cloud users a system-wide visibility into resource utilization, and allows users to set some simple threshold based alarms to monitor and ensure system health. OpenStack [28] is developing a project called Ceilometer [29], to collect resources utilization measurements. However, these approaches only provide a discrete, sampled view of the system. Several emerging startup companies such as DATADOG [30] and librato [31] could monitor in a more fine-grained granularity, provided the required softwares are installed. However, this inevitably

introduces more network overhead to the cloud, which becomes worse when the monitored infrastructure scales up. On the contrary, ATOM significantly reduces the network overhead by utilizing the optimal online tracking algorithm, while providing just about the same amount of information. Furthermore, all these cloud monitoring services offer very limited capability in monitoring and ensuring system health. UBL [8] uses collected VM usage data to train Self-Organizing Maps for anomaly prediction, which serves a similar purpose to ATOM's monitoring component. Besides the detailed comparison in Section 1, SOM requires an explicit training stage and needs to be trained by normal data, while PCA could identify what is normal directly from the history data provided normal data is the majority. Unlike UBL and ATOM which only require VM usage data, PerfCompass collects system call traces and checks the execution units being affected [32] to identify whether a VM performance anomaly is caused by internal fault like software bugs, or from an external source such as co-existing VMs.

Astrolabe [33] is a monitoring service for distributed re-sources, to perform user-defined aggregation (e.g. number of nodes that satisfy certain property) on-the-fly for the host hi-erarchy. It is intended as a "summarizing mechanism". Similar to Astrolabe, SDIMS [34] is another system that aggregates information about large-scale networked systems with better scal-ability, flexibility, and administrative isolation. Ganglia [35] is a general-purpose scalable distributed monitoring system for high performance computing systems which also has a hierarchical design to monitor and aggregate all the nodes and has been used in many clusters. These efforts are similar to the CloudWatch module currently used in AWS/Eucalyptus, and they reduce monitoring overhead by simple aggregations. While the purpose of ATOM's tracking module is to reduce data transfer, but it does so using online tracking instead of simply aggregating which delivers much more fine-grained information.

STAR [36] is a hierarchical algorithm for scalable aggregation that reduces communication overhead by carefully distributing the allowed error budgets. It suites systems like SDIMS [34] well. InfoEye [37] is a model-based information management system for large-scale service overlay networks through a set of monitoring sensors deployed on different overlay nodes with reduced overhead achieved by ad-hoc conditions filters. InfoTrack

[38] is a monitoring system that is similar to ATOM's tracking module, in that it tries to minimize continuous monitoring cost with most information precision preserved, by leveraging temporal and spatial correlation of monitored attributes, while ATOM uti-lizes an optimal online tracking algorithm that is proved to achieve the best saving in network cost without any prior knowledge on the data. MELA [39] is a monitoring framework for cloud service which collects different dimensions of data tailored for analyzing cloud elasticity purpose (e.g. scale up and scale down). ATOM may use MELA to collect, track, and monitor different types of metrics than those already available through CloudWatch.

Cloud security. IaaS system also brings us a new set of secu-rity problems. Leading cloud providers have developed advanced mechanism to ensure the security of their IaaS systems. AWS [40] has many built-in security features such as firewalls, encrypted storage and security logs. OpenStack uses a security component called Keystone [41] to do authentication and authorization. It also has security rules for network communication in its network component Neutron [42]. Other IaaS platforms have similar se-curity solutions, which are mainly firewalls and security groups. Nevertheless, it is still possible that hackers could bypass known security policies, or cloud users may accidentally run some mali-cious software. It is thus critical to be able to detect such anomaly in near real-time to avoid leaving hackers plenty of time to cause significant damage. Hence we need a monitoring solution that could actively detect anomaly, and identify potentially malicious behavior over a large number of VM instances. AWS recently adopts its CloudWatch service for DDoS attacks [3], but it re-quires user to check historical data and set a ”magic value” as the threshold manually, which is unrealistic if user’s underlying workloads change frequently.

In contrast, ATOM could automatically learn the normal behavior from previous monitored data, and detect more complex attacks besides DDoS attacks using PCA. PCA has been widely used to detect anomaly in network traffic volume in backbone networks [12], [13], [17], [43], [44], [45]. As we have argued in Section 4.1, adapting a PCA-based approach to our setting has not been studied before and presented significant new challenges.

The security challenges in IaaS system were analyzed in [7], [46], [47], [48]. Virtual machine attacks is considered a major security threat. ATOM’s introspection component leverages existing open source VMI tools such as Stackdb [10] and Volatility [18] to pinpoint the anomaly to the exact process.

VMI is a well-known method for ensuring VM security [49], [50], [51], [52]. It has also been studied for IaaS systems [53], [54], [55]. However, to constantly secure VM using VMI tech-nique, the entire VM memory needs to be traversed and analyzed periodically. It may also require the VM to be suspended in order to gain access to VM memory. Blacksheep [19] is such a system that detects rootkit by dumping and comparing groups of similar machines. Though the performance overhead is claimed to be acceptably low to support real-time monitoring, clearly user programs will be negatively affected. Another solution was suggested [56] for cloud users to verify the integrity of their VMs. However, this is not an “active detection and reaction” system. In contrast, ATOM enables triggering VMI only when a potential attack is identified, and it also helps locate the relevant memory region to analyze and introspect much more effectively and efficiently using its orchestration component.

PROPOSE SYSTEM

THE ATOM FRAMEWORK

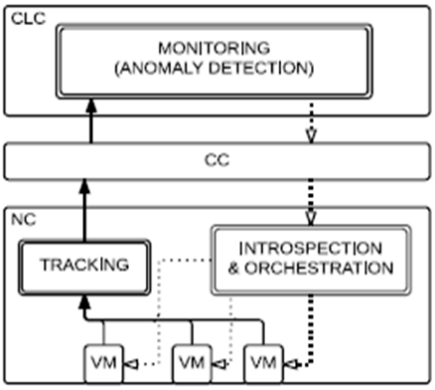

Figure 2 the ATOM framework

CC and one NC are shown in this example. ATOM adds three components to an IaaS system like AWS and Eucalyptus:

(1) Tracking component: ATOM adapts the optimal online tracking algorithm for one-dimension online tracking inside the monitoring service on NCs. This dramatically reduces the over-head used to monitor cloud resources and enables continuous measurements to CC and CLC;

(2) Monitoring component (anomaly detection): ATOM adds this component in CLC to analyze tracking results by the tracking component, which provides continuous resource usage data in real time. It uses a modified PCA method to continuously track the divided subspace, as defined by the multi-dimensional values from the tracking results, and automatically detect anomaly by identi-fying notable shift in the interesting subspace. It also generates anomaly information for further analysis by the orchestration com-ponent when this happens. The monitoring component also adjusts the tracking threshold from the tracking component dynamically online based on the data trends and a desired false alarm rate.

(3) Orchestration component (introspection and debugging): when a potential anomaly is identified by the monitoring compo-nent, an INTROSPECT request along with anomaly information is sent to the orchestration component on NC, in which VMI tools (such as LibVMI [9]) and VM debugging tools (such as StackDB [10]) are used to identify the anomalous behavior inside a VM and raise an alarm to cloud users for further analysis.

ORCHESTRATION COMPONENT

The monitoring component in Section 4 detects the abnormal state and identifies which measurement on which VM might be respon-sible. In this section, we describe how orchestration component is able to automatically mitigate the malicious behavior after an anomaly is detected.

Modern IaaS cloud vendors offer services mostly in the form of VMs, which makes it critical to ensure VM security in order to attract more customers. VMI technique has been widely studied to introspect VM for security purpose. There are also several popular open source general-purpose VMI tools such as LibVMI [9], Volatility [18], and StackDB [10], for researchers to explore and develop more sophisticated applications. LibVMI has many basic APIs that support memory read and write on live memory. Volatility itself supports memory forensics on a VM memory snapshot file, and it has many Linux plugins that are ready to use. StackDB is designed to be a multi-level debugger, while also serves well as a memory-forensics tool. Other more sophisticated techniques developed for special-purpose VMI anomaly detection are generally based on these tools. Blacksheep [19], for instance, utilizes Volatility and specifically developed plug-ins to imple-ment a distributed system for detecting anomalies inside VMs among groups of similar machines. However, as most other VMI strategies to secure VMs, it needs to dump the whole memory space of the target VM, and then analyze each piece, typically by comparing with what's defined a "normal" state. Thus to protect VMs in real time, the whole memory space needs to be analyzed constantly, introducing much overhead into the production system.

ATOM implements its orchestration component based on Volatility (with LibVMI plug-in for live introspection) and StackDB. A crucial difference with other systems is that, ATOM only introspects the VM when an anomaly happens, and only on the relevant memory space of the suspicious VMs. The monitoring component in ATOM serves as a trigger to inform VMI tools when and where to do introspection. The anomalies are found by analyzing previously monitored resource usage data, in monitoring component, which is much more lightweight than analyzing the whole memory space. Then the metrics identification process in monitoring component could locate which dimensions are suspi-cious, indicating the relevant metrics on some particular VMs. This information is sent to orchestration component along with a VMI request, which then only introspects the relevant memoy

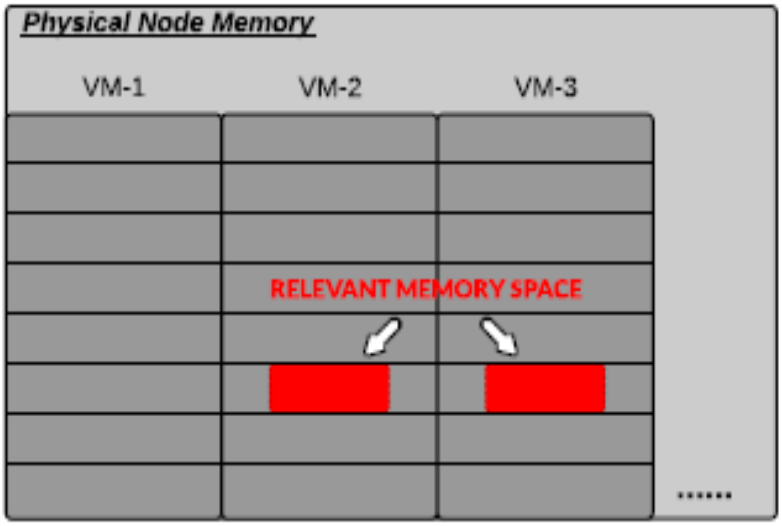

Fig. 3. Memory space introspected by ATOM.

After the orchestration component identifies potential abnor-mal processes, an alarm is raised with associated information identified by VMI tools. The alarm and such information are provided to the VM user. If user confirms this as an abnormal behavior, ATOM is able to terminate the malicious processes inside a VM instance by using tools like StackDB [10]. StackDB could be used to debug, inspect, modify, and analyze the behavior of running programs inside a VM instance. To kill a process, it first finds the task_struct object of the running process using process name or id, and then passes in SIGKILL signal. Next time the process is being scheduled, it is killed immediately.

Although the anomalies that could be detected by ATOM is limited compared with other systems which analyze the whole memory space, we argue the framework of ATOM could be easily extended to detect more complex attacks. First, more metrics could be easily added to monitor for each VM. Also, many other auto-debugging tools could be developed, which are useful to find various kinds of attacks and perform different desirable actions.

Note that killing the identified, potentially malicious process is just one possible choice provided by ATOM, which is performed only if user agrees to (ATOM is certainly able to automate this as well if desired). Alternatives could be to terminate the network connections or to close file handles. A more sophisticated way is to study a rich dataset of known attacks (e.g., Exploits Database) and design rule-based approaches to mitigate attacks based on different patterns. We refer these active actions, together with introspection, as ATOM's orchestration module. Orchestration in ATOM can be greatly

customized to suite the needs for different tasks, such as identification of different attacks, and dynamic resource allocation in an IaaS system.

VM CLUSTERING

ATOM enables a continuous understanding of the VMs in an IaaS system. In addition to anomaly detection, this framework is also useful for many other decision making and analytics applications. Hence, in addition to using a PCA-based approach in the monitoring component, we will demonstrate that it is possible to design and implement a VM clustering module to be used in the monitoring component.

The objective of VM clustering is to cluster a set of VMs into different clusters so that VMs with similar workload characteris-tics end up in the same group. This operation assists making load balancing decisions, as well as developing customized, fine-tuned monitoring modules for each cluster. For instance, a cloud provider

may want to evenly distribute the VMs having similar resource usage patterns to different physical nodes, in order to make sure the physical resources are fully utilized and fewer VMs may suffer from resource starvation. In another example, we may want to use different anomaly detection techniques for VMs running a database server workload than those running a web server.

The basic idea of our proposed approach is as follows. The monitoring component in ATOM, using its PCA-based approach, transforms the original coordinates to a new coordinate system where the principal components (PCs) are ordered by the amount of variations on each direction (as explained in Figure 3). Thus, if two VMs share similar workloads, the directions of the corre-sponding PCs between the two should also be similar. That said,

Step 1. On CLC, a data matrix for each VM is maintained, where the columns are metric types and rows are time instances (i.e., a t d^0 matrix for each VM with a sliding window of t), and is updated over time.

Step 2. ATOM performs a PCA on each VM data matrix without standardization; since for clustering purposes, not only the variations on each direction is important, but also the average usage on each dimension. For example, a VM having a disk usage that oscillates between 10,000 and 20,000 bytes is obviously not the same as one having oscillation between 100 and 200 bytes on the same dimension; whereas a standardization procedure which first performs mean-center and then normalization will make the two oscillations look similar.This step yields a set of PCs for each VM. The direction of each PC is denoted by the corresponding eigen vector while the variation is shown by the associated eigen value.

Step 3. Suppose VM1 has eigen vectors ($v_{11};v_{12};:::$) and corresponding eigen values ($l_{11};l_{12};:::$), while VM2 has ($v_{21};v_{22};:::$) and ($l_{21};l_{22};:::$). We measure the distance be-tween two directions using cosine distance; defined as (1 cosine similarity). Intuitively, the bigger the angle between two directions (the less similar they are), the smaller their cosine similarity is, hence the larger the cosine distance becomes.

Finally, the distance between the two VMs is defined as:

$$\text{VMdist}(\text{VM1}, \text{VM2}) = |\lambda_{11} - \lambda_{21}|(1 - \frac{\mathbf{v}_{11} \cdot \mathbf{v}_{21}}{|\mathbf{v}_{11}| \cdot \mathbf{v}_{21}}) + |\lambda_{12} - \lambda_{22}|(1 - \frac{\mathbf{v}_{12} \cdot \mathbf{v}_{22}}{|\mathbf{v}_{12}| \cdot \mathbf{v}_{22}}) + \cdots.$$

of each corresponding pair of eigen vectors from VM1 and VM2, weighted by the difference of the corresponding eigen values to ensure that the variations do not differ a lot.

Step 4. Using VMdist as the distance measure between any two VMs, we use DBSCAN [20] to cluster similar VMs together. DBSCAN is a threshold-based (aka density based) clustering algorithm which requires two parameters: e which is the density threshold, and minPts which is the number of minimum points to form a cluster. DBSCAN expands a cluster from an un-visited data point towards all its neighboring points provided the distance is within e, and then recursively expands from each of the neighboring point. Points are marked as an outlier if the number of points in their cluster is fewer than minPts. Compared with other popular clustering methods like k-means, density-based clustering algorithm does not require the prior-knowledge on the number of clusters, neither does it need to iteratively compute an explicit "centroid" and re-cluster at every iteration. By default, ATOM sets minPts=10, and computes the thresh-old value e using a sampling based approach. More specifi-cally, we randomly select n pairs of VMs and compute their VMdist. We sort the n VMdist values, and set e = VMdist_i if $\text{VMdist}_{i+1} > 5$ VMdist_i. The intuition is that for any point, the distance to a point in a different cluster is much longer than the distance to a point in the same cluster, and we want to find a large enough "inner cluster" distance and use it as the threshold value e to determine whether two points belong to the same cluster.

CONCLUSION

We exhibit the ATOM-framework that can be effectively incorporated into a standard IaaS framework to give mechanized, constant tracking, monitoring, and orchestration of framework asset use in about ongoing. ATOM is to a great degree valuable for abnormality identification, auto-scaling, and dynamic asset designation and load adjusting in IaaS frameworks. Intriguing future work incorporates expanding ATOM for more advanced asset orchestration and joining the barrier against considerably more intricate assaults in ATOM.

REFERENCES

[1]. Amazon. http://www.aws.amazon.com/. Accessed Nov. 5, 2016.
[2]. ITWORLD. http://www.itworld.com/security/428920/attackers-install-ddos-bots-amazon-cloud-exploiting-elasticsearch-weakness. Accessed Nov. 5, 2016.
[3]. Amazon. AWS Best Practices for DDoS Resiliency. https://d0.awsstatic.com/whitepapers/DDoS White Paper June2015.pdf. Accessed Nov. 5, 2016.
[4]. Eucalyptus. http://www8.hp.com/us/en/cloud/helion-eucalyptus.html. Accessed Nov. 5, 2016.
[5]. D Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Yous-eff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in CCGRID, 2009.
[6]. M Du and F. Li, "Spell: Streaming parsing of system event logs," in ICDM, 2016.

[7]. W Dawoud, I. Takouna, and C. Meinel, "Infrastructure as a service security: Challenges and solutions," in INFOS, 2010.
[8]. D J. Dean, H. Nguyen, and X. Gu, "UBL: Unsupervised behavior learn-ing for predicting performance anomalies in virtualized cloud systems," in ICAC, 2012.
[9]. LibVMI. http://libvmi.com/. Accessed Nov. 5, 2016.
[10]. D. Johnson, M. Hibler, and E. Eide, "Composable multi-level debugging with Stackdb," in VEE, 2014.
[11]. K. Yi and Q. Zhang, "Multi-dimensional online tracking," in SODA, 2009.
[12]. H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for traffic anomaly detection," in SIGMETRICS Performance Evaluation Review, 2007.
[13]. A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," in SIGCOMM, 2004.
[14]. V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift, "Resource-freeing attacks: improve your cloud performance (at your neighbor's expense)," in CCS, 2012.
[15]. W. Li, H. H. Yue, S. Valle-Cervantes, and S. J. Qin, "Recursive PCA for adaptive process monitoring," Journal of process control, 2000.
[16]. J. E. Jackson and G. S. Mudholkar, "Control procedures for residuals associated with principal component analysis," Technometrics, 1979.
[17]. L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft, "In-network PCA and anomaly detection," in NIPS, 2006.
[18]. Volatility. http://www.volatilityfoundation.org/. Accessed Nov. 5, 2016.
[19]. A. Bianchi, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Blacksheep: detecting compromised hosts in homogeneous crowds," in CCS, 2012.
[20]. M. Ester, H.-P. Kriegel, J. Sander, X. Xu et al., "A density-based algorithm for discovering clusters in large spatial databases with noise." in KDD, 1996.
[21]. D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "OLTP-Bench: An extensible testbed for benchmarking relational databases," PVLDB, 2013.
[22]. StackDB. http://www.flux.utah.edu/software/stackdb/doc/all.html#using-eucalyptus-to-run-qemukvm. Accessed Nov. 5, 2016.
[23]. I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approx-hadoop: Bringing approximations to mapreduce frameworks," in ASP-LOS, 2015.
[24]. M. T. Al Amin, S. Li, M. R. Rahman, P. T. Seetharamu, S. Wang, T. Abdelzaher, I. Gupta, M. Srivatsa, R. Ganti, R. Ahmed et al., "Social trove: A self-summarizing storage service for social sensing," in ICAC, 2015.
[25]. J. Kelley, C. Stewart, N. Morris, D. Tiwari, Y. He, and S. Elnikety, "Mea-suring and managing answer quality for online data-intensive services," in ICAC, 2015.
[26]. X. Wang, U. Kruger, and G. W. Irwin, "Process monitoring approach using fast moving window PCA," Industrial & Engineering Chemistry Research, 2005.
[27]. Amazon. Amazon cloudwatch. http://aws.amazon.com/cloudwatch/. Accessed Nov. 5, 2016.
[28]. OpenStack. http://www.openstack.org/. Accessed Nov. 5, 2016.
[29]. Openstack ceilometer. https://wiki.openstack.org/wiki/Ceilometer. Accessed Nov. 5, 2016.
[30]. DATADOG. https://www.datadoghq.com/. Accessed Nov. 5, 2016.
[31]. librato. https://www.librato.com/. Accessed Nov. 5, 2016.
[32]. D. J. Dean, H. Nguyen, P. Wang, and X. Gu, "Perfcompass: toward runtime performance anomaly fault localization for infrastructure-as-a-service clouds," in HotCloud, 2014.
[33]. R. Van Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," TOCS, 2003.
[34]. P. Yalagandula and M. Dahlin, "A scalable distributed information management system," in SIGCOMM, 2004.
[35]. M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system:

design, implementation, and experience," Parallel Computing, 2004.

[36]. N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang, "Star: Self-tuning aggregation for scalable monitoring," in VLDB, 2007.

[37]. J. Liang, X. Gu, and K. Nahrstedt, "Self-configuring information man-agement for large-scale service overlays," in INFOCOM, 2007.

[38]. Y. Zhao, Y. Tan, Z. Gong, X. Gu, and M. Wamboldt, "Self-correlating predictive information tracking for large-scale production systems," in ICAC, 2009.

[39]. D. Moldovan, G. Copil, H.-L. Truong, and S. Dustdar, "MELA: Moni-toring and analyzing elasticity of cloud services," in CloudCom, 2013.

[40]. Amazon. Aws security center. http://aws.amazon.com/security/. Ac-cessed Nov. 5, 2016.

[41]. OpenStack. OpenStack Keystone. http://docs.openstack.org/developer/ keystone/. Accessed Nov. 5, 2016.

[42]. OpenStack Neutron. https://wiki.openstack.org/wiki/Neutron. Ac-cessed Nov. 5, 2016.

[43]. X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and identification of network anomalies using sketch subspaces," in IMC, 2006.

[44]. Y. Liu, L. Zhang, and Y. Guan, "Sketch-based streaming PCA algorithm for network-wide traffic anomaly detection," in ICDCS, 2010.

[45]. L. Huang, X. Nguyen, M. Garofalakis, J. M. Hellerstein, M. I. Jordan, A. D. Joseph, and N. Taft, "Communication-efficient online detection of network-wide anomalies," in INFOCOM, 2007.

[46]. A. S. Ibrahim, J. H. Hamlyn-harris, and J. Grundy, "Emerging security challenges of cloud virtual infrastructure," in APSEC 2010 Cloud Work-shop, 2010.

[47]. L. M. Vaquero, L. Rodero-Merino, and D. Moran,´ "Locking the sky: a survey on iaas cloud security," Computing, 2011.

[48]. C. R. Li, D. Abendroth, X. Lin, Y. Guo, H. wook Baek, E. Eide, R. Ricci, and J. K. V. der Merwe, "Potassium: Penetration testing as a service," in SoCC, 2015.

[49]. T. Garfinkel, M. Rosenblum et al., "A virtual machine introspection based architecture for intrusion detection." in NDSS, 2003.

[50]. J. Pfoh, C. Schneider, and C. Eckert, "A formal model for virtual machine introspection," in ACM workshop on Virtual machine security, 2009.

[51]. B. Dolan-Gavitt, T. Leek, M. Zhivich, J. Giffin, and W. Lee, "Virtuoso: Narrowing the semantic gap in virtual machine introspection," in IEEE Symposium on Security and Privacy, 2011.

[52]. Y. Fu and Z. Lin, "Space traveling across vm: Automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection," in IEEE Symposium on Security and Privacy, 2012.

[53]. A. S. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy, "Cloudsec: a security monitoring appliance for virtual machines in the iaas cloud model," in NSS, 2011.

[54]. F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtu-alization," in SOSP, 2011.

[55]. H. W. Baek, A. Srivastava, and J. Van der Merwe, "CloudVMI: Virtual machine introspection as a cloud service," in IC2E, 2014.

[56]. B. Bertholon, S. Varrette, and P. Bouvry, "Certicloud: a novel tpm-based approach to ensure cloud iaas security," in IEEE Cloud Computing, 2011.

[57]. M. Du and F. Li, "ATOM: automated tracking, orchestration and moni-toring of resource usage in infrastructure as a service systems," in IEEE BigData, 2015.

AUTHOR DEATILAS

K SANDHYA RANI is working an assistant professor in VIGNAN'S LARA INSTITUTE OF TECHNOLOGY & SCIENCE... Vadlamudi-522213 Guntur Dist. She has Experience in the teaching field For 2 years and her interested in research area data mining.

KUNTA SRINU IS PURSUING MCA. Degree from VIGNAN'S LARA INSTITUTE OF TECHNOLOGY &SCIENCE, Vadlamudi, Guntur, Andhra Pradesh, India