iCSI: A Cloud Garbage VM Collector for Addressing Inactive VMs with Machine Learning

In Kee Kim*, Sai Zeng†, Christopher Young†, Jinho Hwang†, and Marty Humphrey*
*University of Virginia, Department of Computer Science, ik2sb@virginia.edu, humphrey@cs.virginia.edu
†IBM T.J. Watson Research Center, {saizeng, ccyoung, jinho}@us.ibm.com

***Abstract*—According to a recent study, 30% of VMs in private cloud data centers are "comatose", in part because there is generally no strong incentive for their human owners to delete them at an appropriate time. These inactive VMs are still scheduled and executed on physical cloud resources, taking valuable access away from productive VMs. In an extreme, cloud infrastructure may deny legitimate requests for new VMs because capacity limits have been hit. It is not sufficient for cloud infrastructure to identify such inactive VMs by monitoring resource utilization (e.g., CPU utilization) – e.g., management processes (e.g. virus-scan, software update) on inactive VMs often consume high CPU and memory resources, and active VMs with lightweight jobs (e.g. text editing) show almost zero resource utilization. To properly detect and address such inactive VMs, we present iCSI: a cloud garbage VM collector to improve resource utilization and cost efficiency of enterprise data centers. iCSI includes three main components; a lightweight data collector, a VM identification model and a recommendation engine. The data collector periodically gathers primitive information from VMs. The identification model infers the purpose of a VM from the data collection and extracts the most relevant features associated with the purpose. The recommendation engine offers proper actions to end users i.e., suspending or resizing VMs. In this prototype phase, iCSI is deployed into multiple data centers in IBM and manages more than 750 production VMs. iCSI achieves 20% better accuracy (90%) in identifying active/inactive VMs compared with state-of-the-art methods. With recommendations to end users, our estimation results show that iCSI can improve internal cost efficiency with 23% and resource utilization more than 45%.**

***Index Terms*—Cloud Computing; IaaS Garbage VM Collection; Identifying Inactive VM; Data Center Management; Support Vector Machine**

I. Introduction

Private clouds have been widely adopted for many enterprise companies and are becoming essential for servicing both internal[1] and external[2] IT activities [1, 2]. Clearly, cloud computing offers many benefits as compared to traditional in-house computing infrastructure. These benefits include efficient resource management via virtualization [3, 4], pay-as-you-go model [5], and simplified process to obtain and release VM (Virtual Machine) instances [6].

However, surprisingly, such infrastructures are not effectively utilized based on a recent study [7]. The study shows that approximately 30% of VMs in the U.S. data centers are "comatose". These inactive (or "zombie") VMs most likely were used at some point in the past but have not been suspended, been snapshot, or been terminated for a variety of reasons. For example, in the public cloud, users tend to more directly see and understand the financial ramifications of continuing to run VMs past their useful stage; in a company's private cloud, such financial ramifications might not be as directly felt by the employee. The inactive VMs dramatically reduce available resource capacity, hurt internal cost efficiency, and show poor utilization of such infrastructure. In an extreme case, these VMs may result in cloud infrastructure being unable to allocate resources for new VMs.

"Cloud garbage VM collection" [8–11] is a mechanism to detect inactive VMs and properly manage them to improve utilization and cost efficiency of private cloud infrastructure. Identification of these inactive VMs is the first step to design a cloud garbage VM collector. Intuitively, resource monitoring [12, 13] might be attractive to identify these inactive VMs but unfortunately this approach does not always work for many real-world use cases. For example, inactive VMs often appear active since VM management processes (e.g. automatic software update, virus-scan) consume high CPU and memory resources on inactive VMs for a short period of time. And active[3] VMs may seem to be inactive when only performing lightweight operations (e.g. text editing) that consume near-negligible amount of resources. As such, it is difficult to accurately differentiate inactive and active VMs by relying on resource monitoring information.

To correctly identify and address these inactive VMs, we create **iCSI** (**i**nactive **C**loud **S**erver **I**dentification) system; a cloud garbage VM collector to improve utilization and cost efficiency of private cloud data centers. iCSI has three main components: a lightweight data collector, a VM identification model, and a recommendation engine. The lightweight data collector periodically gathers primitive information from VMs. The primitive information can be easily obtained by native commands supported by most operating systems. In a public cloud setting, such measurement mechanisms via such OS modifications would generally be prohibited because of privacy concerns [14–18]. However, in this work, we leverage the nature of the private cloud environment, in which such modifications are supported and generally considered to be in-line with enterprise goals as a whole, to measure and collect such information. In other words, our approach is that a

[1]Internal activities include "research" and "development" activities in an organization.

[2]External activities indicate "production service" to end users.

[3]In this paper, we use active VM as a contrary of inactive VM. Active VMs are used for productive works in their organization.

cooperative framework in warranted, in which users want help identifying VMs to "reclaim", and users are willing to tolerate additional intra-VM reporting measures to support this.

With the data collection, we leverage an VM identification model from our previous work [19] in order to create iCSI system. This model is used to identify active or inactive VMs from given VM dataset. The VM identification model first determines the purpose of VMs via its running processes and extracts the most important features for the identification. Direct user feedback with random sampling is used to find specific features that are highly correlated with identifying active/inactive VMs with different purposes. A supervised learning model is employed to identify these active and inactive VMs. This learning model dynamically selects the most proper features according to the purposes of VMs. To reduce misclassification rate, the identification model performs an affinity analysis of network dependencies among target VMs and adjusts identification results. With this identification model, the recommendation engine offers proper recommendations to end users. The recommendation includes suspending VMs, suspending VMs, resizing VMs, and others.

To evaluate the performance of this cloud garbage VM collector, iCSI is deployed into multiple data centers for IBM research division and monitors more than 750 production VMs. We measure the identification accuracy, cost efficiency, and utilization improvement of iCSI compared with two state-of-the-art approaches. The experiment results show that our model has a 20% higher accuracy than other state-of-the-art approaches. Also, iCSI can improve cost efficiency with 23% and shows increased resource utilization with 45%.

The contributions of this paper are:

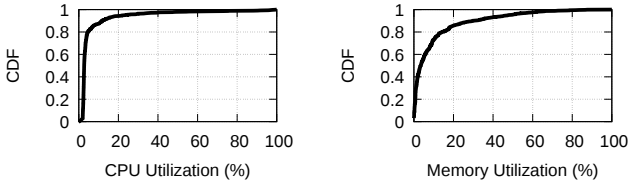
- We design an end-to-end cloud garbage VM collector that identifies inactive VMs and provides proper management plans for these inactive VMs. This system improves cost efficiency and resource utilization of private cloud data centers with the management plans.
- We evaluate the performance of iCSI with real-world cloud dataset with 750 VMs. iCSI can successfully identify inactive VMs with 90% of accuracy.
- We also show that iCSI can save 23% of VM cost and improve resource utilization with more than 45% with proper recommendations for VM management.

Note that this work uses an important finding from our previous work [19] that was focused on creating a model that identifies activeness or inactiveness of VMs in private data centers. This model is embedded as a component into iCSI system. While the model provides important information on VM identification, this work further concentrates on showing that how effectively iCSI system can improve data center management for private clouds. i.e., VM cost and utilization.

The rest of this paper is organized as follows: Section II describes background of cloud garbage VM collector. Section III reviews state-of-the-arts approaches for the cloud garbage VM collections. Section IV describes how we analyze VM data to identify active and inactive VMs. Section V contains the design of iCSI system and Section VI provides the performance evaluation results of iCSI and Section VII concludes this paper.

II. Background

With the promise of providing high availability, scalability, and also efficiency, enterprises and research institutions continue to move to public/private clouds, and cloud providers expand the number of data centers globally at the same time [20]. The revenues of major cloud providers such as Amazon Web Services (AWS) [21], Microsoft Azure [22], Google Cloud Platform [23], and IBM SoftLayer [24] have been largely growing, meaning that more VMs have been provisioned and actively running. However, the overall infrastructure resource usage has been declined as the number of inactive VMs have been increasing. Figure 1 illustrates the VM utilization in an enterprise data center with 200 VMs. More than 90% of VMs use less than 20% of CPU and 35% of memory [25].



(a) CDF of CPU Utilization. (b) CDF of Memory Utilization.

Fig. 1. CDF of VM Utilization in an Enterprise Data Center (measured with 200 VMs).

As customers want to get help from cloud providers to efficiently manage computing resources, cloud providers offer monitoring services such as CloudWatch [26] for resource usage [27], and network traffic [28]. Customers are able to scale in/out based on the monitoring results when VMs are registered in a scaling group [29, 30], but they are limited to manage only replicated resources. Some cloud providers offer a reactive approach for cloud garbage VM collection to recommend to delete inactive servers. The reason why it remains in passive mode is because the recommendations are only based on the resource activities so that the accuracy rate is not high. In this circumstance, customers are not able to intelligently decide which VMs can safely be terminated or suspended in order to reduce operational expenditure.

Since the cloud providers do not properly support the cloud garbage VM collection, customers come up with proprietary approaches for the private clouds. Two main approaches are graph-based and rule-based garbage VM collections. The graph-based method uses a graph reference model to identify connected servers (i.e., graph components) through network connections and propagates the significance of known important servers [8, 9]. On the other hand, the rule-based method defines rules for terminating or suspending VMs based on the experience of resource behaviors, and VM users are required to perform management actions when the VMs are matched with the predefined rules [10, 31, 32]. Both methods are limited in that the dynamic cloud workloads cannot be characterized as

a certain number of behaviors and there are many servers that are not connected with other servers at all.

Instead of relying on the abstract representations such as graphs or rules, it would make more sense to closely look at the features of each VM and characterize each server in different manner because each VM may have distinguishable behaviors according to different purposes. As not all VMs have the same purpose and corresponding behaviors, the classification of the purposes may be able to categorize the behaviors of the certain purpose types. Then, each purpose type can characterize VM behaviors, and determine a VM is active or not. Also, since observing and analyzing features at a certain point of time do not reflect the comprehensive behavior, the cloud garbage VM collectors should consider the time series of certain behaviors such as login activities, resource usages, and network connectivities. We explain how we apply this idea for designing a new cloud garbage VM collector in the following sections.

III. Related Work

There are three approaches for the infrastructure garbage collection with active/inactive VM identification: graph-based [8, 9], rule-based [10, 11, 31, 32], and utilization-based [12, 13, 33]. The graph-based approach leverages a network affinity model that references connectivity between VMs, and propagates importance of VMs to connected ones. The rule-based approach makes use of predefined rules (or annotations) to identify any VMs that violate the rules. The utilization-based approach leverages system-level statistics (e.g. CPU, memory, network) to detect inactive resources/VMs and consider VMs are inactive when utilization of the VMs is lower than the predefined threshold.

Graph-based approach: Pleco [8] is a tool to detect inactive VMs in IaaS clouds. Similar to memory garbage collector [34] (in many programming languages such as LISP, Java, and C#) which identifies garbage objects by examining object references, Pleco constructs a VM reference model according to a dependency of applications (network affinity model [35–37]), assigns weight to each reference, and calculates a confidence level for identified inactive VMs.

Garbo [9] generates a directed acyclic graph with connectivity (dependency) of resources and adopts this idea to garbage VM collection mechanism. However, in many cases, standalone cloud resources – *having no dependency with other cloud resources* – cannot necessarily be identified as unused resources by only using resource dependencies. For example, standalone cloud resources often have very important dataset for an organization (e.g., key files) and many VM instances often have all tier components (e.g., LAMP[4]) for services and they are unlikely to have any connections with other cloud resources.

Rule-based approach: Netflix Janitor Monkey [10] identifies unused resources based on a set of rules defined by data center operators. This set of rules is focused on an "age factor" of cloud resources on AWS. For instance, Janitor Monkey marks resources inactive if they are not used for more than a predefined rules (e.g., 3 days). However, defining the proper rules is a challenging task for the data center/cloud operators because it is very hard to predict when the (individual) resources will be used again. In addition, Janitor Monkey does not consider any connectivity (or dependency) among resources as well as the significance of the target resources, it can hardly adapt to dynamic scenarios for cloud management.

Poncho [31] is used to maintain Magellan cloud [38] at Argonne National Laboratory (open-stack based). The core idea is to use annotation and notification. Resource owners are required to provide detailed annotations (their own rules) for their resources. The rules are not meant for global uses, but defined for specific workloads. The examples of annotations include $reboot\_when$, $terminate\_when$, or $snapshot\_on\_terminate$. $terminate\_when$ means that the VM can be terminated after X hours of execution, and this implicitly offers VM to be used for another job. With predefined annotations by resource owners, inactive time of cloud resources can be minimized as well as resource utilization of the entire infrastructure can be improved, but each time users create VMs, they have to annotate VMs and the annotations need to be changed as the purpose of VMs is changed.

CloudVMI [32] offers a VMI (Virtual Machine Introspection) [39] capability to public cloud users. This CloudVMI has a "garbage VM collection" functionality, which destroys VMs based on their access history. CloudVMI periodically checks "access history" of VMs since they were last used. If VM instances have not been accessed for a specific time, the VMs are destroyed by the garbage VM collector. This CloudVMI only considers the access history, but does not leverage resource utilization, significance, or network dependency (connectivity).

More recently, CloudGC [11] identifies idle VMs if the VMs have been suspended by the end-user or the VMs have not been active for a long period of time. However, CloudGC is more focused on how to recycle (re-use) these idle VMs rather than how to accurately detect the idle VMs.

Utilization-based approach: PULSAR [12] is a cloud resource scheduler based on OpenStack Nova scheduler [40] and is designed to achieve higher resource utilization by preempting resource for inactive (or sprawled [41]) VMs. To detect these inactive VMs, PULSAR only relies on CPU utilization for VMs and the VMs are recognized as inactive if CPU utilization is lower than a predefined threshold.

Snadpiper [13] is a resource management system to address hotspots[5] in data centers. Snadpiper determines idle VM and resources with a combination of three utilization factors (CPU, memory, and network usage) and these idle resources will be used for such hotspots with overprovisioning techniques.

Calheiros et al. [33] introduced an autonomous resource provisioning approach for improving utilization for data centers. Identifying inactive VMs are used to improve data center utilization. For example, If utilization of data center is lower

[4]Linux, Apache, MySQL, and PHP.

[5]high load instances.

TABLE I
EXAMPLES OF THE 25 CLASSES OF SIGNIFICANT USER APPLICATION PROCESSES FOR PROCESS KNOWLEDGE BASE. (WE CAN ONLY SHOW SOME OF 25 CLASSES DUE TO INTERNAL PROPERTY ISSUES.)

Categories	Type of Processes
RDBMS	Including relational DBMS related processes such as MySQL, PostgreSQL, IBM DB2, Oracle DB etc.
noSQLs	Including noSQL processes such as CouchDB, Riak, Cassandra, MongoDB, Redis, and others.
Data Analytics Frameworks	Including large-scale data/stream processing applications such as Hadoop, Spark, Flink, Storm, etc.
Containers	Including Docker and Linux Containers.
Web Servers	Including various types of web servers such as Apache, nginx and custom HTTP servers.
Web Frameworks	Including various types of web service related processes such as Node.js, Flask, Django, Ruby on Rails, and others.
Application Servers	Including WebSphere, WebLogic, and Tomcat.
User Connection Processes	Including processes related to incoming and outgoing connections from users such as ssh, VNC.
Development-related Processes	Including version control system and compile-related processes such as `git`, `svn`, `make`, `cmake`, and others.
User Activity Processes	Including a diverse type of user commands that can reflect user activities on VMs such as `cp`, `rm`, `mv`, `yum`, `apt-get`, and others.
Custom User Scripts	Including a wide range of user script such as shell script, python, ruby script, and others.
Data Transfer Processes	Including processes related to data transfer and communications such as FTP, sFTP, SCP, wget, cURL, and others.

than a predefined threshold, this provisioning system considers the oldest instances are inactive and the resource manager destroys them to achieve higher resource utilization. However, it is unclear which utilization metrics (e.g. CPU, memory, or I/O) are monitored to determine low data center utilization and there is a high possibility that the oldest instances are still active, so this approach could make high false negative errors without internal information of such VMs.

The utilization-based approach is intuitive since it relies on statistics from low-level system or applications. However, in private cloud setting, due to the frequent management processes, it is very challenging to determine inactive VMs only with the utilization statistics.

As we reviewed in this section, these three approaches are limited to achieve efficient cloud garbage VM collection. In this work, we design an end-to-end system that considers more specific aspects of systems (e.g., significance/utilization of target resources, connectivity among resources) for the high accuracy of inactive VM identification as well as provides proper management plans to the end users.

IV. FEATURE SELECTION FOR VM IDENTIFICATION

The first step to design the iCSI system is to determine features on VMs that are collected by iCSI. These features will be used to identify active and inactive VMs. We perform an analysis step with sample dataset of VMs in order to find and extract various features on VMs. For this analysis, we use randomly selected 70 VMs and these VMs are not used for our evaluations in the later section.

A. Creating VM Metadata

With the 70 VMs, we collect "human-readable" raw data by using Linux primitive commands: e.g., `ps`, `netstat`, `last`, `ifconfig`, etc. We then create metadata that includes the factors for the VM identification.

Process Metadata: We first differentiate significant and insignificant processes running on these 70 VMs. As we discussed in Section II, relying on VM utilization is not sufficient to identify active or inactive VMs because several management processes consume high CPU and memory resources. [42] However, our insight is that significant user application processes could be information for the VM identification and the user application processes can more contribute to identification process. With this idea, process metadata is created to give more weights to user application processes and less weights (or zero weight) to OS (e.g., kernel processes) or management-related processes (e.g., patch-update). To this end, we create a process knowledge base with 25 categories of significant processes, which are frequently used by IBM end users. We obtain end users' feedback to properly create these process categories. Examples of the 25 categories of significant user application processes are described in Table I.

Utilization Metadata: We create the utilization metadata with three types of factors: CPU, memory, and I/O usage. We extract CPU and memory utilization for all processes (overall VM utilization) as well as for a particular process. I/O utilization is extracted from refining output from "`ifconfig`" command and I/O metadata includes the amount and size of sending and receiving network packets. We also consider the average and the maximum size of I/O usage.

Login Metadata: Login (user access) history can possibly provide diverse aspects of user access pattern to VMs. While we cannot figure out what kinds of actions VM users took in the past with the login history, we can create several useful login metadata such as frequency/duration of logins, differentiated logins in daytime and nighttime.

Network Connection Metadata: We can extract the number of open TCP ports on VMs and established connections with internal/external entities. The internal/external entities can be users, applications, and other VMs. These factors incorporate with process metadata to figure out the established connections from significant/insignificant processes.

Other Metadata: We can create metadata with IP and host information of VMs, which can be useful to determine the network connectivity among VMs.

B. Feature Selection

This "feature selection" step is to find common (strong) correlations that are able to differentiate active and inactive

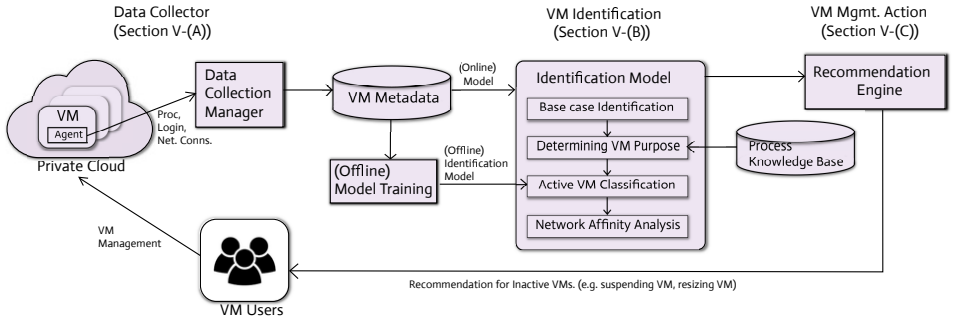


Fig. 2. iCSI System Architecture.

VMs. To decide these features more efficiently, we manually classify (label) the purpose of each VM and compute Pearson correlation coefficient [43] of all factors in the VM metadata according to the classified purpose of each VM. Note that if the coefficient of a factor is close to 1, the factor is highly correlated with a VM's identification. Otherwise (coefficient is close 0), the factor has weak correlation with respect to identifying a VM's activeness or inactiveness identification. Then, we determine the features if the features have more than 0.75 of the coefficient. These features will be used by iCSI system. While the manual classification step for the purpose of VM can help us find and select stronger features for the identification process, it also gives us another research problem meaning that how to properly determine the purpose of all VMs. We will discuss more about this problem in the next section of this paper.

V. System Design

Figure 2 illustrates the architecture of iCSI system – *a cloud garbage VM collector*. iCSI performs three consecutive steps: data collection, identification, and action. A lightweight data collector is designed to manage data collection step. An VM identification model determines active and inactive VMs with the collected data. With the identification results, recommendation engine generates action plans to VM end users.

A. Lightweight Data Collector

The lightweight data collector consists of data collection agents and manager. The agents are running on each VM and periodically execute a `bash script` at predefined time interval. The `bash script` collects the primitive, but useful information from each VM. This information includes the factors that we described in Section-IV-A. The size of data from each collection operation is less than 50 Kbytes and the collected data will be sent to the data collection manager via cURL [44] tool. The manager stores the primitive data and creates metadata that will be used by the VM identification model.

In order to quickly implement and deploy this data collector, we rely on an IBM proprietary infrastructure management system called bigfix [45], but this tool can be easily replaced with other tools such as puppet [46], chef [47], or others [48–50]. Moreover, instead of embedding "the data collection scripts" into target VMs, actual cloud practitioners can consider leveraging VMI-based approaches [39, 51–54] to efficiently collect semantics on the VMs.

Since this data collector will be deployed to production data centers and it must not mess up production services (e.g., no SLA violations) to end users[6], we pick a small-scale data center as our testbed and monitor the end-to-end safety of this collector for three weeks. During this validation period, we also tune several configurations of the data collector such as the data collection interval. The decision for data collection interval is based on the tradeoff between the performance impact of this collector on a VM's main workload and the quality of collected dataset. We start with finer-grained collection interval (e.g., 1 hour or less) and gradually increase the this interval to 12 hours. We realized that the data collection at every 4 hours interval can provide rich dataset that contains sufficient semantics, which timely reflect the change of VM usage pattern (e.g. CPU utilization). And this time interval (4 hours) has negligible impact on the VM's performance. After the validation, this collector can be deployed to multiple IBM data centers in US.

Protecting end users' privacy is a critical concern to design this data collector because the dataset can contain privacy information for end users. Thus we exclude processes related to password (e.g. `passwd` on Linux) or user management (e.g. `adduser`). Moreover, this collector gathers data from only US-owned VMs according to EU's the general data protection regulations [55].

B. VM Identification Model

With VM metadata periodically gathered and created by the data collector, VM identification model determines active and inactive VMs using the insight from Section-IV-B. A summary of the process performed by this model is illustrated in Figure 3. The model identifies VMs through the following procedures:

1) **Procedure #1:** Base case VM identification.
2) **Procedure #2:** Determining the purpose of VMs with running processes.

[6]Although all the end users are internal employees at IBM, QoS (Quality of Service) and SLA (Service Level Agreement) must be guaranteed in all private clouds. So the end-to-end safety of data collector is very critical.

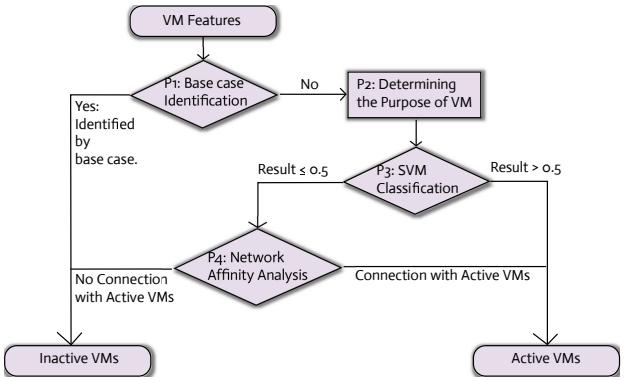


Fig. 3. VM Identification Process.

3) **Procedure #3:** Active/inactive VM classification with linear support vector machine (SVM).
4) **Procedure #4:** Performing network affinity analysis to discover network dependencies among VMs.

While the goal of this identification model is to correctly detect active and inactive VMs, minimizing the number of false negative errors is also critical. False negative errors in the identification means that *active VMs are incorrectly identified as inactive VMs*. If a VM is suspended or terminated with a false negative error, the actual owner of the VM will (temporarily or permanently) lose all data and application configurations in the VM. This is a clear example of SLA (Service Level Agreement) violation cases.

Procedure #1 – Base case VM identification. The identification model filters apparently inactive VMs out for the identification process with the simple, but effective rules. These rules are based on "explicit" usage pattern of inactive VMs. The rules are 1) no reboot for a VM over the last six months, 2) no significant processes running on that VM, 3) no login actives on the VM over the last three months, and 4) no established connections with other external applications/VMs during data collection period. If a VM satisfies all these rules, the identification model considers the VM as inactive one. These rules can detect 5–7% of inactive VMs (according to the analysis with 70 sample VMs). This procedure helps other procedures performed later on by making smaller size of the VM dataset.

Procedure #2: Determining the purpose of VMs. This step is to apply the important findings (Section-IV-B) for the identification process. To use specific (strong) correlated factors, the VM identification model should properly determine the purpose of VMs. The intuition is that the purpose of VMs can be determined by the VMs' running processes. For example, if RDBMS processes (e.g., PostgreSQL, MySQL) are running on a VM, this VM is commonly used for storage purpose. Similarly, if Hadoop processes are running on a VM, this VM is highly likely to be used for analytics purpose. However, this insight does not always work for all cases. While RDBMS is mainly used to store structured data, but it does not always mean that the VM with RDBMS is only used for storage-purpose. In many cases, RDBMS can be a part of an application configuration (e.g., LAMP) for development-purpose or test-purpose VMs.

To properly decide the possible purposes of VMs, we leverage user feedbacks for 70 VMs (used for data analysis/feature selection in Section-IV) and decide weights for the purposes associated with running processes. We then create a function that maps type of running processes to possible purposes. This function returns multiple purposes with different weights as shown in below.

$$\boldsymbol{Input}: process_i$$
$$\boldsymbol{Output}: \{purpose_1 : w_1, purpose_2 : w_2, ..., purpose_n : w_n\}$$

The outputs of this function – *weights associated with different purposes of VMs* – are directly used by a linear SVM classifier in the next procedure.

Procedure #3: Active/Inactive VM classification. The VM identification model employs a supervised learning model called a linear SVM [56] to classify active and inactive VMs. SVM is a widely-used classifier to solve diverse research problems in cloud computing area (e.g., workload prediction and classification [35, 57], application performance modeling [58], anomaly detection [59]). Linear SVM is an optimal margin-based classifier with linear kernel and tries to find a small number of data points that separate all data points of two classes with a hyperplane [56].

A key insight on employing the linear SVM is that this classifier dynamically selects and uses different correlation features for the classification according to the purposes of VMs. Table II describes specific correlated features selectively used by the classifier. As shown in Table II, both analytics- and devops-purpose mostly use VM utilization-related features such as CPU and memory usage, development-purpose more focuses on user access and development related processes (e.g., git [60]) as main features, and backup/storage-purpose concentrates on particular utilization features associated with storage/backup operations (e.g., I/O, memory) for SVM classification. Others-purpose is the most challenging case since most of these VMs do not have any interesting features in terms of running processes or utilization. So, the SVM classifier uses only two features; daytime login frequency/duration and maintenance-related features (e.g., `yum`, `apt-get`). Especially for the daytime login frequency and duration, we collect information from only US-owned VMs for the identification and we assume that all VM users are in the US time zone. Thus, we extracted daytime login duration and frequency features based on the US time zone.

Moreover, as VMs can be used for multiple purposes, the identification model runs the SVM classifier multiple times with different weights. The VMs are classified as active or inactive with a weighted sum of classification results for each purpose of VMs. While the linear SVM provides a binary result ($\{0, 1\}$) for each classification operation, a classification

TABLE II
SPECIFIC FEATURES ACCORDING TO PURPOSE OF VMS USED BY LINEAR SVM CLASSIFIER.

Purpose of VM	List of Specific Features
Analytics	%CPU of VM, %CPU of significant user procs, %MEM of VM, # of open ports, # of established connections.
DevOps[7]	# of significant processes, %CPU/%MEM of significant procs, # of established connections.
Development	# of logins, average login hours (daytime), # of ssh/VNC connections, # user development activity processes.
Storage/Backup	# of storage/backup procs, Network I/O usage, %MEM of significant user processes, # of established connections.
Others	# of user maintenance activity processes, # of daytime logins, duration of daytime logins.

result could have a value between 0 and 1. We use 0.5 of a threshold to differentiate active and inactive VMs and this threshold works very well for our cases. The VM identification model considers VMs as inactive when they have less than or equal to 0.5 of the classification result. For VMs with greater than 0.5 of classification result, the model recognizes them as active VM.

Procedure #4: Network affinity analysis. The last procedure of the VM identification model is to perform a network affinity analysis (NAA) to figure out important dependencies of all VMs. NAA can reduce false negative errors by validating the classification result with the dependencies. NAA is a well-known approach for migrating legacy enterprise applications to cloud environment with discovering network connections (dependencies) with peers (e.g., applications, servers) [36, 37, 62]. The VM identification model investigates all external connections of VMs that are classified as inactive (in the procedure #3). This model adjusts the VM's classification result from inactive to active if a network dependency with active VMs exists. NAA is particularly useful for VMs with cluster configurations (e.g., Hadoop [63], Mesos [64] or Kubernetes [65]). Usually, master VMs of such clusters have strong characteristics to be properly classified as active or inactive VM, but slave ones tend to have weak characteristics for the identification process. NAA propagates the confidence determined from master VMs to slave ones. In the evaluation section, we will assess the impact of NAA on how much it can contribute to accuracy improvement for iCSI system.

C. Recommendations Engine

Through the previous two steps, iCSI can identify active and inactive VMs from data centers. With the identification results, the recommendation engine provides the following action plans to the owner for inactive VMs: terminating VM, suspending VM and resizing VM. The recommendation engine takes defensive policies for inactive VMs since this system is deployed to real production data centers, meaning that misidentification errors could have actual impact on the business in the corporation. The summary of three recommendation policies is described in Table III.

[7]A clipped compound of development and operation [61].

TABLE III
THREE VM RECOMMENDATION POLICIES OF ICSI.

Recommendation	Trigger Conditions
No Actions	Active VMs (Classification result > 0.5.)
Terminating VM	Classification result is equal to 0.
Suspending VM	$0 <$ Classification result ≤ 0.5.
Resizing VM	$0 <$ Classification result ≤ 0.5 and significant processes are running on the VM.

*1) **Terminating VM**:* This recommendation is proposed to users when either their VMs are identified as inactive by the procedure #1's rules or the classification result of VMs is completely inactive (the weighted sum of classification results in procedure #3 is 0). When the users terminate their VM, they can create a snapshot for the VMs if necessary.

*2) **Suspending VM**:* This recommendation is provided when VMs have a classification result of greater than 0 (not completely inactive) and have no significant processes on the VMs. In this case, iCSI does not recommend the users to terminate their VMs since there is a certain possibility that the VMs could be temporarily inactive.

*3) **Resizing (Downsizing) VM**:* This recommendation is suggested to end users of inactive VMs when the VMs have significant processes (in most cases, the classification results of these VMs are close to the active threshold). VM size with lowest cost is recommended to the users for resizing operation. Determining the optimal VM size for down sizing is also very important research problem, however, we think this is out of scope of this paper. We leave this problem as our future work.

VI. PERFORMANCE EVALUATION

In this section, we describe the performance evaluation of iCSI system on the production data centers at IBM.

A. Evaluation Setup

Data centers and VM dataset: iCSI system is deployed into multiple data centers for IBM private clouds. With this deployment, iCSI can get access to 750 VMs in total. The data collector of iCSI gathers VM information in every four hours and total duration of measurement period is four weeks. For obtaining the ground truth, we ask actual users of these 750 VMs and use their feedbacks as the ground truth for this evaluation.

Evaluation criteria: We first evaluate the accuracy of iCSI system for identifying active and inactive VMs in data centers. We then measure both cost saving and utilization improvement of VMs with the recommendations from iCSI system.

Performance metrics: With respect to the identification accuracy, we measure three well-known metrics for the classification accuracy: recall, precision, and f-measure. Recall is more sensitive to the number of false negative errors and precision is affected by the number of false positive errors. F-measure is

TABLE IV
TWO TRUE CASES AND FALSE CASES IN THE IDENTIFICATION ACCURACY. (TP: TRUE POSITIVE, TN: TRUE NEGATIVE, FY: FALSE POSITIVE, FN: FALSE NEGATIVE.)

		VM Identification Result	
		Active	**Inactive**
Truth	**Active**	**TP**: Active VMs are **correctly** identified as active.	**FN**: Active VMs are **incorrectly** identified as inactive.
	Inactive	**FP**: Inactive VMs are **incorrectly** identified as active.	**TN**: Inactive VMs are **correctly** identified as inactive.

TABLE V
VM IDENTIFICATION ACCURACY OF ICSI.

	Recall Rate	**Precision Rate**	**F-measure Score**
Results	0.90	0.81	0.85

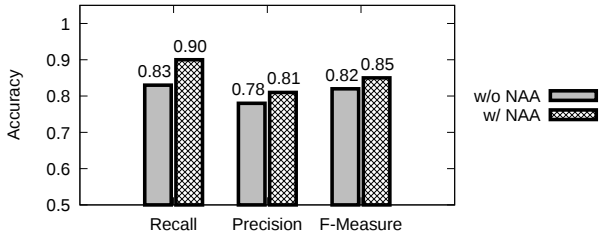
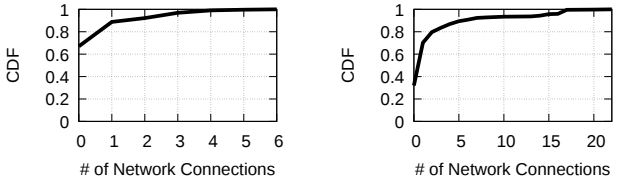


Fig. 4. Impact of Network Affinity Analysis on the Identification Accuracy.



(a) Inactive VMs. (b) Active VMs.

Fig. 5. CDF of External Network Connections of VMs.

TABLE VI
STATISTICS FOR EXTERNAL CONNECTIONS OF VMS.

	Active VMs	**Inactive VMs**
Mean # of External Conns.	2.3	0.5
Standard Deviation	4.1	0.9

a harmonic mean of recall and precision. These three metrics are shown in below.

$$Recall = \frac{TP}{TP + FN}, \quad Precision = \frac{TP}{TP + FP} \tag{1}$$

$$F - Measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{2}$$

Reducing the false negative errors is very critical to iCSI system, so recall is the most important metric for the identification accuracy. To understand these above metrics, Table IV describes two true and error cases in our evaluation.

Cost saving and utilization improvement of VMs are measured with an assumption that users for all inactive VMs are accept and follow the recommendations from iCSI system. Cost saving is normalized over estimated cost for the next billing cycle. Utilization is also normalized over the current utilization of the private cloud infrastructure.

Baselines: We use two state-of-the-art cloud garbage collection approaches for the baselines; Pleco [8] and Garbo [9]. Pleco identifies active and inactive VMs with a combination of a reference model (connectivity) and decision tree classification (utilization). Garbo is a graph-based VM cleaning-up tool for AWS. Garbo creates a directed acyclic graph from core nodes and performs mark and swap operation to clean up inactive VMs. In our evaluations, these two baselines are evaluated from the same setup with that of iCSI. e.g., the same VM dataset from the same data centers.

B. Accuracy of VM Identification

We measured the identification accuracy of iCSI in isolation and then compare its accuracy with two baselines. We also validate the accuracy results with k-fold cross validation. [56] Table V shows accuracy results of iCSI for VM identification. iCSI can identify active VMs with 0.90 of recall, 0.81 of precision, and 0.85 of f-measure. In this evaluation, iCSI identifies 472 (63%) of active VMs and 278 (37%) of inactive VMs. Especially for the recall, iCSI has 47 false negative errors.

To evaluate the impact from Network Affinity Analysis (NAA), we measure two different accuracy results of iCSI with and without of the procedure #4 in the identification step. The results are shown in Figure 4. NAA improves the identification accuracy with 3%–7% and has more impact on recall rate with reduced false negative errors from 69 to 47.

In order to understand this impact, we calculate CDF (Figure 5) and statistics (Table VI) of external network connections for all 750 VMs. We found that only 30% of inactive VMs have external connections and more than 60% of the active ones have external connections with other VMs. Moreover, inactive VMs have average of 0.5 external connections (σ = 0.9) and inactive ones are connected with 2.3 of external connections in average (σ = 4.1). These results indicate that active VMs are highly likely to have connections with other active VMs.

We also compare the accuracy results with other approaches. The results are shows in Table VII. In all accuracy metrics, iCSI outperforms (11%–20% better accuracy) two baseline. Pleco and Garbo only have 0.75 and 0.70 of recall, but iCSI shows 0.90 of recall. To understand these results, we need to understand the technical differences in the three approaches. Pleco and Garbo are a graph-based approach, meaning that connectivity with other VM is the most critical factor to

TABLE VII
ACCURACY COMPARISON WITH TWO BASELINES.

	Recall Rate	Precision Rate	F-Measure Score
iCSI	0.90	0.81	0.85
Pleco	0.75	0.69	0.72
Garbo	0.70	0.67	0.68

determine inactive or active VMs. While we confirm that the connectivity could be a key to identify active/inactive VMs, but other factors (e.g., login, utilization) are also very critical. For example, in many cases, VMs performs significant processing without any external connections. As shown in Figure 5-(b). Approximately 30% of active VMs have no external connections, but they are identified as active by actual users.

These (stand-alone) VMs can be very challenging use cases for the graph-based approach to correctly identify them as active or inactive VMs. Since VMs are created and used for different purposes, identification models should cautiously select proper features from multiple dimensions of information including network connectivity/dependency, utilization, login history, and others.

C. Cloud Cost Saving

With the identification results, we evaluate how much cost iCSI can save for the private cloud infrastructure. To accurately measure the cost saving, we first define a penalty cost function for misidentification error (especially for false negative errors) as expressed in equation-(3) where n is the number of recommendation policies and m is the number of misclassified VMs. And the total cost is calculated by equation-(4). This penalty cost function is inspired by penalty functions [66, 67] for evaluating under-provisioning case of cloud resource management. This is because VM users would (temporarily or permanently) face under-provisioning circumstance if the users suspend or terminate their VMs based on the recommendation with false negative errors. Moreover, the penalty cost function considers differentiated penalty weight (w in equation-(3)) according to users' actions with different recommendations. For example, "termination VM" has the highest penalty weight and "resizing VM" has the lowest penalty weight.

$$PenaltyCost = \sum_{i=1}^{n}(w_i \sum_{j=1}^{m} cost_{vm_j}) \quad (3)$$

$$TotalCost = Cost_{activeVMs} + PenaltyCost \quad (4)$$

Figure 6 shows the estimated results for the VM cost of three approaches. The results are normalized over the (estimated) cost for next internal billing cycle without any VM management for inactive VMs. iCSI can save 23% of cloud cost, which is 12%–14% more savings compared to two baselines (Pleco can save 11% of VM cost and Garbo is able to reduce 9% of VM cost). This result is higly related to the number of false positive errors in the identification process of three approaches. Since iCSI has the lowest numbers of false negative errors, it has the smallest impact from the penalty function. Pleco and Garbo generate 115 and 127 of false negative errors, but iCSI has only 47 false negative errors among 750 VMs. This indicates that iCSI could maximize the cost saving with the lowest penalty cost.

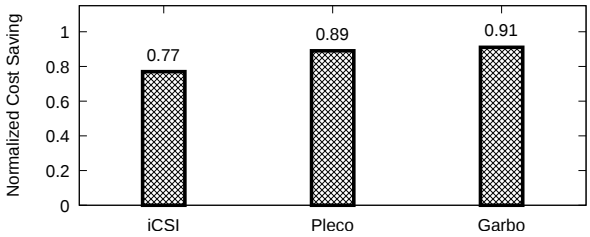


Fig. 6. Normalized VM Cost of Three Approaches. (1.0: Estimated Cost of Next Billing Cycle). This result is normalized over the estimated VM cost without any VM management according to the recommendations from three approaches. In this graph. Lower value indicates better cost efficiency.

D. VM Utilization Improvement

We measure the improvement of VM utilization with three approaches. The measurement results are normalized over the current utilization of the IBM private cloud. Note that we do not consider utilization improvement from false negative errors since these VMs have no actual benefits to the utilization improvement.

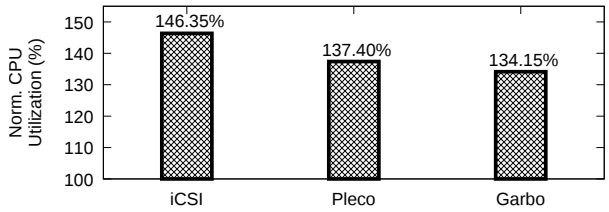
TABLE VIII
AVERAGE UTILIZATION IMPROVEMENT OF THREE APPROACHES.

	iCSI	Pleco	Garbo
Average Improvement of VM Utilization	46%	31%	29%

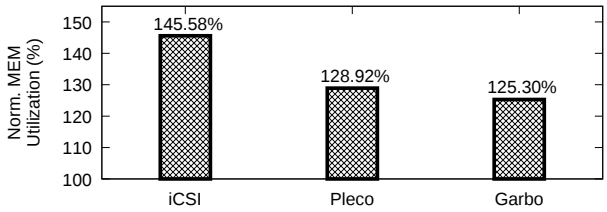
Figure 7 illustrates the evaluation results from the diverse aspects of VM utilization and Table VIII shows the average improvement for these three approaches. While all three approaches can also improve the VM utilization, iCSI achieves the largest improvement (46%) of VM utilization (Pleco's utilization improvement is 31% and Garbo's utilization improvement is 29%). For all the 8 categories of utilization metrics, while the differences between iCSI and baselines are similar, the CPU and memory utilization improvement for user applications (Figure 7-(c) and (d)) show the largest difference. i.e. 20% difference for CPU utilization and 26% difference for memory utilization of user applications. As iCSI considers diverse factors from VMs to identify inactive/active VMs, it has more advantages to archive higher utilization for significant processes (e.g. user applications). However, since the baselines (graph-based approaches) are more concentrating on the connectivity, improving (CPU and memory) utilization for user applications is challenging for the baselines.

VII. CONCLUSION

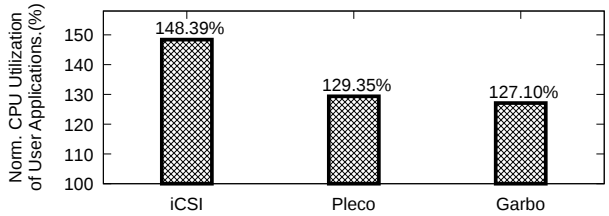
We have presented iCSI – *a cloud garbage VM collector to identify inactive VMs*, which improves the cost efficiency and the VM utilization. iCSI has three important steps of "data collection", "VM identification", and "recommendation."



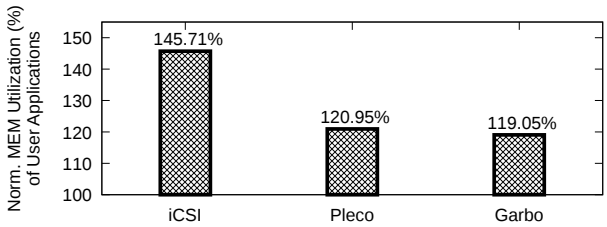
(a) Improvement of VM CPU Utilization.



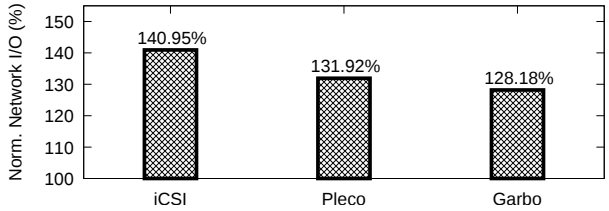
(b) Improvement of VM Memory Utilization.



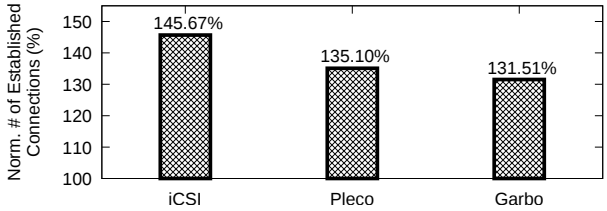
(c) Improvement of CPU Utilization for User Applications.



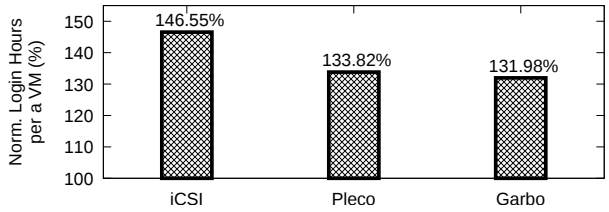
(d) Improvement of Memory Utilization for User Applications.



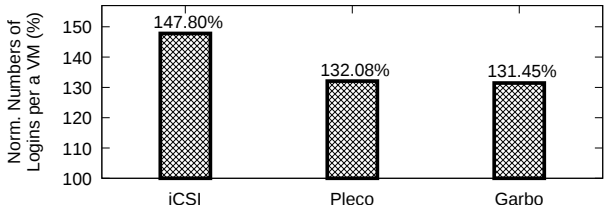
(e) Improvement of VM Network I/O Usage.



(f) Improvement of the Number of Established Network Connections.



(g) Improvement of Login Hours per VM.



(h) Improvement of the Number of Logins per VM.

Fig. 7. VM Utilization Improvement with Three Approaches. All Results are Normalized.

For the data collection, we design a lightweight approach to periodically gather primitive, but holistic information for running VMs. With the data collection, we describe how we analyze the data in order to extract significant features for active/inactive VM identification. iCSI determines active and inactive VMs through the following steps. iCSI first performs a base case classification for inactive VMs, then determines the purpose of each VM. By leveraging a supervised learning technique (SVM), iCSI identifies active/inactive VMs with corresponding features to the purpose. Finally, iCSI validates the identification results using a network affinity model, and propagates the confidence to connected VMs. With the identification results, iCSI recommends proper actions to the inactive VM users. i.e., terminating, suspending or resizing VM.

Our evaluation with 750 VMs from enterprise data centers shows the accuracy is 90%, which is 15% – 20% higher than existing methods. With this accuracy, iCSI can save 23% of cloud cost, which is 12% – 14% better achievement as compared to the baselines. We also demonstrate that iCSI can improve more than 45% of VM utilization.

In the near future, we will investigate an automated approach to address inactive VMs with iCSI. Since the current iCSI system relies on the actual user's decision to manage inactive VMs. Among three recommendations we have now, we will more focus on an intelligent VM resizing scheme that precisely determines an optimal size of VMs based on the actual processing requirements to individual VMs.

ACKNOWLEDGMENT

We would like to show our gratitude to Mr. Ivan Dell'Era, Mr. Carlos Fonseca and Dr. Nikos Anerousis at IBM for their timely and numerous supports for this project. We also thank anonymous reviewers for the invaluable comments and suggestions to improve this paper.

REFERENCES

[1] Melanie Posey Robert P. Mahowald, Chris Morris, Mark Schrutt, Satoshi Matsumoto, Vladimir Kroa, Petr Zajonc, Linus Lai Frank Gens, Hamza Naqshbandi, Melih Murat, David Senf, David Tapper, Alejandro Florean, Mayur Sahni, Thomas Dyer, and Benjamin McGrath. Worldwide Hosted Private Cloud Services Forecast, 2015–2019: New Models for Delivering Infrastructure Services. In *IDC Tech Report*, September 2015.

[2] Gen Lin, David Fu, Jinzy Zhu, and Glenn Dasmalchi. Cloud Computing: IT as a Service. *IT Professional*, 11(2):10–13, March 2009.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In *the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, NY, USA, October 2003.

[4] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live Migration of Virtual Machines. In *the 2nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '05)*, Boston, MA, USA, May 2005.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, April 2010.

[6] Zach Hill and Marty Humphrey. A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: the death of the local cluster? In *the 10th IEEE/ACM International Conference on Grid Computing (Grid '09)*, Alberta, Canada, October 2009.

[7] Jonathan Koomey and Jon Taylor. 30 percent of servers are 'comatose'. http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf. ONLINE.

[8] Zhiming Shen, Christopher Young, Sai Zeng, Karin Murthy, and Kun Bai. Identifying Resources for Cloud Garbage Collection. In *the 12th International Conference on Network and Service Management (CNSM '16)*, Montreal, Quebec, Canada, November 2016.

[9] Netanel Cohen and Anat Bremler-Barr. Garbo: Graph-based Cloud Resource Cleanup. In *the ACM Symposium on Cloud Computing (SoCC '15)*, Kohala Coast, Hawaii, USA, August 2015.

[10] Netflix. Janitor monkey. https://github.com/Netflix/SimianArmy/wiki/Janitor-Home. ONLINE.

[11] Bo Zhang, Yahya Al Dhuraibi, Romain Rouvoy, Fawaz Paraiso, and Lionel Seinturier. CLOUDGC: Recycling Idle Virtual Machines in the Cloud. In *the 5th IEEE International Conference on Cloud Engineering (IC2E '17)*, Vancouver, Canada, April 2017.

[12] David Breitgand, Zvi Dubitzky, Amir Epstein, Oshrit Feder, Alex Glikson, Inbar Shapira, and Giovanni Toffetti. An Adaptive Utilisation Accelerator for Virtualized Environments. In *the 2nd IEEE International Conference on Cloud Engineering (IC2E '14)*, Boston, MA, USA, March 2014.

[13] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *the 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI '07)*, Cambridge, MA, USA, April 2007.

[14] IBM. Softlayer – Privacy Agreement. https://www.softlayer.com/sites/default/files/softlayer_privacy_agreement_-_may_2016.pdf. ONLINE.

[15] Amazon Web Services. Data Privacy. https://aws.amazon.com/compliance/data-privacy-faq. ONLINE.

[16] Microsoft Azure. Trust Center. https://azure.microsoft.com/en-us/support/trust-center. ONLINE.

[17] Hassan Takabi, James B.D. Joshi, and Gail-Joon. Ahn. Security and Privacy Challenges in Cloud Computing Environments. *IEEE Security & Privacy*, 8(6):24–31, November/December 2010.

[18] Francisco Rocha and Miguel Correia. Lucy in the Sky without Diamonds: Stealing Confidential Data in the Cloud. In *the 41st IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Hong Kong, China, June 2011.

[19] In Kee Kim, Sai Zeng, Christopher Young, Jinho Hwang, and Marty Humphrey. A Supervised Learning Model for Identifying Inactive VMs in Private Cloud Data Centers. In *the 17th ACM/IFIP/USENIX Middleware Conference (Middleware '16)*, Trento, Italy, December 2016.

[20] Jinho Hwang, Maja Vukovic, and Nikos Anerousis. FitScale: Scalability of Legacy Applications through Migration to Cloud. In *International Conference on Service Oriented Computing (ICSOC)*, Banff, Alberta, Canada, October 2016.

[21] Amazon Web Services. https://aws.amazon.com. ONLINE.

[22] Microsoft Azure. https://azure.microsoft.com. ONLINE.

[23] Google Cloud Platform. https://cloud.google.com. ONLINE.

[24] IBM SoftLayer. http://www.softlayer.com. ONLINE.

[25] Jinho Hwang. Computing Resource Transformation, Consolidation and Decomposition in Hybrid Clouds. In *the 11th International Conference on Network and Service Management (CNSM '15)*, Barcelona, Spain, November 2015.

[26] Amazon Cloud Watch. https://aws.amazon.com/cloudwatch. ONLINE.

[27] Jinho Hwang, Kun Bai, Michael Tacci, Maja Vukovic, and Nikos Anerousis. Automation and Orchestration Framework for Large-Scale Enterprise Cloud Migration. *IBM Journal of Research and Development*, 60(2-3):1:1–1:12, March 2016.

[28] theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. CloudNaaS: Semi-synchronized Non-blocking Concurrent Kernel Heap Buffer Overflow Monitoring. In *the ACM Symposium on Cloud Computing (SoCC '11)*, Cascais, Portugal, October 2011.

[29] AWS Auto Scaling. https://aws.amazon.com/autoscaling. ONLINE.

[30] Google Cloud Platform Autoscaling Groups of Instances. https://cloud.google.com/compute/docs/autoscaler. ONLINE.

[31] Scott Devoid, Narayan Desai, and Lorin Hochstein. Poncho: Enabling Smart Administration of Full Private Clouds. In *the 27th USENIX Large Installation System Administration Conference (LISA '13)*, Washington D.C., USA, November 2013.

[32] Hyun Wook Baek, Abhinav Srivastava, and Jacobus Van der Merwe. CloudVMI: Virtual Machine Introspection as a Cloud Service. In *the 1st IEEE International Conference on Cloud Engineering (IC2E '14)*, Boston, MA, USA, March 2014.

[33] Rodrigo N. Calheiros, Rajiv Ranjan, and Rajkumar Buyya. Virtual Machine Provisioning Based on Analytical Performance and QoS in Cloud Computing Environments. In *the 40th International Conference on Parallel Processing (ICPP '11)*, Taipei, Taiwan, September 2011.

[34] Matthew Hertz and Emery D. Berger. Quantifying the Performance of Garbage Collection vs. Explicit Memory Management. In *the ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages & Applications (OOPSLA '05)*, San Diego, CA, USA, October 2005.

[35] Ron C. Chiang, Jinho Hwang, H. Howie Huang, and Timothy Wood. Matrix: Achieving Predictable Virtual Machine Performance in the Clouds. In *the 11th International Conference on Autonomic Computing (ICAC '14)*, Philadelphia, PA, USA, June

2014.

[36] Kun Bai, Niyu Ge, Hani Jamjoom, Ea-Ee Jan, Lakshmi Renganarayana, and Xiaolan Zhang. What to Discover Before Migrating to the Cloud. In *the IFIP/IEEE International Symposium on Integrated Network Management (IM '13)*, Ghent, Belgium, May 2013.

[37] Michael Nidd, Kun Bai, Jinho Hwang, Maja Vukovic, and Michael Tacci. Automated Business Application Discovery. In *the IFIP/IEEE International Symposium on Integrated Network Management (IM '15)*, Ottawa, Canada, May 2015.

[38] Magellan the Argonne Cloud Computing Platform. http://cloud.mcs.anl.gov. ONLINE.

[39] Kara Nance, Brian Hay, and Matt Bishop. Virtual Machine Introspection: Observation or Interference? *IEEE Security & Privacy*, 6(5):32–37, September 2008.

[40] Abhishek Gupta, Laxmikant V. Kale, Dejan Milojicic, Paolo Faraboschi, and Susanne M. Balle. HPC-Aware VM Placement in Infrastructure Clouds. In *the 1st International Conference on Cloud Engineering (IC2E '13)*, San Francisco, CA, USA, March 2013.

[41] Maik Lindner, Fiona McDonald, Barry McLarnon, and Philip Robinson. Towards Automated Business-driven Indication and Mitigation of VM Sprawl in Cloud Supply Chains. In *the 6th IFIP/IEEE International Workshop on Business-driven IT Management (BDIM '11)*, Dublin, Ireland, May 2011.

[42] George Amvrosiadis, Angela Demke Brown, and Ashvin Goel. Opportunistic Storage Maintenance. In *the 25th ACM Symposium on Operating Systems Principles (SOSP '15)*, Monterey, California, USA, October 2015.

[43] Wikipedia. Pearson product-moment correlation coefficient. https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient. ONLINE.

[44] Curl: Command Tool and Library. https://curl.haxx.se. ONLINE.

[45] IBM. Bigfix: Endpoint security and management solution. http://www-03.ibm.com/security/bigfix. ONLINE.

[46] Puppet. Configuration management. https://puppet.com/solutions/configuration-management. ONLINE.

[47] Chef. https://www.chef.io. ONLINE.

[48] Ansible. https://www.ansible.com. ONLINE.

[49] SaltStack. https://saltstack.com. ONLINE.

[50] CFEngine. https://cfengine.com. ONLINE.

[51] Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Antfarm: Tracking Processes in a Virtual Machine Environment. In *the USENIX Annual Technical Conference (ATC '06)*, Boston, MA, USA, May 2006.

[52] Stephen T. Jones, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Geiger: Monitoring the Buffer Cache in a Virtual Machine Environment. In *the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '06)*, San Jose, CA, USA, October 2006.

[53] Bryan D. Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares: An Architecture for Secure Active Monitoring Using Virtualization. In *the IEEE Symposium on Security and Privacy (SP '08)*, Oakland, CA, USA, May 2008.

[54] Donghai Tian, Qiang Zeng, Dinghao Wu, Peng Liu, and Changzhen Hu. Kruiser: Semi-synchronized Non-blocking Concurrent Kernel Heap Buffer Overflow Monitoring. In *the 19th Annual Network & Distributed System Security Symposium (NDSS '12)*, San Diego, CA, USA, February 2012.

[55] European-Union. the General Data Protection Regulation. http://www.consilium.europa.eu/en/policies/data-protection-reform/data-protection-regulation. ONLINE.

[56] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. the Element of Statistical Learning: Data Mining, Inference, and Prediction. 2011.

[57] In Kee Kim, Wei Wang, Yanjun Qi, and Marty Humphrey. Empirical Evaluation of Workload Forecasting Techniques for Predictive Cloud Resource Scaling. In *the 9th IEEE International Conference on Cloud Computing (CLOUD '16)*, San Francisco, CA, USA, June 2016.

[58] Palden Lama and Xiaobo Zhou. AROMA: Automated Resource Allocation and Configuration of MapReduce Environment in the Cloud. In *the 9th ACM International Conference on Autonomic Computing (ICAC '12)*, San Jose, CA, USA, September 2012.

[59] Shin-Ying Huang and Yen-Nun Huang. Network Traffic Anomaly Detection based on Growing Hierarchical SOM. In *the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '13)*, Budapest, Hungary, June 2013.

[60] Git: Fast Version Control System. http://git-scm.com. ONLINE.

[61] Wikipedia. Devops: Development and operation. https://en.wikipedia.org/wiki/DevOps. ONLINE.

[62] Jill Jermyn, Jinho Hwang, Kun Bai, Maja Vukovic, Nikos Anerousis, and Salvatore Stolfo. Improving Readiness for Enterprise Migration to the Cloud. In *the 15th ACM/IFIP/USENIX Middleware Conference (Middleware '14)*, Budapest, Hungary, December 2014.

[63] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. the Hadoop Distributed File System. In *the 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST '10)*, Incline Village, NV, USA, May 2010.

[64] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI '11)*, Boston, MA, USA, March 2011.

[65] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale Cluster Management at Google with Borg. In *the 10th European Conference on Computer Systems (Eurosys '15)*, Bordeaux, France, April 2015.

[66] Sadeka Islam, Kevin Lee, Alan Fekete, and Anna Liu. How a Consumer Can Measure Elasticity for Cloud Platforms. In *the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE '12)*, Boston, MA, USA, April 2012.

[67] Zhiming Shen, Sethuraman Subbiah, Xiaohui Gu, and John Wilkes. CloudScale: Elastic Resource Scaling for Multi-Tenant Cloud Systems. In *the ACM Symposium on Cloud Computing (SoCC '11)*, Cascais, Portugal, October 2011.