Identifying Resources for Cloud Garbage Collection

Zhiming Shen
Cornell University
Ithaca, NY, USA
zshen@cs.cornell.edu

Christopher C. Young, Sai Zeng, Karin Murthy, Kun Bai
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{ccyoung, saizeng, kmurthy, kunbai}@us.ibm.com

***Abstract*—Infrastructure as a Service (IaaS) clouds provide users with the ability to easily and quickly provision servers. A recent study found that one in three data center servers continues to consume resources without producing any useful work. A number of techniques have been proposed to identify such unproductive instances. However, those approaches adopt the strategy to identify idle cloud instances based on resource utilization. Resource utilization as indicator alone could be misleading, which is especially true for enterprise cloud environment. In this paper, we present Pleco, a tool that detects unproductive instances in IaaS clouds. Pleco captures dependency information between users and cloud instances by constructing a weighted reference model based on application knowledge. To handle cases of insufficient application knowledge, Pleco also supplements its dependency results with a machine learning model trained on resource utilization data. Pleco gives a confidence level and justification for each identified unproductive instances. Cloud administrators can then take different actions according to the information provided by Pleco. Pleco is lightweight and requires no modification to existing applications.**

I. Introduction

Cloud deployments greatly simplifies the process of acquiring cloud resources. Unfortunately, simplifying the process of procuring cloud instances has also raised challenges on the converse process of terminating them when no longer in use for productive purposes. A recent study found that one in three data center servers continues to consume resources without producing any useful work [1]. It is estimated that there are 3 million such servers in the US and 10 million worldwide, which lead to 10 billion dollars in data center capital investment and contribute to 40% of overall energy waste. In the US alone, one Gigawatt of power is wasted, which is roughly equal to the amount of power consumed by all the households in the city of Chicago [2]. There are many other less obvious but significant costs associated with unproductive servers including: server and associated infrastructure maintenance, software license cost, cooling cost, etc. Cloud providers are stuck with valuable resources being consumed but not satisfying a customer's needs, which becomes a very convoluted discussion to relate cost against the actual productive workload. Cloud consumers are simply overpaying.

A number of techniques have been proposed to address the challenge of detecting virtual machines that appear to have low resource utilization [3], [4], and consolidate them using resource over-subscription [5], [6], [7]. However, workload consolidation faces many practical challenges, and some providers such as Amazon stated that they simply will not employ over-subscription [8]. The key objection is that cloud-native applications can have highly dynamic workloads and resource over-subscription would place cloud providers at great risk of violating Service Level Agreements (SLAs).

Fundamentally, the challenge lies in distinguishing productive and unproductive cloud instances from each other, from the perspective of a consumer. Previous approaches employ a strategy of identifying productive instances based on resource utilization (CPU, I/O, Network). However, as demonstrated in Fig 1, this only addresses quadrant B and C. The challenge with A and D is the fact that some productive instances exhibit very low resource utilization, while some unproductive instances exhibit very high resource utilization. The issue with quadrant A is particularly acute within enterprises due to the wide-scale deployment of per-instance software agents that perform tasks such as backup, virus scanning, security vulnerability scanning, patching, and compliance and configuration management. These agents can drive considerable resource utilization and activate on random schedules, but certainly do not indicate that an instance is productive.

In this paper, we present Pleco, a tool and approach to determine if cloud instances are productive or unproductive. Similar to a memory garbage collector which identifies garbage objects by examining object references [9], [10], Pleco constructs a cloud instance reference model according to the dependency of application workloads, assigns a weight to each reference, and calculates a confidence level for instance productiveness. To this end, Pleco requires application knowledge in order to understand the dependencies. To handle potential errors caused by insufficient application knowledge, Pleco combines the dependency results with a decision-tree model trained on resource utilization. By combing these two techniques, Pleco achieves high accuracy on detecting unproductive cloud instances. It also provides explanations as to why a cloud instance has been identified as productive or unproductive. This reasoning enables users to take appropriate actions. Pleco is light-weight and does not require any modification to applications.

This paper makes the following contributions:

- We identify the challenge of detecting cloud instances that are unproductive.
- We propose a cloud instance reference model for defining and detecting unproductive cloud instances.
- We propose a solution to leverage machine learning to handle potential errors of the cloud instance reference

978-3-901882-85-2

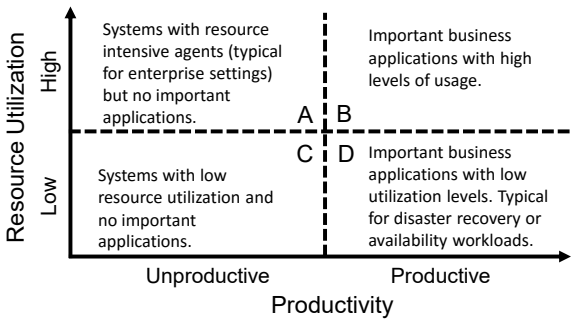

Fig. 1. Resource Utilization vs Productivity Mismatch.

model caused by insufficient application knowledge.

II. Related Work

There are several related approaches for identifying unproductive cloud instances. Janitor Monkey [11] is a service developed by Netflix that considers factors specific to the Amazon Web Services (AWS). This approach is beneficial for Netflix, but it does not take into consideration the activity within the instance. Another common approach is to treat the resource activity of the server as a proxy for it being productive or unproductive [3], [4]. This can help with identifying unproductive cloud instances only when the resource utilization is directly correlated with the productivity, which is usually not the case with production servers (much activity is due to "non-productive" backup agents, security scans, virus checkers etc..). A third approach is specific to desktop applications as it relies on a user interacting with the user interface (UI) in order to determine if it is in use [12], [13]. This approach has shown effectiveness when an UI is present. However, the vast majority of cloud servers do not make use of UIs. Overall these approaches have proven valuable to help identify unproductive servers, VMs, and cloud instances in very specific use cases and domains, but do not generally apply to all cloud providers and all cloud instances.

III. Design

A. Cloud Instance Reference Model

Pleco's reference model is inspired by the memory management reference model found in traditional memory garbage collection algorithms [9], [10]. The memory reference model tracks the in-use state of a particular piece of memory based on the number of external references. The cloud instance reference model is an analogy of the memory reference model (see Fig 2). In this case, the objects or memory segments are cloud instances and any dependencies between any two cloud instances represents an edge in the graph. In order to construct the reference graph we need to decide what to use as the root node. For our approach we introduce an artificial root note that combines all human interaction and any external application outside the analysis scope. Based on this graph, if a particular cloud instance cannot be reached from the root node, we mark the cloud instance as unproductive. The reasoning behind this definition is: if a cloud instance is currently in productive use, then it must be directly used by a user (human or system) or be

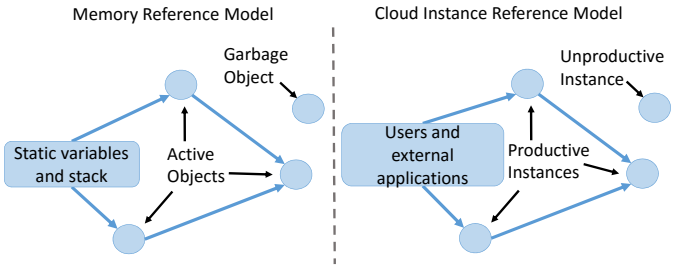

Fig. 2. Memory Reference Model vs. Cloud Instance Reference Model

referenced by another productive cloud instance. Conversely, if a cloud instance is unproductive, it will have no external user (human or system) dependency.

Identifying dependencies requires detailed knowledge about the applications and activities occurring within the cloud instance. Pleco relies on a set of Knowledge Modules (KMs) to discover and analyze different applications and data sources. Examples of sources that are considered include:

- User behavior: a user login or a terminal activity indicates a dependency between the user and the cloud instance.
- Application behavior: network communication traces or application log files contain information about dependencies of applications running in different cloud instances.
- Application configuration: pointers in application configuration files, such as a hostname or IP address of the database instance, indicate application dependencies.
- Cloud meta-data: meta-data from cloud services such as load balancers and auto-scaling groups indicate dependencies from the user to the cloud instance.

B. Identifying Unproductive Cloud Instances

Identifying all of the dependencies between cloud instances is a complicated task due to the sheer number of applications and the time-variance in their activity. An active dependency in the past does not necessarily indicate a current dependency. For example, a cloud instance accessing a webserver in another cloud instance a week ago only indicates that there was a dependency in the past. Conversely, there might be always a dependency between an application instance and its database instance despite there not being an active connection at a particular moment.

In building the underlying reference graph we care mostly about currently active dependencies. However, given an identified dependency and limited information from the user, it is not always clear whether the dependency is active or not. In many cases we must resort to providing heuristic-based confidence levels along with the dependency. For example, if we see an SSH connection from a user to the cloud instance yesterday, we assign a high confidence that the dependency is active. In contrast, if the SSH connect exists but has been idle for over one week we assign a lower confidence. The confidence levels need to be tuned based on typical usage as what is normal in one organization might be abnormal in another. It may take multiple feedback iterations before the

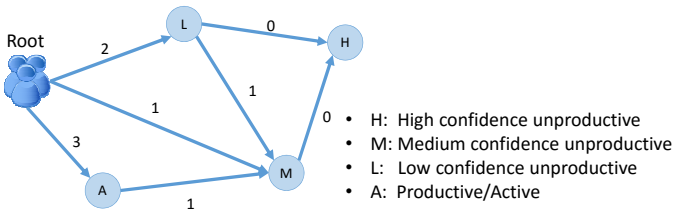

Fig. 3. Identifying unproductive instances with different confidence levels

rules are finalized. Note however that this is a one time effort at setup time and should only need occasional tuning. As mentioned in Section IV, we have implemented a feedback collection system to facilitate this tuning process.

Pleco assigns a confidence to each dependency discovered. The confidence levels are: 0 (inactive), 1 (low confidence active), 2 (medium confidence active), and 3 (high confidence active). More confidence levels can be assigned as long as the assignment is consistent. Example reference graph is shown in Fig 3.

Once the reference graph has been constructed and each edge has a confidence level assigned, we are now able to identify cloud instances that are unproductive. We first remove all edges with a confidence level of 0. Instances that are now no longer reachable in this graph are identified as high-confidence unproductive. Next, we remove all edges with a confidence level of 1. Instances that are now no longer reachable from the root are identified as medium-confidence unproductive. Similarly, by removing edges with a confidence level of 2 we can identify low-confidence unproductive instances, and the remaining instances are classified as productive or active instances (see Fig 3).

C. Decision Tree Verification

As presented above, the accuracy of the cloud instance reference model depends on the number and quality of Knowledge Modules (KMs). Common applications with well-defined configuration files are natural candidates to encode into KMs. However, it is unrealistic to expect that all possible applications can be covered. This leads to some important dependencies being missed and results in false positives for unproductive cloud instances.

To compensate for this practical challenge and reduce false positives while not requiring complete application coverage with KMs, we verify the results discovered from the reference graph with a Decision Tree (DT) classifier [14] trained by examining the resource utilization. Pleco incorporates the results from the DT classifier with the following policy: for productive cloud instances, Pleco ignores the hint from the DT classifier since the productive classification is determined based on application knowledge; for unproductive cloud instances, the output from the DT classifier is considered as follows:

- **Affirmative case**: If the DT model confirms that a cloud instance is unproductive with greater than 80% probability, we increase the confidence level. For example, a low confidence unproductive cloud instance is upgraded to a medium confidence unproductive cloud instance.
- **Conflicting case**: If the DT model classifies an unproductive cloud instance as Active with greater than 80% probability, we lower the confidence level. For example, a high confidence unproductive cloud instance is downgraded to a medium confidence unproductive cloud instance, and a low confidence unproductive cloud instance is downgraded to a productive cloud instance.

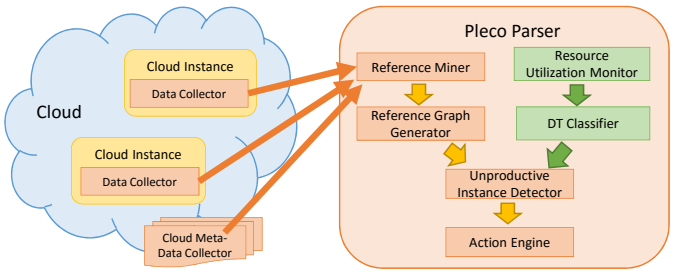

Fig. 4. Architecture of Pleco

IV. Implementation

Fig 4 shows the overall architecture of Pleco. A data collector is running inside each cloud instance and gathers application and user information. This information is periodically sent back to the central parser. Additional data can be collected from other sources such as cloud management meta-data or an enterprise configuration management database. The data collector can be distributed in two ways: (1) It is injected into the target machine by the owner (the only access we could get during our experiments) or (2) for a more wide-spread deployment existing endpoint automation tools (such as chef, puppet, IBM BigFix) can be used to install and execute the data collector on each machine.

The Pleco parser uses a Reference Miner to scan the gathered data and generate the graph with edges representing cloud instance dependencies. To handle the myriad of applications types, the Reference Miner loads a set of Knowledge Modules (KMs) modified from an IBM service called Analytics for Logical Dependency Mapping (ALDM) [15]. Each KM uses its own application specific rules to determine if a particular application is present within the cloud instance, and if it is, it performs deep analysis for dependency information (i.e., scan configuration and log files).

The Reference Graph Generator takes the cloud instance references and generates a reference graph. It maintains a mapping between the host and all its possible aliases, and then makes use of the reconciled list to ensure a single cloud instance is not incorrectly modeled as multiple nodes. It filters out noisy dependencies by leveraging a list of known offenders.

We implemented a Decision Tree (DT) classifier with the traditional CART algorithm [16], and uses a set of metrics including maximum and average CPU utilization, network throughput and disk throughput. The resulting reference graph is processed using the algorithm previously discussed

augmented with the results of the resource-based DT classifier. The output is fed into an Action Engine which is able to take action based on the classifications. For research purposes we have implemented an action that displays Pleco's results visually to owners and allows them to provide feedback on the results. The user can view the entire reference graph including the confidence level assigned to each edge. This has proved to be an excellent mechanism to (1) inform users about the actual activity occurring in their cloud instances, (2) find areas of weakness in the algorithm, and (3) identify gaps in KMs.

V. Evaluations

Our evaluation was conducted in an IBM private cloud that is dedicated to the Research division. The evaluation included 51 cloud instances from eight owners. The owners assisted to manually label each instance as productive or unproductive based on their expert knowledge. Of the 51 cloud instances, 18 were identified as unproductive which represents about 35%. Pleco correctly identified 26 of the cloud instances as productive. Of the remaining, it identified 7 high-confidence unproductive cloud instances (6 correct) and 18 medium-confidence unproductive cloud instances (12 correct).

An interesting finding is the number of users who initially incorrectly labeled their cloud instances and the frequency of erring on the side of productiveness. Among the 51 cloud instances scanned, 1 productive instance was labeled as unproductive, and 8 unproductive instances were labeled as productive. Pleco's ability to display the discovered applications and their dependencies proved very helpful to remind users exactly what was running on their instances. The survey process was conducted in several rounds of feedback-verification process before the labels were finalized.

We compared the results we observed with Pleco to two alternative solutions:

- *Decision-Tree* classified unproductive instances solely based on their resource utilization using the algorithm discussed in Section IV. The DT model was trained using resource utilization traces from one day observations of 25 randomly selected instances. For classification, a six-day trace was used. The model labeled a cloud instance unproductive when the classification probability was greater than 80%.
- *Reference Model* was the approach that identified unproductive cloud instances based on the reference model discussed in Section III-A, without applying any verification. Only high-confidence unproductive cloud instances were labeled unproductive.

Pleco was implemented as a combination of the two approaches. *False Positive* (FP) in our experiment indicates the number of instances that were classified as unproductive, but were actually productive; *False Negative* (FN) indicates the number of instances that were classified as productive, but were actually unproductive. *True Positive* (TP) and *True Negative* (TN) are the number of instances that were correctly classified as unproductive and productive respectively. The precision and recall is calculated the standard way as: Precision = TP / (TP+FP) and Recall = TP / (TP + FN).

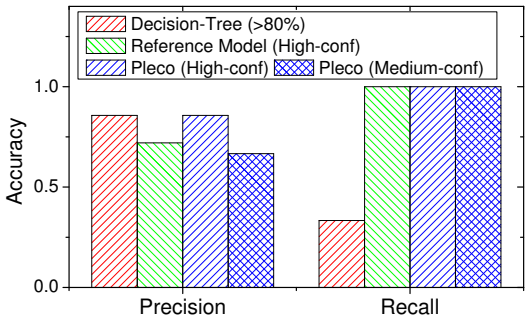

Fig. 5. Accuracy comparison

From Fig 5 we can see that the decision-tree algorithm achieved a high precision, but also a low recall. It shows that by looking at resource utilization, it was common to miss unproductive instances and classify them as productive due to background software agents. Also note that these 51 cloud instances from the research cloud had higher correlation between resource activity and productiveness due to fewer background agents compared to enterprise production clouds and simpler topologies (i.e., no disaster recovery servers, few high-availability deployments). Both of these factors increased the precision achieved by the DT model. Applying the cloud instance reference model improved the recall to 1.0. This was due to the detailed application knowledge eliminating false negatives. However, the precision was lower due to the lack of complete application knowledge. Pleco, through combining the two approaches, achieved the highest results in both precision and recall when the confidence level is high. For medium confidence it continued to achieve reasonable accuracy.

VI. Discussion and Conclusion

Pleco is a light-weight tool that detects unproductive cloud instances in IaaS clouds based on a cloud instance reference model. To compensate for errors caused by insufficient application knowledge, Pleco leverages a decision tree model to classify unproductive cloud instances based on resource utilization, and uses the classification to substantiate the results of the cloud instance reference model.

Pleco employs a gray-box approach, i.e., a data collector must be present in each cloud instance and a Knowledge Module must be available for each application type. ALDM [15] adopts similar approach and has been used by hundreds of customers, a majority of which are fortune 500 companies who own hybrid IT including both cloud and non-cloud infrastructures. This wide-spread footprint shows that Pleco's approach is practical and scalable. Existing research has also demonstrated the feasibility of identifying application dependency by observing low level network communication and resource utilization [17], [18], [19], [20]. This approach could help further improve the scalability of Pleco and simplify the deployment to a large set of instances, and we plan to incorporate it in future work.

REFERENCES

[1] J. Koomey and J. Taylor, "30% of servers are "comatose"," http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf, Jun. 2015.

[2] "Zombie servers: they are here and doing nothing but burning energy," http://www.wsj.com/articles/zombie-servers-theyre-here-and-doing-nothing-but-burning-energy-1442197727.

[3] H. Wu, S. Ren, G. Garzoglio, S. Timm, G. Bernabeu, H. Kimy, K. Chadwick, H. Jang, and S.-Y. Noh, "Automatic Cloud Bursting under FermiCloud," in *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, Dec 2013, pp. 681–686.

[4] J. Stoess, C. Lang, and F. Bellosa, "Energy Management for Hypervisor-based Virtual Machines," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, ser. ATC'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 1:1–1:14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1364385.1364386

[5] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973447

[6] S. A. Baset, L. Wang, and C. Tang, "Towards an Understanding of Oversubscription in Cloud," in *Presented as part of the 2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*. Berkeley, CA: USENIX, 2012. [Online]. Available: https://www.usenix.org/conference/hot-ice12-0/towards-understanding-oversubscription-cloud

[7] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *INFOCOM, 2012 Proceedings IEEE*, March 2012, pp. 2861–2865.

[8] "Amazon does not oversubscribe," http://searchcloudcomputing.techtarget.com/blog/The-Troposphere/Amazon-does-not-oversubscribe.

[9] R. R. Fenichel and J. C. Yochelson, "A LISP Garbage-collector for Virtual-memory Computer Systems," *Commun. ACM*, vol. 12, no. 11, pp. 611–612, Nov. 1969. [Online]. Available: http://doi.acm.org/10.1145/363269.363280

[10] H. Lieberman and C. Hewitt, "A Real-time Garbage Collector Based on the Lifetimes of Objects," *Commun. ACM*, vol. 26, no. 6, pp. 419–429, Jun. 1983. [Online]. Available: http://doi.acm.org/10.1145/358141.358147

[11] "Janitor Monkey," http://techblog.netflix.com/2013/01/janitor-monkey-keeping-cloud-tidy-and.html.

[12] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan, "Jettison: Efficient Idle Desktop Consolidation with Partial VM Migration," in *Proceedings of the 7th ACM European Conference on Computer Systems*, ser. EuroSys '12. New York, NY, USA: ACM, 2012, pp. 211–224. [Online]. Available: http://doi.acm.org/10.1145/2168836.2168858

[13] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin, "LiteGreen: Saving Energy in Networked Desktops Using Virtualization," in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 3–3. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855840.1855843

[14] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986. [Online]. Available: http://dx.doi.org/10.1023/A:1022643204877

[15] "IBM Analytics for Logical Dependency Mapping (ALDM)," http://www-935.ibm.com/services/us/en/it-services/data-center/it-infrastructure-discovery/.

[16] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[17] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang, "Towards Highly Reliable Enterprise Network Services via Inference of Multi-level Dependencies," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '07. New York, NY, USA: ACM, 2007, pp. 13–24. [Online]. Available: http://doi.acm.org/10.1145/1282380.1282383

[18] S. Kandula, R. Chandra, and D. Katabi, "What's Going on?: Learning Communication Rules in Edge Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 87–98, Aug. 2008. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402970

[19] X. Chen, M. Zhang, Z. M. Mao, and P. Bahl, "Automating Network Application Dependency Discovery: Experiences, Limitations, and New Solutions," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 117–130. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855741.1855750

[20] S. Suneja, C. Isci, V. Bala, E. de Lara, and T. Mummert, "Non-intrusive, Out-of-band and Out-of-the-box Systems Monitoring in the Cloud," in *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '14. New York, NY, USA: ACM, 2014, pp. 249–261. [Online]. Available: http://doi.acm.org/10.1145/2591971.2592009