Application of PROV Model for Modeling a VM Overload Mitigating Strategy: Task Eviction

Abdulaziz Albatli[1,2], Lydia Lau[1], Jie Xu[1]

[1] Distributed Systems and Services Research Group
School of Computing, University of Leeds
Leeds, UK
[2] Huraymila College of Science and Humanities
Shaqra University
Riyadh, Saudi Arabia

{sc11a2a, l.m.s.lau, j.xu}@leeds.ac.uk

Abstract. Provenance can be a very important aspect for mass-computation and large-scale computer systems. Cloud environment with its dynamic and scalable nature can benefit from using provenance to better support its management. In this paper, we have demonstrated the application of PROV, a W3C standard, to understand the behavior of Google Cloud from its one-month log data. Its potential contribution to the implementation of task eviction strategy for Virtual Machine overload mitigation will be discussed.

1 Introduction

As defined by W3C, “provenance is a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering a piece of data or a thing” [1]. With regards to distributed systems, Moreau and Groth in [2] stated that provenance can relate to data, documents or recourses since it is a record that computers have produced, processed, and exchanged. Provenance is one essential dimension of process verification, reproducibility, reliability and trust in distributed systems.

W3C’s standard for provenance [3], PROV, requires the following information to be captured in a provenance record: *Entity* - a digital, conceptual or physical thing of which we need to keep the provenance; *Activity* - a process that occurs over a duration of time that act upon entities; and *Agent* - something/someone to which entities and activities are attributed or associated.

Cloud Computing is a rapid evolving paradigm. It delivers virtualized, scalable and elastic resources (e.g. CPU, memory) over a network (e.g. Internet) from off-site data centers to users (e.g. individuals, enterprises, governments), allowing a pay-as-you-go pricing model. National Institute of Standards and Technology's (NIST) defined Cloud Computing as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [4]. Clouds enable users to run high complex operations to satisfy the need of large computation power through resource virtualization. Virtualization is a method to run a number of virtual machines (VM) on a single physical server. Cloud providers tend to over-commit resources aiming to leverage unused capacity and to maximize profits [5]. This over-commitment of resources usually leads to an overload of the actual physical machine, which lowers the performance or crashes the memory [5]. Over-commitment means that a VM can use more resource space than the actual capacity of the physical server. Currently, there are a number of different approaches (strategies) to mitigate the overload, one of which is VM eviction [5]. A number of existing work in the literature [6–9] have examined and evaluated the importance of provenance within a Cloud environment. However, none have actually looked at the usability of PROV with regards to VM's over-commitment and overload. This paper presents a case study on using PROV to model and illustrate the reasoning for task eviction strategy using Google log data.

This paper is organized as follows: Section 2 presents a brief overview of Google log data and its characteristics. Section 3 explains over-commitment and overload situation and their cause and consequences. Section 4 presents our application of PROV model over Google log data. Finally, our future work and conclusion is presented in Section 5.

2 Brief Overview of Google Cloud Log Data

Google workload data consist of a one-month trace of its applications with over 650k jobs (25 million tasks) running across over 12k heterogeneous machines from a Google data center [10]. There are thousands of hosts (machines) that are connected via a high-speed intra-network in one data center. A cluster's scheduler receives and processes a large number of user requests known as jobs; each of which consists of one or more tasks. Every task is assigned with a scheduling priority, and there are 12 different priorities in total. A user can specify a number of constraints for every task submitted, i.e. minimum CPU units, memory size or a specific physical machine as the host. A task can only be in one of the following states: un-submitted, pending, running or dead [11]. All these information are captured in log files. For examples, relevant attributes for a job includes jobID, job name, event type and so on; for a task includes event type, priority, resource for disk, CPU or RAM; and for a machine includes machine ID, capacity for CPU or RAM.

3 Over-Commitment and Overload in Cloud

Virtual machine over-commitment is widely implemented among cloud providers in order to maximize profits and resource utilization [5]. Over-commitment can make use of under-utilized resources, namely CPU, memory, storage and network in the cloud [5]. For any given physical machine, the cloud provider allocates more virtual capacity (VM) than the actual capacity on the physical machine. Every cloud provider has specific policies to determine the amount of over-commitment (over-commitment ratio) [5]. For example, if the ratio is 2 then a server with a capacity of 50 units can accept 100 units.

This method of utilising resources (over-commitment), if not managed carefully, can cause overload on the VMs. This will cause a deteriorating effect on the performance and availability of the cloud service. Even though that 88% of memory overloads are transient (temporary) and lasts for less than 2 minutes [12], it is considered a massive drawback and can still violate the SLAs and QoS agreements, for which the provider need to compensate the client.

4 PROV Application for Task Eviction Strategy

The main contribution of this work is utilising PROV to add the ability to compare or improve an overload mitigating strategy by tracing and understanding the reasoning of those strategies taking place. This understanding will feed into the decision support component of our overall system architecture, which will evaluate and choose the best overload mitigating strategy to go for. This work is the first component of the proposed analytic pipeline, which maps relevant log data into a PROV model. Figure 1 illustrates how task eviction scenario can be represented by a PROV model.

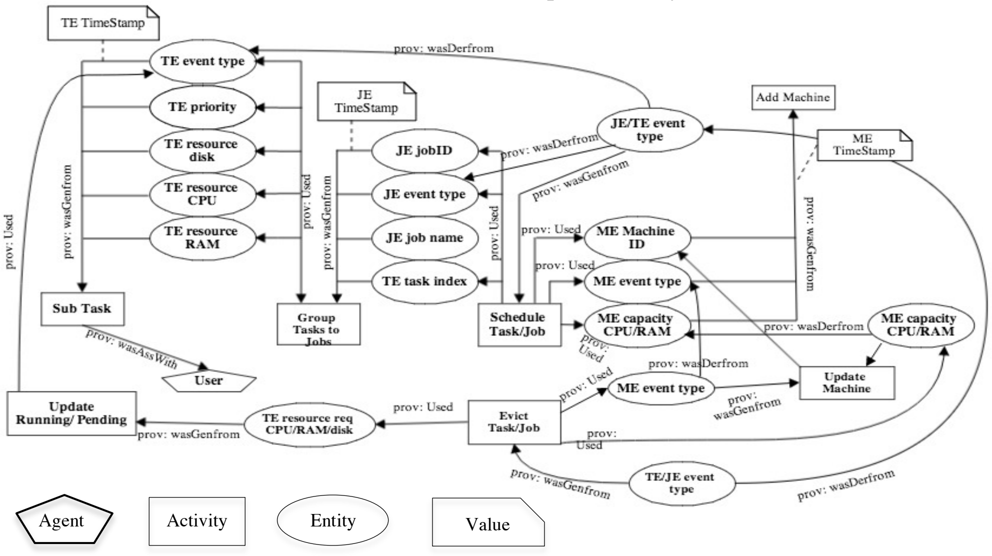

Fig. 1. PROV model of task eviction in Google log data.

Normally, a user submits a task and specifies its scheduling priority. The cluster's scheduler will place the task into its designated VM for execution. There are three possible causes for task eviction [10]. First, physical machine capacity is updated so that its resources are shrunk (i.e. relevant PROV Entity is denoted as 'ME capacity CPU/RAM'). Second, user resources request for a task has been updated for more capacity (i.e. Entities: TE resource CPU/RAM/disk). Third, a new task with a higher priority has been set, so a task with least priority is evicted (i.e. Entity: TE priority).

Following is an illustration of how the PROV model can be used to trace the workflow of a task eviction due to the need to schedule a task with a higher priority. After a task is submitted (Activity: Sub Task), a number of Entities are generated (TE event type, TE priority, TE resource disk/CPU/RAM) and all have a time stamp. Those entities are used by the Activity (Group Task to Job). Then a number of Entities are generated according to the grouping activity (JE jobID, JE event type, JE job name, TE task index) and a time stamp is recorded. The Activity Schedule Task/Job will use those entities and entities related to the designated Machine (ME ID, ME event type, ME capacity CPU/RAM) so that the task/job will be scheduled. When another higher priority task is submitted, the Task Event Activity will react accordingly and processes the eviction of a task with the lowest priority.

5 Conclusion and Future Work

In this paper, we have demonstrated the application of PROV for modeling Google Cloud workload and explain the potential benefits envisioned as a result. In future work, we will model another overload mitigating strategy and complete the process pipeline where we evaluate the best strategy to go for, as a prototype. The need for more real cloud trace logs is immense to enable us continue our investigations.

References

1. W3C: Provenance Definition, http://www.w3.org/TR/2013/REC-prov-dm-20130430/#dfn-provenance. (2013)
2. Moreau, L., Groth, P.: Provenance: An Introduction to PROV. Synth. Lect. Semant. Web Theory Technol. 3, 1–129 (2013).
3. Gil, Y., Miles, S., Belhajjame, K., Deus, H., Garijo, D., Klyne, G., Missier, P., Soiland-Reyes, S., Zednik, S.: PROV Model Primer, http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/. (2013)
4. NIST: The NIST Definition of Cloud Computing. US National Institute of Standards and Technology. (2011) .
5. Baset, S., Wang, L., Tang, C.: Towards an Understanding of Oversubscription in Cloud. USENIX Hot-ICE'12 (2012).
6. Macko, P., Seltzer, M., Muniswamy-Reddy, K.-K.: Provenance for the Cloud. 8th USENIX Conference on File and Storage Technologies (FAST '10) (2010).
7. Muniswamy-Reddy, K.-K., Seltzer, M.: Provenance as First Class Cloud Data. ACM SIGOPS Oper. Syst. Rev. 43, 11 (2010).
8. Imran, M., Hlavacs, H.: Provenance Framework for the Cloud Infrastructure: Why and How? Int. J. Adv. Intell. Syst. 6, 112–123 (2013).
9. Abbadi, I., Lyle, J.: Challenges for Provenance in Cloud Computing. 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP'11) (2011).
10. Reiss, C., Wilkes, J., Hellerstein, J.: Google Cluster-Usage Traces: Format + Schema. Google Inc. (2013).
11. Di, S., Kondo, D., Cappello, F.: Characterizing Cloud Applications on a Google Data Center. 2013 42nd International Conference on Parallel Processing. pp. 468–473. IEEE (2013).
12. Williams, D., Jamjoom, H., Liu, Y.-H., Weatherspoon, H.: Overdriver: Handling Memory Overload in an Oversubscribed Cloud. Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments - VEE '11. p. 205.. (2011).