Opportunistic Virtual Machine Migration for Resource Management in Virtualized Data Centers

M. P. Gilesh, S. D. Madhu Kumar,
Lillykutty Jacob
National Institute of Technology Calicut, Kerala
[gilesh_p140059cs,madhu,lilly]@nitc.ac.in

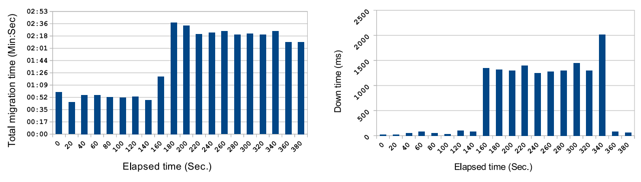
Umesh Bellur
Indian Institute of Technology Bombay
Mumbai, Maharashtra
umesh@cse.iitb.ac.in

1 INTRODUCTION

Virtual Machine (VM) migrations are done in cloud data centers (CDC) in various contexts such as evacuating a host machine for maintenance, balancing the workloads on host machines, optimizing physical resource utilization, and meeting the custom demands of the user applications [5]. However, these migrations are costly in terms of the resource usage and possible service level agreement (SLA) violations during the migration period. The time required for migration is an important parameter and is defined in terms of the size of the VM, the available network bandwidth, the rate of page dirtying, the size of writable working set etc. [6]. Related work in the literature [4] [7] [9] proposed models to predict one or more of these parameters, based on the application behavior or workload characteristics, before scheduling the migration of VMs. In this paper, we propose a model to migrate virtual machines opportunistically such that the SLA violations are minimized. The idea is to find the time instants (called opportunities) at which, if the migration is started, the performance implication is minimal. Scheduling migrations at these opportunities can reduce the count and rigor of SLA violations. We define an offline and an online model for opportunistic migration of VMs.

2 MOTIVATION AND PROBLEM DEFINITION

Each *application instance* in the cloud has its own load behavior, with independent load profile, load level, and load duration. Deployment of an application can be represented as a set of VMs that talk to each other. The VMs may be migrated to a different host for one or more of the reasons stated earlier.



(a) Total Migration Time **(b) Down Time**

Figure 1: Total migration time and down time of migrations starting at different points on the time line

With near-accurate load/workflows prediction tools, [1] [2] [8] [10] the IaaS provider can estimate the parameters that influence VM migration for a longer duration in the future. The idea here is to use this information to decide on when a VM needs to migrate. We motivate the importance of finding an appropriate time for migration using the Figure 1 that shows the total time and down time for the migrations of a VM, at different time instants during the execution of 483xalancbmk[11] [3] workload. From the figure, it can be seen that if the migration starts at 20 seconds from the execution of workload the migration time and downtime are the least. Whereas, if the migration starts at 340 seconds, the total time is 3x and downtime is approximately 200x more compared to that of the migration at 20 seconds. Hence it is important to choose the point of migration appropriately.

We assume that the VM to be migrated and the destination host are already decided, based on the cloud provider's policy, to meet a particular objective. Moreover, a *buffer time*, until the expiry of which migration is worth, is available to complete the migration. The VM hosts application(s) with a dynamic workload, and we do not assume reservation of bandwidth for migrations i.e. the available bandwidth for a migration is also varying. The problem is to find time intervals with in the duration of *buffer time* such that, the migration performed during any of these intervals does not exceed the maximum allowed migration time and it is completed before the expiry of *buffer time*. Opportunities are the starting points of such time intervals. The problem is formally defined as follows.

Let k and l be two host machines in the data center. At time t_0, a VM j on host k is identified for migration to l and the process should be completed before the expiry of *buffer time* $t_d (= t_0 + d)$. Parameter $t^j_{kl}(t_i)$ is the expected time requirement for migrating the VM j from source host k to destination host l, if the migration starts at $t_i (t_0 \leq t_i \leq t_d)$. The time $t^j_{kl}(t_i)$ can be calculated according to migration time estimation models in the literature (Eg.[6]). We define a point t_i in the time line as an *opportunity*, if the migration of the identified VM j can be carried out from this point without any SLA violations (or with the least violation of the SLA). We assume that the migration time is much lesser compared to the *buffer time*. There could be one or more opportunities in the interval (t_o, t_d). For simplicity, we assume the duration of migration as the metric for determining the SLA violation. The objective is to find the opportunities for migration of VM j from k to l when impact of the migration is affordable i.e. to find points t_i $(t_0 \leq t_i \leq t_d)$ in a time series such that the expected time $t^j_{kl}(t_i)$ for migrating VM j starting at time t_i is less than the affordable migration window d_{max} and $t_i + t^j_{kl}(t_i) \leq t_d$.

3 PROPOSED SOLUTION

We propose two models for opportunistic migration based on the level of information available to define an opportunity.

(1) *Offline Model :* wherein the predicted bandwidth and VM dynamics are available for the entire duration d. In such a case, all the opportunities within the buffer time (t_0, t_d) are identified in advance, using the expected time of migration at that point.

(2) *Online Model :* in which only the instantaneous information and the history are known. In this model, the expected time for migration starting at future time instants are not pre-calculated and the present time is either chosen as an opportunity or not, dynamically. The challenge is to decide whether to migrate a VM at an opportunity or wait for upcoming opportunities.

Algorithm 1 Offline Opportunistic Migration

```
1:  Input : The VM $j$, dest. host $l$, buffer time $d$.
2:  Output: Time point $t_m$ for migration.
3:  procedure OFFLINE($j$ , $l$, $d$)
4:      $t_i = t_0$
5:      while $t_i \leq t_i + d$ do
6:          Find $b_i$          ▷ Find average bandwidth to host $l$ at $t_i$
7:          Find $d_i$          ▷ Find average page dirtying rate at $t_i$
8:          $t_i = t_i + s$     ▷ Increment the time by step $s$
9:      end while
10:     $t_i = t_0$             ▷ Initialize the opportunity to $t_0$
11:     $t_{min} = L$           ▷ Initialize $t_{min}$ with large value $L$
12:     while $t_i \leq t_i + d$ do
13:         Find averages of $b_i$ and $d_i$ upto all $t(> t_i)$
14:         Calculate $t_j$ sufficient to migrate $j$ [6]
15:         if $\exists t_j$ and $t_{min} < t_j$ then
16:             Set $t_{min} = t_j$     ▷ Update least migration time.
17:             Set $t_m = t_i$         ▷ Update the opportunity.
18:         end if
19:         $t_i = t_i + s$     ▷ Increment the time by step $s$
20:     end while
21:     Return $t_m$
22: end procedure
```

The offline solution is formally defined as a function in Algorithm 1. For all the points within the *buffer time*, the average bandwidth availability and page dirtying rate between the consecutive points on the time line are calculated. These values are used to estimate the the expected duration of migrations. Sub-arrays of the time instants (a subset of the time line) within the *buffer time* is calculated such that the corresponding migration is expected to complete within the duration of the sub-array. The time point at which the smallest such sub-array starts is selected as the opportunity for migration.

Our experimental setup has two 8-Core x86 servers with 16GB RAM, each running a KVM/QEMU hypervisor and connected by 1 Gbps through a Gigabit Ethernet switch. The VM images are managed using a 1TB shared NFS storage. The virtual machine has 2 cores with 4096 MB RAM and an image size of 40GB. The migration time is calculated by averaging the total time of 20 migrations of a VM running workloads Faban[1], OLTP or FileIO[2]. Proposed LM-O algorithm migrates a VM at the best opportunity identified. The average migration time of these migrations are compared with that of an immediate live migration algorithm (LM-I) i.e. migration starts

[1]http://faban.org

[2]https://github.com/akopytov/sysbench

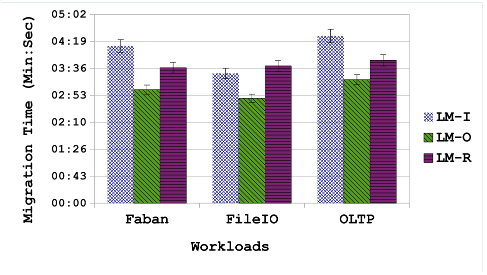


Figure 2: Average migration time of a VM. (for 20 migration instances)

at t_0 and a random model (LM-R) wherein the migration starts at a random time during the *buffer time*.

Figure 2 shows that the proposed offline algorithm (LM-O), reduces the total migration time by 20%, on an average, compared to other methods. An SLA violation can be defined in terms of the maximum migration time or the down time permissible.

4 CONCLUSION AND FUTURE WORK

We have shown that the opportunistic migration reduces the average time for migration. Work is in progress for implementing the online model with good workload prediction techniques.

REFERENCES

[1] Anju Bala and Inderveer Chana. 2016. Prediction-based Proactive Load Balancing Approach Through VM Migration. *Eng. with Comput.* 32, 4 (Oct 2016), 581–592. https://doi.org/10.1007/s00366-016-0434-5

[2] Marko Hoyer, Kiril Schröder, Daniel Schlitt, and Wolfgang Nebel. 2011. Proactive Dynamic Resource Management in Virtualized Data Centers. In *Proceedings of the 2Nd International Conference on Energy-Efficient Computing and Networking (e-Energy '11)*. ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/2318716.2318719

[3] Dawei Huang, Deshi Ye, Qinming He, Jianhai Chen, and Kejiang Ye. 2011. Virt-LM: a benchmark for live migration of virtual machine. *SIGSOFT Softw. Eng. Notes* 36, 5 (Sept. 2011), 307–316. https://doi.org/10.1145/1958746.1958790

[4] V. Kherbache, ÃĽ. Madelaine, and F. Hermenier. 2015. Scheduling Live-Migrations for Fast, Adaptable and Energy-Efficient Relocation Operations. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. 205–216. https://doi.org/10.1109/UCC.2015.37

[5] M. Mishra, A. Das, P. Kulkarni, and A. Sahoo. 2012. Dynamic resource management using virtual machine migrations. *IEEE Communications Magazine* 50, 9 (September 2012), 34–40. https://doi.org/10.1109/MCOM.2012.6295709

[6] Senthil Nathan, Umesh Bellur, and Purushottam Kulkarni. 2015. Towards a Comprehensive Performance Model of Virtual Machine Live Migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC '15)*. ACM, New York, NY, USA, 288–301. https://doi.org/10.1145/2806777.2806838

[7] Minal Patel, Sanjay Chaudhary, and Sanjay Garg. 2016. Machine Learning Based Statistical Prediction Model for Improving Performance of Live Virtual Machine Migration. *Journal of Engineering* (2016), 9. https://doi.org/10.1155/2016/3061674

[8] A. Polze, P. Troger, and F. Salfner. 2011. Timely Virtual Machine Migration for Pro-active Fault Tolerance. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*. 234–243. https://doi.org/10.1109/ISORCW.2011.42

[9] Bane Raman Raghunath and B. Annappa. 2015. Virtual Machine Migration Triggering using Application Workload Prediction. *Procedia Computer Science* 54 (2015), 167 – 176. https://doi.org/10.1016/j.procs.2015.06.019 Eleventh International Multi-Conference on Information Processing-2015 (IMCIP-2015).

[10] Ganesha Shanmuganathan, Ajay Gulati, Anne Holler, Shankari Kalyanaraman, Pradeep Padala, Xiaoyun Zhu, Rean Griffith, and Vmware Inc. 2013. *Towards Proactive Resource Management in Virtualized Datacenters*. Technical Report. VMWare.

[11] SPECcpu. 2006. 483xalancbmk. (2006). https://www.spec.org/auto/cpu2006/Docs/483.xalancbmk.html