2017

COMS: Customer Oriented Migration Service

Kai Huang
kaihuang@u.boisestate.edu

Xing Gao
College of William and Mary, xinggao@cs.wm.edu

Fengwei Zhang
fengwei@wayne.edu

Jidong Xiao
jidongxiao@boisestate.edu

Recommended Citation

Huang, Kai; Gao, Xing; Zhang, Fengwei; and Xiao, Jidong, COMS: Customer Oriented Migration Service (2017). *2017 IEEE 10TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING (CLOUD)*. 10.1109/CLOUD.2017.94

2017 IEEE 10th International Conference on Cloud Computing

COMS: Customer Oriented Migration Service

Kai Huang*, Xing Gao†, Fengwei Zhang‡, Jidong Xiao*

* Boise State University, Boise, ID, USA
kaihuang@u.boisestate.edu, jidongxiao@boisestate.edu
† College of William and Mary, Williamsburg, VA, USA
xinggao@cs.wm.edu
‡ Wayne State University, Detroit, MI, USA fengwei@wayne.edu

***Abstract*— Virtual machine live migration has been studied for more than a decade, and this technique has been implemented in various commercial hypervisors. However, currently in the cloud environment, virtual machine migration is initiated by system administrators. Cloud customers have no say on this: They can not initiate a migration, and they do not even know whether or not their virtual machines have been migrated. In this paper, we propose the COMS framework, which is short for "Customer Oriented Migration Service". COMS gives more control to cloud customers so that migration becomes a service option and customers are more aware of the migration process. We have implemented a suite of modules in our COMS framework. Our evaluation results show that these modules could either bring performance benefit to cloud customers, or mitigate security threats in the cloud environment.**

I. Introduction

Virtual machine (VM) live migration has been studied for more than a decade. It has been implemented in various commercial hypervisors, and has become an indispensable feature in today's virtualization technology. Cloud vendors use VM migration to achieve the goals such as: load balance, resource consolidation, system upgrade/maintenance, fault tolerance. These are all the positive side of VM migration. The problem is, today's virtualization technology enables cloud administrators to do VM live migration, yet it does not enable cloud customers to do VM live migration. When cloud vendors decide to migrate VMs, cloud customers have no choice but to accept; on the other hand, when cloud customers want to migrate their VMs, they simply can not do it. So why is it necessary to give customers more flexibility (or freedom) to drive or initiate VM migration? The benefits are threefold:

First, availability. At some stage of the migration, it incurs a certain amount of service downtime. When that happens, unfortunately, cloud customers have to involuntarily tolerate the service downtime. It is natural to think that, if service downtime is inevitable, can we let the customers choose when to tolerate that downtime? If so, customers can then choose a time window when the migration causes least impact to their business.

Second, performance. if cloud customers can initiate virtual machine migration, then they can mitigate the interference problem between co-located virtual machines [1], [2]. Since the contention of hardware resources could degrade the performance significantly, once customers notice that degradation, they can migrate their virtual machines to a different host machine, in the hope that the same level of interference may not be existing on that new host machine.

Third, security. When migration is initiated, the storage (e.g., memory or disk) is transferred over the network in plain text. This is not secure, as demonstrated by [3]. Attackers, by monitoring the packets transferred over the network, can perform man-in-the-middle attack and eventually compromise the virtual machine. If we enable customers to initiate the migration, customers should be able to choose to encrypt their whole system, or parts of their system in case that encrypting everything incurs too significant performance penalty.

In this paper, we propose the COMS framework, a framework that offers customer oriented migration service. We enable customers to initiate VM migration, let customers be aware of the migration.

The contribution of our work can be summarized as follows. We propose the COMS framework. We present the design and implementation of the COMS framework in a Linux KVM/Qemu based virtualized environment. The framework currently includes four modules: 1) performance monitoring and migration, 2) geolocation migration, 3) encrypted migration, and 4) random migration. We evaluate the COMS framework with various experiments. Our evaluation results show that the first two modules could bring performance benefits to cloud customers, and the latter two modules could mitigate security threats in cloud.

II. Design and Implementation

We designed and implemented the COMS framework in a Linux KVM based virtualized environment. Typically, VM live migration involves two physical machines. One source machine, and one destination machine. Our goal is to integrate existing VM live migration techniques into a web interface, so we need three physical machines: source, destination, and a web server. Generally, the web server will take requests from web front end - submitted by users. The web server will then parse these requests and determine users' needs. Based on users' needs, the web server would then call a corresponding module to serve users' needs. In the following, we describe the design and implementation of these four modules.

A. Module A: Performance Monitoring and Migration

Module A includes two components: monitoring agent and migration agent. The monitoring agent is located inside the virtual machine, and should be provided by the cloud customer. The migration agent is located on the source side host machine.

2159-6190/17 $31.00 © 2017 IEEE
DOI 10.1109/CLOUD.2017.94

These two agents communicate with each other via a pre-defined interface, which in our implementation, is a shared file that both agents can read from and write to.

When the customer's workload is running, the monitoring agent should also keep running, and monitor the workload's performance. When its performance degrades to a certain pre-defined threshold, the monitoring agent should send a signal via the pre-defined interface to the migration agent. Upon receiving that signal, the migration agent would first notify the destination Qemu process to launch a fake VM image and enter into listening mode, and then it issues a migration command to the source Qemu process to initiate a migration procedure. Migration usually takes a certain amount of time, therefore, during migration, the monitoring agent should periodically check the migration status. This can be achieved by communicating with the source Qemu process. Once migration is completed, the monitoring agent should notify the cloud customer, or write to a log file that is visible to the customer.

B. Module B: Geolocation Migration

Module B also includes two components: analysis agent and migration agent. The analysis agent is located inside the virtual machine, and should be provided by the virtual machine owner; the migration agent is located on the source side host machine. These two agents also communicate with each other using a pre-defined interface, such as a shared file.

The analysis agent's job is to analyze the source of clients (who access service running inside the virtual machine), by either using tools such as geoiplookup [4], or searching in some public available database. If the majority of the clients are from some specific region, the analysis agent will then notify the migration agent via writing into that shared file. The migration agent would then take this as a signal to migrate the virtual machine to a host machine that is closer to or within that region.

C. Module C: Encrypted Migration

Module C allows users to enable or disable encryption. If the user chooses not to encrypt migration, then the web backend would issue default migration request to the source Qemu process. If the user chooses to encrypt migration, then in the setup phase, the web backend would notify the destination machine to get ready to receive encrypted migration. This requires more configuration work on both the source and the destination side, as we need to generate a GPG key in advance, so that the source side will later on using the GPG key to encrypt the memory and disk image, and then the destination side will use the GPG key to decrypt the memory and disk image.

D. Module D: Random Migration

We define a list of physical machines for migration. If the user chooses to enable random migration, module D will migrate the virtual machine after some random interval, to a server that is randomly picked up from the above physical machine list. Such procedure will be repeated time and time again, until the user selects to stop it.

From security perspective, we explain why random migration would be useful. Nowadays, attacks happening in cloud mainly include two steps: First, find out the location of a target virtual machine and launch their own virtual machine on the same host machine; next, perform co-resident attacks. As the representative work of the first step, Ristenpart et al. [5] revealed that in leading cloud service such as Amazon EC2, sophisticated attackers can put their virtual machines co-resident with the victim's virtual machine. Once attackers can co-locate their own virtual machine with the target virtual machine on the same host machine, as to the second step of the attack, there are many offense techniques attackers can utilize. These techniques will cause a denial-of-service, performance degradation [2], or information leakage [6], [7] to the target virtual machine. However, both the first step and second step will take a certain amount of time. For example, in [8], the authors show that the time spent to achieve co-residence with a specific target VM in a public cloud is above one hour. In [7], the covert channel attack on a target co-resident virtual machine requires several minutes of setup time, and then it takes even more time to steal sensitive information via the covert channel. Therefore, if we can migrate the specified virtual machine in a random fashion with the interval of a few minutes or even less, then the attackers may neither be able to find the location of their target virtual machine, nor be able to perform the co-resident attacks.

III. EVALUATION

In this section, we mainly describe our evaluation for module A, B, C, D. We used two machines (as the source and destination) located in the same local area network (LAN) to perform experiments for module A, C, and D. We call these two physical machines Host 1 (Intel Core i7-4790 3.60GHz, 16G mem, Fedora 22) and Host 2 (Intel Core i7-4790 3.60GHz, 16G mem, Fedora 22). For module B, i.e. geolocation migration, we use two machines located at two different states - to emulate that they are in two different data centers. We call these two physical machines Host 3 (Intel(R) Xeon(R) CPU E5-1630 v3 @ 3.70GHz, 32G mem, CentOS 7.1) and Host 4 (Intel(R) Xeon(R) CPU E5520 @ 2.27GHz, 32G mem, Ubuntu 14.04.3). Qemu 2.4.0 was used for evaluating module A, C, D; and Qemu 2.6.1 was used for evaluating module B. The virtual machine images (Ubuntu 14.04.1 and Fedora 16, 2G Mem, 40G Disk) were stored on Host 1 and Host 3. The migration we performed was pre-copy migration [9]. In pre-copy migration, there are two metrics: total migration time and service downtime. The former refers to the total time taken between the initiation and the completion of the migration, while the latter means the duration in which the VM is suspended and service is not available.

A. Module A: Performance Monitoring and Migration

To evaluate module A, we ran a computational workload called Cuadro [10] inside a virtual machine (we call it VM1). Then we launched four more other virtual machines, and made them busy, so as to compete the resource with VM1. We expected this competition to downgrade the performance of that workload running inside VM1. When the performance of Cuadro in VM1 reached a low point, migration was triggered. Figure 1 shows our experimental results.

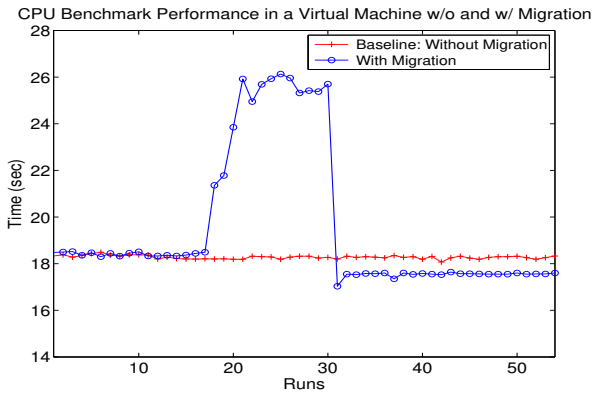

Fig. 1: Performance monitoring and migration.

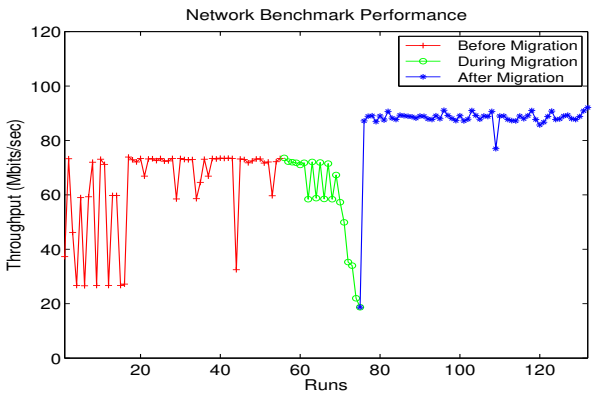

Fig. 2: Geographic location migration.

The baseline (red curve, plus sign marker) represents the Cuadro performance in the first virtual machine, with four other virtual machines staying idle on the same host. It can be seen from the Figure 1 that the baseline performance was very stable: every single run consumed about 18 seconds.

Then that blue curve (circle marker) in Figure 1 demonstrates the performance benefit when module A was enabled. We repeated the experiment, but at some point, we started to run CPU intensive benchmarks on the other virtual machines. These virtual machines would compete CPU resource with the first virtual machine (VM1). When all the virtual machines were running CPU benchmarks, it can be seen from Figure 1 that VM1's performance was degraded to 50% worse than the baseline. With the migration option we implemented, this could be solved. When VM1 noticed that its benchmark's performance was degrading to a certain level, it notified the host to migrate. Therefore we migrated it to another machine, which in our experiment was an idle machine (no other virtual machine was running at all). Therefore we can see the performance was bounced back, and even better than the original one.

B. Module B: Geolocation Migration

Figure 2 depicts our experimental results for evaluating module B. We have two servers, Host 3 and Host 4, which were located at two different states: state A and state B. The virtual machine was originally located at state A, we ran iperf [11] server inside the virtual machine. We ran iperf client from another machine, which was located at state C.

The bandwidth between the server in state A and the client was not stable, and it ranged from 30 to 70 mbps (red part of the curve, plus sign marker); after a certain number of runs, we decided to migrate the virtual machine from state A to state B. It takes time to migrate a virtual machine, in particular across states (in our case, it was over the Internet, not over a LAN). During migration, the throughput downgraded a bit (green part of the curve, circle marker). However, when the migration was finished, we see the throughput was stable and much better than before, about 90 mbps (blue part of the curve, asterisk marker).

C. Module C: Encrypted Migration

To evaluate module C, we first made the virtual machine idle, and measured the cost of encrypted migration. For both encrypted migration and non-encrypted migration, we measured the migration time and downtime respectively. We ran computational intensive benchmark Cuadro in the virtual machine.

Figures 3 and 4 illustrate the experimental results. The story behind Figure 3 is, we let the virtual machine be idle. When we migrated a virtual machine in an encrypted way, it inevitably increased both the migration time, and service down time. It can be seen from these two figures that migration time was roughly increased by 30%. Downtime was even worse: it could be doubled or even tripled. Figure 4 is telling the same story. The difference is, instead of letting the virtual machine be idle, we let it run the network intensive workload iperf.

There are two reasons why encryption increases migration time significantly. On one hand, encryption and decryption consumes more CPU cycles, thus increases the computation time, and slows down the migration; On the other hand, Qemu itself does not do encryption. In fact, on both the source side and the destination side, Qemu has to invoke a separate program named gpg to encrypt and decrypt memory pages. Qemu would then communicate with gpg via pipes, and such a inter-process communication model brings a lot of context switches, thus incurs considerable performance overhead.

D. Module D: Random Migration

Similar to encrypted migration, random migration comes with a price. Generally, migration consumes CPU cycles, memory and network bandwidth. When migration is in process, the workload running inside the virtual machine, or running on the host machine, may experience performance loss.

To evaluate the performance overhead of random migration, we conducted the following experiment: We ran the iperf benchmark on the host machine, and measured the throughput from the host machine to another physical machine. The experiment includes four steps: First, there was no migration. We just ran the iperf benchmark to measure the baseline throughput. Next, we triggered the random migration, and did

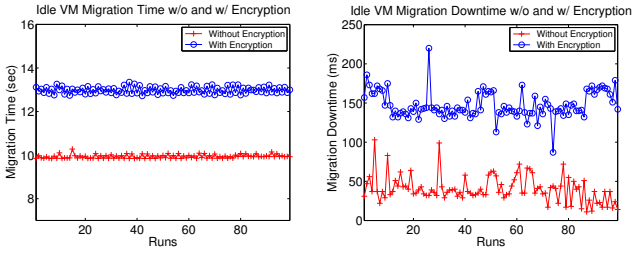

Fig. 3: Encrypted migration time (left) and service downtime (right) when VM is idle.

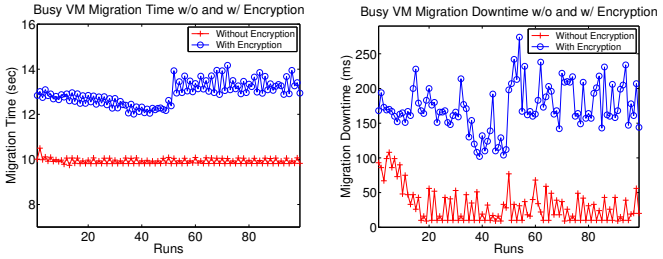

Fig. 4: Encrypted migration time (left) and service downtime (right) when VM is running workload.

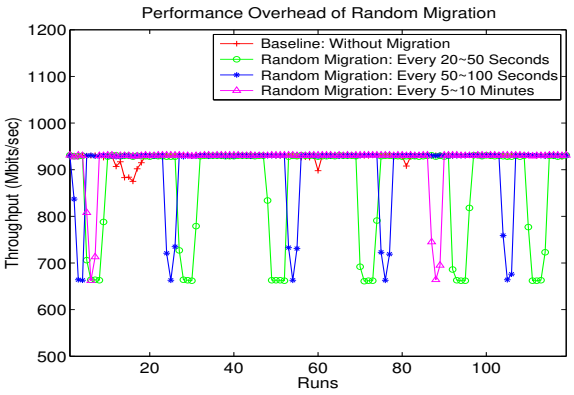

Fig. 5: Random migration.

the random migration every 20 to 50 seconds. Third, we triggered the random migration, but did the random migration in a different frequency - we did it every 50 to 100 seconds. Finally, we triggered the random migration, but the frequency was every 5 to 10 minutes. Figure 5 shows the result. It can be seen that the baseline throughput was around 930 Mbits/sec. Virtual machine migration indeed caused performance degradation on the host, and the throughput degraded to about 670 Mbits/sec. In other words, the performance loss was about 28%. The more frequent we migrated the virtual machine, the more performance loss we experienced: in Figure 5, among 120 runs of the iperf benchmark, we encountered major performance drops twice when migration frequency was every 5 to 10 minutes. We encountered major performance drops five times when the migration frequency was increased to every 50 to 100 seconds, and six times when the migration frequency was increased to every 20 to 50 seconds.

We then repeated the same experiment but ran the iperf benchmark inside a virtual machine. However, we did not observe any obvious performance penalty. There are two reasons: First, the baseline throughput inside the virtual machine was very unstable. It fluctuated a lot even if we did not migrate the virtual machine. Second, the migration traffic was initiated by the host machine, rather than the virtual machine itself. Therefore, the throughput of the virtual machine did not follow the same pattern as the throughput of the host machine during migration.

IV. Conclusion

In this paper, we present the design, implementation, evaluation of a new framework - COMS, which offers a customer oriented virtual machine migration service in cloud environment. We demonstrate how convenient this framework could be, and what kind of benefits it could bring to cloud customers. Once such a framework is deployed, cloud customers have the privilege to decide when to migrate, what to migrate, how to migrate, and where to migrate. We believe such a flexible new model would make cloud computing more attractive.

References

[1] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of i/o workload in virtualized cloud environments," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*. IEEE, 2010, pp. 51–58.

[2] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart, and M. M. Swift, "Resource-freeing attacks: improve your cloud performance (at your neighbor's expense)," in *Proceedings of the 2012 ACM conference on Computer and Communications Security (CCS)*. ACM, 2012, pp. 281–292.

[3] J. Oberheide, E. Cooke, and F. Jahanian, "Empirical exploitation of live virtual machine migration," in *Proc. of BlackHat DC convention*. Citeseer, 2008.

[4] "How to look up the geographic location of an ip address from the command line," http://xmodulo.com/geographic-location-ip-address-command-line.html.

[5] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*. ACM, 2009, pp. 199–212.

[6] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: high-speed covert channel attacks in the cloud," in *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, 2012, pp. 159–173.

[7] J. Xiao, Z. Xu, H. Huang, and H. Wang, "Security implications of memory deduplication in a virtualized environment," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2013, pp. 1–12.

[8] Z. Xu, H. Wang, and Z. Wu, "A measurement study on co-residence threat inside the cloud," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 929–944.

[9] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI) -Volume 2*. USENIX Association, 2005, pp. 273–286.

[10] "Cuadro cpu benchmark," http://sourceforge.net/projects/cuadrocpubenchm.

[11] "iperf - the network bandwidth measurement tool," https://iperf.fr/.