Auditing CPU Performance in Public Cloud

Qiang Huang[1], Lin Ye[1], Xinran Liu[2], and Xiaojiang Du[3]
[1]*School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China*
{*qhuang0409@gmail.com, hityelin@hit.edu.cn*}
[2]*School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China, lxr@cert.org.cn*
[3]*Dept. of Computer and Information Sciences, Temple University, Philadelphia, PA, USA, dxj@ieee.org*

***Abstract*—Cloud computing services offer elastic computing and storage to end-users over the Internet in a pay-as-you-go way. Many businesses have started using cloud computing. A Service Level Agreement (SLA) between a cloud service provider (CSP) and a user is a contract that specifies the resources and performances that the cloud should provide. However, a CSP has the incentive to cheat on SLA, e.g., providing users with less CPU and memory resources than that specified in the SLA, which allows the CSP to support more users and make more profits. Unfortunately, there are no tools to allow users to verify the SLA. We study the important issue of verifying SLA in a semi-trusted (or untrusted) cloud. In this paper, we focus on the verification of CPU speed, which is an important metric in cloud SLA. We propose a lightweight stealthy test algorithm that can check if a CSP provides the CPU speed as specified in the SLA. Using real experiments, we show that the algorithm can detect cloud cheating on CPU speed (i.e., SLA violations) in a stealthy way.**

Keywords-Cloud computing; auditing; CPU

I. Introduction

Cloud computing is gaining increasing attentions because it allows users to rent computing resources in a pay-as-you-go way. Due to the economic, flexible and elastic usage, a number of major IT companies have started offering cloud computing services. Meanwhile, an increasing number of enterprises are migrating their tasks to cloud environment.

Specified by Service Level Agreements (SLAs) between cloud service providers (CSPs) and users, cloud computing promises the users the expected resources and performances, such as memory, CPU, storage, and so on. For example, a small instance of the Amazon EC2 has the following configurations [1]: 1.7 GB memory, one 1.0-1.2 GHz Opteron or Xeon processor, and 160 GB storage. The price that a user pays to the CSP is closely related with the SLA. However, a user's computations run in a virtual machine (VM) in the CSP. How does a user know if he/she is getting the physical memory size or CPU speed as specified in the SLA? A CSP is a profit-based company, and it may provide less resources to a user, which allows the CSP to support more users and make more profits.

It is difficult for a user to verify the SLA because the CSP has complete controls of the resources, including physical machines, hypervisors, VMs, and so on. An untrusted CSP can easily defeat the existing SLA assessment techniques by interfering with the process. In this paper, we present a lightweight stealthy test algorithm for CPU speed verification. Our algorithm utilizes the relationships between the video transcoding parameters(corresponding to the video quality) and the conversion time. Our contributions are summarized as follows.

- We propose a lightweight stealthy test algorithm to detect whether the cloud provides enough CPU speed.
- Using real experiments, we show that the algorithm can detect cloud cheating on CPU speed (i.e., SLA violations) and the overhead is small.

The rest of this paper is organized as follows. Section II discusses the related work on SLAs. Section III describes the system and threat models. Section IV presents the stealthy test algorithm on CPU speed. Section V evaluates the effectiveness of the algorithm for detecting SLA violations on CPU, followed by the conclusion in Section VI.

II. Related Work

Brandic et. al [2] propose a layered cloud architecture to model the bottom-up propagation of failures, and uses it to detect SLA violations by mapping resource metrics to SLA parameters. There are several approaches for SLA assessment with a focus on accurately measuring or estimating Quality of Service parameters. Sommers et. al [3] propose a novel active measurement methodology to monitor whether the characteristics of measured network path are in compliance with performance targets specified in SLAs. Serral-Gracia et. al [4] propose a new passive traffic analysis method for on-line SLAs assessment, which reduces both the need for measuring QoS metrics as well as the interactions between the ingress and egress nodes in the network. Wang and Eugene [5] present a quantitative study of the end-to-end networking performance among Amazon EC2 from users' perspective, and conclude that virtualization can cause significant unstable throughput and abnormal delay variations. Li et. al [6] compare the performance and cost of four major cloud providers (Amazon, Microsoft, Google and Rackspace).

However, none of the previous works consider an untrusted cloud that can interfere with the measurement/monitoring process. A malicious cloud may intentionally modify, delay, drop, inject or preferentially treat packets

978-0-7695-5024-4/13 $26.00 © 2013 IEEE
DOI 10.1109/SERVICES.2013.39

in order to disrupt the measurement. Goldburg et. al [7] consider the presence of an adversary (located in the middle of a path), but their threat model is different from ours where the adversary (the cloud) is at the end of a path. In addition, they focus on the networking SLAs (such as packet-loss rate and delay). In this paper, we study a different SLA parameter - the CPU speed of a VM.

III. Background

A. System and Threat Model

There is a public cloud service provider that offers their virtual machines to end-users over the Internet. A user submits his/her computation-intensive tasks to the cloud, such as video conversion tasks from video hosting websites. The two parties sign a SLA that states the detail of the resources and performances that the cloud should offer, including memory size, CPU speed, storage size, and so on. Then, the user uploads his/her data and starts the task. At the end, the user gets the computation results and pays for the bill.

A CSP is a profit-based company, and it has the incentive to cheat on SLA. For example, a CSP may keep CPU frequency in a low level, which allows the CSP to support more users and make more profits. In fact, an important factor that a user cares about is the actual CPU speed to transcode the videos. If a CSP cheats on CPU speed, it will cause the user's computation to run more time and the user will have to pay more money.

We assume that the CSP has complete controls of its own resources, including physical machines, hypervisors, VMs, et al. If the CSP detects it is being tested, the CSP has the ability to immediately reallocate the resources (e.g., CPU speed) as specified in the SLA. Hence, we want to make the test stealthy.

B. Problem Statement

In order to stealthily verify the SLA of a CSP, the key is to design a test that is relevant with (or the same as) the computation task running in the cloud. In this paper, we focus on one of the most important metrics - CPU, which determines the running time of a task, especially for computation-intensive applications. Nowadays, it is common to use public cloud to perform massive computation, such as matrix computation, video processing and bio-informatics computation. As a case of study, this paper considers the video processing applications. It is well-known that current video hosting websites, such as YouTube and youku.com, provide users with several different versions (e.g., standard-definition, high-definition and full high-definition) of the same video content. Users can choose a version based on their bandwidth and preference. Video conversion requires a lot of computations, which is suitable to run in a cloud. Usually a user submits hundreds of original videos to the cloud in batch, and lets the cloud process the videos. A dishonest cloud may allocate less CPU speed to the user's VM, either for some or all of the video processing.

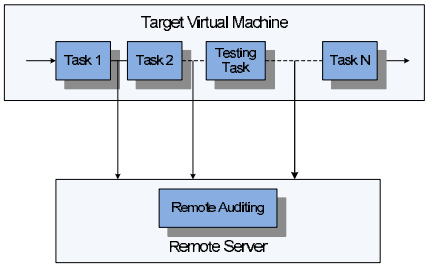

Figure 1. Stealthy Auditing on VM CPU

We want to design a test algorithm that is stealthy to the cloud and can detect any cheating of the CSP on the VM CPU speed.

IV. Stealthy Auditing on VM CPU Speed

To enable CPU auditing in VM under the aforementioned models, the scheme should achieve the following security and performance properties:

- **Stealthy**: To prevent a CSP from dynamically changing CPU speed allocation, the auditing process should be stealthy.
- **Accuracy**: The CPU load may fluctuate over time. The test should measure the CPU for a sufficiently long duration.
- **Efficiency**: The test should have low communication and computation overheads.

The process of stealthy test is illustrated in Fig. 1. Some pre-selected testing videos are randomly mixed up with the normal videos. Then the videos are uploaded to the target VM for processing. A log records the processing time of each video. The user knows the start time and the end time of the video processing, and hence knows the total processing time. The sum of the processing time of each video should equal to the total processing time. If a dishonest CSP reduces the log time of some video processing, it must increases the log time of other video processing such that the total processing time does not change.

A. Evaluation on FFmpeg

In this work, we choose the FFmpeg [8] as the tool for video conversion. FFmpeg is a very fast audio/video converter that can also grab audio/video from a live source. It can convert between arbitrary sample rates and resize a video on the fly with a high quality polyphase filter.

We want to find out the impact of different parameters on video conversion, and we conduct experiments on two VMs *VM-Low* and *VM-High* under the Xen [9] environment, running in two separate physical machines. The configurations of the two physical CPUs are the same: Quad-Core AMD Opteron Processor 2382 at 1891.020MHz. However, we set the cap of the *VM-Low* CPU to 50 (which means the

Table I
THE EVALUATION OF COMMON PARAMETERS

qscale			framerate			bitrate		
Value	VM-High (s)	VM-Low (s)	Value (fps)	VM-High (s)	VM-Low (s)	Value (kb/s)	VM-High (s)	VM-Low (s)
6	69.601	156.648	24	66.231	148.844	2000	70.852	155.847
5	70.437	157.758	23	65.227	146.660	1800	69.581	155.618
4	70.927	158.290	22	64.122	146.300	1600	69.211	154.218
3	72.186	161.258	21	62.880	143.243	1400	68.462	153.124
2	73.829	165.428	20	61.839	139.902	1200	68.158	152.106

max. CPU utilization is 50%), while the cap of *VM-High* is 0 (max. CPU utilization is 100%). Then, we measure the effects of three parameters (namely qscale, framerate and bitrate) on converting a video respectively. Table I shows the conversion time of using FFmpeg under different parameters. The results show that the conversion time closely depends on the parameter settings. The parameters qscale and framerate have more impacts on the conversion time than bitrate. From Table I, we can get some useful rules: 1)**R1**: the lower the qscale is, the longer the conversion time is; 2)**R2**: the higher the framerate is, the longer the conversion time is; and 3)**R3**: the larger the bitrate is, the longer the conversion time is.

B. The Stealthy CPU Testing Algorithm

The stealthy CPU testing algorithm is described below:

1) Suppose there are a total of *M* videos. The auditor randomly selects *r* videos as the testing videos.
2) The auditor configures a VM in our local machine. The VM has the same configuration (such as CPU speed and memory size) as the VM in the cloud.
3) The conversion of the *r* testing videos are run in the local VM, and the auditor records the conversion time as the baseline.
4) The auditor uploads the *M* videos into the cloud VM for conversion.
5) The auditor requests the cloud to log the conversion time of each video.
6) The auditor also records the start time (t_1) and end time (t_2) of the conversion of all the *M* videos.
7) Denote the sum of each video conversion time (logged by the cloud) as *s*. After all the video conversions in the cloud are done, the auditor first checks if $s = t_2 - t_1$. Then the auditor compares the logged conversion time of each testing video with the baseline. If the logged conversion time is much longer than the baseline, then this indicates the cloud is cheating.
8) If the cloud does not provide enough CPU, then the conversion time of a video would be longer than it should be. To avoid being detected, the cloud may change the logged conversion time to a smaller value. For example, when the cloud is cheating (i.e., providing less CPU), the conversion time of a video is 90 seconds. To hide its cheating, the cloud records 70 seconds to the log. However, by doing so, the sum of all the logged conversion time - s would be smaller than $t_2 - t_1$. Hence, this kind of cheating can be detected.
9) Another cheating that a cloud could do is to reduce the logged time of some videos, and at the same time increase the logged time of some other videos, such that the sum $s = t_2 - t_1$. This cheating will have to increase the logged conversion time of some videos. If one of the videos is a testing video, then we can detect the cheating (by comparing the logged conversion time with the baseline).
10) A cloud may only cheat during part of the conversions (e.g., during the first hour), instead of the entire conversion process. In addition to the above main algorithm, the following approach is also used to detect such cloud cheating: Among the total *M* videos, there are videos of the same content to be converted with different parameters, such as framerate and qscale. The auditor places these videos at different locations of the video batch. The conversion time of these videos should follow the rules in Table I. After all the conversions are done, the auditor compares the logged conversion times of these videos and check if they follow the rules. If not, then it indicates a cloud cheating.

V. EXPERIMENTS

We use real experiments to evaluate the performance of our stealthy CPU testing algorithm. We use two machines with the same physical CPU: Quad-Core AMD Opteron Processor 2382 at 1891.020MHz. Xen is used as the hypervisor. We simulate the process that a user wants to convert 100 videos in a cloud, and the CSP intentionally cheats on CPU speed by three different percentages (CPU cap 30%, 50% and 70%). We also simulate the cheating at three different parts of the video conversion process, that is, the cheating happens for video indexes 1 to 33; 34 to 66; and 67 to 100; respectively. Note if a cloud cheats during the entire video conversion process, then it is easier to detect the cheating than in the partial cheating case. We select 10 testing videos and randomly mixed them with 90 other videos. In order to also test the approach in step 10 of algorithm (subsection IV.B), 5 of the testing videos are from the same video content to be converted with different framerates, while the other 5 videos are from the same video content to be converted with different qscales. Other configurations of the

Table II
CHEATING DURING THE FIRST 1/3 DURATION

Position	Parameter	CPU Cap		
		30%	50%	70%
4	framerate=22	747.192	447.213	316.369
10	qscale=2	911.979	543.651	384.854
17	qscale=6	856.897	506.673	356.681
38	qscale=4	199.739	200.022	200.556
47	framerate=23	175.806	176.226	176.311
48	framerate=20	168.758	168.576	169.070
66	qscale=3	203.953	203.947	205.333
80	framerate=24	178.542	178.674	178.477
88	qscale=5	197.069	197.608	197.783
97	framerate=21	170.928	171.160	171.798

Table III
CHEATING DURING THE SECOND 1/3 DURATION

Position	Parameter	CPU Cap		
		30%	50%	70%
4	framerate=22	172.648	173.440	173.159
10	qscale=2	210.656	210.958	210.860
17	qscale=6	194.952	195.961	195.212
38	qscale=4	873.653	515.261	363.590
47	framerate=23	758.009	453.018	320.639
48	framerate=20	725.665	431.945	305.670
66	qscale=3	887.431	524.622	372.145
80	framerate=24	177.922	178.522	178.122
88	qscale=5	196.821	197.716	197.232
97	framerate=21	170.207	171.940	170.504

Table IV
CHEATING DURING THE LAST 1/3 DURATION

Position	Parameter	CPU Cap		
		30%	50%	70%
4	framerate=22	173.633	174.085	173.716
10	qscale=2	211.234	210.608	210.546
17	qscale=6	195.715	196.026	195.342
38	qscale=4	200.182	200.111	199.275
47	framerate=23	176.185	176.134	176.150
48	framerate=20	168.866	169.047	168.421
66	qscale=3	204.394	203.956	203.809
80	framerate=24	772.088	456.599	325.124
88	qscale=5	857.710	513.485	360.064
97	framerate=21	739.306	437.646	312.514

virtual machines are the same. We log the video conversion times.

The experimental results are given in Table II, III, and IV. The results show that when cloud is cheating the conversion time is much longer than the baseline (when there is no CPU cheating). For example, the conversion time of video 4 is 747s, 447s, and 316s when the CPU cap is 30%, 50% and 70% (i.e., Position 4 in Table II), respectively, which are much larger than the conversion time (about 178s) of the same video when no cheating (i.e., Position 80 in Table II). In addition, according to the rule **R2**, the conversion time of Position 4 in Table II should be smaller than that of Position 47. However, the logged times in Table II do not satisfy the rules, which indicates a cloud CPU cheating.

VI. CONCLUSION

In this paper, we presented a lightweight stealthy test algorithm on VM CPU speed. The algorithm is designed for video batch processing applications and may be applied to other computations as well. By randomly selecting a few testing tasks in the batch, our algorithm can detect SLA violations of cloud on CPU speed. Note the testing tasks are part of the original computation tasks, which need to be run in the cloud anyway. Hence, the overhead of our algorithm is very small. Our experimental results demonstrated that the algorithm can effectively detect cloud SLA violations on CPU speed.

ACKNOWLEDGMENT

This research was supported in part by the China National Basic Research Program (973 Program) under grants 2011CB302605, the China National High Technology Research and Development Program (863 Program) under grant 2011AA010705, the National Science Foundation of China (NSF) under grants No. 61100188 and No. 61173144; and by the US National Science Foundation under grants CNS-0963578, CNS-1022552, and CNS-1065444.

REFERENCES

[1] Amazon EC2 Instance Types, [Online]. Available: http://aws.amazon. com/ec2/instance-types

[2] I. Brandic, V. C. Emeakaroha, M. Maurer, S. Dustdar, S. Acs, A. Kertes, and G. Kecskemeti, "LAYSI: A layered approach for SLA-violation propagation in self-manageable cloud infrastructures," *Proceedings of 34th Annual IEEE Computer Software and Applications Conference Workshops*, pp. 366-370, 2010.

[3] J. Sommers, P. Barford, N. Duffield, and A. Ron, "Multi-objective monitoring for SLA compliance," *IEEE/ACM Transactions on Networking*, vol. 18, issue. 2, IEEE Press: NY, USA, pp. 652-665, 2010.

[4] R. Serral-Gracia, M. Yannuzzi, Y. Labit, P. Owezarski, and X. Masip-Bruin, "An efficient and lightweight method for Service Level Agreement assessment," *Computer Networks*, vol. 54, issue. 17, Elsevier: New York, NY, USA, pp. 3144-3158, 2010.

[5] G. Wang and N. T. Eugene, "The impact of virtualization on network performance of Amazon EC2 data center," *Proceedings of the 29th IEEE Conference on Computer Communications*, pp. 1163-1171, 2010.

[6] A. Li, X. Yang, S. Kandula, and M. Zang, "CloudCmp: comparing public cloud providers," *Proceedings of the 10th Internet Measurement Conference*, ACM: New York, NY, USA, pp. 1-14, 2010.

[7] S. Goldburg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality monitoring in the presence of adversaries," *Proceedings of the 2008 ACM SIGMETRICS on Measurement and Modeling of Computer Systems*, ACM: New York, NY, USA, pp. 193-204, 2008.

[8] FFmpeg, [Online]. Available: http://www.ffmpeg.org

[9] Xen Hypervisor, [Online]. Available: http://www.xen.org