Is today's public cloud suited to deploy hardcore realtime services?

A CPU perspective

Kjetil Raaen[1,2,3], Andreas Petlund[2,3], and Pål Halvorsen[2,3]

[1] NITH, Norway
[2] Simula Research Laboratory, Norway
[3] Department of Informatics, University of Oslo, Norway

Abstract. "Cloud computing" is a popular way for application providers to obtain a flexible server and network infrastructure. Providers deploying applications with tight response time requirements such as games, are reluctant to use clouds. An important reason is the lack of real-time guarantees. This paper evaluates the actual, practical soft real-time CPU performance of current cloud services, with a special focus on online games. To perform this evaluation, we created a small benchmark and calibrated it to take a few milliseconds to run (often referred to as a *microbenchmrak*). Repeating this benchmark at a high frequency gives an overview of available resources over time. From the experimental results, we find that public cloud services deliver performance mostly within the requirements of popular online games, where Microsoft Azure Virtual machines give a significantly more stable performance than Amazon EC2.

1 Introduction

A game company planning to deploy an online game will traditionally have to face huge costs for data centre space and bandwidth capacity. Because it is hard to predict a game's popularity before it is launched, the game provider is prone to choose over-provisioning in order to meet potential high demands. Over-provisioning boosts the costs of deployment further. "Cloud computing" is a popular way for other application areas to address this problem. Here, the customer is billed based on the resources used, and can scale resources dynamically. Game companies have, however, been reluctant to place their game servers in a virtualized cloud environment. This reluctance is mainly due to the highly time-dependent nature of games. Many studies have reported that the quality of user experience is highly dependent on latency [3] [4]. For example, Chen et al. [3] examine network latency, but it is safe to assume that delays caused by processing will be perceived identically by players. They find that perceived quality of services in games depends strongly on response time, and variations in response time (jitter) from the server are more important than absolute values. The scale of the delays these papers describe as detrimental to the players lays

between 50 and 100 ms. Since providers must allow for normal network latencies in addition to processing time, any significant proportion of this delay incurred by overhead due to the cloud infrastructure should be noted.

The term "Cloud gaming", as currently used, represents "Software as a Service" *(SaaS)* [10], where the software in question is a game. The players will only run a simple thin client, sending player input to the servers, and receiving video and audio from the server. This is the approach followed by OnLive, Gaikai and others. Cloud gaming thus trades the need for local computing power for increased network throughput requirements as well as QoS demands. These services usually use a custom cloud infrastructure optimised for games.

Streaming the game in this way has some disadvantages. First of all, it has significant computing and bandwidth requirements. Many games require the full power of a modern computer to run, and the network traffic is equivalent to video streaming. Using a thin client which simply receives input and outputs video also precludes using any client-side latency-hiding techniques, so games run this way will be more latency sensitive. "Cloud gaming" in this sense is not the topic of this paper.

Conversely, this paper considers using general purpose public clouds to run traditional game servers, serving fully featured game clients. This configuration can be described as online games using cloud infrastructure. Time-dependent applications, like networked games, usually need to be custom-made, and can rarely be built upon typical web- or enterprise frameworks. Hence we need the kind of cloud service known as "Infrastructure as a Service" (IaaS) [10]. This allows the developers to access complete virtual machines, with full access privileges, allowing them to run their fully customised programs in the cloud.

Barker et al.(2010) [2] have investigated many aspects of latency sensitive applications on the Amazon EC2 cloud service. Investigating CPU, disk IO and network performance as well as running an actual game server, they conclude that the variable access to these resources, especially disk and CPU, are enough to degrade performance of latency sensitive applications.

In this paper build on [2] reexamining the CPU stability three years later and add data from one of the competitors that have entered the scene after Amazon: Microsoft Azure. We evaluate the applicability of public infrastructure clouds for use as infrastructure for online games and other highly time-sensitive applications. The work isolates CPU performance, leaving network and other subsystems for further work. We compare response time stability, with respect to processor availability, between cloud services and natively running software, as well as between two different cloud providers. The result is an updated recommendation for services providers.

Other than [2], work on performance in clouds have so far primarily focused on the typical large, bulk jobs of scientific computing. Schad et al. (2010) [9], Ostermann et al. (2010) [7] and [5] have all worked on long term tasks, most finding relevant variation between runs. For the large workloads investigated in this work, transient performance degradations at a millisecond scale, and even

stalls will be averaged out and difficult to detect. These issues are, however, highly relevant for game servers, as well as other soft realtime applications.

2 Experiment Design

To evaluate the virtual machines provided by different cloud services for their suitability in real-time applications, we created a benchmark and calibrated it to take a few milliseconds to run. The absolute average values of the benchmark runtime is not important, as we are only interested in the stability of the result. Neither does the load need to behave like any specific realistic load. This evaluation is only concerned with the actual availability of computing resources on a fine-grained time resolution.

With this in mind, the benchmark was designed as a small loop using array lookups and arithmetic, where the array is sized to be significantly larger than the cache size. This allows us to examine the memory subsystem and the CPU performance in isolation, ignoring external I/O. The tight loop of the benchmark is designed as a "worst case" load scenario for a virtual machine. Each machine was tested with a number of threads equal to the number of cores available to the virtual machine. Note that each instance of the benchmark is single-threaded, and will not show "better" results by increasing the number of cores. For these reasons, the most relevant single metric is the *coefficient of variation*. This value is defined as: $c_v = \frac{\sigma}{\mu}$, where σ is the standard deviation and μ is the mean.

In addition to the benchmark code, the utility "mpstat" was run in the background. Mpstat gives us "% steal", a metric that reports how much of the physical CPU is unavailable to the current OS instance because the hypervisor has assigned it to other instances.

Although the paper's main concern is to investigate the variance in execution time for the benchmark, a by-product is that we get an indication of the processing power provided by the different providers/service levels. This may also be of interest to game providers considering cloud deployment.

The benchmark was run on three different instances in Amazon EC2, in the zone "us-east-1d" (Table 1), as well as three different instances in Windows Azure (Table 2).

Micro Instance	613 MiB memory Up to 2 EC2 Compute Units (for short periodic bursts) I/O Performance: Low	USD 0.020 per Hour
Medium Instance	3.75 GiB memory 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit) I/O Performance: Moderate	USD 0.120 per Hour
High-CPU Medium Instance	1.7 GiB of memory 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each) I/O Performance: Moderate	USD 0.145 per Hour

Table 1. Technical specifications. from amazon [1].

Extra small instance	768 MiB memory 1 shared core	USD 0.020 per Hour
Small Instance	1.75 GiB memory 1 CPU Core	USD 0.085 per Hour
Medium Instance	3.5 GiB memory 2 CPU Cores	USD 0.160 per Hour

Table 2. Technical specifications from Microsoft [6]

2.1 Short-term runs

To compare the different instances, the benchmarks were run for 18 hours each under the configurations shown in tables 1 and 2. To create a baseline for comparing the cloud services with native dedicated hardware, the same benchmark was run a standalone PC (Intel Core 2 Duo CPU E7500 @ 2.93GHz, 4GB RAM) without any visualization, chosen to be approximately the same class of performance as the cloud computers.

2.2 Time series

Schad et al. [9] suggest that there might be differences in performance of the cloud systems based on when the benchmark is run, either by time of day or day of week. To investigate this, we ran the benchmark described above for three minutes every half hour for a week (the week of 27th of May, 2013). Between each run, we stopped the instance, and started it again before the next run. According to [9], this should allow us to get a different physical machine every time.

3 Evaluation

3.1 Reference system

The reference system (figure 1) shows the baseline of consistency we can expect from this benchmark while running natively. Deviations from this baseline can be assumed to result from the hypervisor or other aspects of the cloud system. As we can observe, the benchmark here reports extremely consistent results. The c_v is 0.02.

3.2 Amazon EC2

CPU Steal This metric is only sampled once a second, and so has a low resolution, but with sufficient samples we build up a picture of how much CPU is allocated to each instance, as well as the distribution over time.

The first thing to note in this data (figure 2) is the fact that all the instance types have a significant number of samples where the current instance has full use

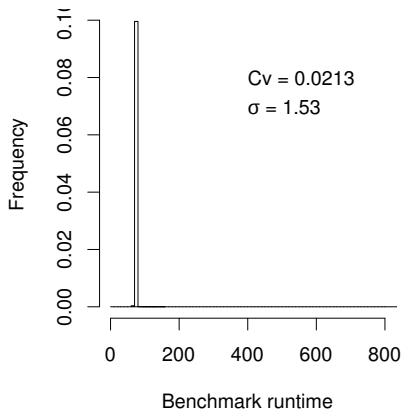


Fig. 1. Benchmark results from reference system.

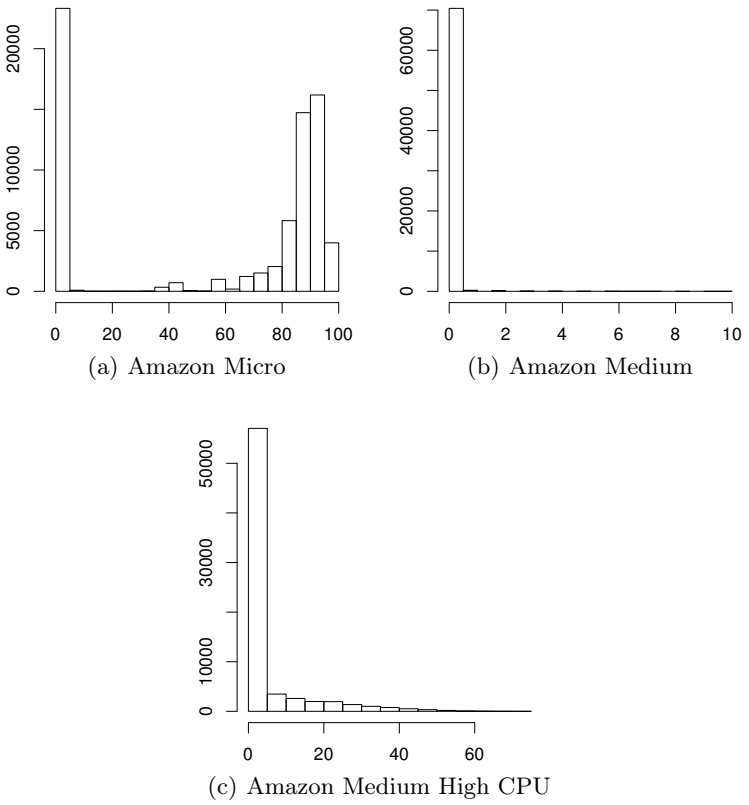
of the CPU. These situations will pull the average CPU availability significantly up, allowing the provider to fulfil their obligations despite other samples with much less available resources.

The next point standing out is the "medium" instance. According to the pricing information, the available CPU resources on this instance type equals exactly one core of the current hardware. This seems to create a situation where our instance gets full use of the core most of the time, with very little steal. Increasing the available CPU power to the "high cpu" instance adds another core, but since the instance on average only has access to part of this core, the cpu steal value increases again.

Benchmark runtime These results (figure 3) show several interesting values. First of all, for both the "micro" as well as the "medium, high-CPU" instance, there are two clear peaks. This matches the CPU-steal data, where the low values represent the situations where our instance got full use of the CPU, and the high run-times are the result of periods of higher CPU-steal. Again we see the advantage of the "medium" instance, in that it has a much more stable runtime, with c_v equal to 0.04. This more stable instance type depends on the fact that it allocates exactly one core to the instance. As the processing power granted each instance type is defined by an abstract, hardware independent metric, clients have no guarantee that this situation will continue indefinitely. Rather, when the underlying hardware is upgraded, it is very likely that each core will provide more power, and the currently stable instance type will become unstable.

3.3 Microsoft Azure

CPU Steal On the "Microsoft Azure Virtual Machine" cloud, the operating system always reports 0 CPU steal, as if it was running on its own dedicated machine. This implies that the hypervisor hides these details from the operating system, or that our instance actually has a dedicated CPU resource.



(a) Amazon Micro

(b) Amazon Medium

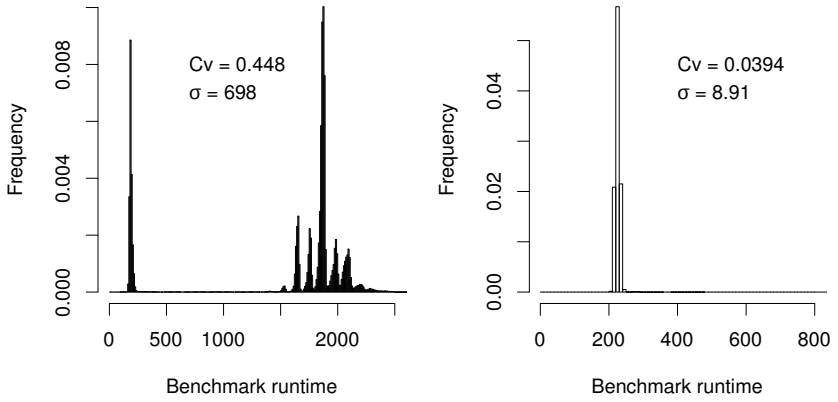
(c) Amazon Medium High CPU

Fig. 2. CPU Steal histograms. Samples every second.

Benchmark runtime Compared to the Amazon case, access to the CPU is significantly more stable in the Azure cloud (figure 4). Regardless of instance type, the runtime of the benchmark is almost completely predictable and stable, c_v is 0.05 for all instance types. The deviation is however twice that of the reference case. Depending on the exact level of time-sensitivity of the load, and its computational cost, this could be acceptable or not. The single-threaded nature of the benchmark explains why the ”small” and ”medium” show almost identical results.

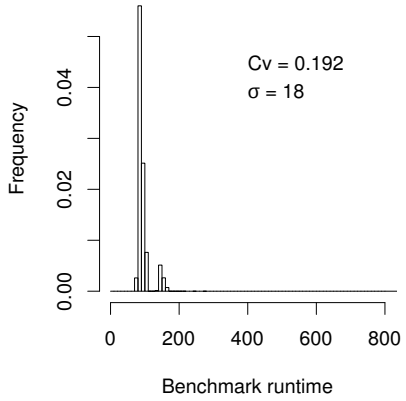
3.4 Time series

Figure 5 shows the results of running the benchmark repeatedly over a week on an EC2 Medium Highcpu instance. The two bands of performance are visible throughout the week, and there is no clear difference based on weekday or time of day.



(a) EC2 Micro

(b) EC2 Medium

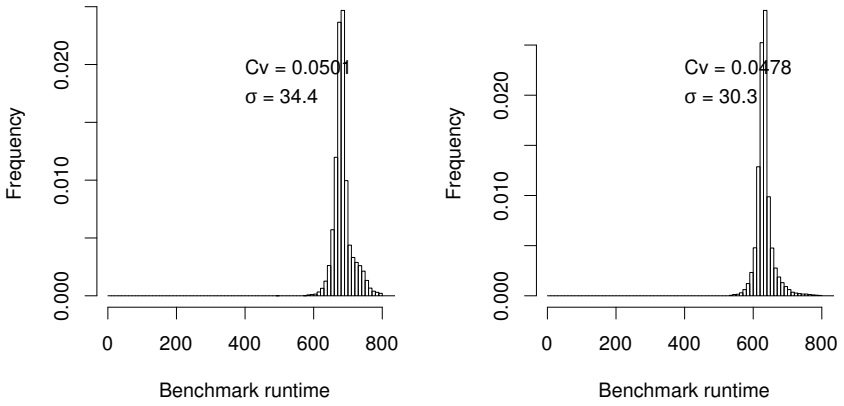


(c) EC2 Medium High CPU

Fig. 3. Amazon EC2: Histogram of benchmark runtimes. Note different scale on the axis of 3(a).

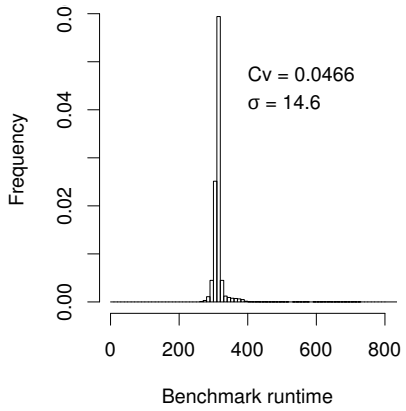
3.5 Overview

Putting it all together (figure 6) we see the quite clear differences between the providers. The low-end instance from Amazon has variations far above the acceptable level for any time-sensitive application. Interestingly the "Amazon EC2 Medium High CPU" is also quite unpredictable, though this configuration is sold as higher performing than the "Medium" type. Among the Amazon offerings examined only the "Medium" instance type is near acceptable. From Microsoft Azure, all instance types are reasonably stable in performance. All show variations above our native machine reference. The significance of these variations depend on the requirements of the application.



(a) Azure Extra Small

(b) Azure Small



(c) Azure Medium

Fig. 4. Microsoft Azure: Histograms of benchmark runtimes.

4 Conclusion

From these simple tests, it seems that the conclusions about CPU usage from [2] still hold for Amazon EC2. For the more time sensitive class of games [4], these delays can add up with network latency to unacceptable levels. Even for less sensitive applications there are some caveats. We do get rare samples where the CPU seems to have stalled for multiple seconds which could discourage users. Microsoft Azure Virtual machines seem to give more stable performance than Amazon EC2, without the extreme latency peaks. Utilizing these results can allow providers of interactive online services to be much more agile and react faster to changes in demand.

For future work on clouds and other virtual machines, we found that the c_v metric of multiple runs of a small, simple benchmark is a good starting point for measuring the consistency of computing resources.

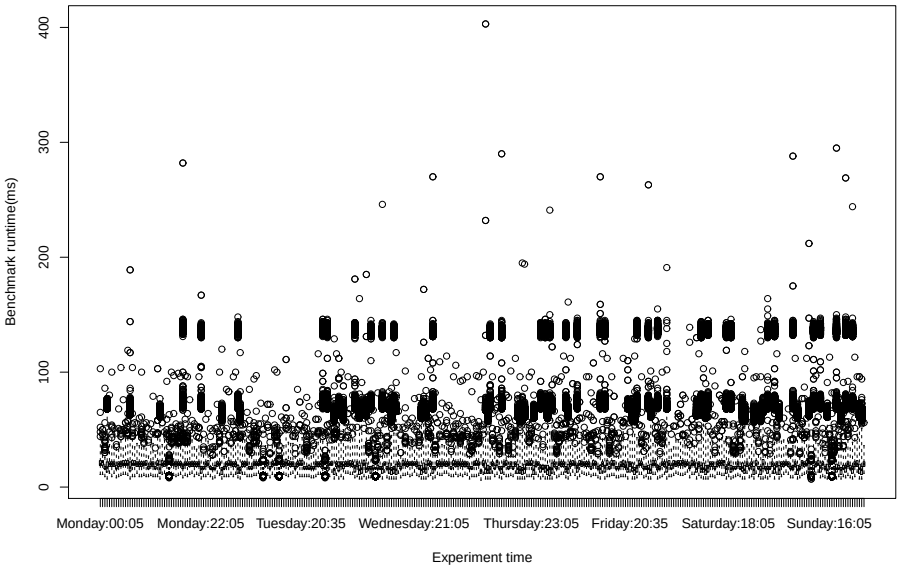


Fig. 5. Time series of Amazon EC2 Medium High CPU.

Based on our measurements, access to processors is sufficiently stable on today's IaaS cloud systems to allow for real-time services to be deployed, assuming the servers are run on the right instance types.

Amazon EC2 is better at the higher service levels, although variance at all levels is larger than for Azure. An important element to keep in mind for Amazon EC2 is that migration to a different instance is necessary if a change in service level is needed.

Azure cloud shows less variance in processor availability results than Amazon. Although you get less processor speed for the same price, it is more reliable for planning stability, reducing the need for a specialized service for managing changes between service levels. If a change in service level is needed, though, Azure allows for dynamic changes without the need to migrate the virtual machine.

In this paper we have evaluated Amazon EC2 and Microsoft Azure Cloud. To give wider recommendations we are planning similar experiments using Rackspace and Google Cloud. Google Cloud currently requires the customer to go through an application process for access. We did not get any reply from Google on our request. Rackspace have not been included due to time and space requirements.

To provide a complete view of the conditions for running real-time services on IaaS instances, we need to extend our result with networking measurements, extending [2].

In order to minimize the experienced processing time variance, monitoring the experienced processing time c_v and changing the service level based on the results should be explored. This will require different methods in Amazon EC2

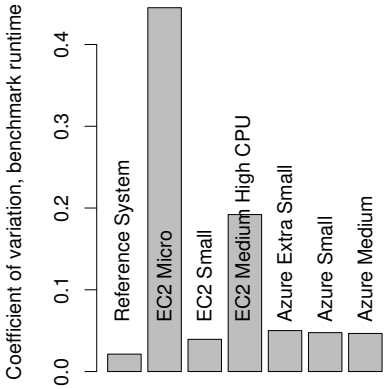


Fig. 6. Summary of benchmarks for the different systems.

and Azure, but should be a useful tool for game service providers that are concerned about processing jitter.

With more detailed knowledge of how virtual machines in the cloud respond, it would be relevant to experiment with more realistic, preferably highly parallel workloads such as LEARS [8].

References

1. Amazon. Amazon EC2 Instance Types. http://aws.amazon.com/ec2/instance-types/, 2013.
2. Barker, S. K., and Shenoy, P. Empirical evaluation of latency-sensitive application performance in the cloud. In *Proceedings of ACM SIGMM on Multimedia systems* (New York, NY, USA, 2010), MMSys '10, ACM, pp. 35–46.
3. Chen, K.-t., Huang, P., Wang, G.-s., Huang, C.-y., and Lei, C.-l. On the Sensitivity of Online Game Playing Time to Network QoS. *Proceedings of IEEE INFOCOM 2006 00*, c (2006).
4. Claypool, M., and Claypool, K. Latency Can Kill : Precision and Deadline in Online Games. *ACM Multimedia Systems Conference* (2010).
5. El-Khamra, Y., Kim, H., Jha, S., and Parashar, M. Exploring the Performance Fluctuations of HPC Workloads on Clouds. *2010 IEEE Cloud Computing Technology and Science* (Nov. 2010), 383–387.
6. Microsoft. Microsoft Azure Pricing Details. http://www.windowsazure.com/en-us/pricing/details/, 2013.
7. Ostermann, S., Iosup, A., and Yigitbasi, N. A performance analysis of EC2 cloud computing services for scientific computing. *Cloud Computing* (2010).
8. Raaen, K., and Espeland, H. LEARS: A Lockless, Relaxed-Atomicity State Model for Parallel Execution of a Game Server Partition. In *Parallel Processing ...* (Sept. 2012), Ieee, pp. 382–389.
9. Schad, J., Dittrich, J., and Quiané-Ruiz, J. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB ... 3*, 1 (2010).
10. Vaquero, L. M., Rodero-merino, L., Caceres, J., and Lindner, M. A Break in the Clouds : Towards a Cloud Definition. *Computer Communication Review 39*, 1 (2009), 50–55.