Migrating an MMO Back-end Infrastructure to the Cloud: a Perspective

Peer-reviewed author version

DOI: 10.1145/2577387.2577393
Handle: http://hdl.handle.net/1942/16570

Migrating an MMO Back-end Infrastructure to the Cloud : a Perspective

Peter Quax Wouter Vanmontfort Robin Marx Wim Lamotte
iMinds - tUL - UHasselt
Wetenschapspark 2
3590 Diepenbeek, Belgium
peter.quax@uhasselt.be

ABSTRACT

Many application providers are currently moving their back-end infrastructure to centralized data centers instead of hosting their own servers. Typically, the IaaS cloud model is most suitable for this type of application, as it provides customers with virtualized hardware that can be utilized as desired, not hindered by platform restrictions or platform-specific programming languages. A similar construct is not yet very prevalent in the MMO community, although some solutions are slowly appearing in the market. In this short description to go along with a poster presentation, some findings of ongoing research projects on these topics are highlighted, including a discussion of metrics that are relevant for successful MMO deployment and a preview of benchmark results obtained on a commercially available IaaS cloud infrastructure. Some of the most important pitfalls associated with the cloud revolution are also identified and discussed. More detailed information and benchmarks are made freely available on the project website mentioned at the end of this paper.

Categories and Subject Descriptors

[**Information Systems**]: [Information Systems Applications, Multimedia Information Systems]

General Terms

Design, Experimentation

Keywords

Massive Multiplayer On-Line Games, Scalability, Cloud

1. INTRODUCTION AND RELATED WORK

The so-called 'cloud revolution' is omni-present and nearly all major software companies are offering products based on remotely hosted servers. Many times, this comes in the form of PaaS offerings, where software is specifically developed to take the burden of scalability and redundancy of the underlying infrastructure away from application design. However, this type of cloud provisioning comes at a price and often entails the development (from scratch) of software specific to a cloud provider. Few game developers are adopting this model for commercial offerings. Using IaaS as a model more closely resembles the 'traditional' setup of a hosted data center, where a standard OS is run on a virtualization platform. The obvious benefit to the application developer/publisher is that existing solutions can easily be migrated to these platforms; however scalability needs to be taken care of in the application design. Popular offerings include Amazon's Elastic Compute Cloud (EC2), Google's Compute Engine and Rackspace. A dynamic server infrastructure like this seems – at first sight – specifically suitable for the deployment of MMOs that have an audience that is only active during certain times of day (e.g. for children).

This paper considers a specific subset of MMO games: those that rely on a number of small virtual worlds – composed of a limited set of regions – that are all instances (shards) of the same basic template. In contrast to vast and continuously growing spaces with user generated content (cf. Second Life), these smaller worlds allows for easier coupling of world instances to the virtualized (cloud) infrastructure. Also, inter-server communication is reduced to a minimum. This model is typically applied by smaller game developers/publishers because it requires lower investments in game content creation and is currently also commercialized (on a small scale) on cloud platforms - e.g. FarmVille servers run on Amazon EC2. Typically, software developers that create back-end solutions require a specific cloud provider to be utilized. This results in a loss of flexibility on the part of the game publisher as switching providers is no longer easily accomplished. Examples of these include HeroCloud and PhotonCloud.

2. THE OMEGA BACK-END ARCHITECTURE

A custom MMO back-end was designed with the abovementioned ideas in mind, referred to as the Omega back-end. It includes multiple server types that can be logically spread over multiple cloud server instances. As pictured in figure **??**, a master server is responsible for distributing clients over all running (game) server processes. Each game server is responsible for handling client updates associated with a certain region in a specific instance of the virtual world. For the initial design, Amazon's EC2 service was chosen as a reference model. Within each instance, a starter process is responsible for keeping track of the running game server processes within the instance. Note that the number of these processes may vary according to the actual load and the capacity of the cloud instance type chosen. Although the

MMVE'14 March 19-21 2014, Singapore, Singapore

ACM 978-1-4503-2708-4/14/03 ...$15.00.
http://dx.doi.org/10.1145/2577387.2577393.

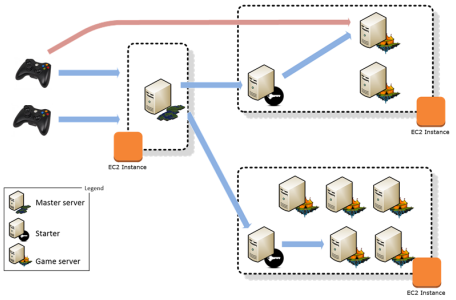

Figure 1: The Omega back-end architecture

initial connection setup for a client entails the establishment of a channel to a game server through the other server types, further data is exchanged directly.

Also specific to the Omega back-end is that it provides an abstraction layer for the APIs of the various underlying service platforms. As such, it is possible to quickly exchange the underlying cloud service provider based on varying conditions and payment models. It also provides for easy integration of timing routines, essential for benchmarking the solution on various setups (described in the next section).

3. BENCHMARKING

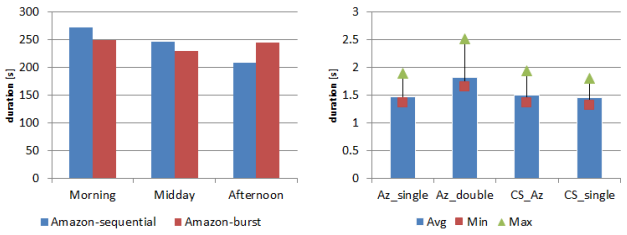

Figure 2: Benchmark results (left: startup duration; right: client migration time

Scalability is one of the main reasons behind the migration towards a cloud model. Therefore, a first important metric to benchmark is the time it takes for an additional (cloud) instance to become ready to support new users through its server processes. Relevant questions posed here are : "is there a drawback to launching multiple instances at the same time (burst)?" and "is the startup delay influenced by the time of day at which we demand more instances ?". It turns out from benchmarking (which took place over several days and runs) that the first question can be answered negatively, as the variation in startup times between tests with sequential and burst-style instance startup scheduling turned out to be insignificant. The assumption in the second question can be confirmed, as some afternoon tests returned shorter startup durations versus morning conditions (figure **??** left). However, this result should not be interpreted in absolute terms, but is rather meant as an indication that performance is likely to vary over time. Overall, startup times (including provisioning and actually booting the instance until ready) are between 3 and 4 minutes.

A second important metric to consider is the time it takes for a client to migrate between game servers (i.e. logging out, finding the new server to connect to and logging in again, no explicit state transfer). This is essential for efficient cloud-based hosting. Tests here included a comparison of migration times on EC2 versus Apache CloudStack (a private and open source cloud solution) and an investigation into differences between data centers of the same provider. Although the timings (including Unity3D scene loading) were higher for EC2 when compared to CloudStack, there turned out to be no significant difference between the private and public solutions (figure **??** right) - taking into account the inevitable additional network delay. Overall, these handover times are in the order of 1-2 seconds, which is acceptable given the type of game supported (using multiple regions and instances). A game developer can easily hide these through clever level and gameplay design (e.g. perform handovers when the user is teleporting or moving through a dark tunnel, as is often done in commercial MMOGs).

A third metric is associated with the instance type and cost model of the cloud provider. To establish whether a more expensive instance also translates into increased performance, real-life benchmarks were run on some of the various EC2 instance types. It turns out that there is no 1:1 mapping of the cost of the instance type to an in/decrease in CPU performance. A trend for more capacity with higher-priced instance types is clear however. Regarding I/O performance, there turns out to be little difference between some instance types; however it is very likely that the performance over time is more stable in the more expensive offerings. In terms of network performance, internal network capacity is increased dramatically for more expensive offerings, but the same is not true for external connectivity, with a significant decrease between the cheapest offerings, but not for the more expensive propositions. External connectivity metrics (delay and packet loss) remain relatively stable over time.

4. PITFALLS

Some conclusions can be drawn on possible pitfalls of moving the infrastructure to a cloud provider. First, this type of migration does not entail a cost reduction per se, but rather depends on the game designer including a mechanism to shut down instances as soon as they are no longer required. In case instances are left running idle, they cost at least as much as a dedicated (hosted) solution and all benefits are lost. This also illustrates why the timings associated with client migration are so important : without a means to efficiently concentrate users on the smallest number of servers possible, costs will easily grow out of control.

Secondly, there are some practical limitations in current cloud provider offerings that increase the complexity. Virtual machines often cannot be suspended (for later resuming) without incurring penalties in terms of storage cost. Therefore, the most economical solution is to completely shut down (terminate) instances and reboot (create) them when needed, cf the first metric.

While cloud providers typically consider CPU power to be the most relevant commodity, bandwidth should not be underestimated as a factor in the total cost. Especially if inter-instance communication is required, it is beneficial to consider a provider that offers this communication for free or at a low cost. By using an the abstraction of cloud vendor specific APIs provided by the Omega architecture, switching between providers based on their (changing) pricing models becomes an option.

5. ACKNOWLEDGMENTS

These results are part of the IWT PIM Omega and iMinds MIX Wanagogo projects (http://www.omega-project.be).