Live VM Migration

Use-Cases for Cloud hosting providers

Tim Felgentreff Tobias Pape

9th March 2011

CONTENTS

1 INTRODUCTION

Today, many organizations deploy a multitude of services on a large number of servers. In some cases, all these services run on different machines not because one machine cannot handle the load, but for reasons of reliability. All software is faulty [Zelo6], and operating systems with thousands of lines of code, running servers written by different software vendors simply cannot be trusted to provide uptimes of 99.9 percent or more. Putting each service on a separate machine will achieve reliability at the expense of maintainability simply because a large number of physical machines is involved.

Virtual machine technology has been around for more than 40 years [TT92]. The technology to host multiple machines on a single system reduces the space and power requirements for data centers and represents enormous cost savings for companies like Amazon, Microsoft or Google, which deploy hundreds or thousands of servers for a variety of services. Unsurprisingly, virtualization for cost reduction and transparent scalability is becoming increasingly popular. Cloud providers such as Amazon, Rackspace or Engine Yard offer scalable hosting solutions for companies and individuals who cannot of do not want to maintain a private cloud.

There are obvious issues with isolation, however. As a service provider, I want to avoid downtime as much as possible. Thus, I cannot easily move my virtual machine from one provider to another. I cannot easily migrate existing VMs to new a hypervisor architecture or a completely new virtualization solu-

tion. If hardware is failing, I cannot simply migrate the VM to new hardware without taking it down.

Asger Jensen and Dr. Jacob Gorm Hansen proposed live VM migration as a solution to these problems. They were the first to demonstrate live migration of a running VM in 2002 on a research hypervisor called NomadBIOS. Since then, HP has equipped it's HP-UX with live migration capabilities, VMWare has implemented it in its vMotion product, the open-source Xen project supports it and IBM is actively researching to enable VM migration on a number of open-source virtualization solutions.

2 STATE OF THE ART

In this part we will explore the state of the art in live VM migration, both from a technical as from a use-case point of view.

2.1 VMWare Live Migration

Dr. Jacob Gorm Hansen described the process of live migration as implemented in his NomadBIOS, which we will recapitulate here.

2.2 Live Migration Use Cases

The following sections summarize some of the real-world use cases for live migration and how it compares to the previous solutions which were applied to these problems.

2.2.1 *Reliability of Virtualization*

Running a large number of virtual machines (VMs) on very few physical nodes seems like betting all your money on one horse only. If one physical node fails a number of services will be unavailable all at once, probably much more than if a dedicated machine would crash. However, most server failures in data centers are due to buggy software, especially operating systems and servers, not hardware failure. Virtualization isolates such failures, because VMs run in an isolated environment atop a *hypervisor*.

A hypervisor controls access to the physical hardware, provides routing from the physical network into the virtual network and isolates machines from one another. If any one machine should fail, all others will be unaffected. Because all of the operations within a VM are virtualized, the hypervisor can checkpoint virtual servers at regular intervals. Should a VM fail, it can be restored from an earlier checkpoint and resume operations.

Critical services can be run in complete isolation to minimize the potential for failure due to faulty or incompatible software stacks. The hypervisor itself

is the only software running in kernel mode on the physical machine and has orders of magnitude fewer lines of code than the average operating system. This means it is equally less likely to encounter a bug in it's programming.

Still, hardware failures, while less likely than software bugs, are an issue that has to be considered. Before live migration of VMs was possible, an alarm state from a hardware monitor could be answered by migrating the relevant processes to new VMs[HJ04]. For a given VM, another one similar to it would be booted on a another machine. The state of the services could be copied and new service requests would be routed to the fresh VM, with the old VM kept online to serve low-level calls and dependencies of in-process requests, before shutting down[CFH+05].

2.2.2 *Compatibility of Virtualization Solutions*

Several process migration systems were developed in the 1980s, examples are Sprite and MOSIX[HJ04]. Such migration solutions depend on capabilities of the hypervisor as well as the operating system, limiting their use across organization boundaries, during system upgrades, as well as for cloud hosting solutions where the OS software is not controlled by the hosting provider.

Virtualized machine hardware varies even today. There are efforts underway[clo11] to standardize different aspects of virtualization, like the format for virtual machine descriptions and the APIs of hypervisors. This is a work-in-progress, however, and moving complete systems between different vendors usually involves shutting those systems down to allow various data conversions to take place.

2.2.3 *Hardware Upgrades without Downtime*

Modern businesses have to be able to react to market change quickly and with minimal cost. New services are first explored on cheap hardware[TT92], but sudden bursts of customer interest might require fast upscaling of the underlying hardware. Being able to quickly scale up to rising interest is crucial for the success of a service. Studies show that slow loading times will cause people to use online services less or leave them and never return[KL07]. Historically, the quick success of such a scenario dependent on quick and correct assertion of upcoming hardware requirements and carefully planned downtime.

Using live migration, services can stay online and better hardware can be booted in parallel to the existing. A new

2.2.4 *Moving between Clouds*

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like »Huardest gefburn«. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

2.2.5 *Uptime Despite Hardware Failure*

Hello, here is some text without a meaning. This text should show, how a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like »Huardest gefburn«. Kjift – Never mind! A blind text like this gives you information about the selected font, how the letters are written and the impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for a special contents, but the length of words should match to the language.

2.2.6 *Scaling through Replication*

```
select * from foo where true;
```

REFERENCES

[CFH+05] 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-Volume 2*, pages 273–286. USENIX Association, 2005.

[clo11] Cloud Standards, 2011.

[HJ04] J.G. Hansen and E. Jul. Self-migration of operating systems. In *Proceedings of the 11th workshop on ACM SIGOPS European workshop*, page 23. ACM, 2004.

[KL07] R. Kohavi and R. Longbotham. Online Experiments: Lessons Learned. *Computer*, 40(9):103–105, 2007.

[TT92] A.S. Tanenbaum and A. Tannenbaum. *Modern operating systems*, volume 271. Prentice Hall Englewood Cliffs, NJ, 1992.

[Zel06] A. Zeller. *Why programs fail*. Elsevier/Morgan Kaufmann, 2006.