Live VM Migration
Principles and Performance

Petter Svärd

LICENTIATE THESIS, MAY 2012
DEPARTMENT OF COMPUTING SCIENCE
UMEÅ UNIVERSITY
SWEDEN

Department of Computing Science
Umeå University
SE-901 87 Umeå, Sweden

petters@cs.umu.se

ISBN 978-91-7459-452-2
ISSN 0348-0542
UMINF 12.12

Printed by Print & Media, Umeå University, 2012

Abstract

Virtualization is a key technology for cloud computing as it allows several operating system instances to run on the same machine, enhances resource manageability and enables flexible definition of billing units. Virtualization works by adding a software layer, a hypervisor, on top of the hardware platform. Virtual Machines, *VMs*, are run on top of the hypervisor, which provisions hardwares resources to the VM guests. In addition to enabling higher utilization of hardware resources, the ability to move VMs from one host to another is an important feature.

Live migration is the concept of migrating a VM while it is running and responding to requests. Since VMs can be re-located while running, live migration allows for better hardware utilization. This is because placement of services can be performed dynamically and not only when the are started. Live migration is also a useful tool for administrative purposes. If a server needs to be taken off-line for maintenance reasons, it can be cleared of services by live migrating these to other hosts.

This thesis investigates the principles behind live migration. The common live migration approaches in use today are evaluated and common objectives are presented as well as challenges that have to be overcome in order to implement an ideal live migration algorithm. The performance of common live migration approaches is also evaluated and it is found that even though live migration is supported by most hypervisors, it has drawbacks which makes the technique hard to use in certain situations. Migrating CPU and/or memory intensive VMs or migrating VMs over low-bandwidth links is a problem regardless of which approach that is used. To tackle this problem, two improvements to live migration are proposed and evaluated, delta compression and dynamic page transfer reordering. Both improvements demonstrate better performance than the standard algorithm when migrating CPU and/or memory intensive VMs and migrating over low bandwidth links. Finally, recommendations are made on which live migration approach to use depending on the scenario and also what improvements to the standard live migration algorithms should be used and when.

Preface

This thesis consists of a brief introduction to the field, a short discussion of the main problems studied, and the following papers.

Paper I	P. Svärd, B. Hudzia, J. Tordsson and E. Elmroth Evaluation of Delta Compression techniques for Efficient Live Migration of large Virtual Machines, In *The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 111-120. ACM 2011
Paper II	P. Svärd, J. Tordsson, B. Hudzia, and E. Elmroth High Performance Live Migration through Dynamic Page Transfer Reordering and Compression, In *Third IEEE International Conference on Cloud Computing Technology and Science 2011*, pages 542-548. IEEE 2011
Paper III	P. Svärd, J. Tordsson and E. Elmroth The Noble art of Live Migration, *Technical Report, UMINF-12.11, May 2012 (Submitted)*, 2012.

Financial support has been provided by the European Commission's Seventh Framework Programme ([FP7/2001-2013]) under grant agreements no. 215605 (RESERVOIR) and 257115 (OPTIMIS) as well as by UMIT research lab and the Swedish Government's strategic research project eSSENCE.

Acknowledgments

First things first, I thank my supervisor Erik Elmroth for his support and help, especially in arranging the collaboration with SAP Research that has been very valuable for this work. I also thank my co-supervisor Johan Tordsson for valuable support and interesting discussions concerning live migration, snowboarding and other matters of importance. At SAP Research, the help and support of Benoit Hudzia, Aidan Shribman and Stuart Hacking has been invaluable concerning design, implementation and debugging issues with the software prototypes developed within the scope of this thesis.

Regarding my colleagues in the Cloud and Grid computing group I wish to thank all of you but there are a few that I want to give a special mention. I thank Tomas Ögren for help with various computer-related issues, Daniel Espling for good discussions and for supporting the one true ice hockey team, Luleå Hockey, P-O Östberg for helping me get settled as a fresh PhD student and last but not least, Wubin 'Viali' Li for managing my aversion against svn and eclipse, for his support and for putting up with my snoring during our frequent travels.

Finally, I thank my family and friends. Without you, this thesis would not have been possible.

Contents

1 Introduction **1**

2 Virtualization **3**
- 2.1 Types of Virtualization 3
- 2.2 Applications of Virtualization 4

3 Live Migration principles **5**
- 3.1 Approaches to Live Migration 6
  - 3.1.1 Pre-copy migration 6
  - 3.1.2 Post-copy migration 7
- 3.2 WAN migration 7
- 3.3 Live Migration performance 7

4 Thesis Contributions **9**
- 4.1 Paper I 9
- 4.2 Paper II 9
- 4.3 Paper III 10

Paper I **17**

Paper II **33**

Paper III **45**

Chapter 1

Introduction

The ability to move, or *migrate*, a Virtual Machine (*VM*) from one physical host to another is an important aspect of virtualization. If the migration is done in such a way that the connected clients do not perceive any service interruption, this is known as *live migration*. Live migration is useful in many scenarios, for example, server consolidation is made easier if VMs do not have to be shut down before they are moved. The technique is also used for administrative purposes, for example, live migrating VMs to other hosts if a server needs to be taken off-line for some reason, and it can be utilized to transfer running VMs between cloud sites over WAN networks.

The focus of this thesis is on principles and performance of live migration. The underlying concepts of live migration are investigated and a number of improvements are suggested. These improvements focus on reducing *migration downtime*, during which service is interrupted, and also reducing the amount of transmitted data as well as the total migration time. Finally, the thesis discusses different approaches to live migration and the suitability of these in a number of scenarios. Conclusions are drawn based on experimental evaluations.

The rest of this thesis is structured as follows. Section 2 provides an introduction to virtualization and outlines a few of the requirements on virtualized infrastructures and environments. Section 3 discusses common approaches to live migration. Some applications of live migration are also discussed. Finally, Section 4 summarizes the contributions of this thesis, and relates the thesis papers to each other.

Chapter 2

Virtualization

Virtualization is the concept of hosting several operating systems on the same physical hardware by running them on top of a software layer known as a *hypervisor* [3] that simulates a hardware platform. The hypervisor handles provisioning and sharing of the common hardware resources between the operating system instances, which does not normally need to be virtualization aware. The physical machine is referred to as the *host* and the virtualized operating system instances running on top of the hypervisor are known as *guests*, or more commonly, virtual machines.

Virtualization is a mature technology and has been around since the 1960s [6] but its use was not widespread until the addition of *virtualization extensions* to mainstream CPUs in 2005 [10]. This hardware-accelerated virtualization allows for a dramatic increase in VM performance and since its introduction, the use of virtualization has increased dramatically.

2.1 Types of Virtualization

Virtualization can be implemented in several ways. A common approach is *full virtualization* where a complete hardware platform is simulated. The VMs run on top of the virtualized hardware and need not be modified in any way, in fact, they need not be aware that they are running in a virtualized environment. Examples of hypervisors providing full virtualization are KVM [8], VMWare [18] and Microsoft Virtual PC [1].

If the simulated platform is not identical to the underlying hardware this is known as *paravirtualization*. Because the interface presented to the guest operating system differs slightly from the hardware, the operating system needs to be modified before it can run on the virtualized platform. An example of a paravirtualization hypervisor is XEN [3].

2.2 Applications of Virtualization

The ability to run several operating system instances on the same hardware is important since it enables easier administration, better usage of resources and reduced power consumption [17]. Virtualization is an enabling technology for cloud environments [2] since it provides the ability to quickly provision pre-configured VM instances. Virtualization technologies also enables scalability and centralization of administrative functionality in cloud environments. Figure 1 shows an example of a virtualized infrastructure. Three servers running a hypervisor platform are used as hosts for the VMs and a shared storage solution is used to store the VMs disk images.

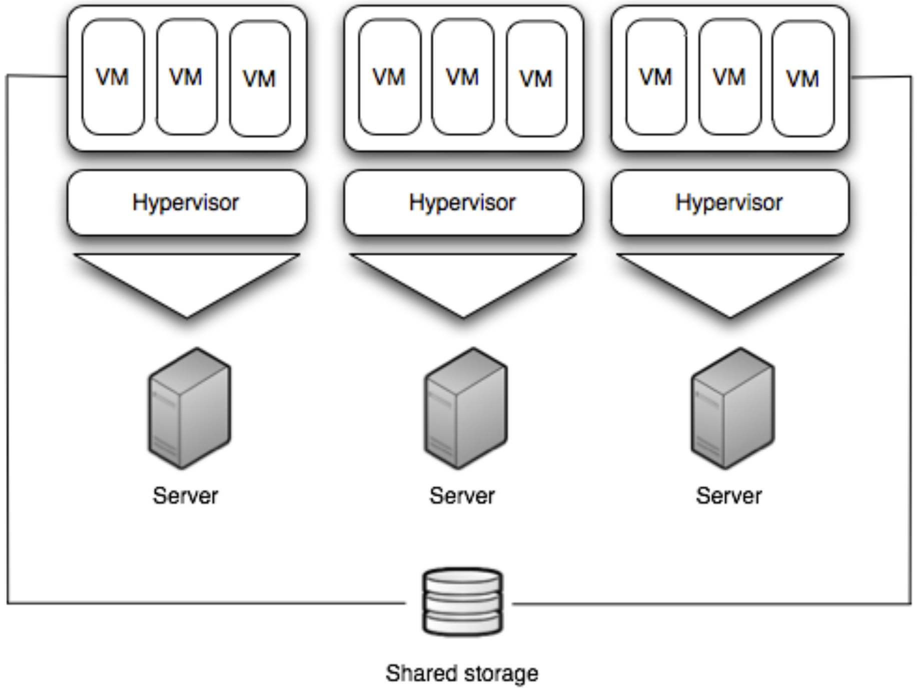

Figure 1: Example of a virtualized infrastructure.

Chapter 3

Live Migration principles

One advantage of virtualization is that VMs can be migrated from one physical host to another. In order to do this, the VM's state, which consists of its RAM contents and any disk images associated with it must be transferred. To migrate a VM, it is first suspended, whereupon the VM's memory contents are written to a file. This file, the VM's description file, and its disk images are then transfered to the new host where the VM's execution is resumed. The files can be transferred either over the network or by using some kind of storage medium. This form of VM migration is known as *cold migration*.

If instead the VM is kept running during the transfer of its state, the migration can be performed with no perceivable interruption in service for any connected peers. Such forms of VM migration are known as *live migration*. Live migration was first demonstrated by Clark et al. [5], and is supported by most current hypervisors. Notably, even when using live migration the VM has to be suspended for a short while, otherwise it would not be possible to transfer all of the state since the VMs memory and disk are constantly written to during execution.

Normally, live migration is performed within the same site and the disk images are kept on a network storage device accessible to both the source and the destination hosts, which means that the images need not be migrated. While the VM is suspended, service is of course interrupted so to achieve the goal of no perceivable service interruption, the migration downtime should be minimal. Cross-site migration can also be performed, as seen in Figure 2, the VM's disk image must normally also be transferred in this case.

Most current hypervisors, such as Xen [3], KVM [8], and VMWare [18] support live migration with downtimes ranging from tens of milliseconds to a second when migrating normal workloads over LANs.

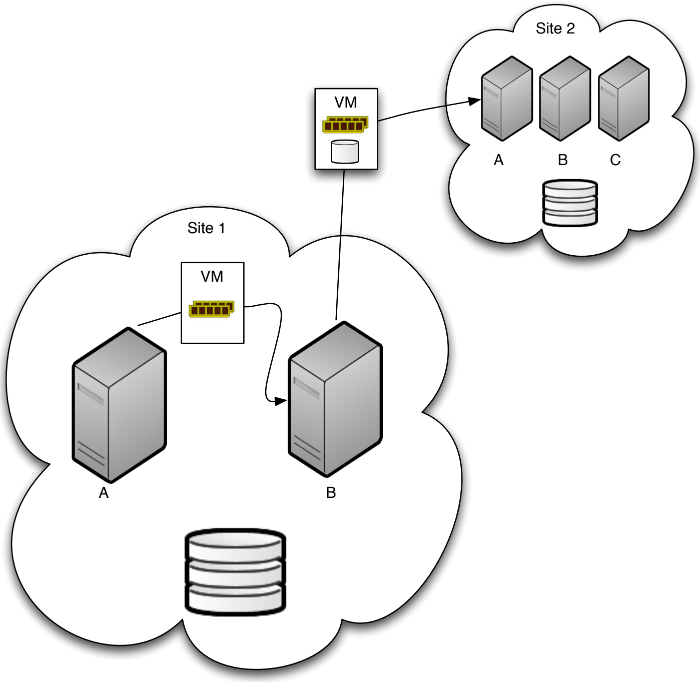

Figure 2: In-site and cross-site migration.

3.1 Approaches to Live Migration

There are two main approaches to live migration in use today. Although both approaches are designed for service migration with continuous operation and minimal disruption, they differ in implementation. The conceptual difference between them is at what point during the live migration process the execution switches from the source to the destination VM.

3.1.1 Pre-copy migration

With *pre-copy migration* the state is transferred in the background in a series of iterations while the source VM is running and responding to requests. When enough of the state has been transferred, the source VM is suspended to stop memory writes. When to switch is typically decided based on the amount of memory remaining to transfer, or after a maximum amount of iterations is reached. After the VM has been suspended, the remaining state is transferred and the VM is finally resumed on the destination host. As seen in Figure 3, the time it takes to complete the whole migration process is known as the *total migration time*.

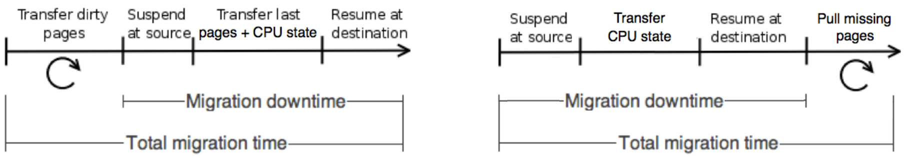

Figure 3: Pre-copy (left) and post-copy (right) migration.

3.1.2 Post-copy migration

Using *post-copy migration*, the execution is switched to the destination almost immediately after the start of live migration, as can be seen in Figure 3. Only the CPU state is transferred before the switch. The remaining state is then pulled from the source over the network, either on demand, pre-emptively or a combination of the two.

3.2 WAN migration

Live migrating a VM over Wide Area Network, *WAN*, poses a number of problems. Firstly, since WAN migration is performed between different sites it is not feasible to use a network storage device for the disk images. This means that a *storage migration* [20] must also be performed. Storage migration works in a similar way to RAM migration, the storage blocks are transferred over the network from the source to the destination while the VM is running. Retransfer is performed for any block modified by the VM during this process. Also, since the VM is migrated to another site, IP address information differs so network address translation has to be performed to route client requests to the VM correctly. In the LAN case, an Address Resolution Protocol, *ARP* [11] message is used to notify peers of the hosts new location. However, in WAN migration scenarios, the VM might be migrated to another IP address space and this solution is not applicable. Finally, the usability of the post-copy approach over WAN is limited due to the long delays in transferring state over low bandwidth links.

Usually, hypervisors assume that migration is performed over a Local Area Network (LAN), however live WAN migration has been demonstrated in several contributions [4, 7, 12, 16, 19]. In these cases, WAN migration is achieved by use of techniques based on, e.g., IP tunneling, Mobile IP, and light path reservation.

3.3 Live Migration performance

Regardless of which approach to live migration that is used, the main objective is no service interruption perceived by users of the migrated VM. Apart from this *continuous operation* objective, live migration performance can be measured in a number of ways. Obvious metrics are the total migration time and the migration downtime, shorter times implying a more efficient algorithm. However, the resource consumption of an algorithm is also important since live migration aims to be transparent. An algorithm that consumes excessive resources risks affecting the performance of the migrating VM as well as any co-located VM in a negative manner. Also, network bandwidth is valuable and should not be wasted if it can be avoided. In this section, we discuss a couple of ways to tackle the challenges of resource consumption, extended migration downtime and performance degradation and thus improve live migration performance.

When migrating VMs running CPU and/or memory intensive workloads, the migration downtime can easily reach several seconds or more [9, 13, 15]. In such cases, there is a high risk of service interruption due to missed database timers, dropped network connections or other issues. To reduce the migration downtime, the memory pages can be compressed before transfer. Because less data is transferred in each iteration, the live migration process has a higher probability of keeping up with the VM writing to its memory. This means that the amount of data remaining to transfer when the source VM is suspended can be reduced, thereby shortening the migration downtime. In Paper I, a delta compression algorithm that is aimed at shortening migration downtime is proposed and evaluated. The algorithm works by reducing the amount of transferred data during the iterative phase of pre-copy live migration. Using compression to improve live migration performance has also been proposed by Wood et al. [19]. Their strategy to increase live migration performance between cloud sites involves data deduplication techniques in addition to compression.

During live migration, the complete contents of the VMs RAM is transferred from the source to the destination host. Since RAM sizes of several gigabytes are not uncommon and pre-copy live migration re-sends many RAM pages several times, live migration often involves transferring large data volumes. This leads to long total migration times. Even though compression techniques reduce the amount of transferred data, they do not necessarily reduce the number of page re-sends. To tackle this problem, Paper II proposes and evaluates a page reordering technique that reduces the amount of transmitted data by sending the memory pages in reverse order of usage frequency to avoid re-transfers. In a related contribution, Zheng et al. [20] leverage block update frequency when performing storage migration. Their idea is similar to the dynamic page transfer reordering scheme [15] but for storage migration instead of memory migration.

Since post-copy migration only transfer pages once, algorithms based on this approach do not have the same issue with excessive resource usage as pre-copy approaches. However, as they pull missing pages from the source over the network, performance can suffer after the execution has moved from the source to the destination. Paper III [14] investigates this problem and evaluates a hybrid live migration algorithm that addresses this issue.

Chapter 4

Thesis Contributions

This section summarizes the contributions of the papers that make up this thesis.

4.1 Paper I

Paper I [13] investigates the use of delta compression techniques to tackle the problem of memory pages being dirtied faster than they can be transferred over the network. By compressing the data stream during the transfer, migration throughput is increased, reducing the risk of the dirtying rate being higher than migration throughput. This means that migration downtime can be reduced. The delta compression extension studied in the paper is a implemented as a modification to the standard KVM pre-copy algorithm. The algorithm's performance is evaluated by migrating VMs running both real-world and benchmark applications and the evaluation demonstrates a significant decrease in migration downtime in all of the test cases. Paper I also discusses some general effects of delta compression on live migration and contains an analysis of when it is beneficial to use the delta compression technique.

4.2 Paper II

Paper II [15] introduces a live migration algorithm where the pages are transferred in an non-sequential order based on the page dirtying frequency. The algorithm, called *Dynamic Page Transfer Reordering*, is designed to reduce the number of page re-sends during migration thereby reducing the total amount of data being sent which in turn leads to a shorter migration time. The page dirtying frequency is kept track of by adding a priority bitmap on top of the standard KVM dirty page bitmap. As pages are being dirtied, they move down in priority. Since the algorithm transfers pages are by order of priority the frequently dirtied pages will not be transferred until the end of the migration process, reducing the risk of having to re-send them. The algorithm also includes the delta compression techniques from Paper I [13]. To evaluate the performance of the algorithm a streaming video server as well as a benchmark application is live migrated. In the evaluation, both migration downtime and total migration time was reduced.

4.3 Paper III

Paper III [14] is a broad study of live migration. The paper defines objectives for live migration and discusses common challenges in meeting these. Three approaches to live migration, pre-copy, post-copy and hybrid live migration are investigated and evaluated by a set of experiments to highlight their characteristics. The paper contains a discussion on the pros and cons of the approaches in different scenarios and give pointers to which approach is best suited in different cases. Common improvements to live migration are also discussed and a flowchart provides help in selecting the appropriate live migration improvement for a given scenario. Finally, the future research landscape in the area is outlined.

Bibliography

[1] Microsoft Virtual PC. `http://www.microsoft.com/windows/virtual-pc/default.aspx`, visited on 2012-05-23.

[2] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM, 2003.

[4] Robert Bradford, Evangelos Kotsovinos, Anja Feldmann, and Harald Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *VEE '07: Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*, pages 169–179. ACM, 2007.

[5] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI '05: 2nd Symposium on Networked Systems Design and Implementation*, pages 273–286. ACM, 2005.

[6] Robert Jay Creasy. The origin of the VM/370 time-sharing system. *IBM Journal of Research & Development*, 25:483–490, 1981.

[7] Eric Harney, Sebastien Goasguen, Jim Martin, Mike Murphy, and Mike Westall. The efficacy of live virtual machine migrations over the internet. In *VTDC '07: 2nd International Workshop on Virtualization Technologies in Distributed Computing*, pages 1–7. ACM, 2007.

[8] Kernel Based Virtual Machine. KVM - kernel-based virtualization machine white paper. `http://kvm.qumranet.com/kvmwiki`, visited on 2012-05-23.

[9] Pengcheng Liu, Ziye Yang, Xiang Song, Yixun Zhou, Haibo Chen, and Binyu Zang. Heterogeneous live migration of virtual machines. Technical report, Parallel Processing Institute, Fudan University, 2009.

[10] Gil Neiger, Amy Santoni, Felix Leung, Dion Rodgers, and Rich Uhlig. Intel virtualization technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 10:167–178, 2006.

[11] David C Plummer. Rfc 826: An ethernet address resolution protocol – or – converting network protocol addresses to 48.bit ethernet address for transmission on ethernet hardware. 1982.

[12] K. K. Ramakrishnan, Prashant Shenoy, and Jacobus Van der Merwe. Live data center migration across WANs: a robust cooperative context aware approach. In *INM '07: The ACM SIGCOMM Workshop on Internet Network Management 2007*, pages 262–267. ACM.

[13] Petter Svärd, Benoit Hudzia, Johan Tordsson, and Erik Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. In *VEE '11: The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 111–120. ACM, 2011.

[14] Petter Svärd, Johan Tordsson, and Erik Elmroth. The Noble Art of Live VM Migration. Technical report, 2012. Tech Report UMINF 12.11. Submitted.

[15] Petter Svärd, Johan Tordsson, Benoit Hudzia, and Erik Elmroth. High performance live migration through dynamic page transfer reordering and compression. In *CloudCom '11: 3rd IEEE International Conference on Cloud Computing Technology and Science*, pages 542–548. IEEE, 2011.

[16] Franco Travostino, Paul Daspit, Leon Gommans, Chetan Jog, Cees de Laat, Joe Mambretti, Inder Monga, Bas van Oudenaarde, Satish Raghunatha, and Phil Yonghui Wang. Seamless live migration of virtual machines over the MAN/WAN. In *SC '06: The international Conference on for High Performance Computing, Networking, Storage and Analysis*, page 290. ACM, 2006.

[17] Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. Server workload analysis for power minimization using consolidation. In *USENIX '09: The 2009 USENIX Annual Technical Conference '09*, pages 28–28. USENIX, 2009.

[18] VMWARE. VMware VMotion: Live migration of virtual machines without service interruption datasheet. `http://www.vmware.com/files/pdf/VMware-VMotion-DS-EN.pdf`, visited on 2012-05-23.

[19] Timothy Wood, K. K. Ramakrishnan, Prashant Shenoy, and Jacobus van der Merwe. CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In *VEE '11: The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 121–132. ACM, 2011.

[20] Jie Zheng, Tze Sing Eugene Ng, and Kunwadee Sripanidkulchai. Workload-aware live storage migration for clouds. In *VEE '11: The 2011 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pages 133–144. ACM, 2011.