# Support for Virtual Machine Malleability in Cloud Computing Using Migratable Components

Ping Wang, Carlos A. Varela
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street, Troy NY 12180-3590, USA

October 29, 2010

## 1 Introduction

Cloud computing is an emerging distributed computing paradigm that promises to offer cost-effective scalable on-demand services to users, without the need for large up-front infrastructure investments [1]. A key enabling technology for cloud computing is virtualization. With virtualization, service providers can ensure isolation of multiple user workloads, provision resources in a cost-effective manner by consolidating virtual machines (VMs) onto fewer physical resources when system load is low, and quickly scale up workloads to more physical resources in peak periods. Cloud computing users also benefit from VM abstraction in that they can quickly deploy VM images with preloaded operating systems, software applications, and application data onto cloud environments without being concerned with lower-level physical infrastructure details.

As cloud computing brings significant benefits to service providers and users, it also introduces new challenges. Its pay-per-use business model gives service users such flexibility in purchasing a service that they could try a service with lower cost and turn away from a service whenever they like, which results in fluctuating user workloads. Meanwhile, the available physical resources for one VM image are non-dedicated and dynamic, because service providers usually multiplex multiple VMs onto one single physical machine (PM) and would turn on/off PMs from time to time for the sake of maintenance or energy savings. In such a computing environment, VMs are likely to be either overprovisioned, incurring unnecessary cost, or underprovisioned, which may degrade the quality of service. Thus, in order for the cloud computing paradigm to fulfill its promises, it is necessary to investigate adaptive VM management techniques that enable the VMs to achieve optimal quality of service in spite of fluctuating user workloads and available physical resources.

While existing works on adaptive VM management mainly focus on VM migration [2, 3, 4], VM malleability, which is the ability to change the gran-

ularity of the VM instances dynamically, has not yet been fully explored. In our previous work [5], it has been demonstrated that VM granularity affects the performance of tightly coupled computational workloads to a large extent, and it is not trivial to identify the optimal VM granularity even for a fixed workload and static physical computing environment. Thus, we realize that it is desirable to equip a cloud computing system with VM malleability.

In this project, we propose a model in which VM malleability can be easily realized under the assumption of malleable workloads. This main ingredients of this model are the transparent component migration supported by the SALSA programming language and a VM malleability middleware. To evaluate the performance of our model, we plan to conduct experiments on two sample applications.

This proposal continues as follows. Section II describes the proposed model from the aspects of software framework, workflow and experimental evaluation. Section III presents related work. Section IV concludes this paper.

# 2 Project Description

## 2.1 Software Framework

The software framework of our VM malleability model as shown in Figure 1 is introduced as below.

### 2.1.1 The SALSA Programming Language

SALSA is an actor-base language that simplifies dynamical reconfiguration for mobile and Internet computing through features like universal names, active objects and migration [6]. Applications composed of SALSA actors can be easily reconfigured even at run time through actor migration.

### 2.1.2 VM Malleability Middleware

VM Malleability middleware contains the following two modules.

*Mediator:* Mediator collects information about applications, e.g. the number of active actors, and the references of these actors. It requests VM Manager to split/merge VMs when VMs are ready to be reconfigured. It also coordinates the migration of the actors of the application to proper destinations after VMs have been split/merged.

*VM Manager:* VM Manager is a module that maintains information about the VMs in the system, e.g. the VM quota per PM, the number of active VM per PM, the CPU usage of each VM. It processes requests from Mediator and perform VM split/merge via invoking the underlying VM hypervisor.

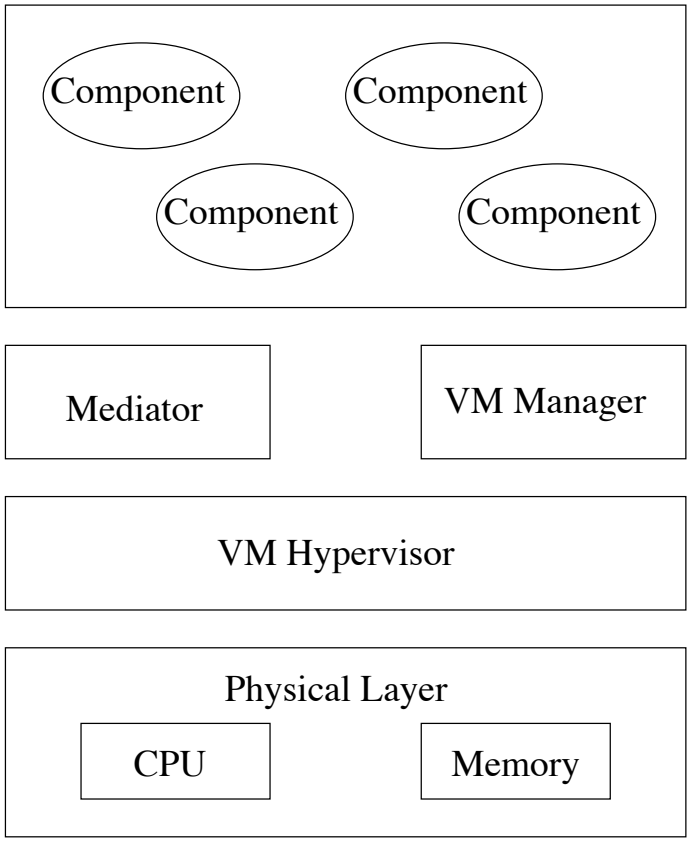


Figure 1: Software framework of our VM malleability model

## 2.2 Workflow

Our VM malleability model is composed of two parts: split and merge. First, one VM can be split into multiple VMs and newly created VMs can be migrated onto other PMs if needed. On the other direction, multiple VMs, which might be located on different PMs, can be merged into one VM on demand. We describe the steps of the splitting and merging operations as below.

*1) Split Operation:* In the split operation, one source VM is split into N partitions, which are restored to N target VMs on N target hosts. The steps that a split operation takes are summarized below.

- Step 1 Ready: Each application actor sends a message to Mediator when it enters the ready status. Thus, when all the actors of the applications are ready, Mediator knows that the source VM is ready. When one application actor becomes ready can be specified according to the application specific needs.

- Step 2 VM Split: Mediator sends a split request to VM Manager, including the ID of the source VM, and the number of VMs to be split. VM Manager checks if there are enough VM quota and distribute the resources (e.g. CPU and Memory ) of the source VM to the new VMs. Lastly, VM Manager invokes hypervisor commands to create the new VMs.

- Step 3 Transfer: Upon receiving the confirmation of the split request from VM Manager, Mediator coordinates the migration of the actors on the source VM to the new VMs.

- Step 6 Clean up: After all the actors on the source VM are migrated, they run on the new VMs transparently. Mediator then sends a clean up request to VM Manager and the latter removes the source VM.

*2) Merge Algorithm:* In the merge operation, N source VMs, which might be located on different hosts, are merged into one target VM. The steps that a merge operation involves are summarized below.

- Step 1 Ready: Each application actor sends a message to Mediator when it enters the ready status. Mediator checks if each source VM enters the ready status and starts the merge process till all the source VMs are ready.

- Step 2 VM Merge: Mediator sends a VM merge request to VM Manager, including the ID of the source VMs. VM Manager assigns the resources (e.g. CPU and Memory ) of the source VMs to the new VMs. Lastly, VM Manager invokes hypervisor command to create the new VM.

- Step 3 Transfer: Upon receiving the confirmation of the merge request from VM Manager, Mediator coordinates the migration of the actors on the source VMs to the new VM.

- Step 6 Clean up: When all the actors on the source VM run on the new VMs transparently, Mediator sends a clean up request to VM Manager and the latter removes the source VMs.

### 2.3 Experimental Evaluation

We choose two sample SALSA applications to evaluate our VM malleability prototype. The first application is a fluid- dynamics application which models heat transfer in a solid. It is a tightly coupled iterative computational application. The second application is a network intensive application: a distributed web crawler.

## 3 Related Work

In this section, we discuss the existing research related to our work in two fields: process/component malleability, and virtual machine migration.

**A. Process/Component Malleability** [7] investigates the impact of process granularity on system-level performance and finds out that while small process granularity generates unnecessary context-switching overhead and increases inter-process communication, large granularity of each process may also yield unsatisfactory performance because the data of the processes do not fit in the upper level of the memory-hierarchy or the processes are too few to be distributed onto the available resources in a balanced way. To enable a parallel application's execution system to run with the right process granularity that utilizes the resources most effectively, process malleability is introduced and implemented as an extension to the process check pointing and migration (PCM) library, a user-level library for iterative MPI applications.

[8] reaches similar observations concerning component granularity and realizes component malleability in the SALSA programming language [6] with SALSA actor being reconfigurable component.

Compared to both process and component malleability, VM malleability has the following advantages. First, while process and component malleability require collaboration from application programmers, VM malleability is transparent to the application layer. Secondly, it is non-trivial to realize malleable processes or components across different programming languages, operating systems and hardware architectures. By contrast, VM malleability is platform independent via the virtualization layer.

**B. Virtual Machine Migration** VM migration has been explored as an adaptive VM management technique that dynamically changes the mapping between VM instances and physical resources. [4] proposes the pre-copy approach, which iteratively copies the memory of a VM from the source to destination host before releasing the VM at the source and resuming it at the destination. [9] describes the post-copy migration that defers the transfer of a VM's memory until its processor state has been sent to the target host. To realize VM migration across wide area network, [10] designs a system that pre-copies local persistent state to the destination while the VM still runs on the source host. Meanwhile, a user-level block device is used to record and forward the write accesses to the destination to ensure consistency. The system also contains a network redirection scheme that redirects the existing connections of the migrated VM using IP tunneling and advertises its new IP address with Dynamic DNS. Recently, there also has been significant work on live migration which supports continuous device access [11, 12].

While VM migration provides VM-based cloud environments with capabilities such as flexible physical resource reallocation, system workload consolidation, and on-line maintenance, the effectiveness and scalability of VM migration solutions are limited by the granularity of the VM instances.

# 4 Conclusion

This proposal introduces the VM malleability model based on the transparent component migration supported by the SALSA programming language and a

VM malleability middleware. We sketch the software framework and workflow of the proposed model, and plan to conduct experiments on two sample applications to evaluate the performance of our model.

# References


[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[2] J. G. Hansen and E. Jul, "Self-migration of operating systems," in *Proceedings of the 11th ACM SIGOPS European workshop*, Leuven, Belgium, September 2004.

[3] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, December 2002, pp. 377–390.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI'05: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.

[5] P. Wang, W. Huang, and C. A. Varela, "Impact of virtual machine granularity on cloud computing workloads performance," in *Workshop on Autonomic Computational Science (ACS'2010)*, Brussels, Belgium, October 2010.

[6] C. Varela and G. Agha, "Programming dynamically reconfigurable open systems with salsa," *OOPSLA '01: SIGPLAN Notices. ACM Object Oriented Programming Languages, Systems and Applications Intriguing Technology Track Proceedings*, vol. 36, no. 12, pp. 20–34, December 2001.

[7] E. M. Kaoutar, D. Travis, K. S. Boleslaw, and A. V. Carlos, "Malleable iterative MPI applications," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 3, pp. 393–413, 2009.

[8] D. Travis, E. M. Kaoutar, and A. V. Carlos, "Malleable components for scalable high performance computing," in *Proceedings of the HPDC'15 Workshop on HPC Grid programming Environments and Components (HPC-GECO/CompFrame)*. Paris, France: IEEE Computer Society, June 2006, pp. 37–44, best paper award.

[9] M. R. Hines and K. Gopalan, "Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning," in *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. New York, NY, USA: ACM, 2009, pp. 51–60.

[10] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiberg, "Live wide-area migration of virtual machines including local persistent state," in *VEE '07: Proceedings of the 3rd International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments*. New York, NY, USA: ACM Press, June 2007, pp. 169–179.

[11] S. Kumar and K. Schwan, "Netchannel: a vmm-level mechanism for continuous, transparentdevice access during vm migration," in *VEE '08: Proceedings of the 4th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. New York, NY, USA: ACM, 2008, pp. 31–40.

[12] A. Kadav and M. M. Swift, "Live migration of direct-access devices," *SIGOPS Operating Systems Review*, vol. 43, no. 3, pp. 95–104, 2009.