Secured and Reliable VM Migration in Personal Cloud

Wei Wang[1], Ya Zhang[1], Ben Lin[2], Xiaoxin Wu[1], Kai Miao[2]
[1]Intel Labs China
[2]Intel Information Technology
Intel Corporation
Beijing, PR China
{vince.wang, ya.zhang, ben.y.lin, xiaoxin.wu, kai.miao}@intel.com

***Abstract*—The security and reliability issues that reside in VM live migration is a critical factor for its acceptance by IT industry. We propose to leverage Intel vPro and TPM to improve security in virtual machine live migration. A role-based mechanism with remote attestation is introduced, under which the VM migration is controlled by specific policies that are protected in seal storage. To provide high reliability in client virtualization, we also implement a checkpoint based method that only consumes 5% overall system resource according to the experimental result.**

Keywords- Live migration; Security, Reliability

I. INTRODUCTION

Virtualization embraces multiple technologies at differing stages of maturity, which poses new questions about the optimal client computing architecture. In our work, we extend Cloud usage to personal environment, because today an individual may have multiple computing or communication devices. For example, a person may have a cellular phone or a Mobile Internet Device (MID) that he always carries with him. He probably also has a laptop or a desktop that has a stronger CPU/GPU set, a larger MEM/disk, a friendly input interface, and a larger display. This stronger device may probably be left somewhere (e.g., office or home) because of inconvenience for portability. Once the owner carries a handheld and approaches to the stronger devices, e.g., when he is set at the office or at home, he can jointly use smart phone/MID and laptop/desktop through different network connections to form a personal Cloud. A user can also migrate a work from laptop to a MID or phone when he is going to a meeting room.

Our vision for future personal-centralized Cloud environment where a personal handheld device, e.g., MID, is probably the center of personal life [5], is shown in Figure 1. There is a public Cloud that is managed by Enterprise IT center. A Cloud user can access such Cloud through corporate network or VPN. There is also a personal Cloud, which are built up by a personal handheld device and its surrounded computing or costumer electronic (CE) devices. The inter-connection for the components in a personal Cloud may be, for example through near field communication (NFC) technology, under which the underlying networks can be direct cable connection or wireless networks (e.g., WLAN, WPAN, Bluetooth, etc.).

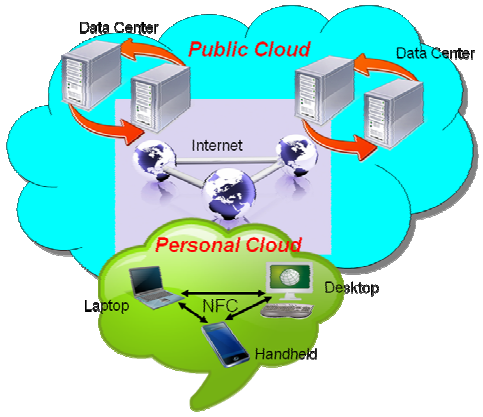

Figure 1. A future Cloud Usage in Enterprise Environment

Live migration is a useful feature and natural extension to virtualization technology that allows for the transfer of a virtual machine from one physical machine to another with little or no downtime for the services hosted by the virtual machines. The migration functionality implemented by vendors such as XEN [1], and KVM (kernel-based virtual machine) [2] now exposes the entire machine state of a VM to device module which listens to the incoming live migration requests from remote platforms. An attack can easily hijack the device module process or hypervisor where these migrations occur. If the process is hijacked, the information of the migrated virtual machine including states of operation system kernel, applications and services running within the operating system, the sensitive data currently being used by those applications and even the inputs from keyboard are accessible to the hackers.

In personal Cloud, because Cloud members may probably join Cloud at an ad hoc manner, it is challenging for any device in a personal Cloud to directly judge or enforce the security for its Cloud peers. However, based on the architecture shown in Figure 1, if each of the members in personal Cloud is connected to a back-end Cloud, the back-end Cloud can act as a trusted third party to force the migration security.

This paper presents security solutions for VM live migration among un-trusted/open platforms in personal clouds. In particular, this work proposes methodologies for 1) checking/verifying the security level of a migration destination and 2) defining migration policy. The solution is

978-1-4244-6350-3/10/$26.00 ©2010 IEEE

based on a trusted computing base (TCB) [9] that should only include a small part of software and hardware. Security depends on these software and hardware that distinguish from a much larger amount that can misbehave without affecting security. Because any vulnerability in the TCB can potentially be exploited by an attacker to initiate serious attacks, in this work, we make the TCB small enough, which only includes vPro hardware and the secure hypervisor. To serve for enterprise scenario, the secure hypervisor is designed in a way so that it is able to be updated form IT center during secure migration process.

Another problem resides in migration is the reliability. There will be a problem when the connection between two devices is shutdown unexpectedly. This may occur when people have an emergent requirement. Without doing a regular migration process, the target machine will not have the same status as the original platform. Currently this can lead the Guest OS to restart in target to rebuild all the environments which make it unavailable for some time and the loss the work context eventually.

We here propose a solution inspired by hypervisor based fault tolerance technology (HBFT), which is an emerging approach to sustain mission-critical applications. A VM is an ideal building block to implement checkpoint/restart algorithms. Without intervention in applications or operating systems, our solution synchronizes state between two virtual machines at frequency. Memory as well as CPU state of the Guest OS in Laptop/PC is saved to a snapshot file stored in MID from time to time at background. Meantime, the disk image is set to read-only at start time and the VMM write all dirty blocks to it when a memory snapshot is ready.

The paper is organized as follows. In section 2 we discuss the secured live migration. In section 3 we present a checkpoint based solution for VM migration reliability. Finally in section 4 we conclude and list future works.

II. SECURED LIVE MIGRATION

In this security framework we propose a Policy-controlled secure live migration that is based on Intel vPro hardware platform for virtual machine migration protection. Figure 2 shows the high level architecture of this proposal.

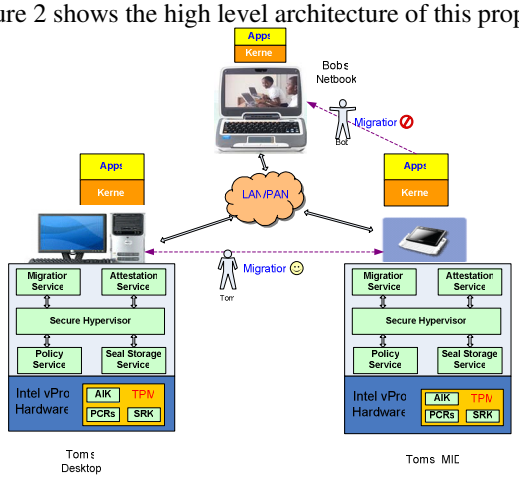

Figure 2. Architecture of Secured Live Migration

This design depends on the following features that hardware may support.

- All platforms support sealed storage and remote attestation.
- Intel vPro technology is enabled.

Based on required hardware technologies, personal Cloud members are enforced to behave correctly, e.g., to tell the truth to each other about security level through integrity check from the back-end Cloud, and to install secure application environment as required. The security framework for secure migration within a personal Cloud can then be built upon these features, which consists of following modules:

Attestation Service: This module allows a running hypervisor to cryptographically identity itself to a remote hypervisor, that is, to tell the remote hypervisor what is running inside the secure box. This allows the remote hypervisor trust the application, i.e. to be confident that any application will behave as required.

Seal Storage: This module encrypts data using the private key of the tamper resistant TPM that is responsible for attestation. A hash of the booted trusted OS is also included with the encrypted data. The TPM only allows a trusted OS with the same hash to unseal it. This functionality is used by the secure hypervisor to persistently store its private key and role-based policies.

Policy Service: This module parses and manages the role-based policies provisioned with the VM image for virtual machine migration decisions, such as who has the right to migrate a virtual machine, and to which hosts this virtual machine can be migrated.

Migration Service: This module is responsible for migration. One of its duties is initiating attestation requests to remote machines to check whether the target machines meet the security requirement, and if not, it will require them to do software upgrade or patching.

Secure Hypervisor: This module protects the process of guest OS by Runtime Memory Measurement [7].

The proposal relies on the key technologies, e,g, Intel vPro and TPM, to enhance the security level for virtual machine migration. They provide the following abilities:

- The ability that makes the virtual machine work in an open environment in a secure way.
- The ability to encrypt and store keys, data or other secrets within hardware on the platform, which makes sure that these secrets can only be released (decrypted) to an executing environment that has the same level of security as when the secrets were encrypted
- The ability of remote attestation to ensure that the trustworthy environment was correctly invoked.

These features help the scheme to secure Virtual Machine during migrations between open platforms.

Considering a typical usage scenario, a user, e.g., Tom, can migrate VMs between his devices, e.g., MID and laptop, if both of the devices have the security framework built in, and both devices reach the security level that Tom has set for

running a VM. When Tom wants to migrate a VM to Bob's device, which either does not have such security supportive technology or cannot reach the required security level, the migration cannot be processed.

Secure migration includes three key steps, which are building trustworthy container for virtual machine, securing VM Migration, and securing hypervisor.

A. *Building trustworthy container for virtual machine*

To build a trustworthy VM container, remote attestation has to be processed to first check whether the destination VM container meets the security requirements. If the security requirements are not met, the back-end Cloud will guide it to build a trustworthy container with the help of hardware support. The back-end Cloud can ensure that all steps for container installation such as OS booting, hypervisor selection/installation, and software patching/upgrading required by the source host of the migrated VM, are correct. Once the trusted container is installed and passes the integrity check, the attestation completes. The detailed flowchart for attestation among Cloud peers is shown in Figure 3.

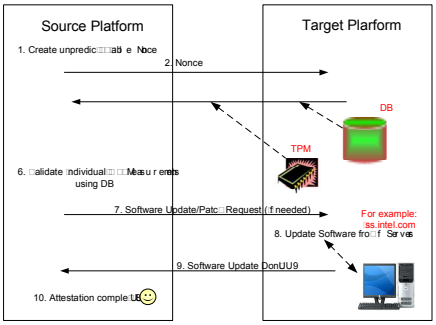

Figure 3. Build Trustworthy Container for Virtual Machine

B. *Securing VM Migration*

After building the trustworthy virtual machine container, one session for the virtual machine migration will be started as shown in Figure 4.

The detail description for this role-based live migration is shown below. A VM will be either migrated to a host, or migrate out from the host.

Migrate-out (flowchart with green lines): The owner of a VM initiates one outgoing request to the migration service module (Step 4). This service checks whether this move is allowed by checking the policy service module, which makes migration permission according to pre-deployed policies for this virtual machine. After the migration service gets the "Allow" permission from the policy service module, it gets key and certificate from the seal storage module to encrypt the entire state of the virtual machine, and then migrates the virtual machine to the targeted platform.

Migrate-in (flowchart with red lines): The owner of VM initiates one incoming migration request to the migration service module. At the mean time, policies regarding to this VM are loaded. After validating the policies, the policy service module stores it to his local environment in seal storage.

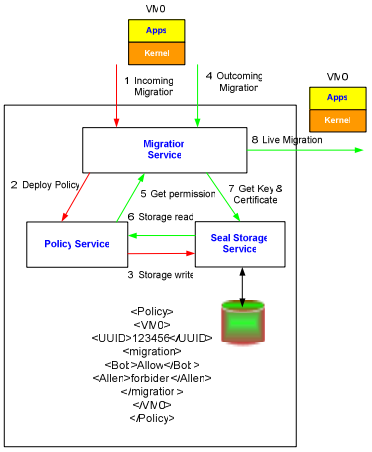

Figure 4. Role-based Live Migration

C. *Securing hypervisor*

Finally, the overall secure framework uses a secured hypervisor design that provides the protection on key applications in a Guest VM. We adopt the work in [6] and [7] to provide runtime memory protection. In this proposal, we utilize hardware techniques to provide trust services to software programs. Without modifying OS, we leverage Intel vPro technology to create a lightweight hypervisor for fine-grain software runtime memory protection. As a result, a program's memory could be hidden from other high-privilege system software in a single commodity OS. The detailed design of secure hypervisor and its functions are shown in Figure 5.

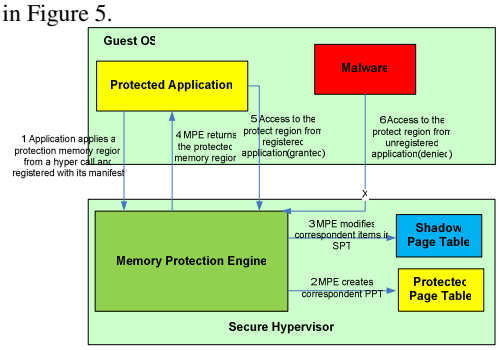

Figure 5. Secure Hypervisor

III. RELIABLE LIVE MIGRATION

It is possible that after a VM has migrated from MID to laptop, the VM is suddenly terminated because of reasons such as the network connection between the MID and laptop is cut off. When this occurs, it is desired that the VM be recovered at the MID continue the applications.

The reliability method (shown in Figure 6) we adopt is to keep a backup for a running VM. The running host, in our scenario the laptop, keeps on sending updated memory status of the VM to the backup site, e.g., MID. If, for some reasons, the VM on the laptop is no more active, the MID can take over immediately from wherever the memory state it obtains points to.

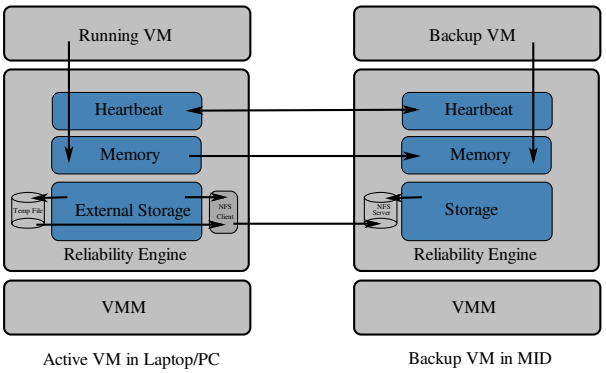

Figure 6. Overview of Checkpoint Solution

To simplify the problem, we assume the Guest OS is running in only one device (MID or Laptop/PC) as multiple VM instances will result in different user working environment and need more effort on synchronization. What's more, it will result in duplicate user data which lowers the storage efficient in MID.

The memory state file sent from laptop to MID is called a snapshot, normally generated by VMM. A design issue is how frequently the snapshots should be sent. Generating a snapshot of a VM needs to suspend all the processes running within it. If snapshots are generated too frequently, the applications have to be interrupted too often, which leads to significant application performance degradation. In addition, more network traffic has to be delivered, which is another factor that affects application performance.

In this work we set the time interval between two consecutive snapshots a large value, e.g., 90s to reduce reliability overhead. The downside of generating snapshots at such a low frequency is it leads to missing the most updated memory state. Therefore, at the backup site the application can only be recovered to states that are out-of-date. However, such an out-of-date problem then is not very serious, because most of real-time applications are stateless and won't care about the loss of previous process states.

A remained design issue is about consistency between memory and disk state in the checkpoint solution. In our design a shared storage on MID is used as the VM disk and it is only modified when a checkpoint is completed. This ensures that the memory and disk are written at the same time stamp, for VM to be resumed correctly. When VMM is doing checkpoint, it creates a new temporary file to track all disk updates generated by Guest OS. After the checkpointing finishes, the temporary file will be merged with the original disk in MID.

The process diagram for snapshot generation, disk update, and VM recovery upon a desktop failure is illustrated in Figure 7.

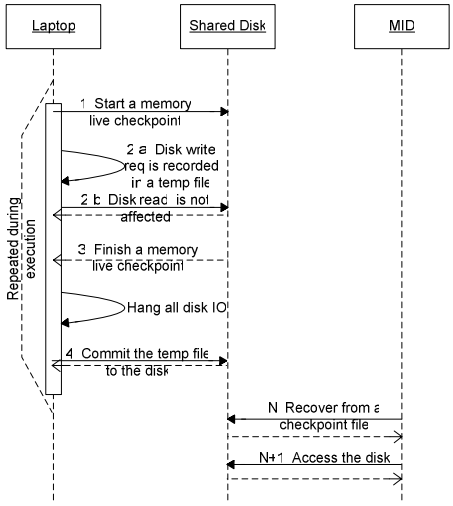

Figure 7. Process of Checkpointing Mechanism

To measure the performance impact brought by live checkpointing process, we run 2 industrial benchmark workloads as Super-Pi and HWinfo32 2.39-238. The test environment is show in Table I:

TABLE I. EVALUATION PLATFORM

CPU	Core2 Duo T7700 2.40GHz
Memory	2G
Host OS	Ubuntu 9.10 32bit
NIC	Intel 82566MM Gigabit NIC
Guest OS	Windows XP SP3
Guest CPU number	1
Guest memory	512M

From the testing, we get three kinds of performance data: no checkpointing enabled for the baseline, checkpointing in raw data format and checkpointing in gzip compressed data format. Taking 1M Pi computing for instance, the raw checkpointing mechanism takes a slightly longer than base. And in gzip checkpointing, the computation is bit more than raw checkpointing. The overall overhead of the checkpointing mechanism is below 5% without data compression. The detailed evaluation result is shown in Table II.

Different from HBFT (hypervisor-based fault tolerance) which implements checkpointing by repeated executions of the final phrase of live migration at a high frequency of tens of milliseconds, we address the problem by modifying current live migration mechanism to make it generate checkpoint continuously. The checkpoint is produced not so frequently as in server side solutions for less overhead. For instance, Maohua Lu [11] reports that HBFT experiences 200% slow down for some realistic data center workloads.

Even with asynchronous state transfer optimization, Remus [12] leads to 103% performance degradation during kernel compilation, compared to native VM performance. Kemari [13] also reports a similar performance penalty. Propagating state synchronously at every change is impractical: it effectively reduces the overall performance. According to the evaluation result, our solution has much lower overhead and is hence more efficient and more suitable for client side virtualization.

TABLE II. BENCHMARK RESULT

	Base	Raw Checkpointing		GZip Checkpointing	
	result	result	overhead	result	overhead
SuperPi[1M]	21.109	21.968	4.069%	23.429	10.991 %
HWInfo32[Benchmark_CPU]	30930	30038	2.884%	29748	3.822%
HWInfo32[Benchmark_FPU]	19186	18318	4.524%	18876	1.616%
HWInfo32[Benchmark_MMX]	39349	37869	3.761%	37788	3.967%
HWInfo32[Benchmark_Memory]	2336	2225	4.752%	1949	16.567 %

The recovery time is the stall time in MID after it detects the failure. From the data shown in Table III, we can conclude that compressed Checkpointing takes much longer (nearly 3 seconds) to recover though it can save about 75% storage space.

TABLE III. RECOVERY TIME AND FILE SIZE

Checkpointing	Recovery Time	Size
Raw	918.122ms	55.298MB
GZip	2799.523ms	217.339MB

IV. CONCLUSIONS AND FUTURE WORKS

In this work we present a migration security framework built upon Intel hardware technologies. In particular, this work proposes methodologies for 1) checking/verifying the security level of a migration destination and 2) securing hypervisor and VMs running over it.

We also considered about the reliability during VM migration, a checkpoint based solution is invented for it and the overhead for background checkpointing is acceptable.

REFERENCES

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, R. N. Alex Ho, I. Pratt, and A. Warfield, “Xen and the Art of Virtualization”, in proceedings of ACM Symposium on Operating Systems Principles, 2003.

[2] KVM Forum, http : //www.linux-kvm.org

[3] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny, Eric July, Christian Limpach, Ian Pratt, Andrew Warfield, “Live Migration of Virtual Machines”, in NSDI 2005.

[4] John Dunlop, “Developing an Enterprise Client Virtualization Strategy”, in Intel IT White Paper, 2008

[5] X Wu, W Wang, B Lin, K Miao, “Composable IO: A Novel Resource Sharing Platform in Personal Clouds”, in CloudCom 2009

[6] Brannock Kirk, Dewan Prashant, McKeen Frank, Savagaonkar Uday, “Providing a safe execution environment”, Intel Technology Journal, Jun 2009, Vol. 13 Issue 2, p36-51

[7] Dewan, P., Durham, D., Khosravi, H., Long, M., and Nagabhushan, G. 2008. “A hypervisor-based system for protecting software runtime memory and persistent storage”. In Proceedings of the 2008 Spring Simulation Multiconference (Ottawa, Canada, April 14 - 17, 2008).

[8] Ravi Sahita et al. “Towards a Virtualization-based Framework for Information Traceability”, Advances in Information Security—Insider Attack and Cyber Security, ISBN 978-0-387-77321-6.

[9] B. Lampson, M. Abadi, M. Burrows and E. Wobber. “Authentication in Distributed Systems: Theory and Practice”, ACM Transactions on Computer Systems, page 6, 1992.

[10] X. Wu, W. Wang, B. Lin, M. Kai, ” Composable IO: A Novel Resource Sharing Platform in Personal Clouds”, in Proc.of CoudCom, 2009.

[11] L. Maohua and C. Tzi-cker. Fast memory state synchronization for virtualization-based fault tolerance. In Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'09), Estoril, Lisbon, 2009.

[12] C. Brendan, L. Geoffrey, M. Dutch, F. Mike, H. Norm, and W. Andrew. Remus: High availability via asynchronous virtual machine replication. In Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), Berkeley, CA, USA, 2008.

[13] T. Yoshiaki, S. Koji, K. Seiji, and M. Satoshi. Kemari: Virtual Machine Synchronization for Fault Tolerance. 2008.

[14] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. In Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP'95), New York, USA, 1995.