PolyHype: Towards Poly-Hypervisor Platform for Cloud Computing

Taeweon Suh
Korea University
Seoul, South Korea
suhtw@korea.ac.kr

Jong Beom Lim
Korea University
Seoul, South Korea
jblim@korea.ac.kr

Dong Hyuk Woo
Intel Labs
Santa Clara, CA 95052, USA
donghyuk.woo@intel.com

Weidong Shi
Nokia Research Center
Palo Alto, CA 94304, USA
larryshi@ymail.com

Xinwen Zhang
Huawei America R&D Center
Santa Clara, CA 95050 USA
xinwenzhang@gmail.com

Jong Hyuk Lee
Korea University
Seoul, South Korea
spurt@korea.ac.kr

ABSTRACT

This paper presents PolyHype, a novel architecture that leverages the power of emerging many-core based cloud-on-chip systems to support multiple hypervisors (or virtual machine monitors) on a single physical platform. A PolyHype platform consists of a control plane and multiple hypervisors created on-demand. Each hypervisor can further create multiple guest virtual machines within a resource and management realm. PolyHype provides improved scalability, security, and dependability comparing with mono-hypervisor systems in today's cloud environment.

Categories and Subject Descriptors

D.4 [**Operating Systems**]:

General Terms

Design

Keywords

Virtualization, Architecture, Security, Scalability

1. INTRODUCTION

Cloud computing is emerging as a viable alternative to premise-based deployment of hardware and software solutions. The economy of scale and elasticity offered by cloud computing has garnered rapid adoption for an increasingly dynamic and competitive business climate. As a consequence, cloud computing is quickly altering the landscape of the information technology service industry. Virtualization plays a critical role in cloud computing by multiplexing the resources and computing power of a single platform to multiple logical platforms. The development of virtualization technology has turned traditional software into *virtual appliance*s and allows software and its execution environment to be rapidly deployed and delivered as services in ways that are both massively scalable and elastic. According to IDC's analysis, cloud services will be in the order of $44.2bn in 2013 [4].

Virtualization has existed long before the emergence of cloud computing. However, with the light of recent advance on low cost many-core processors, virtualiation has made cloud computing economically viable. Many-core processors have increased virtualization density to the point at which large numbers of virtual servers can be ran concurrently on a single physical server. In foreseeable future, the number of processor cores in a single processor will continue to double steadily [6]. Sooner or later, we will reach the era of hyperscale virtual server consolidation where hundreds or even thousands of virtual servers can be packed on a single many-core based physical server. This will enable virtualization based computing at epic scale.

Although cloud computing holds great potential and promises, security is one of the main challenges and deficiencies in today's cloud environment. Not surprisingly, the characteristics of multi-tenancy and shared resources introduce new risks and threats to any resources on cloud platform. Potential risks include failure of separation mechanisms for storage, memory, routings between different tenants, and hypervisor subversion [18]. Further threats come from the possibilities of "escape" from a guest virtual machine (VM) and being able to inject codes into the host system or other VMs [9, 12]. Public consent and study [5, 10] have shown major security concerns, including the reluctance to deploy virtual machines on shared physical servers (which run against the fundamental cloud computing principles of resource sharing and on-demand provisioning), potential leak and disclosure of confidential and proprietary information to third parties, and compromising of co-located virtual machines. These concerns are well justified by identified and potential vulnerabilities associated with commodity hypervisors and virtual machine systems on shared platforms [15, 3, 9, 12, 18]. In the near future, we can expect to see many new security exploitations on cloud environment towards platforms and user information.

In line with these concerns and challenges, we propose *PolyHype*, a poly-hypervisor architecture to improve the dependability, scalability, and security of many-core based cloud servers. Comparing with today's mono-hypervisor (MonoHype) based systems, PolyHype supports running multiple hypervisors or VMMs (virtual machine monitors) on a single physical platform, in turn, each of which can execute multiple VMs. By leveraging many-core based cloud-on-chip processor architecture, this extra abstraction provides strong isolation between physical resources managed by individual hypervisor *realms*. As a direct consequence, the vulnerabilities of one hypervisor or a VM within a hypervisor can be confined within its own domain, which makes platform-wide attacks much harder. With careful design on physical separation of CPUs, memory, storage, and I/O devices, PolyHype can achieve more dependable, secure, and scalable cloud server platform that fits the requirement of hyper-scale virtual server consolidation.

This paper first analyzes and summarizes security risks, threats, and attacks on existing cloud servers (section 2), and then presents the PolyHype architecture to support multiple isolated hypervisors and realms on a single many-core based physical server (section 3). We highlight several key design challenges for PolyHype, including strong separation at physical platform resource level such as CPU, memory, and storage. We also identify variant I/O virtualization options and discuss their merits on performance and compatibility aspects.

2. HYPERVISOR RELATED THREATS TO CLOUDS

In cloud computing environment, virtual machines from different cloud customers share the same physical server and hypervisor. Virtualization offers "layered defense" for system security, usually by assuming that a malicious attacker who controls or penetrates one guest virtual machine cannot compromise the underlying system and other virtual machines. This should not always be taken for granted. Previous studies have demonstrated the vulnerabilities and real attacks that determined attackers can exploit hypervisor vulnerability, and consequently compromise services of co-located cloud users [15, 3, 9, 12, 18, 14]. We summarize several risks and threats of virtualized platform in cloud computing environments as follows.

2.1 Hypervisor Vulnerabilities

On a cloud server platform, virtual machines from different customers sit above a common hypervisor that manages both the physical hardware resources and customer resources. Like any other software layer, a hypervisor can have vulnerabilities and is prone to attacks or unexpected failure. Commodity hypervisor has significantly grown in functions and features, and thus in code size. These make them look closer to a real operating system (OS) with large trusted computing base (TCB), and increasing design and implementation vulnerabilities. Their isolation and security functions might be compromised by attacks from guest OS [7].

An attacker can compromise a hypervisor by hacking it from inside a guest virtual machine and exploit all the guests. Hypervisor layer attacks are very attractive. In a MonoHype system, the hypervisor fully controls the physical resources and all guest virtual machines that run on top of it. Past few years have seen a number of successful hypervisor subversions [15, 3, 9, 12, 18, 14].

King et al., [8] described the concept of a virtual machine-based rootkit and demonstrated the subversion of VMWare and VirtualPC using hypervisor rootkit SubVirt. Blue Pill [14] is a rootkit that can trap a running native OS into a guest virtual machine "on-the-fly" with hardware-assisted virtulization technology such as Intel VT-x or AMD Pacifica. In [12], the author investigated several popular x86 based virtual machine implementations and tested whether the assumed hypervisor security and virtual machine isolation can be taken for granted. The authors performed hypervisor stress tests by injecting random instructions and I/O activities to the hypervisor from a guest virtual machine. The results identified vulnerabilities in all popular virtual machine implementations for x86 architecture in use today. If exploited, a vulnerable VMM can be subverted to execute arbitrary code on the host with the privileges of the VMM process. In addition, an exploit from a virtual machine guest could cause VMM to terminate unexpectedly or trigger an infinite loop that prevents the host from performing normal administration operations for other virtual machines.

When a hypervisor is subverted, an attacker can escape the isolation between different customers. For example, a documented attack on VMWare [15] allows "guest to host escape". After a hypervisor is subverted, an attacker may take control of other virtual machines running on the same hypervisor or gain access to the data contained inside them. Furthermore, an attacker may manipulate resource allocation; reduce resources assigned to other virtual machines and as a consequence cause denial of service.

2.2 Weak Separation between VMs

Cloud computing infrastructures mostly rely on architectural designs where physical resources such as computing capacity, storage, and network, are shared by multiple virtual machines and therefore multiple customers. Multi-tenancy and resource sharing are two of the defining characteristics of cloud computing environments. In future many-core based systems, hundred or even thousand of virtual machines may share the same physical platform. Failure of resource separation between different tenants could lead to potential devastating results such as unauthorized access to shared resources, provoking denial of services (e.g., manipulate resource allocated to other customer's virtual machines or terminate other customer's running virtual machines), and side-channel data leakage. In [13], using Amazon EC2 cloud system, the authors illustrated the steps for accessing confidential information from running EC2 instances and demonstrated side-channel exploits to EC2 instances.

2.3 Resource Exhaustion

Cloud services acquire resources in on-demand manner. In multi-tenancy working environment, malicious attackers may trigger resource exhaustion and cause denial-of-resources attacks to other users' virtual machines. Shared resources with capacity limitation include memory, storage, I/O bandwidth, networking buffers, CPU, and etc. If an attacker can trigger the allocation of these limited resources, but the number or size of the resources is not controlled, the attacker could cause a denial of service by consuming all available resources on a physical platform. For example, a memory exhaustion attack against an application could slow down the application as well as its host OS. A malicious customer may run mischievous guest virtual machines that use certain resources intensively. For example, a virtual machine can deliberately trigger a lots of interrupts or generate switches between virtual machine and hypervisor at extremely high frequency.

3. ARCHITECTURE AND DESIGN

This section describes our PolyHype design and architectural support in details.

3.1 Requirements

Comparing with the MonoHype system [2, 11], a PolyHype system in ideal scenario should satisfy the following requirements.

- Separated hypervisors that can scale with large scale many-core based platform and support hyper-scale virtual server consolidation for multiple customers;
- Two-tiered resource allocation and isolation mechanism: resources such as CPUs are first allocated and partitioned among hypervisors and then for each hypervisor, the resources are shared among guest virtual machines; and
- Security breach compartmentalization: an architecture capable of preventing security breach from spreading to other customer's virtual machines.

One desirable feature to support these requirements is that different hypervisors share minimal physical resources, thus the normal function of a hypervisor requires little or no interaction from other hypervisors. With this, an attack on one hypervisor by a malicious customer or attacker may not affect other customers' hypervisors and guest virtual machines. Ideally, PolyHype achieves the same level of availability and dependability as running each hypervisor on a dedicated MonoHype platform using single physical server.

3.2 PolyHype and VM Realms

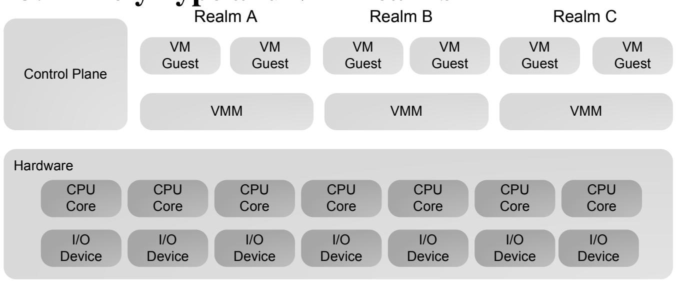


Figure 1: Concept of PolyHype Platform

On a PolyHype server, a *VM realm* refers to all guest virtual machines supported by one VMM or hypervisor. A PolyHype server may constitute multiple concurrent VM realms, and launch new VMMs in on-demand manner. A PolyHype server has a single control plane that administrates the physical machine and is able to retain selective control of resources, including processor cores, physical memory, interrupt management, and I/O devices. The control plane can be a VM realm, aka *manager realm*, while others are *regular realms*. The manager realm does not run guest virtual machine for cloud customers directly. It can allocate physical resources to a VMM, bootstrap, or terminate the VMM. After started, a VMM can function as a normal hypervisor, i.e., it can manage a number of virtual machine guests and act as a host for the guests. The control plane runs at the highest privilege level, thus ensures isolation among VMMs by allocating or partitioning resources between individual VMMs. The allocated resource can be physical or virtual (details discussed in the remainder of this section).

	Dedicated Servers	MonoHype	PolyHype
Hypervisor vulnerabilities	NA	all VMs affected	limited
Resource monopolization risks	NA	high (shared by all VMs)	low (two tiered resource partition and sharing
Many-core scalability	NA	limited	better
Exploit isolation	yes	limited	better

Table 1: Comparison of Three Frameworks

For strong isolation purpose, each realm comprises at least one physical processor core and allocated physical RAM space. There is no overlap on processor cores and RAM space for different realms. For a regular realm, its processor cores (one or more) run at lower privilege level than the manager realm. This prevents a regular hypervisor from changing resource allocation made by the control plane. After a hypervisor is started, the control plane delegates control of the allocated physical resources to the started hypervisor. In turn, the hypervisor can further create guest virtual machines and allocate assigned resources by the control plane to the guests. The design presents an additional abstraction layer between the guest virtual machines and the physical host. Interrupts for each VM realm are routed and handled by the corresponding hypervisor for the realm. Page faults and exceptions caused by guest virtual machines of a realm are handled by the realm's hypervisor just like in normal MonoHype systems.

When a hypervisor starts, it boots from a modified BIOS that bypasses physical RAM initialization. The control plane retains the control of certain physical resources such as physical memory allocation and I/O device discovery.

Table 1 compares PolyHype, MonoHype, and dedicated physical servers from aspects of vulnerabilities to hypervisor breach, resource monopolization risks, many-core scalability, and exploit isolation. Overall, PolyHype is more scalable and secure, and well positioned for the emerging cloud-on-chip and growth of hyper-scale virtual machine density.

3.3 VM Realm Memory Mapping

To support strong memory partitioning between multiple hypervisors on a single platform, we propose a physical memory remapping mechanism. In particular, physical memory space is divided into chunks of equal size (e.g., 8MB). The control plane assigns physical memory chunks to each hypervisor and its VM realm. The remapping mechanism restricts memory access from a hypervisor and VM realm to pre-assigned physical memory regions. This is achieved by a hardware physical memory remapping logic situated in the memory controller. The remapping logic creates a virtual continuous physical memory space for each hypervisor and its VM realm. It is programmed by the privileged control plane. A regular hypervisor running at lower privilege level cannot modify or program the remapping logic. For each memory access (e.g., read or write access from a hypervisor and VM realm), the memory remapping logic translates the address of the access request to its corresponding physical memory address. It serves as a memory access reference monitor and performs access control based on configurations provided by the control plane. For translating memory addresses, the memory controller uses either a VM realm to physical memory remapping table managed by the control plane or a local cache of the remapping table. The remapping table cache is part of the memory controller. It resembles TLB inside MMU and caches recently-used entries of the VM realm-to-physical memory remapping table.

Unlike a regular hypervisor, the control plane or privileged manager realm can access the entire physical memory space without using translation. However, only a portion of the physical memory space is allocated to the control plane. Operating systems or virtual machines within the control plane can use the physical memory space allocated to them freely for their own purposes. The rest physical memory space is reserved for other hypervisors and VM realms, while the control plane has read/write access rights to it. Overall, physical memory isolation for PolyHype is achieved by restricting memory access from hypervisor and VM realm within the space that is assigned by the control plane, by using the remapping or the address-translation tables.

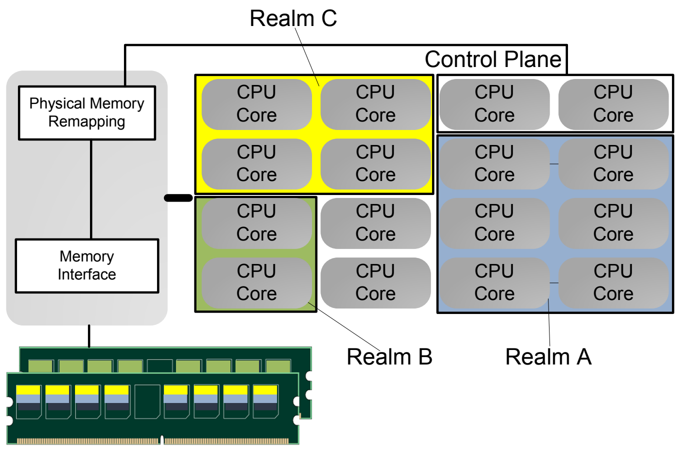


Figure 2: Physical Memory Remapping

Figure 2 shows an example physical memory allocation for three VM realms: A, B, and C. When a hypervisor or a VM tries to access to a certain memory location, the remapping hardware looks up the address-translation tables for access permission of the realm to the specific location. If the hypervisor or VM realm tries to access outside of the memory range assigned to it by the control plane, the remapping hardware blocks the access and reports a fault to the control plane, which is achieved by raising exception to the processor core running the control plane.

The described physical memory remapping is different from traditional virtual memory management in many aspects. Traditional memory paging and MMU are tied with process management, while our physical memory remapping mechanism is used for partitioning physical memory resources among multiple VM realms. It presents a "virtual" continuous physical memory space for each hypervisor.

3.4 I/O Support

One of the main design challenges of PolyHype is I/O virtualization support. In this subsection, we briefly explain several design options towards this and discuss the trade-offs between them.

To virtualize I/O devices, the following operations need to be supported: 1) device discovery and configuration: It includes operations for querying I/O devices on a hardware platform and setting up the devices' configuration registers for initialization; 2) I/O transactions: Data Transfer to and from devices including DMA; and 3) interrupts, by notifying a hypervisor on state updates and events of a device.

Typically I/O virtualization in MonoHype is implemented using one of three different approaches: emulation, para-virtualization, and hardware assisted virtualization (e.g., direct assignment on Intel VT-d [1] or single-root IOV where an I/O device can be shared by multiple VMs). Emulation implements I/O devices and hardware in software. Para-virtualization requires modification of a guest OS. Hardware assisted I/O virtualization such as VT-d can allocate an I/O device to a VM using IOMMU, a memory address translation for I/O transactions. Single-root IOV enables virtual functions on I/O devices and allows an I/O device to be shared by multiple VMs.

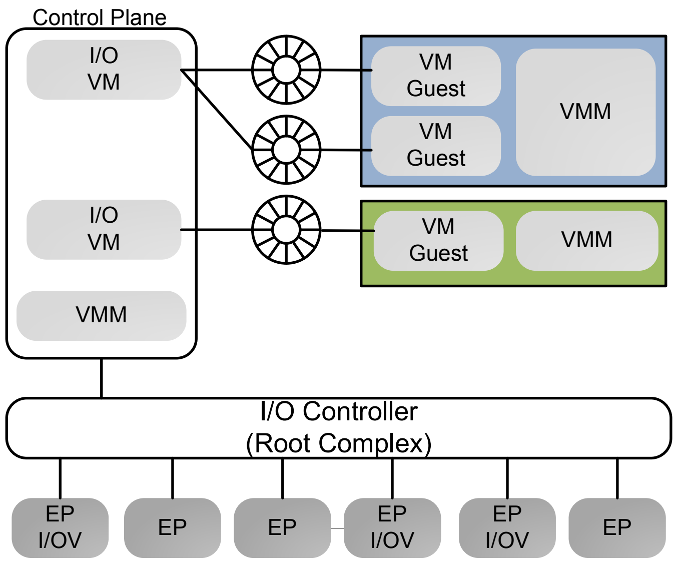
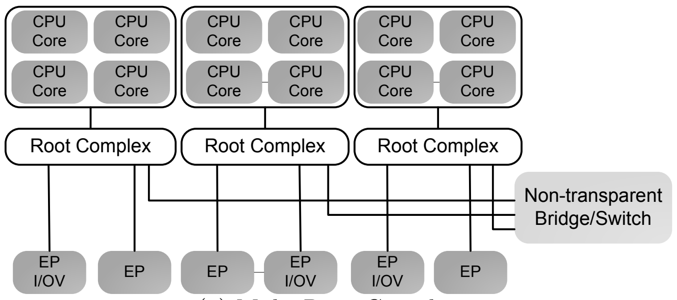
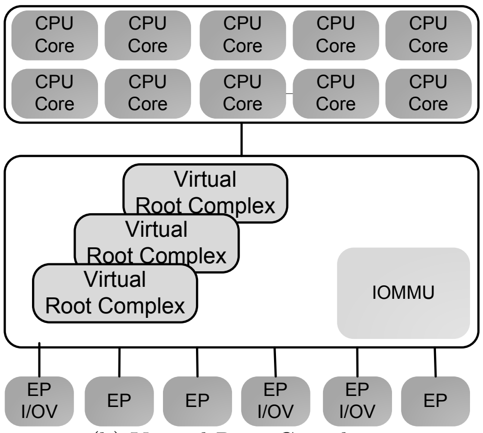


Figure 3: IO Proxy Mode

IO Proxy Mode: One design option is to use device emulation for I/O virtualization, as Figure 3 shows. The con-



(a) Multi Root Complexes



(b) Virtual Root Complexes

Figure 4: Multi vs. Virtual Root Complexes

trol plane consists of a hypervisor and a number of I/O virtual machines, and has the full control of physical I/O resources. I/O devices are allocated to I/O virtual machines in control plane using hardware assisted I/O virtualization such as single-root IOV or VT-d. A regular hypervisor and its VMs can only access the physical I/O resources using the control plane services. In a VM realm, a guest virtual machine uses emulated device drivers, which communicate with the I/O virtual machines in control plane using physical shared memory. This is feasible because the control plane has control of the entire physical memory space. Note that the control plane does not run virtual machines for cloud customers. Virtual machines on the control plane are executed by dedicated processor cores in parallel with customers' virtual machines in regular realms. At high level, the system functions like a distributed systems where the control plane and I/O virtual machines act as I/O proxies for the guest virtual machines of a regular realm, by performing I/O transactions and issuing DMA data transfer on behalf of the emulated device drivers. A drawback of this approach is that it does not scale well if there are numerous I/O transactions from multiple guests in a VM realm or VM realms since the control plane is one central place to process the transactions.

Multiple Root Complexes: Yet another option is to use systems with multiple physical I/O root complexes. As Figure 4a shows, hardware I/O and CPU resources are partitioned among hypervisors and VM realms. Each hypervisor controls the hardware I/O resources and devices allocated to it. The control plane retains certain high level control of the hardware resources such as start/reset of the CPU cores and I/O devices. For each VM realm, its hypervisor manages the I/O devices under its control. It performs device discovery, control, and configuration for the realm's virtual machines. These virtual machines in a VM realm share the I/O devices managed by its hypervisor. Within the realm, the hypervisor can assign I/O devices to virtual machines using hardware I/O virtualization support (e.g., IOMMU). The system uses a non-transparent I/O bridge/switch to interconnect the multiple I/O root complexes.

Using Existing I/O Virtualization: A third option is to leverage the existing hardware support for MonoHype I/O virtualization. It works as follows. The control plane retains the control of I/O devices and assigns I/O resources to guest virtual machines of a realm. This allows I/O virtualization on a PolyHype system using existing available I/O virtualization support. The control plane performs device discovery, manages the I/O devices, and assigns devices to guest virtual machines of a VM realm. One guest virtual machine can issue DMA data transfer using hardware I/O virtualization such as VT-d or IOMMU without involving the control plane. When it needs to access protected resources (such as I/O configuration and interrupt management), it first exits into the hypervisor of its realm. The hypervisor then sends an interrupt (such as Inter-processor Interrupt (IPI)) to the processor cores running the control plane. The control plane handles the request and returns results to the hypervisor. For each regular VM realm, its hypervisor cannot perform these I/O controlling functions which are reserved for the control plane. It can only forward the requests from its guests to the control plane.

Virtual Root Complexes: A fourth option is to use a virtual I/O root complex mechanism shown in Figure 4b. An I/O controller on a single platform can support the discovery of multiple virtual I/O root complexes. Both device assignment assisted by platform hardware I/O virtualization such as IOMMU and device level IOV are supported. The control plane configures the I/O controller and allocates virtual I/O root complex to each hypervisor and VM realm.

Of these four options, the first and third option work with the existing I/O virtualization support in MonoHype based systems. The third option can leverage existing hardware support for I/O virtualization such as IOMMU or single-root IOV. The multiple root complex approach does not require any new standard and supports PolyHype I/O virtualization using replicated I/O hardware. It can achieve higher performance but has less flexibility comparing with the first and third option. The virtual root complex approach requires hardware support and standard upgrade for multiple virtualized I/O root complexes.

3.4.1 VM Migration

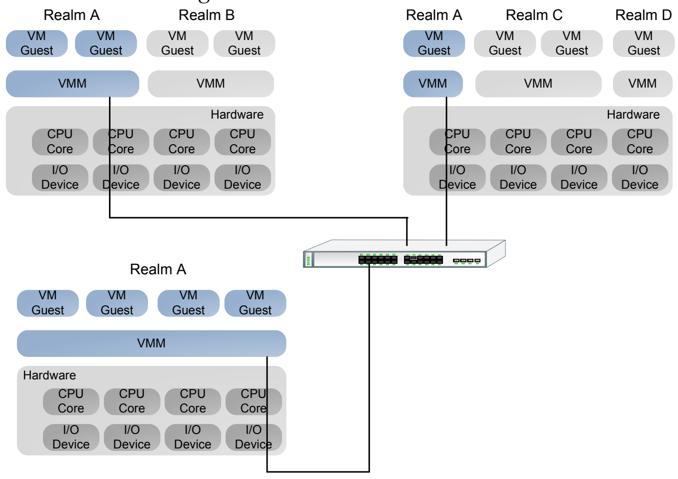


Figure 5: Guest VM migration.

In general, a VM realm can cross multiple physical PolyHype servers, as Figure 5 shows. To migrate a guest virtual machine from one physical server to another, the target server has to create a corresponding realm hypervisor and allocate resources first. Then the guest virtual machine itself can be migrated.

4. RELATED WORK

In this section we compare our system with related works, which we classify them under two categories:

The first category includes hardware and architectural support for CPU and I/O virtualization (e.g., [1, 17]). To our knowledge, majority of the published studies and designs in this space focus on supporting single hypervisor based systems. In contrast to these systems, our system is one of the firsts that employ new architectural features to support multi-hypervisor based platform.

Another body of related research includes micro-kernel based hypervisor. NOVA [16] is a micro-kernel based hypervisor that uses a thin and simple virtualization layer to reduce the attack surface and as a result improve system security. Our multi-hypervisor based system is orthogonal and complementary to micro-kernel hypervisor research. Comparing with micro-kernel based hypervisor, our system eliminates the necessity of sharing hypervisor by different cloud customers on a platform and thereby improves platform scalability, reliability, and security.

5. CONCLUSION

We present the concept and high level architecture of PolyHype, a platform framework to support multiple hypervisors on single physical platform by leveraging the emerging many-core cloud-on-chip processor. Each hypervisor manages guest virtual machines like traditional virtualization mechanism. The strong isolation between hypervisors and their realms is achieved by separation of physical resources provided by a control plane of the platform, which makes each hypervisor run on logically isolated host. PolyHype provides improved scalability, security, and dependability beyond legacy virtualization platform. Many technical challenges have to be addressed towards the support of PolyHype, including physical memory isolation and I/O virtualization.

6. ACKNOWLEDGEMENTS

This research is supported in part by Korea Research Foundation (KRF-2010-0015511).

7. REFERENCES

[1] Abramson, D., Jackson, J., Muthrasanallur, S., Neiger, G., Regnier, G., Sankaran, R., Schoinas, I., Uhlig, R., Vembu, B., and Weigert, J. Intel Virtualization Technology for directed I/O. *Intel Technology Journal 10*, 3 (Aug. 2006), 179–192.

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Virtual machine monitors: Xen and the art of virtualization. In *SOSP '03: proceedings of the 19th ACM Symposium on Operating Systems Principles: the Sagamore, Bolton Landing, Lake George, New York, USA, October 19–22, 2003* (New York, NY 10036, USA, Dec. 2003), ACM, Ed., vol. 37(5) of *Operating systems review*, ACM Press, pp. 164–177. ACM order number 534030.

[3] Ferrie, P. Attacks on virtual machine emulators. *Symantec Security Response 5* (2006).

[4] Frank Gens, Robert P Mahowald, R. L. V. An empirical study into the security exposure to hosts of hostile virtualized environments, 2007.

[5] Heiser, J., and Nicolett, M. Assessing the security risks of cloud computing. `http://www.gartner.com/DisplayDocument?id=685308`, 2009.

[6] Held, J., Bautista, J., and Koehl, S. White paper from a few cores to many: A tera-scale computing research review.

[7] Karger, P. A., and Safford, D. I/O for virtual machine monitors: Security and performance issues. *IEEE Security & Privacy 6*, 5 (2008), 16–23.

[8] King, S. T., Chen, P. M., min Wang, Y., Verbowski, C., Wang, H. J., and Lorch, J. R. Subvirt: Implementing malware with virtual machines. In *IEEE Symposium on Security and Privacy* (2006), pp. 314–327.

[9] Kortchinsky, K. Cloudburst – hacking 3D and breaking out of VMware. In *Black Hat USA* (2009).

[10] Mell, P. Nist presentation on effectively and securely using the cloud computing paradigm v26. `http://csrc.nist.gov/groups/SNS/cloud-computing/index.html`, 2009.

[11] Neiger, G., Santoni, A., Leung, F., Rodgers, D., and Uhlig, R. Intel Virtualization Technology: Hardware support for efficient processor virtualization. *Intel Technology Journal 10*, 3 (Aug. 2006), 167–177.

[12] Ormandy, T. An empirical study into the security exposure to hosts of hostile virtualized environments. In *CanSecWest* (2007).

[13] Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. Hey, you, get off of my cloud: exploring

information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security* (New York, NY, USA, 2009), CCS '09, ACM, pp. 199–212.

[14] Rutkowska, J. Blue pill. In *Black Hat USA* (2006).

[15] Secunia. Advisory sa37081 - VMware ESX sever uodate for DHCP, kernel, and JRE. `http://secunia.com/advisories/37081/`.

[16] Steinberg, U., and Kauer, B. NOVA: a microhypervisor-based secure virtualization architecture. In *Proceedings of the 5th European conference on Computer systems* (New York, NY, USA, 2010), EuroSys '10, ACM, pp. 209–222.

[17] Uhlig, R. Forward: Intel Virtualization Technology: Taking virtualization mainstream on Intel architecture platforms. *Intel Technology Journal 10*, 3 (Aug. 2006), v–vi.

[18] Wojtczuk, R. Subverting the Xen hypervisor. In *Black Hat USA* (2008).