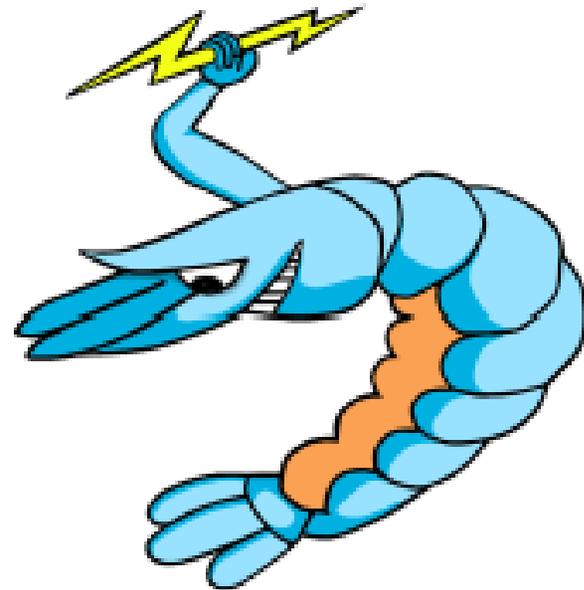


Architecting a Symbiotic Virtual Machine Monitor for Scalable High Performance Computing

John R. Lange



Department of Electrical Engineering and Computer Science
Northwestern University

January 29, 2010

Outline

- Palacios
 - Developed **first** VMM for scalable High Performance Computing (HPC)
- Largest scale study of virtualization
 - **Proved HPC virtualization is effective at scale**
- Symbiotic Virtualization
 - Created high level interfaces to enable guest/VMM cooperation
- SwapBypass
 - Leveraged symbiotic interfaces to improve swap performance

Past and Current Research

- **Palacios:** Symbiotic virtualization at scale
 - *OSDI 2010* (in preparation): Scalable virtualization for HPC
 - *USENIX 2010* (in submission): Symbiotic Virtualization
 - *IPDPS 2010*: Palacios VMM and scaling
 - **USENIX 2010** (in submission): Adaptive virtual paging
 - **WIOV 2008**: Virtual passthrough I/O
 - **OSR 2009**: Virtual passthrough I/O
- **Empathic Systems:** Bridging systems and HCI
 - **INFOCOM 2010**: Empathic networks
 - **SIGMETRICS 2009**: Empathic networks
 - **USENIX 2008**: Speculative remote display
- **Virtuoso:** Adaptive virtual infrastructure as a service (IaaS) cloud
 - **HPDC 2007**: Transparent network services
 - **ICAC 2006**: Formalization of cloud adaptation
 - **MAMA 2005**: Formalization of cloud adaptation
 - **HPDC 2005**: Automatic optical network reservations
 - **Patent # 20080155537**
- **Vortex:** Cooperative traffic aggregation for intrusion detection systems
 - **RAID 2007**: Cooperative selective wormholes

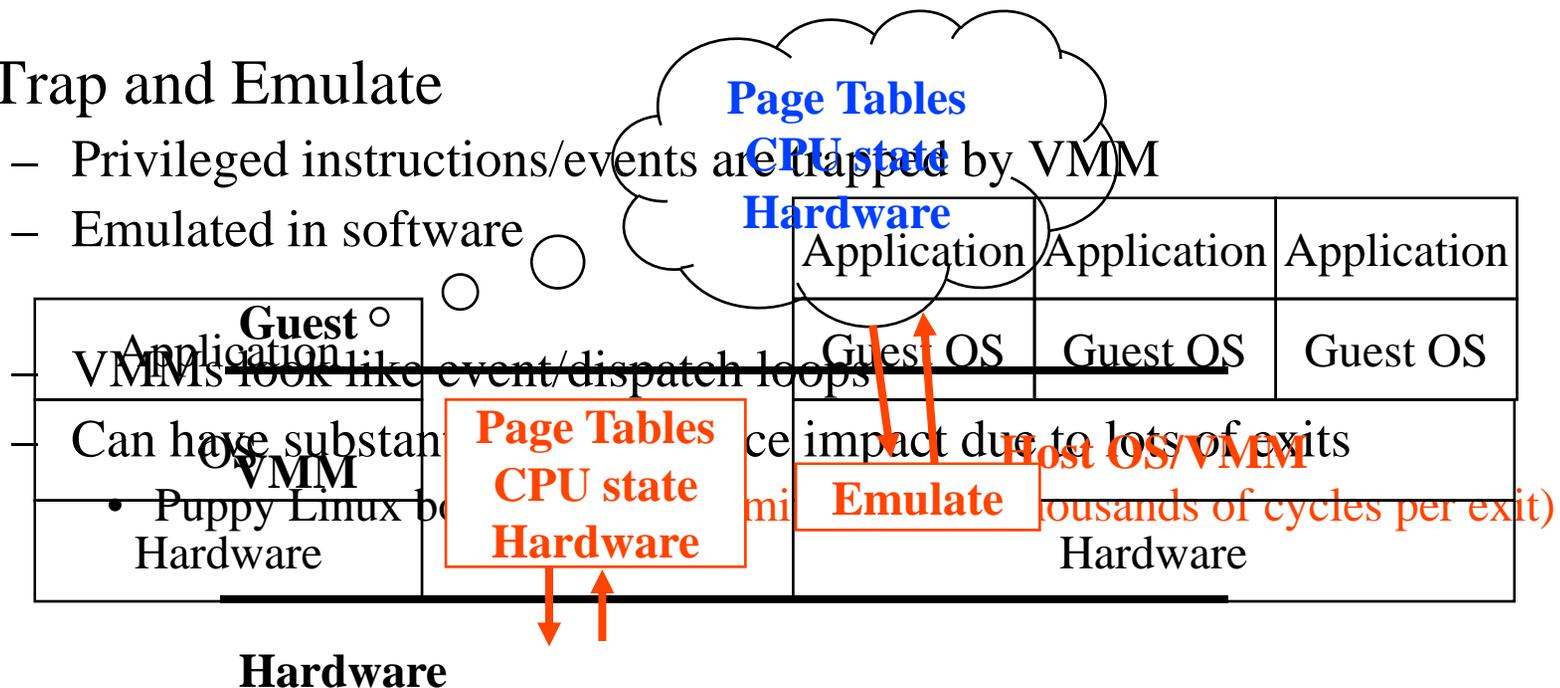


What is a virtual machine?

- Run an OS as an application
 - Run multiple OS environments on a single machine
 - Start, stop, pause
 - Can easily move entire OS environments

- Trap and Emulate

- Privileged instructions/events are trapped by VMM
- Emulated in software



What are VMMs currently used for?

- Server Consolidation
 - Fault tolerance
 - Legacy application support
 - Debugging
 - Isolation
 - Virtual appliances
 - Failover and disaster recovery
-
- Market size
 - 2007: \$5.5 billion
 - 2011: \$11.7 billion

amazon.com®

**CITRIX**®
\$7.58 Billion

vmware®
\$16.70 Billion

SWSOFT®
PARALLELS®

Eucalyptus
Systems

Virtualization in HPC

- Fault tolerance
 - Red Storm MTBI target: 50 hours
 - Red Storm MTTR: 30 minutes – 1 hour

System	# CPUs	MTBF/I
ASCI Q	8,192	6.5 hrs
ASCI White	8,192	5/40 hrs ('01/'03)
PSC Lemieux	3,016	9.7 hrs
Google	15,000	20 reboots/day

A.B. Nagarajan, F. Mueller, C. Engelmann, and S.L. Scott
Proactive Fault Tolerance for HPC with Xen Virtualization
ICS 2007

- Broader usage
 - Allow applications to select best OS
- Only if it doesn't degrade performance...
 - Tightly coupled parallel applications
 - Very large scale

Scalability

- Scale causes lots of problems
 - Small intra-node performance losses become incredibly large
- Per-node overhead has a Butterfly Effect
 - OS timers, deferred work, kernel threads
 - “OS Noise”
- ~~Linux reduces performance by X%~~
- Linux delivers performance of ~0%
- Performance at scale is not always correlated with local performance
 - 5% loss for 1 node does not equal 5% loss at scale
 - Example: Paragon

Kitten

- Open-source Lightweight Kernel
 - Exports Linux compatible ABI
 - Subset of features
- LWK for a wide range of HPC applications
 - Open source version of Catamount lineage
- **Contributing developer**
 - <http://software.sandia.gov/trac/kitten>
 - <http://code.google.com/p/kitten/>



Palacios VMM

- OS-independent embeddable virtual machine monitor
- Developed at Northwestern and University of New Mexico
 - Lead developer and graduate student
- Open source and freely available
 - Downloaded over 1000 times as of July
- Virtualization layer for Kitten
 - Lightweight supercomputing OS from Sandia National Labs
- Successfully used on supercomputers, clusters (Infiniband and Ethernet), and servers

Palacios

An OS Independent Embeddable VMM

<http://www.v3vee.org/palacios>



Palacios as an HPC VMM

- Minimalist interface
 - Suitable for an LWK
- Compile and runtime configurability
 - Create a VMM tailored to specific environments
- Low noise
- Contiguous memory preallocation
- Passthrough resources and resource partitioning

HPC Performance Evaluation

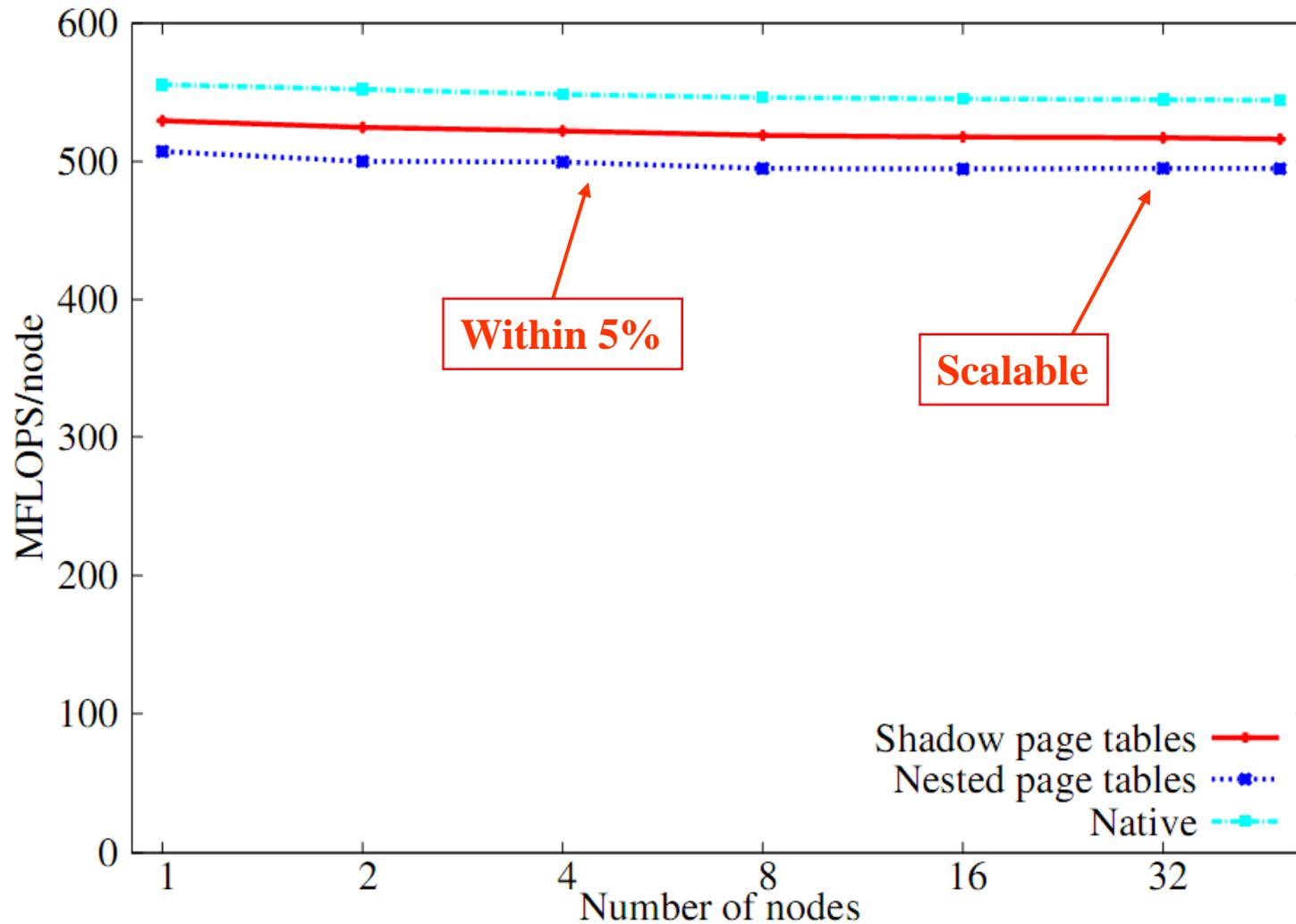
- Virtualization is very useful for HPC, but...
Only if it doesn't hurt performance
- Virtualized RedStorm with Palacios
 - Evaluated with Sandia's system evaluation benchmarks

17th fastest supercomputer

Cray XT3
38208 cores
~3500 sq ft
2.5 MegaWatts
\$90 million

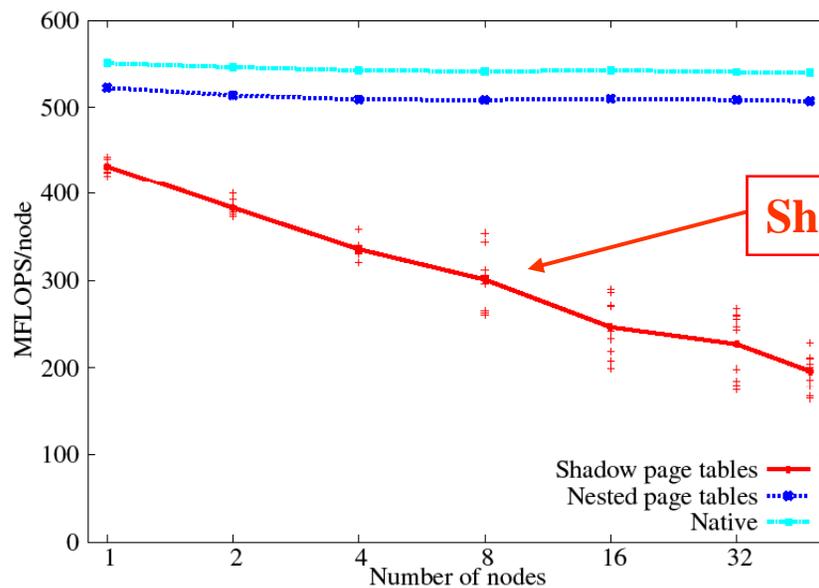


Scalability at Small Scales

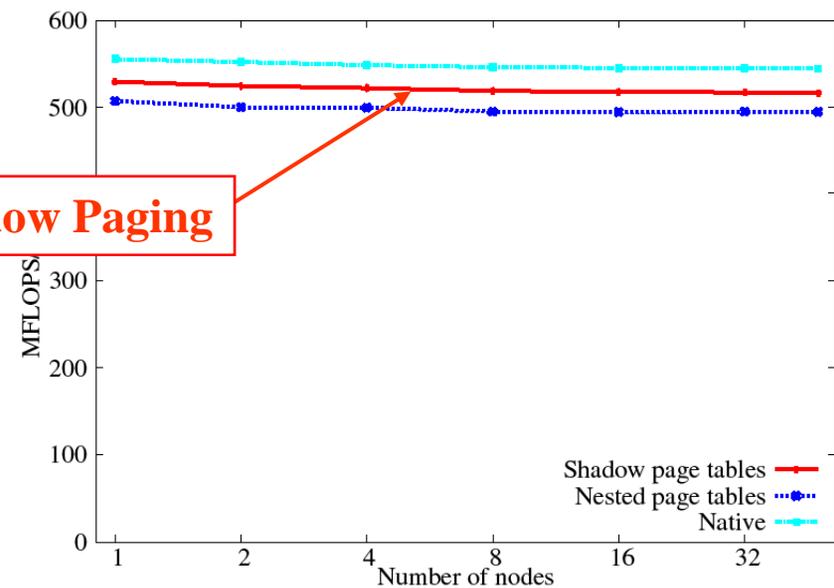


HPCCG: conjugant gradient solver

Comparison of Operating Systems



Compute Node Linux

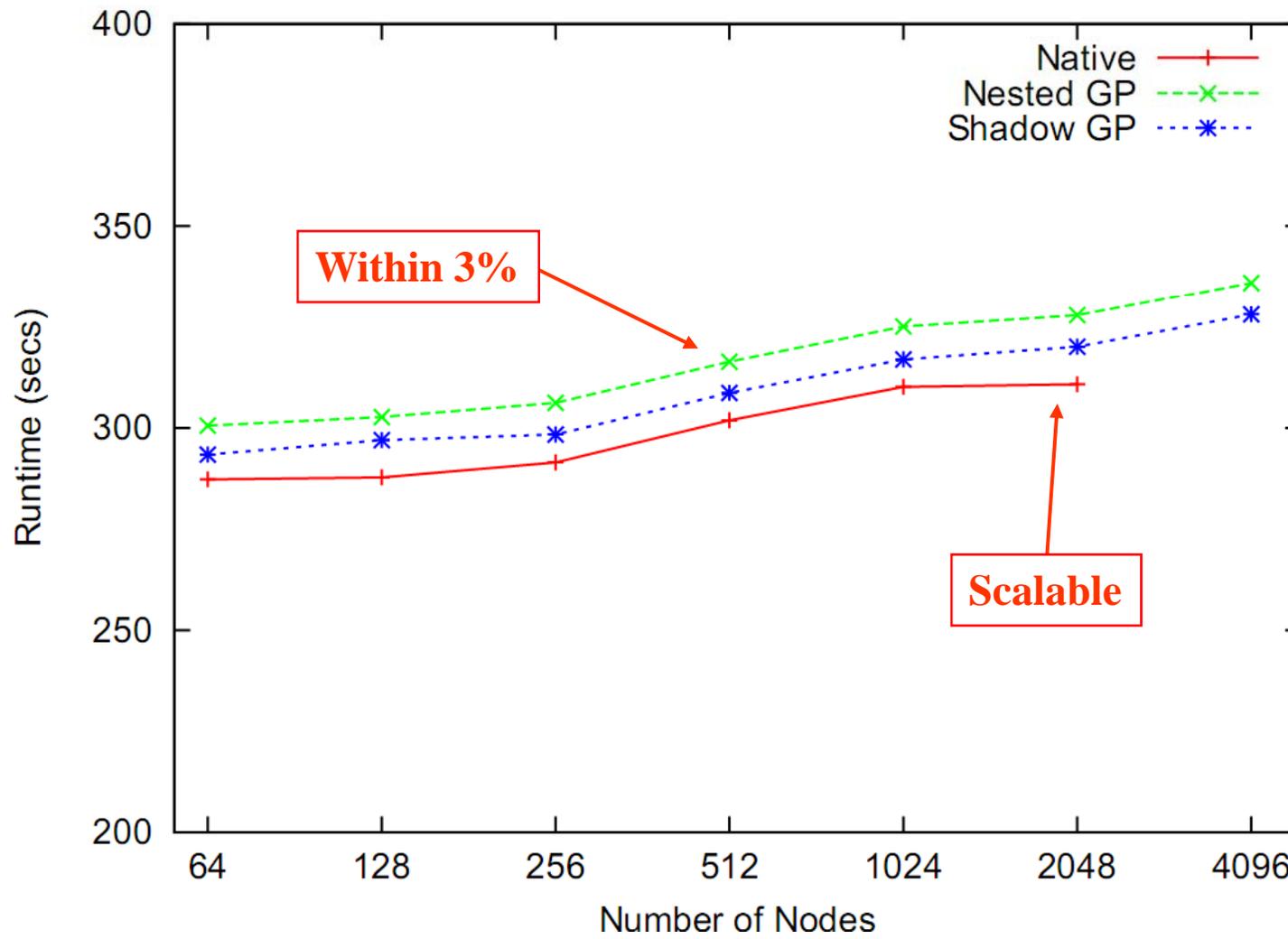


Catamount

Large Scale Study

- Evaluation on full RedStorm system
 - 12 hours of dedicated system time on full machine
 - Largest virtualization performance scaling study to date
- Measured performance at exponentially increasing scales
 - Up to 4096 nodes
- Publicity
 - *New York Times*
 - *HPCWire*
 - *Communications of the ACM*
 - *PC World*

Scalability at Large Scale



CTH: multi-material, large deformation, strong shockwave simulation

Summary

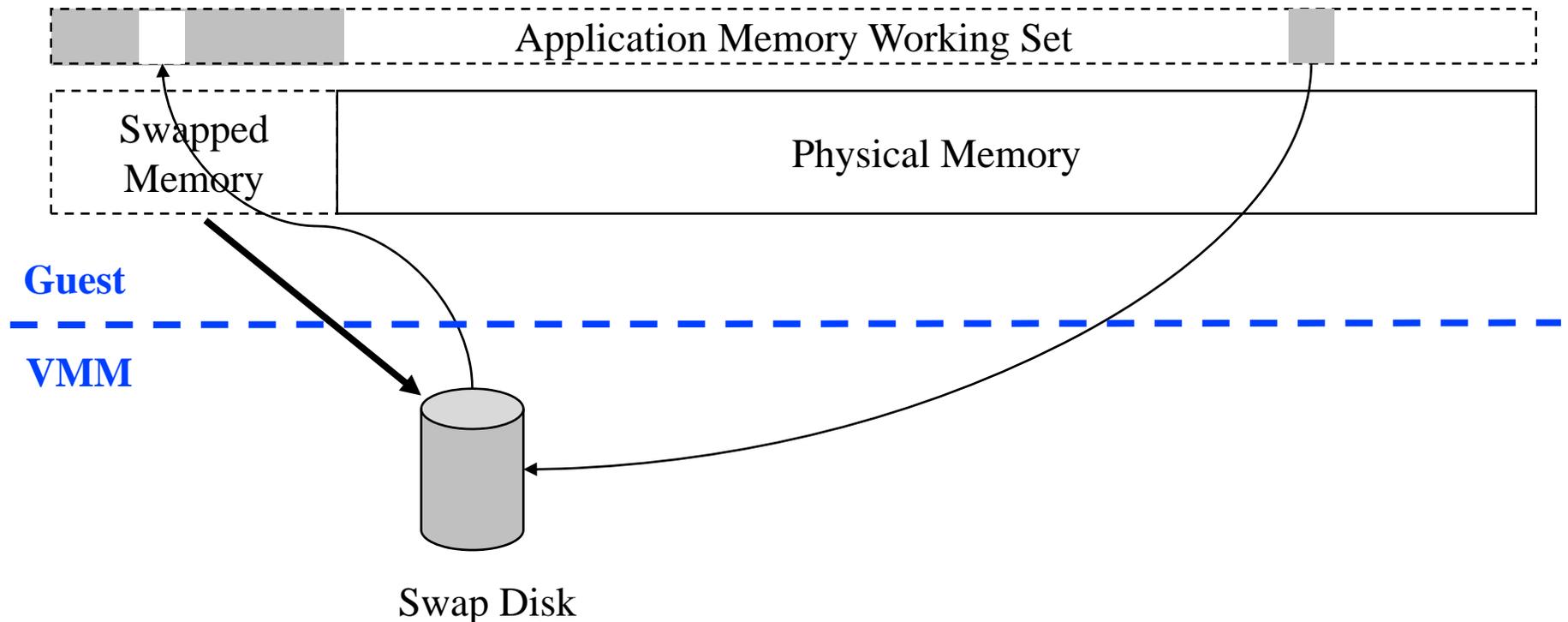
- Virtualization can scale
 - Near native performance for optimized VMM/guest (within 5%)
- VMM needs to know about guest internals
 - Should modify behavior for each guest environment
 - Example: Paging method to use depends on guest
- Black Box inference is not desirable in HPC environment
 - Unacceptable performance overhead
 - Convergence time
 - Mistakes have large consequences
- Need guest cooperation
 - Guest and VMM relationship should be symbiotic (Thesis Work)

Semantic Gap

- VMM architectures are designed as black boxes
 - Explicit OS interface (hardware or paravirtual)
 - Internal OS state is not exposed to the VMM
- Many uses for internal state
 - Performance, security, etc...
 - VMM must recreate that state
 - “Bridging the Semantic Gap”
 - [Chen: HotOS 2001]
- Two existing approaches: Black Box and Gray Box
 - Black Box: Monitor external guest interactions
 - Gray Box: Reverse engineer internal guest state
 - Examples
 - Virtuoso Project (Early graduate work)
 - Lycosid, Antfarm, Geiger, IBMon, many others

Example: Swapping

- Disk storage for expanding physical memory



Only basic knowledge without internal state

Proposal

- Bridging the semantic gap is hard
 - Can we design a virtual machine interface with no gap?
- Symbiotic Virtualization
 - Design both guest OS and VMM to minimize semantic gap
 - 2 components
 - Guest OS provides internal state to VMM
 - Guest OS services requests from VMM
 - Interfaces are optional

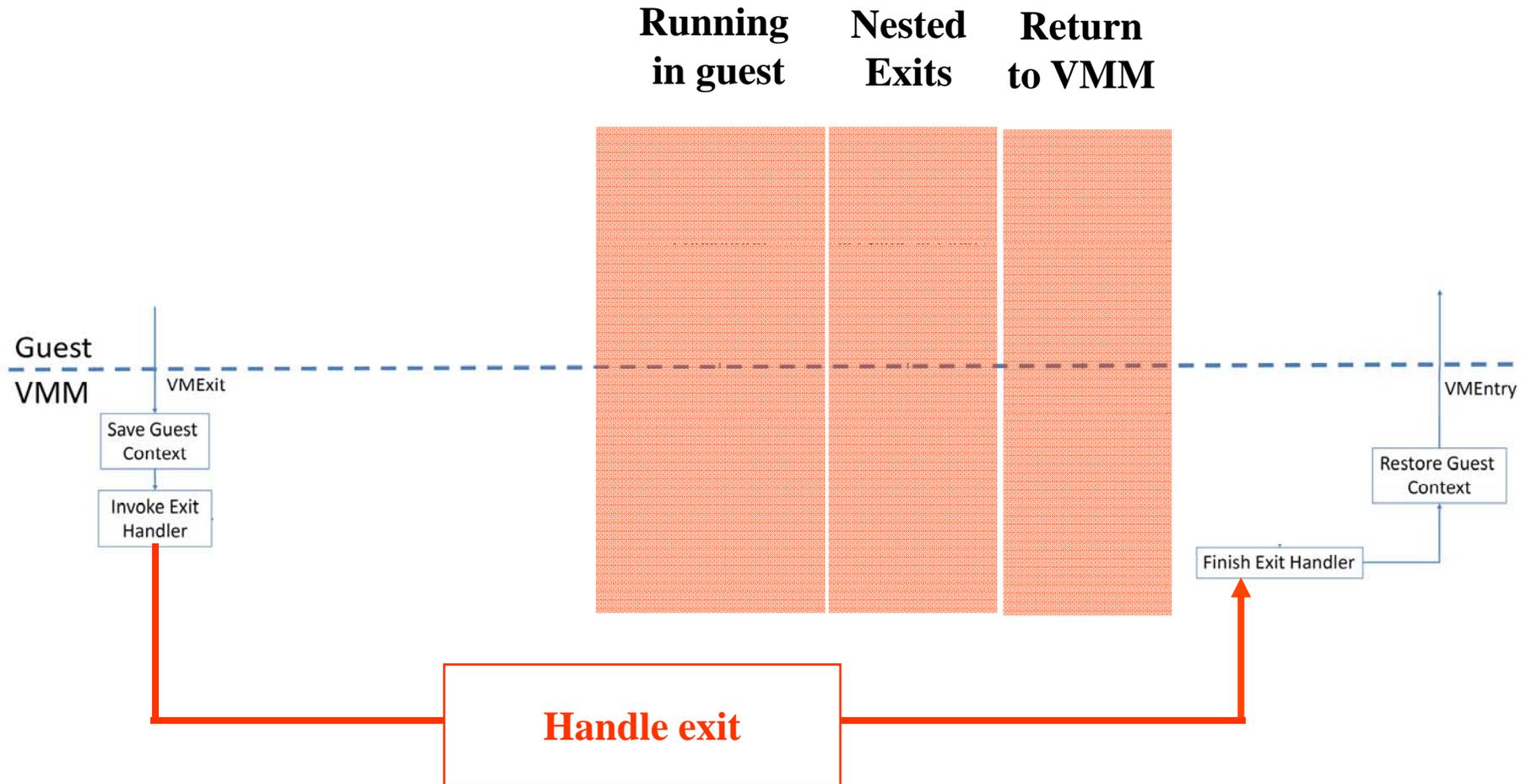
Symbiotic Interfaces

- **SymCall Functional Interface**
 - **Synchronous** upcalls into guest *during exit handling*
 - **API**
 - Function call in VMM
 - System call in Guest
 - Brand new interface construct
- **SymSpy Passive Interface**
 - Internal state already exists but it is hidden
 - **Asynchronous** bi-directional communication
 - via shared memory
 - Structured state information that is easily parsed
 - Semantically rich

SymCall (Symbiotic Upcalls)

- Conceptually similar to System Calls
 - System Calls: **Application** requests OS services
 - Symbiotic Upcalls: **VMM** requests OS services
- Designed to be architecturally similar
 - Virtual hardware interface
 - Superset of System Call MSR
 - Internal OS implementation
 - Share same system call data structures and basic operations
- Guest OS configures a special execution context
 - VMM instantiates that context to execute synchronous upcall
 - Symcalls exit via a dedicated hypercall

SymCall Control Flow



Existing Implementation

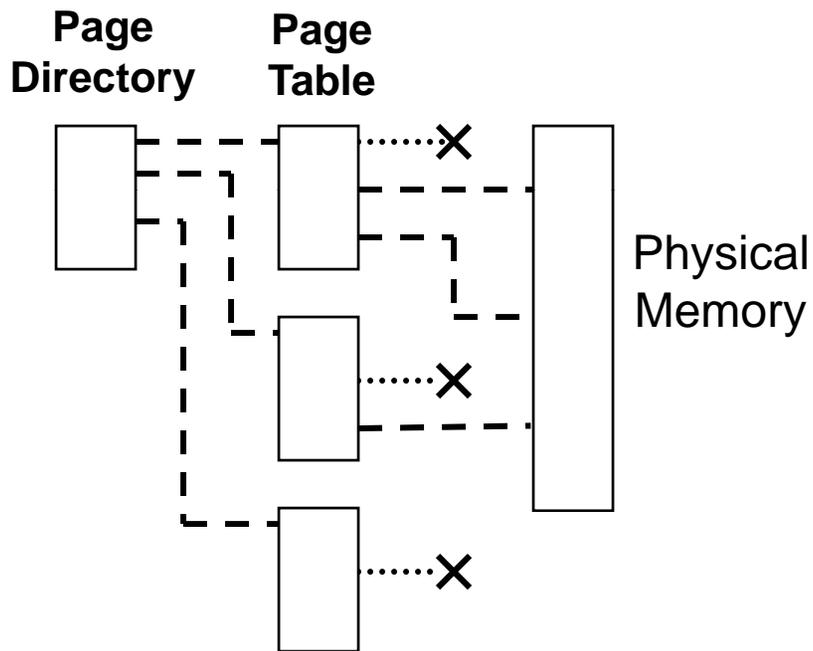
- Symbiotic Linux guest OS
 - Exports **SymSpy** and **SymCall** interfaces
- Palacios
 - Fairly significant modifications to enable nested VM entries
 - Re-entrant exit handlers
 - Serialize subset of guest state out of global hardware structures

SwapBypass

- Purpose: improve performance when swapping
 - Temporarily expand guest memory
 - Completely bypass the Linux swap subsystem
- Enabled by SymCall
 - Not feasible without symbiotic interfaces
- VMM detects guest thrashing
 - Shadow page tables used to prevent it

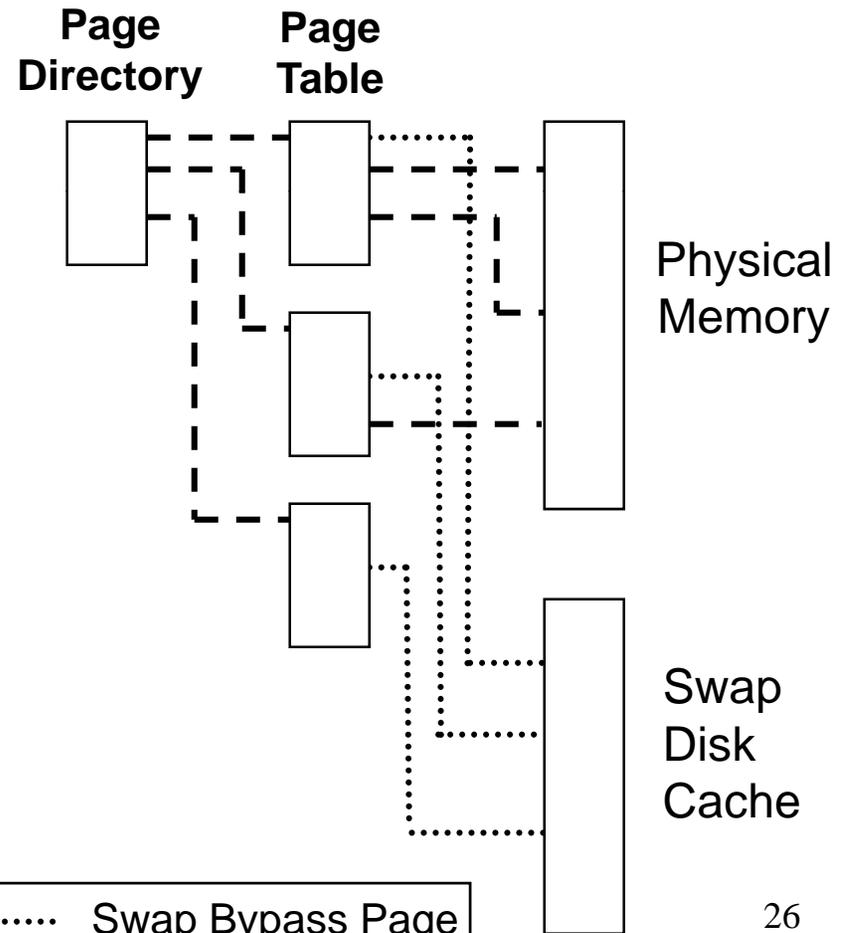
Symbiotic Shadow Page tables

Guest Page Tables



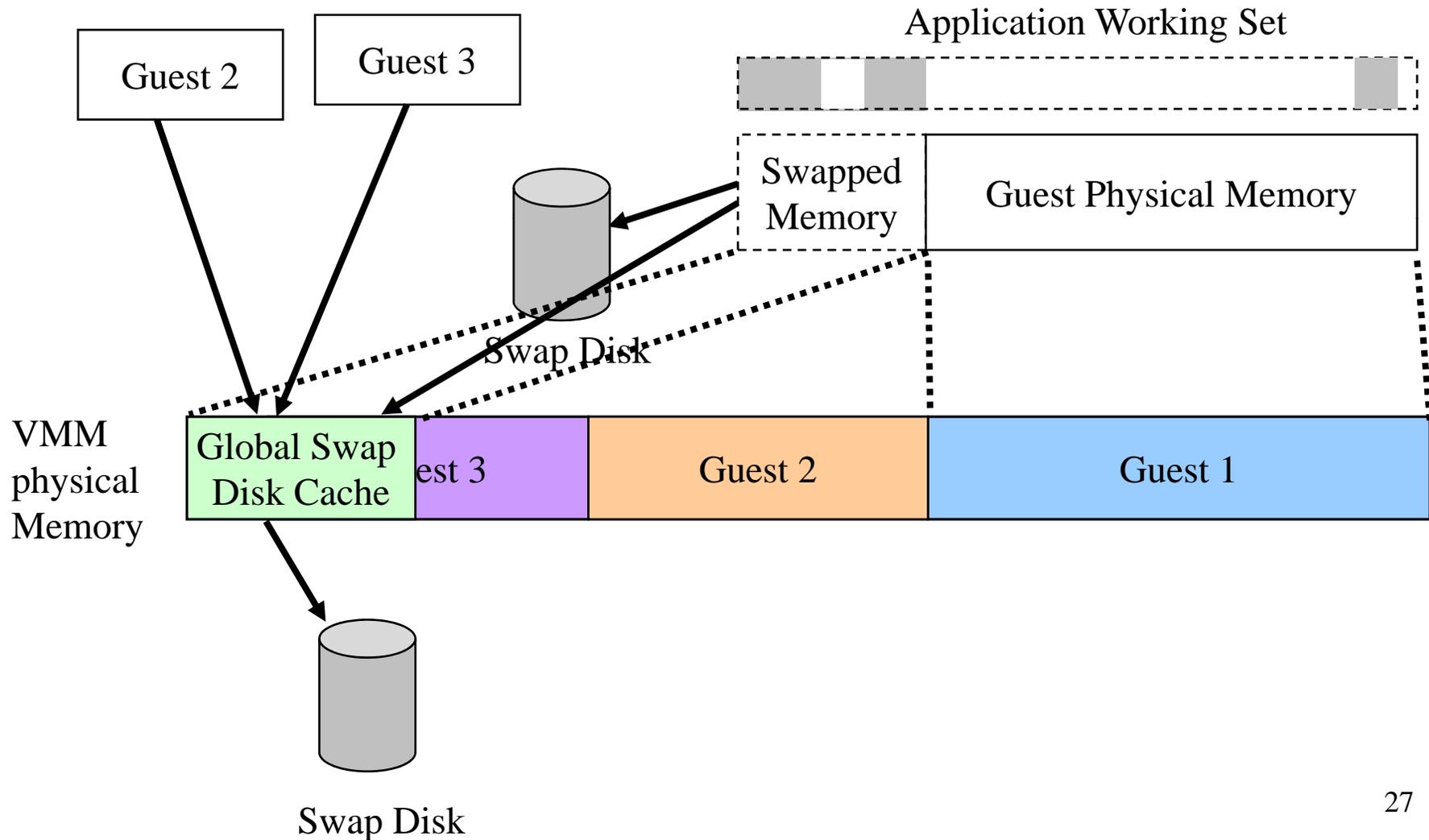
.....X Swapped out page

Shadow Page Tables



..... Swap Bypass Page

SwapBypass Concept



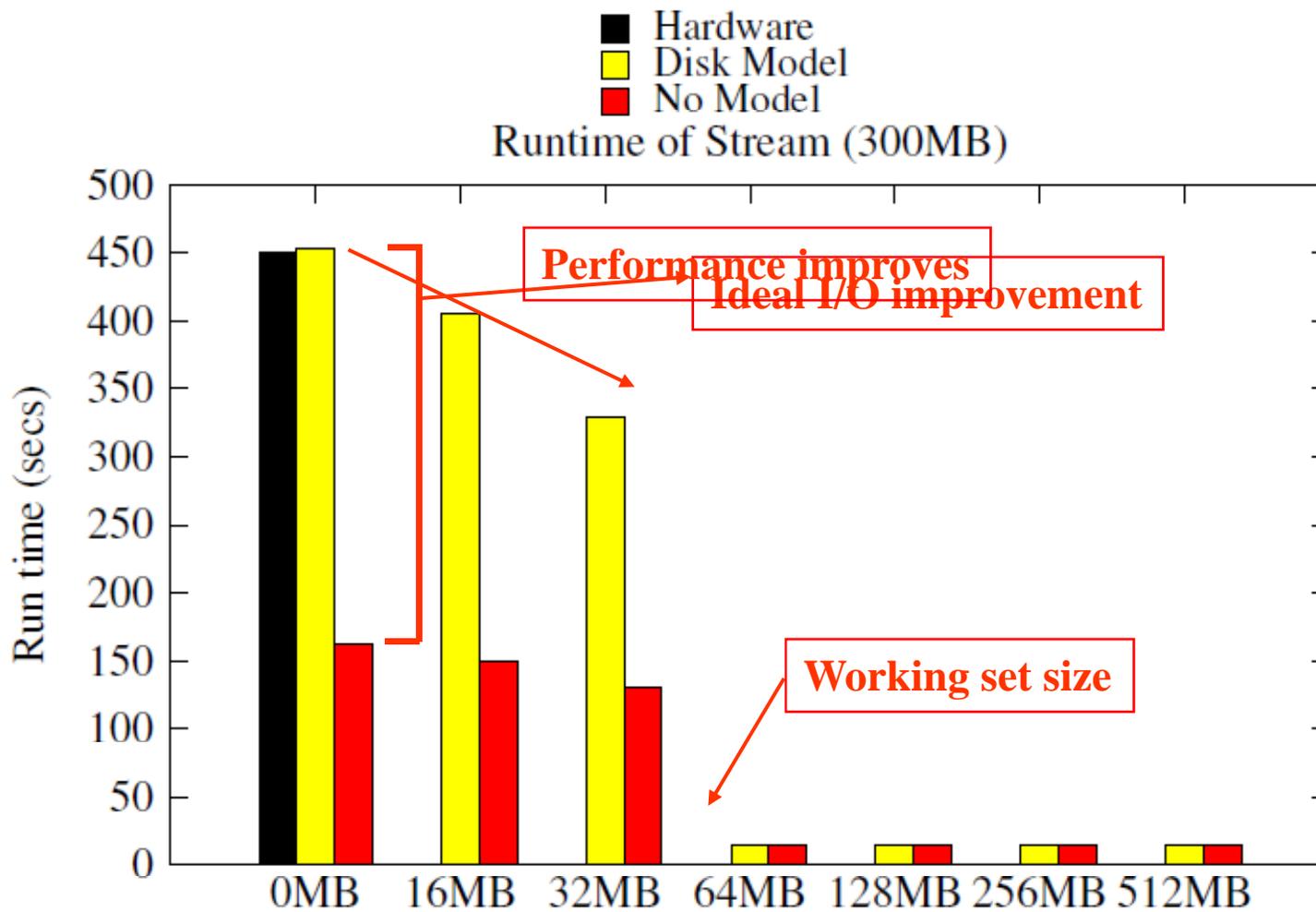
Necessary SymCall: `query_vaddr()`

1. Get current process ID
 - `get_current();` (Internal Linux API)
 2. Determine presence in Linux swap cache
 - `find_get_page();` (Internal Linux API)
 3. Determine page permissions for virtual address
 - `find_vma();` (Internal Linux API)
- **Information extremely hard to get otherwise**
 - **Must be collected while exit is being handled**

Evaluation

- Memory system microbenchmarks
 - Stream, GUPS, ECT Memperf
 - Configured to overcommit anonymous memory
 - Cause thrashing in the guest OS
- Overhead isolated to swap subsystem
 - Ideal swap device implemented as RAM disk
 - I/O occurs at main memory speeds
 - Provides lower bound for performance gains

Stream Runtime



Future Work (short term)

- Continue exploring virtualization in HPC
 - UNM and Sandia collaboration
 - Granted 5 million hours on Jaguar
 - **Current fastest supercomputer in the world**



Oak Ridge National Labs

Cray XT5

224,256 cores

4352 sq. ft

6.95 MegaWatts

\$104 million

- Continue exploring Symbiotic interfaces
 - Symbiotic modules

Future Work (long term)

- **Large scale and heterogeneous multicore architectures**
 - Virtualization as a basic building block
 - How to use massively multicore architectures
- **Symbiotic Virtualization**
 - Leverage symbiotic interfaces
- **Future HPC architectures**
 - Continue OS research in HPC
- **User driven adaptation and optimization**
 - Direct user input to drive optimizations

Backup Slides

Restrictions

- Currently designed for short queries
 - Narrow focus allows behavioral guarantees
- Restrictions:
 - Only 1 symcall active at a time
 - Symcalls run to completion
 - No blocking
 - Blocking would allow asynchronous behavior
 - No context switches
 - No injected exceptions or interrupts
 - Symcalls cannot wait on locks
 - Waiting would make deadlocks possible

Jaguar



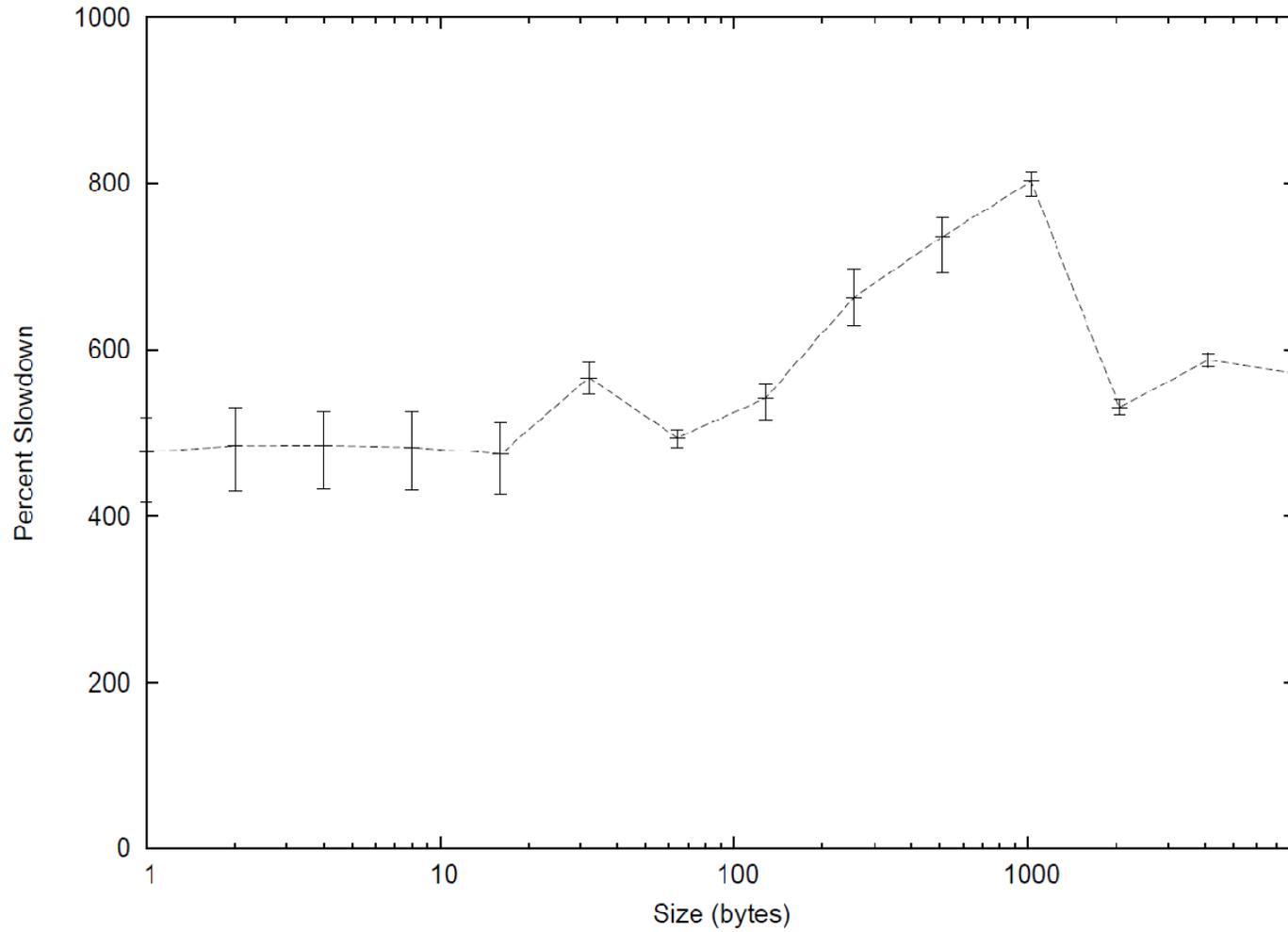
Oak Ridge National Labs

Jaguar Info

- #1 on Top500
- Cray XT5
- Hex core AMD Opterons
- 224,256 cores across 18,688 nodes
 - 2,332 TFlops
- 4352 sq. ft
- 6.95 MegaWatts
- \$104 million

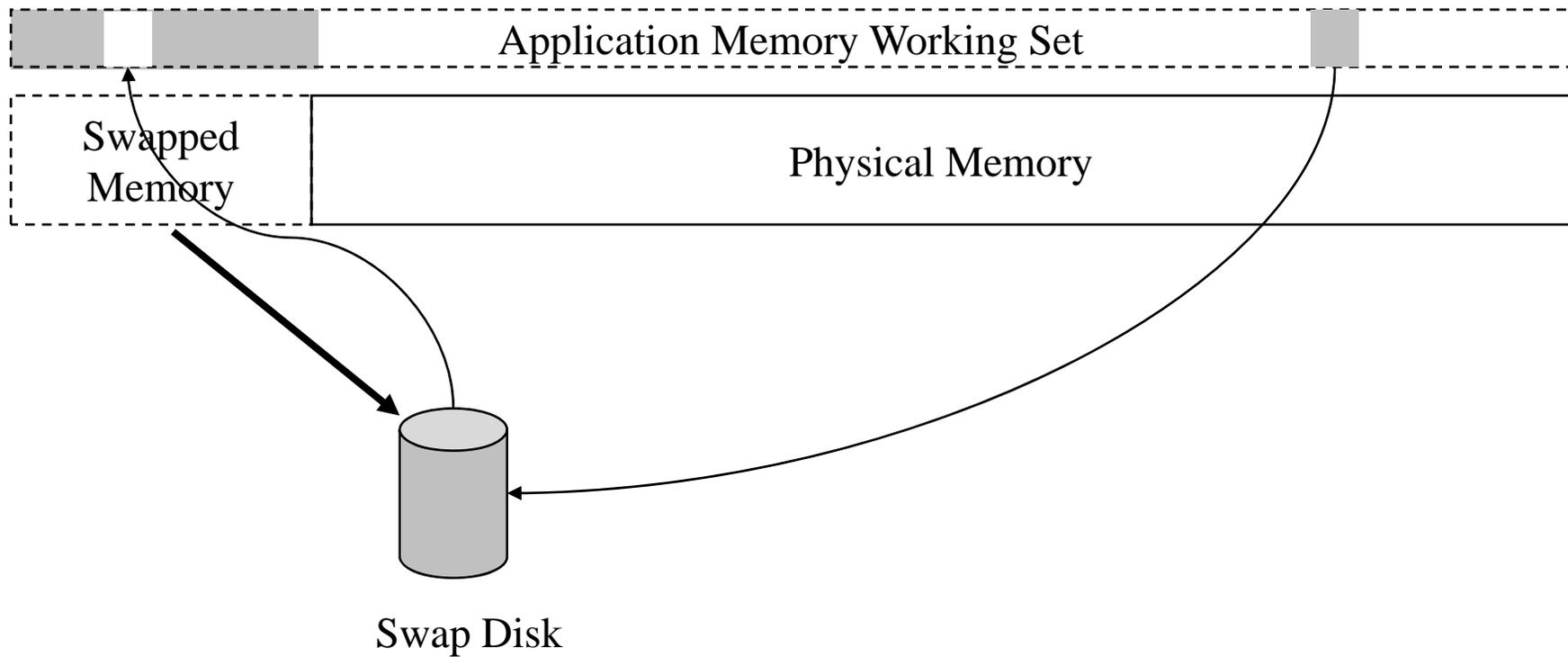
Jaguar Specifications	XT5	XT4
Peak Teraflops	2,332	263
Six-Core AMD Opterons	37,376	
Quad-Core AMD Opterons		7,832
AMD Opteron Cores	224,256	31,328
Compute Nodes	18,688	7,832
Memory (TB)	299	62
Memory Bandwidth (GB/s)	478	100
Disk Space (TB)	10,000	750
Interconnect Bandwidth	374	157
Floor Space (ft ²)	4,352	1,344
Cooling Technology	Liquid	Air

Noise effects, cont'd



(b) MPI_Bcast

Swapping



Symbiotic Virtualization in HPC

- HPC environments are well suited to symbiotic techniques
- Full trust of the software stack
 - Fewer security concerns
- Specific hardware configurations
 - Limited number of devices
- Environments are much smaller
 - Internal OS state is simpler than a general purpose OS
- At large scale performance impact is dramatic
 - Large impetus to optimize VMM and OS

Market prevalence

- Springboard Research: 2010 forecast for Asia/Pacific
 - \$1 billion for virtualization services
 - \$350 million for virtualization products
- 451 Group: US Market
 - Expected compound annual growth of 36%.
 - Through 2013

RedStorm Info

- #7 on Top500 in 2009, now
- Cray XT3
 - Quad and Dual core AMD Opterons
 - Custom SeaStar RDMA interconnect
- 38208 cores across 12,960 nodes
 - 284.16 TFlops

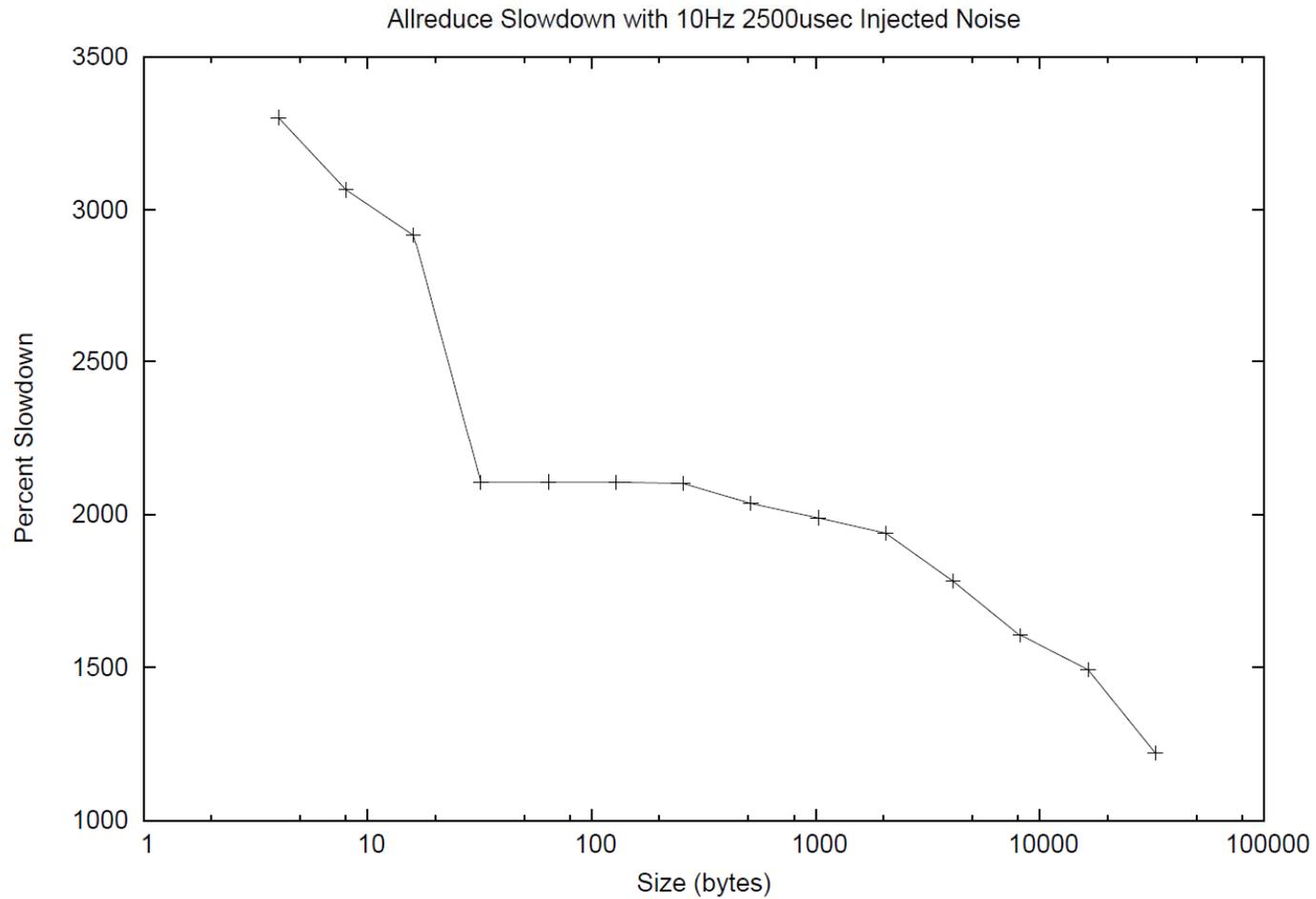
Performance is key

- Applications are tightly coupled and parallel
 - Lots of inter-node communication
- Capability machines
 - Entire machine is used for a single application
- Massive scale
 - 200,000 node reduction trees...

No really, Linux doesn't work

- Work by UNM collaborators:
 - Kurt B. Ferreira, Ron Brightwell, and Patrick G. Bridges. **Characterizing application sensitivity to OS interference using kernel-level noise injection**. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (Supercomputing'08)*, November 2008.
- Example: kernel threads
 - MPI operations at fairly small scale
 - 128 nodes
 - Periodic noise injection
 - 2500 usec
 - 10HZ
 - 2.5% overhead

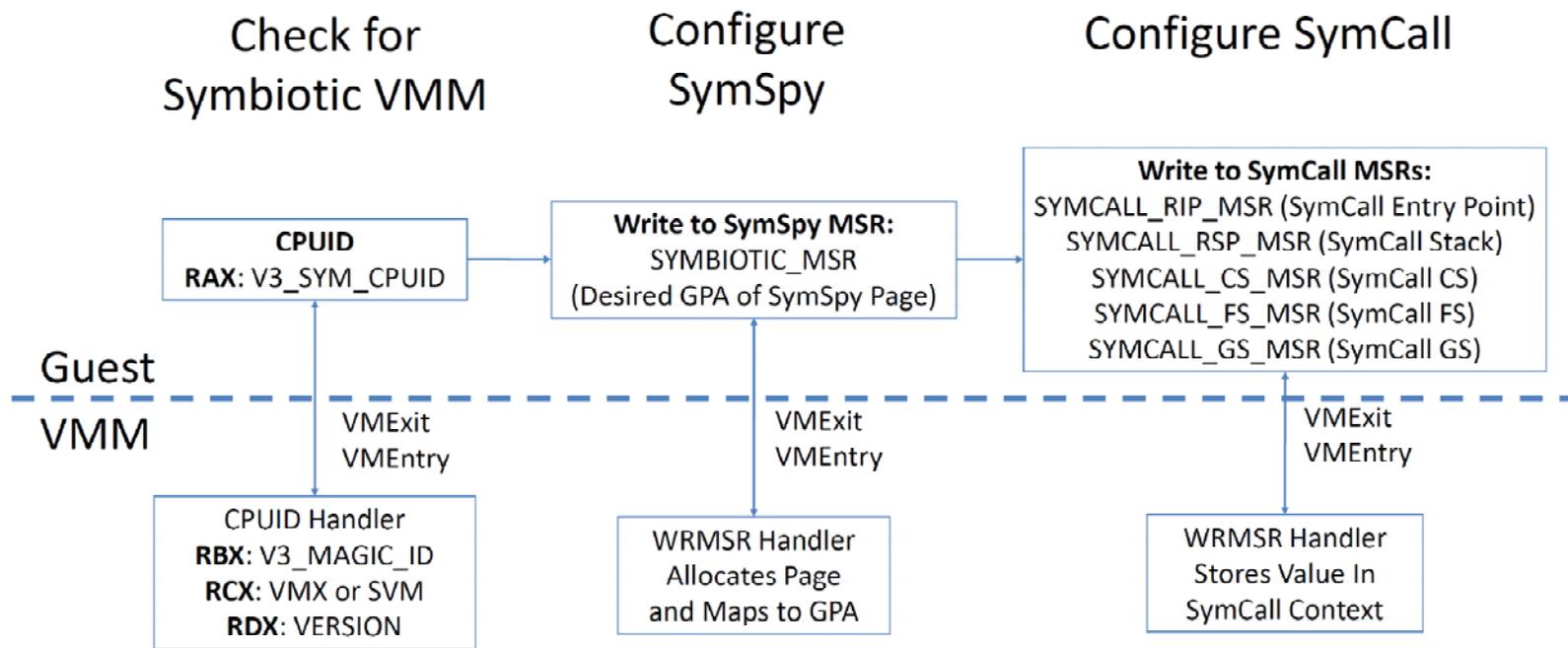
Noise effects



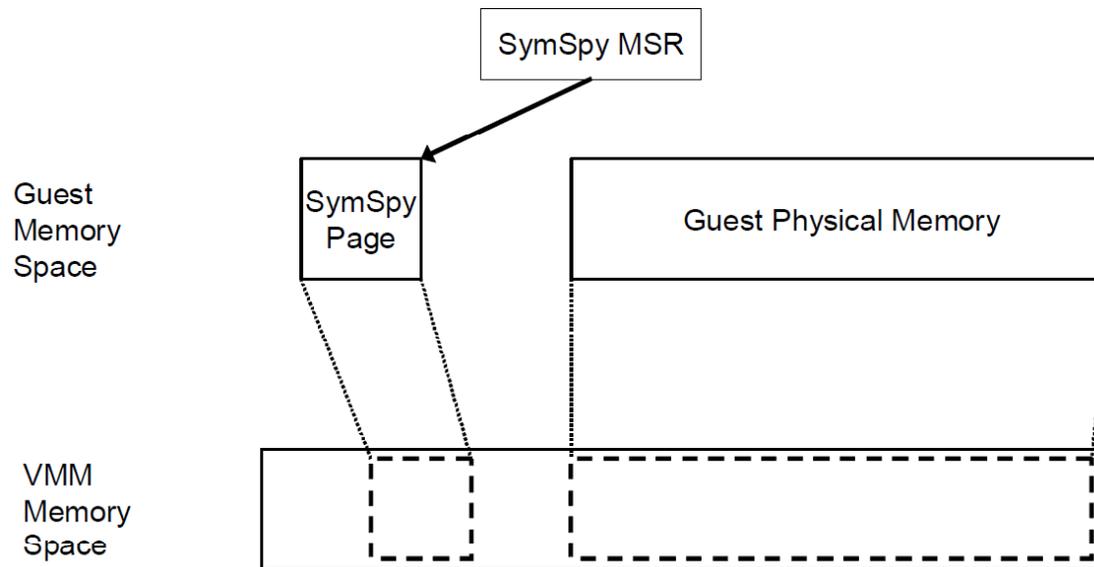
(a) MPI_Allreduce

Symbiotic Discovery

- A symbiotic OS must run on real hardware
 - Interface must be based on hardware features



SymSpy



- **Shared memory page between OS and VMM**
 - OS uses MSR to map into address space
- **Standardized data structures**
 - Shared state information
- **Read and write without exits**

SymCall environment

- Specified in virtual SYMCALL MSR_s
 - VMM copies values into hardware control registers
- SYMCALL MSR_s
 - SYMCALL_RIP
 - Global code entry point for all symcalls
 - SYMCALL_RSP
 - Stack frame used by SymCall
 - SYMCALL_CS (+SS)
 - Code segment and Stack segment used for SymCall
 - Enables kernel mode execution
 - SYMCALL_GS and SYMCALL_FS
 - Special segments that point to per CPU data structures

System Calls

- Entry points into Kernel
 - Provides privileged services for applications
 - Device IO, address space modifications, etc...
- Extensive hardware support
 - Fast entry to preconfigured execution state
 - Switch from user mode to kernel mode
 - Kernel specifies execution environment to hardware
 - Model Specific Registers (MSRs)
 - Special instructions for kernel entry/exit
 - SYSENTER and SYSEXIT
 - SYSCALL and SYSRET

Palacios as an HPC VMM

- Minimalist interface:
 - Does not require extensive host OS features
 - Easily embedded into even small kernels
- Full system virtualization:
 - Runs existing kernels without any porting
 - Linux, Kitten, Catamount, Cray CNL, and IBM's CNK
- Contiguous memory preallocation:
 - Preallocates guest memory as a physically contiguous region
 - Simplified implementation and deterministic performance for memory operations
- Passthrough resources and resource partitioning:
 - Host resources are easily mapped directly into a guest environment
 - Provides access to high performance devices, with existing device drivers, with no virtualization overhead.
- Low noise:
 - Minimizes the amount of OS noise injected by the VMM layer.
 - No internal timers and no accumulated deferred work.
- Compile and runtime configurability
 - Compile a VMM tailored to specific environments
 - Extensive ability to adapt VMM behavior at runtime

Takeaway

- 2.5% local overhead → 5X–35X slowdown at scale
 - For only 128 nodes
- These are large tightly coupled systems
 - Must be thought of as a single system
- Specialized Operating Systems are necessary
 - Lightweight kernels

Why does Virtual Paging Matter?

- Different approaches have very different architectures
- Guest behavior during context switch
 - CNL swaps page tables
 - Catamount does not, just invalidates pages
- Shadow paging
 - Software emulation
 - VMM emulates guest page tables
 - Page table changes are expensive
- Nested paging
 - Hardware support
 - 2 sets of hardware page tables
 - TLB misses are expensive

Symbiotic Modules

- Currently in development
- Runtime loading of modules from the VMM
 - Run VMM code in guest context
 - Create new Symbiotic interfaces at runtime
- Linux already has module support
 - Load from user space
 - Extension to allow loading from VMM
 - No user intervention necessary

Device Drivers

- Guests often need direct device access
 - High performance networks
 - Driver included inside guest OS
- Self-virtualization
 - Devices still require their own drivers
 - Not all devices are capable
- Does not map well to virtual environments
 - Migration changes underlying hardware
 - Difficult to share between multiple VMs
 - VMM must fully trust guest driver

Symbiotic Device Drivers

- VMM provides passthrough driver to guest as a module
- Guest OS no longer needs to include full set of drivers for all possible hardware
- VMM can optimize driver behavior to the environment
- Drivers can be dynamically swapped as conditions change
 - Passthrough network driver
 - Overlay network driver
 - Paravirtual driver

Palacios Details

- Full hardware virtualization
 - Intel and AMD virtualization architectures
- Supports 32 and 64 bit environments
 - Host and Guest
- Supports Linux and HPC guest OSes

