TECHNISCHE UNIVERSITÄT DRESDEN

Department of Computer Science – Institute for Systems Architecture – Operating Systems Group

NOVA: A Microhypervisor-Based Secure Virtualization Architecture

Udo Steinberg, Bernhard Kauer

Motivation

- Virtualization widely used for consolidation of workloads
- Attackers have begun targeting the virtualization layer
  - Xen VM escape[1]
  - VMware VM escape[2]
- Alarming prediction
  - „60% of virtual servers will be less secure than the physical servers they replace through 2012“[3]

[1] Rutkowska/Wojtczuk: Xen 0wning Trilogy, Blackhat 2008

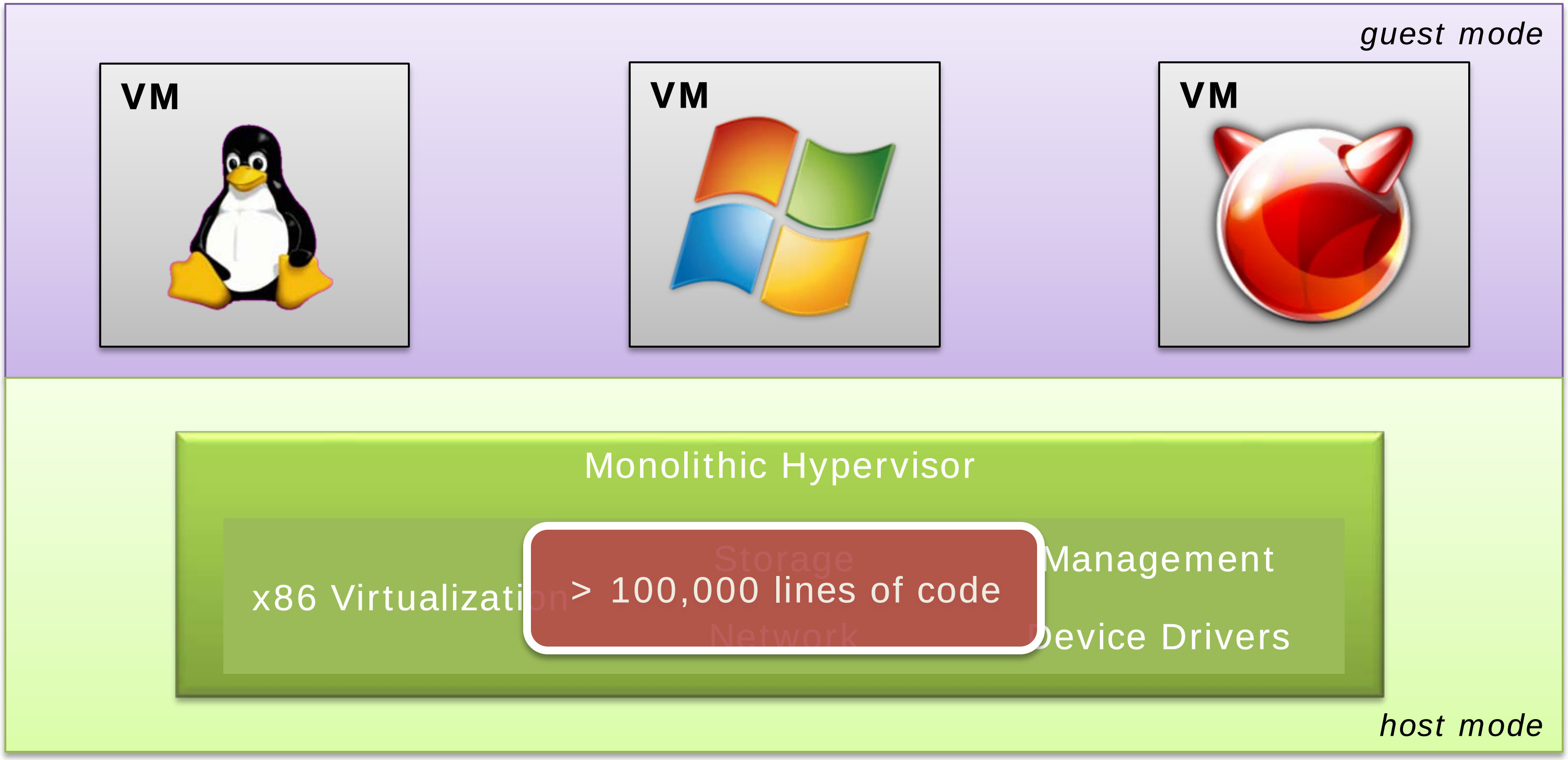
[2] Kortchinsky: Cloudburst - Hacking 3D and Breaking out of VMware, Blackhat 2009

[3] Gartner Inc., Press Release March 15 2010

Security Risks in Virtual Environments

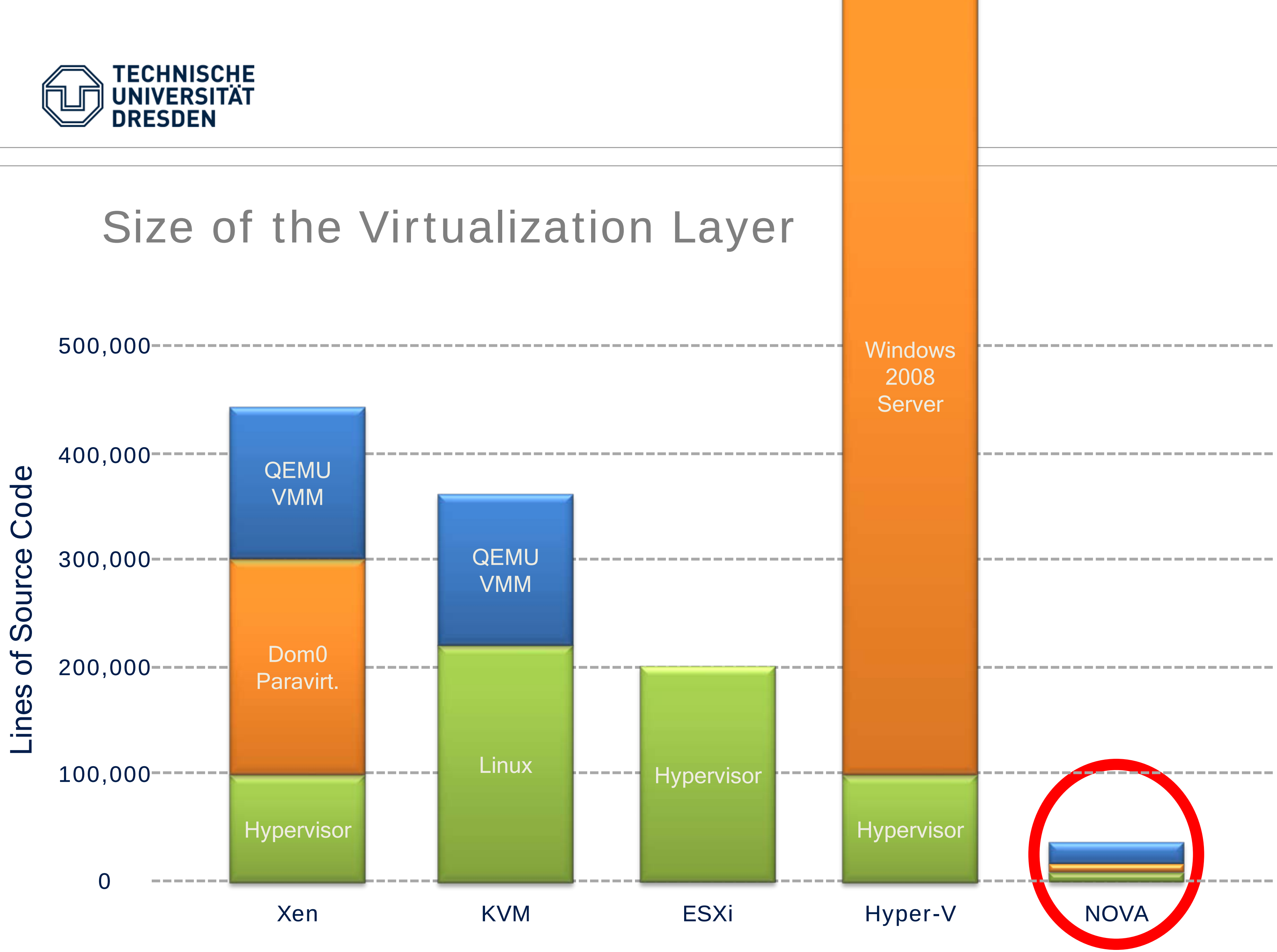
- New layer of software underneath hosted workloads
  - can contain exploitable vulnerabilities
  - must be configured and maintained
- Breaking into the hypervisor
  - compromises all hosted workloads at once
  - facilitates attacks from below the guest OS kernel
- Consolidation of workloads with different trust levels
  - requires strong separation

State of the Art: Monolithic Hypervisors



Monolithic hypervisor is single point of failure

Size of the Virtualization Layer



Improving the Status Quo

Virtualization layer is critical. Make it as small as possible.

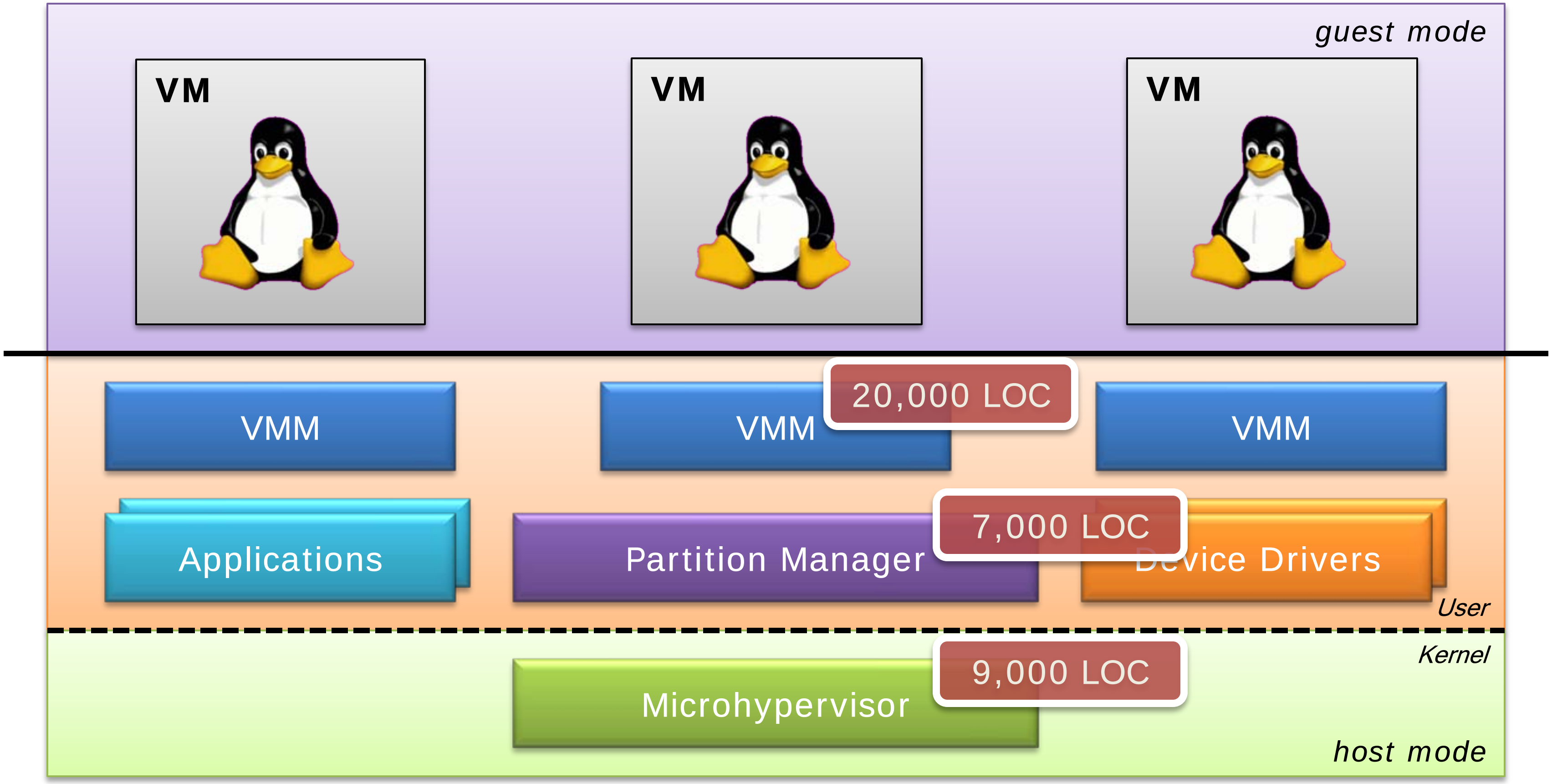
Design Principles:

1) Fine-grained functional decomposition
   - Microhypervisor (privileged)
   - Multiple user-level VMMs (unprivileged)
   - User-level drivers, applications (unprivileged)

2) Principle of least privilege among all components
   - Capability-based authorization model

Ideas adopted from the microkernel world

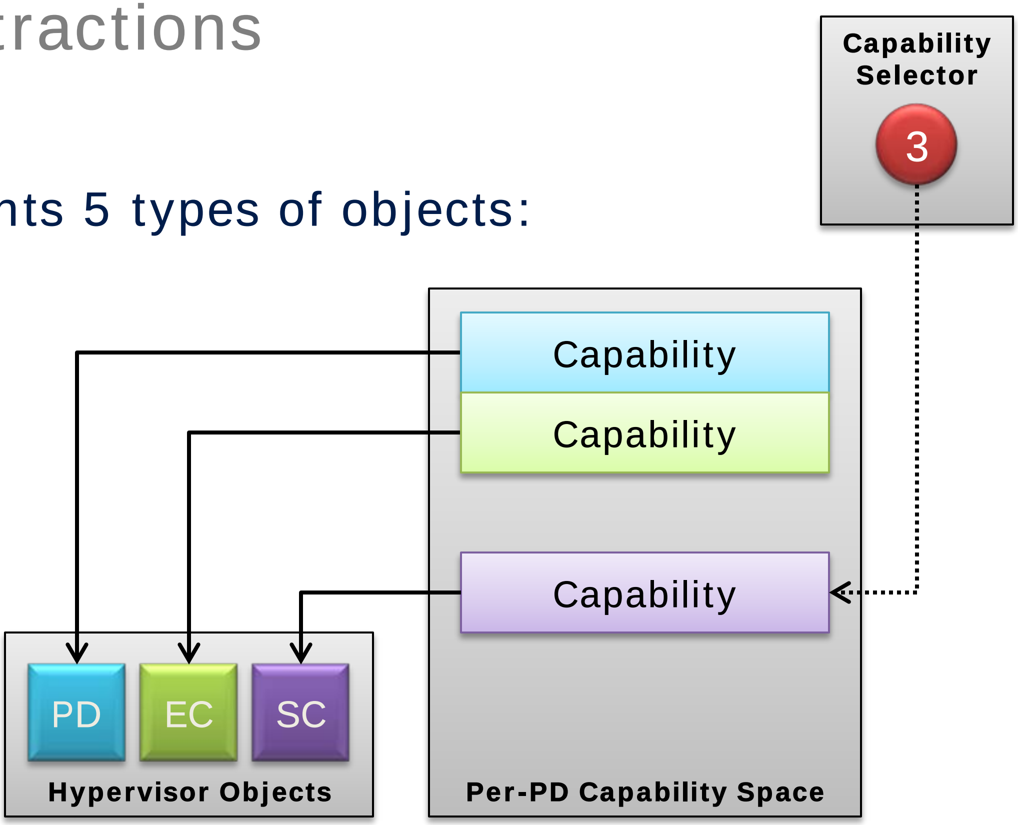
NOVA OS Virtualization Architecture



Microhypervisor Abstractions

Microhypervisor implements 5 types of objects:

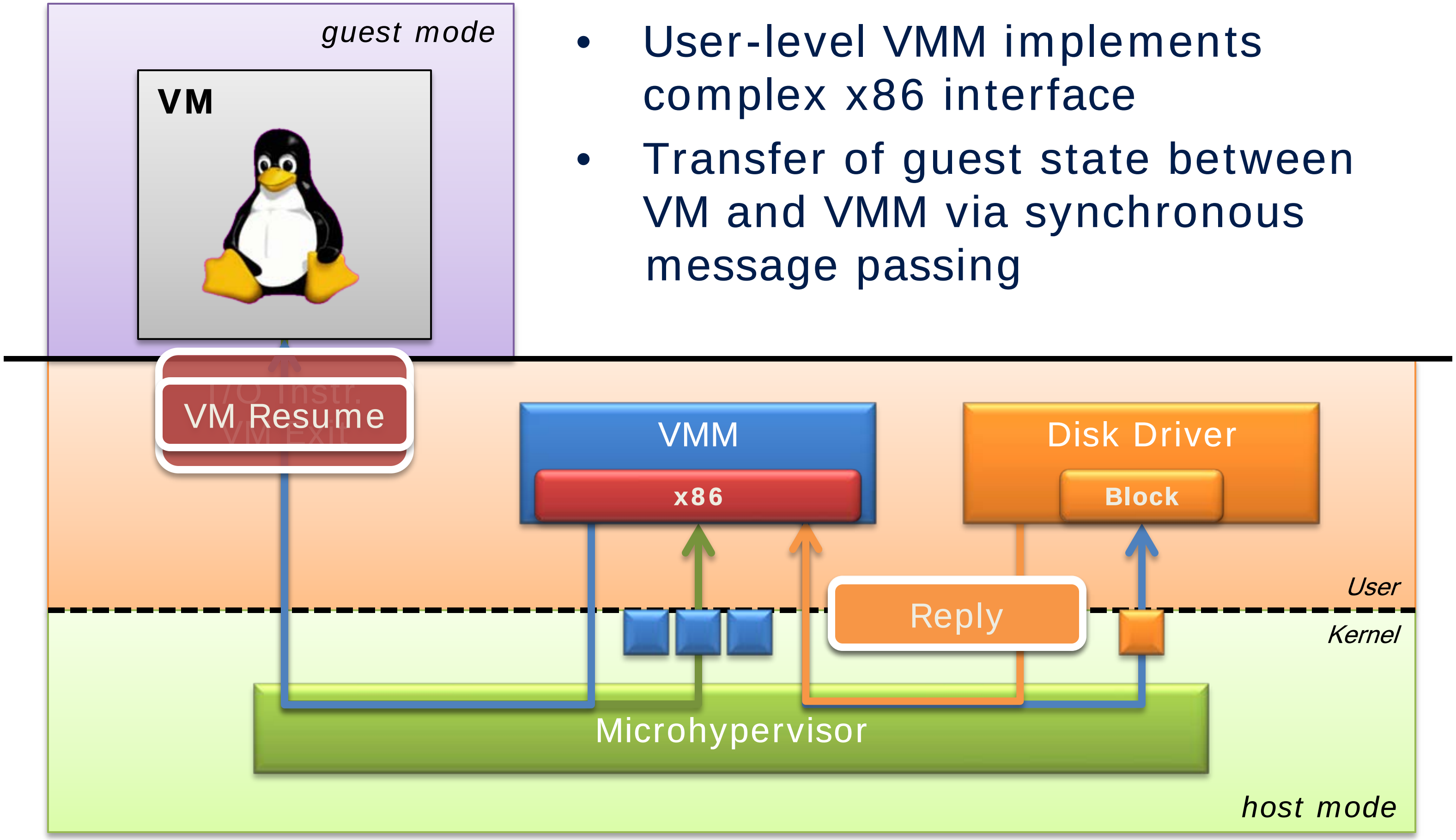
- Protection Domain
- Execution Context
- Scheduling Context
- Portal
- Semaphore



Hypercall interface uses capabilities for all operations.

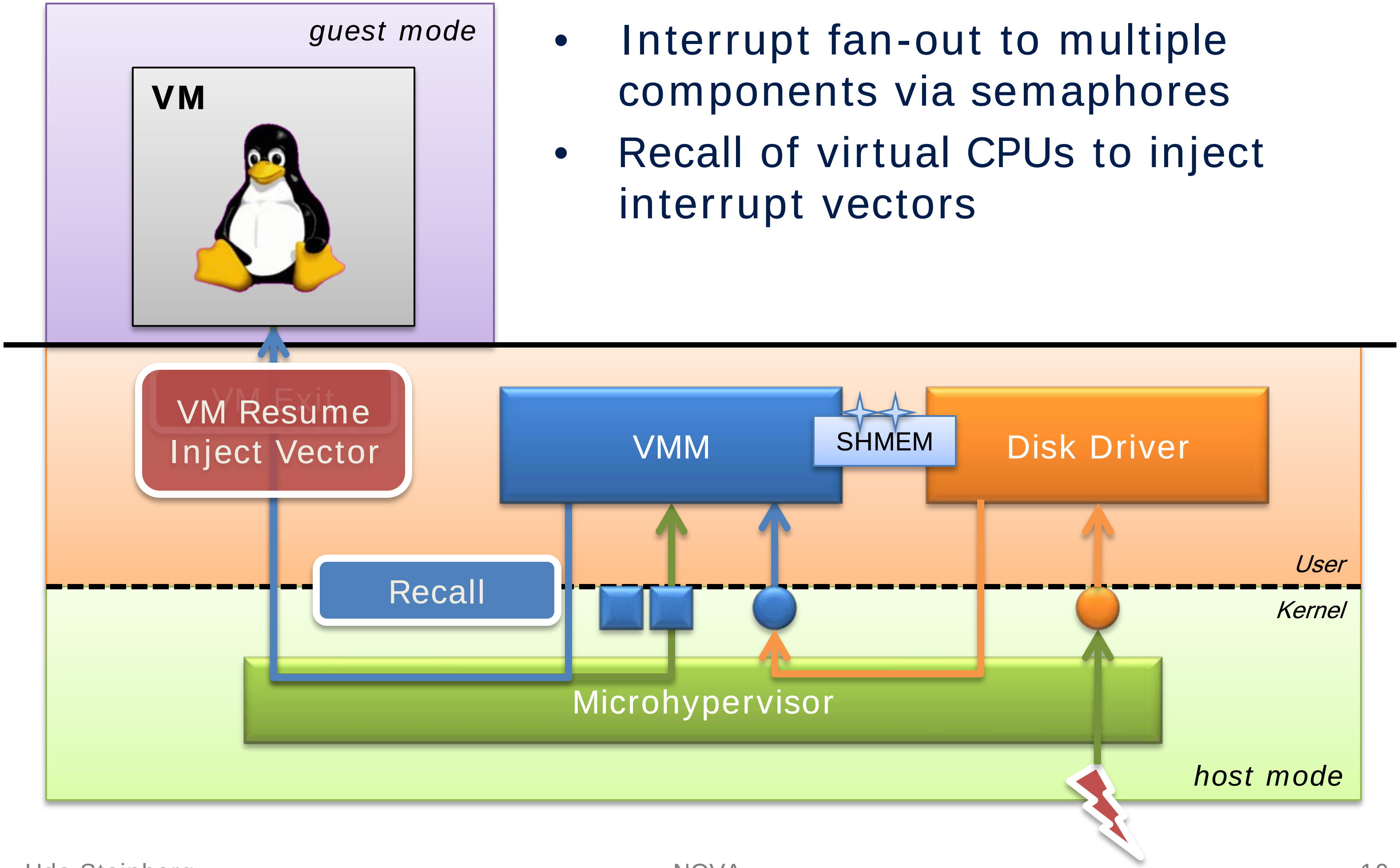
Handling of Virtualization Events

- User-level VMM implements complex x86 interface
- Transfer of guest state between VM and VMM via synchronous message passing



Interrupt Delivery

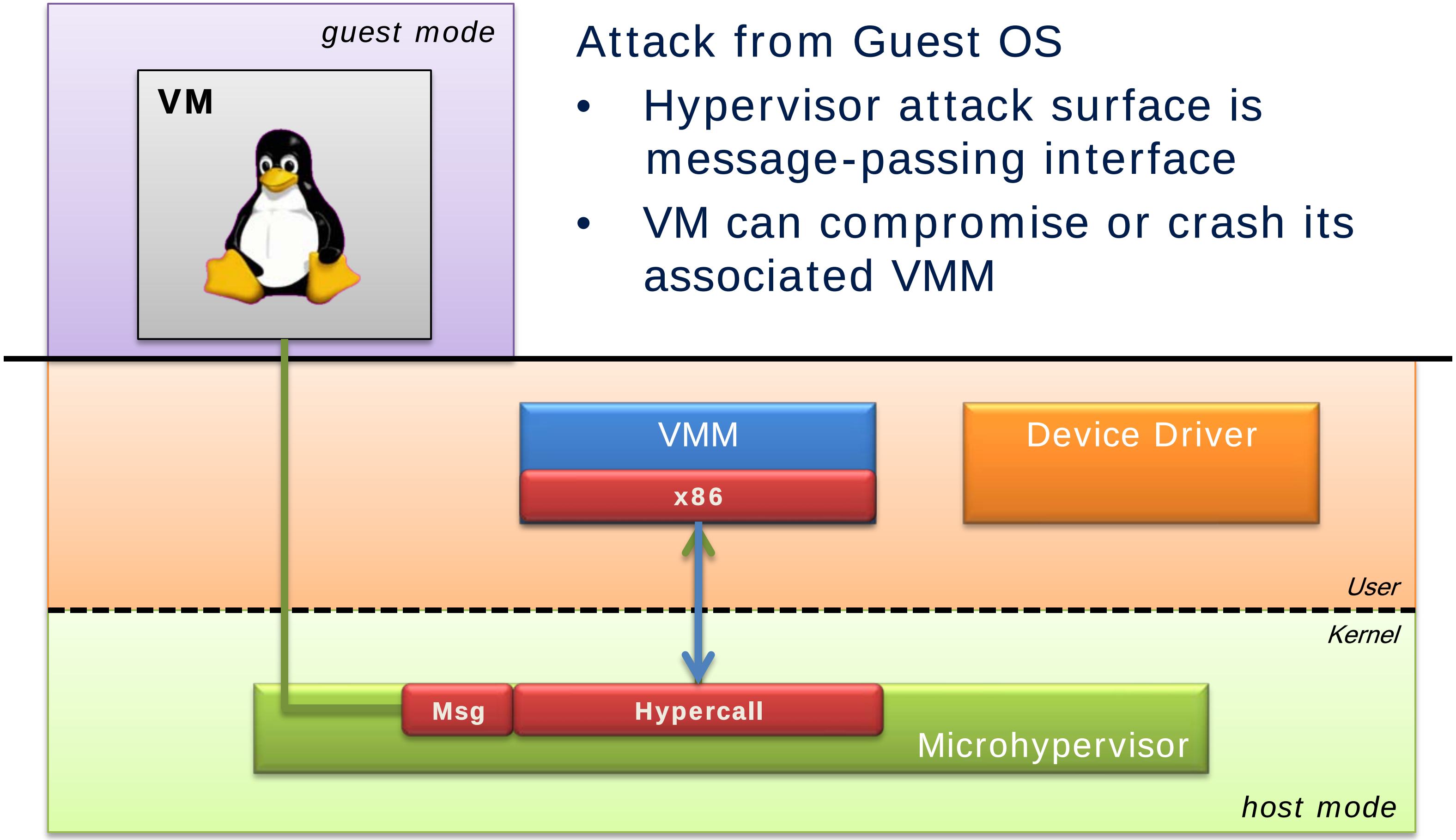
- Interrupt fan-out to multiple components via semaphores
- Recall of virtual CPUs to inject interrupt vectors



Impact of Attacks in NOVA

Attack from Guest OS

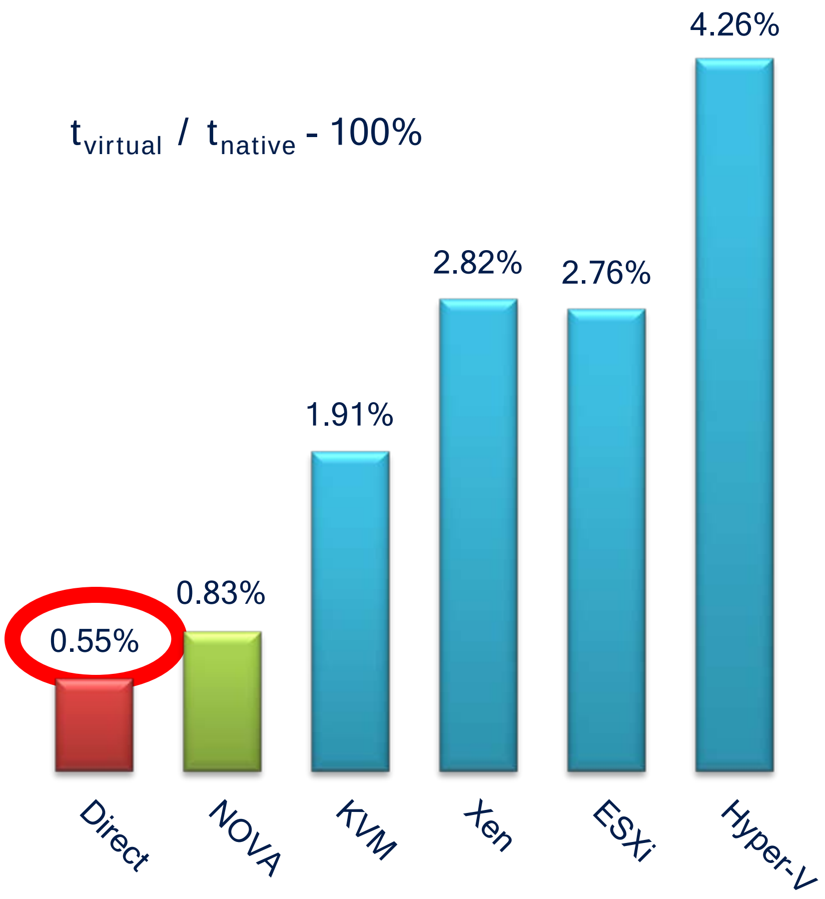
- Hypervisor attack surface is message-passing interface
- VM can compromise or crash its associated VMM



Virtualization Interface: Lessons Learned

- One simple communication mechanism
  - Fast synchronous IPC with hand-off scheduling
  - Selective transfer of execution state
  - HV need not care about x86 virtualization details
- One synchronization mechanism
  - Counting semaphores
  - Also used for interrupt delivery
- Unified abstractions
  - Protection Domain = Virtual machine or User Task
  - Execution Context = Virtual CPU or Thread

Virtualization Overhead



Kernel-Compile Benchmark:

CPU: Intel Core i7 2.67 GHz

VM Configuration:

Single virtual CPU, virtual disk
512 MB Guest Memory, EPT+VPID

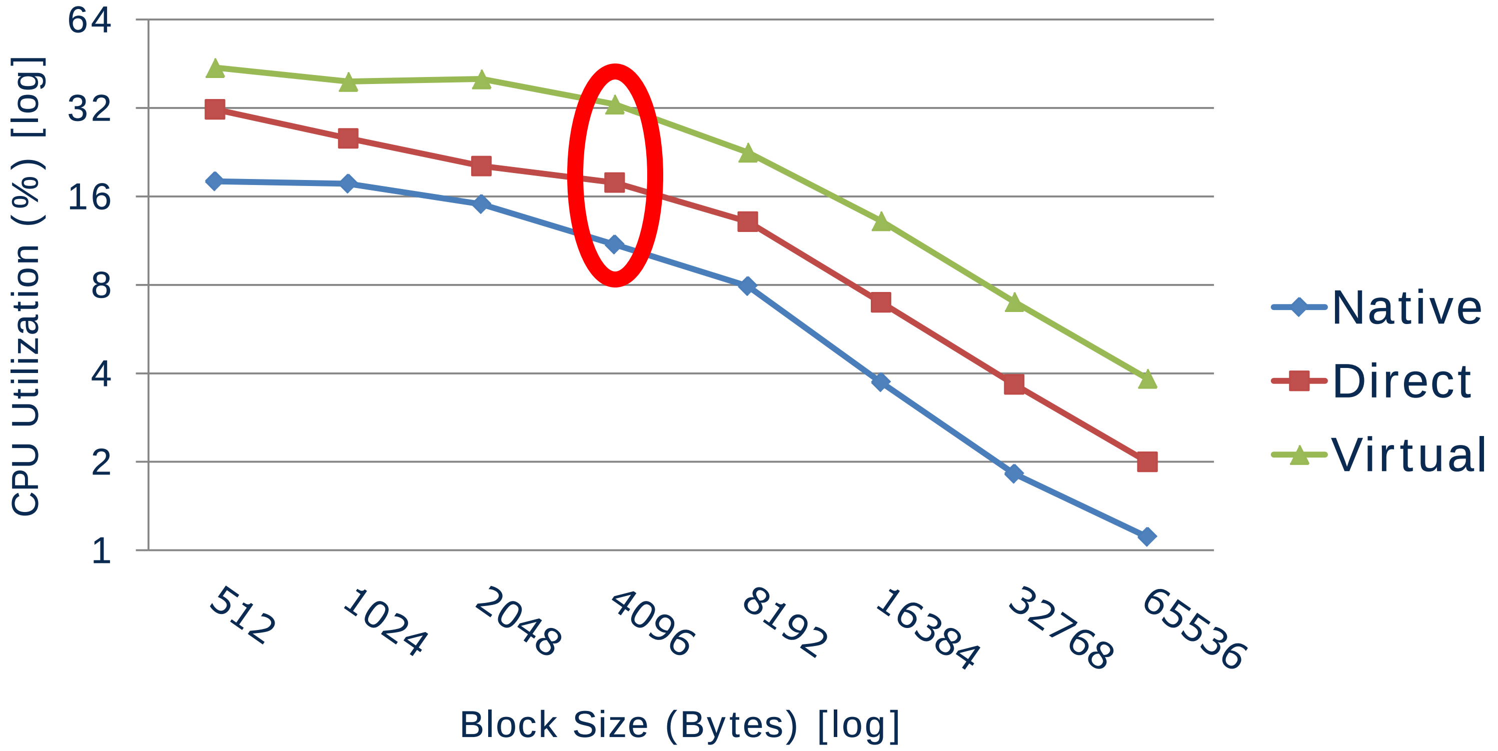
Direct Assignment:

0.55% performance overhead
caused by nested paging

NOVA:

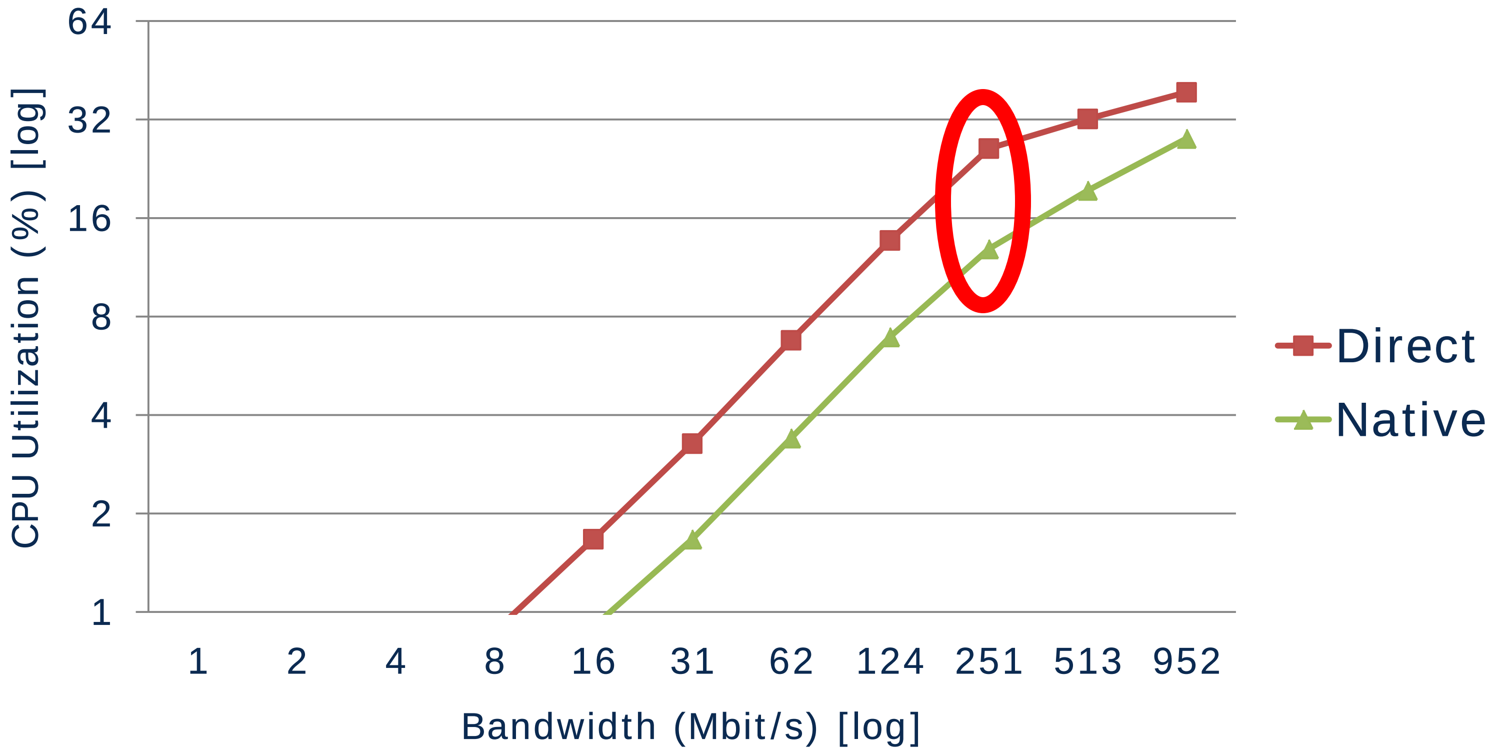
Additional 0.3% overhead
~3900 cycles/exit

I/O Virtualization Overhead (AHCI controller)



Stream of sequential disk reads with increasing block sizes

I/O Virtualization Overhead (e1000 NIC)



Receive UDP packet stream with increasing bandwidth

Current Status

- Hypervisor
  - Runs on Intel VT-x and AMD-V
  - Supports SMP, Nested Paging, VT-d IOMMU
- User-Level Virtual-Machine Monitor
  - Implements virtual PCI, SATA, NIC, BIOS, ...
  - Supports PCI Pass-Through (direct assignment)
- Ongoing work
  - Windows as Guest OS
  - SR-IOV Devices

Conclusion

- Decomposed virtualization layer provides additional isolation boundaries at the cost of more context switches

- Lower context-switch overhead resulting from simple code paths and selective state transfer

NOVA achievements:

- **TCB reduction by an order of magnitude**
- **Performance improvement over monolithic hypervisors**

Code available under GPLv2: http://www.hypervisor.org