

# Breaking Up is Hard to Do: Security and Functionality in a Commodity Hypervisor

## Authors

Department of Computer Computer Science, University of British Columbia

- Patrick Colp
- Mihi Nanavati
- William Aiello
- Andrew Warfield

Citrix Systems R&D

- Jun Zhu
- Tim Deegan

National Security Agency

- George Coker
- Peter Loscocco

## Presenters

- Scott Kellish
- Jesse Campbell

# Presentation Outline

- Problem Space
- Xen Platform
- TCB (Trusted Computing Base)
- XOAR Goals
- XOAR Architecture Overview
- XOAR Design Details
- XOAR Security Evaluation
- XOAR Performance
- Conclusions

# Problem Space

Cloud computing virtualization involves leasing data center servers to individuals in multi-tenant environments, i.e. many clients share each server.

It's important to prevent “breaches of isolation” where one client directly or indirectly affects the experience of another on the same server.

- Indirect
  - ex: causing the server to run slowly for the other clients
- Direct
  - hacking the hypervisor from a virtual machine (VM)
    - ex: accessing files and memory, or running programs on others' VMs

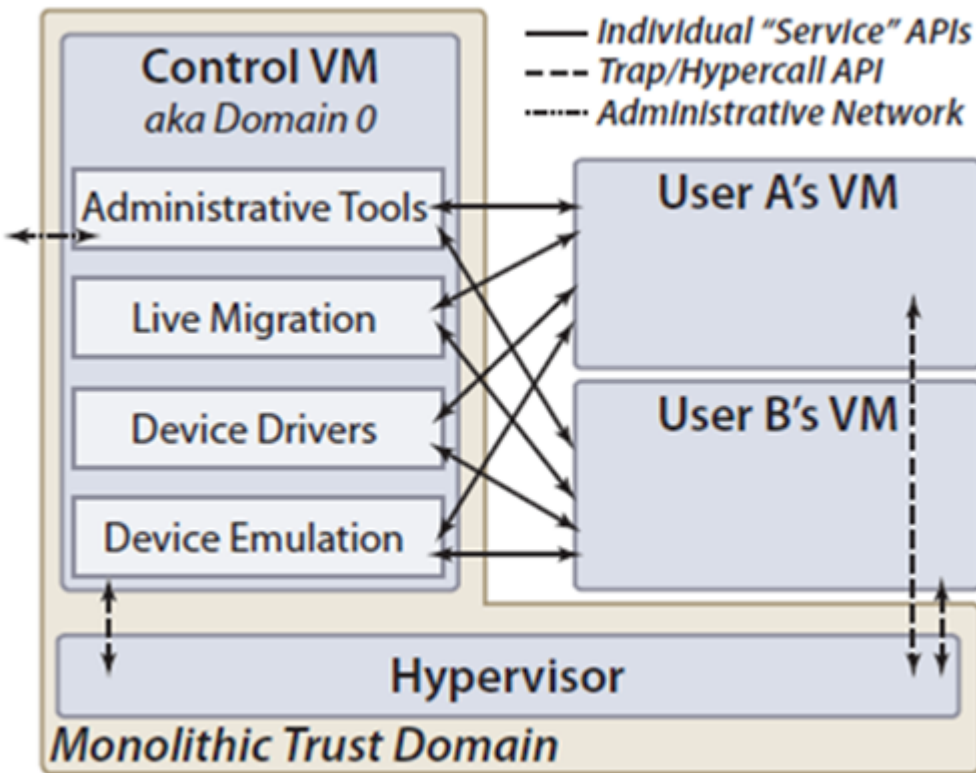


# The Xen Platform

- Type-1 Hypervisor
- Device Drivers
  - delegated to Dom0 which exposes to guest VM's
  - devices may be virtualized , passed through, or emulated
  - unmodified OSes run on VMs through Qemu
- XenStore
  - key-value store; a system-wide registry and naming service
  - Most critical component
    - Vulnerable to DoS attacks, Performs most admin operations
- Toolstack
  - provides administrative functions for management of VMs
- System Boot
  - the hypervisor creates Dom0
    - initializes hardware, devices, and back-end drivers
  - XenStore is loaded before guest VMs

# Trusted Compute Block (TCB)

Defined as “the totality of protection mechanisms within a computer system -- including hardware, firmware, and software -- the combination of which is responsible for enforcing a security policy” -- Source wikipedia



Xen Hypervisor + Control VM provide:

- protection mechanisms that enforce a security policy
- responsible for guest VM
  - isolation
  - scheduling
  - memory management
- hardware management
- device emulation
- inter-VM communication
- virtual consoles
- configuration state management

# Hypervisor Attack Vectors<sup>1</sup>

- 44 vulnerabilities for Type-1 Hypervisors,
- 23 XEN Vulnerabilities (by type):

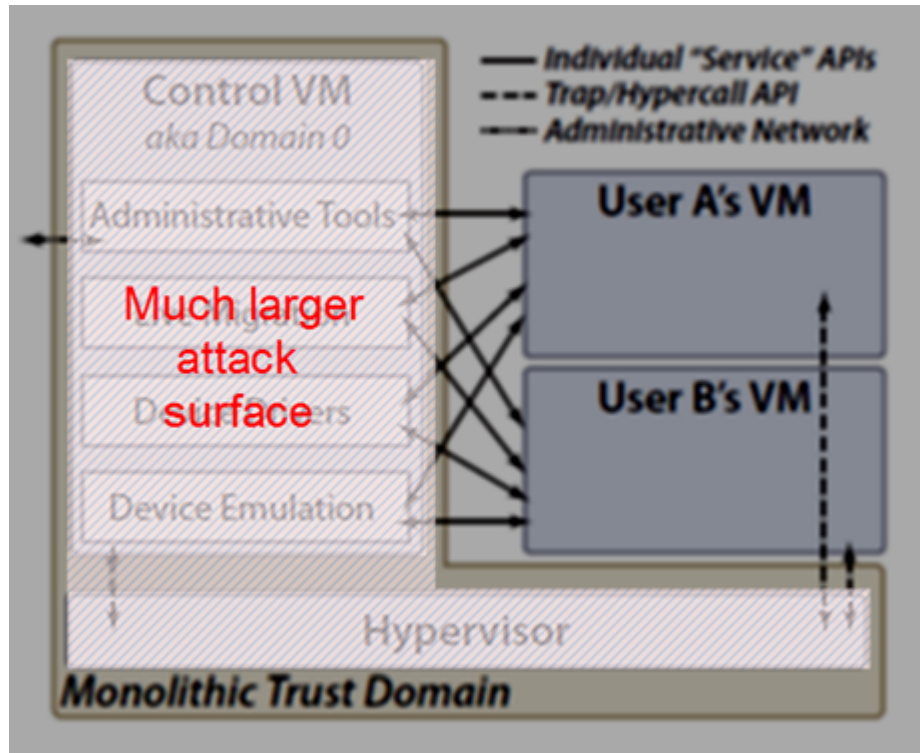
14	Device emulation layer
2	Virtualized Device Layer
5	Management components
2	Hypervisor Exploits

- 21 of 23 attacks targeted service components in the control VM
- Threat Model:
  - Assume professionally/well managed
  - Assume hypervisor is trusted but the Control VM *will* contain bugs
  - Attacker: guest VM looking to violate security of another guest VM
  - Goal: isolate components so single attack is not sufficient

<sup>1</sup>Source: CERT vulnerability database and VMWare's list of security advisories

# Xoar: Architecture Goals

- Remain transparent to existing VM interfaces
- Tight control of privileges
- Minimize interface of all components to attack footprint
- Eliminate sharing or make it explicit to allow meaningful auditing and logging
- Limit time windows when system components run to reduce attack opportunity



- ❑ Don't reduce functionality, performance or maintainability



# About Xoar

- **Xoar** is a modification to Xen
  - Dom0 disaggregation (division and separation)
    - adds modularity and isolation
    - divides the control VM into a single-purpose components
      - called: Service VMs
- Xen's Domain0 (a.k.a, Dom0)
  - Host OS
    - accesses the hardware directly
  - Hypervisor
    - service console that controls VMs
  - Monolithic Trusted Compute Block (TCB)
    - one single, unified operating system





# Xoar allows

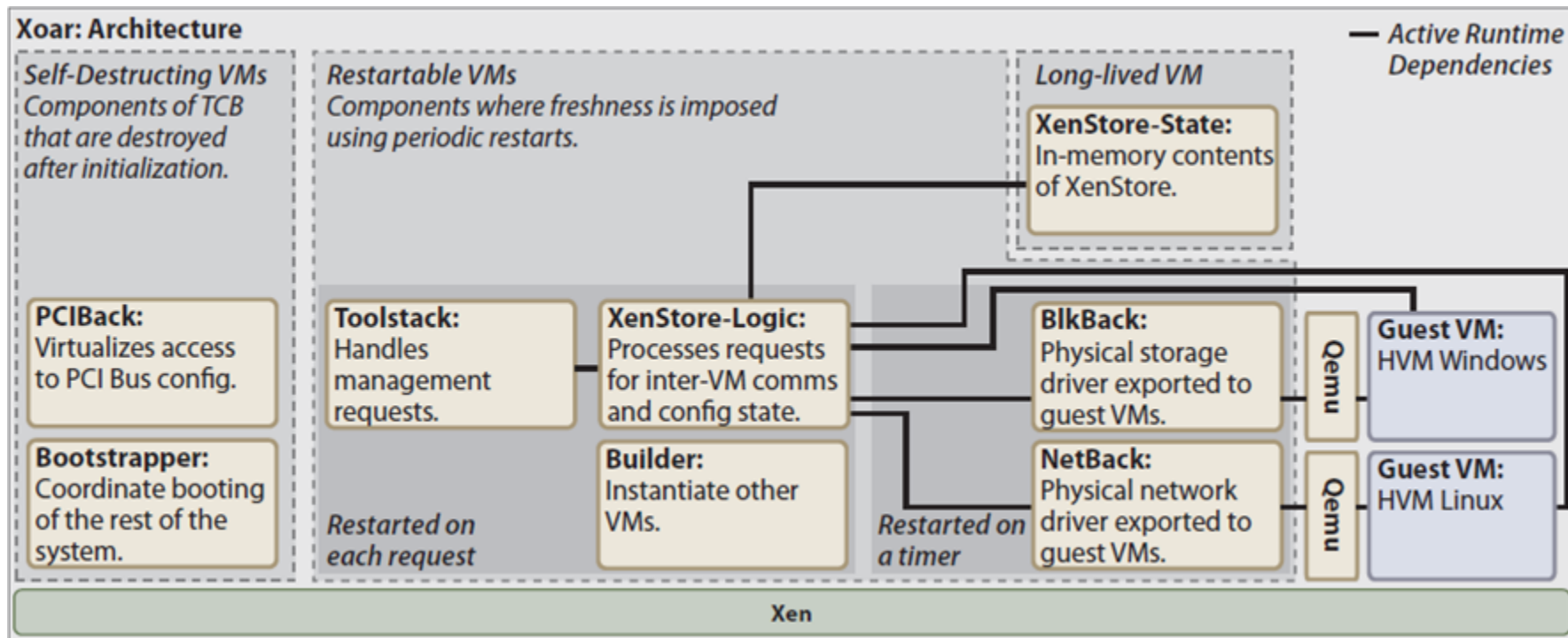
- **Xoar** allows:
  - configurable and auditable sharing of components
  - micro-rebooting
    - small, quick subsystem reboots
    - blocks some time-based attack methods
      - such as: a buffer overflow attack
        - exceed a host's memory limit, repeatedly attempt to execute code, corrupt the host OS and run malware
      - will possibly violate:
        - confidentiality
        - integrity
        - availability



# Xoar Contributions

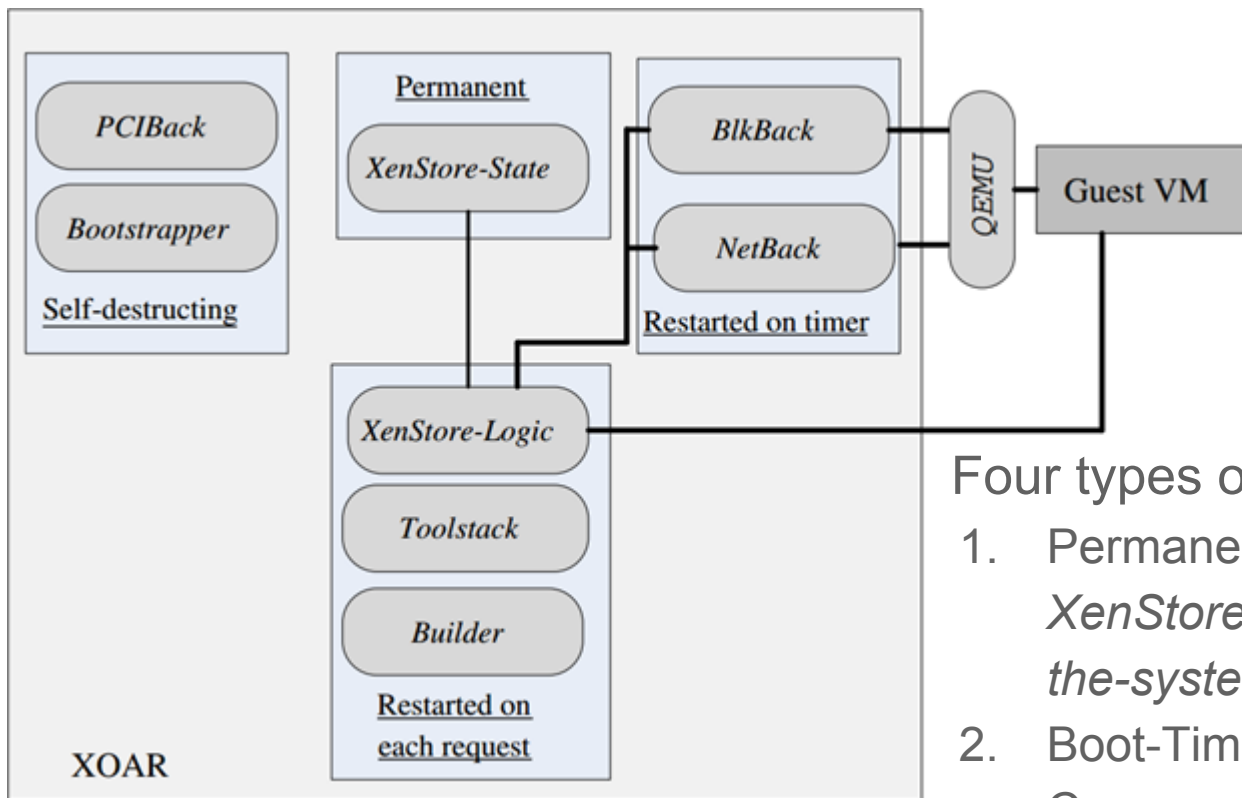
- Disposable Bootstrap
  - destroys the VM that booted the computer when booting the physical computer
    - runs complex, privileged code
- Auditable Configurations
  - logging
- Hardening of Critical Components
  - isolation and micro-reboots

# Xoar: Architecture Overview



- Control VM disaggregated into nine classes of service VMs
- Each contains single-purpose control logic
- Some components may have multiple instances each for a client

# Xoar: Architecture Components



Four types of components:

1. Permanent:  
*XenStore-State* maintains all *state-of-the-system* info
2. Boot-Time:  
Components self-destruct before and User VMs run
3. Restart on each request:  
*XenStore-Logic*, *Toolstack*, *Builder*
4. Restart on timer:  
*Blk-Back* (Disk), *NetBack* (Net)

# Xoar: Architecture Components

Component	P	Lifetime	OS	Parent	Depends On	Functionality
Bootstrapper	Y	Boot Up	nanOS	Xen	-	Instantiate boot service VMs
XenStore	N	Forever (R)	miniOS	Bootstrapper	-	System configuration registry
Console	N	Forever	Linux	Bootstrapper	XenStore	Expose physical console as virtual consoles to VMs
Builder	Y	Forever (R)	nanOS	Bootstrapper	XenStore	Instantiate non-boot VMs
PCIBack	Y	Boot Up	Linux	Bootstrapper	XenStore Builder Console	Initialize hardware and PCI bus, pass through PCI devices, and expose virtual PCI config space
NetBack	N	Forever (R)	Linux	PCIBack	XenStore Console	Expose physical network device as virtual devices to VMs
BlkBack	N	Forever (R)	Linux	PCIBack	XenStore Console	Expose physical block device as virtual devices to VMs
Toolstack	N	Forever (R)	Linux	Bootstrapper	XenStore Builder Console	Admin toolstack to manage VMs
QemuVM	N	Guest VM	miniOS	Toolstack	XenStore NetBack BlkBack	Device emulation for a single guest VM

- “P” column indicates if component is privileged
- (R) in lifetime means component can be restarted

# Xoar: Architecture

- Service VMs Types
  - Self-Destructing
    - bring up the physical platform
      - PCIBack, Bootstrapper
  - Restartable
    - Toolstack, XenStore-Logic, Builder, BlkBack, NetBack
  - Long-lived
    - XenStore-State
  - Guest virtualization
    - Qemu

# Design Goals

- maintain exact same functionality as Xen
- complete transparency with existing management and VM interfaces
- for the components, three goals:
  - reduce privilege
  - reduce sharing
  - reduce staleness
- service VM concept
  - an entire VM capable of full hosting
  - can receive additional privileges from the hypervisor
  - can provide services to other VMs
  - only component that can be shared besides hypervisor

# Design Goals: Privilege

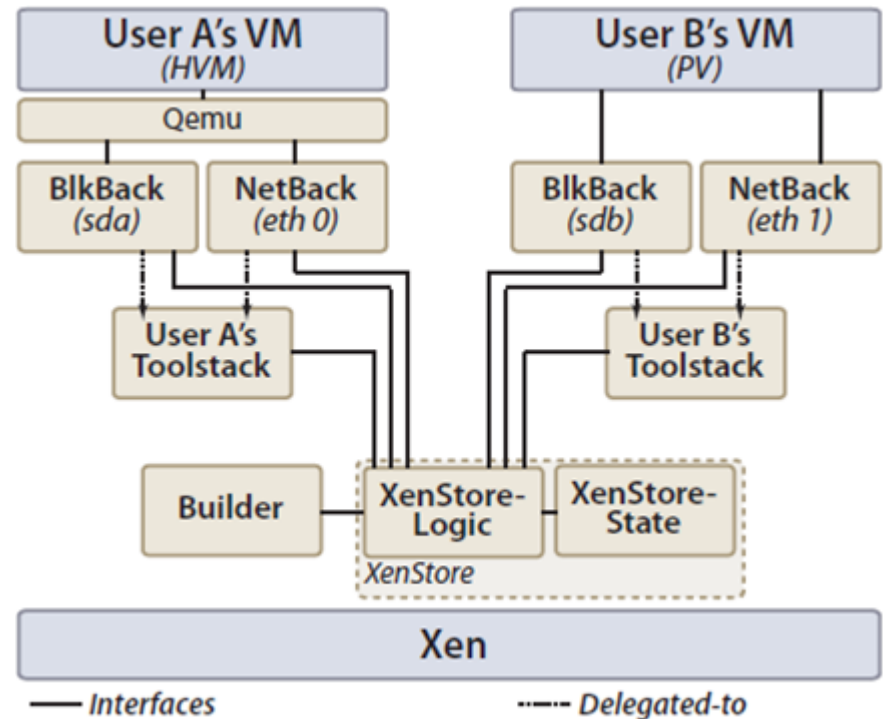
- only get privilege required for its purpose
- minimize exposed interfaces
- direct hardware assignment
- privileged hypercalls
  - hypercalls allow a service VM access to some of the privileged functionality provided by the hypervisor
- ability to delegate privileges to other VMs on creation

```
assign_pci_device (PCI_domain, bus, slot)
permit_hypercall (hypercall_id)
allow_delegation (guest_id)
```



# Design: Sharing

- avoid sharing components
- manage exposure
- confine and restrict attacks
- configuration constraints
  - user can specify whether to only share service VMs with guest VMs they control
- secure audit
  - logs written to append-only database



# Design: Sharing

- secure audit logging
  - written by Xoar to append-only database (postgres)
  - read by administrator for forensics

```
SELECT e1, e2 FROM log e1, log e2 WHERE
    e1.name = e2.name AND
    e1.action = 'create' AND
    e2.action = 'destroy' AND
    e1.dependency = 'NameOfCompromisedNetBack' AND
    overlaps(period_intersect(e1.time, e2.time),
              compromise_period);
```

Finding list of compromised VM;s

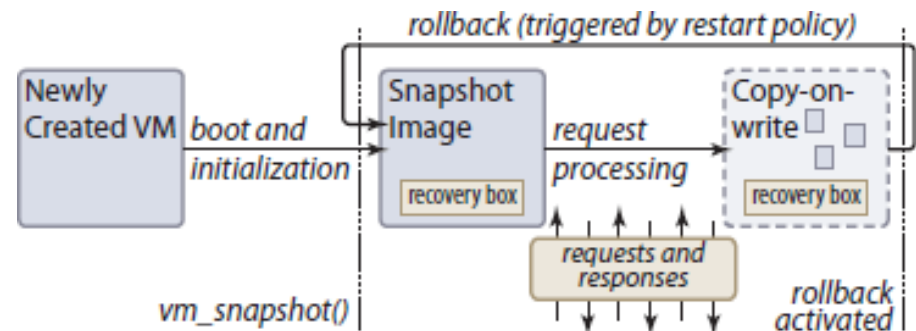
---

```
SELECT e1.name FROM log e1 WHERE
    e1.dependency = 'NetBack' AND
    e1.dependency_version = vulnerable_version;
```

Finding list of guest VM's which used a specific Service VM version

# Design: Staleness

- a component should only run as long as needed
- Micro-reboots
  - reason: a program is more likely to be correct at the beginning of execution rather than over long periods of time
- Snapshot and Rollback
  - instead of full restarts, components can be snapshotted once booted
    - faster to restore state than to reboot
- Restart Policy
  - notification-based or timer-based
- Maintain State
  - “recovery box” - a block of memory that exists across restarts



# Design Summary

- Reduce
  - privilege - access on a need-only basis
  - sharing - avoid when possible
  - staleness - maintain healthy state, VMs should only exist long enough to perform its task before restored to a known, secure state
- Service VMs are entire machines capable of hosting OSes and application stacks

# Deployment Scenarios

- Public clouds
  - ex: Amazon Web Services
- Private clouds
  - Free Open Source with Xen Community Edition
  - Xen can be tested within an application virtual machine

# XenStore

- XenStore-Logic
  - enforces access control
  - contains transactional logic
  - connection management
- XenStore-State
  - key-value store
  - long term storage

# Security Evaluation

- reduced TCB
  - bootstrapper, PCIBack and Builder are the most privileged components
  - bootstrapper and PCIBack destroyed once initialized but before guest VMs
- TCB is reduced from the control VM's 7.5 million lines of code (Linux) to Builder's 13,500 (on top of Xen)

# Vulnerability Mitigation

- Solved through isolation of services
  - device emulation
  - virtualized drivers
- XenStore re-written
- Hypervisor vulnerabilities remain

Component	Arbitrary Code Execution	DoS	File System Access
Hypervisor	0 / 1	0 / 1	0 / 0
Device Emulation	8 / 8	3 / 3	3 / 3
Virtualized Drivers	1 / 1	1 / 1	0 / 0
XenStore	0 / 0	1 / 1	0 / 0
Toolstack	1 / 1	2 / 2	1 / 1

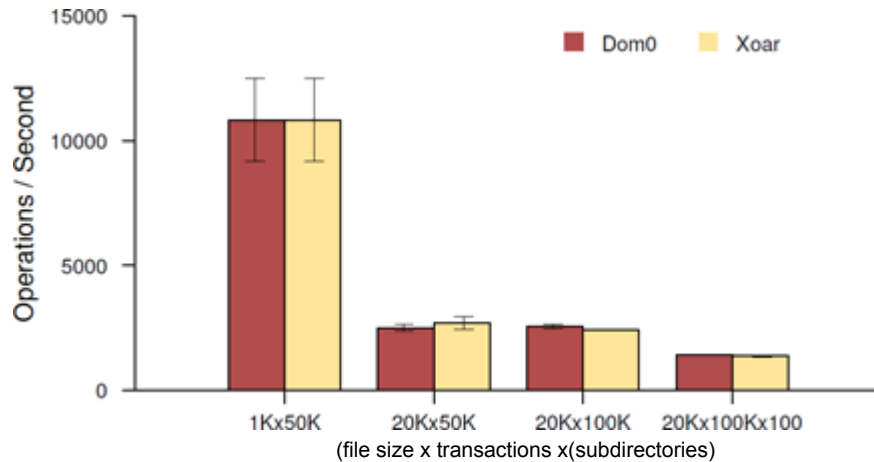


# Performance Evaluation

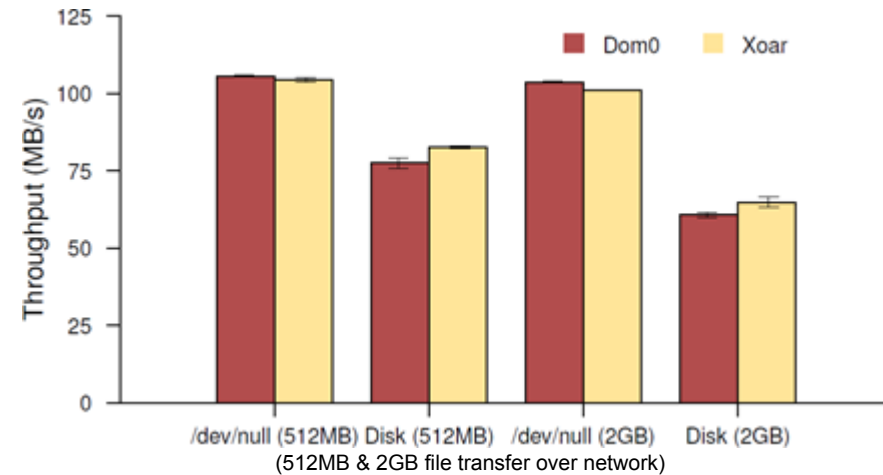
- Test system
  - Ca. 2011 server
  - Quad-core Xeon, 4Gb RAM
  - All virtualization features enabled
- Memory overhead
  - 512Mb – 896Mb in Xoar vs.
  - 750Mb in XenServer

# I/O Performance

Disk Performance (using Postmark)  
(higher is better)



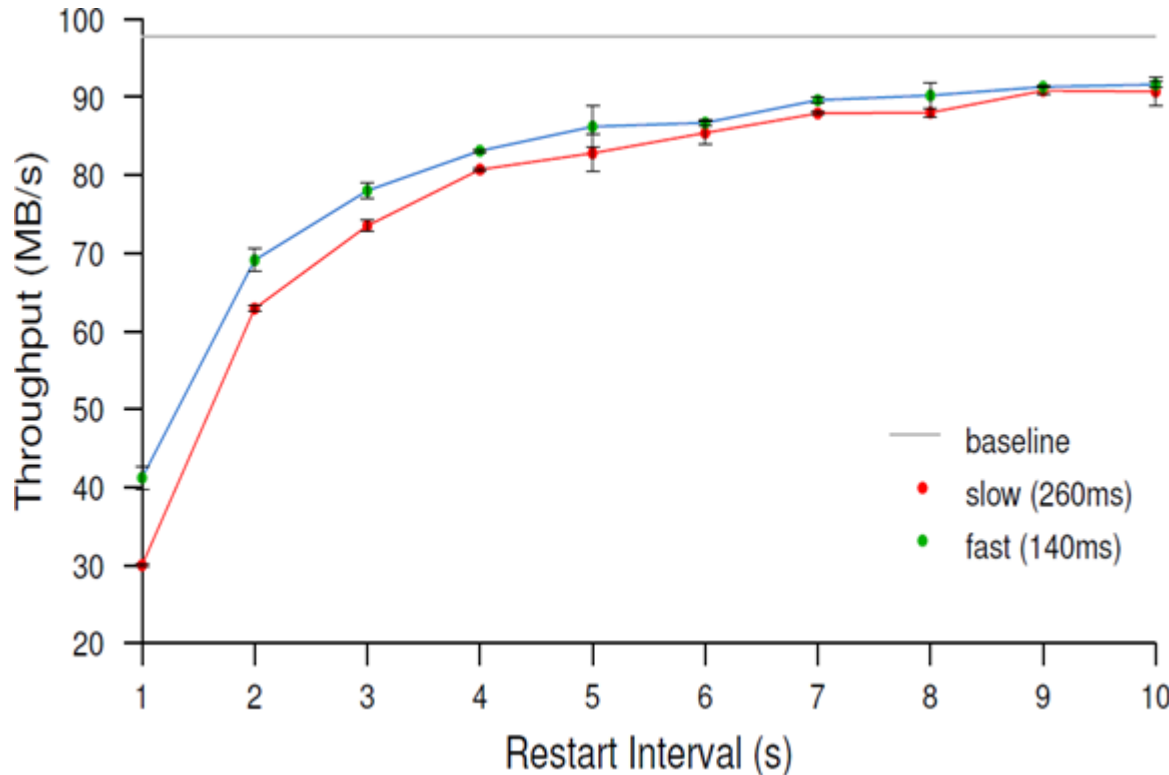
Network Performance (using wget)  
(higher is better)



- Disk Performance overall unchanged
- Network throughput down 1-2.5%
- Combined throughput of network à disk increased by 6.5%

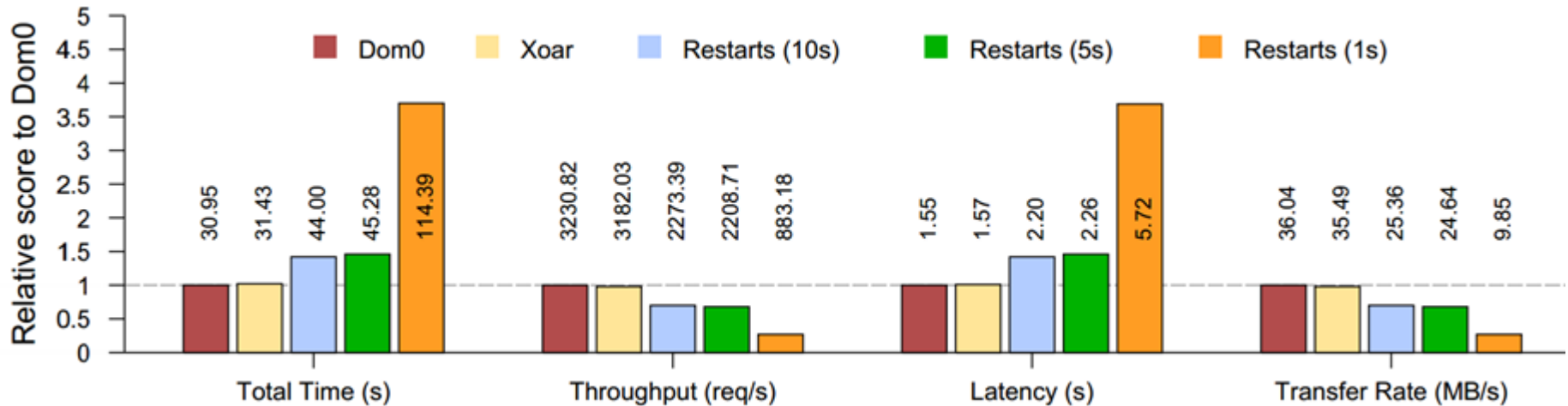
# Effects of Micro-rebooting

## Effects of Micro-rebooting



- Downtime:
  - slow: 260ms
  - fast: 140ms
- Restart frequency on throughput:
  - 1s: 58%
  - 10s: 8%
  - >10s: negligible
- Fast mode helps with frequent reboots but only 1% @ 10s

# Real-World Benchmarks



- Apache Benchmark: serving static page (10KB / 100K / 5 clients)
- Performance decreases non-uniformly with the restart frequencies
- 5s→1s = significant performance loss
- Dropped packets / network timeouts = longer time to complete packets:
  - Dom0 / Xoar: 8-9ms
  - 5/10s: 3000ms
  - 1s: 7000ms
- Disaggregation overhead quite low as
- Effect of driver restarts, while noticeable, can be tuned to balance security vs performance

# Xen Screenshots

GNU GRUB version 0.97 (638K lower / 3143616K upper memory)

```
Xen 3.2-1 / Debian GNU/Linux - live, kernel 2.6.26-1-xen
Debian GNU/Linux - live, kernel 2.6.26-1-amd64
Debian GNU/Linux - live, kernel 2.6.26-1-amd64 (fail-safe mode)
mentest86+
```



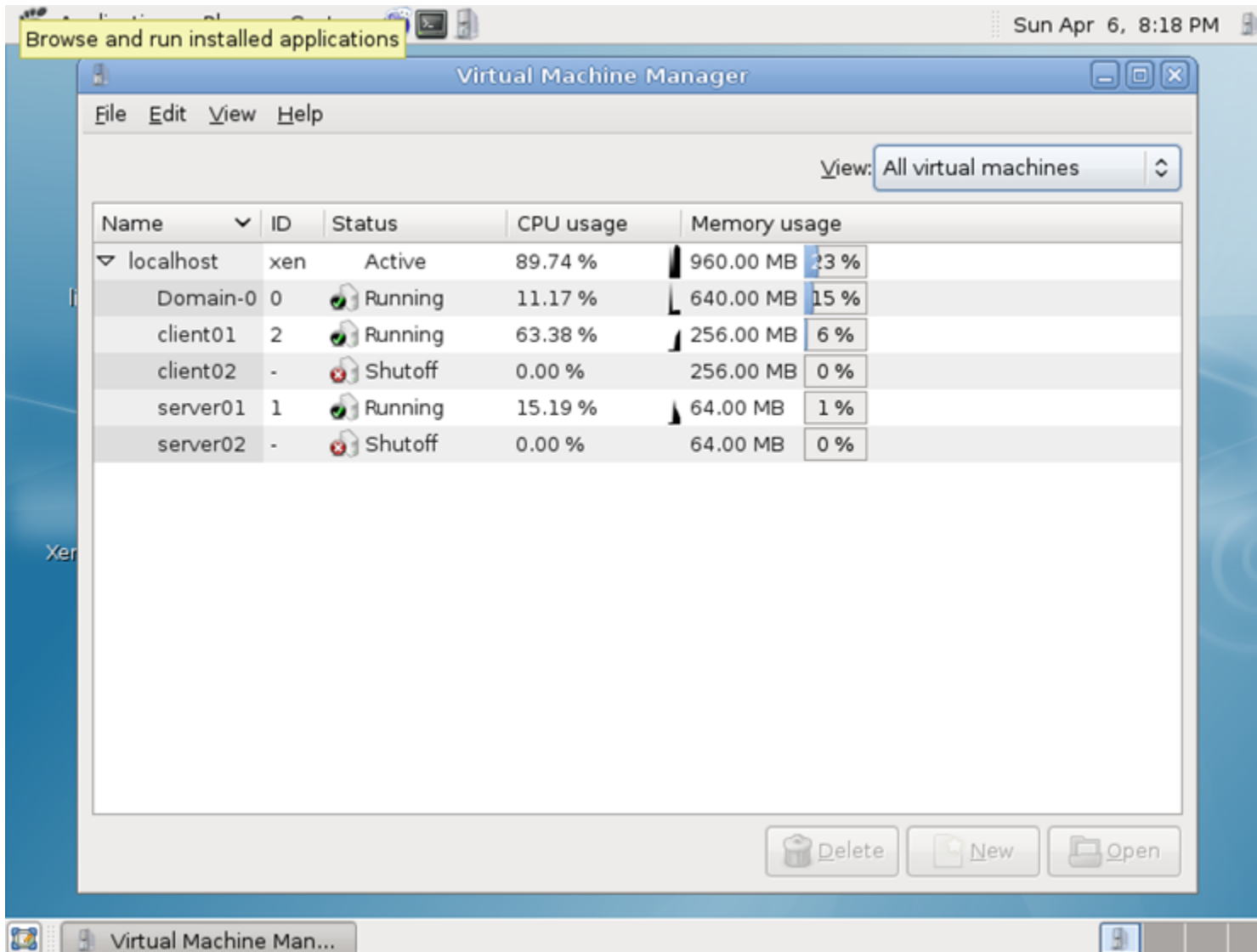
Use the ↑ and ↓ keys to select which entry is highlighted.  
Press enter to boot the selected OS, 'e' to edit the  
commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 27 seconds.

# Xen Screenshots

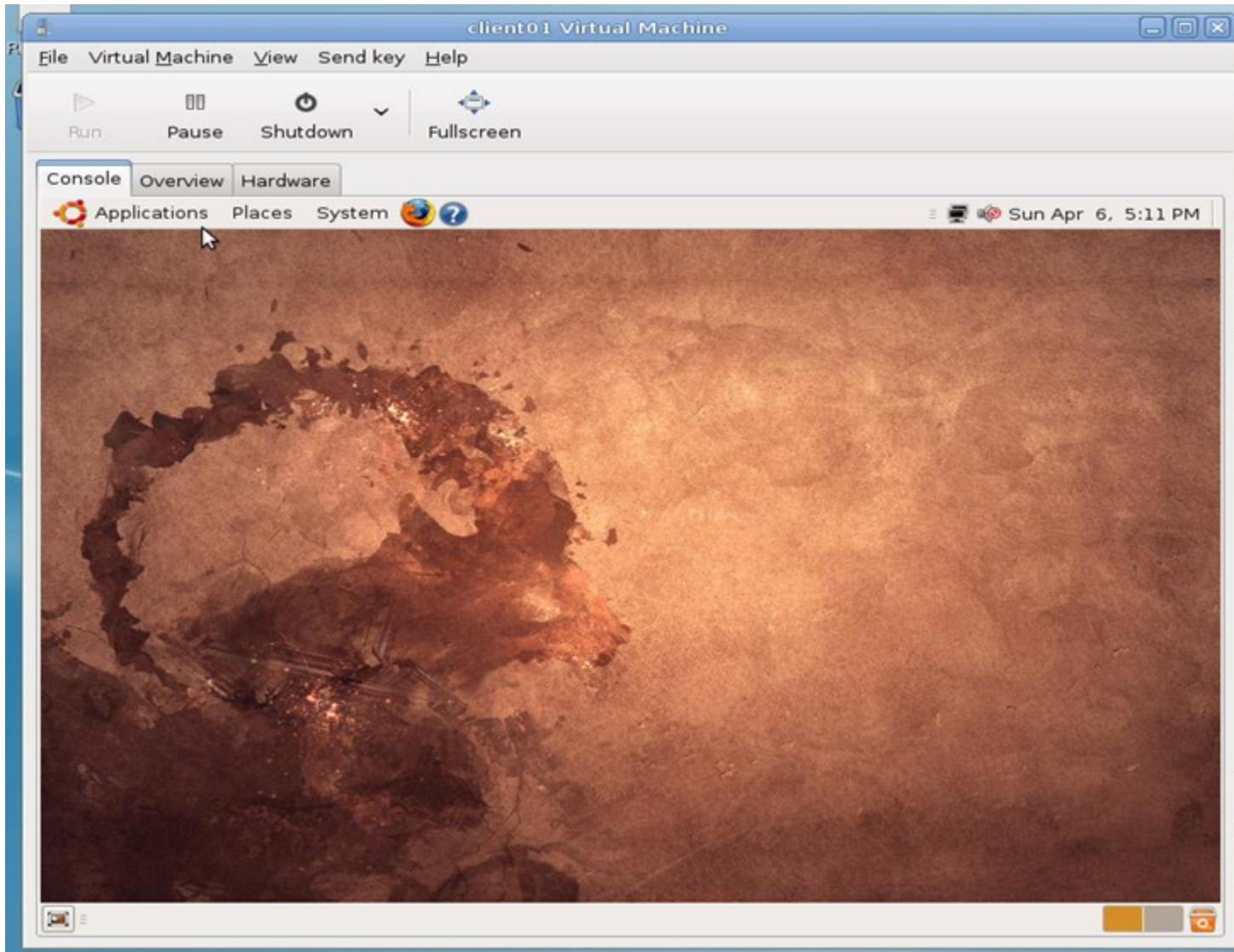
```
(XEN) -> Using new ACK method
(XEN) Platform timer overflows in 14998 jiffies.
(XEN) Platform timer is 14.318MHz HPET
(XEN) Brought up 1 CPUs
(XEN) xenoprof: Initialization failed. Intel processor model 42 for P6 class family is not supported
(XEN) AMD IOMMU: Disabled
(XEN) *** LOADING DOMAIN 0 ***
(XEN) Xen kernel: 64-bit, lsb, compat32
(XEN) Dom0 kernel: 64-bit, lsb, paddr 0x200000 -> 0x631918
(XEN) PHYSICAL MEMORY ARRANGEMENT:
(XEN) Dom0 alloc.: 0000000134000000->0000000138000000 (147456 pages to be allocated)
(XEN) VIRTUAL MEMORY ARRANGEMENT:
(XEN) Loaded kernel: ffffffff80200000->fffffff80631918
(XEN) Init. ramdisk: ffffffff80632000->fffffff8201be00
(XEN) Phys-Mach map: ffffffff8201c000->fffffff8215c000
(XEN) Start info: ffffffff8215c000->fffffff8215c4a4
(XEN) Page tables: ffffffff8215d000->fffffff82172000
(XEN) Boot stack: ffffffff82172000->fffffff82173000
(XEN) TOTAL: ffffffff80000000->fffffff82400000
(XEN) ENTRY ADDRESS: ffffffff80200000
(XEN) Dom0 has maximum 1 VCPUs
(XEN) Initrd len 0x19e9e00, start at 0xffffffff80632000
(XEN) Scrubbing Free RAM: .....
```

# Xen Screenshots



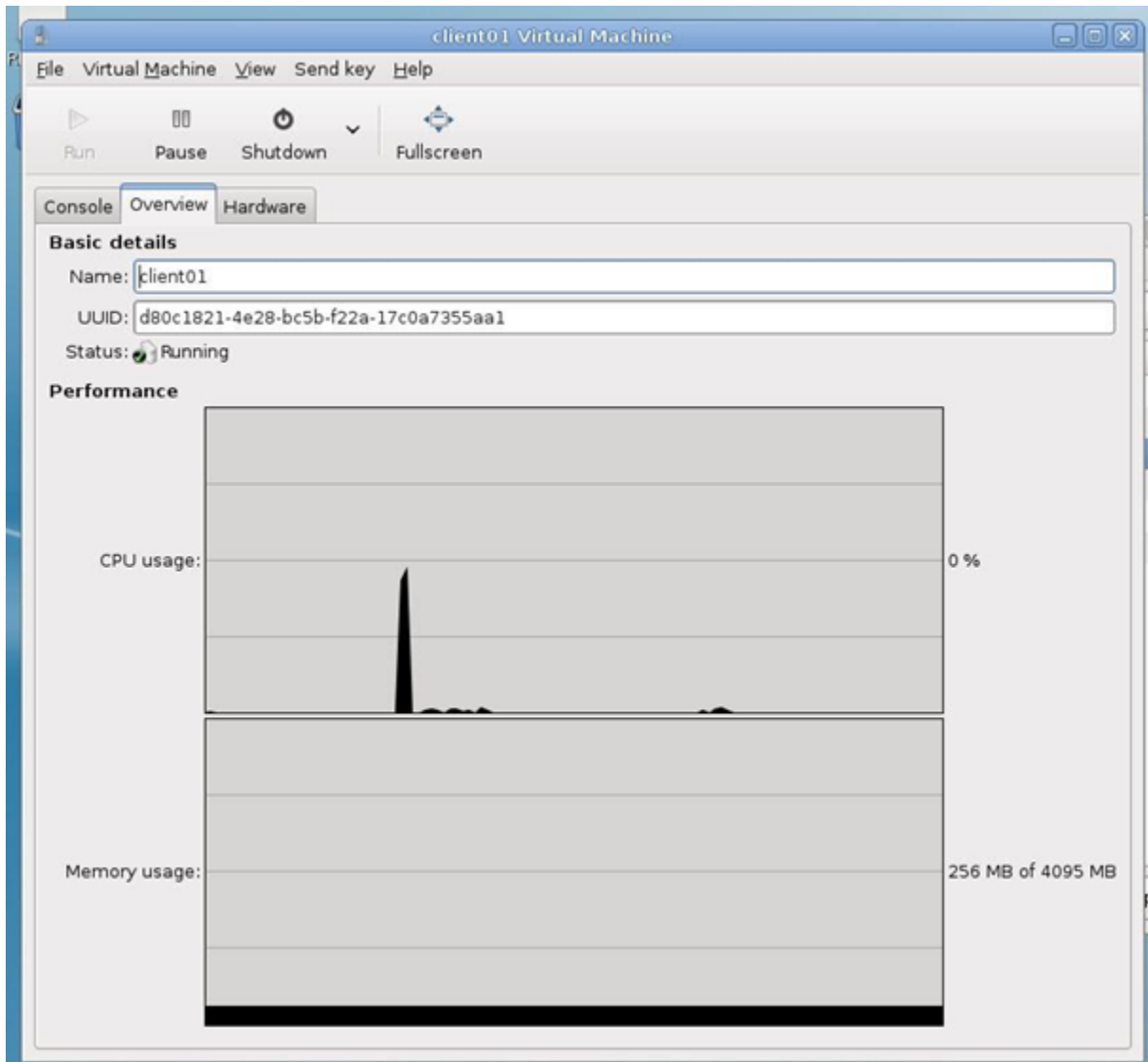


# Xen Screenshots

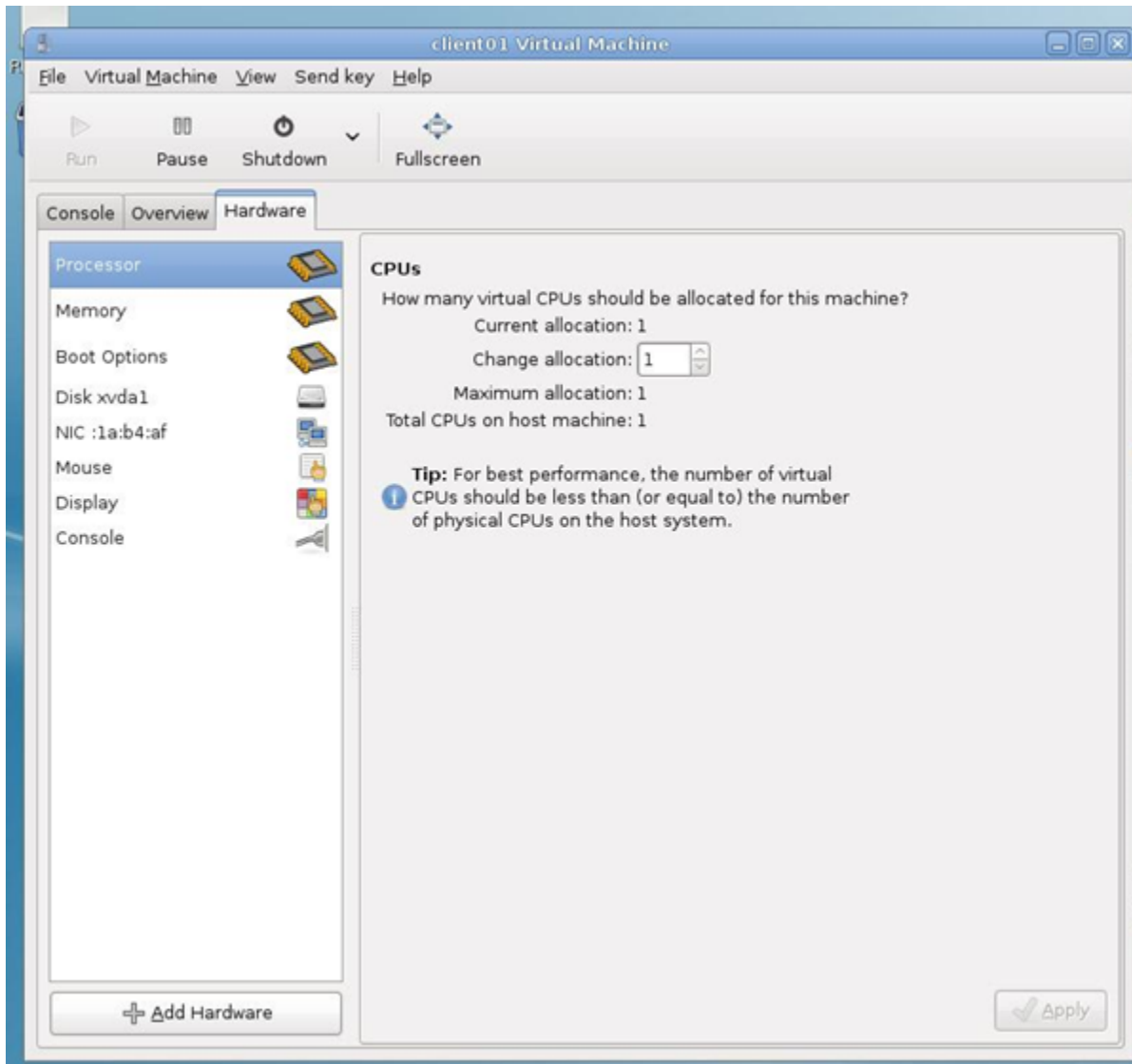




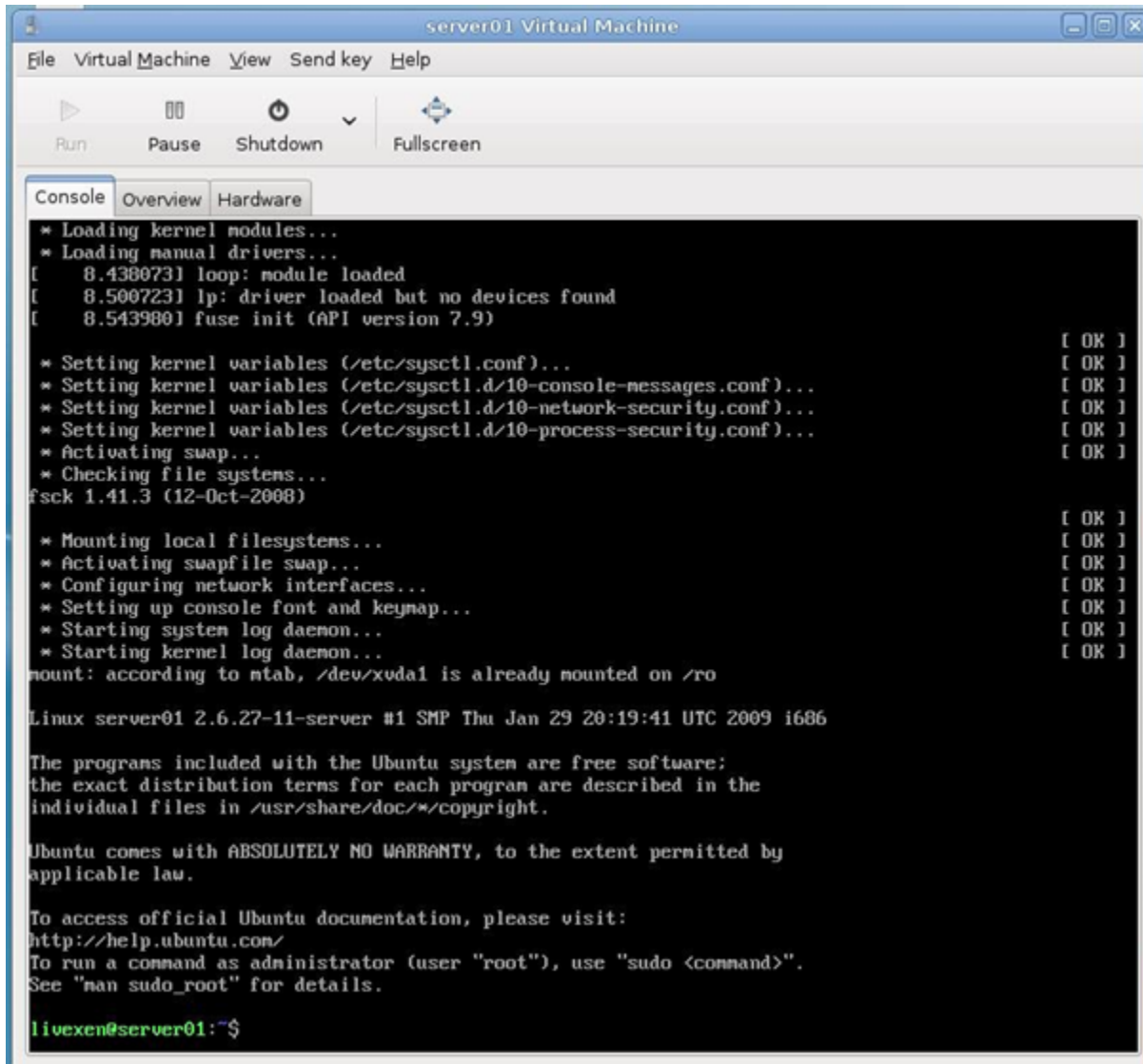
# Xen Screenshots



# Xen Screenshots



# Xen Screenshots



```
server01 Virtual Machine
File Virtual Machine View Send key Help

Run Pause Shutdown Fullscreen

Console Overview Hardware
* Loading kernel modules...
* Loading manual drivers...
[ 8.438073] loop: module loaded
[ 8.500723] lp: driver loaded but no devices found
[ 8.543980] fuse init (API version 7.9)
[ OK ]
* Setting kernel variables (/etc/sysctl.conf)...
[ OK ]
* Setting kernel variables (/etc/sysctl.d/10-console-messages.conf)...
[ OK ]
* Setting kernel variables (/etc/sysctl.d/10-network-security.conf)...
[ OK ]
* Setting kernel variables (/etc/sysctl.d/10-process-security.conf)...
[ OK ]
* Activating swap...
[ OK ]
* Checking file systems...
fsck 1.41.3 (12-Oct-2008)
[ OK ]
* Mounting local filesystems...
[ OK ]
* Activating swapfile swap...
[ OK ]
* Configuring network interfaces...
[ OK ]
* Setting up console font and keymap...
[ OK ]
* Starting system log daemon...
[ OK ]
* Starting kernel log daemon...
[ OK ]
mount: according to mtab, /dev/xvda1 is already mounted on /ro

Linux server01 2.6.27-11-server #1 SMP Thu Jan 29 20:19:41 UTC 2009 i686

The programs included with the Ubuntu system are free software:
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

livexen@server01:~$
```

# Resources

- Available for free download at [xenproject.org](http://xenproject.org)
- Screenshots were taken on VMWare Workstation
  - Xen Live CD
    - <http://wiki.xen.org/wiki/LiveCD>

# Conclusion

- Xoar improves hypervisor security with a small performance penalty
- Components of the Control VM are a major source of attack
- Xoar isolates components in space and time
  - Exploits are constrained
  - Makes the exposure to risk explicit
  - Provides means for later forensic analysis
- Functionality, performance, and maintainability are not impacted
- Comments or questions?
- Thank you for your attention

# Backup

# Purpose

- **Xoar**
  - attempts to divide the hypervisor
    - maintain the same hypervisor capabilities
    - minimize performance overhead
    - strongly isolate components
    - reduce attacks



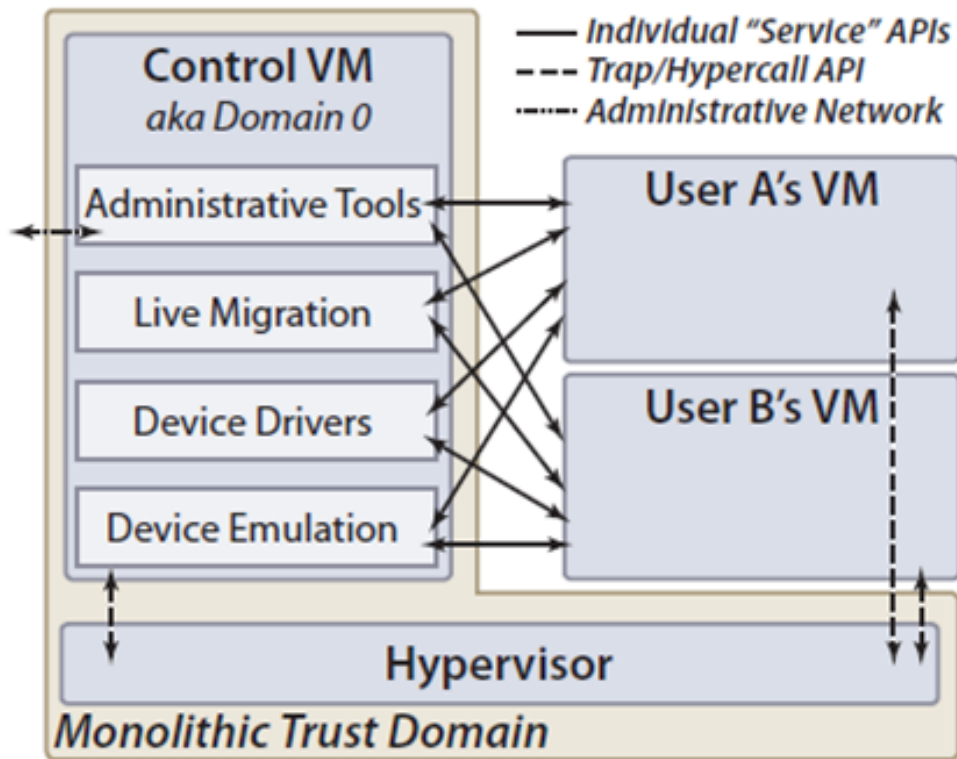
# Trusted Compute Block (TCB)

- protection mechanisms that enforce a security policy
- responsible for guest VM
  - isolation
  - scheduling
  - memory management
- hardware management
- device emulation
- inter-VM communication
- virtual consoles
- configuration state management

Defined as *“the totality of protection mechanisms within a computer system -- including hardware, firmware, and software -- the combination of which is responsible for enforcing a security policy”* -- Source wikipedia



# Trusted Compute Block (TCB)



Xen, by virtue of privilege, is part of the TCB. Compromise of any component provides:

- Privilege of that component
- Use of its interfaces to other components

# Example Attack Vectors

- CERT vulnerability database
- VMWare's list of security advisories
  - Most attacks were against service components in the control VM

# Xoar Threat Model

## Assumptions:

- Well & professionally managed virtualization platform
- attacker is using a guest VM
- the attacker aims to violate security of another guest
- guests are on the same platform (VM host server)
- the control VM will contain bugs

## Goal:

Rather than fighting bugs, isolate functional components in space/time so an exploit of one component is not sufficient to mount an attack against another guest or the underlying platform.

# The Xen Platform

- Device Drivers
  - delegated to Dom0 which exposes to guest VM's
  - devices may be virtualized, passed through, or emulated
  - unmodified OSes run on VMs through Qemu
- XenStore
  - key-value store; a system-wide registry and naming service
  - Most critical component
    - Vulnerable to DoS attacks, Performs most admin operations
- Toolstack
  - provides administrative functions for management of VMs
- System Boot
  - the hypervisor creates Dom0
    - initializes hardware, devices, and back-end drivers
  - XenStore is loaded before guest VMs

# Notes

- <http://643-spring14:cloud14amazon@cs.njit.edu/~borcea/cs643/docs/CS643-s14-l10.pptx>  
has info about Xen that will be presented by the professor
- Presentation should be 25 slides long and take 45 minutes
- Supposed to show how the technology works
- Grading based on clarity, presentation is 10% of class grade, 50% group and 50% individual
- Slide #'s needed in footer
- Add screenshot of Xen Live CD
- Send slides to professor by ~~Monday~~ Wednesday
- <http://www.slideshare.net/RussellPavlicek/xen-security-cloudopen20131>
  - similar, more recent presentation