Operating System Virtualization: Practice and Experience

Oren Laadan and Jason Nieh
Columbia University
{orenl,nieh}@cs.columbia.edu
Virtualization

“All problems in computer science can be solved with another layer of indirection”

-- (attributed to) David Wheeler
Virtualization - Why

- Computers are faster, cheaper, more connected
- (Re)Emerged to address management complexity and security hazards
- Provides isolation and mobility
  - consolidation, security
  - fault tolerance, load balancing, availability
Virtualization - Where

- **Hardware or Operation system?**
  - HW: decouple entire operating system
  - OS: decouple individual applications
OS Virtualization - Taxonomy

complete ?
host independent ?
OS Virtualization - Taxonomy

complete?

BSD Jail

host independent?
OS Virtualization - Taxonomy

- BSD jail
- virtual memory

complete?

host independent?
OS Virtualization - Taxonomy

complete ?

Solaris zones

BSD Jail

virtual memory

host independent ?
OS Virtualization - Taxonomy

- complete ?
- host independent ?
- BSD Jail
- Solaris zones
- Zap
- virtual memory
OS Virtualization - Taxonomy

- BSD Jail
- Solaris zones
- Zap
- virtual memory

complete?

host independent?
Virtual Private Namespace

Virtual Execution Environment (VEE)

Virtualization Layer

Operating System

Hardware
Private Namespace

VEE #1

VEE #2

......

VEE #n

Virtualization Layer

Operating System

Hardware
Virtual Namespace

getpid()?

Virtualization Layer

Operating System

Hardware
Virtual Namespace

getpid() ?
Virtual Namespace

getpid() ?

Virtualization Layer

Operating System

Hardware
Virtual Namespace

getpid()?

Kernel: 4150 (real)
Virtual Namespace

getpid()?

Kernel: 4150 (real)

**VEE:** 305 (virtual)
OS Virtualization - Where

- **Interposition**
  - user-level
  - process tracing
  - in-kernel
The Virtual Diner...
The Virtual Diner...

- HW or OS?
The Virtual Diner...

- HW or OS?

  “I'd like OS virtualization, please”
The Virtual Diner...

- HW or OS?
  - “I'd like OS virtualization, please”
- Complete? Host independent?
The Virtual Diner...

- HW or OS?
  “I'd like OS virtualization, please”
- Complete? Host independent?
  “I'll have host independent, and add complete on the side.”
The Virtual Diner...

- HW or OS?
  
  "I'd like OS virtualization, please"

- Complete? Host independent?
  
  "I'll have host independent, and add complete on the side.

- Userspace or in-kernel?
The Virtual Diner...

- **HW or OS?**
  - “I'd like OS virtualization, please”

- **Complete? Host independent?**
  - “I'll have host independent, and add complete on the side.”

- **Userspace or in-kernel?**
  - “can I have it as a kernel module?”
“All problems in computer science can be solved with another layer of indirection ...  
... but that usually will create another problem.”

-- (attributed to) David Wheeler
Virtualization - Challenges

- Performance overhead
- Race conditions
Interposition Revisited

getpid()?

Kernel: 4150 (real)

VEE: 305 (virtual)
Race Conditions

- syscall wrapper
  - translate virtual → physical
  - invoke kernel system call
  - translate physical → virtual

(1)
(2)
Race Conditions

- syscall wrapper
  - translate virtual → physical
  - invoke kernel system call
  - translate physical → virtual

- Wrapper preamble (1), epilogue (2)
- Initialization, deletion, re-use
PID Races - reuse

Process A
pid 100/vpid 400

SYS_GETPGID(420)
virt2phys(420)
→ 110
getpgid(110)
→ 110

phys2virt(110)
→ 655

Process B
pid 110/vpid 420

SYS_EXIT(0)
→ exited

Process C

CREATED
→ pid=110
→ vpid=655
**PID Races - init**

**Parent**
pid 100/vpid 400

SYS_FORK()
fork()
→ pid = 110

**Child**

CREATED
→ pid = 110

SYS_GETPID()
 getpid()
→ 110
phys2virt(110)
→ undefined

→ vpid = 420
IPC Races - reuse

**Process A**
- SYS.MSGSND(55, ...)
- virt2phys(55)
  - → 10
- msgsnd(10, ...)  
  - → illegal

**Process B**
- SYS.MSGCTL(55, IPC_RMID)
- virt2phys(55)
  - → 10
- ipcrm(55, 10)
  - → deleted

**Process C**
- SYS.MSGGET(...)
  - → 10
  - → vpid=655
Experimental Evaluation

- **Micro benchmarks**
  - overhead on select system calls
- **Macro benchmarks**
  - overhead on select applications
- **Scalability**
  - multiple VEEs running concurrently
Micro-benchmark

→ Runtime overhead is low
Macro-benchmark

→ Runtime overhead is lower
Scaling-benchmark

→ excellent scalability
Experience

- Application checkpoint-restart (Zap)
  - guarantee same environment at restart
- Application record-replay (Scribe)
  - guarantee same environment at replay
Conclusions

- **OS Virtualization**
  - useful but challenging
- **Virtualization via interposition**
  - address race conditions
  - meet performance needs
Conclusions

- OS Virtualization
  - useful but challenging
- Virtualization via interposition
  - address race conditions
  - meet performance needs

Thank You!
Virtualization – How

✦ Decouple execution from underlying environment
 ✦ execution should exhibit effect identical to un-virtualized system
 ✦ dominant part of the execution should be direct