

# SIOEMU: Self IO EMUlation

Tristan Gingold  
tgingold@free.fr

---

---

# *Every solution comes from a problem*

These OS can be fully virtualized on Xen/ia64:

- Linux
- Windows

But they come from the x86 world (so does Xen)

What about non-x86 OS ?

- HP-UX: hp-pa -> ia64 (m68k is dead)
  - OpenVMS: alpha -> ia64 (vax is dead)
- 
-

## *The Initial Issue*

I wanted to run OpenVMS on Xen/ia64.

- OpenVMS uses more Itanium features, eg:
  - Uses the 4 protection levels
  - tak instruction

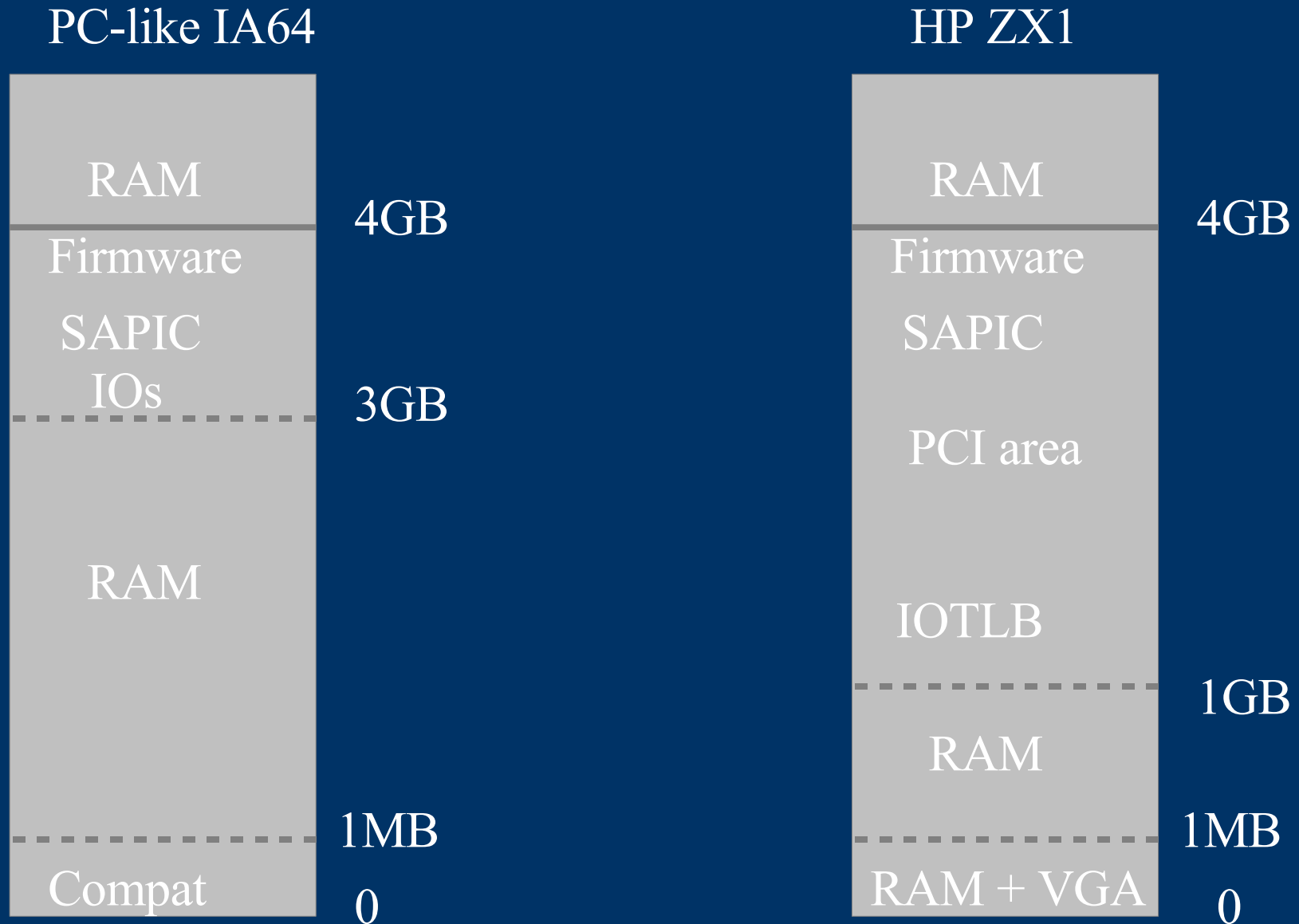
OpenVMS works only on HP machine.

But Xen/ia64 HVM domains emulate a PC-like machine.

Initial try failed at a point:

- OpenVMS requires an IO-TLB
  - HVM domains don't provide it.
- 
-

# PC vs HP ZX1 machines



# ZX1 IOTLB

ZX1 machines are server-class

- Many PCI buses

Need to work with 32bits PCI cards

- At least the IDE controller is 32bits (used for DVD)

ZX1 has an IOTLB

- 32bits card can access the whole memory without bounce buffers
  - Translation only (no isolation) – like AMD GART
- 
-

# *Possible solutions*

- Modify OpenVMS
    - The IOTLB was required only by IDE driver
    - But the sources are not easily available
    - Too much work...
  - Modify Qemu to support ZX1 machines
    - Possible but extensive work is necessary
    - HVM domains are not flexible enough
  - Make HVM domains more flexible
    - Appealing
    - SIOEMU
- 
-

# *Why is HVM not flexible ?*

- Memory map is defined by:
    - The domain builder (which allocates memory)
    - The hypervisor (which defines IO regions)
    - QEMU (which defines devices)
  - Changing memory map is awkward!
  - QEMU doesn't really support 64bits PCI cards
  - QEMU pci layer doesn't have an IOTLB interface
- 
-

## *Other HVM issues*

- IO emulation is slow
    - IO accesses trapped by the hypervisor
    - Sent to QEMU through shared page + event
    - QEMU is running in domain 0 as a Linux process
  - Not secure (fixed by stub-domain):
    - QEMU run as root in domain 0
    - An intrusion could crash the whole system
    - Workload in domain 0
  - Difficult to account
    - HVM time is spent in the domain and in domain 0
- 
-

# *SIOEMU: an elegant approach ?*

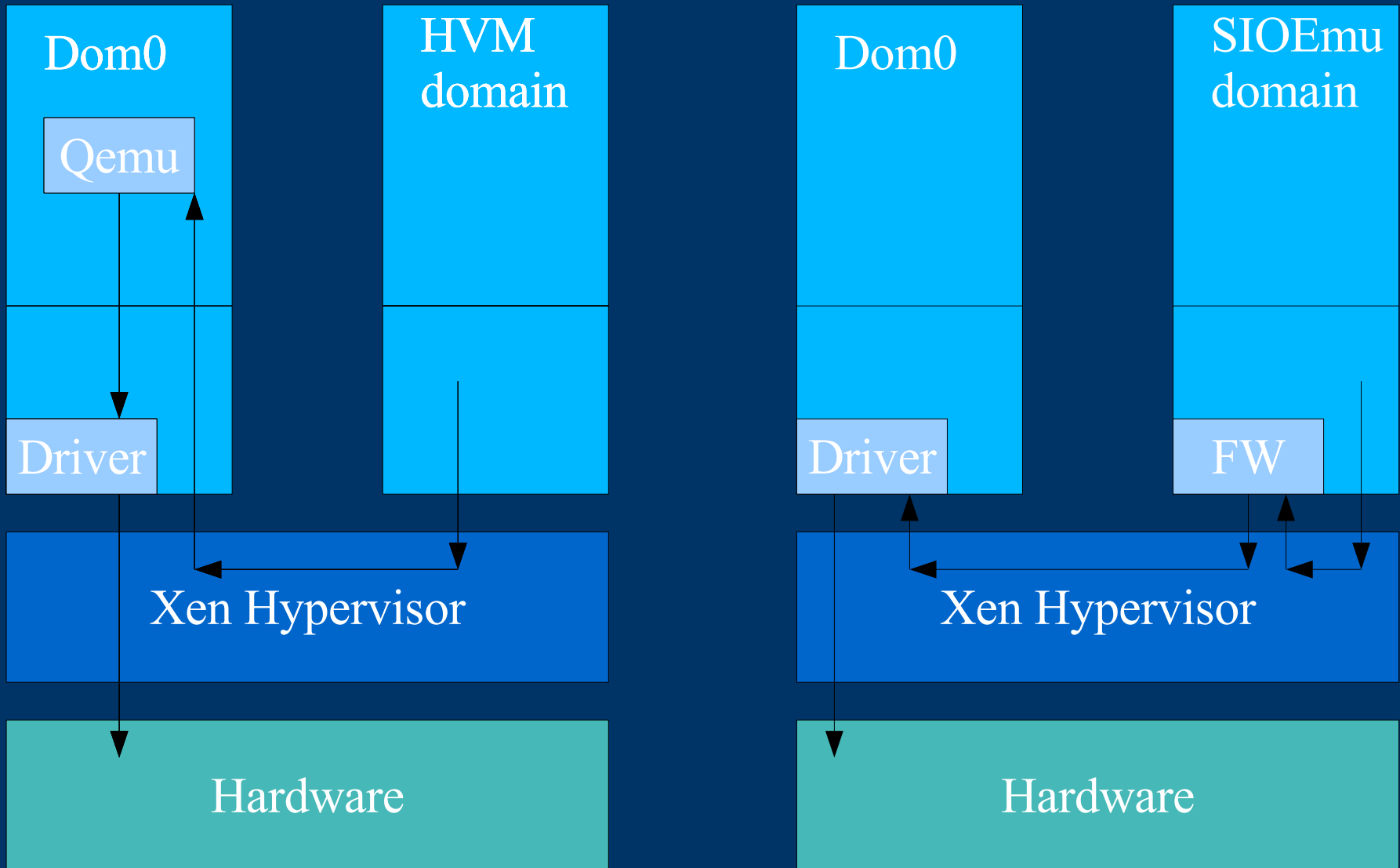
Basic idea:

- Do all the work within the domain.
- IO emulation is done by a firmware.

External accesses is done through PV drivers.

- So it looks like a PV domain
  - But can run non-PV OSES
- 
-

# *IO Emulation*



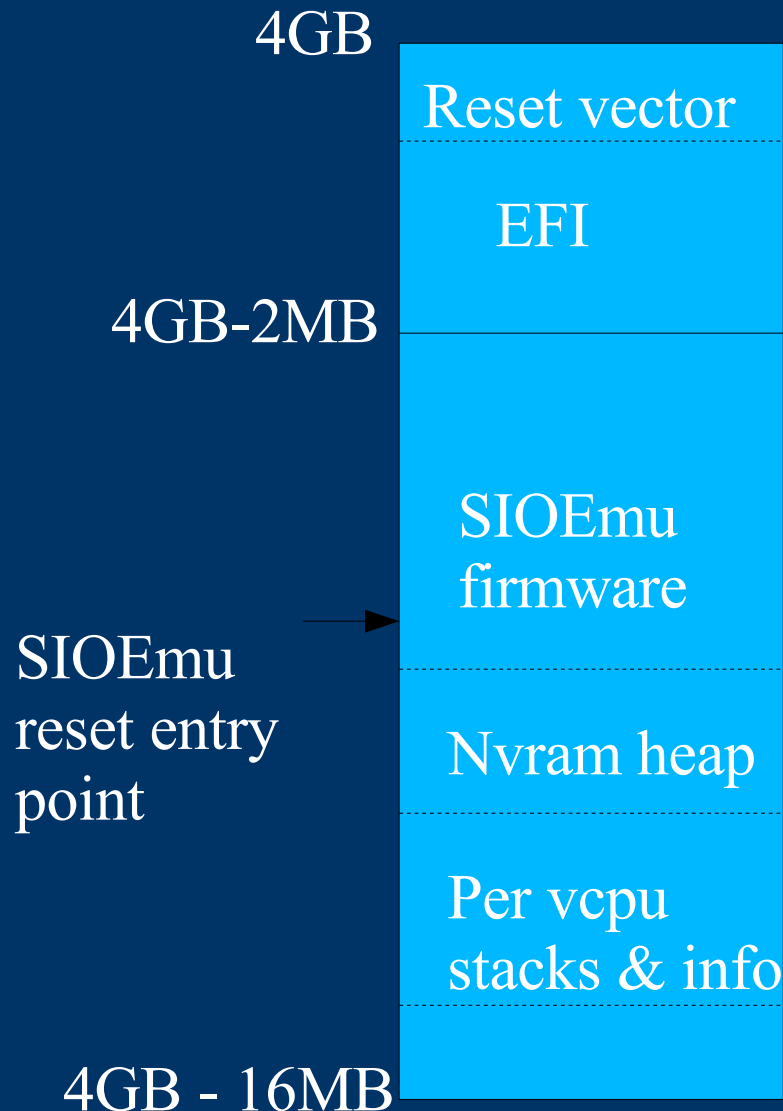
# *IO Emulation*

- Hypervisor traps IO accesses
    - Decode the instruction.
  - Like HVM domains
  - Hypervisor save the context, points to a firmware entry point, switch to physical mode.
  - The firmware emulate the IO
    - VBD/VNIF/Console accesses if required
  - The firmware restore the context to continue the execution
- 
-

# *The firmware*

- No dynamic allocation beyond boot:
    - Resources are allocated during initialization
      - Grant table slots, event ids, VNIF/VBD pages...
    - Avoids potential crash during run
  - No threads, fully event based
    - Easier to debug
  - Invoked by the hypervisor when:
    - vcpu does an IO/MMIO access (per vcpu).
    - An event has to be delivered
    - EFI/SAL/PAL requests (set rdv vector)
- 
-

# Firmware memory map



The firmware area is defined by the Itanium architecture.

EFI image is partially compressed (on real hardware, flash accesses are slow).

Runs at PL=0, physical mode

No free – allocated during boot

64KB stack per vcpu

Shared info, console, xenstore, start

# *Domain builder*

- Domain builder loads the firmware (specific loader)
    - PV domain from xend POW
    - The firmware is the kernel image ( $\leq 16\text{MB}$ )
    - Map start\_info, console and xenstore pages
    - Alternate entry point.
  - Firmware contains:
    - IO emulation firmware
    - PAL/SAL/EFI (ie the ia64 BIOS)
  - No current use of ramdisk image
- 
-

# *Firmware setup*

- Set callback
  - Initialize console (useful for early debug)
  - Decode command line
  - Initialize xenbus and grant table
  - Initialize PV drivers (VBD, VNIF)
  - Setup memory map:
    - Set IO and MMIO areas
    - Populate RAM
      - Naturally NUMA friendly
  - Start EFI (using regular reset vector)
- 
-

# *The firmware – devices model*

- IO emulation code is from qemu
    - Use only a small number of files, mostly from hw/
      - ide.c, cdrom.c, serial.c, e1000.c, pci.c...
      - This API looks stable (large number of uses)
    - Try to be in synch with upstream
  - ZX1 drivers added
  - IO and MMIO dispatching is custom (64bits accesses allowed).
  - IOSAPIC emulated in the firmware
    - Hypervisor version can be used (less flexible)
- 
-

# Status

- SIOEmu domains can run
    - Linux (both DIG and ZX1)
    - OpenVMS
  - Hypervisor code merged
  - A few bugs found in the hypervisor
    - IO instruction decoding
    - Hypercall continuation in VTi mode
  - Some features added
    - Better support for dt != it
    - tak emulation improved
- 
-

# *Performance (1)*

Real potential:

- No domain scheduling required to do an IO emulation without IO access.
- Cache & NUMA friendly (IO emulation runs on the same processor)

Boot very quickly

- Nice when doing firmware or EFI development
- Can start before xencons is connected

No cpu cycles spent on other domain budget

- Except real hardware accesses
- 
-

## *Performance (2)*

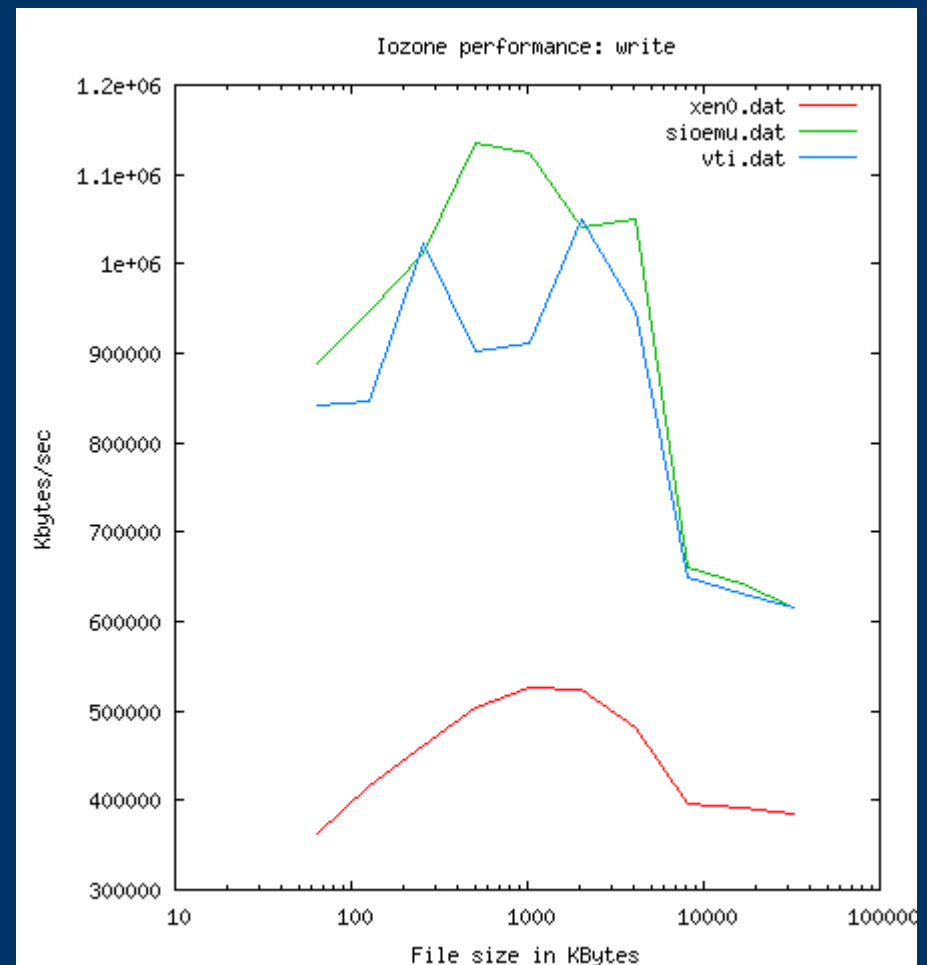
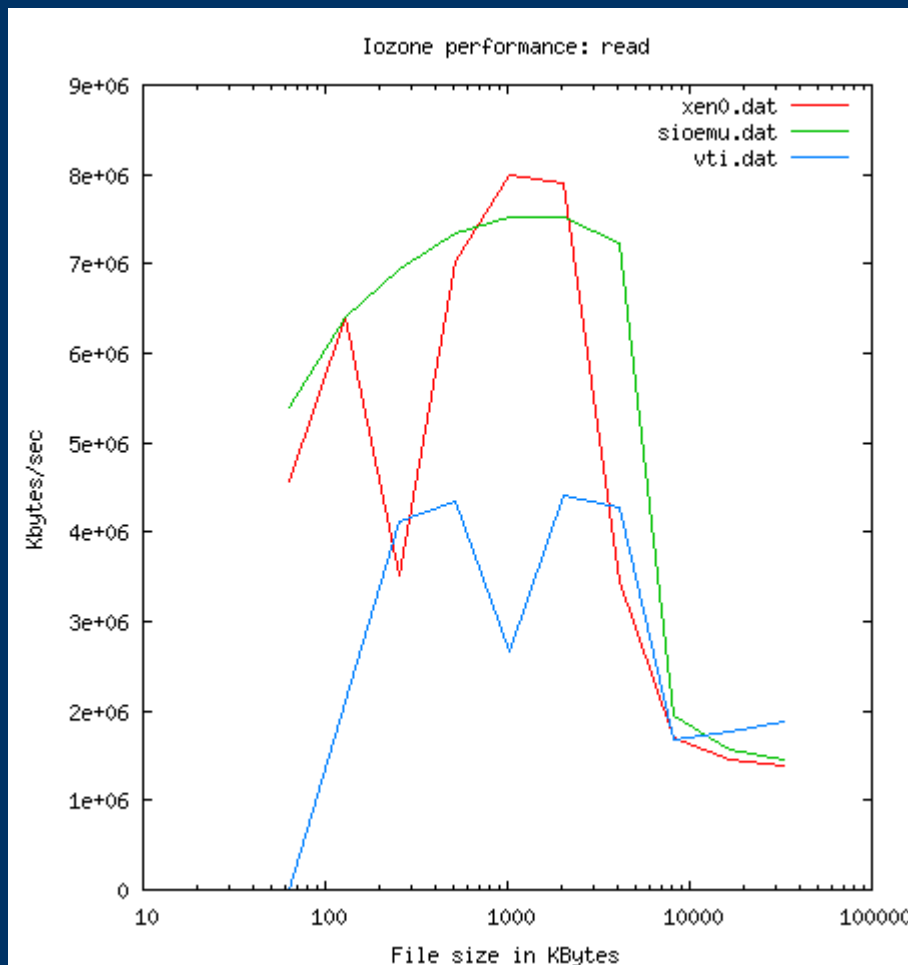
Provides an elegant method to extend the hypervisor

- No need to add IO emulation in the hypervisor
  - (if transition to FW is quick enough)
- Has the same memory view that a device
  - No need to have a map cache

No performance tuning yet

- TODO: zero copy IDE emulation.
  - TODO: zero copy NIF emulation (netchannel2)
- 
-

# IDE Performance



# *SIOEmu vs HVM*

## Pros:

- The firmware defines the machine
- Same security level as PV domains
- Keep hypervisor and tools small
- Resource accounting

## Cons:

- Need to write the firmware :-)



# *Work in progress*

SMP: which lock strategy ?

- Global lock – IO emulation is serialized
    - Like QEMU
    - Not friendly to IO bounded applications
  - No lock
    - Err – doesn't work
  - Per device lock
    - Might diverge from upstream QEMU.
- 
-

# WIP & TODO

Save & restore:

- Need support in firmware PV drivers

VGA/KBD/Mouse:

- Required to do a first installation of Windows
    - No support in QEMU for recent cards
    - Might leverage on EFI GOP
  - Not required by Linux or OpenVMS
  - Performance tuning area (avoid scanning)
- 
-

# WIP & TODO

## NVRAM

- Required by EFI
- Current HVM implementation is ad-hoc
- Rely on a VBD ?

## Populate memory with continuous machine page

- Take advantage of larger pages

## Virtualize other Itanium machine

- SGI Altix

## X86 Port

---

---

*Questions ?*

