



# Gelato

## UNSW

Performance and  
Scalability on Itanium

[www.gelato.unsw.edu.au](http://www.gelato.unsw.edu.au)



# Linux as Hypervisor

**Peter Chubb**

Reporting on work by

**Matthew Chapman      myself**

**Gelato@UNSW**

National ICT Australia

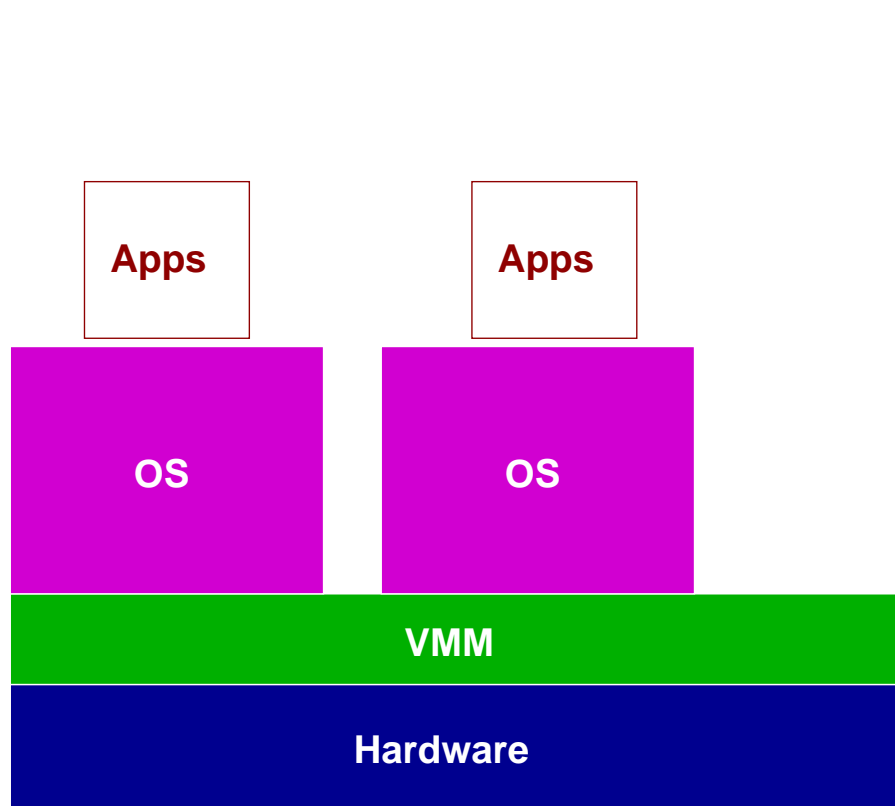
and

The University of New South Wales

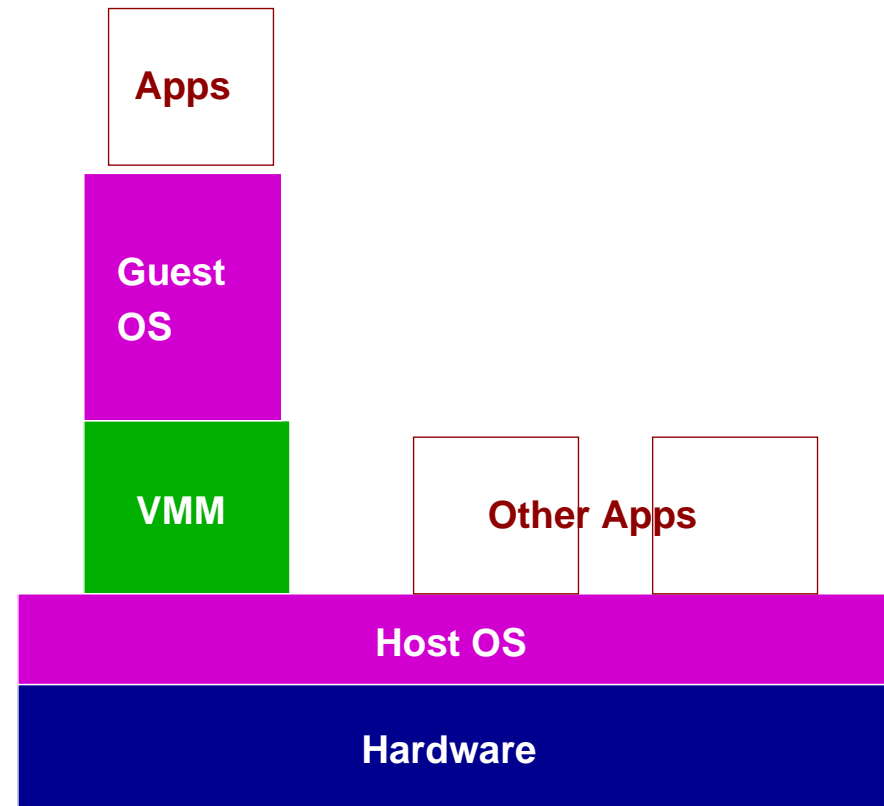
**April 2007**



# Virtual Machine Monitors



**Type I VMM  
(Native)**



**Type II VMM  
(Hosted)**



# The Core Virtualisation Problem

- ✓ Performing a privileged operation traps



# The Core Virtualisation Problem

- ✓ Performing a privileged operation traps
- ✗ Reading system state doesn't always trap



# The Core Virtualisation Problem

- ✓ Performing a privileged operation traps
- ✗ Reading system state doesn't always trap
- ✗ System and User state **not cleanly separated**
- ✗ Trapping is **slow**.



# Architecture Extensions for Virtualisation

- *Silverdale* extensions:
  - add *extra privilege level*
  - *All* changes to privileged state trap



# Type II VMMs on Linux

- UML on IA32



## Type II VMMs on Linux

- UML on IA32 — Heavily paravirtualised (so Linux only)
  - one host process per guest process
  - Poor security: don't run as root!



## Type II VMMs on Linux

- UML on IA32 — Heavily paravirtualised (so Linux only)
- VMware Workstation (IA32)
  - Rewrites binaries on the fly
  - Works with arbitrary guest



## Type II VMMs on Linux

- UML on IA32 — Heavily paravirtualised (so Linux only)
- VMware Workstation (IA32)
- LinuxOnLinux
  - One process per virtual processor
  - Some host assist patches
  - Reuse device etc., infrastructure for SKI Simulator



## LinuxOnLinux Aims

- IA-64
- High performance
- SMP; maybe NUMA
- Minimal changes to guest
- Direct access to hardware
- GDBable



# LinuxOnLinux Issues

- Virtualisation
- System Calls
- Memory Management
- I/O
- Security



# Afterburning

- Rewrite sensitive instructions at Assembly time.



# Afterburning

- Rewrite sensitive instructions at Assembly time.
- Hacks to guest prevent use on bare metal or simulator



# Afterburning

- Rewrite sensitive instructions at Assembly time.
- Hacks to guest prevent use on bare metal or simulator
- Development harder than it might be
  - At least until the Afterburner and Wedge are bug-free



## Afterburning

- Rewrite sensitive instructions at Assembly time.
- Hacks to guest prevent use on bare metal or simulator
- Development harder than it might be
  - At least until the Afterburner and Wedge are bug-free
- 5 days part time (4 h/day) work!!!!



# Some Linux-on-Linux results

	2p/0k ctxsw (usec)	Pipe (usec)	AF UNIX (usec)	TCP (usec)	File reread (MB/s)	Mm rere (MB)
L-o-L	252	490	859	540	1684	886
virt	1550	3071	5100	4060	185	887
Native	1.270	4.197	7.89	13.7	2273.5	889
Manual/Xen	2.720	6.68	11.6	18.8	2200	879
Auto/Xen	2.530	6.84	11.9	17.3	2200	879



- Better than before
- Main overheads are system call, I/O and context switch time



- Better than before — **But still too slow!**
- Main overheads are system call, I/O and context switch time



# System call handling





# System call handling



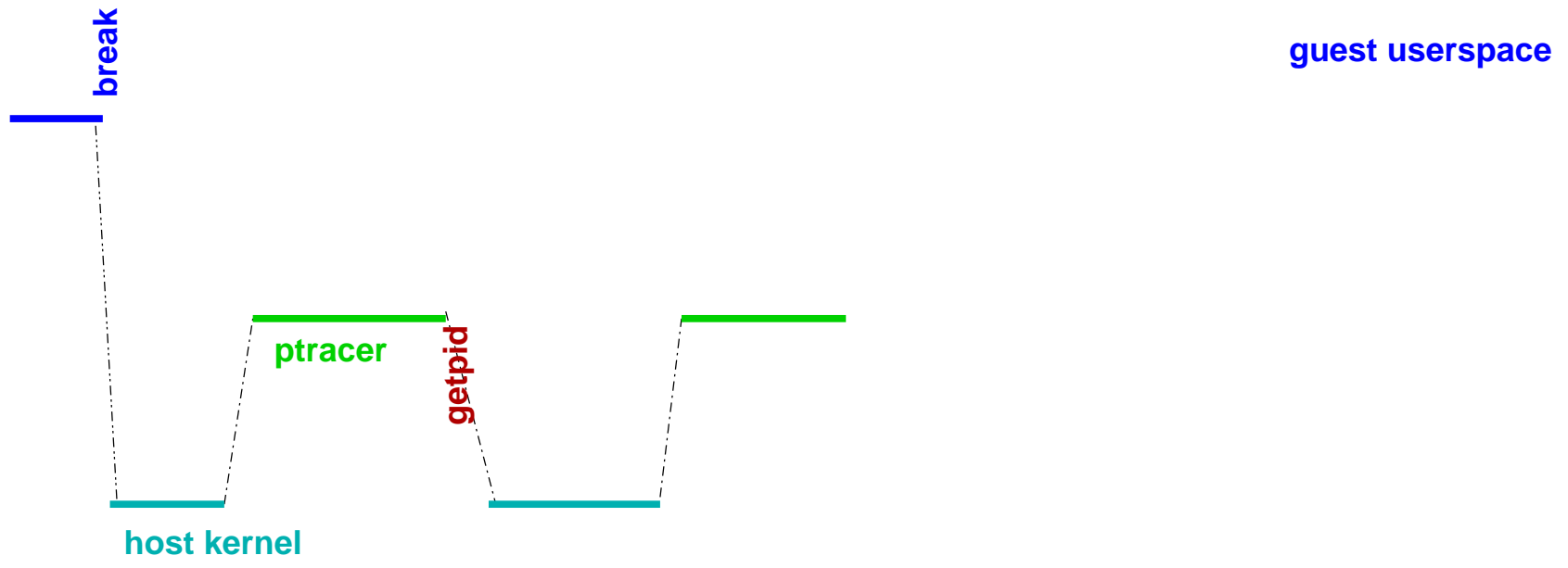


# System call handling



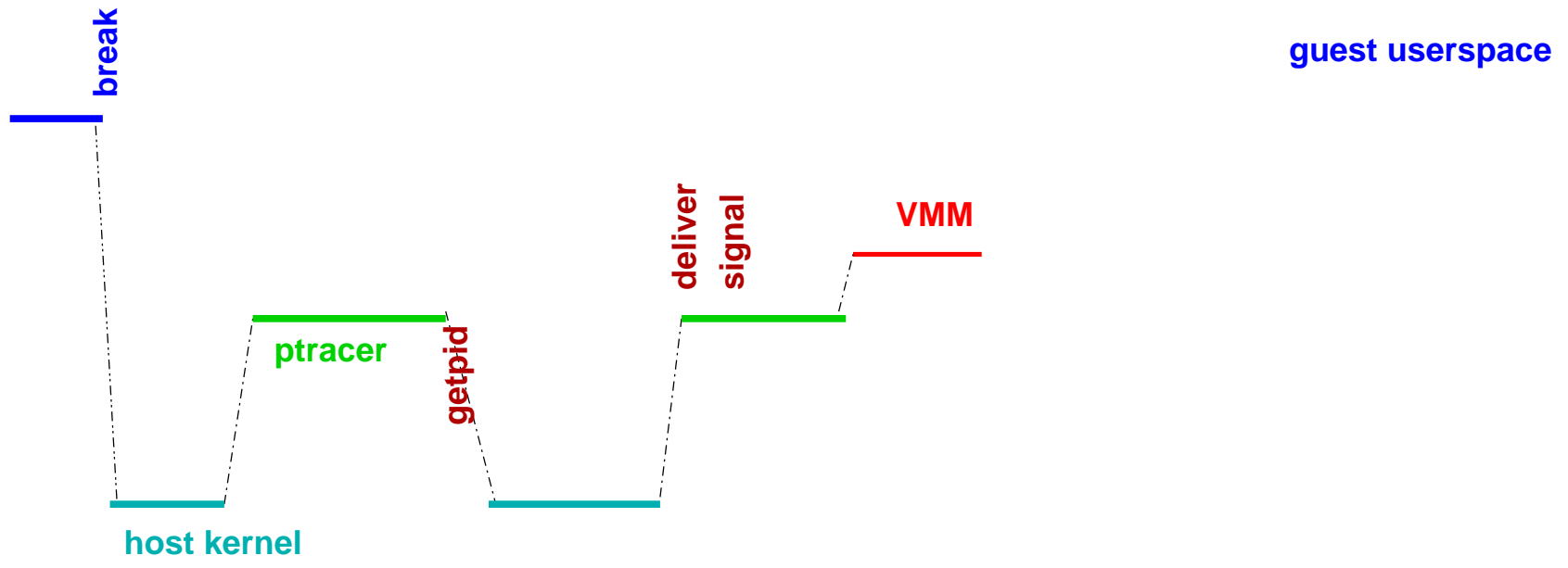


# System call handling



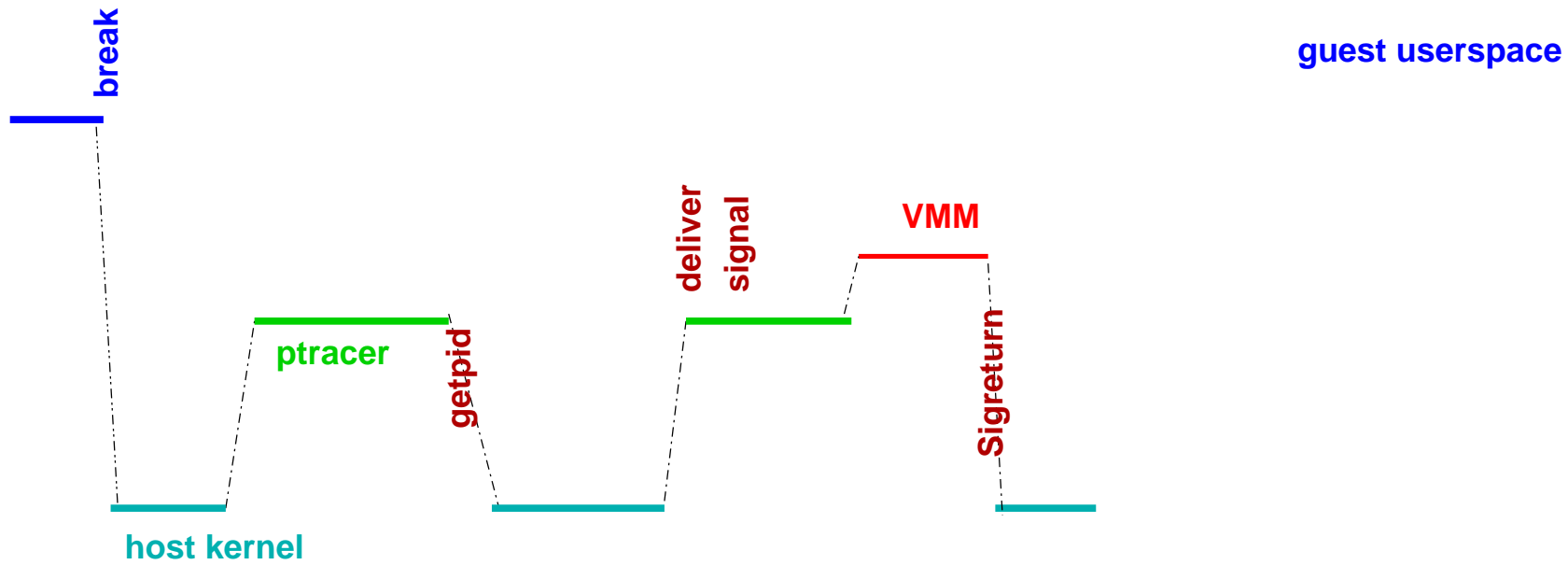


# System call handling



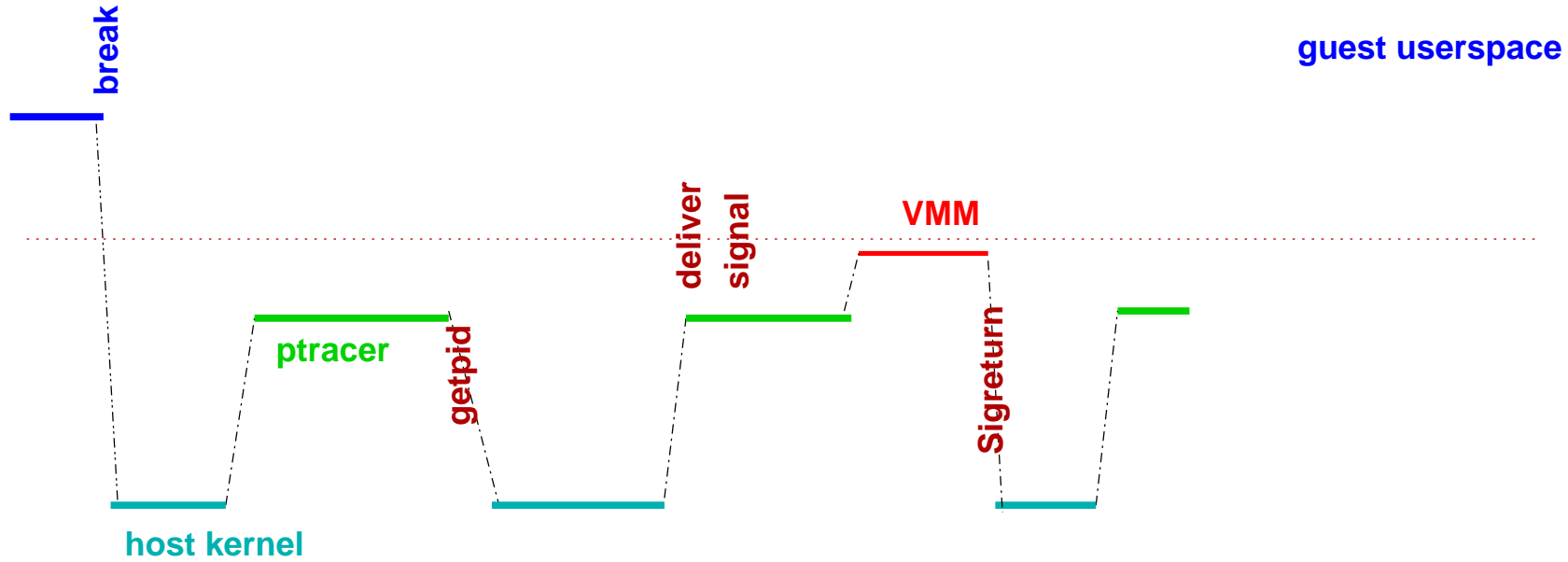


# System call handling



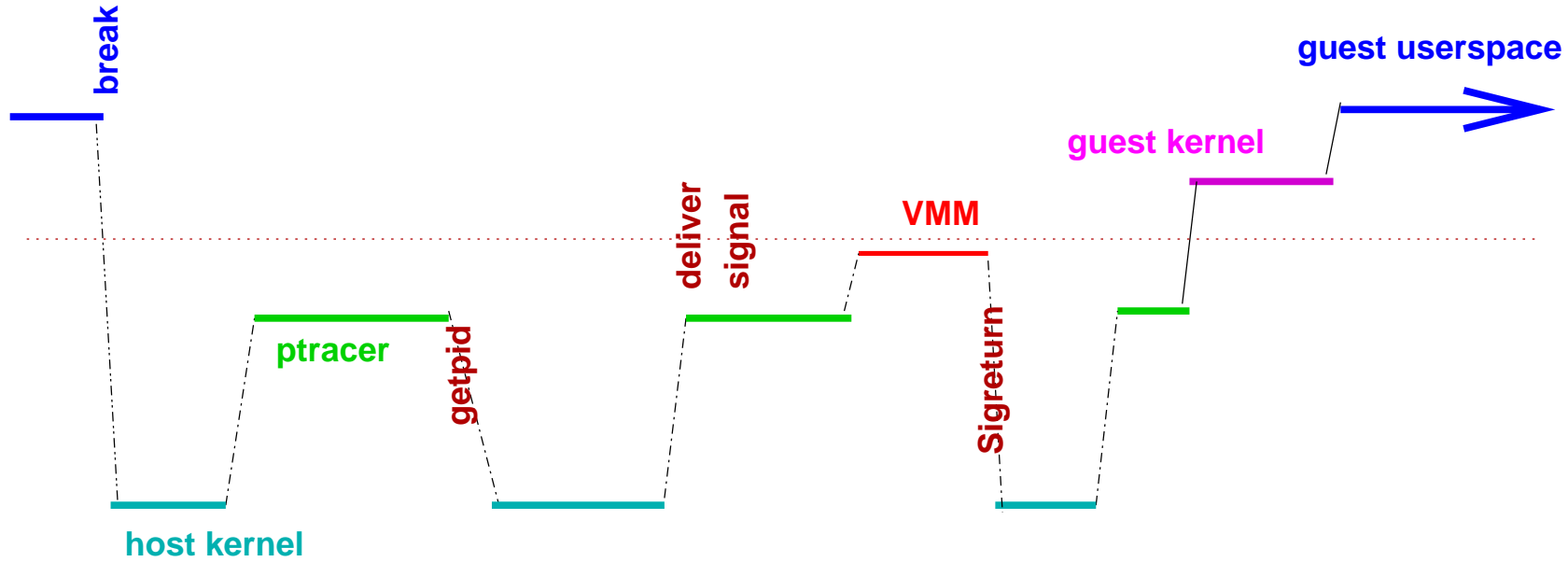


# System call handling



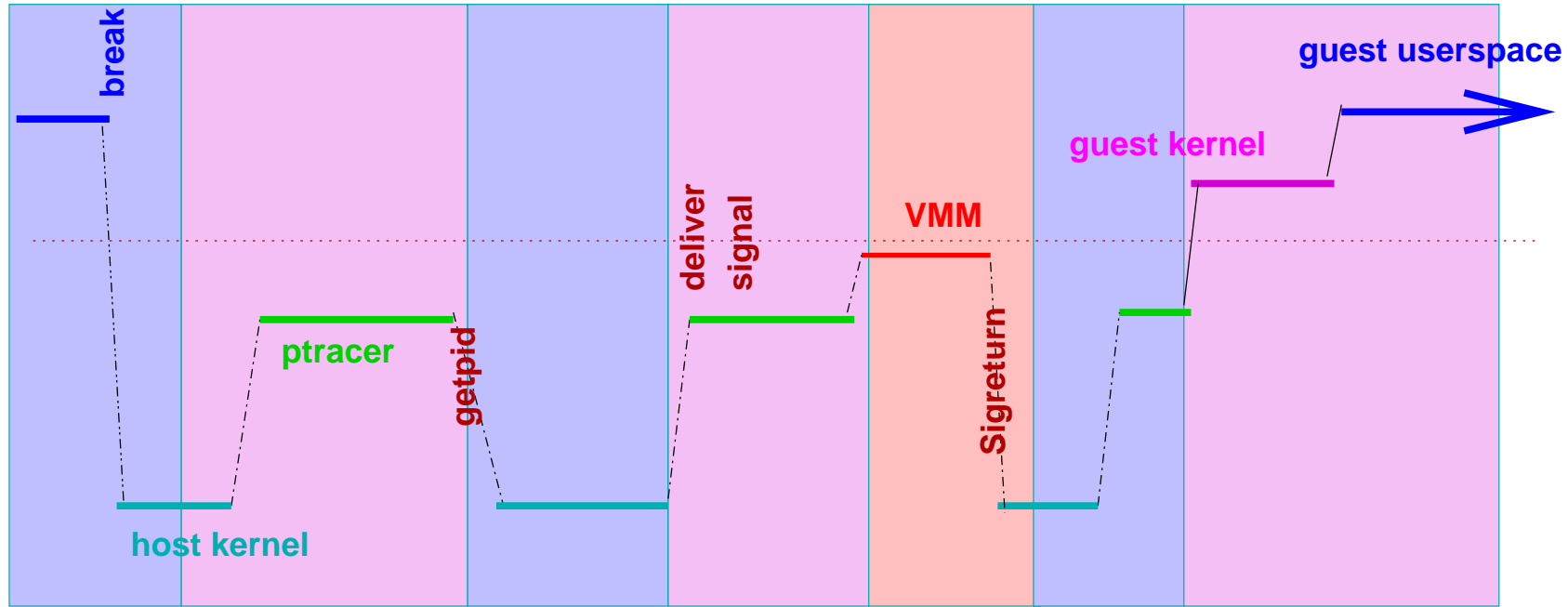


# System call handling





# System call handling





# Hack the host

- Restrict ptrace to ranges of addresses



# Hack the host

- Restrict ptrace to ranges of addresses
- Add **PT\_ONESHOT** flag



# Hack the host

- Restrict ptrace to ranges of addresses
- Add **PT\_ONESHOT** flag
- Add **PT\_NOSIGSTOP** flag



# Hack the host

- Restrict ptrace to ranges of addresses
- Add **PT\_ONESHOT** flag
- Add **PT\_NOSIGSTOP** flag
- Add way to set `psr.dfh` (necessary for correctness)



## More Improvements

- Fast system calls
  - 4 fewer context switches per syscall — ptracer mostly unused
  - Except for `clone`, `fork` etc
  - And legacy statically linked programmes

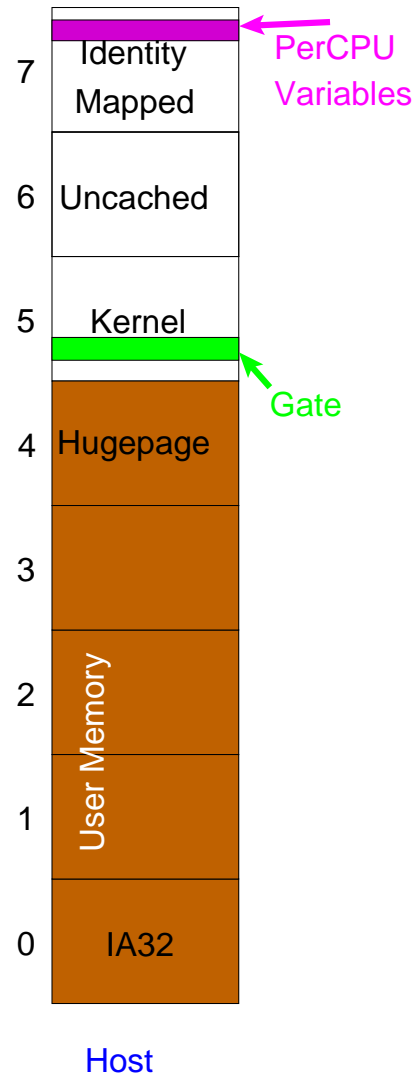


## More Improvements

- Fast system calls
- Ptrace improvements
  - Avoid ptrace overhead for Signals
  - Avoid ptrace overhead — full stop

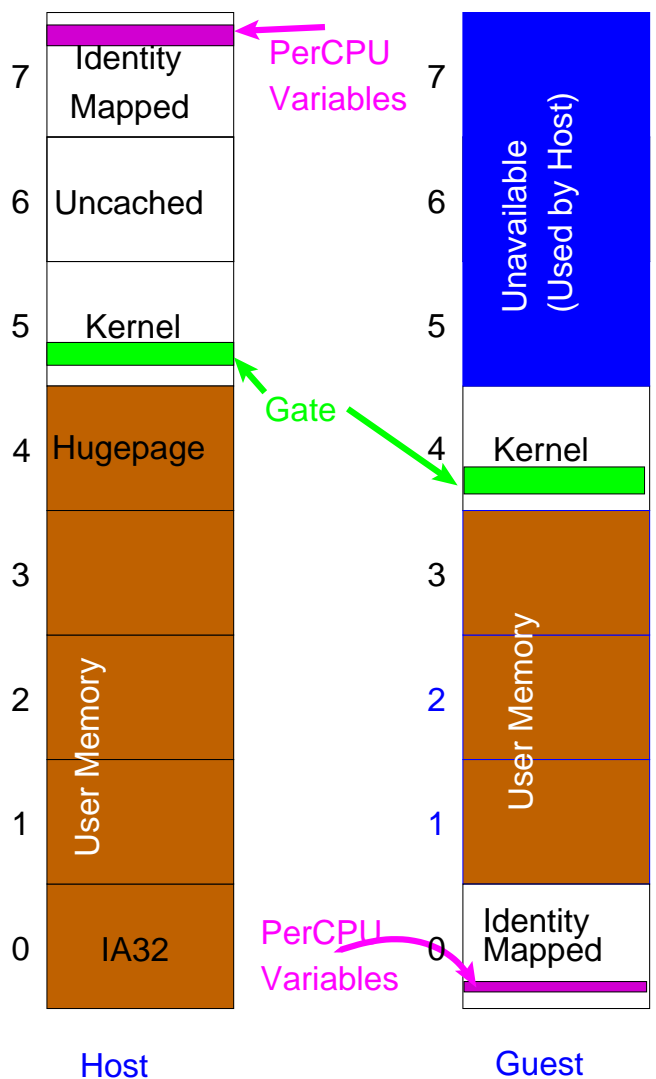


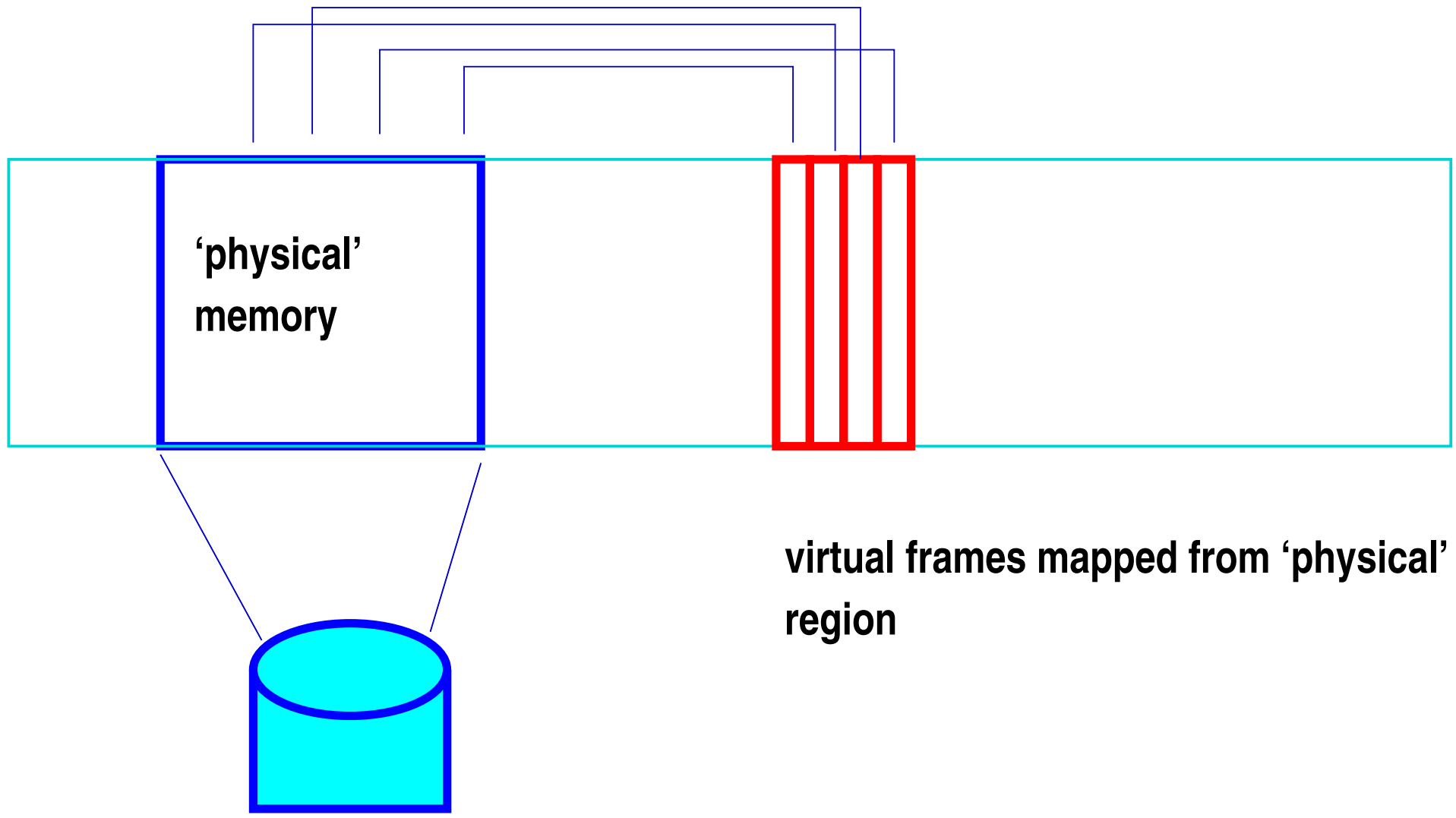
# Memory Management





# Memory Management



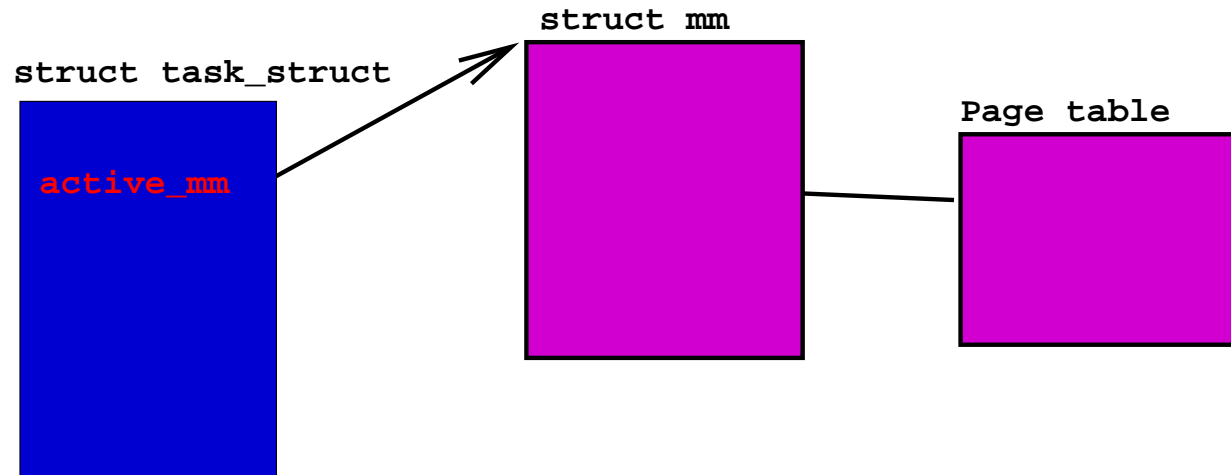


**'physical'  
memory**

**virtual frames mapped from 'physical'  
region**

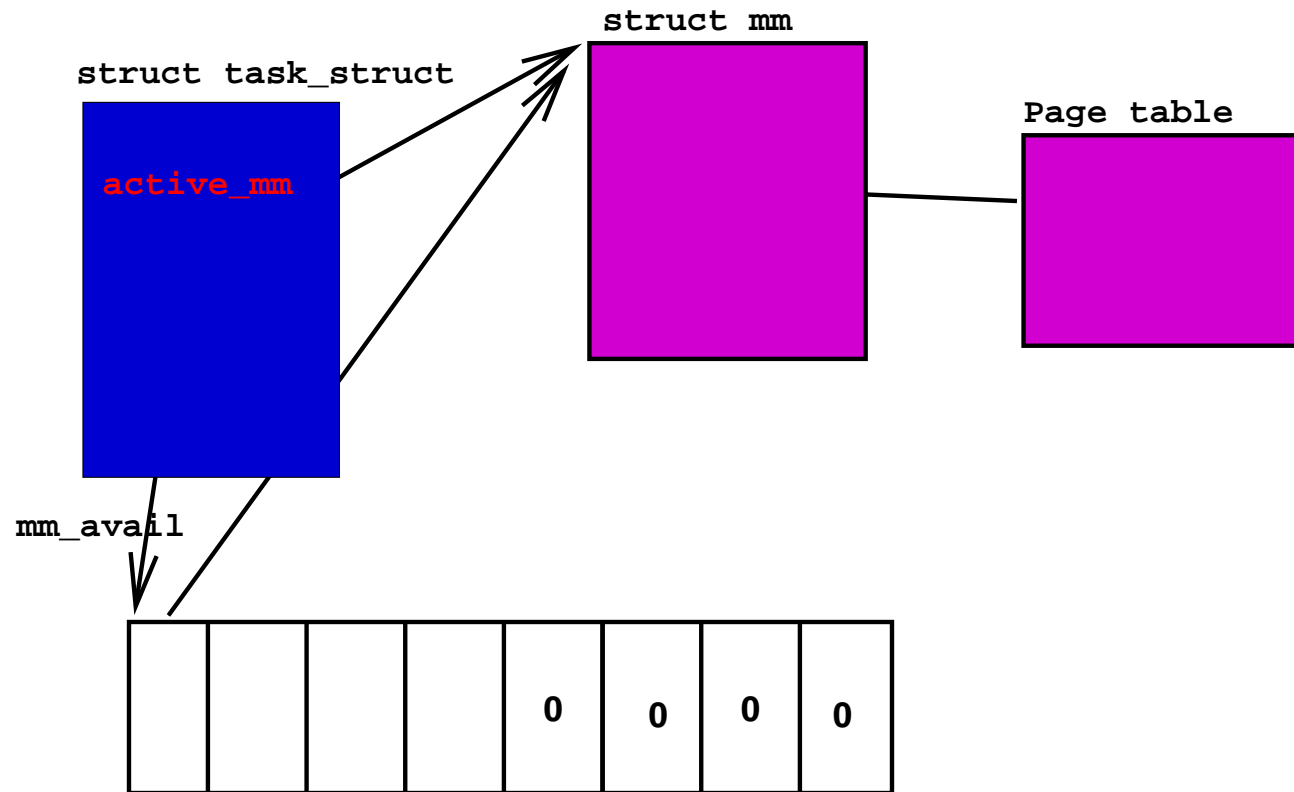


# The multi-address space hack



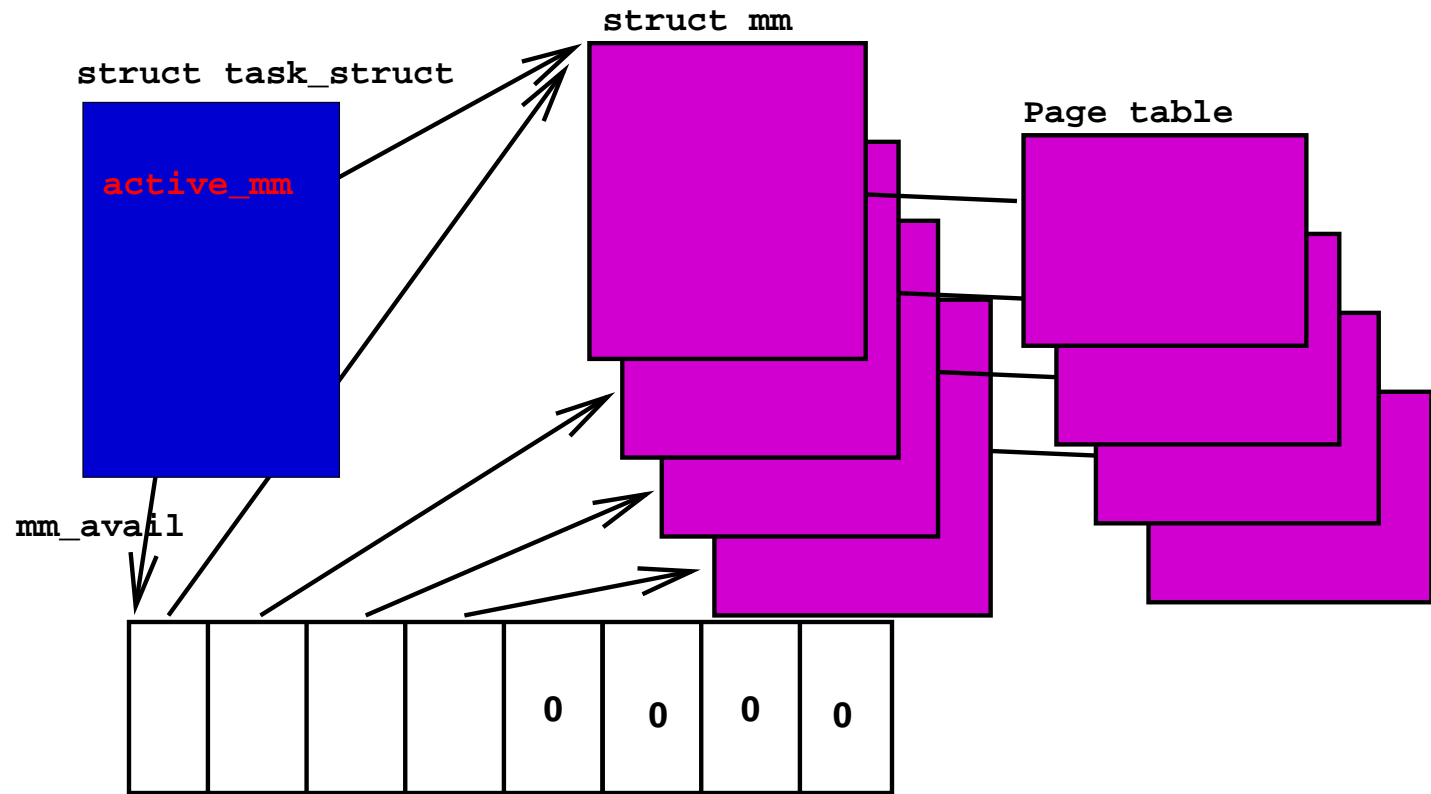


# The multi-address space hack





# The multi-address space hack





# Results

Kernel compilation time.

150m





# Results

Kernel compilation time.



←  
**PT\_ADDRANGE**



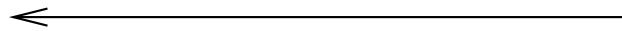
# Results

Kernel compilation time.

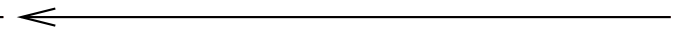
50m

98m

150m



**PT\_NOSIGSTOP**



**PT\_ADDRANGE**



# Results

Kernel compilation time.





# Results

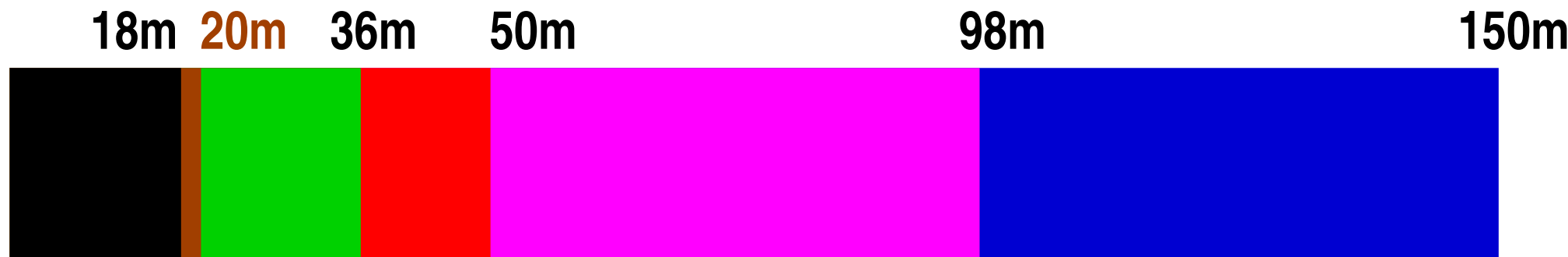
Kernel compilation time.



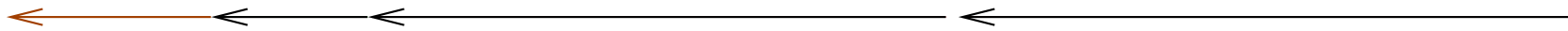


# Results

Kernel compilation time.



Native



multi-as

Paravirt

PT\_NOSIGSTOP

PT\_ADDRANGE



- Currently use SKI HP-simulator devices
  - Ethernet: `/dev/tap?` on the host
  - Scsi: `readv()/writev()` to file on host
  - Console: `stdin/stdout`.



# Asynchronous I/O

- Standard `simscsi` is *synchronous*
- Added:
  - tagged queuing
  - completion notification by interrupt



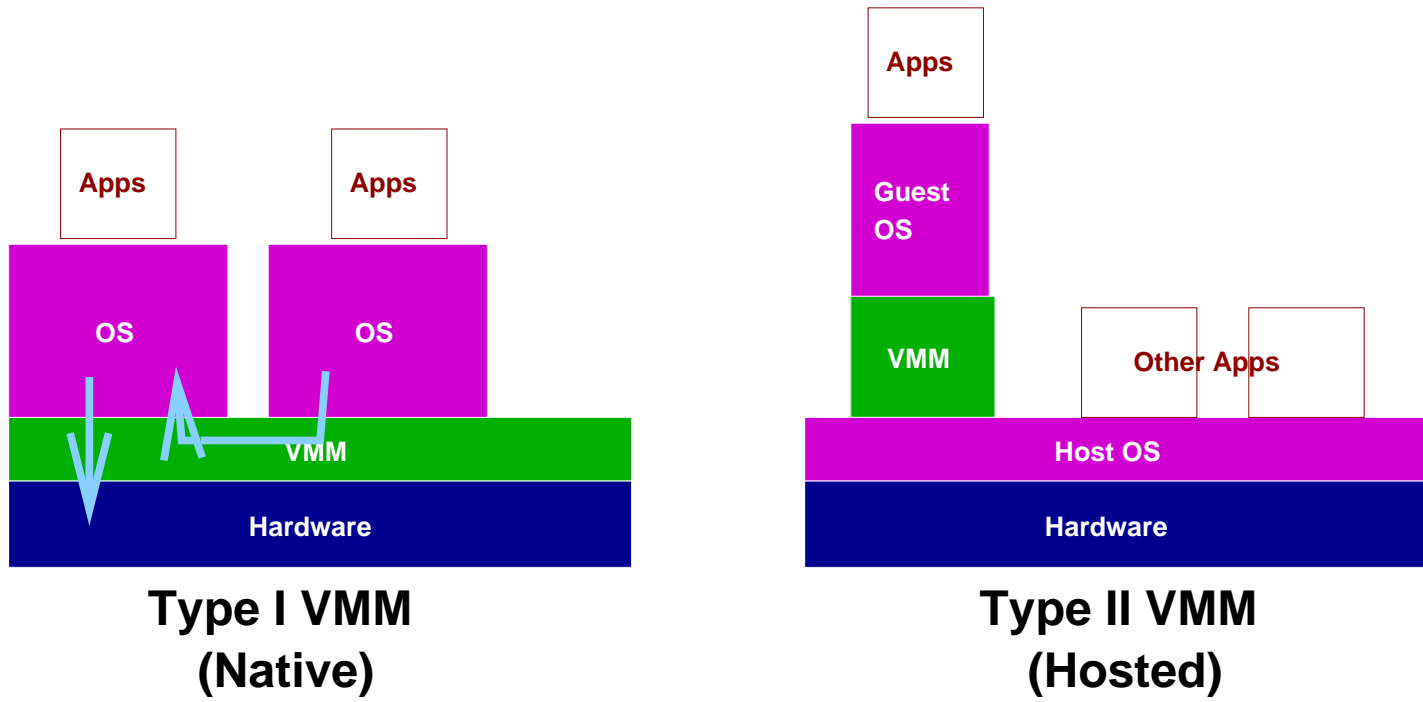
# Asynchronous I/O (cont)

## Results:

- No change for single-threaded benchmarks
- Slight improvement on SMP and multithreaded iozone.
- Overall around 86% of native performance.

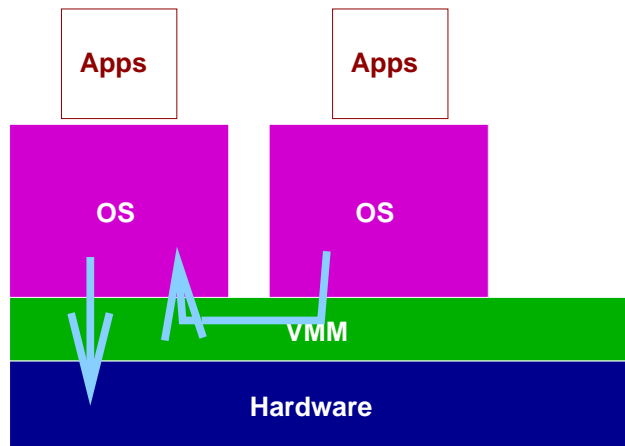


# Virtual I/O

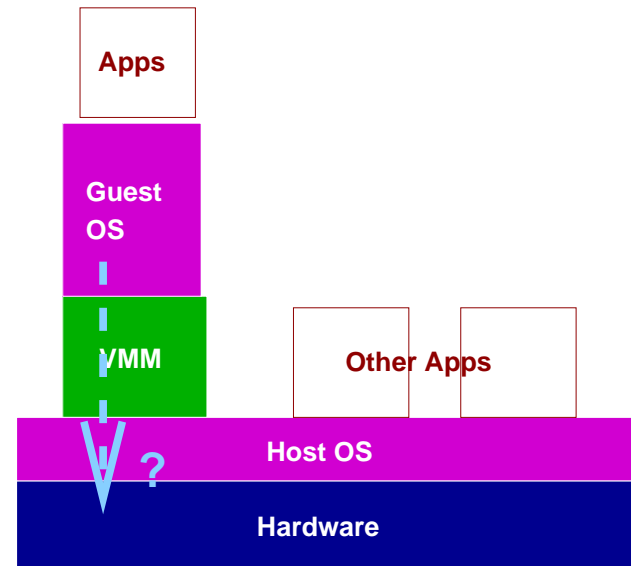




# Virtual I/O



Type I VMM  
(Native)



Type II VMM  
(Hosted)



- Use User-Level Driver framework developed at UNSW



- Use User-Level Driver framework developed at UNSW
- **Device Discovery**, IO-space access, Interrupts, DMA



- Use User-Level Driver framework developed at UNSW
- Device Discovery, **IO-space access**, Interrupts, DMA



- Use User-Level Driver framework developed at UNSW
- Device Discovery, IO-space access, **Interrupts**, DMA



- Use User-Level Driver framework developed at UNSW
- Device Discovery, IO-space access, Interrupts, **DMA**



# Security

## Major Holes!

- Host gate page mapped into guest process AS
- pseudo physical memory in guest process AS
- guest kernel in guest process address space



# Security

## Major Holes! But Same as UML

- Host gate page mapped into guest process AS
- pseudo physical memory in guest process AS
- guest kernel in guest process address space



## Current state

- Performance
- SMP; maybe NUMA
- Minimal changes to guest
- Direct access to hardware
- GDBable



## Current state

- Performance — around 10% worse than native
- SMP; maybe NUMA
- Minimal changes to guest
- Direct access to hardware
- GDBable



## Current state

- Performance — around 10% worse than native
- SMP; maybe NUMA — One process per virtual processor; scales with host
- Minimal changes to guest
- Direct access to hardware
- GDBable



## Current state

- Performance — around 10% worse than native
- SMP; maybe NUMA — One process per virtual processor; scales with host
- Minimal changes to guest — around 500 line patch
- Direct access to hardware
- GDBable



## Current state

- Performance — around 10% worse than native
- SMP; maybe NUMA — One process per virtual processor; scales with host
- Minimal changes to guest — around 500 line patch
- Direct access to hardware — Coming soon
- GDBable



## Current state

- Performance — around 10% worse than native
- SMP; maybe NUMA — One process per virtual processor; scales with host
- Minimal changes to guest — around 500 line patch
- Direct access to hardware — Coming soon
- GDBable — doable but awkward at present



# Current and Future work

- Understand current performance problems



## Current and Future work

- Understand current performance problems — suspect signal handling overhead



## Current and Future work

- Understand current performance problems — suspect signal handling overhead
- Analyse security



## Current and Future work

- Understand current performance problems — suspect signal handling overhead
- Analyse security — and fix it!



## Current and Future work

- Understand current performance problems — suspect signal handling overhead
- Analyse security — and fix it!
- I/O performance
  - Direct I/O
  - Zero-copy /dev/tun???



## Current and Future work

- Understand current performance problems — suspect signal handling overhead
- Analyse security — and fix it!
- I/O performance
- Gate page for VMM calls
  - Reduce cost of many ops currently using **break**



This work was sponsored by HP, SGI, the Australian Research Council, UNSW and NICTA.

