



Hardware-Supported Virtualization on ARM

Prashant Varanasi and Gernot Heiser
NICTA and University of New South Wales, Sydney



Australian Government
Department of Broadband, Communications
and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners



Australian
National
University

UNSW
THE UNIVERSITY OF NEW SOUTH WALES



Griffith
UNIVERSITY



QUT
Queensland University of Technology



THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

THE UNIVERSITY OF
SYDNEY

Queensland
Government

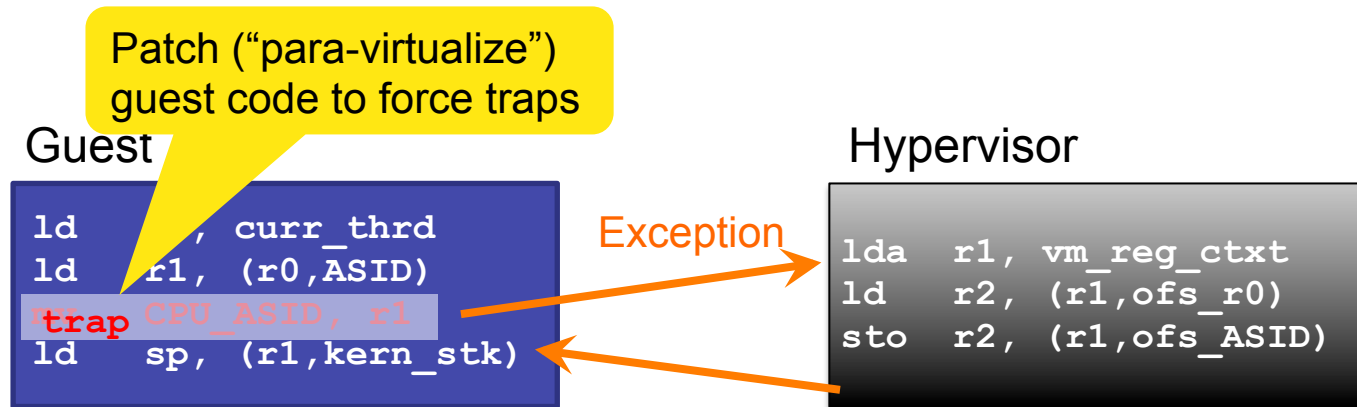
Virtualization for ARM Processors



- Virtualization is becoming prevalent in embedded systems
 - Open Kernel Labs (and others) have been selling hypervisors for years
- ARM announced hardware support in September 2010
 - Architecture specification released late 2010
 - Simulators released to partners in late 2010
 - Silicon samples expected late 2011
 - Products expected to ship in late 2012
- We built first complete hypervisor prototype for this architecture
 - ... able to run unmodified Linux binaries

Pure vs Para-Virtualization

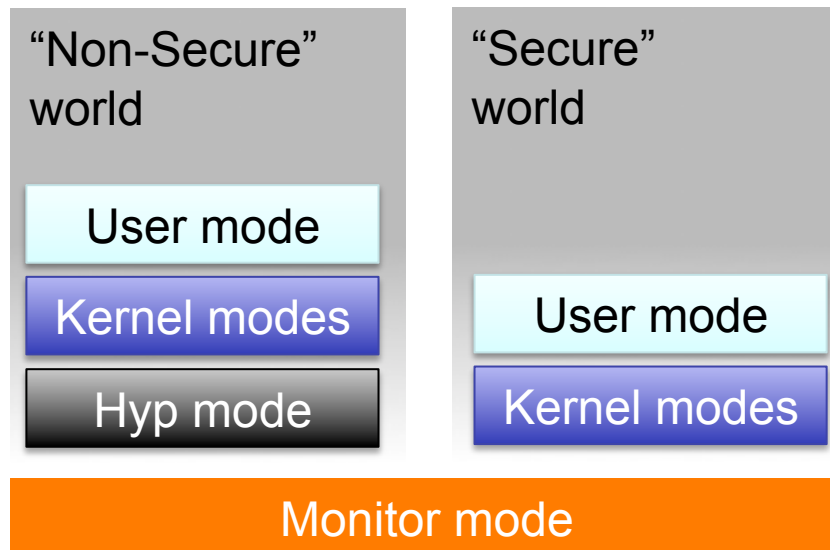
- Hypervisor must *emulate* sensitive instructions executed by OS



- On ARM (as on x86) not all sensitive instructions trap
⇒ Para-virtualization
 - Significant engineering cost
- Hardware extensions support pure virtualization
 - Ability to run unmodified binaries

ARM Virtualization Extensions (1)

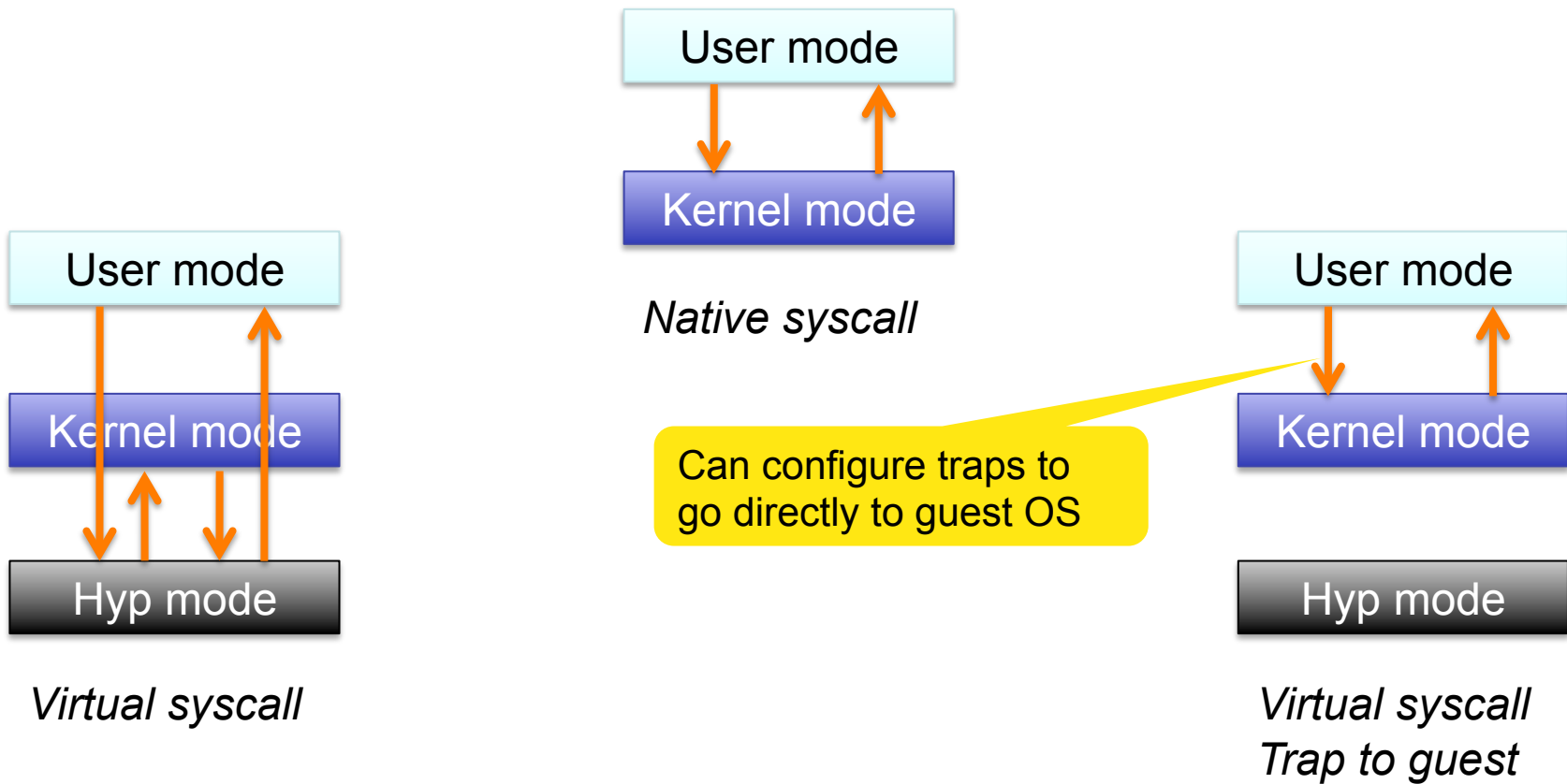
Hyp mode



- New privilege level
 - Strictly higher than kernel
 - Virtualizes or traps *all* sensitive instructions
 - Only available in ARM TrustZone “non-secure” mode
- Note: different from x86
 - VT-x “root” mode is orthogonal to x86 protection rings

ARM Virtualization Extensions (2)

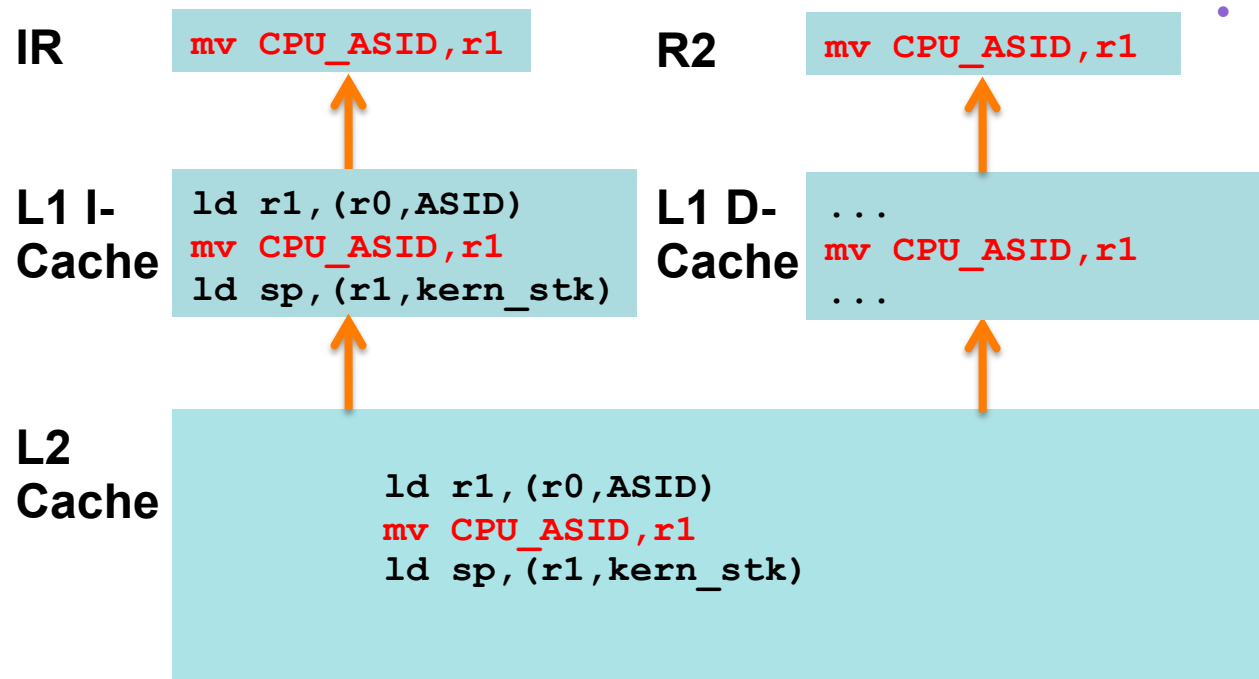
Configurable Traps



ARM Virtualization Extensions (3)

Emulation

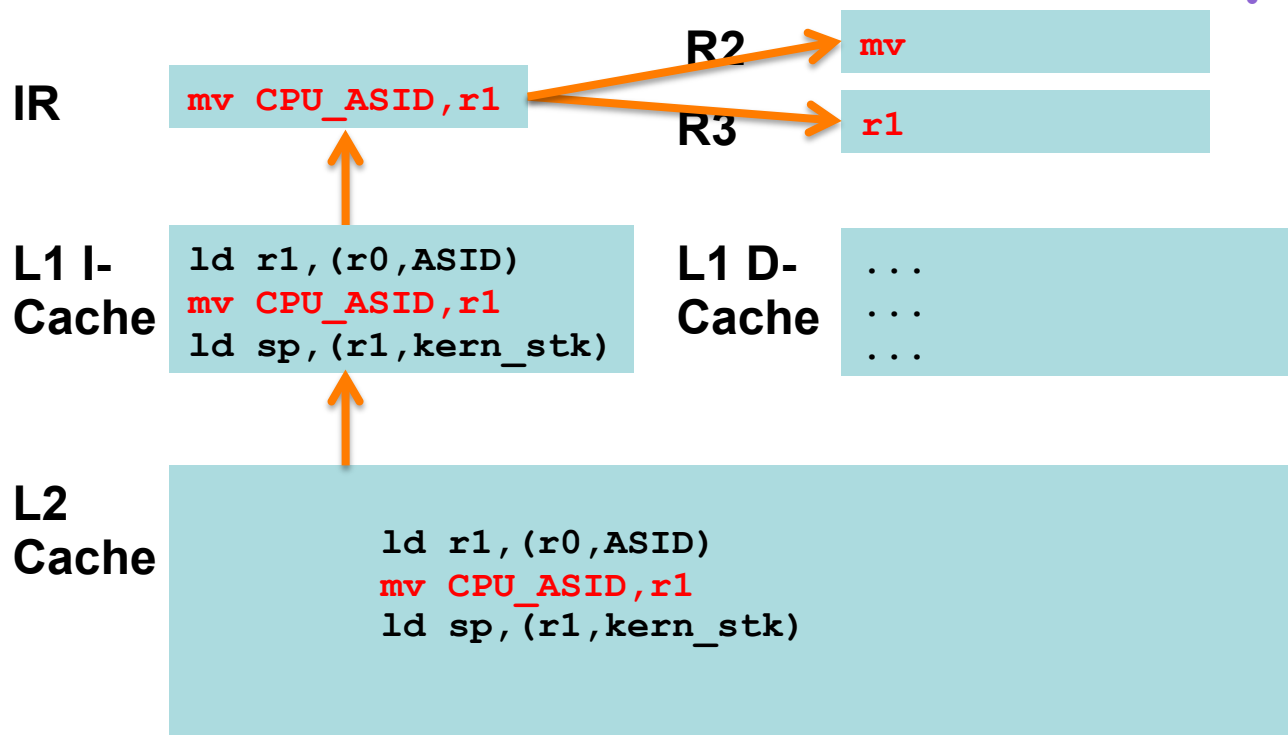
- 1) Load faulting instruction
 - Compulsory L1-D miss!
- 2) Decode instruction
 - Complex logic
- 3) Emulate instruction
 - Usually straightforward



ARM Virtualization Extensions (3)

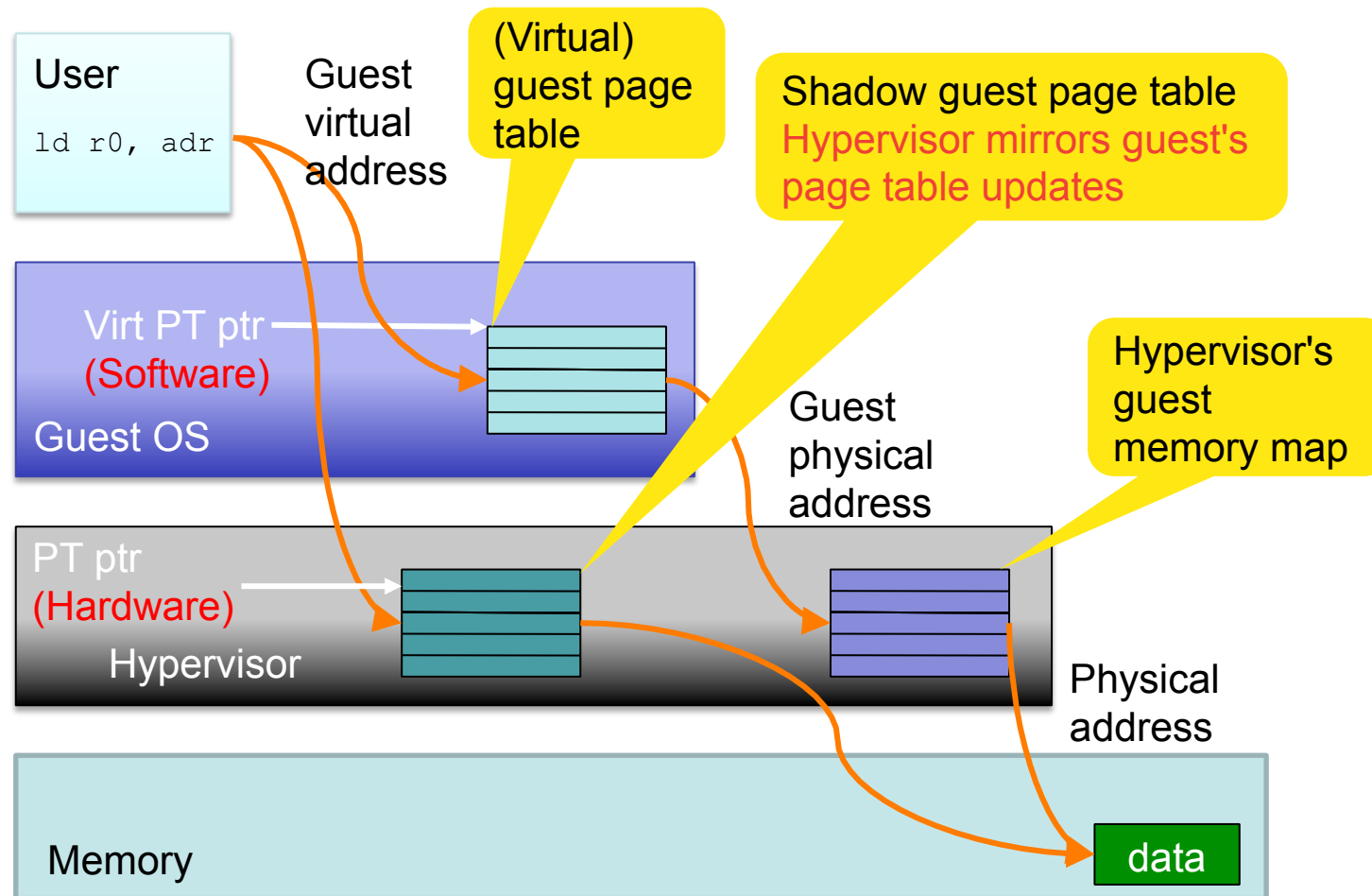
Emulation Support

- HW decodes instruction
 - No L1 miss
 - No software decode
- SW emulates instruction
 - Usually straightforward



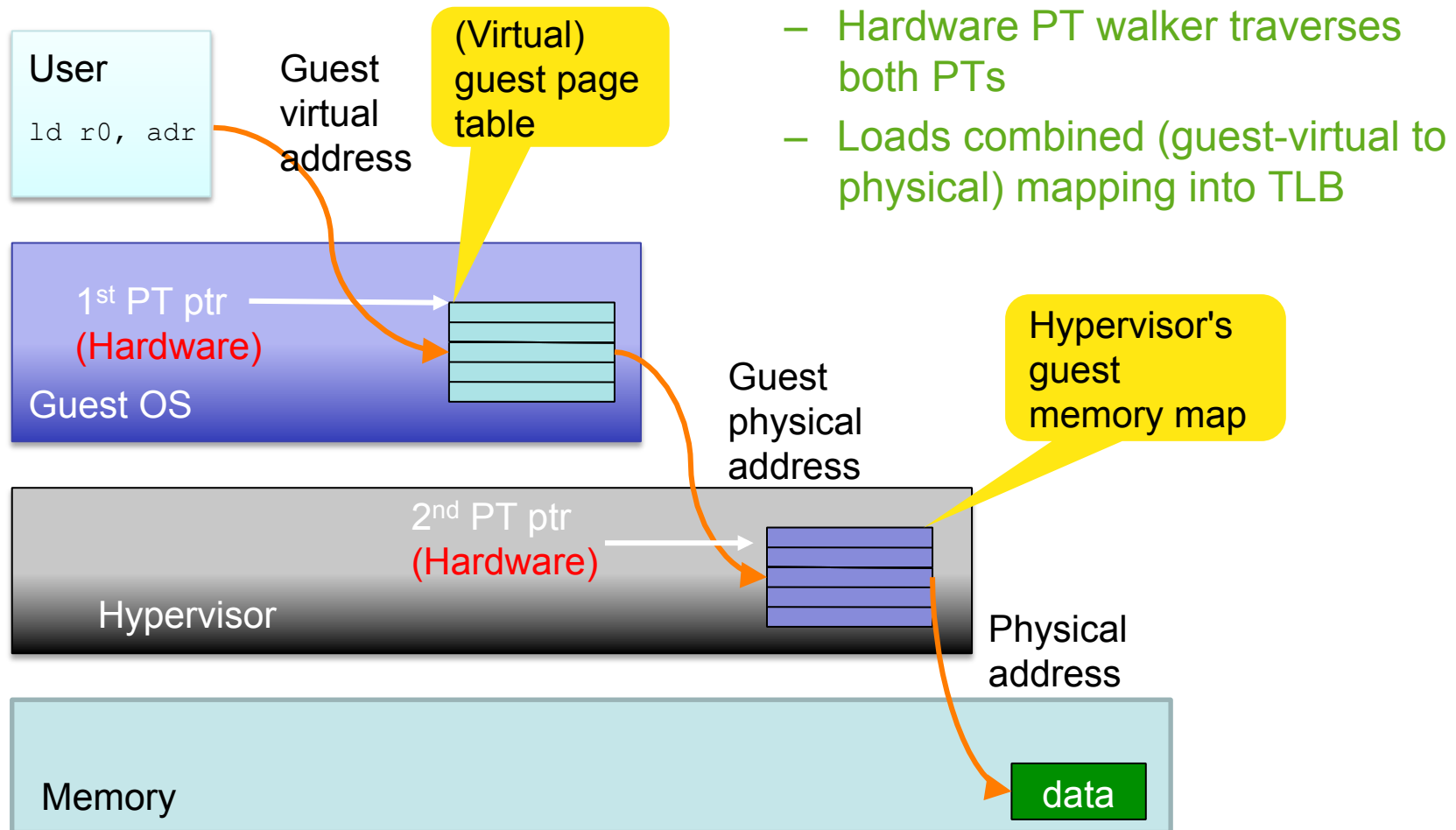
ARM Virtualization Extensions (4)

Page table virtualization through shadow page tables



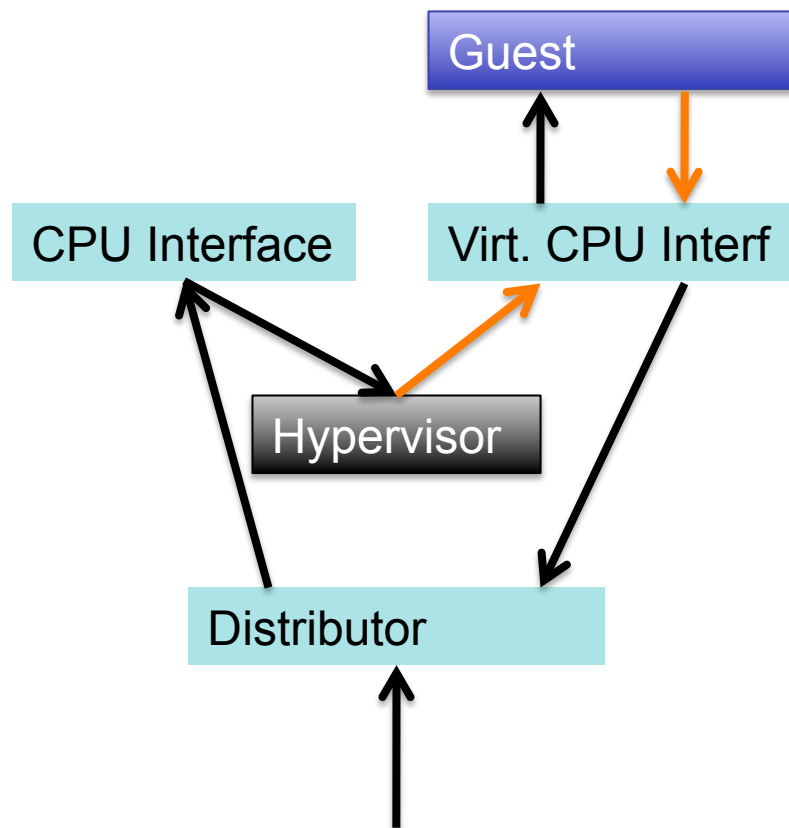
ARM Virtualization Extensions (4)

Page table virtualization through 2-stage translation



ARM Virtualization Extensions (5)

Virtual Interrupts



- ARM has 2-part IRQ controller
 - Global “distributor”
 - Per-CPU “interface”
- New H/W “virt. CPU interface”
 - Mapped to guest
 - Used by HV to forward IRQ
 - Used by guest to acknowledge
- Reduces hypervisor entries for interrupt virtualization

Hypervisor Size



Hypervisor	ISA	Type	Kernel	User
OKL4	ARMv7	para-virtualization	9.8 kLOC	0
<i>This work</i>	ARMv7	pure virtualization	6 kLOC	0
Nova	x86	pure virtualization	9 kLOC	27 kLOC

- Size (& complexity) reduced about 40% wrt to para-virtualization
- Much smaller than x86 pure-virtualization hypervisor
 - Mostly due to greatly reduced need for instruction emulation

Overheads (Estimated)

Operation	Pure virtualization		Para-virtualiz.
	Instruct	Cycles (est)	Cycles (approx)
Guest system call	0	0	300
Hypervisor entry + exit	120	650	150
IRQ entry + exit	270	900	300–400?
Page fault	356	1500	700
Device emul.	249	1040	N/A
Device emul. (accel.)	176	740	N/A
World switch	2824	7555	200

- No overhead on regular (virtual) syscall – unlike para-virtualization
 - Invoking hypervisor 500–1200 cycles (0.6–1.5 μ s) more than para
 - World switch in ~ 10 μ s compared to 0.25 μ s for para
- ⇒ Trade-offs differ

Experience

- Well-designed virtualization architecture \Rightarrow simple hypervisor
 - Emulation simplified by hardware support
- Especially IRQ handling simpler and much faster than on x86
- Hypervisor-entry/exit costs much higher than with para-virtualization
 - On-going need for para-virtualization, especially for device drivers
 - Para-virtualized drivers can co-exist with purely-virtualized guest
- World-switch is expensive
 - Due to large amount of extra privileged state
 - Performance challenge for shared drivers
 - However, this is highly sensitive to implementation of hardware
- More details in paper!

<mailto:gernot@nicta.com.au>

Google: “ertos”