

---

# Virtualisation Case Study: Itanium and vNUMA

Matthew Chapman

University of NSW, National ICT Australia, HP Labs

matthewc@cse.unsw.edu.au

UNSW



i n v e n t

---

## ITANIUM

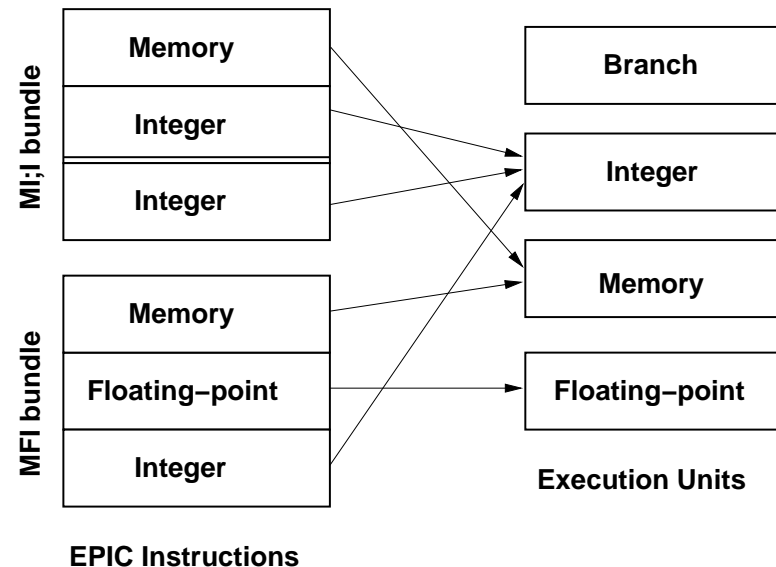
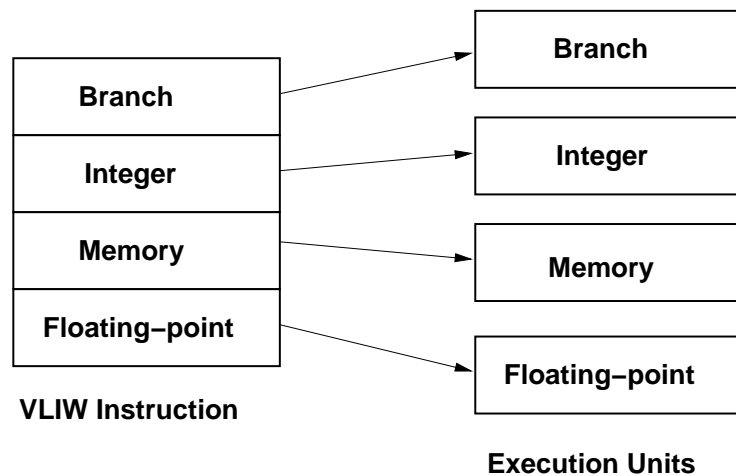
- High-performance processor architecture
- Also known as IA-64
- Joint venture between HP and Intel
- *Not* the same instruction set as AMD64/EM64T
- *Explicitly Parallel Instruction-Set Computing*
- Very good floating-point performance

2000:	Itanium (Merced)	0.18 $\mu\text{m}$ , 733/800Mhz
2002:	Itanium II (McKinley)	0.18 $\mu\text{m}$ , 900Mhz/1Ghz
2003:	Itanium II (Madison)	0.13 $\mu\text{m}$ , 1.3-1.6Ghz
2005:	Montecito	0.09 $\mu\text{m}$ , 1.8-2Ghz

---

# EXPLICITLY PARALLEL INSTRUCTION-SET COMPUTING (EPIC)

- Goal to increase *instructions per cycle*
  - Itanium can have similar performance to x86 at a lower clock speed
- Based on *Very Long Instruction Word (VLIW)*
- Explicit parallelism in instruction set
- Simplified instruction decode and issue
- Scheduling decisions made by compiler



---

## EXAMPLE

Load and add three numbers in assembly code:

```
ld8    r4 = (r1)
ld8    r5 = (r2)
ld8    r6 = (r3)
;;
add    r7 = r4, r5
;;
add    r8 = r6, r7
;;
```

---

Resulting instructions:

<b>MMI</b>	<b>ld8</b>	<b>r4 = (r1)</b>	
	<b>ld8</b>	<b>r5 = (r2)</b>	
	<b>nop.i</b>	<b>0</b>	
<b>M;MI;</b>	<b>ld8</b>	<b>r6 = (r3)</b>	
	<b>::</b>		
	<b>add</b>	<b>r7 = r4, r5</b>	<b>// Arithmetic on M too</b>
	<b>nop.i</b>	<b>0</b>	
	<b>::</b>		
<b>M;MI</b>	<b>add</b>	<b>r8 = r6, r7</b>	
	<b>::</b>		
	<b>nop.m</b>	<b>0</b>	
	<b>nop.i</b>	<b>0</b>	

---

A better way:

<b>MMI;</b>	<b>ld8</b>	<b>r4 = (r1)</b>
	<b>ld8</b>	<b>r5 = (r2)</b>
	<b>nop.i</b>	<b>0</b>
	<b>::</b>	

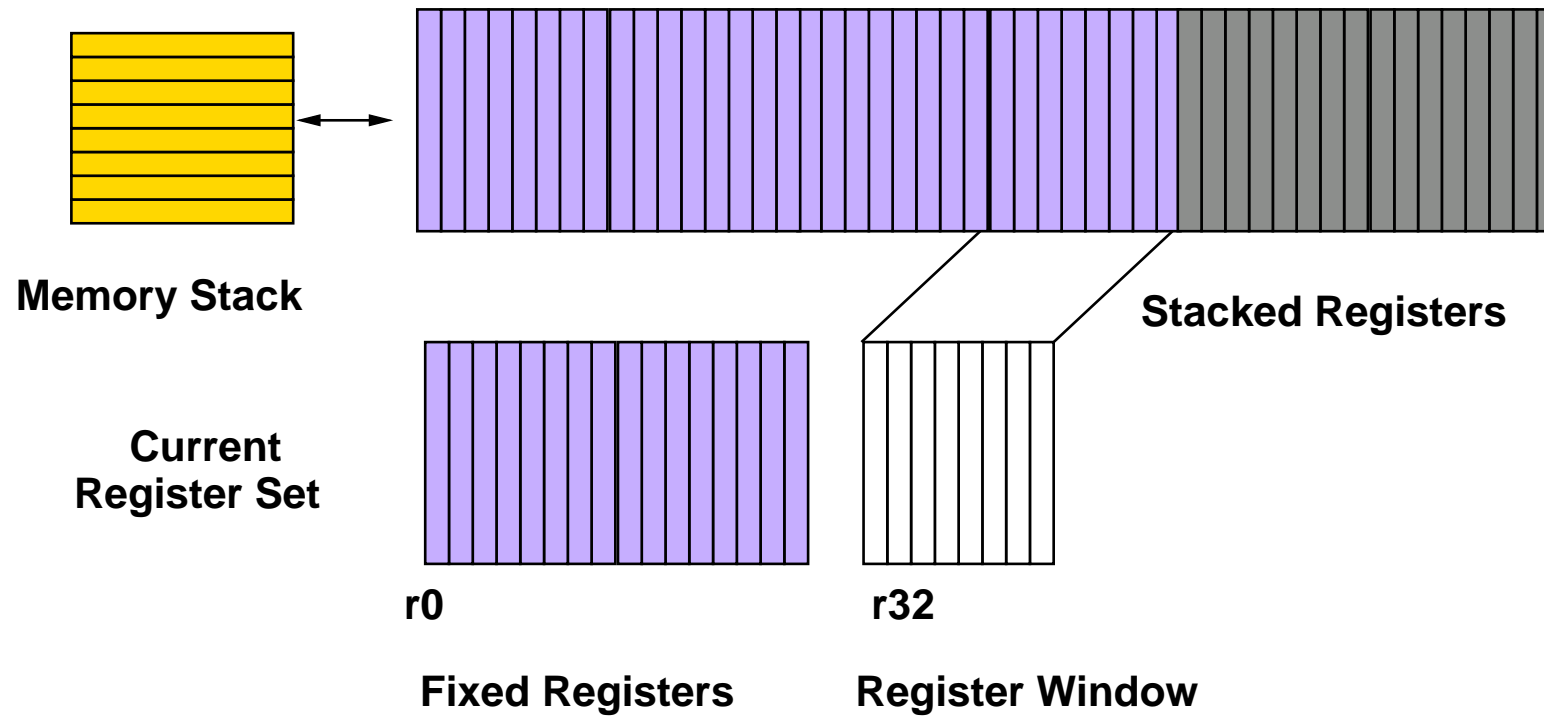
<b>MI;l;</b>	<b>ld8</b>	<b>r6 = (r3)</b>
	<b>add</b>	<b>r7 = r4, r5</b>
	<b>::</b>	
	<b>add</b>	<b>r8 = r6, r7</b>
	<b>::</b>	

---

# ITANIUM REGISTER SET

Large general register file

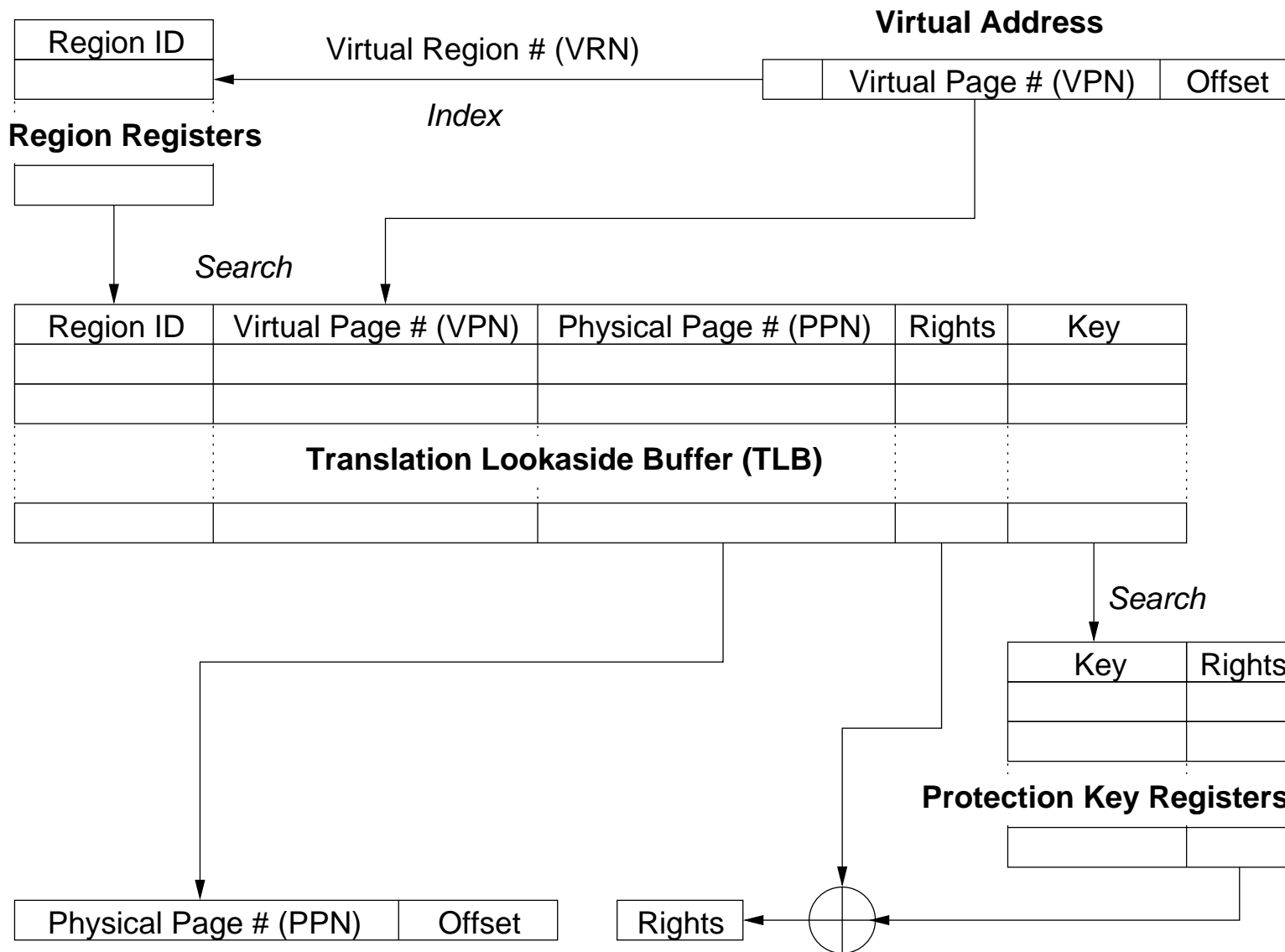
→ partially managed by the *register stack engine* (RSE)



---

## ITANIUM MMU

- 64-bit addressing
- Process address spaces are constructed from 8x61-bit *regions*
- Regions are 61-bit segments of a global 85-bit address space
- Page sizes from 4kB to 4GB
- Protection keys
- Choice of two hardware-walked page table formats



---

## VIRTUALISING THE ITANIUM

- We demote the OS from privilege level 0 (most privileged) to 1
- Which instructions behave differently, and do not trap?
  - *Sensitive instructions*

---

# ITANIUM INSTRUCTION SET

## Fixed point arithmetic

add	mix	pavg	pshl	shrp
addp4	mov	pavgsub	pshladd	sub
and	mov	pcmp	pshr	sxt
andcm	mov pr	pmax	pshradd	tbit
cmp	movl	pmin	psub	tnat
cmp4	mux	pmpy	shl	unpack
czx	or	pmpyshr	shladd	xchg
dep	pack	popcnt	shladdp4	xor
extr	padd	psad	shr	zxt

---

## Floating point arithmetic

fabs	fcvt	fnma	fpms	fsetc
fadd	fma	fnmpy	fpneg	fsub
famax	fmax	fnorm	fpnegabs	fswap
famin	fmerge	for	fpnma	fsxt
fand	fmin	fpabs	fpnmpy	fxor
fandcm	fmix	fpack	fprcpa	getf
fchkf	fmpy	fpamax	fprsqta	mov fr
fclass	fms	fpmerge	frcpa	setf
fclrf	fneg	fpmin	frsqta	xma
fcmp	fnegabs	fpmpy	fselect	xmpy

---

## Branch

br

brl

brp

chk

mov br

## Register Stack Engine

alloc

clrrb

cover<sup>a</sup>

flushrs

invala

loadrs

mov ar.bspstore

mov ar.rsc<sup>b</sup>

---

<sup>a</sup>side-effect at privilege level 0 when psr.ic=0

<sup>b</sup>ar.rsc contains real privilege level

---

## Memory access

cmpxchg	fwb	ldfp	probe	sync
<b>fc</b> <sup>a</sup>	ld	lfetch	st	
fetchadd	ldf	mf	stf	

## Other unprivileged

break	mov ar	mov ip	nop	srlz
epc	<b>mov cpuid</b>	mov um	rum	
hint	<sup>b</sup>		sum	

---

<sup>a</sup>bypasses protection check at privilege level 0

<sup>b</sup>returns cpuid of real processor

---

## Memory management (privileged)

itc	mov pkr	ptc	tak	tpa
itr	mov rr	ptr	thash <sup>a</sup>	ttag <sup>a</sup>

## Other privileged

bsw	mov dbr	mov pmc	mov psr	rsm
mov cr	mov ibr	mov pmd <sup>b</sup>	rfi	ssm

---

<sup>a</sup>not privileged

<sup>b</sup>unprivileged reads return 0 instead of trapping

---

## DEALING WITH NON-VIRTUALISABLE INSTRUCTIONS

- Paravirtualisation
  - Modify guest code
- Static translation (at compile-time)
- Dynamic translation (at runtime)

---

## OTHER ISSUES

- Fixed set of privilege levels (0..3)
  - Guest pl 0..3 now mapped onto 1..3

---

## OTHER ISSUES

- Fixed set of privilege levels (0..3)
  - Guest pl 0..3 now mapped onto 1..3
- Exception handlers live in virtual memory
  - Need to co-ordinate address with guest kernel

---

## OTHER ISSUES

- Fixed set of privilege levels (0..3)
  - Guest pl 0..3 now mapped onto 1..3
- Exception handlers live in virtual memory
  - Need to co-ordinate address with guest kernel
- Separate control over instruction/data/register stack translation
  - e.g. data physical, register stack virtual
  - Difficult to replicate in virtual mode

---

## OTHER ISSUES

- Fixed set of privilege levels (0..3)
  - Guest pl 0..3 now mapped onto 1..3
- Exception handlers live in virtual memory
  - Need to co-ordinate address with guest kernel
- Separate control over instruction/data/register stack translation
  - e.g. data physical, register stack virtual
  - Difficult to replicate in virtual mode
- RSE not completely virtualisable
  - Partially loaded frames cannot exist outside p10

---

## VANDERPOOL TECHNOLOGY FOR ITANIUM (VT-I)

### Virtualisation enhancements for Itanium

- Adds new processor operating mode for virtualisation
- Guest OS runs in p10, but with restricted permissions
- Privileged and sensitive instructions trap in this mode
- Some address space reserved for VMM
- Certain issues still problematic (physical mode, RSE)

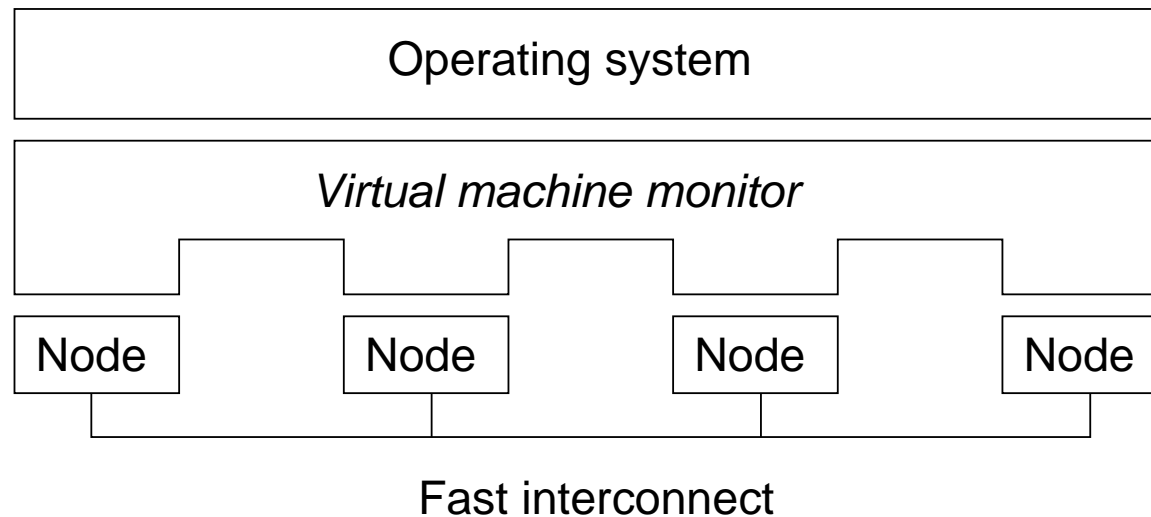
---

## vNUMA

- Itanium virtual machine monitor developed at UNSW
- Native/standalone/"Type I" VMM
- Good performance
- Supports previrtualised guests for even better performance
- Distributed!

---

## vNUMA DISTRIBUTION



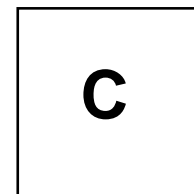
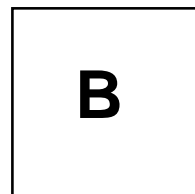
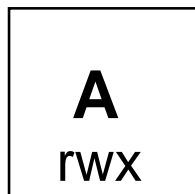
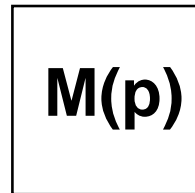
- Run vNUMA on multiple nodes of a cluster
- vNUMA locates and manages CPU/memory/IO resources
- Presents illusion of a single large NUMA machine to guest OS
- Inter-processor communication sent over Ethernet
- "Physical memory" of virtual machine is distributed using *distributed shared memory* techniques

---

## EXAMPLE

M(p): manager for page p

owner=A

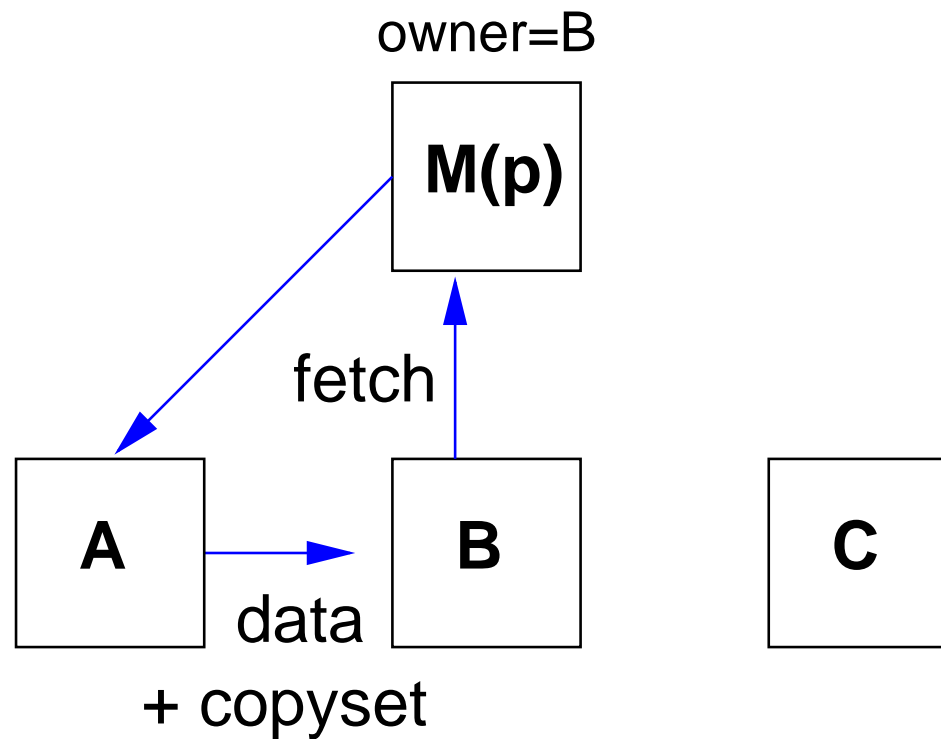


copyset={}

---

## EXAMPLE

$M(p)$ : manager for page  $p$



---

Why?

- Single machine easy to use and administer
- Allows distribution of legacy applications
- Cool application of virtualisation!

---

QUESTIONS?