Virtualisation Case Study: Itanium and vNUMA

Matthew Chapman
University of NSW, National ICT Australia, HP Labs
matthewc@cse.unsw.edu.au

Slide 1

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 µm, 733/800Mhz
2002: Itanium II (McKinley) 0.18 µm, 900Mhz/1Ghz
2003: Itanium II (Madison) 0.13 µm, 1.3-1.6Ghz
2005: Montecito 0.09 µm, 1.8-2Ghz

Slide 2

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

Slide 3

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:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI ld8 r4</td>
<td>r4 = r1</td>
</tr>
<tr>
<td>ld8 r5</td>
<td>r5 = r2</td>
</tr>
<tr>
<td>nop.i 0</td>
<td></td>
</tr>
<tr>
<td>M;MI ld8 r6</td>
<td>r6 = r3</td>
</tr>
<tr>
<td>:: add r7</td>
<td>r7 = r4, r5 // Arithmetic on M too</td>
</tr>
<tr>
<td>:: nop.i 0</td>
<td></td>
</tr>
<tr>
<td>:: add r8</td>
<td>r8 = r6, r7</td>
</tr>
<tr>
<td>::</td>
<td></td>
</tr>
<tr>
<td>:: nop.m 0</td>
<td></td>
</tr>
<tr>
<td>:: nop.i 0</td>
<td></td>
</tr>
</tbody>
</table>

A better way:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMI ld8 r4</td>
<td>r4 = r1</td>
</tr>
<tr>
<td>ld8 r5</td>
<td>r5 = r2</td>
</tr>
<tr>
<td>nop.i 0</td>
<td></td>
</tr>
<tr>
<td>M;MI ld8 r6</td>
<td>r6 = r3</td>
</tr>
<tr>
<td>:: add r7</td>
<td>r7 = r4, r5</td>
</tr>
<tr>
<td>:: add r8</td>
<td>r8 = r6, r7</td>
</tr>
<tr>
<td>:: add r9</td>
<td>r9 = r6, r7</td>
</tr>
</tbody>
</table>

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 4k 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
### Slide 13

<table>
<thead>
<tr>
<th>Memory access</th>
<th>Other privileged</th>
<th>Memory management (privileged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmpxchg tc</td>
<td>mov cr</td>
<td>ltc</td>
</tr>
<tr>
<td>tc</td>
<td>mov abr</td>
<td>mov plk</td>
</tr>
<tr>
<td>fwb</td>
<td>mov ibr</td>
<td>ltr</td>
</tr>
<tr>
<td>ld</td>
<td>mov pmc</td>
<td>mov rr</td>
</tr>
<tr>
<td>ldfp</td>
<td>mov psr</td>
<td>ptc</td>
</tr>
<tr>
<td>probe st</td>
<td>mov rfi</td>
<td>tak</td>
</tr>
<tr>
<td>sync</td>
<td>rsm</td>
<td>tpa</td>
</tr>
<tr>
<td>fetchadd ldf mf sf</td>
<td>ssm</td>
<td>thash a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ttag a</td>
</tr>
</tbody>
</table>

**Branch**

- br
- brl
- brp
- chk
- mov br

**Register Stack Engine**

- alloc
- climb
- cover
- mov ar.bspstore
- invala
- loadrs
- mov liar

**Slide 14**

<table>
<thead>
<tr>
<th>Memory access</th>
<th>Other privileged</th>
</tr>
</thead>
<tbody>
<tr>
<td>break mov ar</td>
<td>mov cr</td>
</tr>
<tr>
<td>epc mov cpuid</td>
<td>mov abr</td>
</tr>
<tr>
<td>hint mov ip</td>
<td>mov ibr</td>
</tr>
<tr>
<td>mov um</td>
<td>mov pmc</td>
</tr>
<tr>
<td>nop rum sum</td>
<td>mov psr</td>
</tr>
<tr>
<td>sr1z</td>
<td>mov rfi</td>
</tr>
</tbody>
</table>

**Slide 15**

- side-effect at privilege level 0 when psr.ic=0
- ar.rsc contains real privilege level

- bypasses protection check at privilege level 0
- returns cpuid of real processor
**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 pl0

---

**VANDERPOOL TECHNOLOGY FOR ITANIUM (VT-I)**

Virtualisation enhancements for Itanium

- Adds new processor operating mode for virtualisation
- Guest OS runs in pl0, 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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>rwx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>copyset={}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Slide 21**

**EXAMPLE**

M(p): manager for page p

owner=B

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fetch</td>
<td></td>
</tr>
<tr>
<td>data</td>
<td></td>
<td>+ copyset</td>
</tr>
</tbody>
</table>

**Slide 22**

Why?

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

**Slide 23**