Linux Scheduling

on

Intel Performance Hybrid Architecture

Len Brown, Ricardo Neri

Intel Open Source Technology Center
Linux tasks on Intel Hybrid "Pcores and Ecores"

Uniform Instruction Set on all CPUs

Linux Plumbers Conference - Dublin, Ireland Sept 2022
Linux v4.9 (before ITMT)

Task Placement:
1. Pcore equal to Ecore
2. Pcore HT sibling

Performance variability due to random placement
Linux v4.10 - v5.15 ITMT

Task Placement:
1. Pcore
2. Pcore HT sibling
3. Ecore

Scheduler erroneously prefers HT sibling over Ecore.

Disable ITMT: # echo 0 > /proc/sys/kernel/sched_itmt_enabled

Linux Plumbers Conference - Dublin, Ireland Sept 2022
Linux v5.16 ITMT

Task Placement:
1. Pcore
2. Ecore
3. Pcore HT sibling

Scheduler correctly spreads to Ecore before HT sibling.
ITMT Scheduler Architecture

**User**
- cgroups
- classes
- affinity
- idle-injection

**Kernel**
- slow wake
- fast wake
- periodic load balance
- idle load balance

**Software**
- ITMT - Intel Turbo Boost Max Tech.
- ASYM_PACKING
- Priority: Pcore, Ecore, HT
- intel_pstate

**Hardware**
- HWP - HardWare P-States
Pcores are faster than Ecores\textsuperscript{1}

At ISO Frequency, for a \textit{nominal instruction mix}

Pcore/Ecore = 1.27

Intel Hardware Feedback Interface (HFI)

Every CPU has Performance and Efficiency "Scores"

<table>
<thead>
<tr>
<th>Index</th>
<th>CPU</th>
<th>Performance</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,1</td>
<td>56</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>2,3</td>
<td>56</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>4-7</td>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>8-11</td>
<td>30</td>
<td>100</td>
</tr>
</tbody>
</table>

56 = 1.27 * 4.4 Ghz
30 = 1.0 * 3.0 Ghz
Pcore Performance advantage depends on instructions

Performance ratio at ISO frequency:

<table>
<thead>
<tr>
<th>Class</th>
<th>ISA Example</th>
<th>Pcore/Ecore Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SSE</td>
<td>1.27</td>
</tr>
<tr>
<td>1</td>
<td>AVX2</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>VNNI</td>
<td>2.0</td>
</tr>
<tr>
<td>3</td>
<td>PAUSE</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Pcore/Ecore performance depends on Instruction mix (ISA class)
Intel Thread Director (ITD)

ITD adds (4) ISA classes to HFI

<table>
<thead>
<tr>
<th>Index</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,1</td>
</tr>
<tr>
<td>1</td>
<td>2,3</td>
</tr>
<tr>
<td>2</td>
<td>4-7</td>
</tr>
<tr>
<td>3</td>
<td>8-11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>P0</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>56</td>
<td>66</td>
<td>88</td>
<td>44</td>
</tr>
<tr>
<td>1</td>
<td>56</td>
<td>66</td>
<td>88</td>
<td>44</td>
</tr>
<tr>
<td>2</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>1</td>
<td>92</td>
<td>92</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Benefit of running on Pcore depends on type of instructions (ISA class)
ITD ISA Classification

Linux user clock tick:

\[
\text{curr->class} = \text{MSR\_IA32\_HW\_FEEDBACK\_CHAR}
\]

Linux context switch:

\[
\text{HRESET}
\]
ITMT+ITD Scheduler Architecture

User
- cgroups
- classes
- affinity
- idle-injection

Kernel
- slow wake
- fast wake
- periodic load balance
- idle load balance

ITMT - Intel Turbo Boost Max Tech.
ASYM_PACKING
Priority: Pcore, Ecore, HT

Software
- intel_pstate
- task.class

Hardware
- HWP - HardWare P-States
- intel_hfi
- ITD - Intel Thread Director

* Other names and brands may be claimed as property of others
Idle Load Balance (Ecore -> Pcore)

When: partially idle: #PCORES <= #task <= #CPU

What: CPU "dst" enters idle, searches for "busy" to offload

ITMT: Pcore pull from Ecore, Ecore pull from HT, always
ITD: Opportunity to break tie on which busy is "busiest"
Idle Load Balance (HT -> Ecore)

When: partially idle: \#PCORES < \#task <= \#CPU

What: CPU "dst" enters idle, searches for busiest to offload

ITMT: Pcore pull from Ecore, Ecore pull from HT, always
ITD: *Opportunity* to break tie on which busy is "busiest"
Periodic Load Balance - Live Exchange

When: partially idle or fully utilized or overutilized

What: CPU "dst1" enters periodic load balancer, searches dst2

Today: NUMA load balance live exchange (on page fault)
Opportunity: ITD live exchange can increase throughput
Periodic Load Balance - Pull from Queue

When: overloaded/unbalanced: \#CPU < \#task

What: CPU "dst" enters periodic load balancer, searches busiest to help

Today: dst selects busy, and pulls until balance reached

Opportunity: ITD can break tie on which sg is busiest
Periodic Load Balance - Queue Exchange

When: overloaded/balanced: \#CPU < \#task

What: CPU "dst1" enters periodic load balancer, searches dst2

Today: "balanced" queues are untouched

Opportunity: ITD can increase throughput
Linux EAS

Task Placement:

- Capacity Model
- Energy Model

Energy Model used as tie breaker when multiple possible Capacity Scenarios
Why EAS is not a good match for ADL

Capacity Model Challenges

1. SMT
2. Turbo
3. HWP

Energy Model Challenges

1. SMT
2. Static ITD Energy table
3. Pcore often more efficient than Ecore
Discussion
ITMT SMT migration Improvement (post Linux 6.0)

1. Remove superfluous Jr-sibling -> Sr-sibling migrations
2. Remove restriction on Ecore pulling from only Jr-sibling
3. Remove artificial software assignment of different priority for Jr and Sr

**Sr.** | **Jr.**
---|---

*SMT siblings are independent CPUs with Jr/Sr priority*

*SMT Siblings are equal parts of one shared Core.*