Agenda

- The problem
- The enhancements
- Some figures
- What’s next?
The problem

- Can’t get best performance when using PELT
  - Compared to WALT as an example
- There are/were several issues with PELT
  - Not enough responsive
  - Task migrations in groups not reflected
  - Load/utilization metrics not always stable
  - ...
The Problem

- **TaskA**
- **TaskB**

- **stalled load & utilization of idle CPUs**
- **load sharing inside group not updated correctly**
- **no load/utilization propagation with migration**
- **freq decreases when task migrates**
- **unstable load for small task**
The enhancements

● Almost 20 patches that impacts PELT merged since v4.7

● Some noticeable changes
  ○ Propagate utilization/load in cgroup (merged)
  ○ Fix small task tracking (merged)

● More to come
  ○ Update blocked load
  ○ New invariance scheme
The target

Load & utilization of idle CPUs updated

Load sharing inside group updated

TaskA

TaskB

load/utilization propagate with migration

No freq changes when task migrates

stable load for small task
New invariance scheme

- Utilization @ max freq
- Utilization and freq scaling w/ new invariance
- Utilization and freq scaling w/ legacy invariance

120ms
More realistic UC figures

- Previous results and tests were unitary ones
- Move to more realistic bench
  - Android Hikey: Kernel v4.4 (w/ EAS)
  - Vellamo bench
  - UI jank test
Thermal mitigation impact

● Thermal mitigation impacts on results
  ○ No simple rule
  ○ Done tests with external active cooling
    ■ Prevent SoC temp to go above 65°C and trig thermal mitigation
  ○ Performance impact of thermal mitigation is not obvious
    ■ Sometime results at the opposite.
    ■ Vellamo Threadbench seems to take advantage of thermal mitigation ...
Vellamo Multicore bench

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Android kernel v4.4</th>
<th></th>
<th>Android kernel v4.4</th>
<th></th>
<th>Android kernel v4.4</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>score</td>
<td>stdev</td>
<td>score</td>
<td>stdev</td>
<td>score</td>
<td>stdev</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>Multicore</td>
<td>1,407</td>
<td>4%</td>
<td>1,295</td>
<td>13%</td>
<td>1,548</td>
<td>3%</td>
</tr>
<tr>
<td>multi.Linpack</td>
<td>130</td>
<td>2%</td>
<td>125</td>
<td>2%</td>
<td>132</td>
<td>0%</td>
</tr>
<tr>
<td>multi.LinpackJava</td>
<td>176</td>
<td>2%</td>
<td>166</td>
<td>3%</td>
<td>176</td>
<td>2%</td>
</tr>
<tr>
<td>multi.Stream</td>
<td>165</td>
<td>3%</td>
<td>155</td>
<td>0%</td>
<td>165</td>
<td>4%</td>
</tr>
<tr>
<td>multi.Membench</td>
<td>208</td>
<td>7%</td>
<td>266</td>
<td>62%</td>
<td>206</td>
<td>7%</td>
</tr>
<tr>
<td>multi.Sysbench</td>
<td>182</td>
<td>6%</td>
<td>166</td>
<td>6%</td>
<td>232</td>
<td>12%</td>
</tr>
<tr>
<td>multi.Threadbench</td>
<td>201</td>
<td>31%</td>
<td>105</td>
<td>19%</td>
<td>279</td>
<td>8%</td>
</tr>
<tr>
<td>multi.Parsec</td>
<td>219</td>
<td>3%</td>
<td>176</td>
<td>13%</td>
<td>223</td>
<td>9%</td>
</tr>
<tr>
<td>multi.ProcessCommunication</td>
<td>125</td>
<td>1%</td>
<td>136</td>
<td>1%</td>
<td>134</td>
<td>1%</td>
</tr>
</tbody>
</table>
## UI interaction bench

<table>
<thead>
<tr>
<th></th>
<th>Android kernel v4.4 WALT</th>
<th>Android kernel v4.4 legacy PELT</th>
<th>Android kernel v4.4 new PELT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ms</td>
<td>Stdev (ms)</td>
<td>ms</td>
</tr>
<tr>
<td>hwuitest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50th percentile</td>
<td>22.00</td>
<td>0.00</td>
<td>22.21</td>
</tr>
<tr>
<td>90th percentile</td>
<td>24.00</td>
<td>1.32</td>
<td>25.04</td>
</tr>
<tr>
<td>95th percentile</td>
<td>24.79</td>
<td>1.50</td>
<td>27.29</td>
</tr>
<tr>
<td>99th percentile</td>
<td>29.75</td>
<td>4.06</td>
<td>34.63</td>
</tr>
<tr>
<td>recentfling</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90th percentile</td>
<td>33.50</td>
<td>0.88</td>
<td>33.75</td>
</tr>
<tr>
<td>95th percentile</td>
<td>36.17</td>
<td>0.82</td>
<td>36.67</td>
</tr>
<tr>
<td>99th percentile</td>
<td>39.58</td>
<td>1.44</td>
<td>40.75</td>
</tr>
</tbody>
</table>
What next?

- RT task utilization tracking
- Improve responsiveness furthermore
RT task utilization tracking

- RT tasks run at max freq
  - At least, it tries
  - Frequency change on hikey needs around 1.5ms
  - Slow path of schedutil uses RT thread

- Current load tracking: rt_avg
  - Fine for slow variation around hundreds of ms
  - Fine when averaging period is larger than updating period

- Use a similar load tracking than CFS
  - Only at rq level
  - Same responsiveness than CFS
Improve PELT responsiveness

- Decay windowing
- Change half decay period
  - Current period is 32ms
- Use other scheduling class
Next

- Continue the discussion during hacking session
Thank You

#BUD17

For further information: [www.linaro.org](http://www.linaro.org)

BUD17 keynotes and videos on: [connect.linaro.org](http://connect.linaro.org)