EAS Update

September 2015

Amit Kucheria – Linaro Technical lead – Power Management
Robin Randhawa – ARM Powersoft Tech lead
Ian Rickards – ARM Powersoft Product Manager
Motivation

- Hardware topologies are becoming more varied, accommodating different power/performance budgets:
  - SMP, multi-cluster SMP, ARM big.LITTLE technology.
  - Per core/per cluster DVFS (Dynamic Voltage & Frequency Scaling)

- Linux power management frameworks are uncoordinated and hard to tune for different topologies.

- We need a common upstream solution to minimize software costs.

_All_ policy, all metrics, all averaging should happen at the scheduler power saving level, in a single place, and then the scheduler should directly drive the new low level idle state driver mechanism.
Goals

Introduce generic **energy-awareness** in upstream Linux:
1. Integrate **Idle, DVFS, scheduler big.LITTLE** support
2. Clean design rather than short-cuts.
3. Based on measureable **energy model** data rather than magic tunables.
4. Support future CPU topologies
5. Maintained in upstream Linux, reduced software maintenance costs.
Power Fundamentals

Static Power
- Area of silicon (mm²)
- Threshold voltage (Vt)
  - “Low Vt” implementation faster but more leaky
  - “High Vt” implementation slower
- Temperature

Dynamic Power
- Toggling nodes x capacitance x voltage
OS task scheduling – throughput policy

Scheduling policy decides task placement
- Affect performance and energy consumption.

Mainline Linux policy is ‘work preserving’
- Considers only maximizing throughput.
- DVFS and idle-states controlled by independent policy governors.

Designed for SMP, not energy-aware
OS task scheduling – energy-aware policy

Energy-Aware Scheduling (EAS) policy:
- Pick CPU with sufficient spare capacity and smallest energy impact.

Requirements:
1. Tracking of task utilization.
2. Platform energy model.

Supports all topologies, including SMP and big.LITTLE.
What is EAS – the energy model

- Waking task
- Capacity
- Utilization

Max capacity
- big
Max capacity
- little
Current capacity

Placing task on cpu3:
No P-state changes.

Placing task on cpu1:
P-state change for cpu0 and cpu1.

CPU cluster
- 0: little
- 1: big

Compute capacity (Performance)
EAS
New Energy Aware Scheduling

- Generic energy model based approach fits all platforms and topologies.
- Foundation for further enhancements.

vs
big.LITTLE HMP
Existing Heterogeneous MP patchset

- big.LITTLE topology only.
- Hard coded behaviors.
- In Linaro LSK kernels (not mainline).

Source patches

Generic EAS Foundation +1
Additional tuning patches +2
Upstreaming EAS +3

Source patches

b.L HMP +1
(big.LITTLE specific)

Designed-for- Upstream
Additional performance patches
Partner patches
DVFS in Linux (cpufreq)

- Sampling based governors are slow to respond and hard to tune.
  
  - **Sampling too fast:** OPP changes for small utilization spikes.
  
  - **Sampling too slow:** Sudden burst of utilization might not get the necessary OPP change in time.
Scheduler-driven DVFS

- With scheduler task utilization tracking DVFS can be notified immediately when CPU utilization changes = improved responsiveness.
SchedTune

Current:
- A set of governor-specific tunables.

Goal:
- Single tunable to bias the energy/performance trade-off.

Prototypes:
- Global boost tunable:
  /proc/sys/kernel/sched_cfs_boost
- Task group (cgroup) based tuning:
  /sys/fs/cgroup/stune/<group>/schedtune.boost
Tunability improvements

### Existing CFS with HMP

<table>
<thead>
<tr>
<th>HMP tunables</th>
<th>hmp_domains, up_threshold, down_threshold, packing_enable, packing_limit, frequency_invariant_load_scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interactive governor</td>
<td>min_sample_time, hispeed_freq, go_hispeed_load, above_hispeed_delay, timer_rate, input_boost, boost, boostpulse</td>
</tr>
</tbody>
</table>

### EAS

<table>
<thead>
<tr>
<th>EAS tunables</th>
<th>NONE - energy model only</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchedTune</td>
<td>‘boost’ margin { boost, boostpulse }</td>
</tr>
</tbody>
</table>
# Analysis tools

<table>
<thead>
<tr>
<th>Tool name / function</th>
<th>Location</th>
</tr>
</thead>
</table>
| **rt-app**/ WorkloadGen (Linaro)  
| **workload-automation** (ARM)  
Automating benchmark runs and ftrace log capture  
(Linux, Android, ChromeOS) | https://github.com/ARM-software/workload-automation |
| **TRAPpy** (ARM)  
Python-based visualization tool to help analyze ftrace data generated on a device. Uses ipython & javascript | https://github.com/ARM-software/trappy |
| **BART** (ARM)  
Behavior Analysis Regression Testing  
Thread residency checker, used as the framework for regression testing for EAS. | https://github.com/ARM-software/bart |
| **Idlestat** (Linaro)  
Idlestat uses kernel ftrace to monitor and capture C-state and P-state transitions of CPUs over a time interval. | https://wiki.linaro.org/WorkingGroups/PowerManagement/Resources/Tools/Idlestat |
| **kernelshark**  
X11/GTK tool for analysis of ftrace data, useful for detailed scheduler analysis but does not offer the API capability of ‘trappy’ above. | http://people.redhat.com/srostedt/kernelshark/HTML/ |
All policy, all metrics, all averaging should happen at the scheduler power saving level, in a single place, and then the scheduler should directly drive the new low level idle state driver mechanism.

--- Ingo Molnar (31 Mar 2013)
## Current patchsets for review/testing

<table>
<thead>
<tr>
<th>Patchset</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduler driven DVFS PATCH v3</td>
<td><a href="https://lkml.org/lkml/2015/6/26/620">https://lkml.org/lkml/2015/6/26/620</a></td>
</tr>
<tr>
<td>EAS RFCv5</td>
<td><a href="https://lkml.org/lkml/2015/7/7/754">https://lkml.org/lkml/2015/7/7/754</a></td>
</tr>
<tr>
<td>SchedTune proposal</td>
<td><a href="https://lkml.org/lkml/2015/8/19/419">https://lkml.org/lkml/2015/8/19/419</a></td>
</tr>
<tr>
<td>Foundational Patches (frequency and microarchitecture contribution to capacity/utilization, split out from RFCv5) <em>(already queued for merging!)</em></td>
<td><a href="https://lkml.org/lkml/2015/8/14/296">https://lkml.org/lkml/2015/8/14/296</a></td>
</tr>
<tr>
<td>Yuyang Du PELT rewrite v10 containing ARM enhancements to utilization calculation <em>(already queued for merging!)</em></td>
<td><a href="https://lkml.org/lkml/2015/7/15/159">https://lkml.org/lkml/2015/7/15/159</a></td>
</tr>
</tbody>
</table>

- Request reviewers to send ‘tested-by’ or ‘acked-by’
EAS RFCv5 update – posted 7-Jul-2015

- Linaro sched-DVFS integration + ARM improvements
- Maps all 6 HMP behaviors
- Landed on ChromeOS
  - (Linux 3.18 kernel)
- SchedTune equivalent to ‘interactive’

### HMP behaviors vs. Equivalent in EAS?

<table>
<thead>
<tr>
<th>HMP behaviors</th>
<th>Equivalent in EAS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake migration</td>
<td>Yes – from wakeup pathways</td>
</tr>
<tr>
<td>Fork migration</td>
<td>Yes – new task initialized to max load</td>
</tr>
<tr>
<td>Forced migration</td>
<td>Yes – from periodic load balance</td>
</tr>
<tr>
<td>Idle-pull migration</td>
<td>Yes – from idle load balance</td>
</tr>
<tr>
<td>Offload migration</td>
<td>Yes – one task per CPU</td>
</tr>
<tr>
<td>Small task packing</td>
<td>Yes – built into design. From energy model.</td>
</tr>
</tbody>
</table>
Results – RFCv5 @ ARM TC2

Energy

Lower is better.

Performance

Higher is better.
EAS Near-term Plans

**LSK 3.18**
- (Linaro – targeting 15.10) allowing direct HMP vs. EAS comparisons

**Testing**
- Use-cases (ChromeOS then Android)
- More platforms – can you help test?

**Android testing & tuning** targeting December 2015
- Starting with HMP sched_tests, migrating to ‘bart’ tests

**Productization**
- Analysis tools / test suites / tuning flow & documentation
- Energy model flow (based on power/perf measurements of dhrystone or sysbench)
EAS Needs YOU!

Now get on the list and ACK some patches!