EAS in Android Common Kernel

AOSP Common Kernel Update

EAS Mainline Strategy

EAS Upstreaming
AOSP Common Kernel Update
AOSP Common Kernel Update

- EAS r1.3, July 2017
  - android-4.4, android-4.9
  - Default cpufreq governor switched to schedutil, sched-freq removed
  - Backports of upstream schedutil changes
  - Upstream backports of relevant scheduler features
AOSP Common Kernel Update

- EAS r1.4, November 2017
  - android-4.4, android-4.9
  - Upstream backports of more scheduler and schedutil patches
  - Energy diff improvements & fixes
  - android-4.14 EAS released including 1.4 & most 1.5 functionality

© 2017 Arm Limited
EAS in android-4.14

A new set of patches implementing EAS rather than forward-porting

• Based upon our latest mainline-focussed integration branch
• Refactored latest android-eas on top to build clean set of patches
• More Experimental features placed behind sched_features
  • Feature configuration matches android-4.9
• Produced during linux-4.14 rc phase, ready 2-weeks after linux-4.14
EAS in android-4.14

android-specific
- Use of idle states
- schedtune

WALT
- Sync Wakeups
- Trace & Debug
- Schedutil changes
- find_best_target
- Load balance tweaks

Upstream-targeted
- Topology Detection
- Invariance Support
- NOHZ Signal Updates
- Energy Diff Calculation
- Misfit Tasks & Overutilized Flags
- Load balance tweaks
AOSP Common Kernel Update

- EAS r1.5, Feb 2018 (eas-dev), merging to android-4.9 soon
  - android-4.9 only, most changes already in android-4.14
  - Refactored energy diff to make calculation more efficient
  - Further refinement of EAS CPU pre-selection (find_best_target)
    - Thanks for excellent contributions from Qualcomm, Spreadtrum, Mstar, Linaro
  - Aggressive up-migrate of Misfit tasks & WALT updates from CodeAurora
AOSP Common Kernel Update

- EAS r1.6, eas-dev starting April 2018
  - Moving to android-4.14
  - Adding back Schedtune PE space filtering
  - Util_est backport, with PELT decay rate changes
  - Use mainline wakeup code for prefer_idle tasks
  - Remove ordering dependency in find_best_target
    - (better tri-gear support when using find_best_target)
AOSP Common Kernel Update

Branches:

• android-4.4, android-4.9 & android-4.14
  • Common kernel upstream for device kernels
  • Only post against this for bugfixes
  • People merge these into device kernels, so need to be selective about changes
AOSP Common Kernel Update

More branches:

• android-4.9-eas-dev (soon android-4.14-eas-dev)
  • This is where in-development patches should be posted
  • Arm power team usually post patches at RFC stage to stimulate discussion
  • Changes picked or merged back to common

• android-4.4-eas-test (android-4.9-eas-test later)
  • Test branch is against android common for the latest well-supported public device
  • Intended to hold backports of EAS patches which merged into the active common branch, but did not get back to the branch we test with
AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:
AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability
AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability
- New features where necessary
AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability
- New features where necessary
- Open, collaborative development
AOSP Common Kernel Update

Open Development

- Patches for AOSP are reviewed on AOSP Gerrit
  - https://android-review.googlesource.com
  - We always try to justify patches with performance & energy numbers – use wltests for this
  - Wltests is part of LISA https://www.github.com/arm-software/lisa

- Discussion of other topics and announcements are on Linaro’s eas-dev list
  - https://lists.linaro.org/mailman/listinfo/eas-dev
EAS Mainline Strategy
EAS Mainline Strategy

• EAS is a large, complex piece of functionality

• EAS being in AOSP helps a lot of users but not all

• Upstream development results in better code
EAS Mainline Strategy

• We make regular bi-weekly integrations of all our upstream-focussed code
  • Available on linux-arm.org & announced on eas-dev
  • Allows us to more easily see when changes impact us and work to resolve as soon as possible

• Have been identifying suitable code we already have
  • Working on getting them into acceptable shape
  • Pushing when we think they are good enough for a review
  • Hoping to upstream quite a lot of EAS this year
EAS Mainline Strategy

- Also working upstream where we can and backporting to Android
  - schedutil fixes
  - cpu signal updates
  - any fix/change applicable and potentially useful elsewhere
  - participating in reviews and testing
## EAS in AOSP

### EAS Code Size by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android-specific</td>
<td>3797</td>
</tr>
<tr>
<td>WALT</td>
<td>1470</td>
</tr>
<tr>
<td>Upstreamable Features</td>
<td>2785</td>
</tr>
<tr>
<td>Documentation</td>
<td>1153</td>
</tr>
</tbody>
</table>

### Target EAS Code Size by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android-specific</td>
<td>824</td>
</tr>
<tr>
<td>WALT</td>
<td>0</td>
</tr>
<tr>
<td>Upstreamable Features</td>
<td>2785</td>
</tr>
<tr>
<td>Documentation</td>
<td>1153</td>
</tr>
</tbody>
</table>
EAS Size

ANDROID-SPECIFIC EAS CODE SIZE

- EAS Code size in android-4.14: 100.00%
- Android Specific Code Target (rough estimate if everything goes to plan, mid-2019): 14.76%
Bringing EAS in AOSP Closer to Mainline

1. Reach performance/energy parity with WALT
   - WALT is great for mobile but not popular upstream
   - It’s also 1.5k LoC
   - Touches many parts of the scheduler we want to change upstream, which makes backporting harder
Bringing EAS in AOSP Closer to Mainline

1. Reach performance/energy parity with WALT
   - Disable WALT by default in android-common when ready

2. Push better support for big.LITTLE into mainline scheduler
   - Push out-of-tree wakeup and periodic balance changes upstream
   - Push energy diff calculations upstream
Bringing EAS in AOSP Closer to Mainline

1. **Reach performance/energy parity with WALT**
   - Disable WALT by default in android-common when ready

2. **Push better support for big.LITTLE into mainline scheduler**
   - Push out-of-tree wakeup and periodic balance changes upstream
   - Push energy diff calculations upstream

3. **Expect to continue to carry mobile-specific changes in AOSP**
   - Schedutil up/down throttle split
   - Rt-rq signals
   - Performance/Energy task classification
EAS Upstreaming
## EAS Upstreaming

7 areas identified for upstreaming.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Model</td>
<td>On LKML (v1 March 2018, during Connect!)</td>
</tr>
<tr>
<td>Frequency and Cpu Invariant Engines (FIE/CIE)</td>
<td>Merged in v4.15</td>
</tr>
<tr>
<td>Idle Cpu PELT update (Remote status update)</td>
<td>Merged in tip/sched/core</td>
</tr>
<tr>
<td>Util Est</td>
<td>Merged in tip/sched/core (during Connect!)</td>
</tr>
<tr>
<td>Util Clamp</td>
<td>Almost ready (v1 on LKML April 2018)</td>
</tr>
<tr>
<td>Misfit Task</td>
<td>On LKML (v2 March 2018)</td>
</tr>
<tr>
<td>Dynamic Topology Flag Detection</td>
<td>In development, many scenarios to cover</td>
</tr>
</tbody>
</table>
EAS Upstreaming

• Util-Est
  • Add an **aggregator** on top of the PELT estimator
    - keep track of what “we learned” about task’s previous activations
    - generate a “new” signal on top of PELT
  • Build a **low-overhead statistic** for SEs and CPUs
    - Tasks at dequeue time
    - Root RQs at task enqueue/dequeue
• Lots of detail at last year’s OSPM Summit and lkml
  • Patches merged into upstream tip/sched/core branch
EAS Upstreaming

• **Misfit Tasks**
  
  • Promote long-running tasks to most capable CPUs
  
  • Key to achieving consistent performance in heterogenous systems
  
  • Tasks which don’t sleep need active migration
A Simplified Energy Model for EAS
An Energy Model: why?

- Power/perf. characteristics differ between different SoCs
- Heuristics don’t perform well on many platforms
- The Energy Model enables the design of a platform-agnostic algorithm in the scheduler
- Designed for mainline
Summary

1. Today’s Energy Model
2. Which simplified Energy Model?
3. Mainline implementation
4. Conclusion
Summary

1. Today’s Energy Model
2. Which simplified Energy Model?
3. Mainline implementation
4. Conclusion
### The Energy Model in Android / Hikey960

#### CPU LEVEL

<table>
<thead>
<tr>
<th>MHz</th>
<th>Cap.</th>
<th>Cost</th>
<th>MHz</th>
<th>Cap.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>533</td>
<td>133</td>
<td>87</td>
<td>903</td>
<td>390</td>
<td>404</td>
</tr>
<tr>
<td>999</td>
<td>250</td>
<td>167</td>
<td>1421</td>
<td>615</td>
<td>861</td>
</tr>
<tr>
<td>1402</td>
<td>351</td>
<td>265</td>
<td>1805</td>
<td>782</td>
<td>1398</td>
</tr>
<tr>
<td>1709</td>
<td>429</td>
<td>388</td>
<td>2112</td>
<td>915</td>
<td>2200</td>
</tr>
<tr>
<td>1844</td>
<td>462</td>
<td>502</td>
<td>2362</td>
<td>1024</td>
<td>2848</td>
</tr>
</tbody>
</table>

#### Diagram

![Graph showing Power vs. Capacity for different CPU levels](image-url)
# The Energy Model in Android / Hikey960

## CPU LEVEL

<table>
<thead>
<tr>
<th>MHz</th>
<th>Cap.</th>
<th>Cost</th>
<th>MHz</th>
<th>Cap.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>533</td>
<td>133</td>
<td>87</td>
<td>903</td>
<td>390</td>
<td>404</td>
</tr>
<tr>
<td>999</td>
<td>250</td>
<td>167</td>
<td>1421</td>
<td>615</td>
<td>861</td>
</tr>
<tr>
<td>1402</td>
<td>351</td>
<td>265</td>
<td>1805</td>
<td>782</td>
<td>1398</td>
</tr>
<tr>
<td>1709</td>
<td>429</td>
<td>388</td>
<td>2112</td>
<td>915</td>
<td>2200</td>
</tr>
<tr>
<td>1844</td>
<td>462</td>
<td>502</td>
<td>2362</td>
<td>1024</td>
<td>2848</td>
</tr>
</tbody>
</table>

![Power vs Capacity Graph](#)

<table>
<thead>
<tr>
<th>LITTLE</th>
<th>big</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## The Energy Model in Android / Hikey960

### CPU Level

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHz</td>
<td>MHz</td>
</tr>
<tr>
<td>533</td>
<td>903</td>
</tr>
<tr>
<td>999</td>
<td>1421</td>
</tr>
<tr>
<td>1402</td>
<td>1805</td>
</tr>
<tr>
<td>1709</td>
<td>2112</td>
</tr>
<tr>
<td>1844</td>
<td>2362</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cap.</td>
<td>Cap.</td>
</tr>
<tr>
<td>133</td>
<td>390</td>
</tr>
<tr>
<td>250</td>
<td>615</td>
</tr>
<tr>
<td>351</td>
<td>782</td>
</tr>
<tr>
<td>429</td>
<td>915</td>
</tr>
<tr>
<td>462</td>
<td>1024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Cost</td>
</tr>
<tr>
<td>87</td>
<td>404</td>
</tr>
<tr>
<td>167</td>
<td>861</td>
</tr>
<tr>
<td>265</td>
<td>1398</td>
</tr>
<tr>
<td>388</td>
<td>2200</td>
</tr>
<tr>
<td>502</td>
<td>2848</td>
</tr>
</tbody>
</table>

### Cluster Level

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHz</td>
<td>MHz</td>
</tr>
<tr>
<td>533</td>
<td>903</td>
</tr>
<tr>
<td>999</td>
<td>1421</td>
</tr>
<tr>
<td>1402</td>
<td>1805</td>
</tr>
<tr>
<td>1709</td>
<td>2112</td>
</tr>
<tr>
<td>1844</td>
<td>2362</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cap.</td>
<td>Cap.</td>
</tr>
<tr>
<td>133</td>
<td>390</td>
</tr>
<tr>
<td>250</td>
<td>615</td>
</tr>
<tr>
<td>351</td>
<td>782</td>
</tr>
<tr>
<td>429</td>
<td>915</td>
</tr>
<tr>
<td>462</td>
<td>1024</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Little</th>
<th>Big</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>Cost</td>
</tr>
<tr>
<td>12</td>
<td>102</td>
</tr>
<tr>
<td>22</td>
<td>124</td>
</tr>
<tr>
<td>36</td>
<td>221</td>
</tr>
<tr>
<td>67</td>
<td>330</td>
</tr>
<tr>
<td>144</td>
<td>433</td>
</tr>
</tbody>
</table>

### Diagram

- [Graph showing power consumption vs. capacity for CPU and cluster levels.]

### Table

- [Table with MHz, Cap., and Cost values for CPU and cluster levels.]

© 2017 Arm Limited
Device-tree bindings
Device-tree bindings

 [...]
cpu0: cpu@0 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
    [...]
}

 [...]
cpu1: cpu@1 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
    [...]
}

 [...]
cpu4: cpu@100 {
    [...]
    sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;
    [...]
}

arch/arm64/boot/dts/hisilicon/hi3660.dtsi
Device-tree bindings

```c
[...]
cpu0: cpu@0 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
    [...]
}
[...]

cpu1: cpu@1 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
    [...]
}
[...]

cpu4: cpu@100 {
    [...]
    sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;
    [...]
}
```
Device-tree bindings

```
[...] cpu0: cpu@0 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;  
    [...]
}
[...]

cpu1: cpu@1 {
    [...]
    sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;  
    [...]
}
[...]

cpu4: cpu@100 {
    [...]
    sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;  
    [...]
}

CPU_COST_A72: core-cost0 {
    busy-cost-data = <
        390 404
        615 861
        782 1398
        915 2200
        1024 2848 >;
    idle-cost-data = < 18 18 0 0 >;
}

CPU_COST_A53: core-cost1 {
    busy-cost-data = <
        133 87
        250 164
        351 265
        429 388
        462 502 >;
    idle-cost-data = < 5 5 0 0 >;
}

CLUSTER_COST_A72: cluster-cost0 {
    busy-cost-data = <
        [...]
}
```

```
arch/arm64/boot/dts/hisilicon/hi3660.dtsi
```

```
arch/arm64/boot/dts/hisilicon/hi3660-sched-energy.dtsi
```
Sched domains
Sched domains
Sched domains

CPU span

0 1 2 3

MC

groups

0 1 2 3

CPU 0

© 2017 Arm Limited
Sched domains

CPU span

0 1 2 3 4 5 6 7

DIE

groups

MC

parent

groups

0 1 2 3

CPU 0

CPU 0

LITTLE

<table>
<thead>
<tr>
<th>Cap.</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>87</td>
</tr>
<tr>
<td>250</td>
<td>167</td>
</tr>
<tr>
<td>351</td>
<td>265</td>
</tr>
<tr>
<td>429</td>
<td>388</td>
</tr>
<tr>
<td>462</td>
<td>502</td>
</tr>
</tbody>
</table>

CPU

LITTLE

<table>
<thead>
<tr>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
Need for simplification

- Comprehensive Energy Model, but ...
Need for simplification

• Comprehensive Energy Model, but ...

• Complex to measure for new platforms
Need for simplification

- Comprehensive Energy Model, but ...
- Complex to measure for new platforms
- Computationally expensive scheduling decisions
Need for simplification

• Comprehensive Energy Model, but ...
• Complex to measure for new platforms
• Computationally expensive scheduling decisions
• Existing code relies only on out-of-tree bindings
Need for simplification

• Comprehensive Energy Model, but ...
• Complex to measure for new platforms
• Computationally expensive scheduling decisions
• Existing code relies only on out-of-tree bindings
• Inaccurate assumptions for future platforms
Summary

1. Today’s Energy Model

2. Which simplified Energy Model?

3. Mainline implementation

4. Conclusion
Which simplified EM?
Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active costs</td>
<td>Idle costs</td>
</tr>
<tr>
<td>FULL</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>NOIDLE</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>NOCLUSTER</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>NOCLUSTER_NOIDLE</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>NO_EAS</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
### Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active costs</td>
<td>Active costs</td>
</tr>
<tr>
<td></td>
<td>Idle costs</td>
<td>Idle costs</td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

© 2017 Arm Limited
## Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active costs</td>
<td>Idle costs</td>
</tr>
<tr>
<td>FULL</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>NOIDLE</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>
## Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Active costs</strong></td>
<td><strong>Idle costs</strong></td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NOCLUSTER</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>
Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th></th>
<th>Cluster Level</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Active costs</td>
<td>Idle costs</td>
<td>Active costs</td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NOCLUSTER</strong></td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NOCLUSTER_NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
## Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active costs</td>
<td>Idle costs</td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NOCLUSTER</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOCLUSTER_NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NO_EAS</strong></td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>
Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Active costs</strong></td>
<td><strong>Idle costs</strong></td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NO_IDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NO_CLUSTER</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NO_CLUSTER_NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NO_EAS</strong></td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

- Tested on Android-4.4: Hikey960, Pixel2, Hikey620
### Which simplified EM?

<table>
<thead>
<tr>
<th>Name</th>
<th>CPU Level</th>
<th>Cluster Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active costs</td>
<td>Idle costs</td>
</tr>
<tr>
<td><strong>FULL</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NOCLUSTER</strong></td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td><strong>NOCLUSTER_NOIDLE</strong></td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td><strong>NO_EAS</strong></td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

- Tested on Android-4.4: Hikey960, Pixel2, Hikey620
- SchedTune disabled, no cpusets
Jankbench / list_view - Hikey960

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

device_total_energy [joules]
Jankbench / list_view - Hikey960

The chart shows the performance of different configurations of the Hikey960 device. The configurations include:

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

The y-axis represents `jankbench:device_total_energy`, and the x-axis represents `device_total_energy [joules]`. The configurations are plotted on the chart, with the mean performance indicated for each.
Jankbench / list_view - Hikey960

The graph shows the impact of different configurations on device total energy consumption. The configurations include:

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

The x-axis represents device_total_energy in joules, ranging from 240 to 340 joules. The y-axis categorizes different configurations.
Jankbench / list_view - Hikey960

The graph shows the device total energy in joules for different configurations:
- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

The graph indicates a 50% reduction in device total energy for the NOCLUSTER_NOIDLE configuration compared to the FULL configuration.
Jankbench / list_view - Hikey960

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

~99%
50%
Jankbench / list_view - Hikey960

The diagram below shows the device_total_energy for different configurations on Hikey960:

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

The x-axis represents device_total_energy in joules, and the y-axis represents different configurations.
Jankbench / image_list_view - Hikey960

The diagram shows a comparison of energy consumption for different configurations:

- FULL
- NOIDLE
- NOCLUSTER_NOIDLE
- NOCLUSTER
- NO_EAS

The x-axis represents device_total_energy in joules, ranging from 240 to 340 joules.

The y-axis represents different configurations, with each configuration having a box plot indicating the distribution of energy consumption.

© 2017 Arm Limited
Jankbench / low_hitrate_text - Hikey960

The diagram shows a box plot for the device total energy (in joules) for different configurations:

- NOIDLE
- NOCLUSTER_NOIDLE
- FULL
- NOCLUSTER
- NO_EAS

The energy values range from approximately 200 to 350 joules.
Homescreen / Hikey960

The diagram illustrates the distribution of device total energy for different configurations:

- **NOCLUSTER**
- **NOCLUSTER_NOIDLE**
- **NO_EAS**
- **NOIDLE**
- **FULL**

The x-axis represents the device total energy in joules, ranging from 290 to 330 joules. The data points are plotted to show the variation and central tendency of the energy consumption under each configuration.
ExoPlayer Video / Hikey960

The diagram shows the comparison of device total energy across different configurations:

- **NOIDLE**
- **NOCLUSTER**
- **NOCLUSTER_NOIDLE**
- **FULL**
- **NO_EAS**

The x-axis represents device_total_energy in joules, ranging from 250 to 360 joules.
ExoPlayer Audio / Hikey960
Results of experiments

- Hikey960: all energy models show comparable energy savings
- Pixel2: same conclusions with smaller savings (up to 13%, screen on)
- Hikey620 (SMP): No significant savings
Results of experiments

• Hikey960: all energy models show comparable energy savings
• Pixel2: same conclusions with smaller savings (up to 13%, screen on)
• Hikey620 (SMP): No significant savings

The simplest EM (*noidle_nocluster*) is a reasonable option for modern platforms
Summary

1. Today’s Energy Model
2. Which simplified Energy Model?
3. Mainline implementation
4. Conclusion
Dynamic power model
Dynamic power model

\[ P = C \times V^2 \times f \]

\[ 2 \times 2 \times f \]
Dynamic power model

\[ P = C \times V^2 \times f \]

Power
Dynamic power model

\[ P = C \times V^2 \times f \]

- Power
- Capacitance
Dynamic power model

\[ P = C \times V^2 \times f \]

- Power
- Capacitance
- Voltage
Dynamic power model

\[ P = C \times V^2 \times f \]

- Power
- Capacitance
- Voltage
- Frequency
Dynamic power model

\[ P = C \times V^2 \times f \]

- Power
- Capacitance
- Voltage
- Frequency

Mainline DT binding:
dynamic-power-coefficient
Dynamic power model

\[ P = C \times V^2 \times f \]

- **Power**
- **Capacitance**
- **Voltage**
- **Frequency**

*Mainline DT binding:*
- dynamic-power-coefficient

*Managed by CPUFreq / OPP*
Energy Model Comparison / Hikey960

Measured

\( C \times V^2 \times f \)
Architecture

Thermal / IPA

\[ P = CV^2 f \]

PM OPP

Device Tree

\[ C \]
Architecture

Thermal / IPA

Device Tree

PM OPP

\( V \)  \( f \)
Architecture

Thermal / IPA

Device Tree

PM OPP

\[ CV^2 f \]

\[ V \]

\[ f \]

\[ P \]
Architecture

Thermal / IPA

Device Tree

Scheduler

Energy Model

PM OPP

$CV^2f$

$V$

$f$

$P$
Architecture

Thermal / IPA

Device Tree

PM OPP

Scheduler

Energy Model

Device Tree

PM OPP

$CV^2f$

$V$

$f$

$P$

SCMI

© 2017 Arm Limited
Architecture

Thermal / IPA

Device Tree

PM OPP

Scheduler

Energy Model

SCMI

Other ?

$V$, $f$, $P$

$CV^2f$
[PATCH v3 0/2] thermal, OPP: move the CPU power estimation to the OPP library
- [PATCH v3 1/2] PM / OPP: introduce an OPP power estimation helper
- [PATCH v3 2/2] thermal: cpu_cooling: use power models from the OPP library
Implementation

• No hierarchical data, no need to use the scheduling domains

• Data structures:
  • Loaded from PM / OPP at boot time, after CPUfreq is setup
  • Energy models are stored in a flat per-cpu array
  • Frequency domains are stored in cpu-masks
Assumptions

• All CPUs in a freq. domain share capacity states
  • All CPUs in a freq. domain have the same micro-architecture
  • Possible to relax this if needed, but higher computational cost
• EAS enabled for asymmetric platforms only (SDASYM_CPUCAPACITY flag set)
  • EAS shines on heterogeneous platforms
  • Avoid “conflicts” for purely perf-oriented platforms (servers, ...)

© 2017 Arm Limited
Tests against mainline

• Test setup:
  • Platform: Hikey960 and Juno r0
  • Debian userspace
  • Base kernel: tip/sched/core – 4.16-rc2

• Test cases:
  • Energy: “X” RTApp tasks, 16ms period, 5% duty cycle, 30 seconds
  • Performance: `perf bench sched messaging -pipe -thread -group X -loop 50000`
Tests against mainline / Energy

Hikey960 (ACME / full SoC + memory)

Juno (HW monitor / b.L CPUs only)
Tests against mainline / Perf.

### Hikey960

- **Time (s)**
  - 40 tasks
  - 80 tasks
  - 160 tasks
  - 320 tasks

#### Chart Details:
- **Y-axis:** Time (s)
- **X-axis:** Number of tasks (40, 80, 160, 320)
- **Graphs:**
  - **tip/sched/core**
  - **EAS**

### Juno

- **Time (s)**
  - 40 tasks
  - 80 tasks
  - 160 tasks
  - 320 tasks

#### Chart Details:
- **Y-axis:** Time (s)
- **X-axis:** Number of tasks (40, 80, 160, 320)
- **Graphs:**
  - **tip/sched/core**
  - **EAS**
Posted to LKML this week

[RFC PATCH 0/6] Energy Aware Scheduling

[RFC PATCH 1/6] sched/fair: Create util_fits_capacity()

[RFC PATCH 2/6] sched: Introduce energy models of CPUs

[RFC PATCH 3/6] sched: Add over-utilization/tipping point indicator

[RFC PATCH 4/6] sched/fair: Introduce an energy estimation helper …

[RFC PATCH 5/6] sched/fair: Select an energy-efficient CPU on task …

Summary

1. Today’s Energy Model
2. Which simplified Energy Model?
3. Mainline implementation
4. Conclusion
Next steps

• Ideal scenario: simplified EM goes in the next LTS (4.19 ?)
• Test & assessment on android-4.14
• In case of gaps with the full EM, they will be filled in product
Thanks.

Any questions?
The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

www.arm.com/company/policies/trademarks
Algorithm complexity

LITTLE CPUs

BIG CPUs

cap=1000
cost=3000
cap=800
cost=1800
cap=600
cost=1000
cap=400
cost=700
Algorithm complexity

T1

CPU0  CPU1  CPU2  CPU3

T2

CPU4  CPU5  CPU6  CPU7

LITTLE CPUs

BIG CPUs

© 2017 Arm Limited
Algorithm complexity

LITTLE CPUs

1. **T1**
   - cap=100
   - cost=100

BIG CPUs

1. **T2**
   - cap=1000
   - cost=3000

2. cap=800
   - cost=1800

3. cap=600
   - cost=1000

4. cap=400
   - cost=700
Algorithm complexity

- **T1**: cap=100, cost=100
- **T2**: cap=350, cost=400
- **CPU0**: cap=500, cost=800
- **CPU1**: cap=350, cost=400
- **CPU2**: cap=100, cost=100
- **CPU3**: cap=500, cost=800
- **CPU4**: cap=1000, cost=3000
- **CPU5**: cap=800, cost=1800
- **CPU6**: cap=600, cost=1000
- **CPU7**: cap=400, cost=700

© 2017 Arm Limited
Algorithm complexity

CPU0  CPU1  CPU2  CPU3

cap=500  cost=800

cap=350  cost=400

cap=100  cost=100

CPU4  CPU5  CPU6  CPU7

cap=1000  cost=3000

cap=800  cost=1800

cap=600  cost=1000

cap=400  cost=700
Algorithm complexity

LITTLE CPUs

BIG CPUs

CPU0 | CPU1 | CPU2 | CPU3
---|---|---|---
cap=500 | cap=350 | cap=100 | cap=1000
cost=800 | cost=400 | cost=100 | cost=3000

cpu=800 | cap=600 | cap=400 | cap=800
cost=1800 | cost=1000 | cost=700 | cost=1000
Algorithm complexity

LITTLE CPUs

- CPU0: cap=500, cost=800
- CPU1: cap=350, cost=400
- CPU2: cap=100, cost=100

BIG CPUs

- CPU4: cap=1000, cost=3000
- CPU5: cap=800, cost=1800
- CPU6: cap=600, cost=1000
- CPU7: cap=400, cost=700
Algorithm complexity

LITTLE CPUs

CPU0  CPU1  CPU2  CPU3

T1
cap=100
cost=100

cap=350
cost=400

cap=500
cost=800

BIG CPUs

CPU4  CPU5  CPU6  CPU7

cap=400
cost=700

cap=600
cost=1000

cap=800
cost=1800

cap=1000
cost=3000

T2

© 2017 Arm Limited
Algorithm complexity

T1

cap=100
cost=100

cap=350
cost=400

cap=500
cost=800

CPU0 CPU1 CPU2 CPU3

T2

cap=500
cost=800

cap=350
cost=400

cap=100
cost=100

LITTLE CPUs

T1

BIG CPUs

cap=1000
cost=3000

cap=800
cost=1800

cap=600
cost=1000

cap=400
cost=700

CPU4 CPU5 CPU6 CPU7

© 2017 Arm Limited
Algorithm complexity

**LITTLE CPUs**
- **T1**: cap=100, cost=100
- **T2**: cap=500, cost=400

**BIG CPUs**
- **CPU0**: cap=1000, cost=3000
- **CPU1**: cap=800, cost=1800
- **CPU2**: cap=600, cost=1000
- **CPU3**: cap=400, cost=700
- **CPU4**: cap=1000, cost=3000
- **CPU5**: cap=800, cost=1800
- **CPU6**: cap=600, cost=1000
- **CPU7**: cap=400, cost=700
Algorithm complexity

CPU0 | CPU1 | CPU2 | CPU3
---|---|---|---
cap=500 | cost=800
cap=350 | cost=400
cap=100 | cost=100

CPU4 | CPU5 | CPU6 | CPU7
---|---|---|---
cap=1000 | cost=3000
cap=800 | cost=1800
cap=600 | cost=1000
cap=400 | cost=700
Algorithm complexity

LITTLE CPUs

CPU0: cap=100, cost=100
CPU1: cap=350, cost=400
CPU2: cap=500, cost=800
CPU3: cap=1000, cost=3000

BIG CPUs

CPU4: cap=1000, cost=3000
CPU5: cap=800, cost=1800
CPU6: cap=600, cost=1000
CPU7: cap=400, cost=700

© 2017 Arm Limited
Algorithm complexity

LITTLE CPUs

<table>
<thead>
<tr>
<th>CPU</th>
<th>cap</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU0</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>CPU1</td>
<td>350</td>
<td>400</td>
</tr>
<tr>
<td>CPU2</td>
<td>500</td>
<td>800</td>
</tr>
</tbody>
</table>

BIG CPUs

<table>
<thead>
<tr>
<th>CPU</th>
<th>cap</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU4</td>
<td>400</td>
<td>700</td>
</tr>
<tr>
<td>CPU5</td>
<td>600</td>
<td>1000</td>
</tr>
<tr>
<td>CPU6</td>
<td>800</td>
<td>1800</td>
</tr>
<tr>
<td>CPU7</td>
<td>1000</td>
<td>3000</td>
</tr>
</tbody>
</table>
Algorithm complexity

- **T1**
  - cap = 100
  - cost = 100

- **T2**
  - cap = 1000
  - cost = 3000

**LITTLE CPUs**

- CPU0, cap = 100, cost = 100
- CPU1, cap = 350, cost = 400
- CPU2, cap = 500, cost = 800
- CPU3, cap = 1000, cost = 1800

**BIG CPUs**

- CPU4, cap = 400, cost = 700
- CPU5, cap = 600, cost = 1000
- CPU6, cap = 800, cost = 1800
- CPU7, cap = 1000, cost = 3000
Algorithm complexity

LITTLE CPUs

BIG CPUs

cap=500, cost=800

cap=350, cost=400

cap=100, cost=100

cap=1000, cost=3000

cap=800, cost=1800

cap=600, cost=1000

cap=400, cost=700
Algorithm complexity

LITTLE CPUs

BIG CPUs

CPU0

CPU1

CPU2

CPU3

CPU4

CPU5

CPU6

CPU7

Algorithm complexity

T2

T2