What is EAS?

- Task scheduling considering energy implications.
  - Scheduling decisions based on system topology, power management features, and workload.

![Diagram showing the relationship between system topology, workload, task scheduling, frequency selection, CPU selection, idle-state selection, and performance energy.](image-url)
Task scheduling

Energy-Aware Scheduling (EAS):
- 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.
Estimating energy

Requirements:

1. Decomposable model of CPU topology in *sufficient* detail.
2. Must be simple for fast energy estimates. To be used in critical paths of the kernel.

\[ E = P_{busy}t_{busy} + P_{idle}t_{idle} \]
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)
Upstream strategy until RFCv5

- Introducing energy-awareness across multiple framework and core kernel components is complex.

- Upstreaming strategy:
  - Prototype the essential parts and create a consolidated story.
    1. Provides a concrete example of how EAS could work.
       • Enables the community to better understand the overall objective. Never meant for merging in one go, hence the RFC tag.
    2. Provides a prototype for validation on real hardware.
       • Prove that EAS works, which means a consolidated patch stack that interested partners can evaluate.
Upstreaming strategy beyond RFCv5 (now)

- The full patch stack has been evaluated on several platforms.
- Upstream strategy:
  - Split out into smaller feature specific patch sets and mature:
    1. Encourage more detailed reviews/discussions.
    2. Ease review burden on maintainers.
    3. Parallelize the upstreaming process.
- Progress:
  - schedfreq: Has been split out developed separately.
  - schedtune: Core bits are being progressed independently.
  - PELT fixes: Posted separately.
- Consolidated branches are maintained in linux-arm.org for evaluation.
EAS components – currently in flight

- Scheduler-driven DVFS control
- Capacity capping support (thermal)
- Task scheduling using energy model
- Energy model data (architecture)
- Energy model infrastructure (scheduler)
- Scheduler capacity awareness
- Frequency invariance support

+ more...
EAS components – Frequency invariance support

- Implements arm/arm64 frequency scaling input to task scheduler.
- Integrates with cpufreq.
- No dependencies.
EAS components – Scheduler capacity awareness

- Basic support for capacity awareness in CFS.
  - Remove assumption that the system is SMP and RT/Deadline tasks have no impact.
- Benefits big.LITTLE systems even without an energy model.
- Depends on architecture support for cpu capacity and accurate PELT utilization signals.
- Currently being revised.

| sched: rt scheduler sets capacity requirement |
| sched: deadline: use deadline bandwidth in scale_rt_capacity |
| sched: remove call of sched_avg_update from sched_rt_avg_update |
| sched/cpufreq_sched: add trace events |
| sched/fair: jump to max OPP when crossing UP threshold |
| sched/fair: cpufreq_sched triggers for load balancing |
| sched/core/fair: trigger OPP change request on fork() |
| sched/fair: add triggers for OPP change requests |
| sched: scheduler-driven cpu frequency selection |
| cpufreq: introduce cpufreq_driver_is_slow |

arm64: Enable max freq invariant scheduler load-tracking and ..
arm: Enable max freq invariant scheduler load-tracking and ..
sched: update max cpu capacity in case of max frequency constraints

cpufreq: Max freq invariant scheduler load-tracking and cpu ..

sched: Disable energy-unfriendly nohz kicks
sched: Consider a not over-utilized energy-aware system as balanced
sched: Energy-aware wake-up task placement
sched: Determine the current sched.group idle-state
sched: cpuidle: Track cpuidle state index in the scheduler
sched: Add over-utilization/tipping point indicator
sched: Estimate energy impact of scheduling decisions
sched: Extend sched.group_energy to test load-balancing decisions
sched: Calculate energy consumption of sched.group
sched: Highest energy aware balancing sched.domain level pointer
sched: Relocated cpu_util() and change return type
sched: Compute cpu capacity available at current frequency

arm64: Frequency invariant scheduler load-tracking and capacity support
arm: Cpu invariant scheduler load-tracking and capacity support
arm64, topology: Define JUNO energy and provide it to the scheduler
arm: topology: Define TC2 energy and provide it to the scheduler

cpufreq: Frequency invariant scheduler load-tracking support

energy_model_rfc_v5.2

- Basic support for capacity awareness in CFS.
  - Remove assumption that the system is SMP and RT/Deadline tasks have no impact.
- Benefits big.LITTLE systems even without an energy model.
- Depends on architecture support for cpu capacity and accurate PELT utilization signals.
- Currently being revised.
EAS components – Energy model infrastructure

- Extending scheduler topology representation to carry energy model data.
- Table data to be provided by architecture.
- No dependencies.
EAS components – Energy model data

- Provide energy model data to scheduler.
- Hardcoded in RFC, but separate thread ongoing for how to obtain data (DT, DT+dynamic profiling, user-space).
  – Discussion today @ 6 pm, Secretariat 2
- Dependencies: Resolve DT representation of data and/or dynamic profiling.
Providing energy model data

arch/arm{,64}/*

kernel/sched/{core,fair}.c

scheduler topology

cluster
cpu

Perf

Measurements

Power

DT bindings

Generate tables

Dynamic performance profiling

User-space
EAS components – Scheduling using energy model

- Core functions to compute energy estimates.
- Use of energy estimates in task placement decisions.
- Dependencies: Energy model, accurate PELT utilization signals (frequency and cpu invariant).

| sched: | rt scheduler sets capacity requirement |
| sched: | deadline: use deadline bandwidth in scale_rt_capacity |
| sched: | remove call of sched_avg_update from sched_rt_avg_update |
| sched/cpufreq_sched: | add trace events |
| sched/fair: | jump to max OPP when crossing UP threshold |
| sched/fair: | cpufreq_sched triggers for load balancing |
| sched/(core,fair): | trigger OPP change request on fork() |
| sched/fair: | add triggers for OPP change requests |
| sched: | scheduler-driven cpu frequency selection |
| cpufreq: | introduce cpufreq_driver_is_slow |

arm64: Enable max freq invariant scheduler load-tracking and ..
arm: Enable max freq invariant scheduler load-tracking and ..
sched: update max cpu capacity in case of max frequency constraints
cpufreq: Max freq invariant scheduler load-tracking and cpu ..

- sched: Disable energy-unfriendly nohz kicks
- sched: Consider a not over-utilized energy-aware system as balanced
- sched: Energy-aware wake-up task placement
- sched: Determine the current sched_group idle-state
- sched, cpuidle: Track cpuidle state index in the scheduler
- sched: Add over-utilization/tipping point indicator
- sched: Estimate energy impact of scheduling decisions
- sched: Extend sched_group_energy to test load-balancing decisions
- sched: Calculate energy consumption of sched_group
- sched: Highest energy aware balancing sched_domain level pointer
- sched: Relocated cpu_util() and change return type
- sched: Compute cpu capacity available at current frequency

arm: Cpu invariant scheduler load-tracking and capacity support
arm64, topology: Define JUNO energy and provide it to the scheduler
arm: topology: Define TCE energy and provide it to the scheduler
sched: Introduce SD_SHARE_CAP_STATES sched_domain flag
sched: Initialize energy data structures
sched: Introduce energy data structures
sched: Make energy awareness a sched feature
sched: Documentation for scheduler energy cost model

- arm: Prevent unnecessary active balance of single task in sched ..
- sched: Enable idle balance to pull single task towards cpu with ..
- sched: Consider spare cpu capacity at task wake-up
- sched: Add cpu capacity awareness to wakeup balancing
- sched: Store system-wide maximum cpu capacity in root domain

arm: Update arch_scale_cpu_capacity() to reflect change to define
arm64: Enable frequency invariant scheduler load-tracking support
arm: Enable frequency invariant scheduler load-tracking support
cpufreq: Frequency invariant scheduler load-tracking support
EAS components – Capacity capping support

- Adjust scheduler cpu compute capacities when capping frequency (thermal or user-space).
- Depends on scheduler capacity awareness to have an effect.
- Session on EAS/IPA integration:
  - Hacking Session today @ 2 pm lead by Punit Agrawal, Secretariat 2.
EAS components – Scheduler-driven DVFS

- Use scheduler PELT utilization and hints to drive frequency selection.
- No dependencies.
- Separate thread ongoing.
- Covered in detail in the Monday sched-freq session.

<table>
<thead>
<tr>
<th>energy_model_rfc_v5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>sched: rt scheduler sets capacity requirement</td>
</tr>
<tr>
<td>sched: deadline: use deadline bandwidth in scale_rt_capacity</td>
</tr>
<tr>
<td>sched: remove call of sched_avg_update from sched_rt_avg_update</td>
</tr>
<tr>
<td>sched/cpufreq_sched: add trace events</td>
</tr>
<tr>
<td>sched/fair: jump to max OPP when crossing UP threshold</td>
</tr>
<tr>
<td>sched/fair: cpufreq_sched triggers for load balancing</td>
</tr>
<tr>
<td>sched/(core, fair): trigger OPP change request on fork()</td>
</tr>
<tr>
<td>sched/fair: add triggers for OPP change requests</td>
</tr>
<tr>
<td>sched: scheduler-driven cpu frequency selection</td>
</tr>
<tr>
<td>cpufreq: introduce cpufreq_driver_is_slow</td>
</tr>
<tr>
<td>arm64: Enable max freq invariant scheduler load-tracking and ...</td>
</tr>
<tr>
<td>arm: Enable max freq invariant scheduler load-tracking and ...</td>
</tr>
<tr>
<td>sched: update max cpu capacity in case of max frequency constraints</td>
</tr>
<tr>
<td>cpufreq: Max freq invariant scheduler load-tracking and cpu ...</td>
</tr>
<tr>
<td>sched: Consider a not over-utilized energy-aware system as balanced</td>
</tr>
<tr>
<td>sched: Energy-aware wake-up task placement</td>
</tr>
<tr>
<td>sched: Determine the current sched_group idle-state</td>
</tr>
<tr>
<td>sched: Add over-utilization/tipping point indicator</td>
</tr>
<tr>
<td>sched: Estimate energy impact of scheduling decisions</td>
</tr>
<tr>
<td>sched: Extend sched_group-energy to test load-balancing decisions</td>
</tr>
<tr>
<td>sched: Calculate energy consumption of sched_group</td>
</tr>
<tr>
<td>sched: Highest energy aware balancing sched_domain level pointer</td>
</tr>
<tr>
<td>sched: Relocated cpu_util() and change return type</td>
</tr>
<tr>
<td>sched: Compute cpu capacity available at current frequency</td>
</tr>
<tr>
<td>arm64, topology: Define JUNO energy and provide it to the scheduler</td>
</tr>
<tr>
<td>arm: topology: Define TC2 energy and provide it to the scheduler</td>
</tr>
<tr>
<td>sched: Introduce S0_SHARE_CAP_STATES sched_domain flag</td>
</tr>
<tr>
<td>sched: Initialize energy data structures</td>
</tr>
<tr>
<td>sched: Introduce energy data structures</td>
</tr>
<tr>
<td>sched: Make energy awareness a sched feature</td>
</tr>
<tr>
<td>sched: Documentation for scheduler energy cost model</td>
</tr>
<tr>
<td>sched: prevent unnecessary active balance of single task in sched ...</td>
</tr>
<tr>
<td>sched: Enable idle balance to pull single task towards cpu with ...</td>
</tr>
<tr>
<td>sched: Consider spare cpu capacity at task wake-up</td>
</tr>
<tr>
<td>sched: Add cpu capacity awareness to wakeup balancing</td>
</tr>
<tr>
<td>sched: Store system-wide maximum cpu capacity in root domain</td>
</tr>
<tr>
<td>arm: Update arch_scale_cpu_capacity() to reflect change to define arm64: Enable frequency invariant scheduler load-tracking support</td>
</tr>
<tr>
<td>arm: Enable frequency invariant scheduler load-tracking support</td>
</tr>
<tr>
<td>cpufreq: Frequency invariant scheduler load-tracking support</td>
</tr>
</tbody>
</table>
Additional areas of improvement work

• PELT utilization signal
  ▪ Crucial for determining workload behavior.
  ▪ Input to capacity aware load-balancing decisions and energy estimates.
  ▪ Known issues related to task groups and tickless idle and busy (in particular).
  ▪ Better responsiveness (util_est, Patrick Bellasi).

• Better use of PELT utilization/load in load-balancing
  ▪ Ensure full throughput on big.LITTLE systems.

• Tuning – SchedTune.
  ▪ Tuning cpu selection and frequency scaling.

• User experience optimizations
Next steps

• Most EAS components can now be progressed in parallel.
  - Some have been split out already.
• Clean up existing code to prepare for EAS modifications.
  - Mainline scheduler code contains parts that are fragile and not well understood.
  - Need to build on well understood code.
• Continuously maintain public consolidation branch.
  - Post subsets individually for detailed reviews.
• LSK branch for evaluation.
  - 3.18 backport available today.
  - 4.4 in progress.
Plans for RFCv6

- Repost EAS core patches with fixes and clean-ups.
- Reworked wake-up path code:
  - Split out non-energy model capacity awareness and post separately.
- Better SMP support
  - Topology data for single and multi-cluster SMP.
Challenges in getting EAS-core merged

- Complex topic – but an important direction for Linux to take
  - People get put off by the implementation complexity.
  - Existing scheduler’s complexity.
  - Building a community.
  - Validation data needs to come from products with sensitivity to product requirements (interactivity, use-cases, standard perf benchmarks, etc.).
  - Not doing so risks putting stuff in the mainline that compromises the product story.

- Working with products is challenging
  - The holy trinity (product grade silicon, with power measurement ability, with a close to mainline kernel, with product software stack)
  - Lots of silicon vendors looking at EAS for product, but very limited support on LKML.
Supporting EAS upstreaming

- Review and active discussions on LKML
- Product testing & validation
- Testing and feedback
  - Preferably on public mailing lists.
CFS code age

- $ git checkout v4.5-rc5
- $ git blame kernel/sched/fair.c

- Load-balance code so far largely unchanged despite fundamental change in load/utilization expression.