CPUFreq and The Scheduler

Revolution in CPU Power Management

Rafael J. Wysocki

Intel Open Source Technology Center

August 24, 2016

Intel OpenSource TECHNOLOGY CENTER

Outline

Intel OpenSource TECHNOLOGY CENTER

CPU Power Management

Goal

Optimize energy consumption by CPUs.

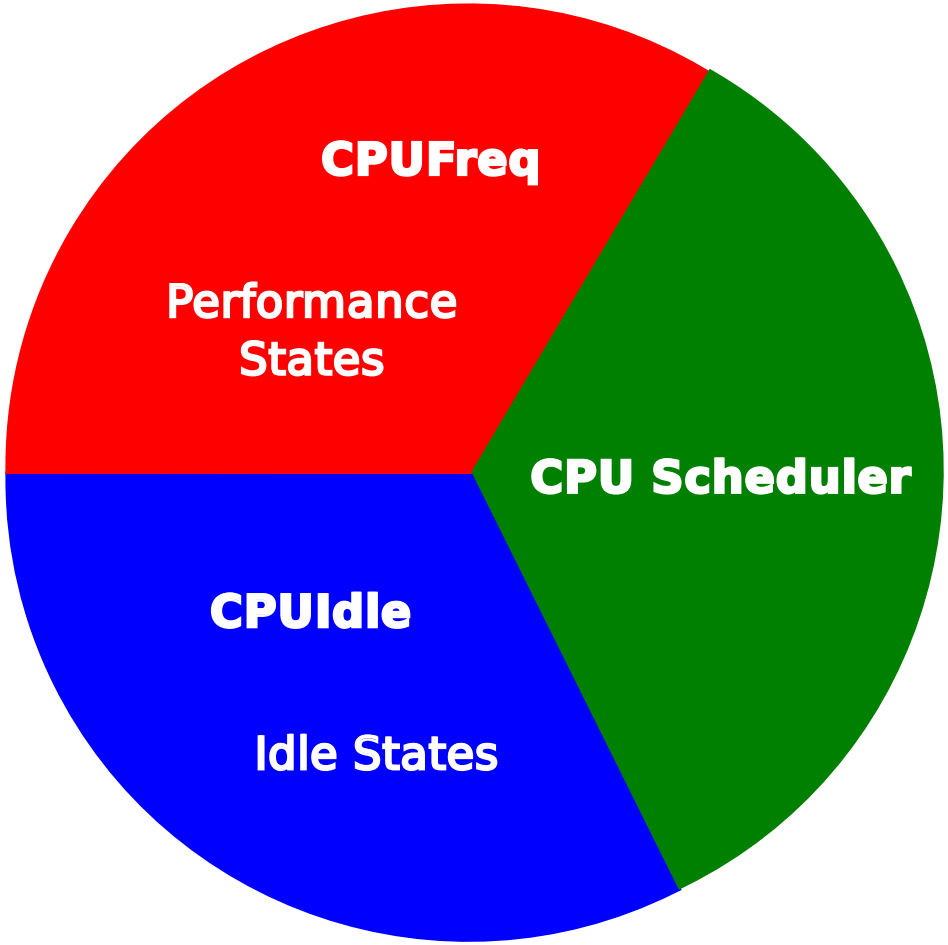
Observations

- Many workloads require a specific level of performance.
- Capacity provided beyond the required level is excessive.

Strategy

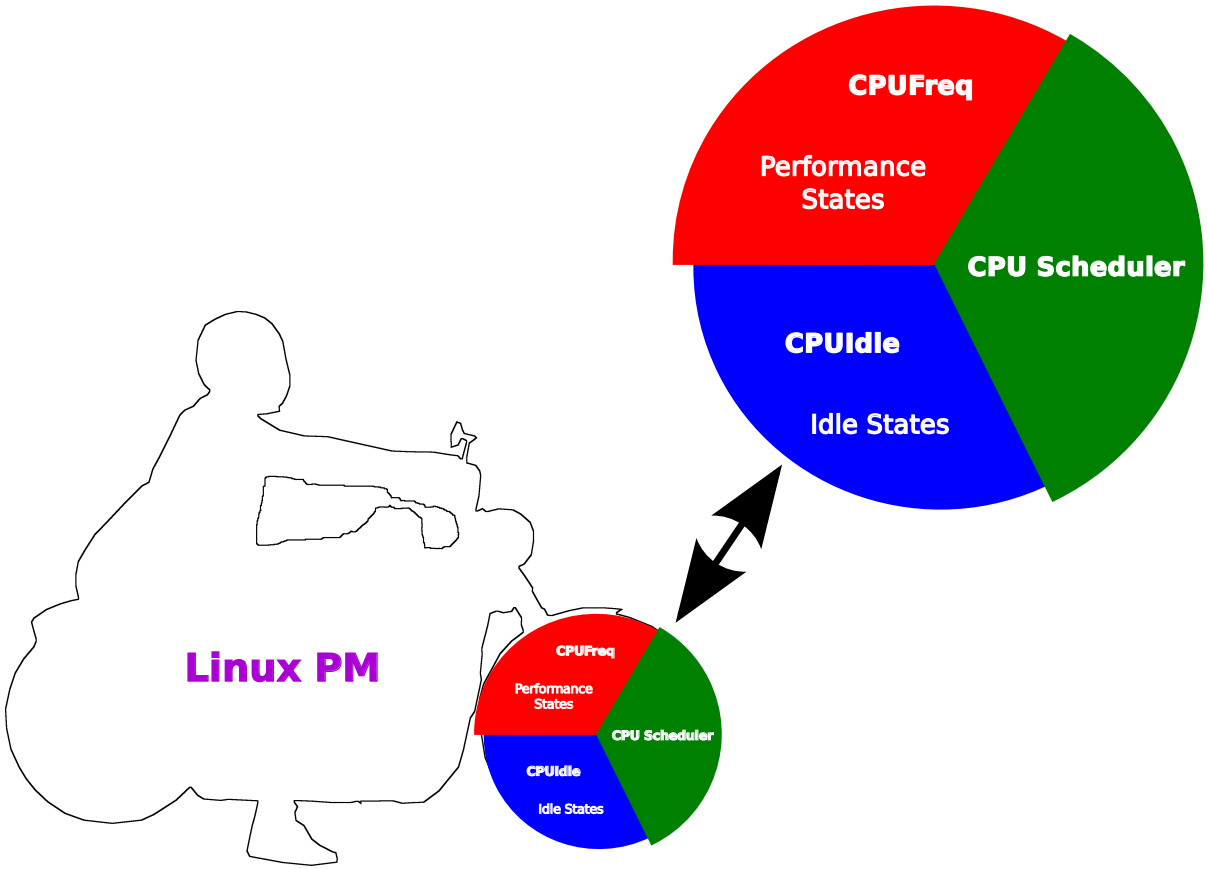
Avoid over-provisioning workloads.

Components of CPU Power Management in Linux*



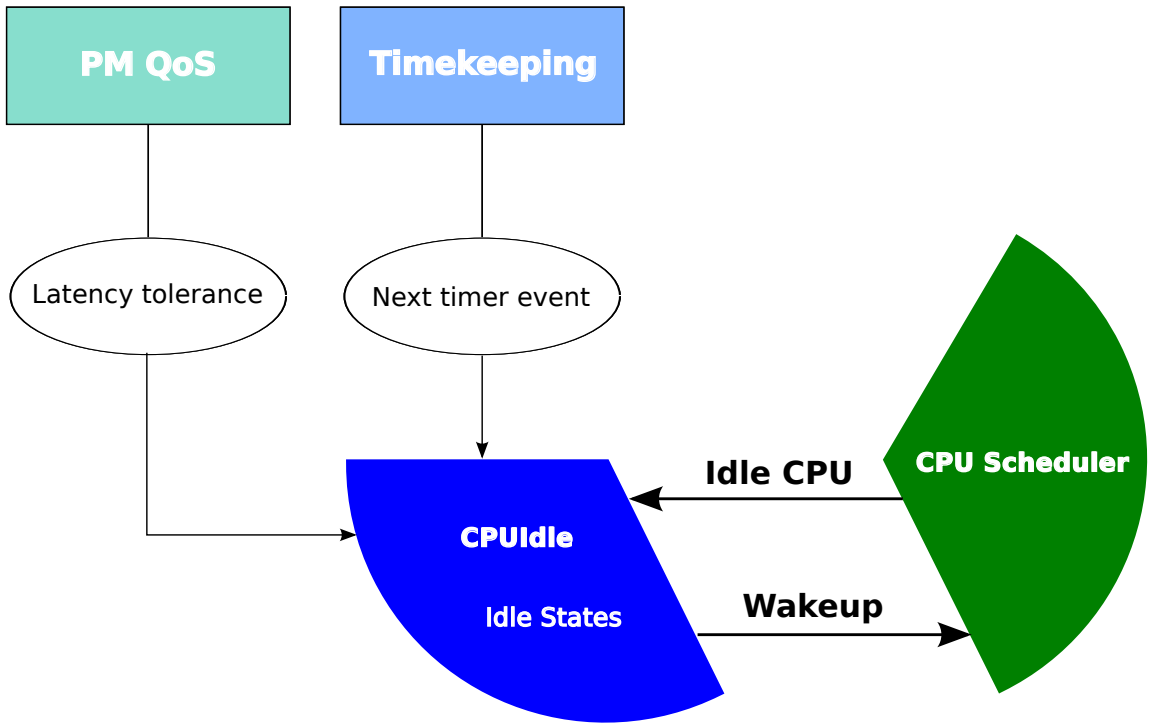
Intel OpenSource TECHNOLOGY CENTER

Cooperation Between Components Is Key



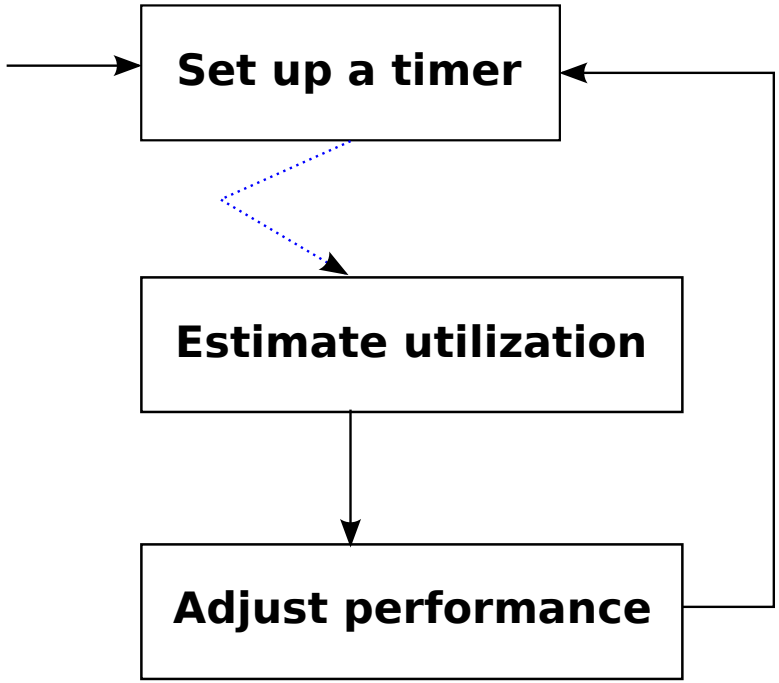
Intel OpenSource TECHNOLOGY CENTER

CPUIdle and The Scheduler



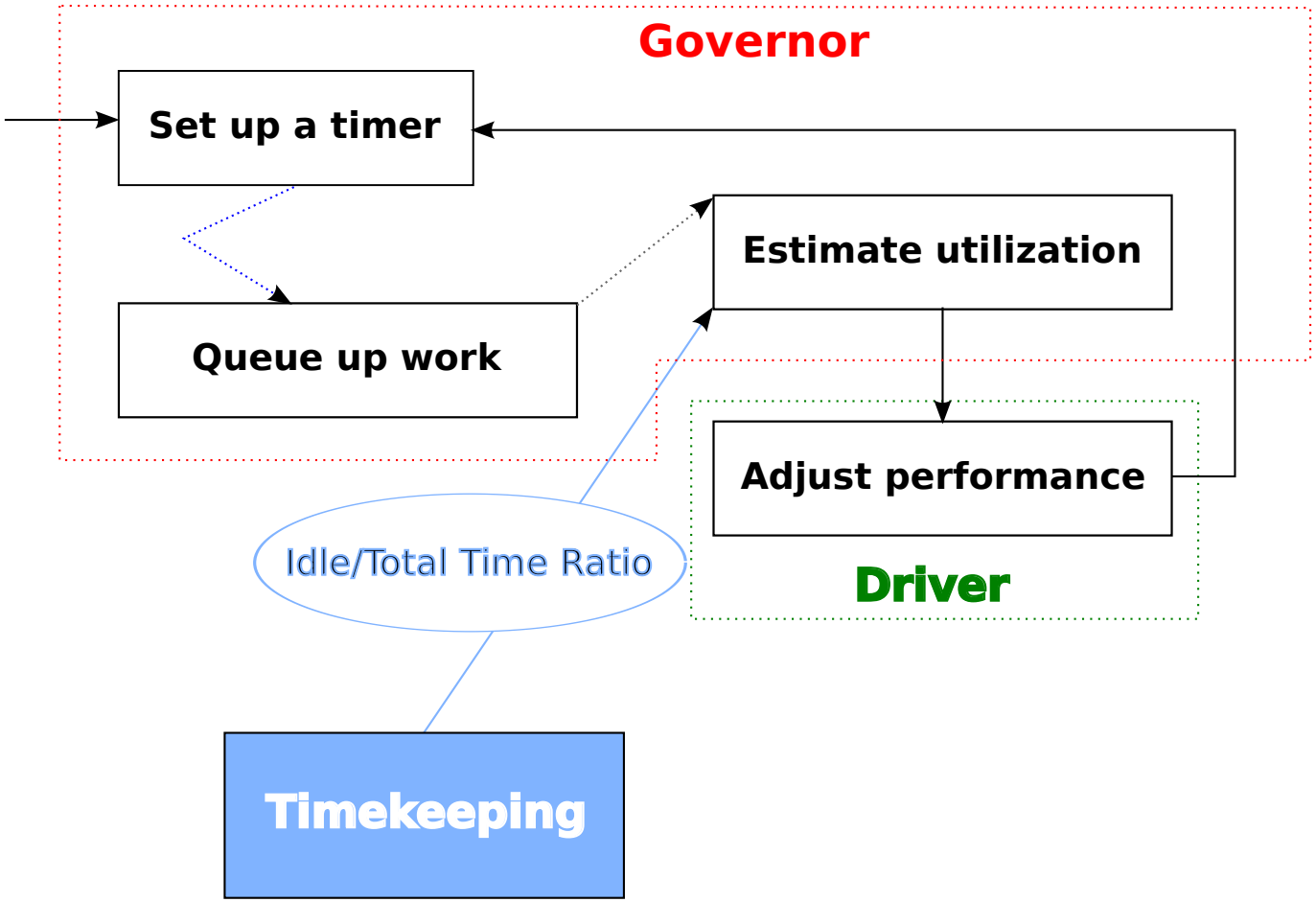
Intel OpenSource TECHNOLOGY CENTER

CPUFreq: `intel_pstate` (Linux 4.5 and Earlier)



Intel OpenSource TECHNOLOGY CENTER

CPUFreq: Generic Governors (Linux 4.5 and Earlier)

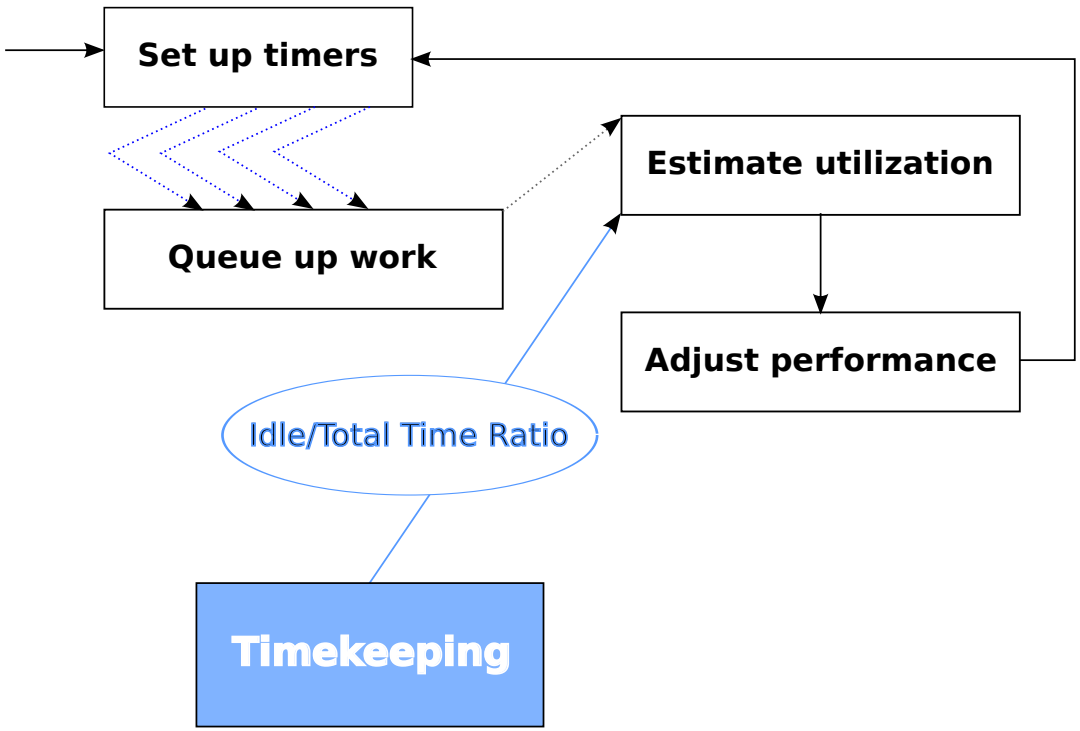


Intel OpenSource TECHNOLOGY CENTER

CPUFreq: CPUs and Policy Objects

CPUFreq policy

Set of CPUs to be handled together.



Intel OpenSource TECHNOLOGY CENTER

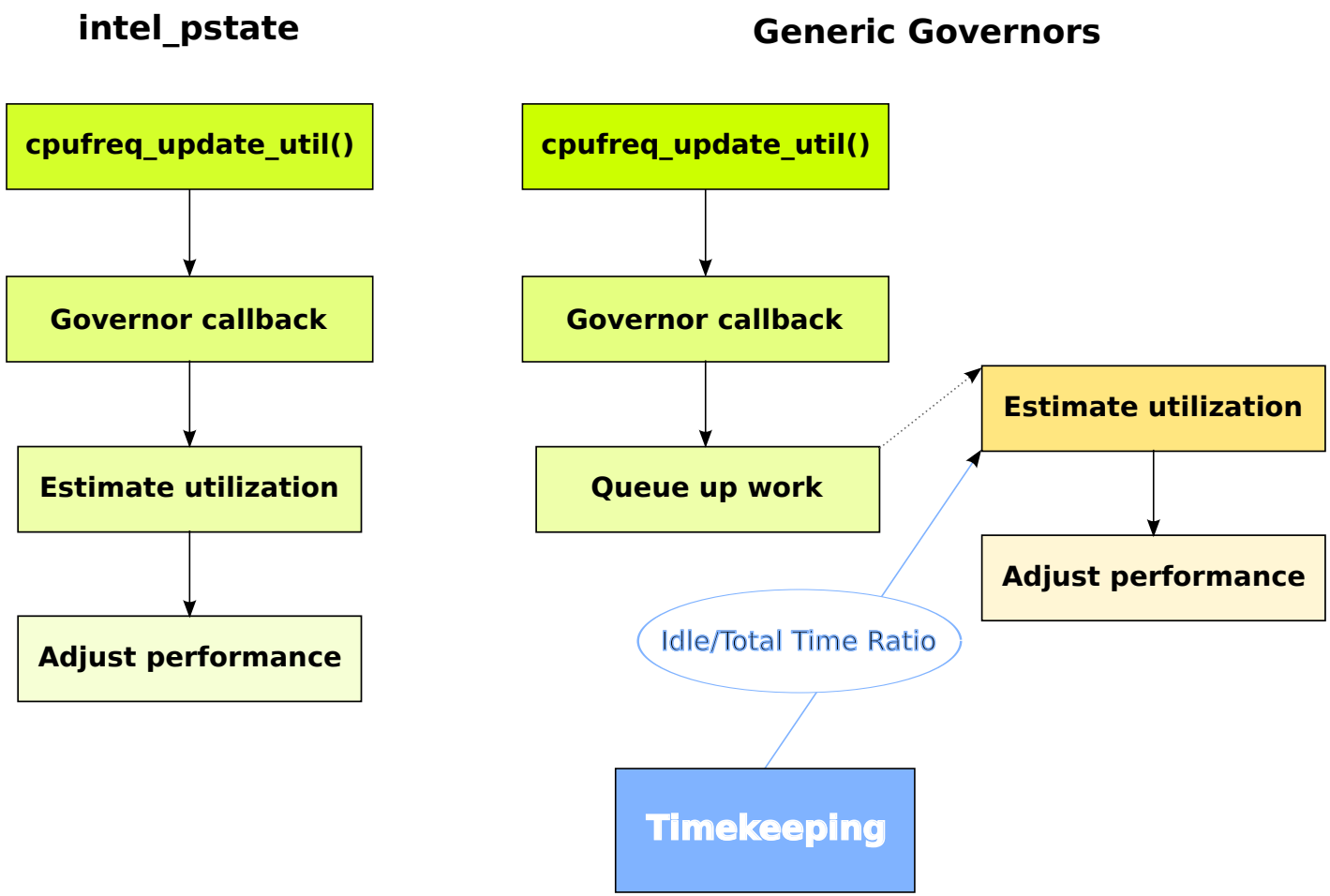
Step 1: Get Rid of The Timers

Scheduler-driven mechanism (Linux 4.6)

- Callbacks invoked by the scheduler: `struct update_util_data`
- Installed per-CPU
- Protected by RCU-sched

Intel OpenSource TECHNOLOGY CENTER

CPUFreq: Governor Callbacks Invoked by The Scheduler



Intel OpenSource TECHNOLOGY CENTER

Step 2: Use The Scheduler's CPU Utilization Data

CFS: Per-entity load tracking (PELT)

Contribution to system load from each scheduling entity:

$$L = L_0 + L_1 \cdot q + L_2 \cdot q^2 + L_3 \cdot q^3 + \ldots = L_0 + L_{prev} \cdot q$$

where $q^{32} = 0.5$.

Next frequency formula

$$f_{new} = C \cdot f_{max} \cdot L/L_{max}$$

If PELT numbers are not frequency-invariant, $L \approx L_{raw} \cdot f_{current}/f_{max}$

Intel OpenSource TECHNOLOGY CENTER

Frequency Tipping Point Assumption

Frequency tipping point

If PELT numbers are not frequency-invariant, we have:

$$f_{new} = C \cdot f_{current} \cdot L_{raw}/L_{max}$$

and $f_{new} = f_{current}$ is the tipping point condition.

Assumption

Choose $C = 1.25$ for the frequency tipping point at $L_{raw}/L_{max} = 0.8$

Problem

That was a new territory.

Intel OpenSource TECHNOLOGY CENTER

`schedutil` Governor and Fast Frequency Switching

New CPUFreq governor: `schedutil` (Linux 4.7)

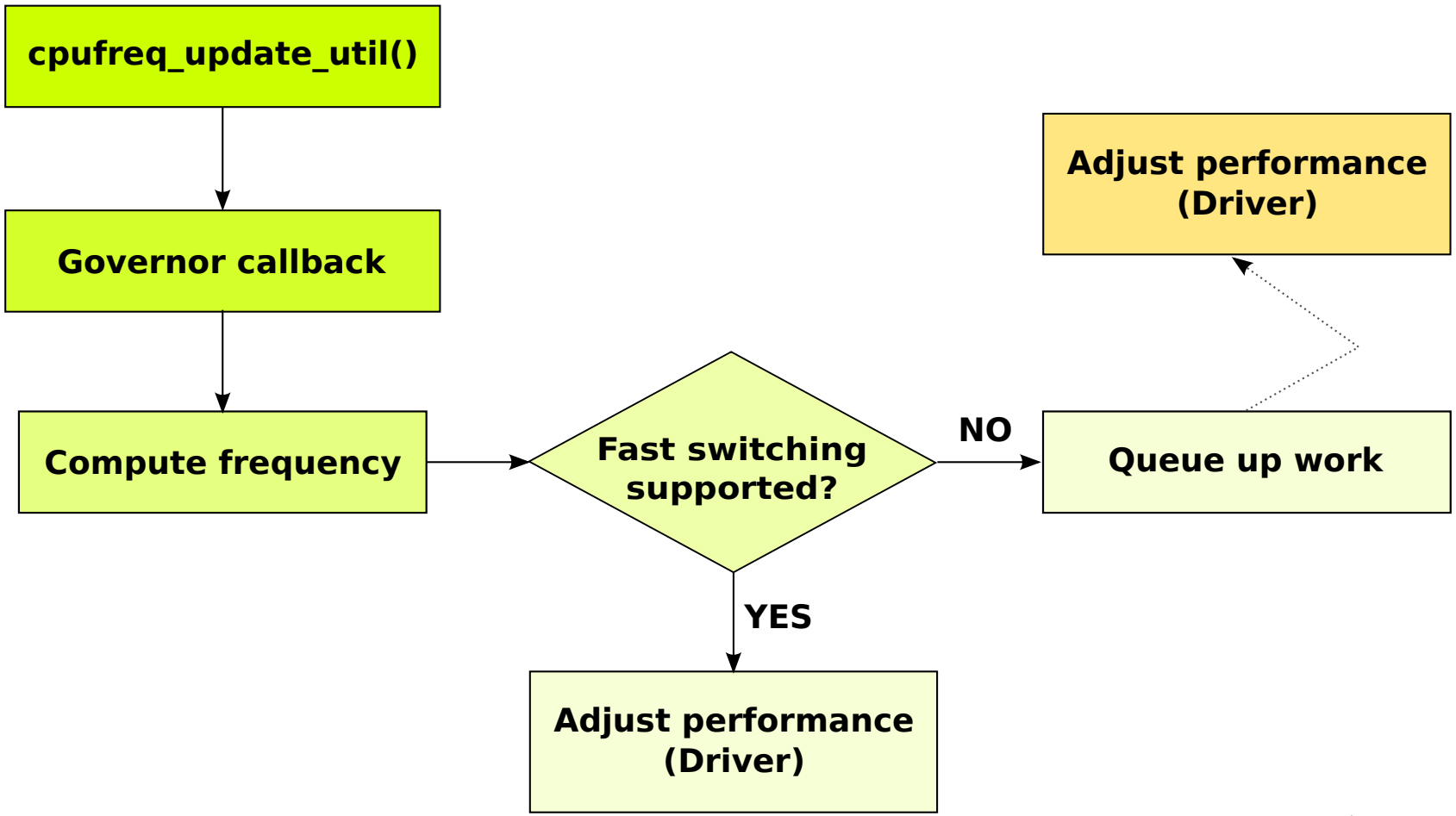
Uses the PELT-based next frequency formula.

Fast frequency switching

- CPUFreq driver feature (in Linux 4.7 supported by `acpi-cpufreq`).
- Allows `schedutil` to adjust CPU performance in the scheduler.

Intel OpenSource TECHNOLOGY CENTER

How schedutil Works



Intel OpenSource TECHNOLOGY CENTER

Step 3: Hints, Cross-CPU Updates Etc.

Problem

No PELT for the real-time (RT) and deadline (DL) scheduling classes.

Observation

It is good to take blocking on I/O (*iowait*) into account.

Idea

- Pass hints from the scheduler to `cpufreq_update_util()`.
- A hint can represent the reason update or similar.

Intel OpenSource TECHNOLOGY CENTER

Updates from Different CPUs

Limitation

Governor callbacks must be run on the CPU being updated.

Without that limitation

Reaction to workload changes may be faster.

Conclusion

- CPUFreq has been scheduler-driven since Linux 4.6.
- It does not use timers any more.
- The `schedutil` governor (Linux 4.7) uses data from the scheduler.
- It is a foundation to build on.

Questions?

Intel OpenSource TECHNOLOGY CENTER

References


Neil Brown, *Improvements in CPU frequency management* (`http://lwn.net/Articles/682391/`).


Jonathan Corbet, *Per-entity load tracking* (`https://lwn.net/Articles/531853/`).


R. J. Wysocki, *Power Management in the Linux Kernel – Current Status and Future* (`http://events.linuxfoundation.org/sites/events/files/slides/kernel_PM_plain.pdf`).


Jonathan Corbet, *The cpuidle subsystem* (`http://lwn.net/Articles/384146/`).

Intel OpenSource TECHNOLOGY CENTER

Legal Information

Intel OpenSource TECHNOLOGY CENTER

Thanks!

Thank you for attention!

Intel OpenSource TECHNOLOGY CENTER