Advances in CPU Performance Scaling

Rafael J. Wysocki

Intel Open Source Technology Center

September 11, 2017
Capacity and Cost
Overview of CPU Power Management in Linux*

- CPUFreq
- Performance States
- CPU Scheduler
- CPUIdle
- Idle States

Rafael J. Wysocki (Intel OTC)
Task Placement and CPU Power Management

- Task placement
- Available CPU capacity
- CPU utilization
- Performance scaling
- CPU idle time
- Idle states selection
- Performance
- Energy usage
General High-Level Control Flow

1. `cpufreq_update_util()`
2. Governor callback
3. In-Band? (YES/NO)
   - YES: Estimate target performance
   - NO: Queue up work
4. Estimate target performance
5. In-Band? (YES/NO)
   - YES: Adjust performance
   - NO: Queue up work

Rafael J. Wysocki (Intel OTC)
Out-of-Band Scaling Governors

On-Demand and Conservative Governors

1. `cpufreq_update_util()`
2. Governor callback
3. Estimate target performance
4. Queue up work
5. Idle/Total Time Ratio
6. Timekeeping
7. Adjust performance
In-Band Scaling Governors

intel_pstate (Active Mode)

- cpufreq_update_util()
- Governor callback
- Estimate target performance
- Adjust performance

SchedUtil Governor

- cpufreq_update_util()
- Governor callback
- Compute frequency
- Fast switching supported?
  - NO: Queue up work
  - YES: Adjust performance (Driver)

Adjust performance (Driver)
Choose a constant $C > 1$ and look for $f_{\text{target}}$ such that

$$1 - \frac{T_{\text{idle}}}{T} = U_{\text{raw}}(f_{\text{target}}) = \frac{1}{C}$$

where $U_{\text{raw}}(f)$ is the frequency-dependent (raw) utilization.

In theory, for a stable workload, $f_{\text{target}}$ can be approximated by iterating

$$f_{\text{new}} = C \cdot f_{\text{current}} \cdot U_{\text{raw}}(f_{\text{current}})$$

Quick quiz: Why isn’t $C$ equal to 1?
Direct Next Frequency Formula

Frequency-invariant utilization

\[ U = \frac{W}{W_{max}} \approx U_{raw}(f) \cdot \frac{f}{f_{max}} \]

- \( W \): Work (number of executed instructions in a given time)
- \( W_{max} \): Work at the maximum frequency \( f_{max} \)

Next frequency formula

\[ f_{new} = C \cdot f_{max} \cdot \underbrace{U}_{f \cdot U_{raw}(f)} \]
SchedUtil Governor and PELT

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 \( q^{32} = 1/2 \).

Next frequency formula (for frequency-invariant PELT)

\[ f_{new} = \frac{1.25 \cdot f_{max} \cdot L_{rq}/L_{max}}{C \cdot U} \]
P-States Interface in Intel Processors

\[ \Delta \text{APERF} < \Delta \text{MPERF} \]

\[ \Delta \text{APERF} > \Delta \text{MPERF} \]

Sustainable  Turbo

P-state
Active Mode of intel_pstate

Next P-state formula (simplified)

\[ p_{\text{new}} = p_{\text{max}} \cdot 0.5 \cdot \left( 1.25 \cdot \frac{\Delta \text{MPERF}/\Delta \text{TSC}}{\% \text{busy}} + \frac{\Delta \text{APERF}/\Delta \text{MPERF}}{\text{current speed}} \right) \]

If \( \% \text{busy} > \text{current speed} \)

\[ p_{\text{new}} = p_{\text{max}} \cdot 1.25 \cdot \frac{\Delta \text{MPERF}/\Delta \text{TSC}}{C_U} \]
Hardware-Controlled CPU Performance States (HWP)

\[ \Delta \text{APERF} < \Delta \text{MPERF} \quad \Delta \text{APERF} > \Delta \text{MPERF} \]

Sustainable \quad Turbo

Guaranteed_Performance

energy_performance_preference

Policy attribute in sysfs for adjusting HWP behavior.

x86_energy_perf_policy

Low-level HWP configuration utility (use with care!).
Scheduler Hints Passed to Governor Callbacks

Currently defined hints

- **SCHED_CPUFREQ_RT**: Update related to an RT task.
- **SCHED_CPUFREQ_DL**: Update related to a deadline task.
- **SCHED_CPUFREQ_IO**: The current task was previously waiting on I/O.

Usage

- `schedutil`: max frequency for **SCHED_CPUFREQ_RT/SCHED_CPUFREQ_DL**
- `schedutil`: boost frequency for **SCHED_CPUFREQ_IO**
- `intel_pstate`: boost P-state for **SCHED_CPUFREQ_IO**
Energy-aware Scheduling

- Task placement
- Available CPU capacity
- CPU utilization
- Performance scaling
- CPU idle time
- Idle states selection
- Performance
- Energy usage

Keep in sync
Information Passing Between OS and Hardware

- Operating System
  - Topology and constraints
  - Available capacity
  - QoS information

- Hardware
  - Information on special conditions
DVFS Interfaces Complexity Problem

Dynamic Voltage and Frequency Scaling

- CPUs may not be mutually independent.
- Updates may require sleeping and/or deferred execution.
- Rails may be shared between CPUs and I/O devices.

Changes under way

Cross-CPU updates in SchedUtil and other governors.
Utilization Metric and Deadline Tasks

RT and deadline tasks are latency-sensitive

- RT requires maximum CPU frequency (or performance).
- The needs of deadline tasks may be estimated.

Goal: Design utilization metric taking deadline tasks into account

- Should reflect the capacity required to meet the deadlines.
- Needs to include PELT for CFS tasks.

Work in progress
Patches posted and discussed.
Questions?


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

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

Disclaimer

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

© Intel Corporation