Evolution of load tracking mechanism in scheduler

Vincent Guittot
Agenda

● Introduction
● Main changes
● Usages
● Next steps
Introduction

● Per Entity Load Tracking (aka PELT)
  ○ Track the “load” of scheduler runqueues and entities
  ○ The time is divided in segment of 1ms (1024us)
  ○ “load” = u_0 + u_1y + u_2y^2 + u_3y^3 + ...
  ○ Geometric series with half period at 32ms (y^32 = 0.5)

● “Load” is made of 3 metrics:
  ○ Util_avg : running time
  ○ Load_avg : runnable time weighted with nice priority
  ○ Runnable_load_avg : For rq, this is the \( \sum \) load of runnable entities
CPU util_avg = ∑Sum entity util_avg

TaskA util_avg

TaskB util_avg
**Introduction**

CPU load\_avg = /Sum entity load\_avg

TaskA load\_avg

TaskB load\_avg

CPU runnable\_load\_avg = /Sum runnable entity load\_avg
Main changes - v4.9

- No load/utilization propagation with migration.
- Freq decreases when task migrates.
- Load sharing inside group not updated correctly.
- Unstable load for small task.
- Stalled load & utilization of idle CPUs.
Main changes since v4.9

● Propagate migration (v4.10)
  ○ Propagate utilization and load across rq tree

● Optimize load computation (v4.12)
  ○ Optimize algorithm
  ○ Increase accuracy of small tasks

● Stabilize load (v4.13)
  ○ Take into account current position in 1ms time window
  ○ Remove noise and instability in load
Main changes - v4.14

Power Consumption or anything else you want to display

- No freq changes when task migrates
- Stable load for small task
- Load sharing inside group not updated correctly
- Load/utilization propagate with migration
- stalled load & utilization of idle CPUs
Latest changes since v4.14

● New propagation mechanism (v4.15)
  ○ Include propagation of runnable load of sched_group
  ○ Improve task group share computation

● Deadline bandwidth (v4.16)
  ○ Implemented deadline “utilization”
  ○ Implemented invariance and OPP selection for SCHED_DEADLINE
Latest changes since v4.14

- Blocked idle (v4.17)
  - Idle CPU might be seen as busy
  - Decay blocked load and utilization

- Util est (v4.17)
  - Save last utilization before sleeping
  - Estimate final CFS utilization level
  - Start at final frequency
Latest changes since v4.14

- RT/DL utilization tracking (v4.19)
  - Track CFS stolen time
  - Track other class utilization

- IRQ utilization tracking (v4.19)
  - Track interrupt activity
  - Estimate full system utilization level
Main changes - v4.19

- Frequency starts directly at final value.
- Load & utilization propagate with migration.
- Load sharing inside group updated correctly.
- Runnable load updated correctly (not displayed here).
- Load & utilization of idle CPUs decay.
Usage of PELT

● Task placement and load balance
  ○ Balance the load across CPU and ensure fair distribution on runtime between tasks
  ○ Detect when CPU has capacity or is overloaded
  ○ Compute spare capacity when selecting a CPU for task wake up
  ○ Compute share of a task group between CPUs

● Schedutil governor
  ○ Scale CPU frequency
  ○ Prevent spurious frequency switch

● Other usage?
Next steps

● Thermal pressure
  ○ Similarly to RT, compute the capacity stolen by thermal mitigation

● Update scale invariance
  ○ Remove the capping of utilization and load by current frequency and micro architecture

● Use HW counter instead of time / frequency / microarchitecture
  ○ Current utilization is an estimation of CPU cycles used by a task
  ○ Can’t make difference between CPU bounded and Memory bounded task
Thanks