The first Operating-System-Directed Power-Management Summit

May 3, 2017
This article was contributed by Dietmar Eggemann

The first Operating-System-Directed Power-Management (OSPM) Summit took place at the ReTiS Lab of the Scuola Superiore Sant'Anna in Pisa on April 3 and 4, 2017. This summit was organized as a collection of collaborative sessions focused on trying to improve how operating-system-directed power management and the kernel's task scheduler can work together to achieve the goal of reducing energy consumption while still meeting performance and latency requirements. This subject is receiving great interest, not least since the advent of energy-aware scheduling (EAS) and heterogeneous CPU designs.

Since such a complex endeavor requires experts from multiple areas of expertise, the summit brought together people from academia, open-source maintainership (Peter Zijlstra, Rafael Wysocki, Thomas Gleixner), and industry (Google, Red Hat, Linaro, NVIDIA, LG Electronics, BayLibre, Huawei, Qualcomm, and ARM). The relaxed atmosphere of the venue and the manageable group of attendees allowed for intense debate and face-to-face discussions about current problems. Several proposals or proof-of-concept solutions were presented during the sessions.

The following chapters, clustered by functional area, summarize the discussions of the individual sessions. Each chapter headline represents a link to a YouTube recording of the corresponding session.

Per-entity load tracking

Latest evolution in PELT. Vincent Guittot started his presentation by showing the improvements on the stability and accuracy of the per-entity load tracking (PELT) signals achieved since v4.9. One of the biggest problems remaining is that an idle CPU can have stale load and utilization values and can therefore appear as loaded or busy even though it has been idle for some time. He proposed changing the frequency- and CPU-invariance mechanism from scaling individual load and utilization contribution values (in response to voltage and frequency changes) to scaling the running time.

Another topic was the possible integration of a PELT-based mechanism for realtime class utilization tracking that would represent time stolen from the completely fair scheduling (CFS) class by realtime tasks. Currently, due to the lack of any utilization signal for realtime tasks, we go to maximum CPU frequency at the first tick after a realtime task is enqueued. When that task blocks and a CFS task starts running, the schedutil CPU-frequency governor receives the (by now decayed) CFS utilization value, which leads to a sudden CPU-frequency drop. This could be avoided by considering the realtime task utilization when invoking schedutil. The mechanism should work similarly to the calculation of the remaining CPU capacity for CFS tasks via `scale_rt_capacity()`, but for utilization instead.

A similar issue that was discussed is the frequency drop that occurs when a CFS task migrates away from an overloaded CPU.

PELT decay clamping/UTIL_EST. Morten Rasmussen showed the results of his PELT decay-clamping implementation, an idea that was first proposed by Paul Turner during the 2016 Linux Plumbers Conference in Santa Fe. The problem is that we lose history for a big task that sleeps for a long time. Decay clamping places a limit on the time for which a task's load-tracking signals can decay after the task sleeps, making the task appear to have slept for less time. The actual implementation revealed some issues with the idea. Since entity load and utilization are accrued in a CFS run queue, they would have to be de-clamped in case the entity is attached or detached to or from the queue. Another issue is that the preempted time of a co-scheduled task is not accounted as sleep time.

Patrick Bellasi demonstrated an alternative approach, UTIL_EST, which keeps PELT as an estimator and adds a new signal, the estimated utilization (hence the name) as a sort-of low pass filter on PELT's derived values. The current proposal tracks both the utilization at task deactivation time as well as an exponential moving average of these values. A series implementing this idea is going to be posted soon on the linux-kernel mailing list for a better evaluation.

Tracepoints for PELT. Dietmar Eggemann explained his patch set, that introduces trace events for the scheduler's load and utilization signals. There is definitely a need to be able to trace the input signals for task scheduler wake-up and load-balancing decisions as well as schedutil cpufreq governor behavior. Several problems with PELT have been fixed over the last few months and having standard tracepoints would have likely helped to locate them quicker. There was feedback that these trace events should support other scheduler classes as well rather than just CFS.

Energy-aware scheduling

EAS: where we are. Morten Rasmussen started with a short introduction on EAS and its importance in maximizing both CPU utilization and power efficiency, along with a short overview of why frequency- and CPU-invariant scaling for PELT load and utilization signals is important. He made it clear that having an energy model is key for scheduling decisions based on energy consumption and that the EAS energy model is the best we can currently do. He continued by pointing out that there is no place to store energy data (energy consumption per operating performance point (OPP) and per idle state at every scheduling domain level) in mainline Linux.

The device tree would be a good place to store such information but there is a lot of resistance from the device-tree community. For them, energy data should describe hardware configuration (in this case CPU core) data, which should be provided by the core designer. Or at least they require a strictly defined method to determine the energy consumption. What they don't want is a situation where different vendors have different values for the same CPU core.

But energy data is a platform-specific property rather than a core-specific one, so each vendor would have to provide it. The methodology to obtain the values for the energy model is to measure the energy consumption while the platform is forced to run a workload on subsets of cores at fixed OPPs or idle at fixed idle states. The workload is not standardized and it is clear that the values gathered are for the instruction mix of this specific workload. This also requires that energy consumption is measurable at the core and cluster scheduling domain levels.

EAS uses the relative numbers of the energy model; in other words, the energy model is like a single tuning knob for EAS. This is another reason why the device-tree folks argue that it does not belong into the device tree. A possible simplification of the energy model based solely on CPU-level data and not on aggregate data (e.g. on cluster-level) might be the way forward.

Schedtune. Patrick Bellasi presented the latest version of the per-task capacity clamping series recently posted on the mailing list and described in this LWN article. People seem to agree on the usefulness of a better interface to collect task-related information from an informed run-time system. However, whether this interface should be the one proposed is not yet agreed. The main problem is the identification of a proper level of abstraction that achieves the goals without exposing too many implementation and platform details.

Other discussion points were about the use of the proposed features outside of the Android-specific domain. There are other run-time systems (e.g. ChromeOS and Kubernetes) that might be interested in the proposed interface. Moreover, it can be used to improve existing features such as avoiding a jump to the maximum frequency every time a realtime task runs. Alternative implementations to boost the minimum capacity have been discussed but some of them were already rejected in the past for various reasons. There was no agreement that the control-group-based interface would be sufficient so it is likely that a per-task interface will have to be added. Overall the discussion was useful to make key maintainers aware of the proposal.

Schedutil governor improvements. Rafael Wysocki led a discussion about issues related to the current frequency change-rate limiting in schedutil. This CPU-frequency governor has been around for a year now, people are starting to use it, and bug reports are beginning to appear. He is keen to address those issues now so that people don't lose interest in deploying schedutil.

The rate-limiting mechanism, designed to prevent frequency updates from happening too often (which may lead to some undesirable effects), is arguably suboptimal, but some relatively inexpensive improvements appear to be possible. A few of them were discussed during the session, including allowing cpufreq drivers to instruct the governor to use shorter intervals between consecutive frequency updates by default, aggregating utilization updates that occur between consecutive frequency updates, and making the governor reduce frequencies less aggressively.

There was also some brief discussion regarding other schedutil improvements proposed so far, including cross-CPU utilization updates.

Energy model & exotic topologies. Brendan Jackman gave an overview of the energy model that EAS uses to estimate the energy impact of a task-placement decision. He mentioned that a simplified version of the model, one that considers each CPU in isolation, may be the best approach for initial upstreaming of EAS. He then briefly described some of the key features of ARM's new DynamIQ cluster architecture and the challenges they present to Linux's OSPM and task scheduler implementations.

Realtime scheduling

`SCHED_DEADLINE` and reclaiming. Luca Abeni gave an overview of his work implementing the GRUB bandwidth reclaiming algorithm for `SCHED_DEADLINE`. In fact, one problem with using `SCHED_DEADLINE` is that it is not usually easy to come up with the right set of parameters (execution time, deadline, and period) to describe task behavior. Bandwidth reclaiming helps with this by allowing a task that occasionally needs more CPU time than its parameters would allow to reclaim bandwidth left over from other tasks. Another good reason to merge this patch is that it adds information about how much CPU capacity is required to service the deadline tasks. This information is key for proper clock-frequency selection (see below).

Luca's patch set is pretty mature, so he should be able to get it merged relatively quickly after having addressed a few comments received during the discussion and as part of the review of the latest version on the list.

Schedutil for `SCHED_DEADLINE`. Juri Lelli and Claudio Scordino described their recent patch set that implements frequency- and CPU-invariance as well as clock-frequency selection for `SCHED_DEADLINE`. The scale invariance bits seemed uncontroversial since they are based on the same assumption as PELT's scale invariance (run time scales linearly with current capacity).

The changes required to modify clock frequency for SCHED_DEADLINE tasks, at least on platforms that require a non-atomic context to implement clock scaling (like ARM), generated a lively discussion instead. A major point of contention was the fact that the kernel threads responsible for changing clock frequency have to be able to preempt all other tasks, including SCHED_DEADLINE tasks. The currently proposed solution is based on a mechanism that treats these threads as "special"; they appear as SCHED_DEADLINE tasks with an immediate deadline and infinite bandwidth. While this seems to work in practice, it breaks admission control and priority inheritance from a theoretical point of view. However, the agreement seems to be that a solution that works in practice might be accepted to foster adoption and further experimentation with the schedutil governor while a better, longer-term approach, such as making clock-frequency changes atomic even on ARM, is sought.

SCHED_DEADLINE group scheduling. Luca Abeni presented a live demonstration of an RFC implementation of SCHED_DEADLINE group scheduling (the set was recently posted on the list by Alessio Balsini and is based on patches originally implemented by Andrea Parri). This patch set provides hierarchical realtime scheduling by nesting SCHED_{FIFO/RR} fixed-priority scheduling entities within SCHED_DEADLINE reservations, allowing for groups of tasks to be scheduled inside a single SCHED_DEADLINE entity. The informed reader should notice some resemblance with the base idea of the “old” IRMOS scheduler; such a reader wouldn’t be too far from being correct, as IRMOS certainly inspired the current set of patches. In its current form the mechanism is also proposed as a replacement for realtime throttling.

The RFC still has to ripen quite a bit (starting with proper changelogs), but the necessity of extending SCHED_DEADLINE to deal with groups of tasks seemed to be acknowledged by the people attending the summit. However, the key question on how to properly integrate this new feature still has to be answered: how to make it work even when group scheduling is not in use.

Power management

Scheduler decisions regarding idle. Daniel Lezcano presented some ideas on how the interrupt-prediction functionality could be used to guide the scheduler's wakeup and load-balancing decisions to guarantee maximal energy savings by leaving CPUs and clusters in deeper idle states. This mechanism would have to be switched on via a scheduler feature flag indicating that the user considers saving energy as more important than performance.

It is implemented with the following four changes:

1. Prevent waking a CPU or cluster that hasn't reached its target residency yet (a CPU entering an idle state consumes more energy if the break-even point hasn’t been reached).
2. A newly idle CPU doesn’t pull tasks if it is the last CPU in the cluster, since there is a high probability the entire cluster will be powered down.
3. A newly idle CPU pulls tasks from another cluster if this allows that cluster to go idle.
4. Compute the average time it takes to execute the different steps (heuristic, entering idle state, switch timers, etc.) of the idle path. This will allow evaluation of whether it is worth entering a deeper idle state through the costly idle path or if we should bail out sooner.

These features can only work if we change the way how we compute the idle duration and if we have the next-event information available when the task scheduler makes its task placement decision. The change of the next event from average to deadline duration will allow for higher precision, since we then see how close a CPU or cluster is from being woken up.

A unified solution for SoC idling. Ulf Hansson mentioned that, when it comes to saving energy on a SoC, there are many devices besides the CPU that can enter low-power states. He presented an approach to unify runtime and system-idle management scenarios under a single runtime PM framework. One of the steps is to extend runtime PM and generic PM domain support to include CPUs to be able to manage idle states across the entire SoC.

Miscellaneous

Parameterizing CFS load balancing. Dietmar Eggemann gave an overview of the CFS scheduler's load balancer. The use of different input signals in the applied heuristics, the support of multiple platforms with, possibly, specific features (such as asymmetric packing), and the dependency of the code on input signals that are no longer used makes it difficult to understand. He argued that it should be possible to streamline the existing code in such a way that, after the statistics-gathering phase, the load-balancer can work only on one input signal, which could be any of the number of tasks, the utilization, or the load.

I/O scheduling and power management with storage devices. Luca Miccio started the session by presenting the current status of the no-op, CFQ, and deadline I/O schedulers. He then went on highlighting problems (showing a recorded demo of several Android workloads running on a HiKey board) that existing schedulers are faced with when applications have low latency requirements. The proposed solution for these issues is the Budget Fair Queuing (BFQ) scheduler, which seemed (still looking at the same recorded demo) capable of providing low latency even in the presence of background I/O. He announced that BFQ is finally expected to land in the mainline during the v4.12 merge window in its BFQ-MQ incarnation.

Luca then handed over to his professor Paolo Valente, who moderated the more forward-looking and open-ended part of the session, trying to find intersections between I/O scheduling and OSPM. Starting with the current state of the art (a black slide causing the audience to giggle) he then gathered feedback around possible strategies for saving energy, such as idling during I/O or controlling components' OPPs.

Tooling

Tooling/LISA. Patrick Bellasi and Brendan Jackman showed how ARM is testing EAS behaviors. They presented the rt-app tool used to create synthetic workloads and the LISA toolkit that provides regression testing and interactive analysis of Linux kernel behavior.

The need to power-instrument the Linux kernel. Patrick Titiano stated that today we're not able to do continuous-integration tests on energy consumption even though there is clear evidence that this is becoming more and more important. This is mainly due to the fact that today's devices and the majority of the development boards are not equipped for power measurements so people still rely on hardware modifications. There was overall agreement that another infrastructural bit missing is a standardized energy (consumption) model for a device.

Research

A hierarchical scheduling model for dynamic soft-realtime systems. Vladimir Nikolov presented a user-space based proposal for runtime administration of concurrent soft real-time applications, which is part of the ARTOS research project. The presented approach involves mechanisms for automated application monitoring and cost estimation, extraction of required processing bandwidth, and selection of optimal quality levels for the applications during runtime. Thereby, capacity reservations are established and updated at well-defined instants under the premise that applications and their tasks remain consistent with their actual measured costs. The system's ability to handle cyclically occurring loads and to suppress recurring re-configurations of application quality levels were discussed as well. An interesting discussion was triggered around the abstraction levels proposed as well as the usefulness of an out-of-kernel solution.

Conclusions

This event has shown that there is a real need for detailed technical discussion on how to improve Linux OSPM and task scheduler coordination in addition to the limited amount of time available at the yearly Linux

Plumbers Conference. Many of the sessions helped to accelerate the corresponding discussions on the linux-kernel mailing list. The organizers received positive feedback from the participants about the usefulness of such an event so they will look into scheduling a follow-on summit next year.

Many thanks to all presenters, participants, and note-takers as well as to the ReTiS Lab for providing the premises.

(Log in to post comments)