Improving scheduling latency with SCHED_IDLE task

Viresh Kumar (Linaro)
Scheduling classes and policies (recap)

- **Stop**
  - No policy
- **Deadline**
  - SCHED_DEADLINE
- **Real Time**
  - SCHED_FIFO
  - SCHED_RR
- **Fair (CFS)**
  - SCHED_NORMAL
  - SCHED_BATCH
  - SCHED_IDLE
- **Idle**
  - No policy
SCHED_IDLE scheduling policy

- One of the scheduling policies available in CFS.
- Nice range -20 to +19, where +19 means lowest priority.
- Priority of SCHED_IDLE tasks is lower than tasks with +19 nice.
- Useful for background tasks with lowest priority in system.
- Any non-idle task can preempt a running SCHED_IDLE task.
- It is still not a real idle time scheduler and still gets a chance to run on a fully busy system to avoid priority inversion.
- Not widely used currently. That needs to change.
What’s new?

Before:
- Special handling only in ->check_preempt_curr() in fair.c
  - A SCHED_OTHER task will preempt a SCHED_IDLE task.

Now (Merged in 5.4-rc1):
- Special handling added in ->select_task_rq() in fair.c
  - Both fast and slow paths updated
  - CPU running only SCHED_IDELE tasks considered as idle
    - Will immediately get preempted by a SCHED_NORMAL task.
  - Improves scheduling latency for the SCHEDOTHER task
    - Even better than running on an idle-cpu as we don’t need to wake-up the CPU.
rt-app json, 8 normal and 5 sched-idle threads

```json
{
    "tasks": {
        "cfs_thread": {
            "instance": 8,
            "run": 5333,
            "timer": { "ref": "unique", "period": 7777 },
            "policy": "SCHED_OTHER"
        },
        "idle_thread": {
            "instance": 5,
            "run": 3000,
            "policy": "SCHED_IDLE"
        }
    },
    "global": {
        "duration": 5,
        "calibration": "CPU0",
        "default_policy": "SCHED_OTHER",
        "pi_enabled": false,
        "gnuplot": false
    }
}
```
Wu-latency (usec) from rt-app for CFS tasks

Hardware: Octa core cortex A7

Without the patchset:

<table>
<thead>
<tr>
<th>N</th>
<th>min</th>
<th>max</th>
<th>sum</th>
<th>mean</th>
<th>stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>4710</td>
<td>0</td>
<td>67664</td>
<td>5.25956e+06</td>
<td>1116.68</td>
<td>2315.09</td>
</tr>
</tbody>
</table>

With the patchset:

<table>
<thead>
<tr>
<th>N</th>
<th>min</th>
<th>max</th>
<th>sum</th>
<th>mean</th>
<th>stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>5095</td>
<td>0</td>
<td>7773</td>
<td>523170</td>
<td>102.683</td>
<td>475.482</td>
</tr>
</tbody>
</table>

mean latency dropped by 90% and stddev dropped by 75%
Kernelshark without sched-idle modifications
Kernelshark without sched-idle modifications ...
Kernelshark without sched-idle modifications ...
Kernelshark with sched-idle modifications
Kernelshark with sched-idle modifications
Who should use SCHED_IDLE policy?

- SCHED_IDLE policy isn’t widely used currently.
- Can be used for most of background tasks which aren’t time critical.

- Google is interested in using SCHED_IDLE policy for background Android tasks, like dex2oat (compiles dex files).

- Facebook’s use-case involves using spare CPU cycles on servers (running latency sensitive workloads) to run side jobs like video encoding but that interferes with main workload. Making the side jobs SCHED_IDLE has proven to be very useful.
Thank you

Join Linaro to accelerate deployment of your Arm-based solutions through collaboration

contactus@linaro.org