Real-Time Systems Laboratory

Real-Time Scheduling on Linux and SCHED_DEADLINE

ISTITUTO
DI TECNOLOGIE DELLA
COMUNICAZIONE,
DELL'INFORMAZIONE
E DELLA
PERCEZIONE

Scuola Superiore
Sant'Anna

Linux CPU Schedulers

Scheduling classes and policies

- Classes listed in priority order, from STOP (highest) to Idle (lowest)
- STOP
  - → Only used by migration/N kthread
  - → Needed for hotplug, ...
- DEADLINE
  - → Since v3.14 (2013)
  - → Hard CBS + G-EDF
- POSIX
  - → Fixed Priority
  - → 100 rt-priority levels
- Fair (CFS, virtual time-based for fairness)
- Idle
  - → Only used by CPU idle kthread, bringing the CPU to low-power

Class	Policy
STOP	--
Deadline	SCHED_DEADLINE
Real-Time (POSIX)	SCHED_FIFO
	SCHED_RR
Fair	SCHED_NORMAL
	SCHED_BATCH
	SCHED_IDLE
Idle	--

Devil's in the details

RT Throttling

```
$ grep -H '' /proc/sys/kernel/sched_rt_*
/proc/sys/kernel/sched_rt_period_us:1000000
/proc/sys/kernel/sched_rt_runtime_us:950000
```

- budget exhausted => task is **throttled** till next period

SMP

- Each CPU has its own set of runqueues

```
struct rq {
    ...
    struct cfs_rq cfs;
    struct rt_rq rt;
    struct dl_rq dl;
    …
}
```

- when a task wakes up, it or the current task can be pushed to another CPU
- when a task suspends, a task can be pulled from another CPU

Further devil's details

Frequency switching

- cpufreq
- Intel P-state

Memory locking

- mlock(), mlockall()

CPU deep idle states

- `cpuidle` & wake-up latency

Disabling power management

We always need to

- Disable Turbo Boosting

```
sudo bash -c "echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo"
```

- Set CPU governor to performance, for all CPUs

```
for c in $(ls -d /sys/devices/system/cpu/cpu[0-9]*); do
    echo performance > $c/cpufreq/scaling_governor
done
```

When using cpufreq, eg, old Intel or AMD CPUs

- nothing else

When using Intel P-state, eg, recent Intel CPUs

- Set min and max perf_pct to same value, below turbo boosting threshold

```
turbo_pct=$(cat /sys/devices/system/cpu/intel_pstate/turbo_pct)
perf_pct=$[ 100 - $turbo_pct ]
echo $perf_pct > /sys/devices/system/cpu/intel_pstate/max_perf_pct
echo $perf_pct > /sys/devices/system/cpu/intel_pstate/min_perf_pct
```

SCHED_DEADLINE

```
struct sched_attr attr = {
   .size = sizeof(struct sched_attr),
   .sched_policy = SCHED_DEADLINE,
   .sched_flags = 0, // SCHED FLAG RECLAIM
   .sched_runtime  =  runtime_us * 1000,
   .sched_deadline = deadline_us * 1000,
   .sched_period   = period_us * 1000
};


if (sched_setattr(0, &attr, 0) < 0) {
   perror("setattr() failed");
   exit(-1);
}
```

SCHED_DEADLINE

```
#define gettid() syscall(__NR_gettid)

#define SCHED_DEADLINE          6
#define SCHED_FLAG_RESET_ON_FORK 0x01

/* use proper syscall numbers */
#ifdef __x86_64__
#define __NR_sched_setattr      314
#define __NR_sched_getattr      315
#endif

#ifdef __i386__
#define __NR_sched_setattr      351
#define __NR_sched_getattr      352
#endif

#ifdef __arm__
#define __NR_sched_setattr      380
#define __NR_sched_getattr      381
#endif
```

```
struct sched_attr {
  __u32 size;

  __u32 sched_policy;
  __u64 sched_flags;

  /* SCHED_NORMAL, SCHED_BATCH*/

  __s32 sched_nice;

  /* SCHED_FIFO, SCHED_RR */
  __u32 sched_priority;

  /* SCHED_DEADLINE (nsec) */
  __u64 sched_runtime;
  __u64 sched_deadline;
  __u64 sched_period;
};
```

pthreads interface

SCHED_DEADLINE hates pthreads!

- `sched_setattr()` needs a Linux Thread ID
- father doesn't know its children tid after `pthread_create()`
- children can retrieve their tid via `gettid()` syscall
- SCHED_DEADLINE & pthreads
  - → the easy way (good)
    - child sets its own scheduling class & params on its own
  - → the hard way (bad)
    - child retrieves its tid, communicates it to father
    - father sets scheduling class & params for the child (needs synchroniz.)
  - → the fancy way (ugly)
    - father inspects its own `/proc/<pid>/task/<tid>` filesystem to infer children tid after a `pthread_create()`
    - father sets scheduling class & params for the child

Real-Time Systems Laboratory

Thanks!

Questions ?

tommaso.cucinotta@santannapisa.it

http://retis.santannapisa.it/~tommaso