INSTITUTE
OF COMMUNICATION,
INFORMATION
AND PERCEPTION
TECHNOLOGIES

Scuola Superiore
Sant'Anna

Retis
Real-Time Systems Laboratory

RT bandwidth constraints enforced by hierarchical DL scheduling

Luca Abeni, Alessio Balsini,
Tommaso Cucinotta

Scuola Superiore Sant'Anna, Pisa

Motivations

Real-time guarantees

- replace **current RT throttling**, designed purely as a runtime **limitation** mechanism
- **new mechanism**, based on SCHED_DEADLINE, trying to provide **guarantees** to groups of RT tasks
- from different viewpoint: SCHED_DEADLINE for groups of RT tasks
- goal: use hierarchical RT analysis to check whether all tasks will meet their deadlines

Code simplification

- remove a lot of code from rt.c
- especially “dangerous” code (runtime share)

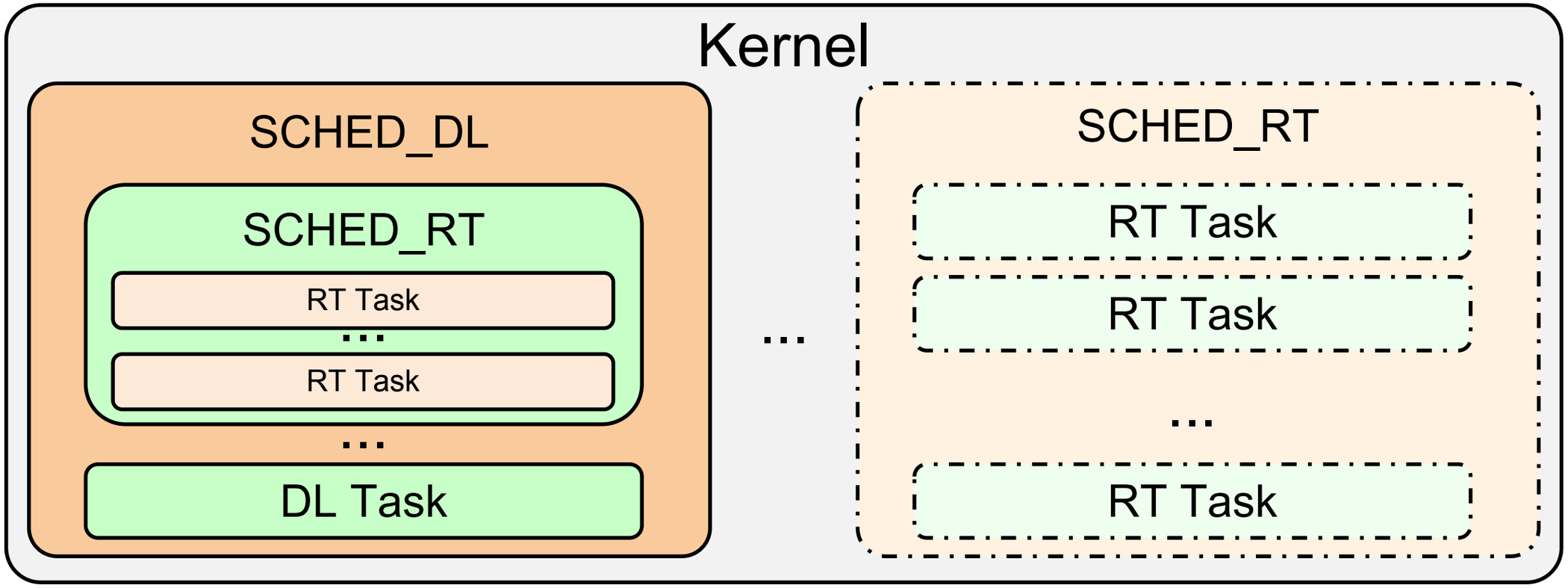
Features at a glance

- **2-levels scheduling hierarchy**
  - hierarchical EDF+CBS/FP scheduling
- **cgroup-based interface**
  - somehow compatible with existing rt throttling
- **multi-processor group reservations**
  - do not migrate runtime, migrate tasks!
  - can reuse existing real-time analysis (multi-supply functions)
- **use dl scheduling entities for “scheduling” RT runqueues**
  - no migrations (scheduling entities have no afffinities, etc…)

Hierarchical EDF+CBS/FP scheduling

2-levels scheduling hierarchy

- An **RT CGroup** is scheduled as a **special DL entity**
- **Period** and **runtime** are assigned by the Cgroup interface, **deadline** implicitly equal to the period



DL Scheduling entities

“Regular” DL Entities

- connected to DL tasks
- can migrate (G-EDF)
  - affinity = whole root domain
  - all the CPUs of the cpuset
- support deadline<period

DL Entities representing RT cgroups

- connected to RT runqueues
- bound to a single CPU
  - a DL entity per CPU
  - all with the same parameters
- implicit deadline=period

Issue: Admission control!

- For the moment, only guarantee no overload

Patchset overview

- *lkml submission (31 March 2017): 1490982277-4437-1-git-send-email-a.balsini@sssup.it*

- patchset with 3 patches
  1. remove RT throttling & cgroup-related from sched/rt.c
  2. hierarchical DL scheduling of RT groups
  3. allow RT tasks to migrate between the RT runqueues of the control group
- git metrics
  - 7 files changed, 837 insertions(+), 1036 deletions
  - deletions mainly in rt.c
  - to tell the whole story, some functionalities are removed

cgroup-based interface

Example of usage

```
mkdir /cgroup/cpu/rt1
echo 100000 > /cgroup/cpu/rt1/cpu.rt_period_us
echo 10000 > /cgroup/cpu/rt1/cpu.rt_runtime_us
echo $tid1 > /cgroup/cpu/rt1/tasks
echo $tid2 > /cgroup/cpu/rt1/tasks
chrt -r -p $rtprio1 $tid1
chrt -r -p $rtprio2 $tid2
```

Overall available runtime for group on M cores

`M * cpu.rt_runtime_us` **every** `cpu.rt_period_us`

Current Status

Very Preliminary code

- Works well, but has some style issues
- The design has to be discussed

Posted early to discuss it

- Is this what people want / need?
- Is the design acceptable?
- Is it ok to have the same period / runtime on all CPUs?

Examples of open issues

- Push/Pull code is duplicated
- Admission test
- Very large patches...

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

Retis Real-Time Systems Laboratory

Thank you!

Luca Abeni

luca.abeni@santannapisa.it

INSTITUTE OF COMMUNICATION, INFORMATION AND PERCEPTION TECHNOLOGIES

Scuola Superiore Sant'Anna

Retis Real-Time Systems Laboratory

Backup Slides

Why SCHED_DEADLINE / CBS

Reservation-based scheduling

- guarantee of Q time units every P time units

CBS temporal isolation

- with FP, high priority tasks can arbitrarily delay low priority tasks
- with CBS/EDF, attempt to exceed the runtime results in being throttled
- CBS/EDF wake-up rule

Example of hierarchical analysis

2 tasks within a reservation

- priority, period, WCET of the 2 tasks
- Q, P parameters

schedulability analysis check

- magic formula...