

TOSHIBA

Leading Innovation >>>

Deadline Miss Detection with SCHED_DEADLINE

Yoshitake Kobayashi

Advanced Software Technology Group
Corporate Software Engineering Center

TOSHIBA CORPORATION

Resources

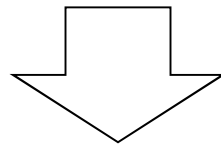
- **The latest version slide will be available on elinux wiki**
 - http://elinux.org/ELC_2013_Presentations
- **Related source code can be downloaded from GitHub.**
 - <https://github.com/ystk/sched-deadline/tree/dlmiss-detection-dev>

Outline

- Motivation
- Deadline scheduler
- SCHED_DEADLINE and its evaluation
- Budget management
- *Deadline miss detection*
- Conclusion

Motivation

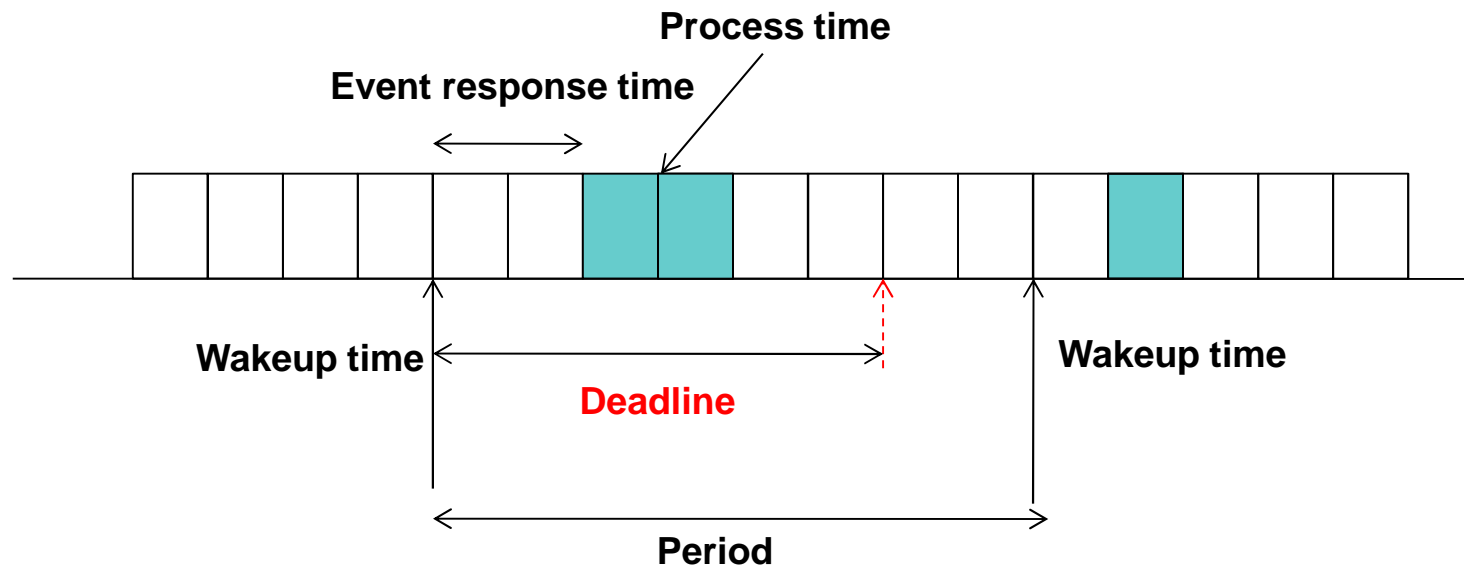
- We would like to use Linux on control systems
- Real-time is one of the most critical topic
- Problem statement
 - Need to evaluate deeply to meet the deadline
 - CPU resource used too much by higher priority tasks



EDF scheduler

Definition of deadline

- **Wakeup time:** The time of an event occurred (Ex. Timer interrupt) and target task's state changed to runnable.
- **Event response time:** Interrupt latency
- **Deadline:** The time for a target task must finish

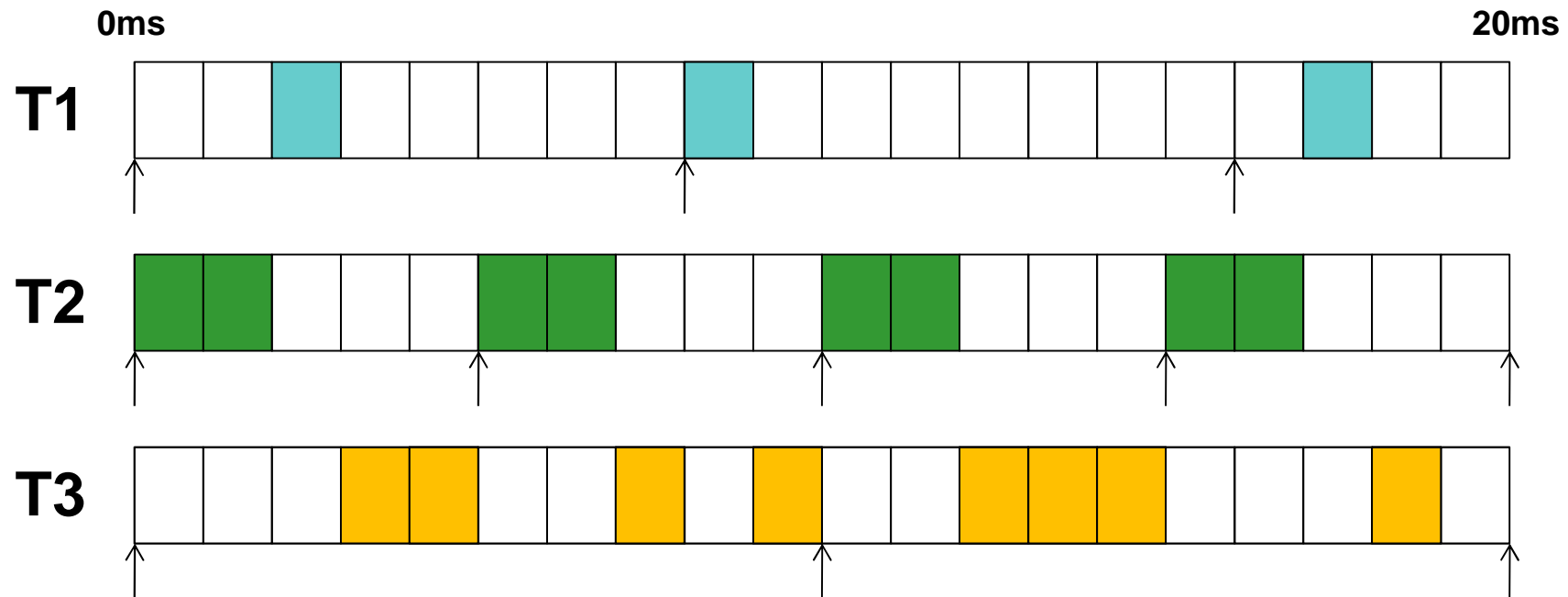


Earliest Deadline First scheduling (EDF)

- The earliest deadline task has the highest priority
- Task's priority is dynamically changed and managed
 - SCHED_FIFO is static priority management
- Theoretically the total CPU usage by all tasks is up to 100%
 - Includes the kernel overhead
 - If usage of CPU by all tasks is less than 100%, all tasks meet the deadline
- Reference
 - http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

An example of EDF Scheduling

- Task1: budget 1ms period 8ms
 - Task2: budget 2ms period 5ms
 - Task3: budget 4ms period 10ms
- CPU usage = 0.925% < 100%



Rate-Monotonic Scheduling (RMS)

- One of the popular scheduling algorithm for RTOS

- Assumptions for task behavior

- NO resource sharing such as hardware, a queue, or any kind of semaphore
- Deterministic deadlines are exactly equal to periods
- Static priorities (the task with the highest static priority that is runnable immediately preempts all other tasks)
- Static priorities assigned according to the rate monotonic conventions (tasks with shorter periods/deadlines are given higher priorities)
- Context switch times and other thread operations are free and have no impact on the model

- CPU utilization

- n: number of periodic tasks, T_i : Release period, C_i : Computation time

$$U = \sum_{i=0}^n C_i / T_i \leq n(\sqrt[n]{2} - 1) \xrightarrow{n=\infty} \ln 2 \approx 0.69$$

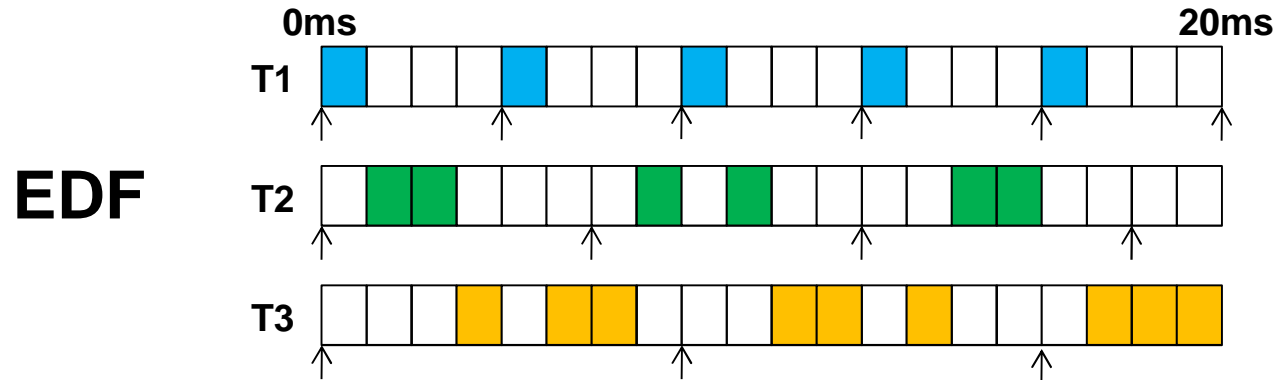
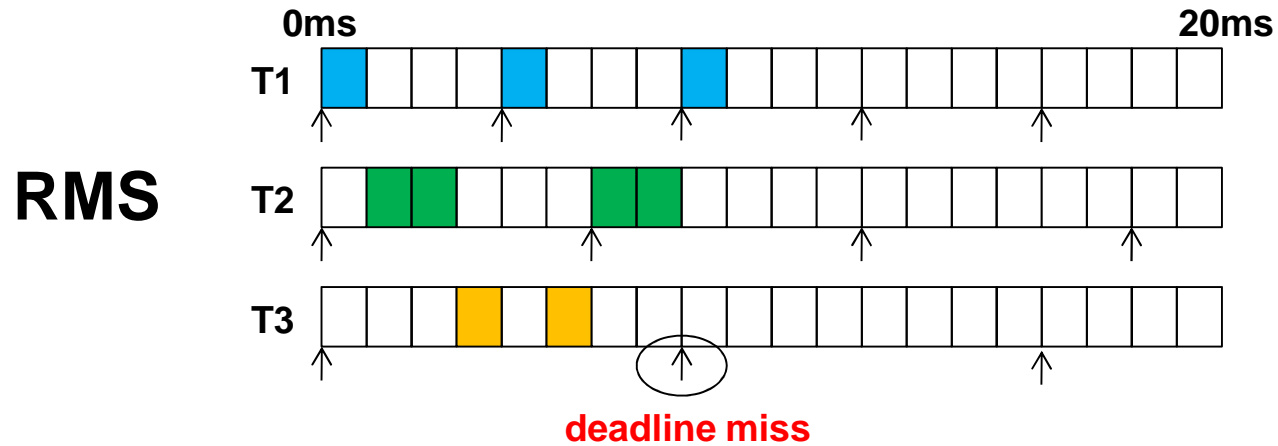
- CPU utilization depends on the combination of periodic tasks and it is possible to meet the deadline even the CPU utilization is around 80%

- Reference

- http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

Compared with the RMS scheduling

- Task1: budget 1ms period 4ms
 - Task2: budget 2ms period 6ms
 - Task3: budget 3ms period 8ms
- CPU usage=0.958%



Comparison of deadline algorithms

	Advantage	Disadvantage
RMS	Easier to implement	Evaluation for scheduling possibility is required to meet the deadline
EDF	No evaluation for scheduling possibility is required to meet the deadline	Difficult to implement

What SCHED_DEADLINE is?

- http://www.evidence.eu.com/sched_deadline.html

- Implements the EDF scheduling algorithm
- Posted to LKML by Dario Faggioli and Juri Lelli
- Latest version is V7 (2013/2/12)
 - Our work based on V2 and V4.

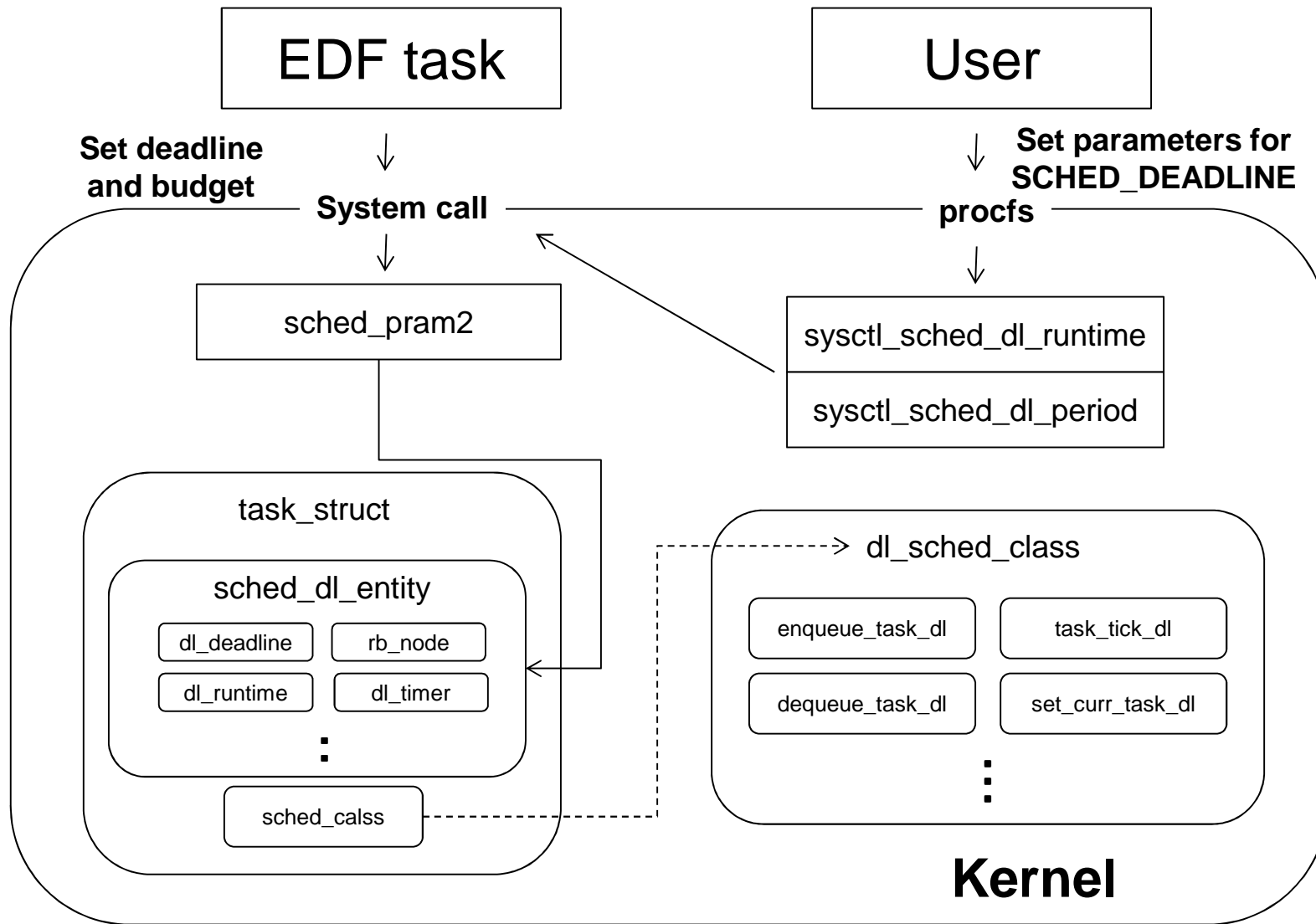
- **Key features of SCHED_DEADLINE**

- Temporal isolation
 - The temporal behavior of each task (i.e., its ability to meet its deadlines) is not affected by the behavior of any other task in the system
 - Each task is characterized by the following aspects:
 - Budget: `sched_runtime`
 - Period: `sched_period`, equal to its deadline

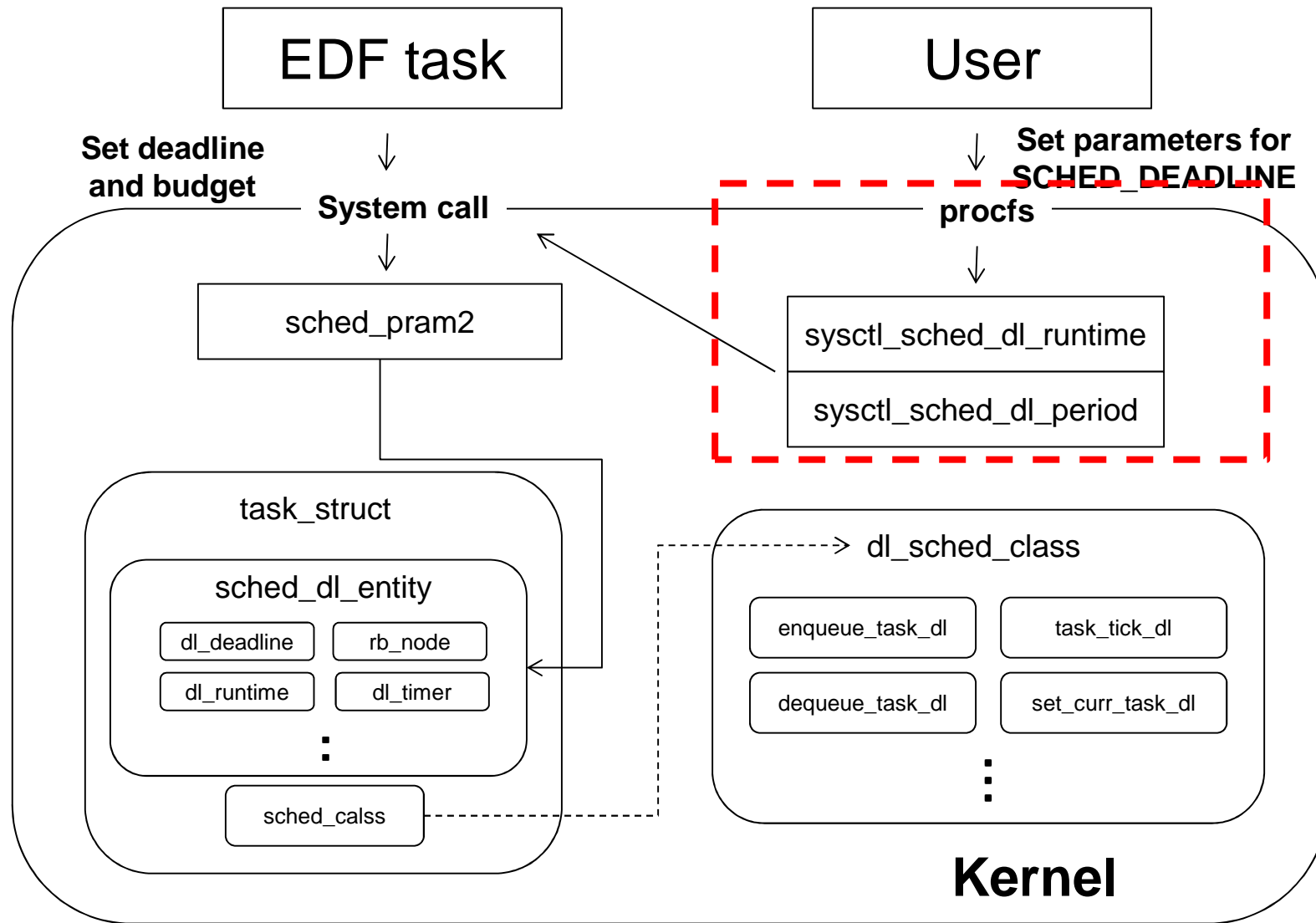
Build SCHED_DEADLINE enabled kernel

- **Get the source code from the following places**
 - `git clone git://github.com/jlelli/sched-deadline.git`
(for V4 to V7)
 - `git clone git://gitorious.org/rt-deadline`
(for V2)
- **Kernel configuration**
 - `CONFIG_EXPERIMENTAL = y`
 - `CONFIG_CGROUPS = y`
 - `CONFIG_CGROUP_SCHED = n`
 - `CONFIG_HIGH_RES_TIMERS = y`
 - `CONFIG_PREEMPT = y`
 - `CONFIG_PREEMPT_RT = y`
 - `CONFIG_HZ_1000 = y`

Overview of SCHED_DEADLINE



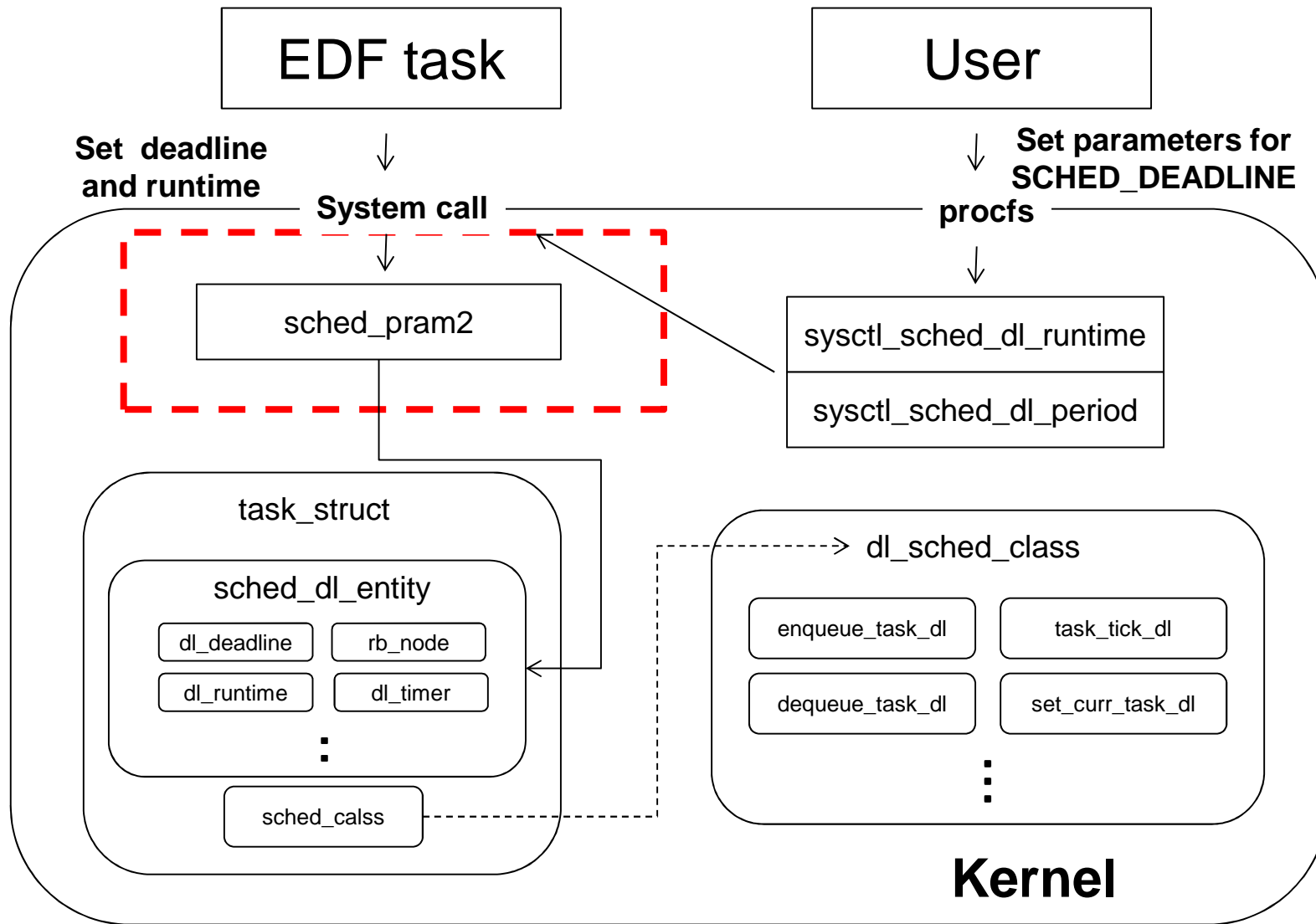
Overview of SCHED_DEADLINE



Setting CPU utilization for EDF tasks

- **Parameters can be setted via procfs**
 - CPU utilization for rt(SCHED_FIFO and SCHED_RR) and dl(SCHED_DEADLINE) should be under 100%
 - Parameters for EDF scheduler
 - /proc/sys/kernel/sched_dl_period_us
 - /proc/sys/kernel/sched_dl_runtime_us
- **When a task requires more than above limit, the task cannot submit to run**
- **An example setting (rt: 50%, dl:50%)**
 - # echo 500000 > /proc/sys/kernel/sched_rt_runtime_us (500ms)
 - # echo 100000 > /proc/sys/kernel/sched_dl_period_us (100ms)
 - # echo 50000 > /proc/sys/kernel/sched_dl_runtime_us (50ms)

Overview of SCHED_DEADLINE



Run a EDF task

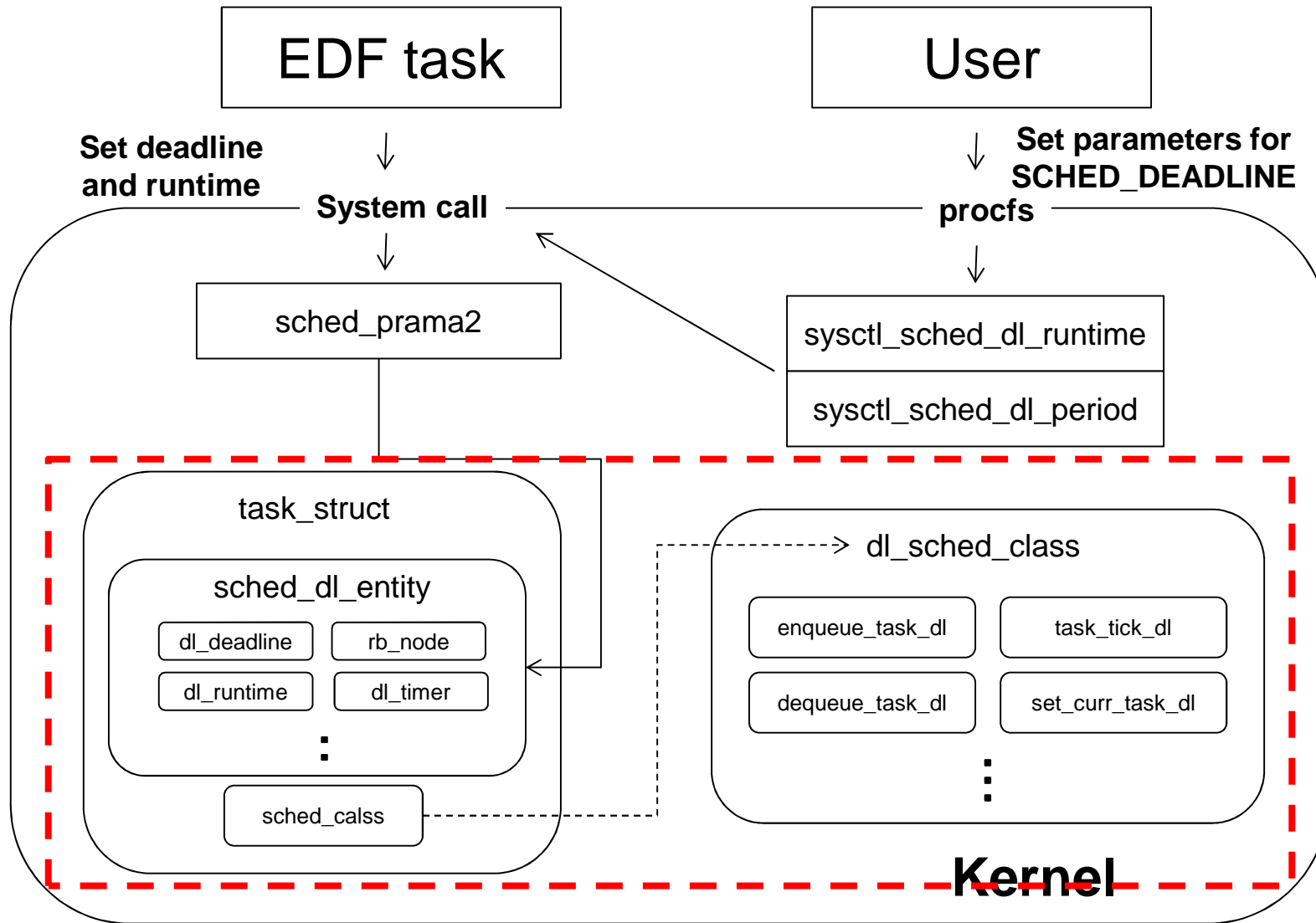
■ Schedtool

- # schedtool -E -t 10000:100000 -a 0 -e ./yes
- Options
 - E: a task runs on SCHED_DEADLINE
 - t: <execution time> and <period> in micro seconds
 - a: Affinity mask
 - e: command

■ System call

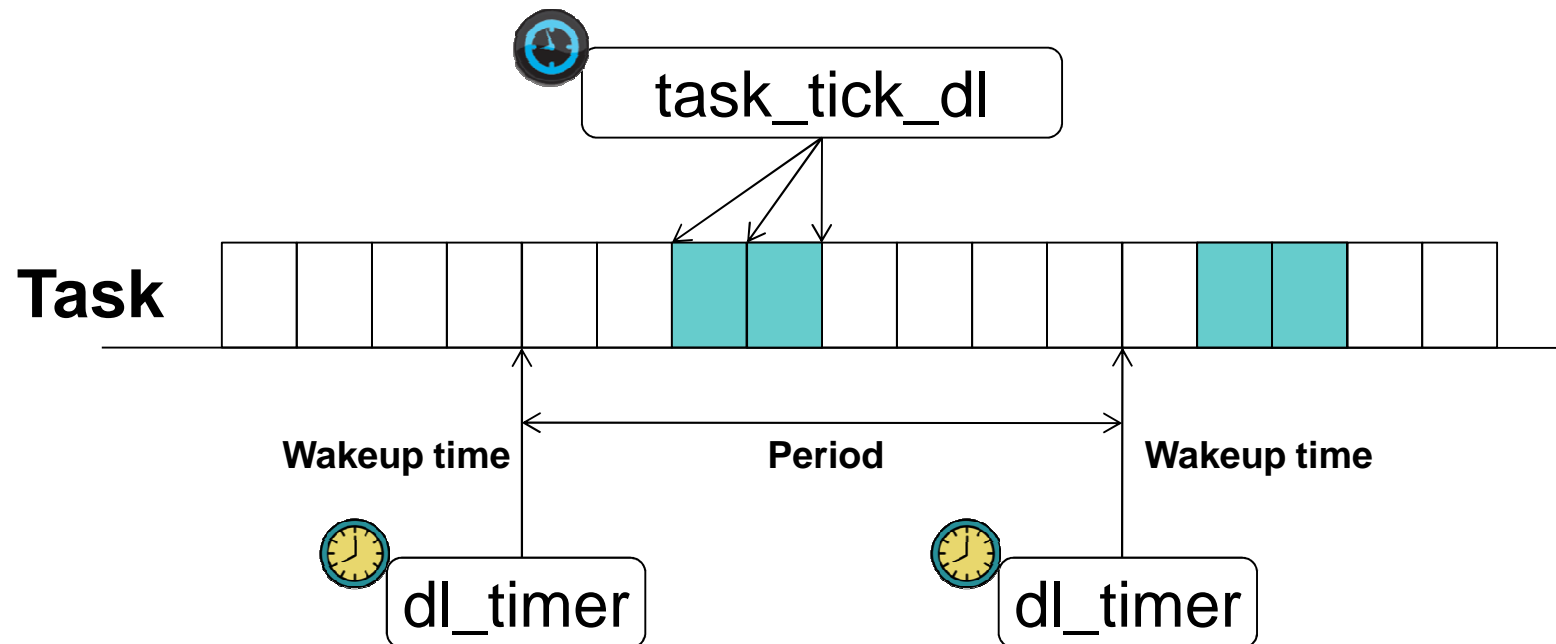
- sched_setscheduler2()

Budget management for EDF tasks

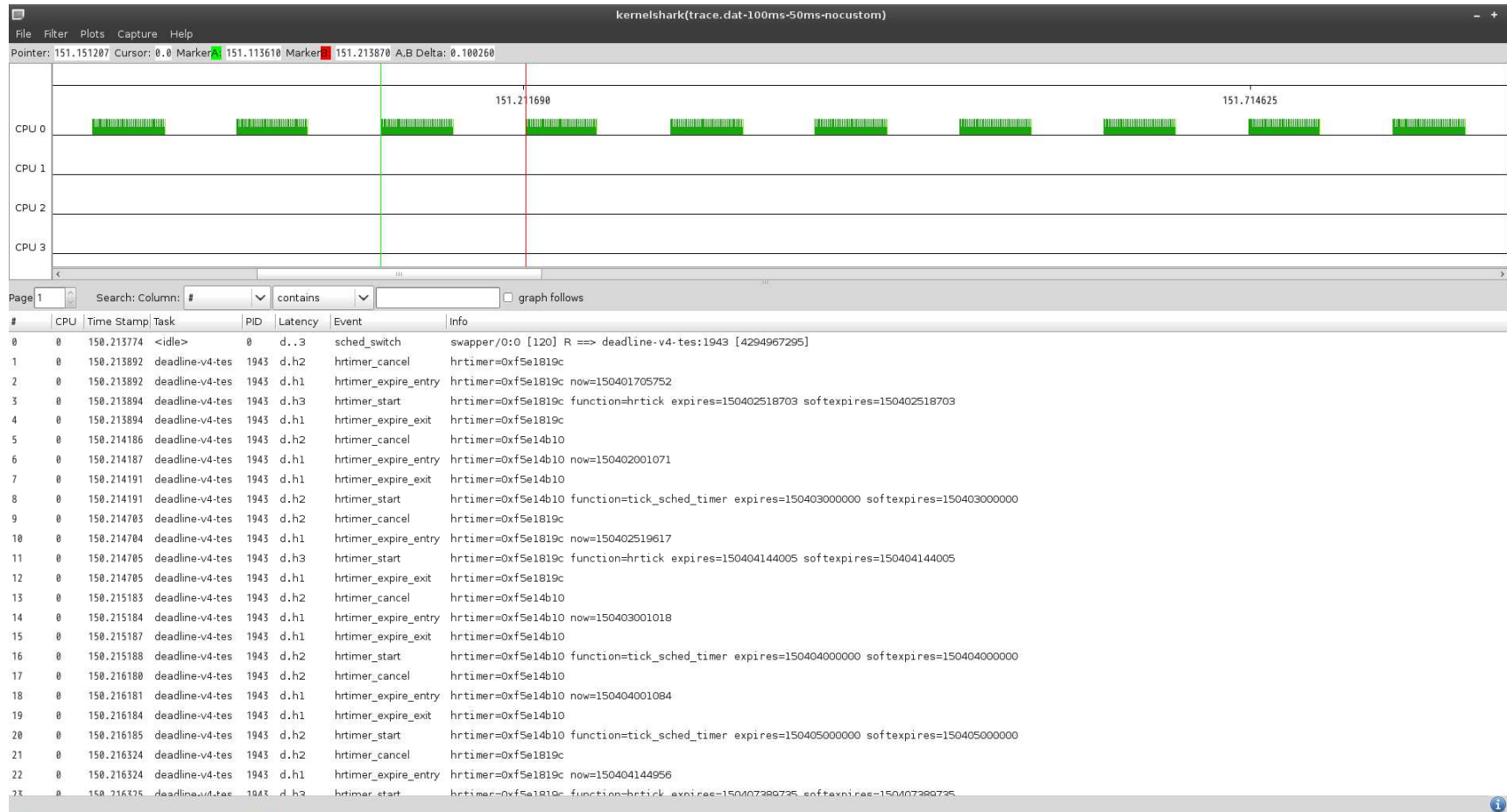


Budget management for EDF tasks

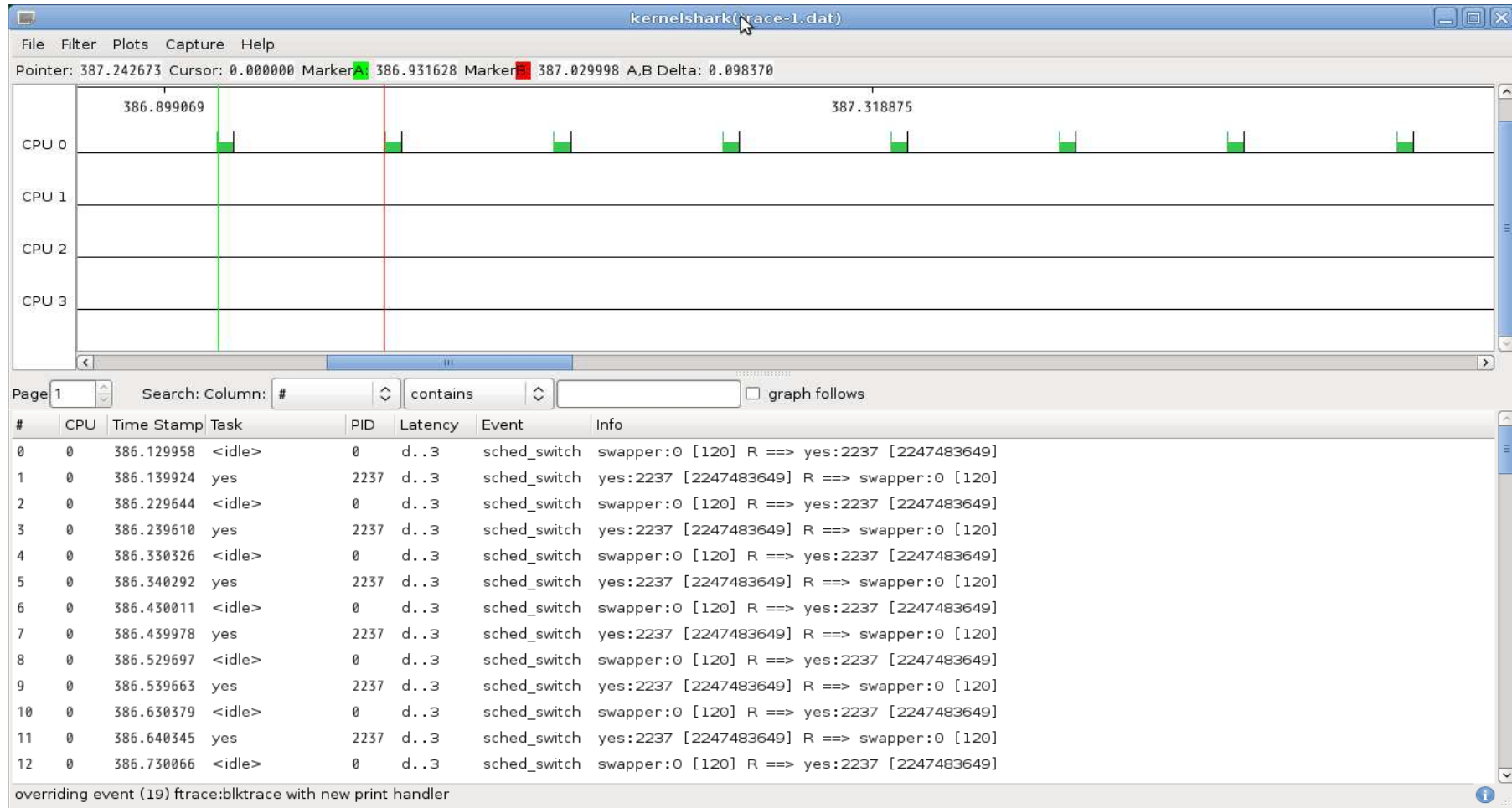
- Each task on **SCHED_DEADLINE** has budget which allows it to use CPU
- **Budget management**
 - Refill budget : `dl_timer` (high resolution timer)
 - Use budget : `task_tick_dl` (tick based)



Evaluation (Period:100ms, Budget: 50ms)

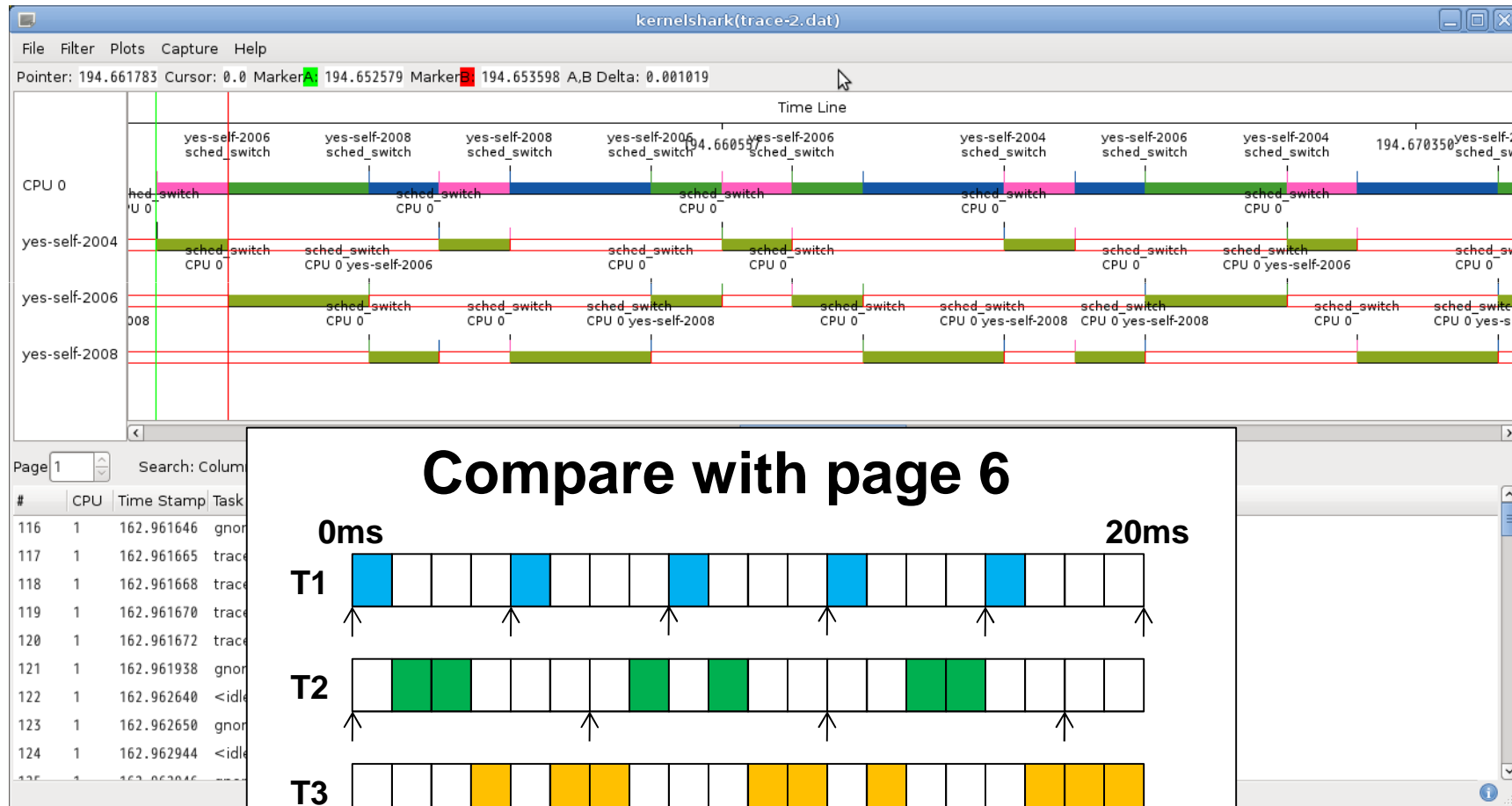


Evaluation (Period: 100ms , Budget: 10ms)

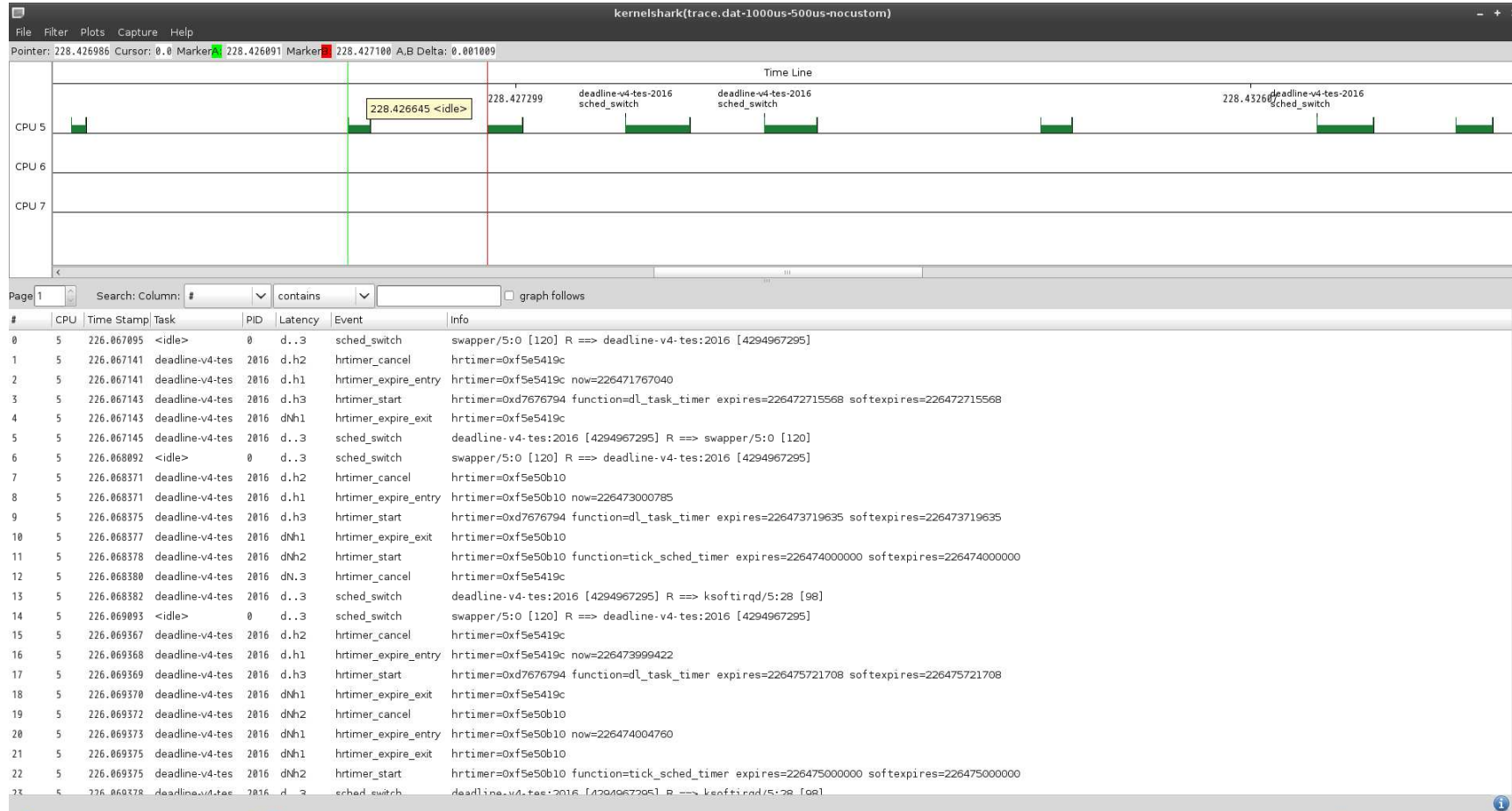


Evaluation

- Task T1: budget 1ms period 4ms
- Task T2: budget 2ms period 6ms
- Task T3: budget 3ms period 8ms

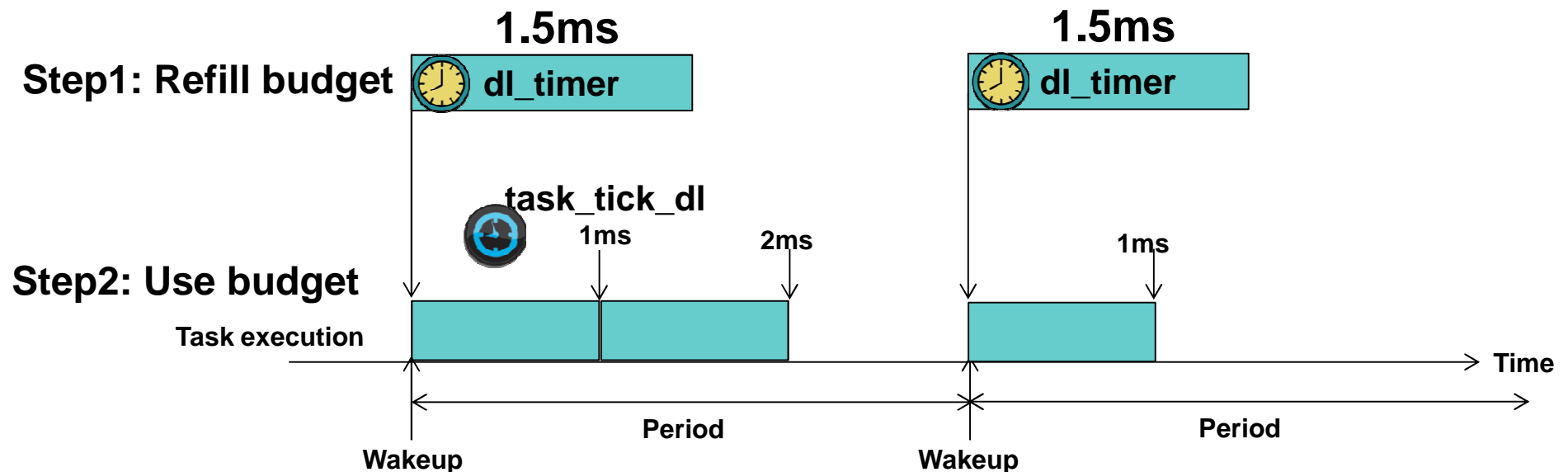


Evaluation (Period: 1ms, Budget: 0.5ms)



Budget management for EDF tasks

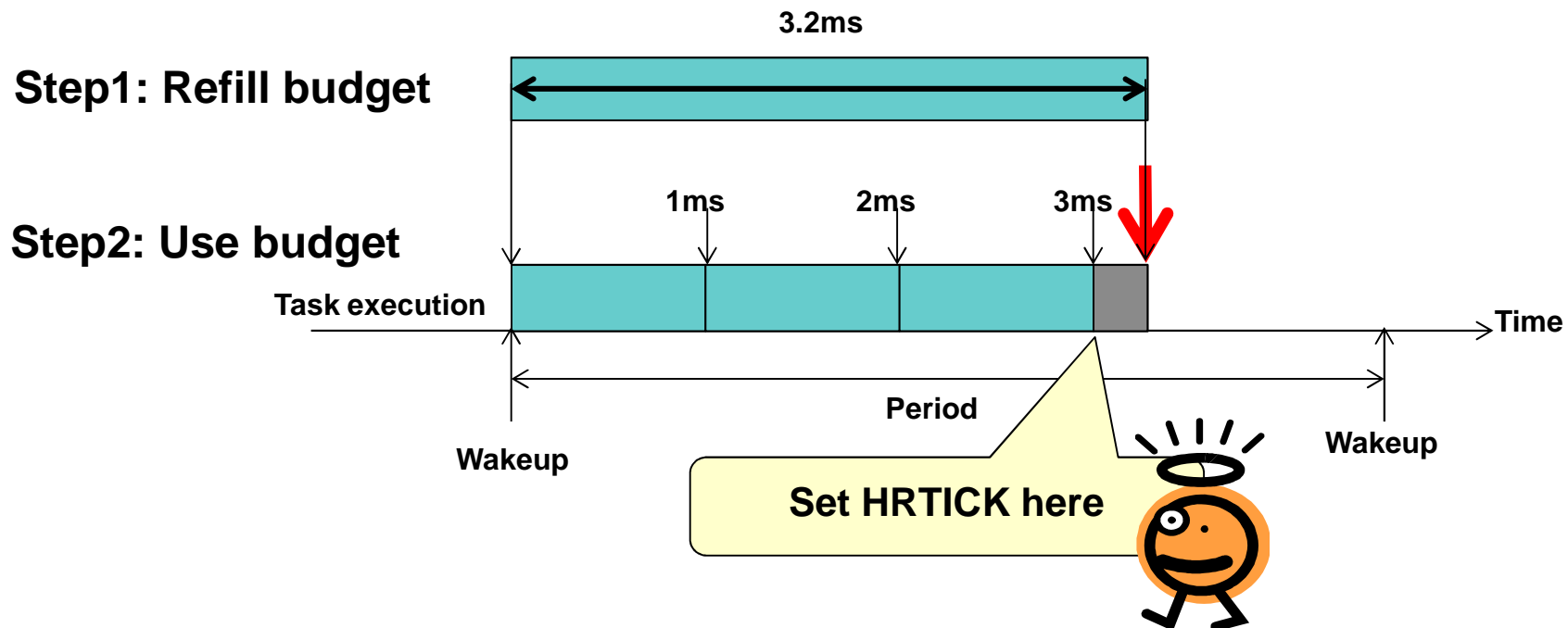
- Each task on `SCHED_DEADLINE` has budget which allows it to use CPU
- Budget management
 - Refill budget : `dl_timer` (high resolution timer)
 - Use budget : `task_tick_dl` (tick based)
- An Issue
 - Difficult to keep task's budget if the budget has micro seconds granularity



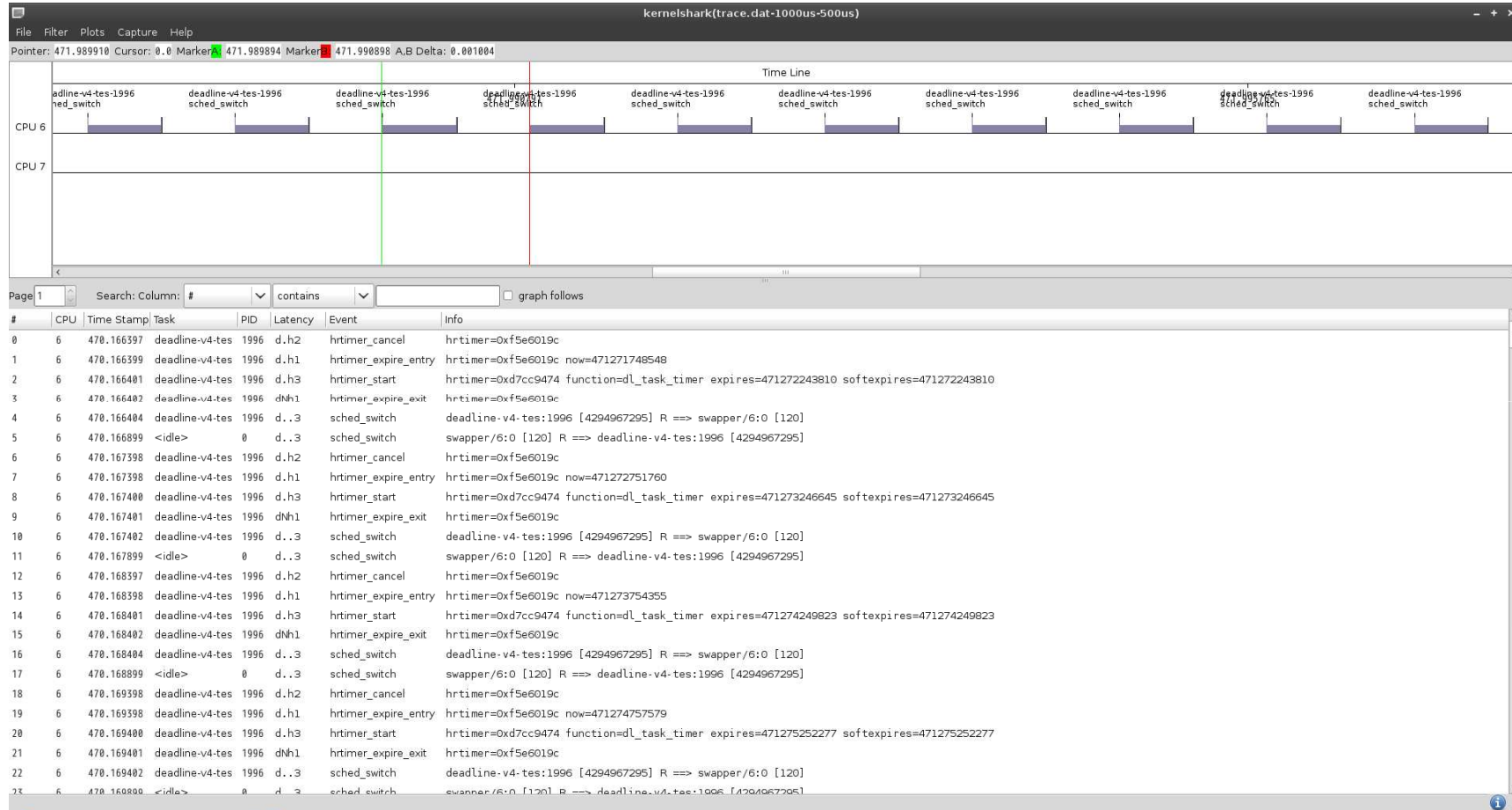
Support for micro seconds granularity

■ Overview

- When a task's budget is less than 1ms, set HRTICK for the rest of budget



Evaluation (Cycle: 1ms, Budget: 0.5ms)



Advantage and Disadvantage

- **Advantage**

- Easy to support high resolution budget

- **Disadvantage**

- Increase overhead

Summary of scheduling granularity improvement

- **An Enhancement for budget management**
 - Support fine grained budget such as 100 micro seconds
 - HRTICK is needed to support fine grained budget

DEMO

- “Big Buck Bunny”

- <http://www.bigbuckbunny.org/>
- (c) copyright 2008, Blender Foundation
- License: Creative Commons Attribution 3.0



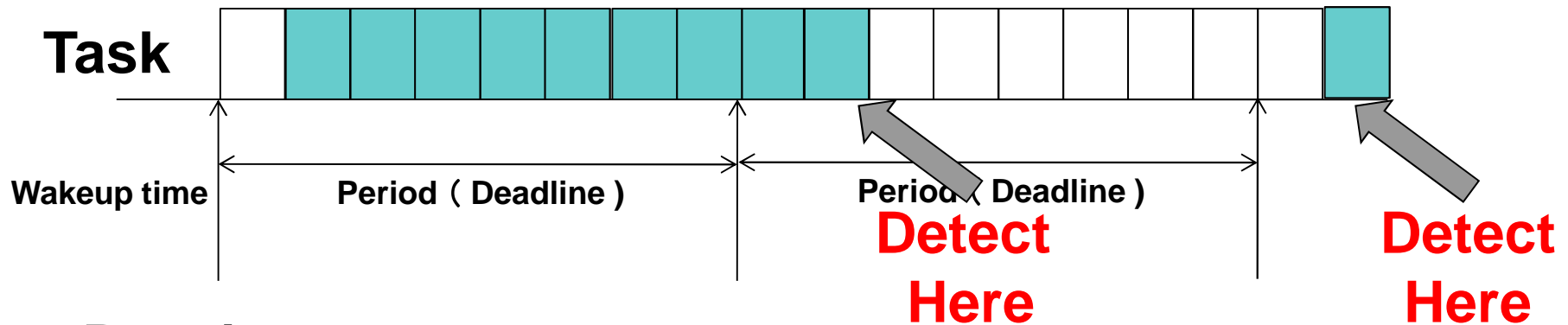
- **Play the movie about 3 times faster than normal speed**

- Requirement for a frame decoding
 - Period: 10ms
 - CPU budget: 3.6ms
- When a frame decode is not finished in a period, write “DEADLINE MISS” on screen and add penalty just a bit (3ms)

Requirement for deadline miss detection

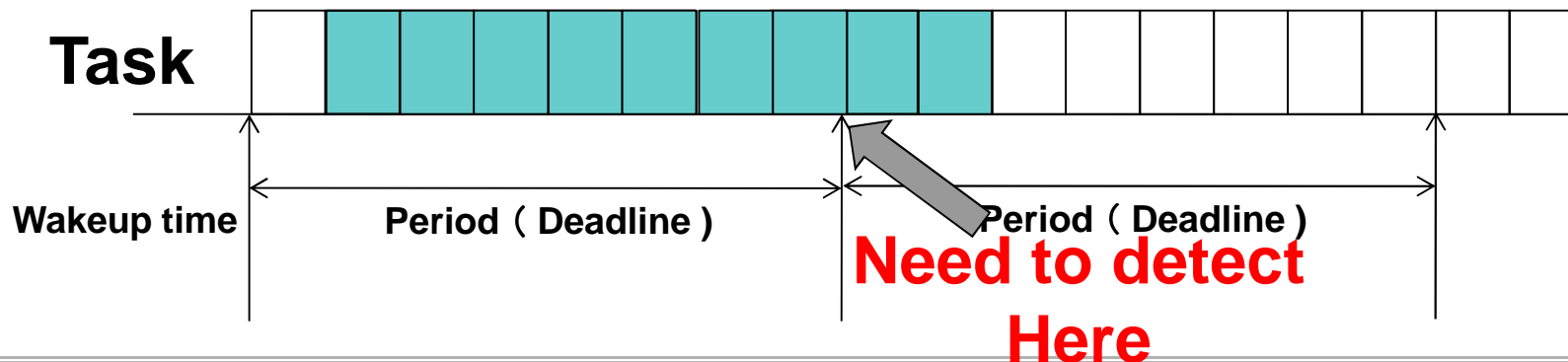
■ Current behavior

- Have a counter to count the number of deadline misses
- Not able to control tasks that missed the deadline



■ Requirement

- Detect the deadline miss at the beginning of period



Current limitation of SCHED_DEADLINE

- **SCHED_DEADLINE is able to count the number of deadline misses**
- **SCHED_DEADLINE is not able to control tasks that missed the deadline**

Possible approaches to detect deadline misses

■ Userland

- Monitor process or thread
 - Run at the end or beginning of each period
- A pair of periodic timer and signal handler
 - Have to cancel if the process meet the deadline
 - or
 - Have to catch the signal to check the deadline miss status
- ...

■ Kernel

- This presentation takes this approach

Overview of deadline miss detection

- Set parameters to `sched_param2.sched_flags`
- Parameters for `sched_flags`
 - Enable or disable the deadline miss detection
 - `SCHED_DL_DMISS_DETECT_DISABLE`
 - `SCHED_DL_DMISS_DETECT_ENABLE`
 - Behaviors
 - next slide..

Behaviors (sched_param2.sched_flags)

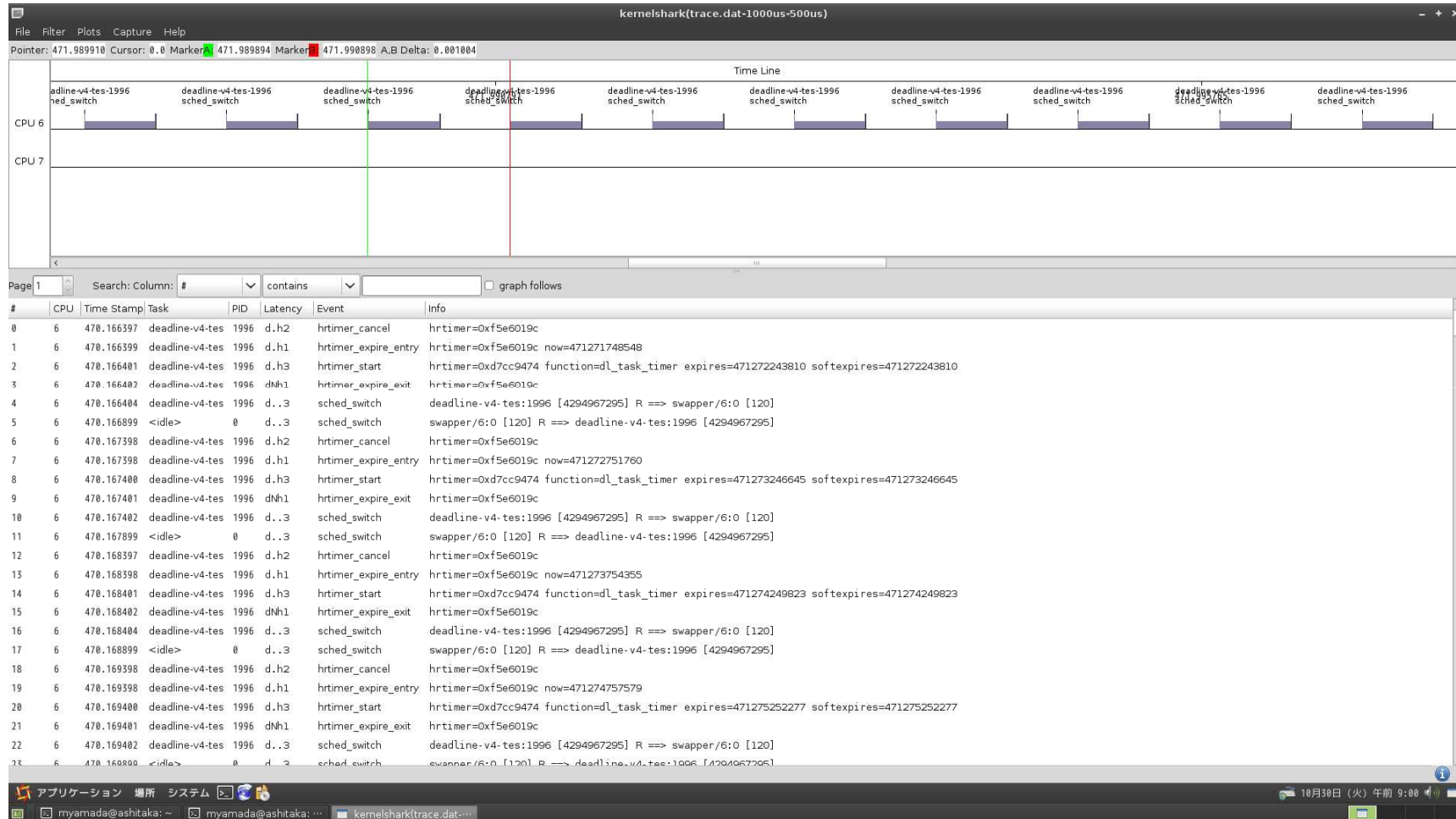
■ Nothing to do

- SCHED_DL_DMISS_ACT_NONE
- When the SCHED_DL_DMISS_DETECT_ENABLE flag is set, the kernel will count the number of deadline misses on /proc/sys/kernel/sched_dl_dmiss_count

■ Send a signal

- To the process itself
 - SCHED_DL_DMISS_ACT_SEND_SIG_OWN
 - Send a SIGXCPU signal
 - SCHED_DL_DMISS_ACT_PROCESS_END
 - Send a SIGKILL signal
 - SCHED_DL_DMISS_ACT_PROCESS_STOP
 - Send a SIGSTOP signal
- To the other task
 - Set the PID to /proc/sys/kernel/sched_dl_dmiss_sig_pid
 - SCHED_DL_DMISS_ACT_SEND_SIG_OTHER_WITH_RUN
 - Send a signal to the specified process
 - Deadline missed process continues to run
 - SCHED_DL_DMISS_ACT_SEND_SIG_OTHER_WITH_STOP
 - Send a signal to the specified process
 - Deadline missed process is stoped

DEMO2



Conclusion

- **SCHED_DEADLINE is useful for real time systems**
- **This presentation enhanced SCHED_DEADLINE**
 - For budget management
 - Support fine grained budget such as 100 micro seconds
 - HRTICK is needed to support fine grained budget
 - For deadline miss detection
 - Add a function to control deadline missed tasks
- **All source code are available at the following URL**
 - <https://github.com/ystk/sched-deadline/tree/dlmiss-detection-dev>

Thank you