Adaptive Scheduling Parameters Manager for SCHED_DEADLINE — RELEASE CANDIDATE —

Alessio Balsini
`a.balsini@sssup.it`

Università di Pisa, Scuola Superiore Sant'Anna

Workshop on Real-Time Scheduling in the Linux Kernel
27 June 2014

Scheduling Soft Real-Time Periodic Tasks - What Happens

Computational request at each activation may heavily differ.

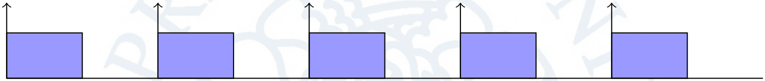


Figure : Desired

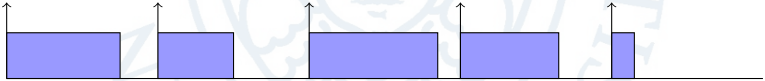


Figure : Realistic

Scheduling Soft Real-Time Periodic Tasks - Benchmarks

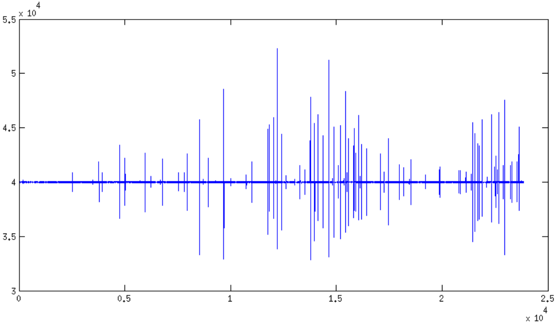


Figure : Back to the Future (MKV)

Scheduling Soft Real-Time Periodic Tasks - Benchmarks

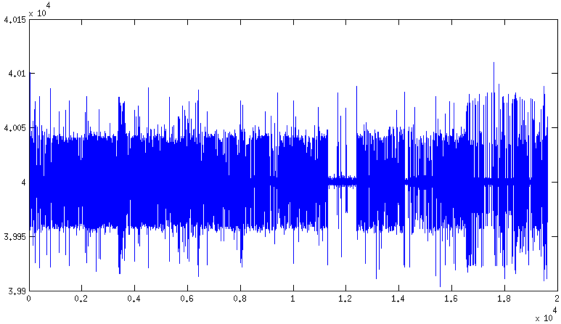


Figure : Blade Runner (AVI)

Scheduling Soft Real-Time Periodic Tasks - Benchmarks

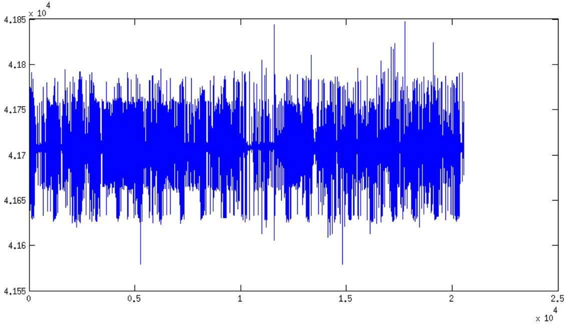


Figure : Superman Returns (MP4)

Scheduling Soft Real-Time Periodic Tasks - Which Parameter Really Matters?

In SCHED_DEADLINE it is possible to choose

- Period
- Relative Deadline
- Bandwidth

But in the just seen examples, a single parameter is enough

Response Time

Scheduling Soft Real-Time Periodic Tasks - Which Parameter Really Matters?

In SCHED_DEADLINE it is possible to choose

- Period
- Relative Deadline
- Bandwidth

But in the just seen examples, a single parameter is enough

Response Time

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth
   - [illegible]

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth
   - [illegible]

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth
   - Hm...

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth
   - Hm...

Response Time to SCHED_DEADLINE: $\mathbb{R} \rightarrow \mathbb{R}^3$

How to generate SCHED_DEADLINE parameters starting from the Response Time?

1. Period
   - Euqal to the Response Time
2. Relative Deadline
   - Euqual to the Period
3. Bandwidth
   - Hm...

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic	Pessimistic
1 Low QoS	Best QoS
2 Resources saving	Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Is the glass half empty or half full?

Warning: choosing the bandwidth may be the cause of several headaches

✌

Optimistic

1. Low QoS
2. Resources-driven?

Pessimistic

1. Best QoS
2. Waste of resources

Another possible approach? Dynamic! - Let's see how much you drink

The bandwidth is dynamically (periodically) chosen *for each SCHED_DEADLINE task* depending on the history of the required computational times.
So, it's basically a feedback loop controller.

But if this controller modifies the bandwidth, isn't it just like removing the CBS to SCHED_DEADLINE?

Yes and no.

1. Adaptation delay for the transitory
2. Global controller for the fairness

Another possible approach? Dynamic! - Let's see how much you drink

The bandwidth is dynamically (periodically) chosen *for each SCHED_DEADLINE task* depending on the history of the required computational times.
So, it's basically a feedback loop controller.

But if this controller modifies the bandwidth, isn't it just like removing the CBS to SCHED_DEADLINE?

Yes and no.

- Adaptation delay: for the transitory
- Global controller: for the fairness

Another possible approach? Dynamic! - Let's see how much you drink

The bandwidth is dynamically (periodically) chosen *for each SCHED_DEADLINE task* depending on the history of the required computational times.
So, it's basically a feedback loop controller.

But if this controller modifies the bandwidth, isn't it just like removing the CBS to SCHED_DEADLINE?

Yes and no.

- Adaptation delay: for the transitory
- Global controller: for the fairness

Another possible approach? Dynamic! - Let's see how much you drink

The bandwidth is dynamically (periodically) chosen *for each SCHED_DEADLINE task* depending on the history of the required computational times.
So, it's basically a feedback loop controller.

But if this controller modifies the bandwidth, isn't it just like removing the CBS to SCHED_DEADLINE?

Yes and no.

1. Adaptation delay: for the transitory
2. Global controller: for the fairness

Another possible approach? Dynamic! - Let's see how much you drink

The bandwidth is dynamically (periodically) chosen *for each SCHED_DEADLINE task* depending on the history of the required computational times.
So, it's basically a feedback loop controller.

But if this controller modifies the bandwidth, isn't it just like removing the CBS to SCHED_DEADLINE?

Yes and no.

1. Adaptation delay: for the transitory
2. Global controller: for the fairness

Tools

Hi-level Point of View

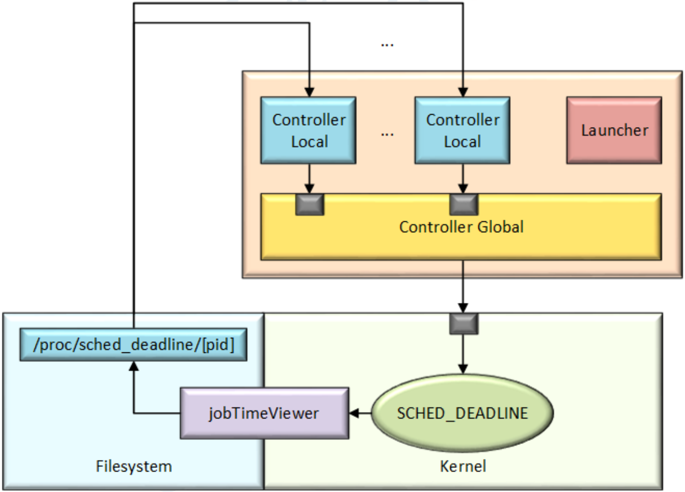


Figure : Overall Block Scheme

Kernel Module: SCHED_DEADLINE Spy - Userspace

This module creates a file for each SCHED_DEADLINE task

/proc/sched_deadline/[PID]

Containing something like

```
1401468028 22757242 10149132 N
1401468028 22757242 86353 N
1401468028 62757243 37679835 Y
1401468028 94757243 26311134 N
...
```

- first two rows represent the kernel time (seconds and nanoseconds) at which the measurement was taken
- third row represents the read job execution time
- last row nitifies if there was a deadline miss

Kernel Module: SCHED_DEADLINE Spy - Implementation Hints

Kernel probes: *Kprobe*s
The module places probes around the kernel and the instrumentation codes are executed when the processor encounters those probe point.
This module attaches probes to

enqueue_task_dl

and

update_dl_entity

With *Jprobe*s it is also possible to access function arguments, used to create tasks' statistics.
Other several callbacks are provided, managing all the statistics sequential files.

Daemon: SCHED_DEADLINE Dynamic Manager - Interface

This tool provides the DBus interface

core.sched.dl.ProcessManager

with the following methods:

- xml: requires a string input, corresponding to the path of the XML file containing the task information
- fixed_add: adds a new fixed task to the control list, with the defined SCHED_DEADLINE parameters
- fixed_launch: creates a new fixed task and adds it to the control list, with the defined SCHED_DEADLINE parameters
- control: adds a new dynamic task to the control list, with the defined response time parameter
- launch: creates a new dynamic task and adds it to the control list, with the defined response time parameter

Daemon: SCHED_DEADLINE Dynamic Manager - Interface, XML File Example

```
<?xml version=”1.0”?>
<SchedulingAlgorithm name=”SCHED_DEADLINE”>
  <path>/usr/bin/executable</path>
  <args>-p parameter</args>
  <runtime>28000000</runtime>
  <deadline>33333333</deadline>
  <period>33333333</period>
</SchedulingAlgorithm>
```

Daemon: SCHED_DEADLINE Dynamic Manager - Implementation, Local Controllers

One local controller for each dynamic scheduled task.
It performs the following operations cyclically

- obtains task statistics
- runs the Control Algorithm to calculate the best utilization factor.
  Current Control Algorithm Implementation: uses the worst case of a window of samples
- sends the new utilization factor to the global controller

Daemon: SCHED_DEADLINE Dynamic Manager - Implementation, Global Controller

It performs the following operations cyclically

- checks the schedulability of all the SCHED_DEADLINE utilization factors

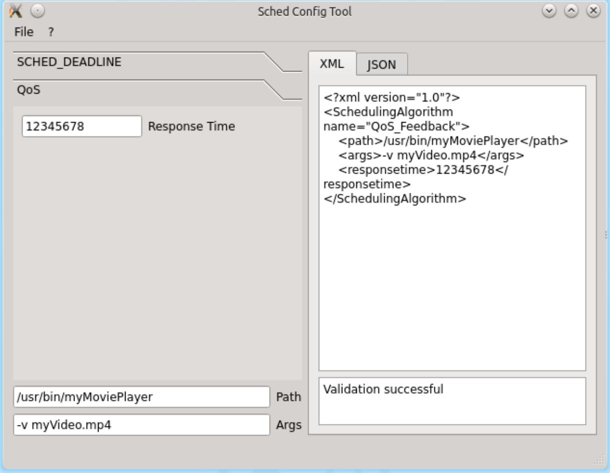
$$\sum_{i=1}^{n} \frac{B_{D,i}}{T_{D,i}} + \sum_{i=1}^{m} \frac{B_{F,i}}{T_{F,i}} \leq B_{SD}$$

  - if not verified, uses the *Spring With no Length Constraints* algorithm to compress the dynamic tasks' requirements

$$\forall i,\ U'_{D,i} = \frac{B_{D,i}}{T_{D,i}} - (U_D - B_{residual}) \cdot \frac{T_{D,i}}{\sum_{i=1}^{n} T_{D,i}}$$

- updates SCHED_DEADLINE parameters

Configuration Generator: SchedConfigTool



Response Time: 12345678

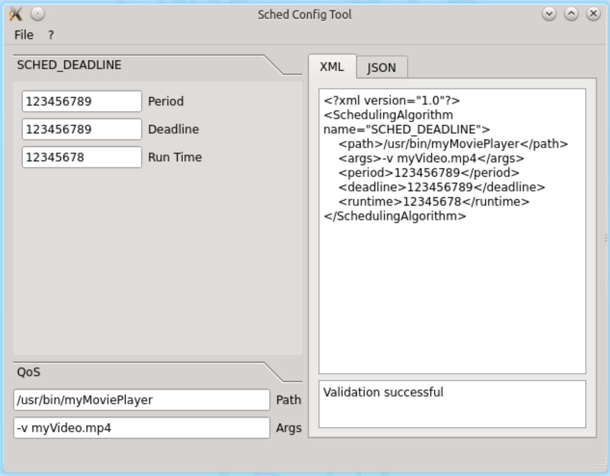
Path: /usr/bin/myMoviePlayer

Args: -v myVideo.mp4

```
<?xml version="1.0"?>
<SchedulingAlgorithm
name="QoS_Feedback">
   <path>/usr/bin/myMoviePlayer</path>
   <args>-v myVideo.mp4</args>
   <responsetime>12345678</
responsetime>
</SchedulingAlgorithm>
```

```
Validation successful
```

Configuration Generator: SchedConfigTool



SCHED_DEADLINE

123456789 Period

123456789 Deadline

12345678 Run Time

QoS

/usr/bin/myMoviePlayer Path

-v myVideo.mp4 Args

```
<?xml version="1.0"?>
<SchedulingAlgorithm
name="SCHED_DEADLINE">
    <path>/usr/bin/myMoviePlayer</path>
    <args>-v myVideo.mp4</args>
    <period>123456789</period>
    <deadline>123456789</deadline>
    <runtime>12345678</runtime>
</SchedulingAlgorithm>
```

Validation successful

Performances

Response Times - MPlayer Without SCHED_DEADLINE

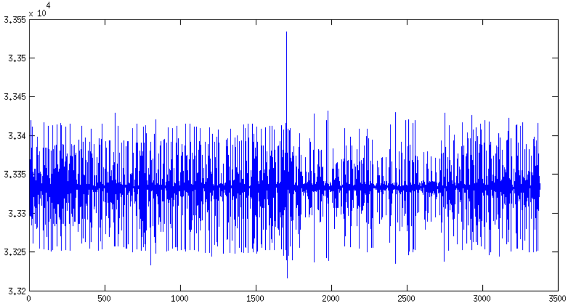


Figure : Eve Online Rubicon (MP4) without SCHED_DEADLINE

Response Times
- MPlayer With Dynamic Manager, Alone

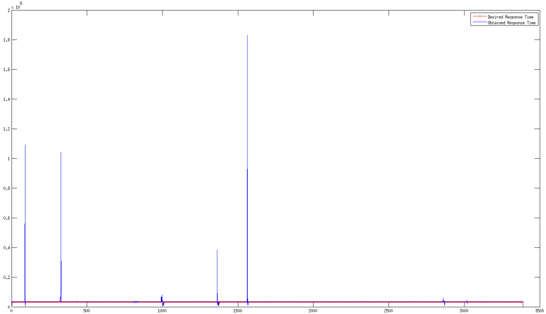


Figure : Eve Online Rubicon (MP4) Dynamic Manager, Alone

Response Times
- MPlayer With Dynamic Manager, With Fixed

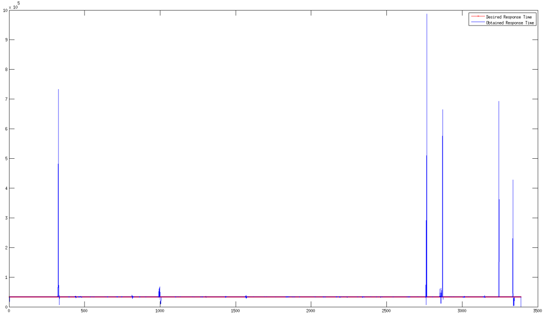


Figure : Eve Online Rubicon (MP4) Dynamic Manager, Running Together With Several Fixed Parameters SCHED_DEADLINE Tasks

Response Times
- MPlayer With Dynamic Manager, With Other

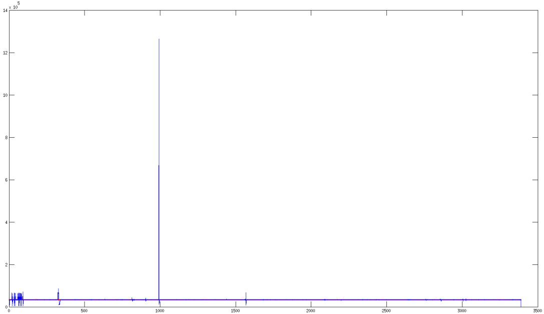


Figure : Eve Online Rubicon (MP4) Dynamic Manager, Running Together With Several Other Linux Tasks

Practical Session

Repositories

- SCHED_DEADLINE Spy
  - github.com/balsini/sched-deadline-spy
- SCHED_DEADLINE Dynamic Manager
  - github.com/balsini/sched-deadline-dynamic-manager
- SchedConfigTool
  - github.com/balsini/SchedConfigTool

References

- L. Palopoli, T. Cucinotta, L. Marzario, G. Lipari, *AQoSA - Adaptive Quality of Service Architecture*. Wiley InterScience, 2008.
- G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications 3rd*. Springer Publishing Company, 2011.