

Real-Time Deadline Scheduling and More...

Dario Faggioli

ReTiS Lab, Scuola Superiore Sant'Anna - Pisa (Italy),
10th Annual Linux KERNEL SUMMIT, Cambridge, MA

1st, November 2010

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic** result"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

"Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic result**"

What is Real-Time?

Real-Time is all about determinism and guarantees:

- guarantee that your system is deterministic,
- achieve determinism, so that you can provide guarantees.

Classic Real-Time theory says:

”Hey, we can do that, just give us the following:

- knowledge of **how many** and **what** task there will be,
- knowledge of their **periodicity** and computation **times**,
- either tasks are **independent** ...
- ... or you must know **what**, for **how long** and **how many** times they access shared resources,
- use **uniprocessors** ...
- ... or must run a schedulability test taking **minutes** ... For a **pessimistic** result”

Resource Reservations

Tasks are reserved Q (*runtime* or *budget*) time units every T time units (*period*) (i.e., they have a "CPU bandwidth" of $\frac{Q}{T}$).

Resource Reservations

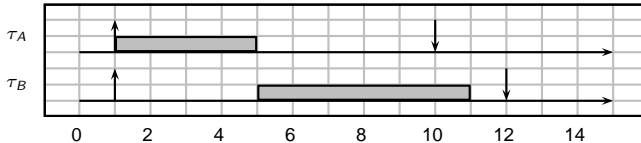
Tasks are reserved Q (*runtime* or *budget*) time units every T time units (*period*) (i.e., they have a "CPU bandwidth" of $\frac{Q}{P}$).

Applications get guaranteed CPU time, and misbehaviors are confined within each task's "bandwidth capsule".

Resource Reservations

Tasks are reserved Q (*runtime* or *budget*) time units every T time units (*period*) (i.e., they have a "CPU bandwidth" of $\frac{Q}{T}$).

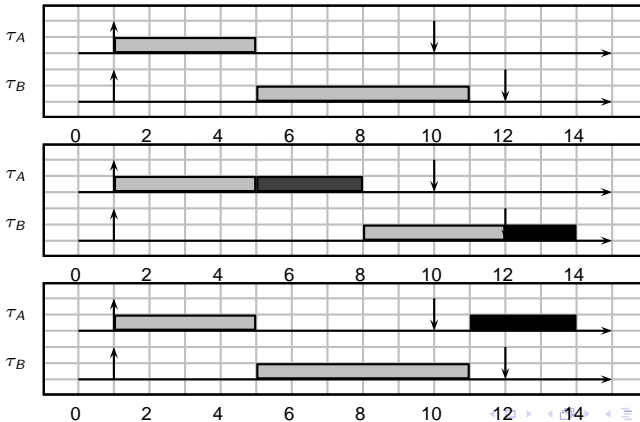
Applications get guaranteed CPU time, and misbehaviors are confined within each task's "bandwidth capsule".



Resource Reservations

Tasks are reserved Q (*runtime* or *budget*) time units every T time units (*period*) (i.e., they have a "CPU bandwidth" of $\frac{Q}{P}$).

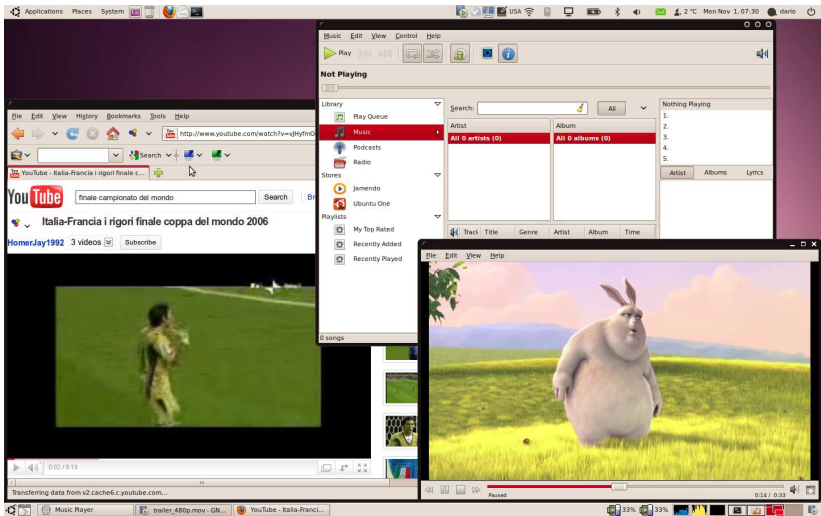
Applications get guaranteed CPU time, and misbehaviors are confined within each task's "bandwidth capsule".



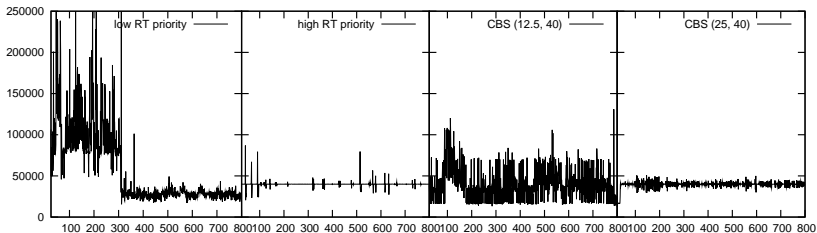
What Applications are Real-Time? (I)



What Applications Are Real-Time? (II)

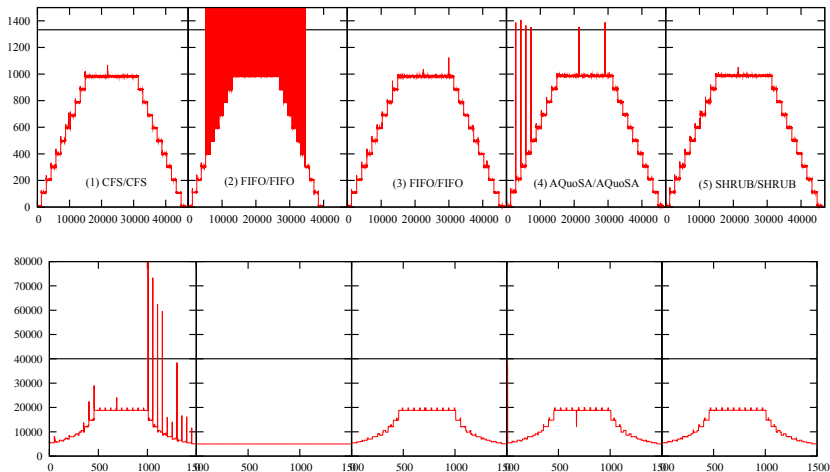


Concerning Latency... (I)



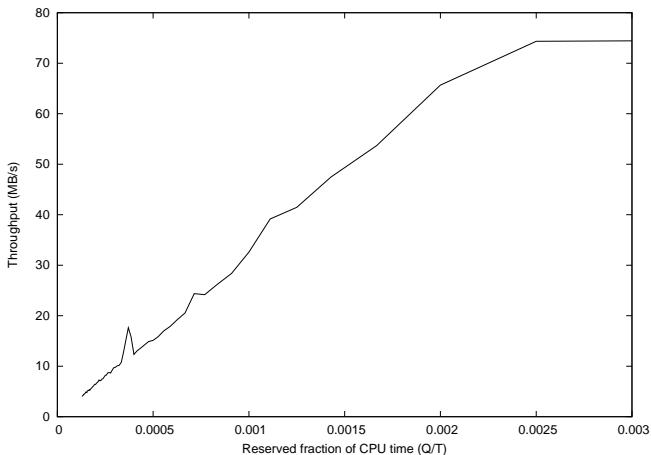
Two **concurrently running** video players

Concerning Latency... (II)



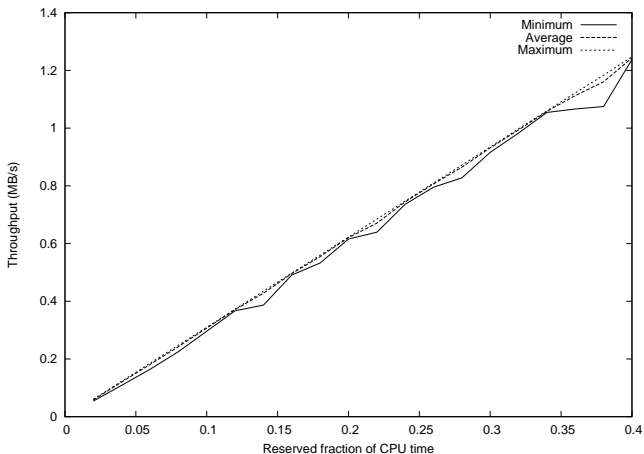
JACK with **10 periodic clients** plus a (fake!) **real-time task**

Concerning Throughput... (I)



Disk throughput (DMA enabled).

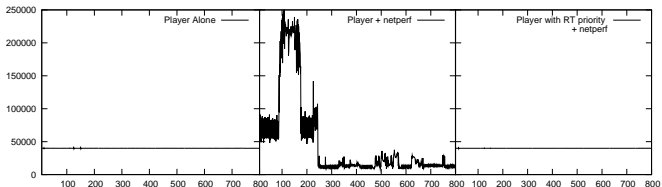
Concerning Throughput... (II)



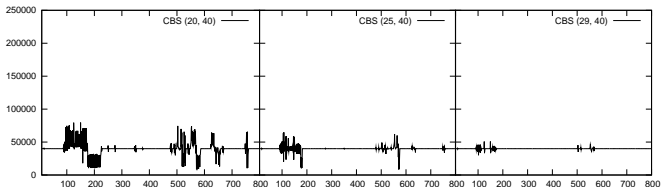
Disk throughput (without DMA).

Concerning Latency and Throughput...

A **video player** plus interrupt load by **netperf**.



Network throughput: -, 88 Mbps, 58 Mbps.



Network throughput: 88 Mbps, 76 Mbps, 59 Mbps.

Can Linux Do This... (I)

With POSIX-like fix priority scheduling?

- **Yes**, by means of implementing the SCHED_SPORADIC policy (part of POSIX).
Have a patch for that! :-)

With deadline-based scheduling (more convenient, sometimes more efficient)?

- **Yes**, by means of SCHED_DEADLINE¹ and/or EDF-throttling.
Have a patch for that! :-)

¹Together with Evidence s.r.l. within <http://www.artodroma-project.eu>

Can Linux Do This... (I)

With POSIX-like fix priority scheduling?

- **Yes**, by means of implementing the SCHED_SPORADIC policy (part of POSIX).
Have a patch for that! :-)

With deadline-based scheduling (more convenient, sometimes more efficient)?

- **Yes**, by means of SCHED_DEADLINE¹ and/or EDF-throttling.
Have a patch for that! :-)

¹Together with Evidence s.r.l. within <http://www.actor-project.eu>

Can Linux Do This... (I)

With POSIX-like fix priority scheduling?

- **Yes**, by means of implementing the SCHED_SPORADIC policy (part of POSIX).
Have a patch for that! :-)

With deadline-based scheduling (more convenient, sometimes more efficient)?

- **Yes**, by means of SCHED_DEADLINE¹ and/or EDF-throttling.
Have a patch for that! :-)

¹Together with Evidence s.r.l. within <http://www.actors-project.eu>

Can Linux Do This... (I)

With POSIX-like fix priority scheduling?

- **Yes**, by means of implementing the SCHED_SPORADIC policy (part of POSIX).
Have a patch for that! :-)

With deadline-based scheduling (more convenient, sometimes more efficient)?

- **Yes**, by means of SCHED_DEADLINE¹ and/or EDF-throttling.
Have a patch for that! :-)

¹Together with Evidence s.r.l. within <http://www.actors-project.eu>

Can Linux Do This... (II)

I just released v3 of SCHED_DEADLINE (last Friday).

Can Linux Do This... (II)

I just released v3 of SCHED_DEADLINE (last Friday).

I planned to have more code ready by now, but I got **preempted** by a very high priority "task", about a month ago...

Can Linux Do This... (II)

I just released v3 of SCHED_DEADLINE (last Friday).
I planned to have more code ready by now, but I got **preempted**
by a very high priority "task", about a month ago...



Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...

Now, The Really Important Part...

... Questions and open issues:

- Do you want this ?
- In what fashion? Priorities? Deadlines? Hierarchical? **All of them ;-P** ?
- With what user interface? New syscalls? Cgroups? Binary?
- At what privilege level/with what security model? FIFO/RR ones? Maybe less restrictions?
- Are users/application-developers ever start using this (if it's accepted)? Is there something that can/needs to be done toward this?
- How to balance stability of the API(s) and the new feature requests that might come out?
- ...