Deadline scheduling in the Linux kernel

Juri Lelli, Claudio Scordino, Luca Abeni and Dario Faggioli

Benno Fünfstück

July 8, 2019

Introduction

Story: A new coffee machine

Components:

- Brewing controller
- User interface controller
- Web interface

Linux for real-time

First approach: run Linux as a task in real-time hypervisor

- examples: RTAI, RTLinux, Xenomai
- maintenance of HAL and microkernel for real-time
- custom tools and API necessary for real-time part

Linux for real-time

First approach: run Linux as a task in real-time hypervisor

- examples: RTAI, RTLinux, Xenomai
- maintenance of HAL and microkernel for real-time
- custom tools and API necessary for real-time part

Second approach: make the Linux kernel itself suitable for real-time

- PREEMPT_RT patchset

Linux for real-time

First approach: run Linux as a task in real-time hypervisor

- examples: RTAI, RTLinux, Xenomai
- maintenance of HAL and microkernel for real-time
- custom tools and API necessary for real-time part

Second approach: make the Linux kernel itself suitable for real-time

- PREEMPT_RT patchset
- real-time scheduler

Design of a realtime scheduler

Scheduler requirements

Predictability: worst-case behaviour must be apparent

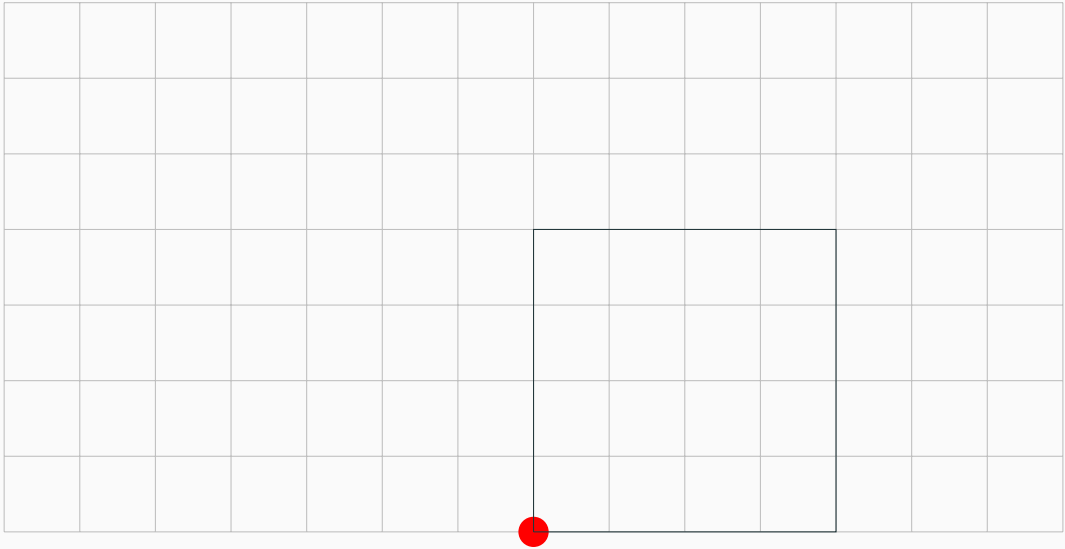
-

Temporal isolation: other tasks should not negatively affect our task

Two important properties

- Predictability
- Temporal isolation

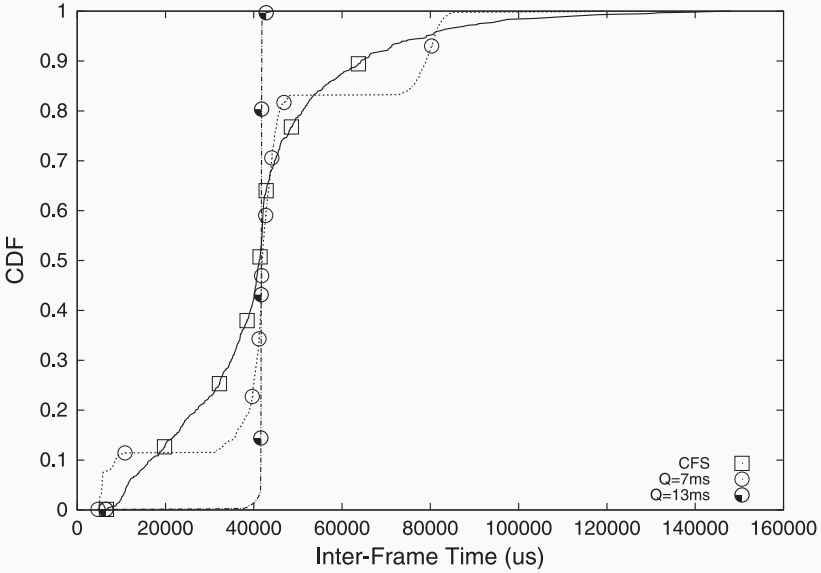
Todo



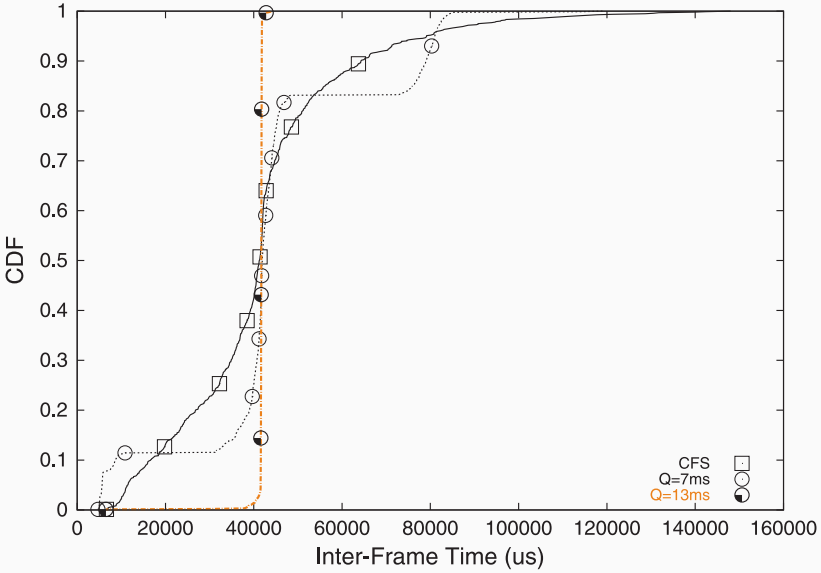
Evaluation

Synthetic workload

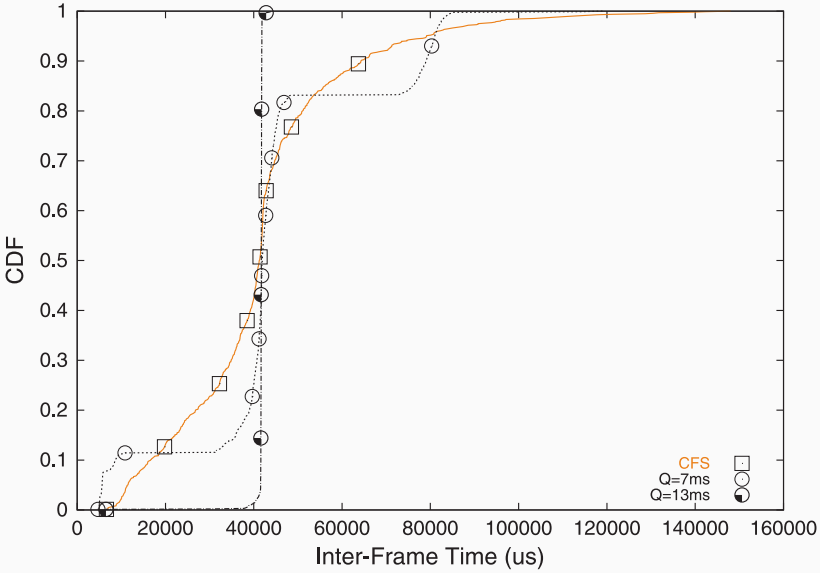
MPlayer on loaded system



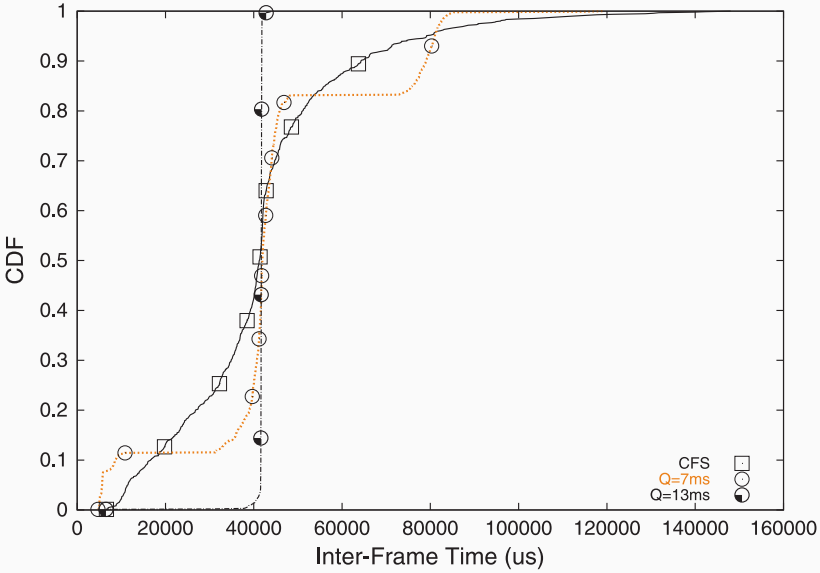
MPlayer on loaded system



MPlayer on loaded system



MPlayer on loaded system



Virtual router

Conclusion

Still some work left:

- M-BWI: Multiprocessor Bandwidth Inheritance
- Power aware algorithms (example: GRUB-PA)
- Support cgroups interface

Not rocket science, but solid implementation of proven concepts (EDF, CBS)

... ready for production use
... upstream in the Linux kernel
... with simple to use API

We can run our coffee machine on stock linux! :)