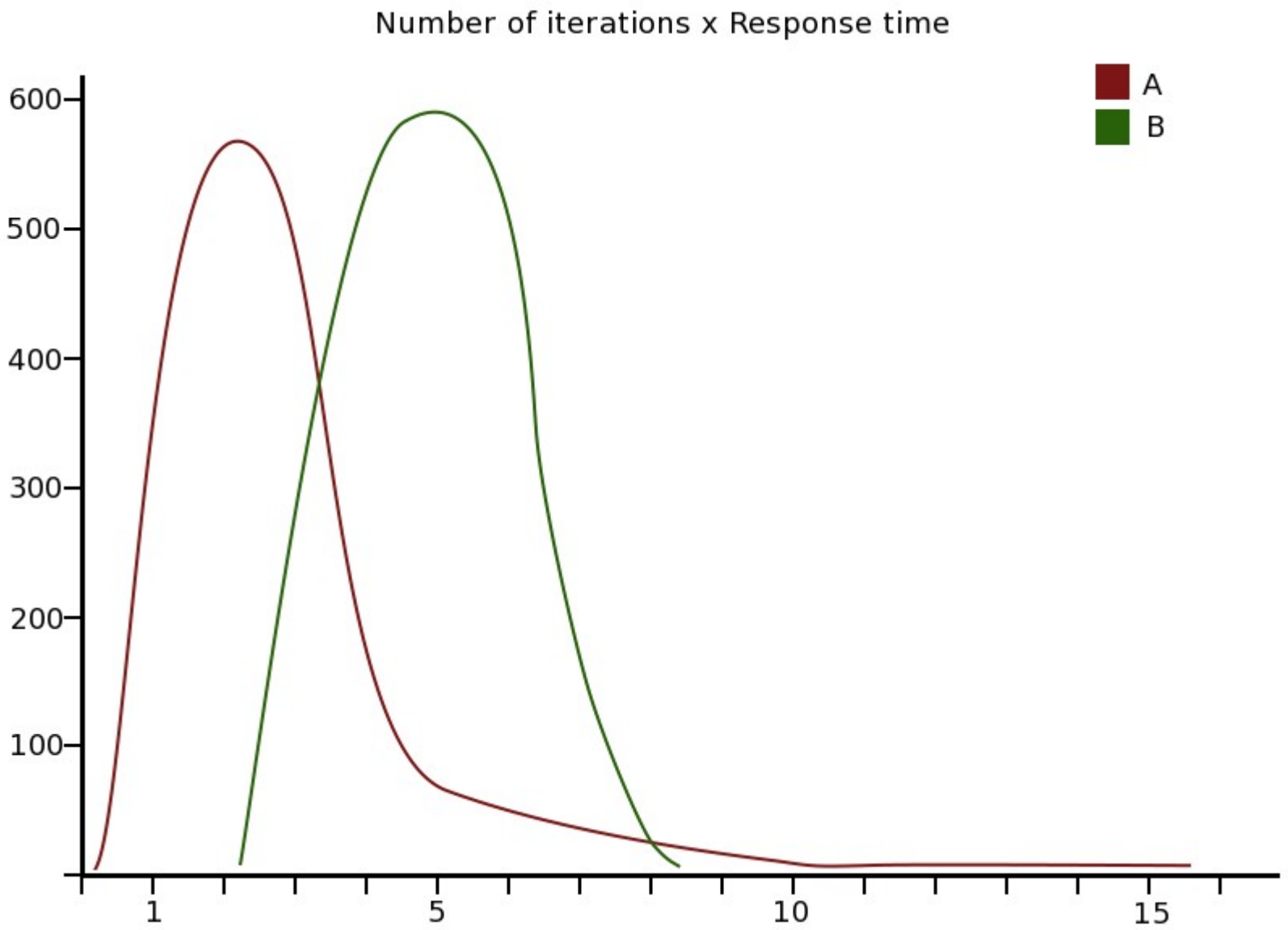
Real-Time Linux: From theory until SCHED_DEADLINE

bristot@redhat.com

What does real-time mean?

Response Time
Deadline
Determinism
Worst Case

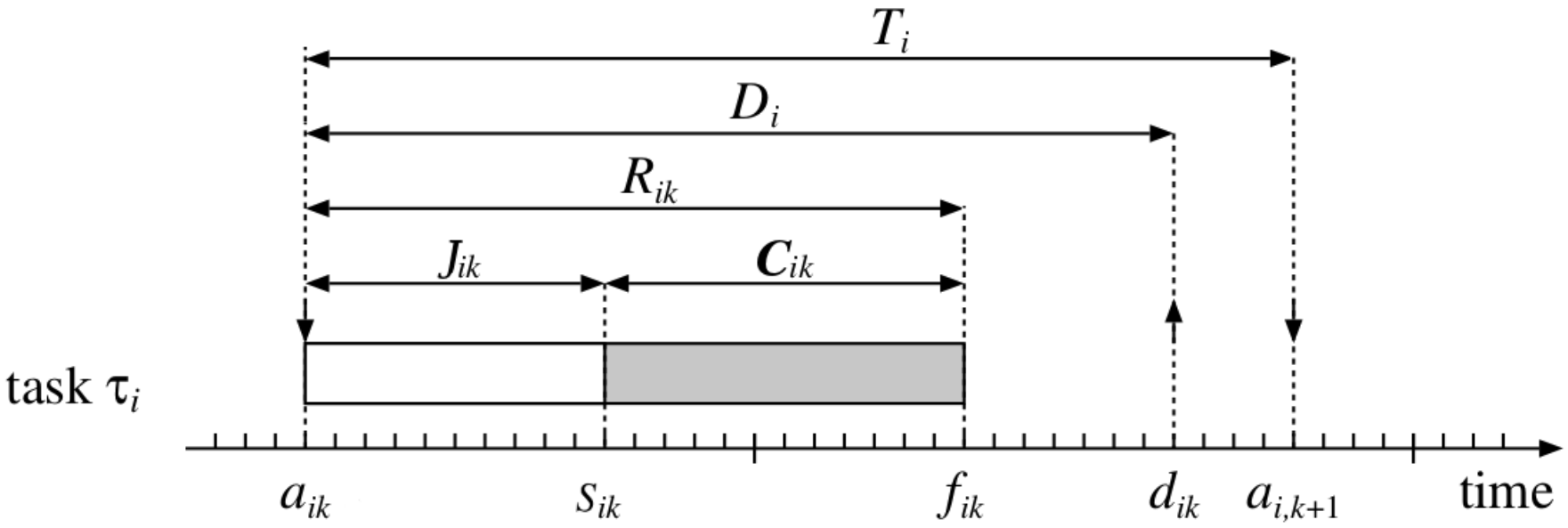
Real Time != Real Fast



RT Theory

- set of n tasks $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$
- set of m processors $\rho = \{\rho_1, \rho_2, ..., \rho_m\}$
- set of q resources $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_q\}$.
- A system is schedulable when it is possible to assign processors from ρ and resources from σ to tasks in τ, in order to complete all tasks in τ while meeting the timing requirements.

Common task description



Example of tasks

τ_1 {

T = 5

C = 3

D = T = 5

}

τ_2 {

T = 3

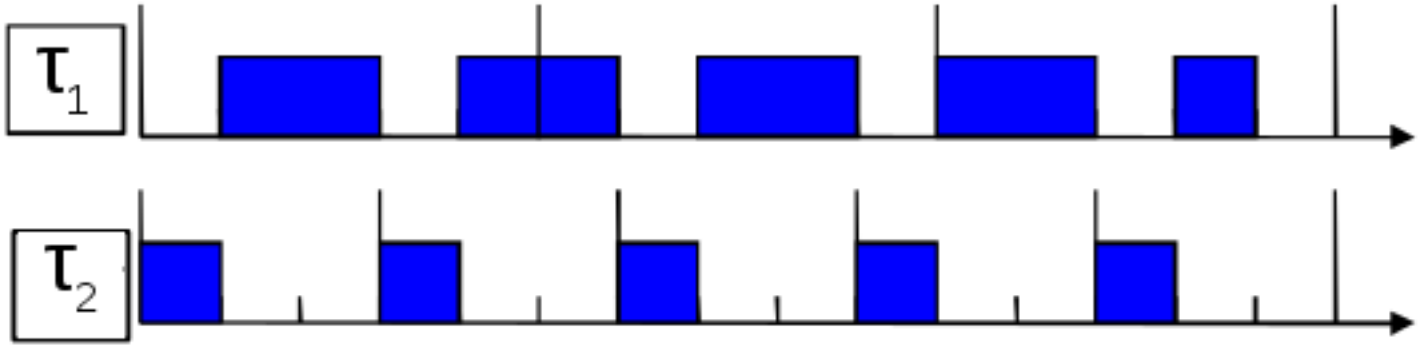
C = 1

D = T = 3

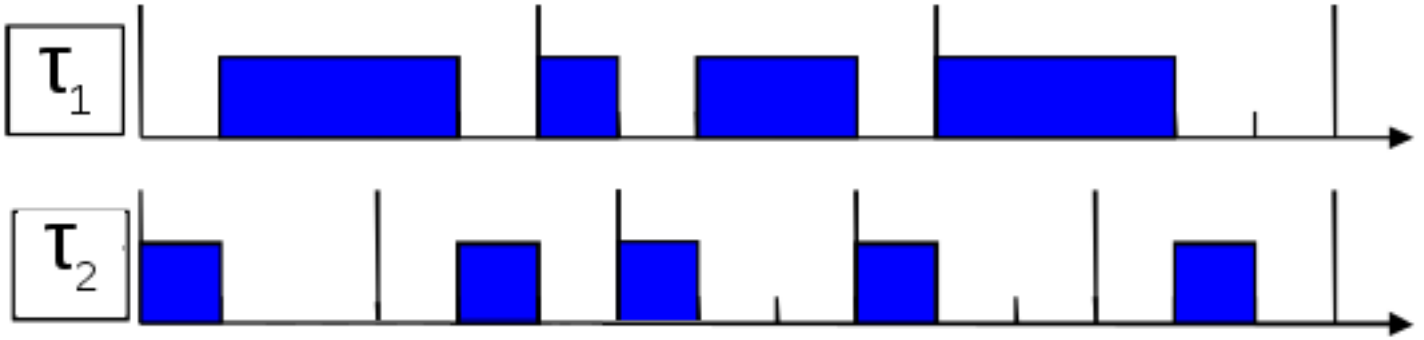
}

Sched algorithms

- Fixed priority (Rate monotonic):



- Dynamic priority (EDF):



Math proof (single core)

- Fixed priority is not optimum
  - response time analysis:

$$\mathrm{R}_i = W_i + J_i$$

$$\mathrm{W}_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{W_i + J_j}{P_j} \right\rceil . C_j$$

- EDF is optimum!
  - U <= 1 is enough

Multi core

- How to spread tasks over the cores?
  - Partitioned – M single core processors
  - Global
  - Semi partitioned
  - Clustered
- This changes break a lot of rules
  - Anomalies
  - Global EDF is not optimum
  - A new theory is/was needed

Real-Time Linux

- Linux is a General Purpose OS
  - The majority of users wants a Real Fast OS!
    - Including the majority here...
- Linux is too much complex to be math proof
  - A lot of contexts
  - A lot of lock implementations
    - Some of them not deterministic
- Linux does not uses RT abstractions
  - What does a deadline mean?

What does Real-Time Linux mean?

Real-Time Linux

- Real-Time Scheduler (until 3.14)
  - Fixed priority RT schedulers:
  - Priorities: 1-99
- Is it enough?
- Linux is a General Purpose OS
  - Throughput is the main metric for the majority
    - Sources of latency
      - Go away! I'm a softirq!
  - That implies on priority inversions = !determinism

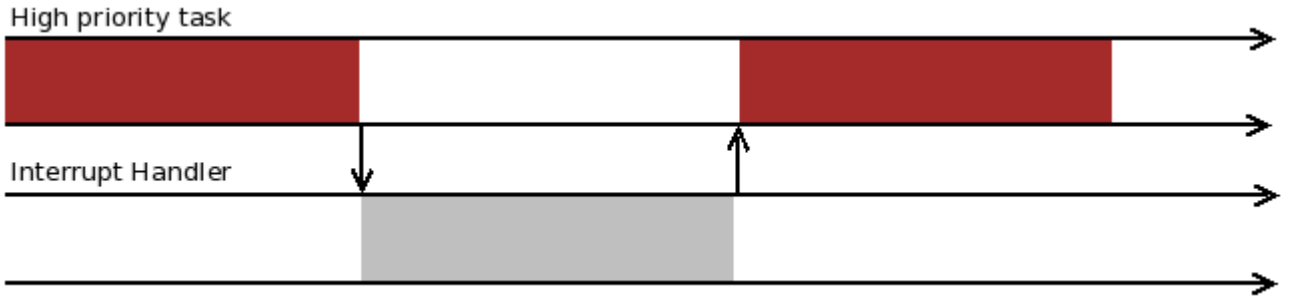
The PREEMPT RT Patch

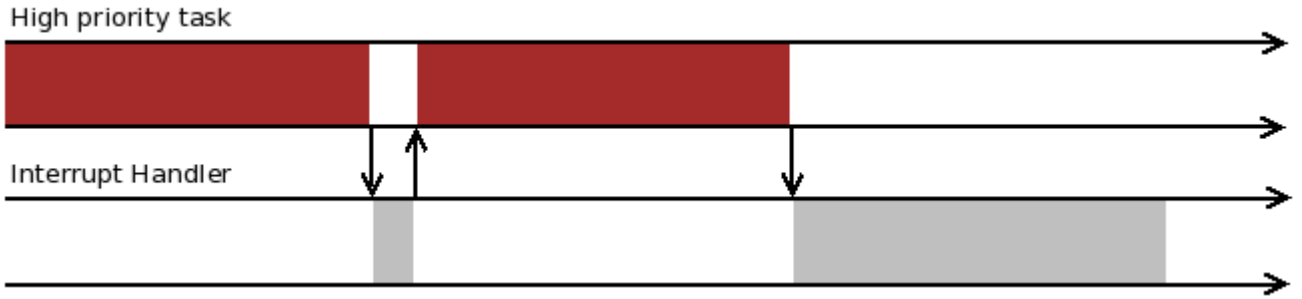
- Adds more determinism to the Linux kernel
- Full Preemptive kernel
- Fine grained prioritization of kernel tasks
- Eliminates sources of unbounded priority inversion
- low latency (< 150us ?, < 50 us?)

The PREEMPT RT Patch

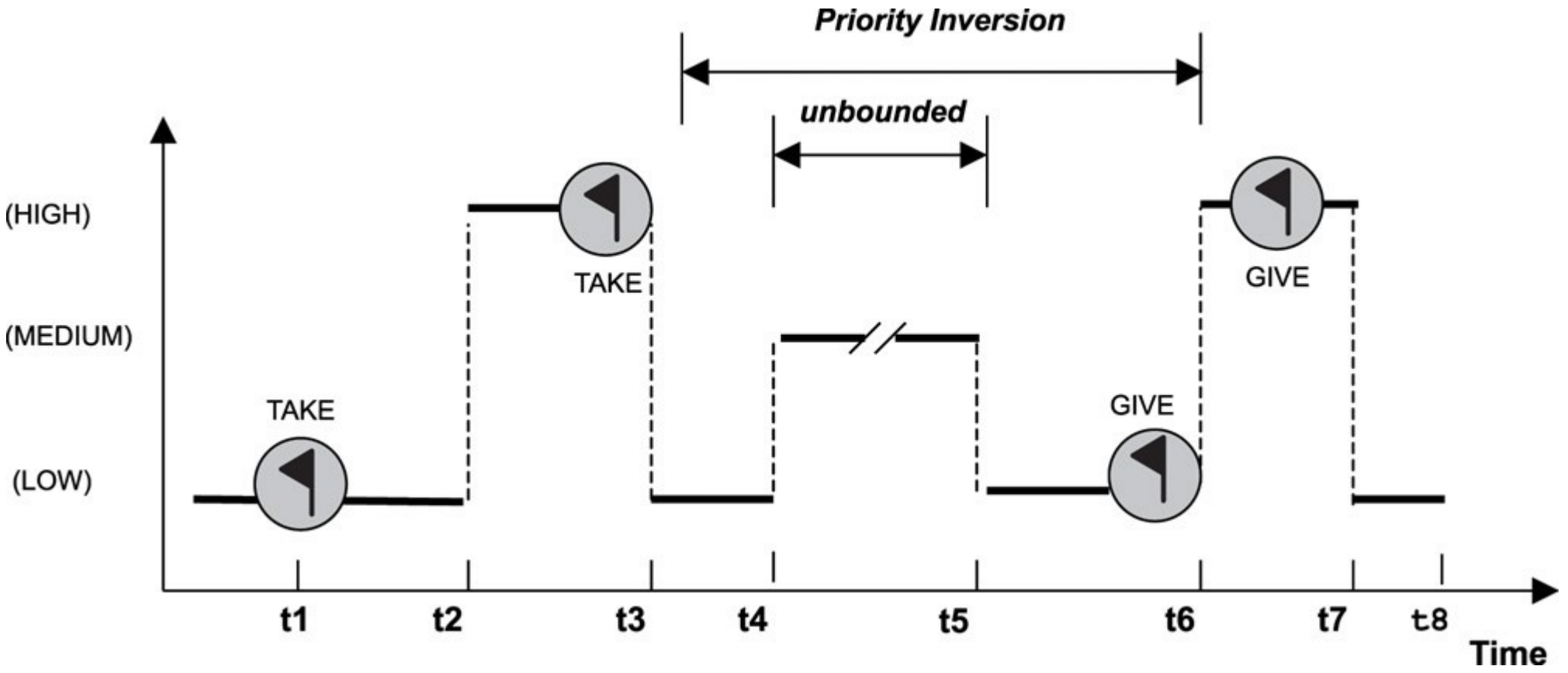
- Linux has several preemption models:
  - PREEMPT_NONE:
    - Better for throughput
    - High latency
  - [...]
  - PREEMPT_RT_FULL (RT exclusive):
    - Better for latency
    - Who cares about throughput?
      - I bet most of you guys care, that's why I told you that RT != FAST.

IRQ as threads





RT Mutex: PHP


Priority Inversion
unbounded
(HIGH)
(MEDIUM)
(LOW)
TAKE
TAKE
GIVE
GIVE
t1
t2
t3
t4
t5
t6
t7
t8
Time

PREEMPT-RT

- + a lot of patchs to add more determinism
- You don't need to change your APP
- /me uses it on ARM and Intel
- RHEL6 MRG Realtime kernel

SCHED DEADLINE

- EDF Scheduler
- Uses the RT Abstractions:
  - runtime >= C (WCET)
  - deadline = D
  - period <= T
- CBS – Controls the task bandwidth

Scheduler hierarchy

- stop_machine
- deadline
- rt (fixed priority)
- fair
- idle

Litmus RT

- Linux testbed for multiprocessors scheduling in Real-Time systems
- The state of art - academic approach
- Simplify the prototyping of multiprocessor real-time scheduling and synchronization algorithms
- http://www.litmus-rt.org/

Litmus RT

- Scheduler Plugins
  - GSN-EDF (global)
  - PSN-EDF (partitioned)
  - C-EDF (Clustered)
  - Pfair
  - A lot of research algorithms for RT
- It's used on a lot of high quality papers
  - It's easy to understand them if you understand the LK

Questions?