Understanding a Real-Time System

More than just a kernel

Steven Rostedt
rostedt@goodmis.org
srostedt@redhat.com

What is Real-Time?

Deterministic results

Repeatable results

Doing what you expect when you expect it

No unbounded latency

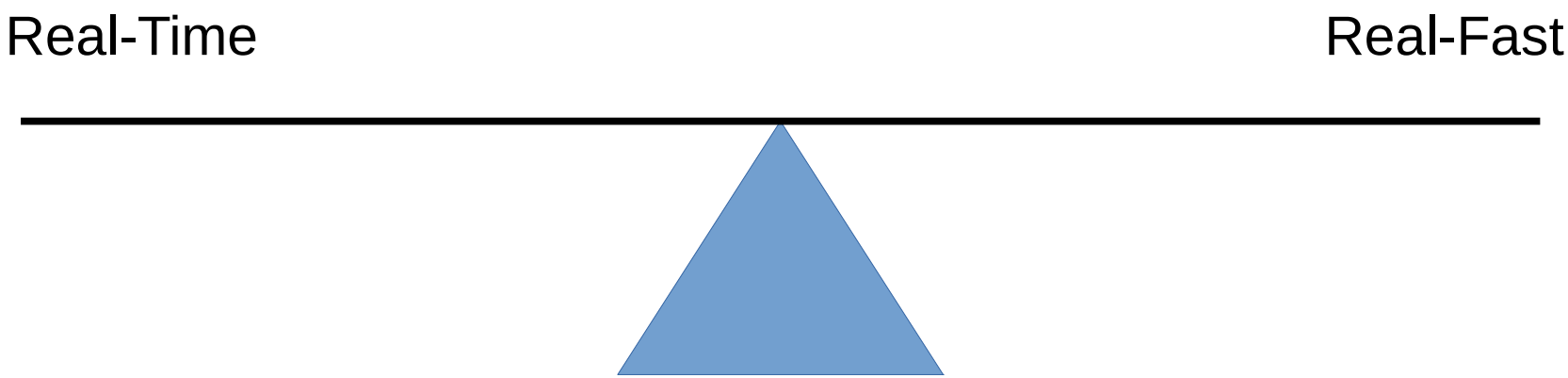
Can calculate worst case scenarios

All environments are Real-Time.

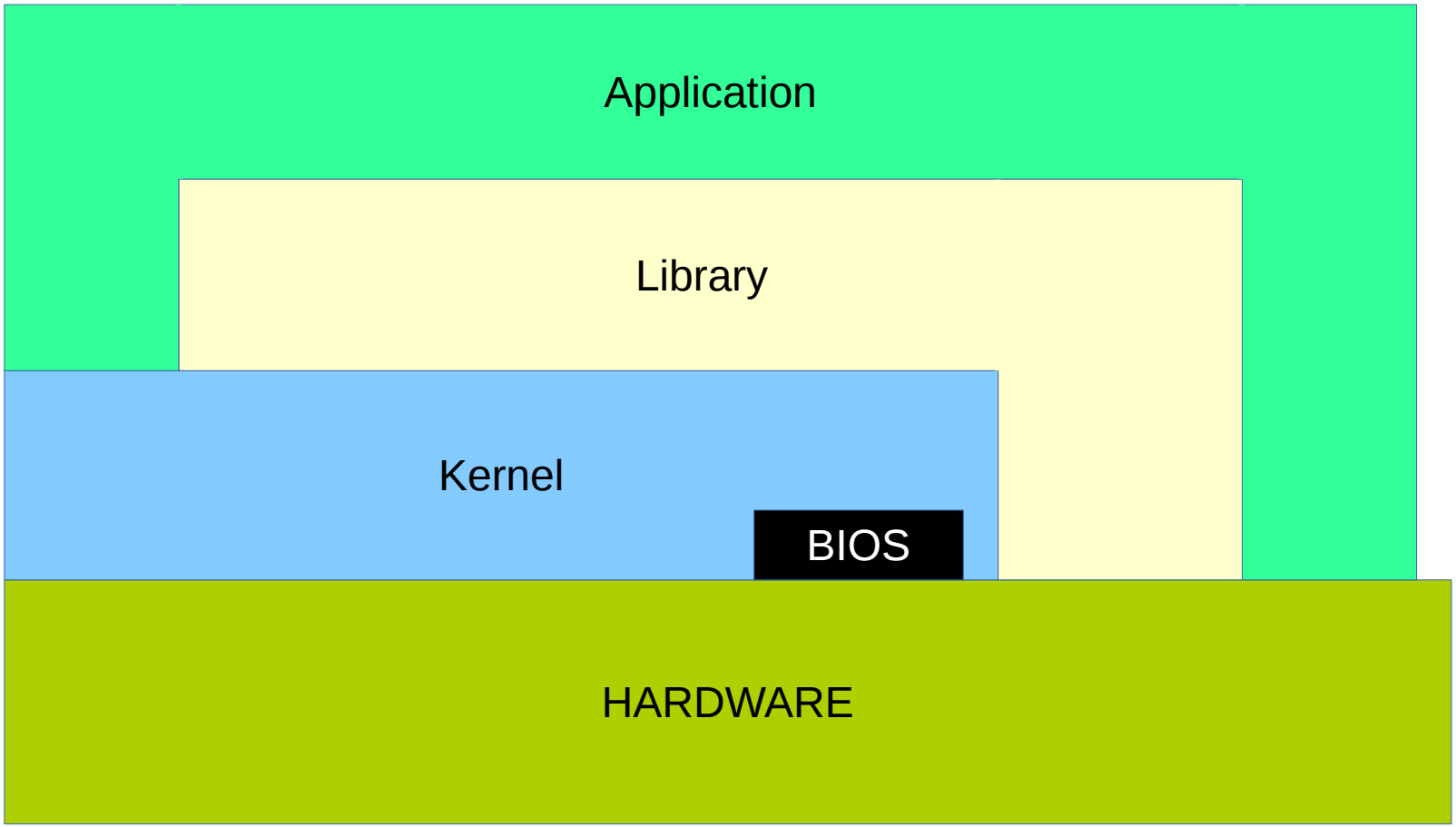
What is Real Fast?

- **Hot cache**
  - **Look ahead features**
- **Paging**
  - **Translation Lookaside Buffer (TLB)**
- **Least interruptions**
- **Optimize the most likely case**
  - **Transactional Memory**

Real-Time vs Real-Fast



The System



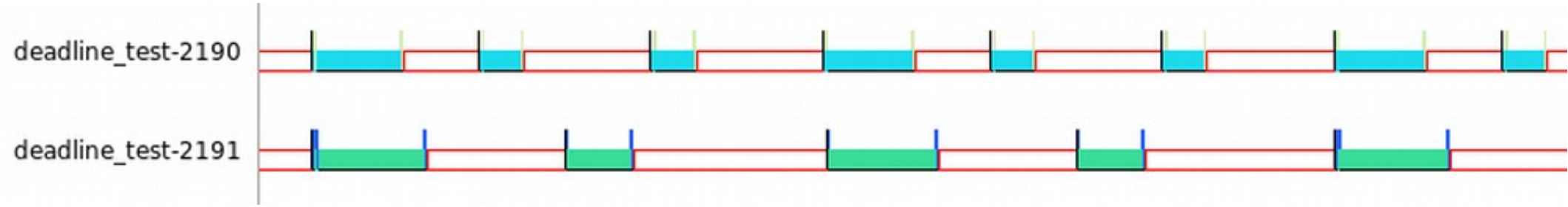
The Hardware

- **The foundation**
  - **If this isn't deterministic, forget the rest**
- **Memory Cache**
- **Branch Prediction**
- **NUMA**
- **Hyper-Threading**
- **TLB**
- **Transactional Memory**
- **SMI**
- **CPU Frequency scaling**

Memory Cache

- Try to run tests with cold cache
- Try to find the worse case scenario
- If you system works without cache, it should work with cache
  - Except for race conditions (You just can't win can you?)
  - Non cache is more deterministic
  - Cache may allow the “slower” path to run faster

Memory Cache



Branch Prediction

CPU recognizes branch patterns

Optimizes the pipeline

But what happens when logic changes?

Branch Prediction

```
Good:
 Performance counter stats for './deadline_test -c 0,3':

      15309.175906      task-clock (msec)         #    1.272 CPUs utilized            (100.00%)
            16,693      context-switches          #    0.001 M/sec                    (100.00%)
                 6      cpu-migrations            #    0.000 K/sec                    (100.00%)
               220      page-faults               #    0.014 K/sec                    (100.00%)
    45,336,201,800      cycles                    #    2.961 GHz                      (100.00%)
    27,839,671,679      stalled-cycles-frontend   #   61.41% frontend cycles idle     (100.00%)
   <not supported>      stalled-cycles-backend
    24,654,001,731      instructions              #    0.54  insns per cycle
                                                  #    1.13  stalled cycles per insn  (100.00%)
     5,846,443,551      branches                  #  381.891 M/sec                    (100.00%)
           798,866      branch-misses             #    0.01% of all branches          (100.00%)
    15,143,395,012      L1-dcache-loads           #  989.171 M/sec                    (100.00%)
         6,830,685      L1-dcache-load-misses     #    0.05% of all L1-dcache hits    (100.00%)
         5,646,962      LLC-loads                 #    0.369 M/sec                    (100.00%)
   <not supported>      LLC-load-misses

      12.037594790 seconds time elapse
```

Branch Prediction

```
Bad:
 Performance counter stats for './deadline_test -c 0,3':

       9191.898036      task-clock (msec)         #    0.763 CPUs utilized            (100.00%)
            16,693      context-switches          #    0.002 M/sec                    (100.00%)
                 9      cpu-migrations            #    0.001 K/sec                    (100.00%)
               219      page-faults               #    0.024 K/sec                    (100.00%)
    22,043,401,852      cycles                    #    2.398 GHz                      (100.00%)
    13,531,252,221      stalled-cycles-frontend   #   61.38% frontend cycles idle     (100.00%)
   <not supported>      stalled-cycles-backend
    12,012,005,499      instructions              #    0.54  insns per cycle
                                                  #    1.13  stalled cycles per insn  (100.00%)
     2,841,672,774      branches                  #  309.150 M/sec                    (100.00%)
         4,689,983      branch-misses             #    0.17% of all branches          (100.00%)
     7,339,066,676      L1-dcache-loads           #  798.428 M/sec                    (100.00%)
         6,443,901      L1-dcache-load-misses     #    0.09% of all L1-dcache hits    (100.00%)
         5,131,751      LLC-loads                 #    0.558 M/sec                    (100.00%)
   <not supported>      LLC-load-misses

      12.040237863 seconds time elapsed
```

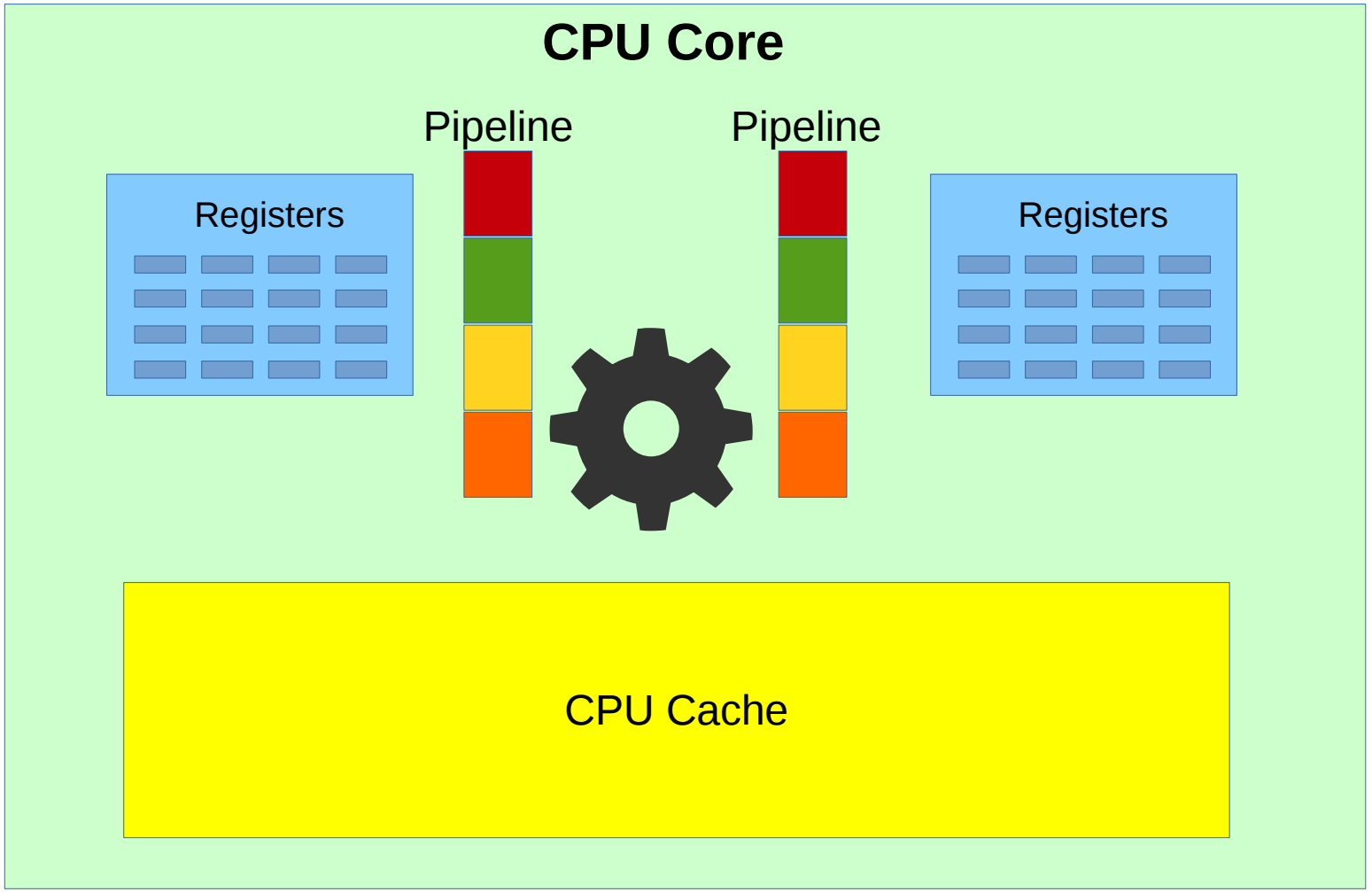
NUMA

- Memory speeds dependent on CPU
- Need to organize the tasks
- Make sure RT tasks always have their memory in one place

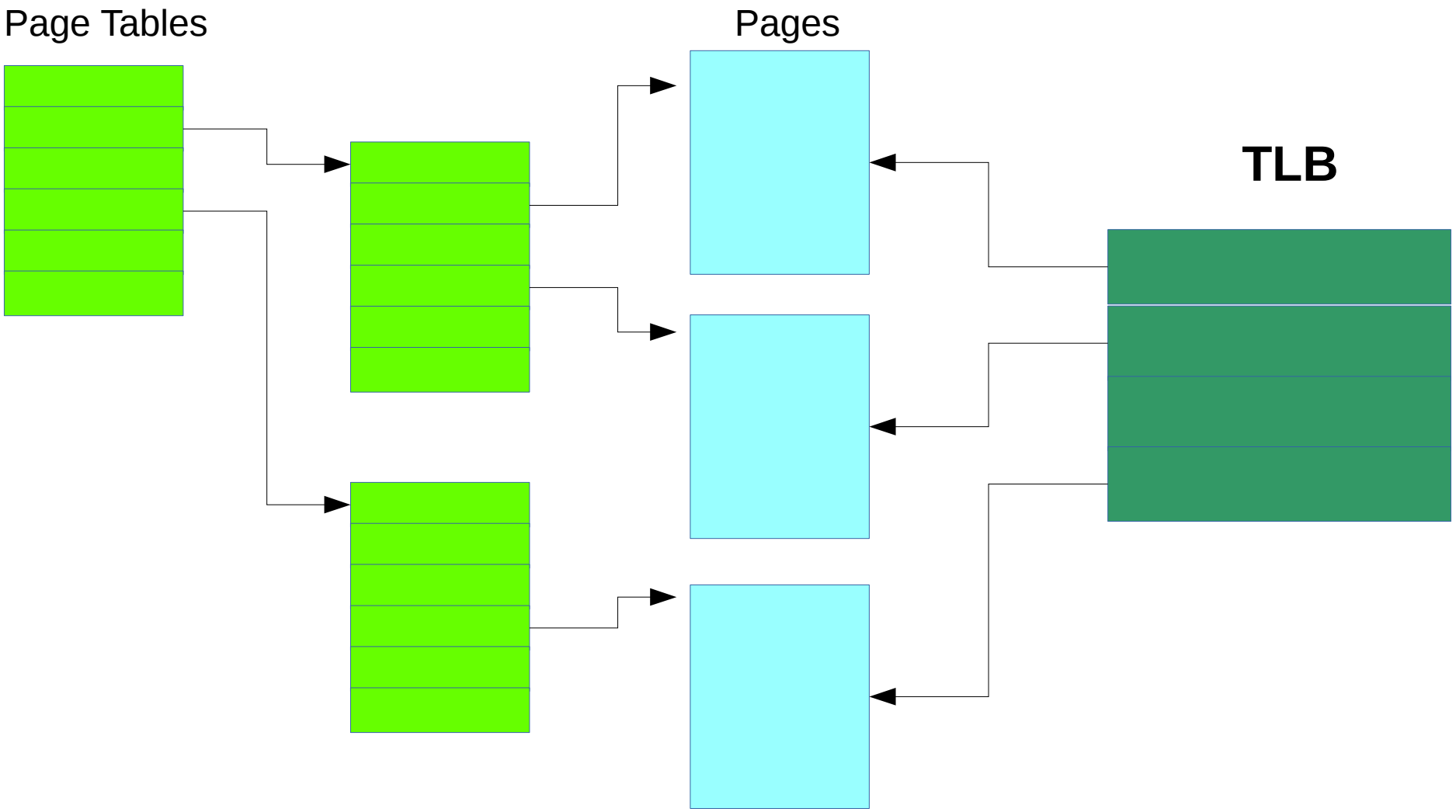
Hyper-Threading

- Intel processor
- One execution unit
- One system bus
- One cache
- Two sets of registers
- Two sets of CPU pipelines
- Execution engine switches between them on stall
- Recommended to disable for RT

Hyper-Threading



Translation Lookaside Buffer (TLB)



Transactional Memory

- Allows for parallel actions in the same critical section
- Backs out when the same memory is touched
- Restart the transaction or take another path

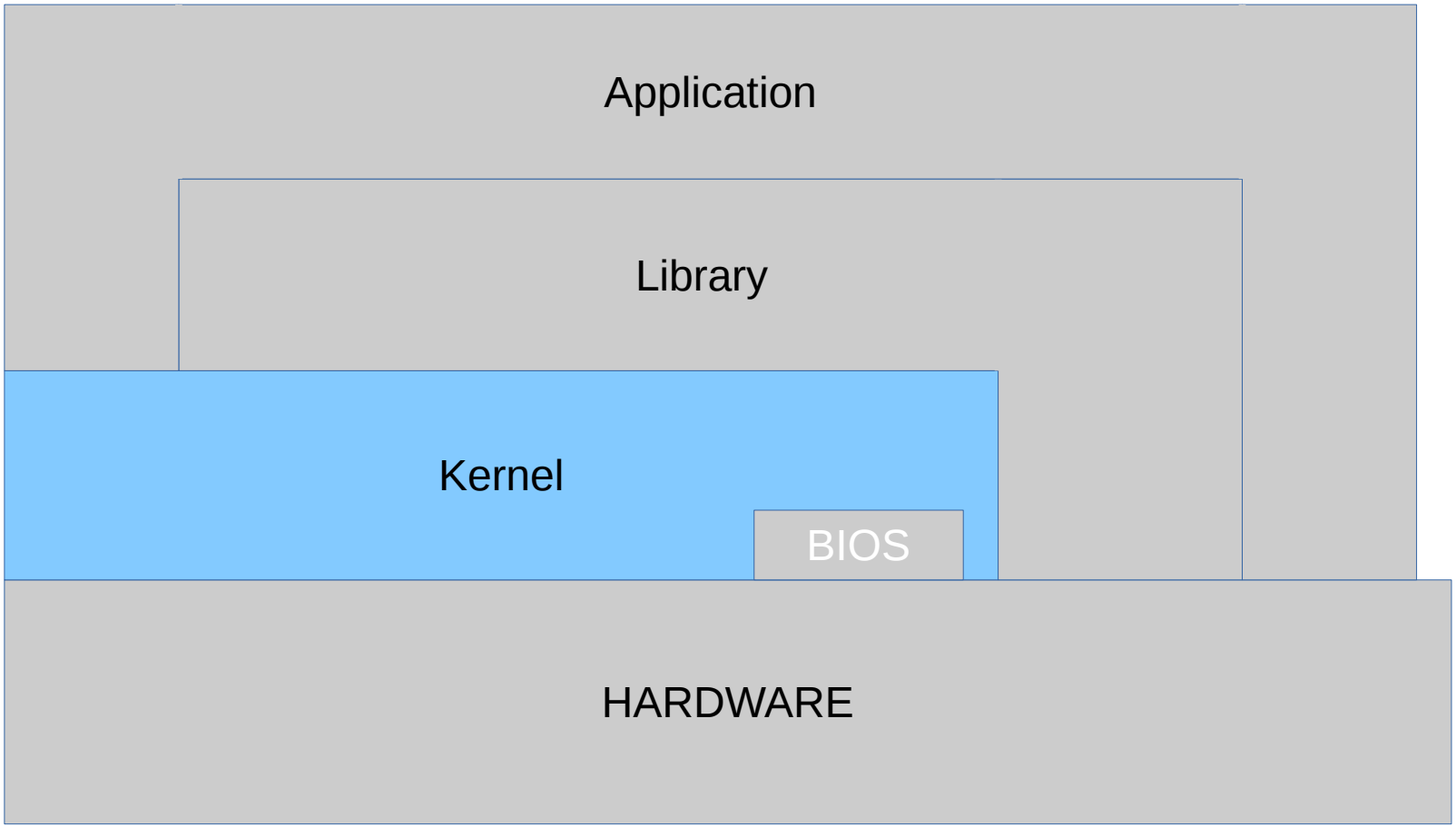
System Management Interrupt (SMI)

- Puts processor into System Management Mode (SMM)
- HW functionality done in software
- Check CPU temperature Change frequency
- Perform ECC memory scans
- Causes the system to stop what it was doing

CPU Frequency Scaling

- **Battery saving**
  - **Run at full blast!**
- **CPU Idle**
  - **Run a polling loop**
  - **Don't go into a deep sleep**
  - **Comes out groggy**

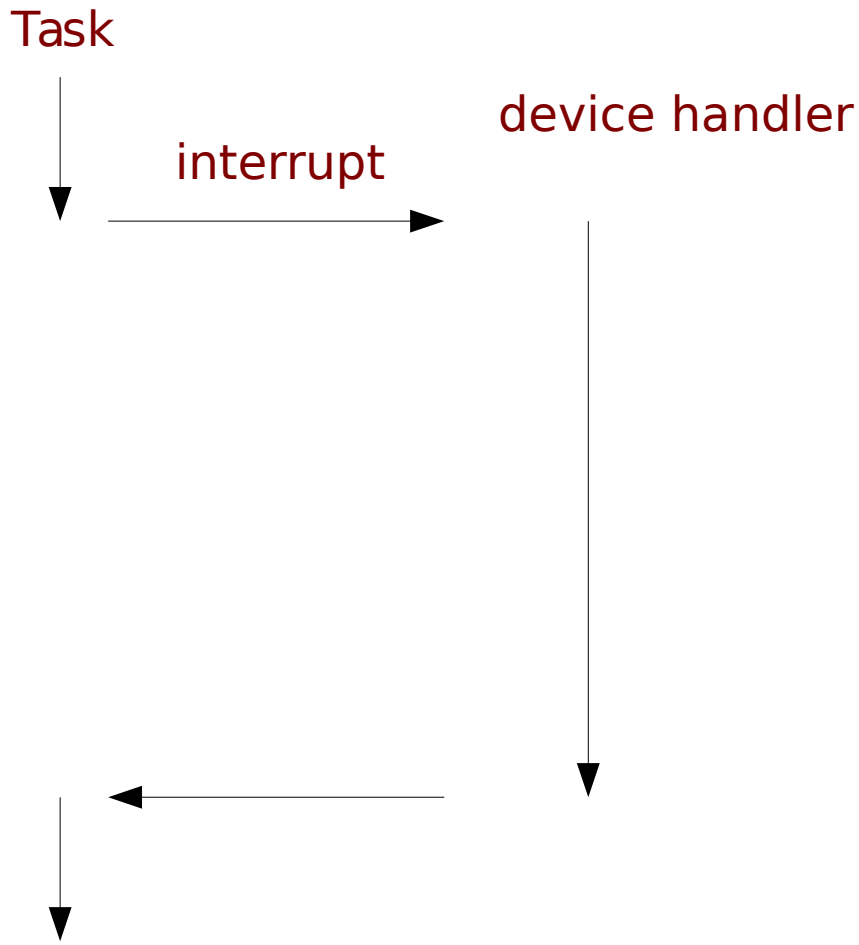
The Kernel



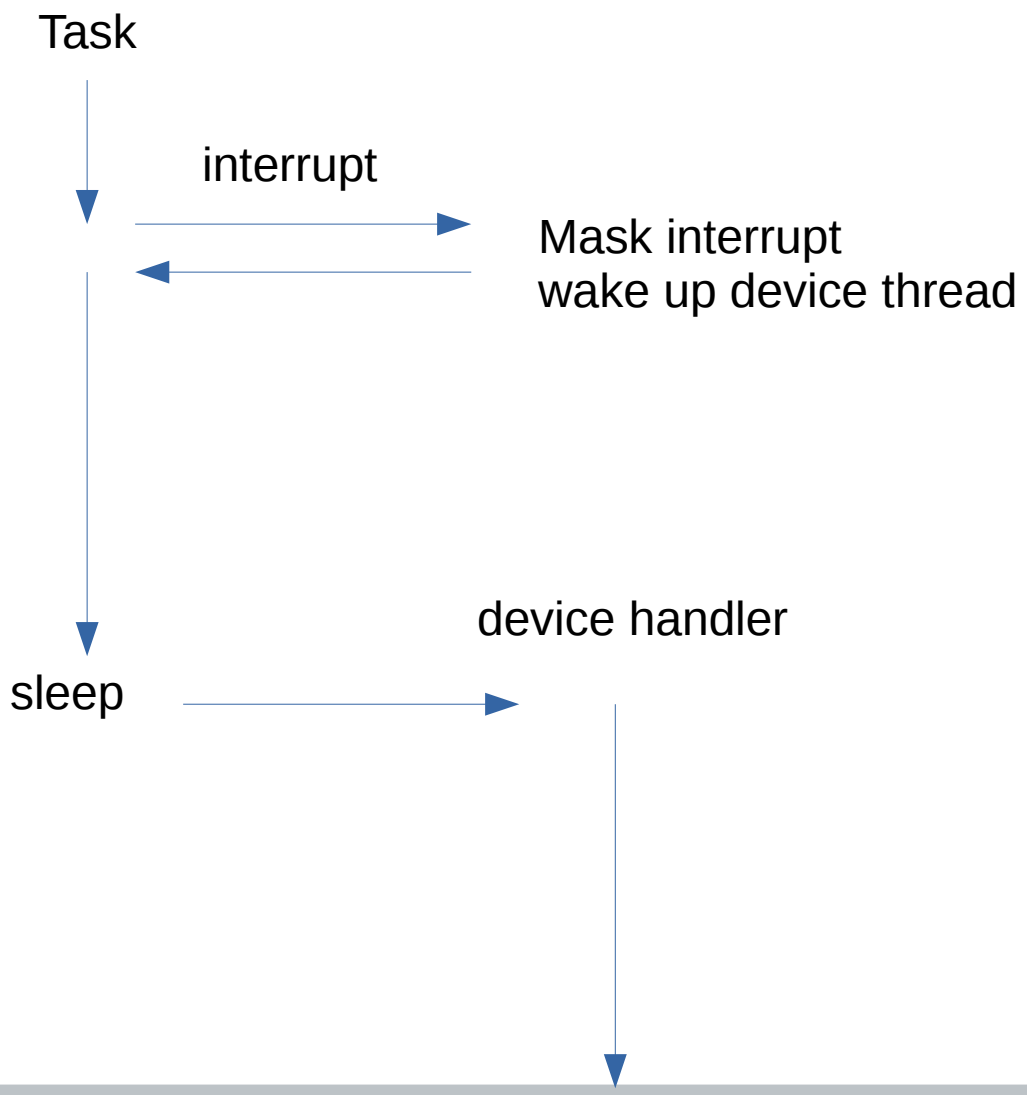
RT Kernel

- Threaded interrupts
- System management threads
- High res timers
- CPU Isolation
- No HZ
- No HZ Full

Normal Interrupts



Threaded Interrupts



Threaded Interrupts

- User tasks can run higher priority than interrupts
- Set required interrupts higher than your task
  - i.e. Don't poll waiting for network if task is higher priority than networking interrupts
- Know your system!

Soft Interrupts

- With PREEMPT_RT, softirqs run from the context of who raises them
  - Network irq will run network softirqs
- Except for softirqs raised by real Hard interrupts
  - RCU
  - Timers
  - Run in ksoftirqd

System Management Threads

RCU

Watchdog

Migrate

kworker

ksoftirqd

posixcputimer

Timers

- **setitimer()**
  - Requires ksoftirqd to run (on PREEMPT_RT)
- **timer_create() / timer_settime()**
  - Timer interrupt wakes up posixcputimer thread
- **Uses high resolution timer kernel infrastructure**
- **Sends via signals**

CPU Isolation

Kernel parameter: isolcpus=1-3

- **no longer the preferred method**

cpusets

- **cd /sys/fs/cgroup/cpuset/**
- **echo 1 > cpuset.cpu_exclusive**
- **mkdir myset**
- **echo 1-3 > myset/cpuset.cpus**
- **echo 1 > myset/cpuset.cpu_exclusive**
- **echo $$ > myset/tasks**

NO_HZ

CONFIG_NO_HZ

When CPU is idle, turn off timers

Lets CPUs go into a deep sleep

Great for power savings

Sucks for latency (deeper sleep, longer wakeup)

NO HZ FULL

CONFIG_NO_HZ_FULL

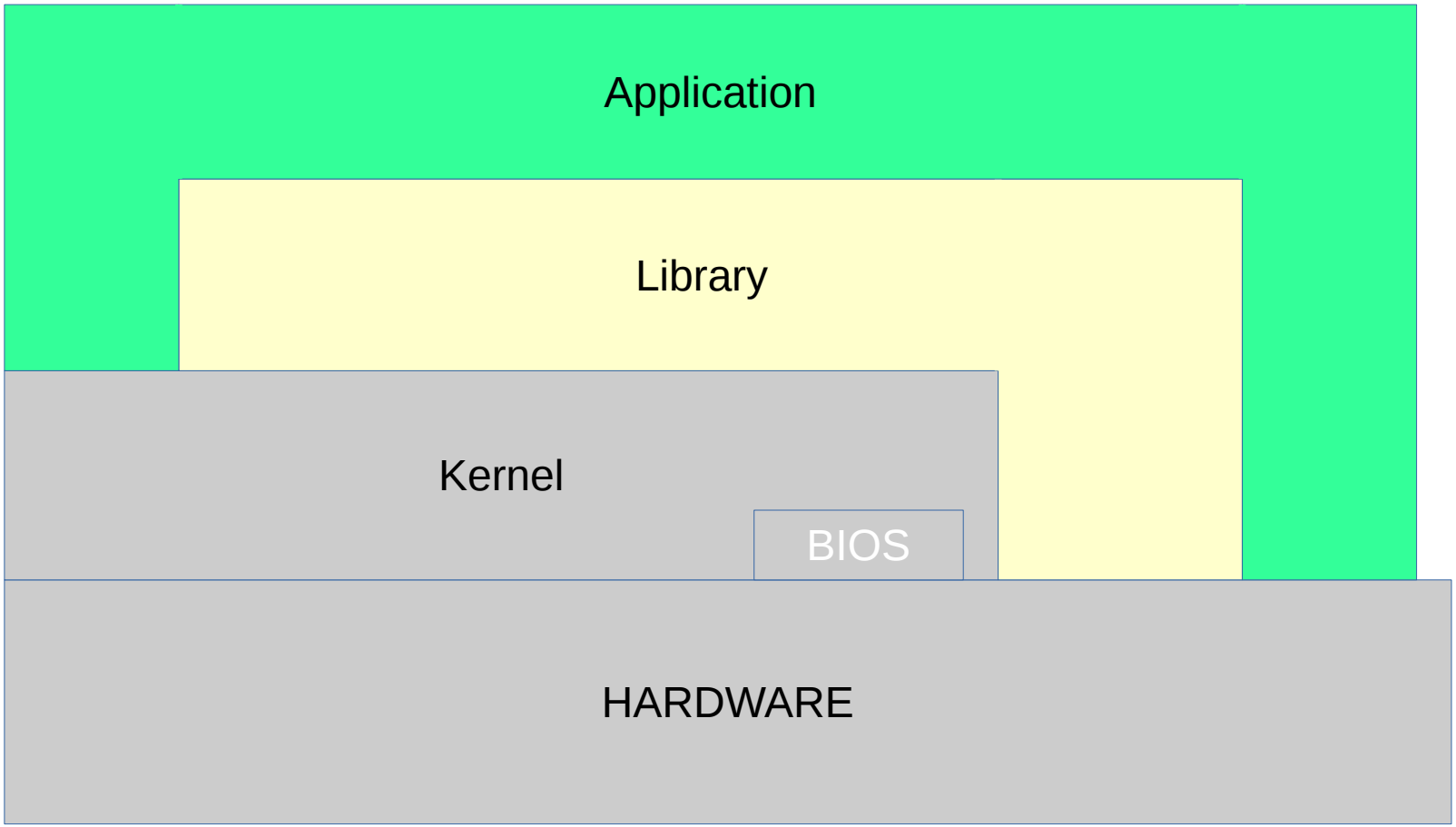
Keep kernel processing from bothering tasks

Kernel parameter: nohz_full=3 rcu_nocbs=3

Works when only one task is scheduled

Adds overhead to kernel entry and exit

The Application / Library



RT Tasks

- memory locking
- Priority inheritance locks
- Task and interrupt thread dependencies
- Migration is different

memory locking

mlockall()

- Lock in memory to prevent page faults

MCL_CURRENT

- Lock in all current pages

MCL_FUTURE

- Lock in pages for heap and stack and shared memory

Careful about how much you lock in!

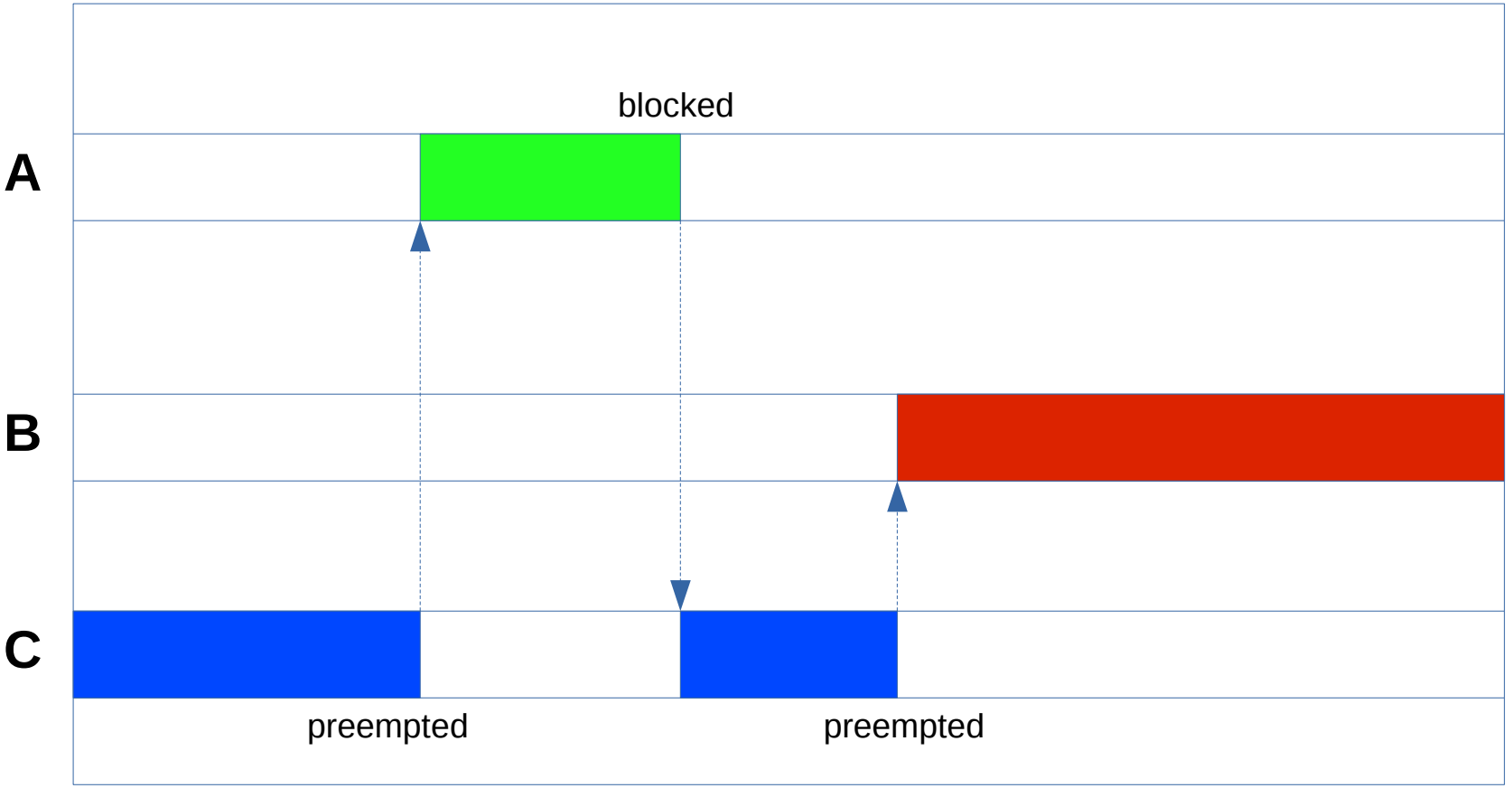
Priority Inheritance Locking

Prevents Unbounded Latency

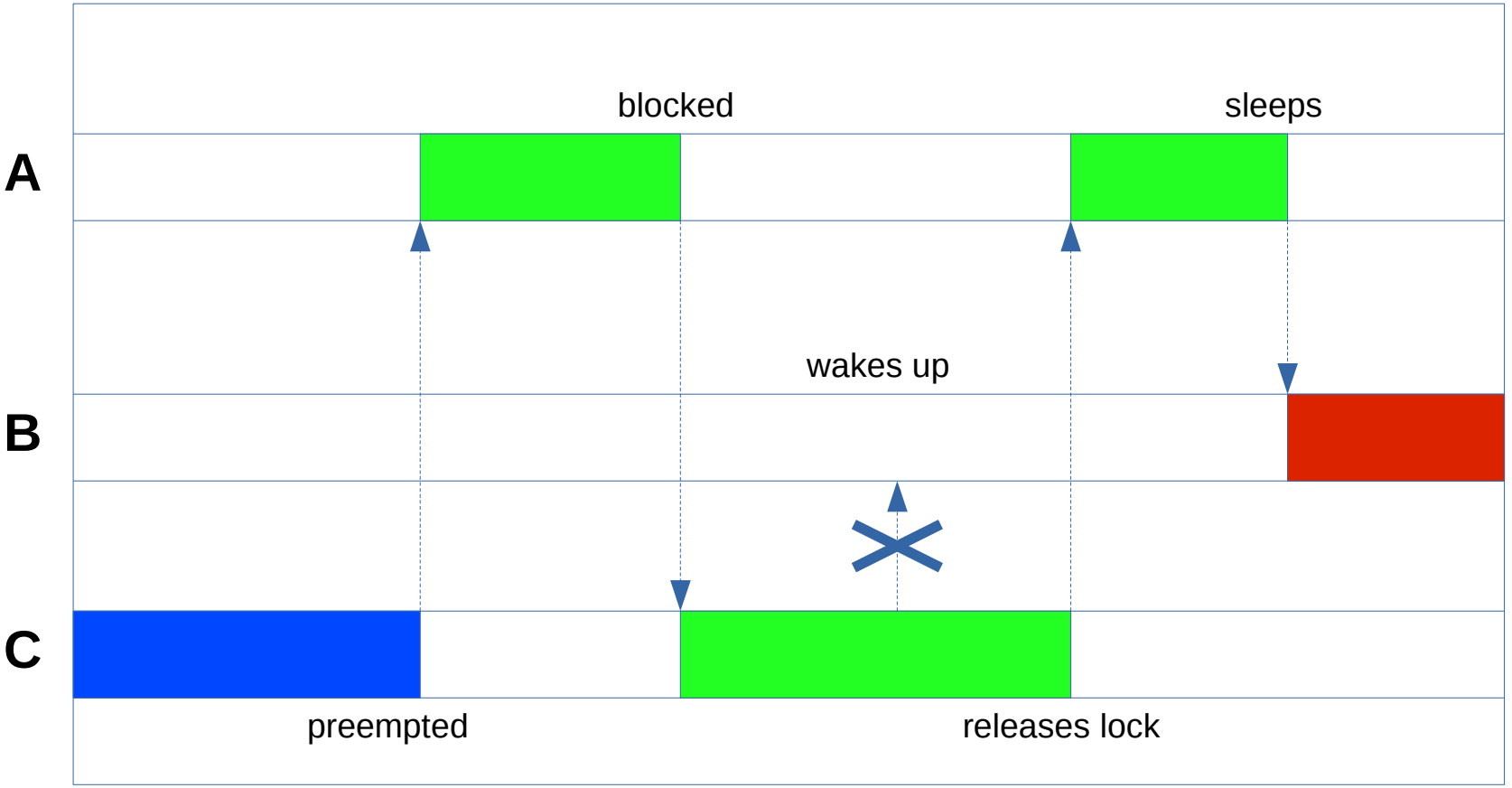
For threaded applications

pthread_mutexattr_setprotocol (&attr, PTHREAD_PRIO_INHERIT)

Unbounded Latency



Priority Inheritance



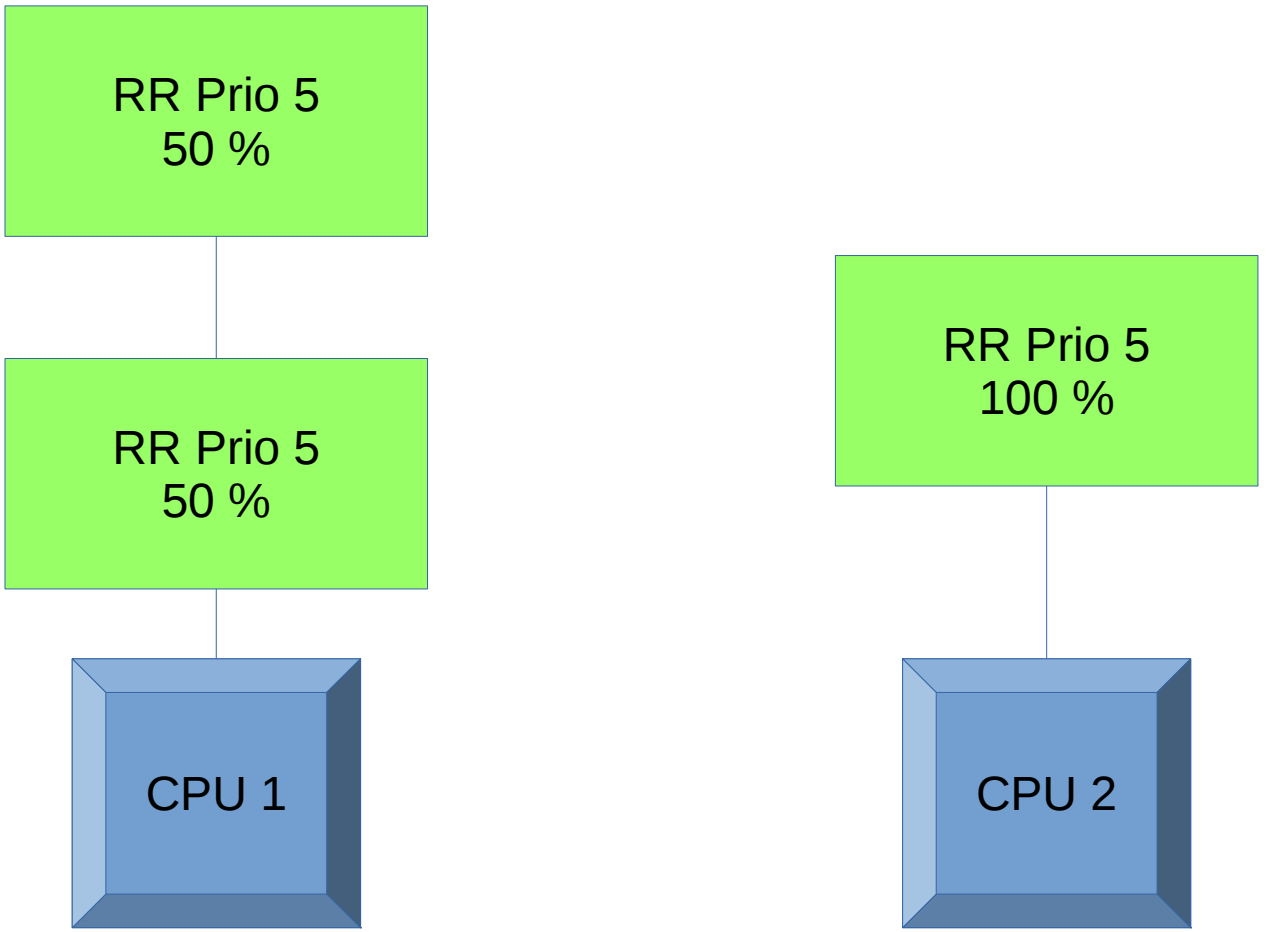
Task and interrupt thread dependencies

- Understand how threads interact
- Know your interrupt threads
- cpuposixtimer
- Workqueues
- Beware of pitfalls

Real-time vs Multi processors

- Migration clears Caches (memory and TLB)
- The RT kernel gives you a “best effort”
  - Your mileage may vary
  - Tries to run the highest prio tasks on all CPUs it can
- Can cause unexpected results for Round Robin

Round Robin



SCHED_DEADLINE

Earliest Deadline First

Guaranteed amount of CPU time per task

Relatively new in Linux

Questions?