Project 2 Help Document

Oct 15, 2020

DCSLab

SNU Operating Systems

What you did in Project 1

1. Build your kernel
2. Make image files
3. Connect your SD card to your computer
4. Flash your SD cards
5. Connect your SD card to your RPI3
6. Turn on your RPI3
7. Connect your RPI3 to your computer with UART cable
8. Use screen or putty or minicom to interact with RPI3

What you did in Project 1

1. Build your kernel
2. Make image files
3. Connect your SD card to your computer
4. Flash your SD cards
5. Connect your SD card to your RPI3
6. Turn on your RPI3
7. Connect your RPI3 to your computer with UART cable
8. Use screen or putty or minicom to interact with RPI3

What you did in Project 1

1. Build your kernel
2. Make image files
3. Emulate Tizen on RPI3 with QEMU
4. Play with QEMU

What is QEMU

QEMU is a generic and open source **machine emulator** and **virtualizer**.

How to use QEMU

1. Install QEMU
   a. sudo apt-get install qemu (or sudo apt-get install qemu-system-aarch64)

2. Update **/arch/arm64/configs/tizen_bcmrpi3_defconfig** file

3. Run Tizen on RPI3 emulation with QEMU

We prepared config file and qemu run script for you :)

Move files into Tizen when using QEMU

1. Mount `rootfs.img` on ${mnt_dir}
2. Move files under ${mnt_dir}/root/ (You may need sudo)
3. Unmount ${mnt_dir}

QEMU Troubleshooting

1. Ran Tizen on RPI3 with QEMU, but nothing shows up on Terminal.
    a. http://jake.dothome.co.kr/qemu/
        i. Refer *QEMU의 스탠다드 콘솔 출력문제*

So, let's start to look at Project 2 :)

Project 2 Overview

- Design and implement WRR (Weighted Round-Robin) scheduler
  - Define and implement a new scheduler
    - Implement load balancing mechanism
  - Examine the scheduler performance with `trial`
  - Improve the scheduler
    - Open question

WRR Scheduler

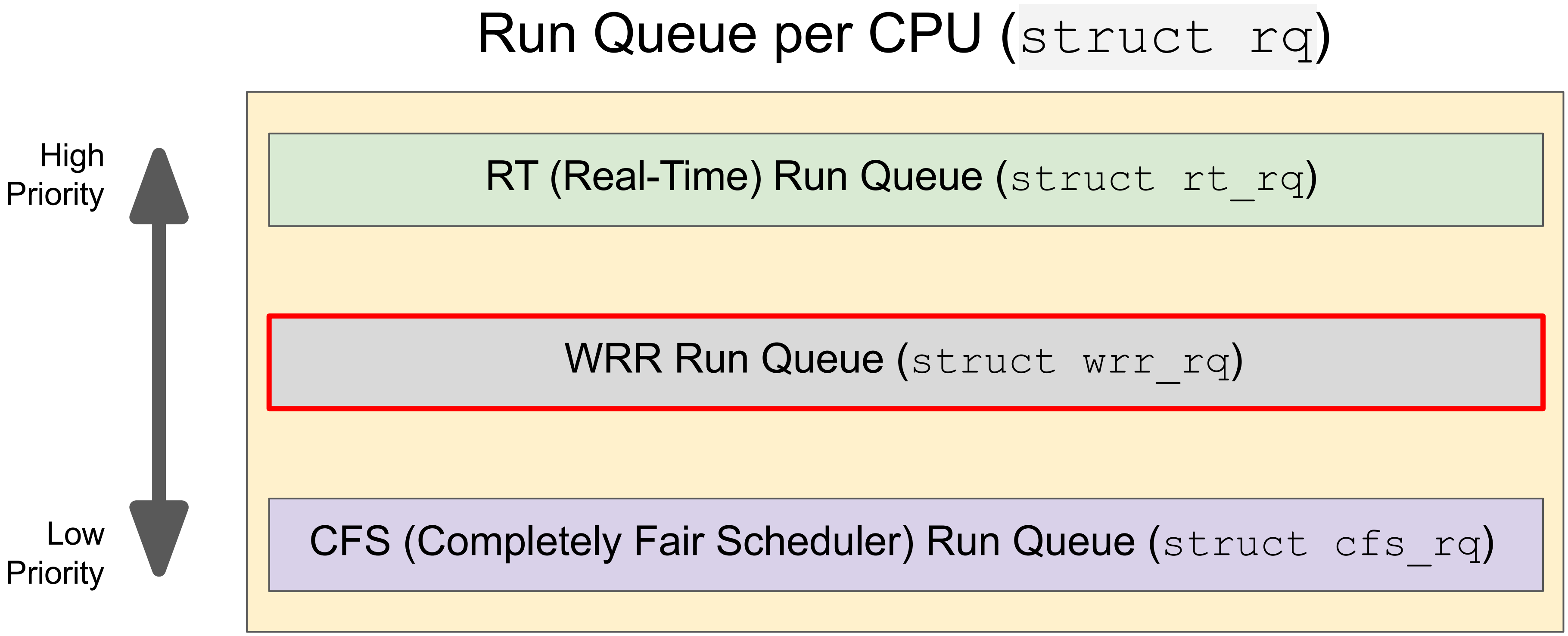
Linux Scheduler Basics

- Multi-level scheduling
  - Real-time tasks has priority over other tasks
- Real-time tasks: FCFS, RR, DL, …
- Other tasks: CFS
- Each CPU maintains separate run queues for tasks
  - To prevent contention while accessing run queue

WRR Scheduler

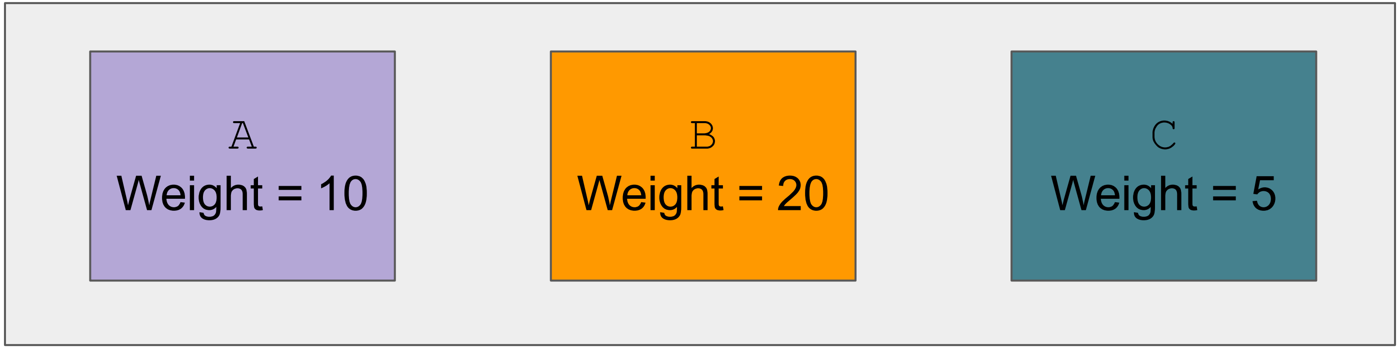
- Weighted Round-Robin Scheduler
- Tasks are executed in a round-robin fashion, but get different time slices according to their weights
  - Default weight is 10
  - Time slice = Weight * 10ms
- Priority: RT > WRR > CFS
- Load balancing

Multi-level Run Queue with WRR



WRR Scheduling Example

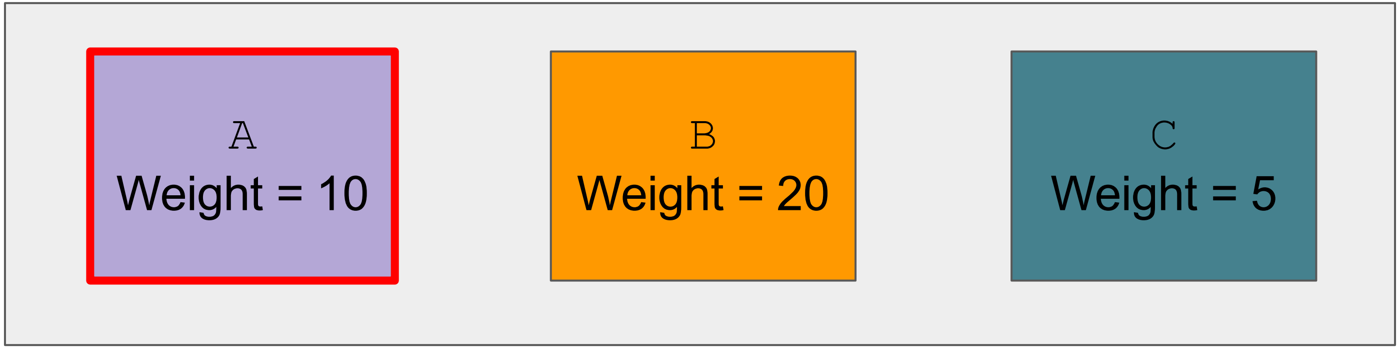
Three tasks currently in WRR run queue



WRR Scheduling Example

t = 0ms

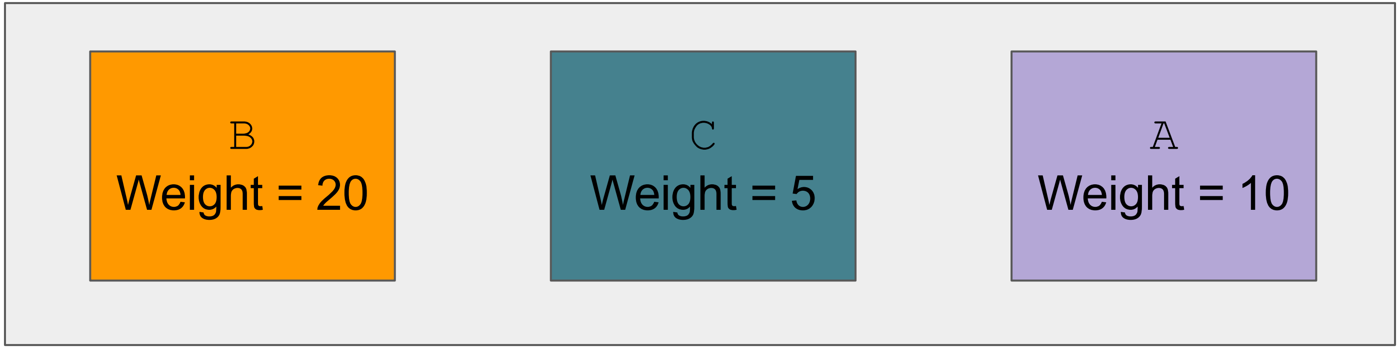
`A` starts running first



WRR Scheduling Example

t = 100ms (Δt = 100ms)

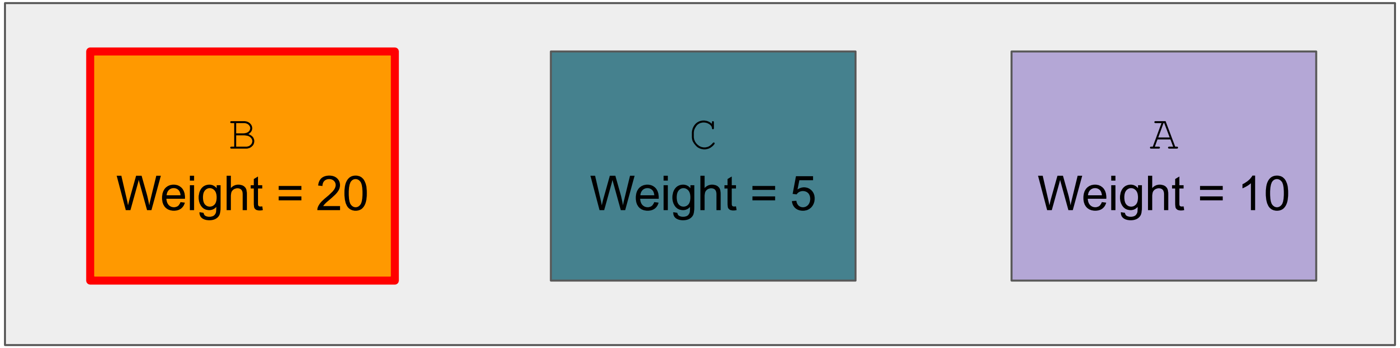
`A` stops, and is moved to the tail of the run queue because the task is not finished…



WRR Scheduling Example

t = 100ms

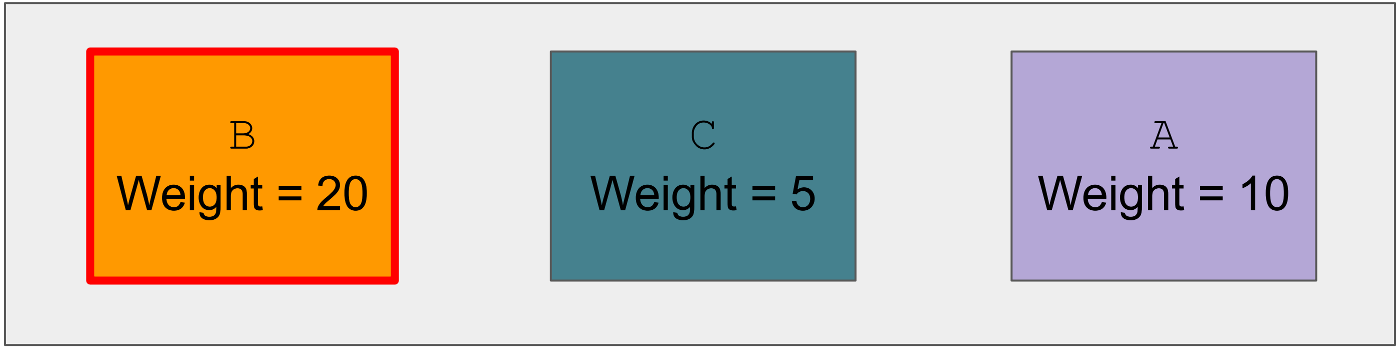
… and the next task (`B`) starts running



WRR Scheduling Example

t = 200ms (Δt = 100ms)

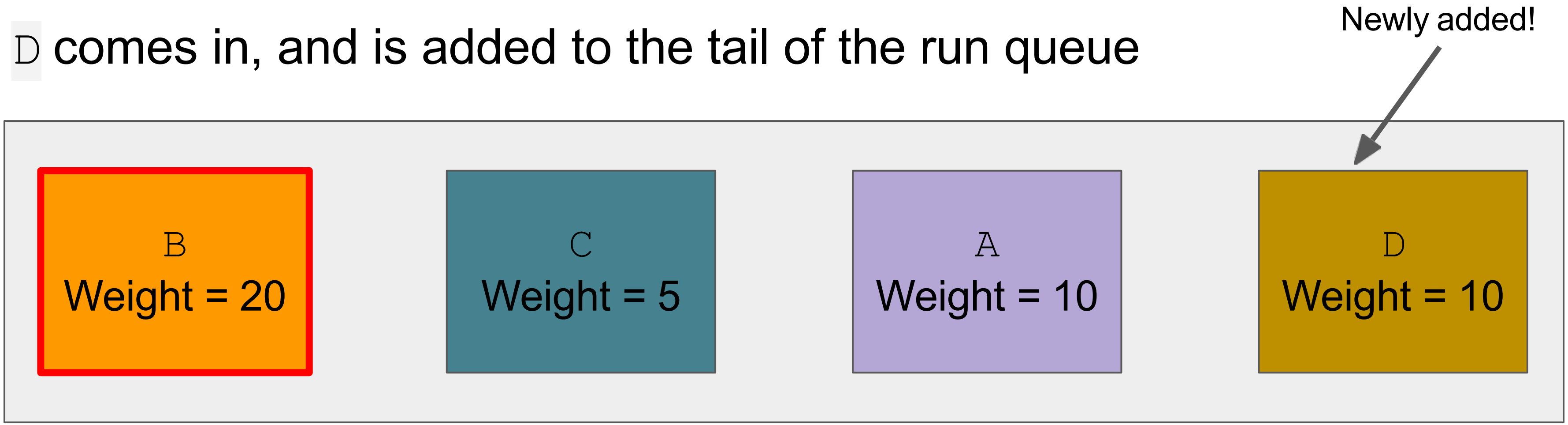
`B` is still running, because its time slice is 200ms



WRR Scheduling Example

t = 250ms (Δt = 50ms)

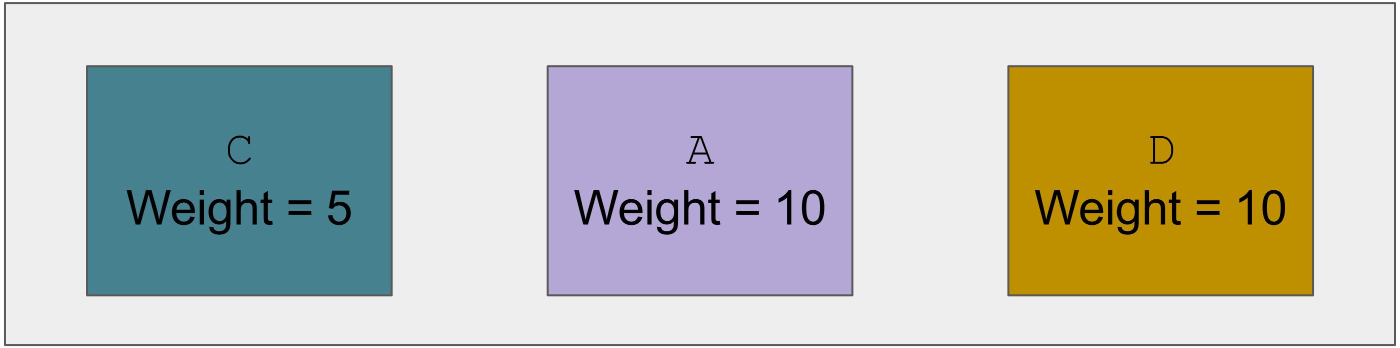
`D` comes in, and is added to the tail of the run queue



WRR Scheduling Example

t = 280ms (Δt = 30ms)

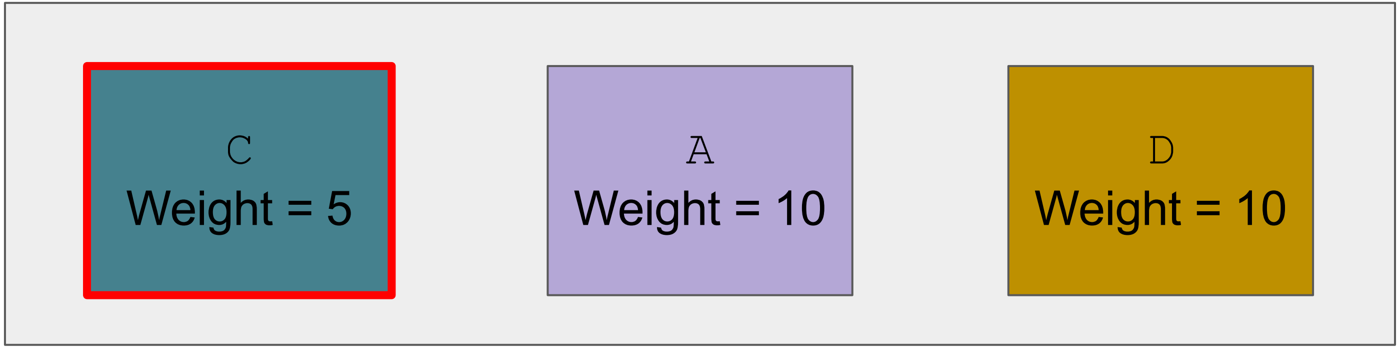
`B` has finished its work and is terminated; now removed from the run queue…



WRR Scheduling Example

t = 280ms

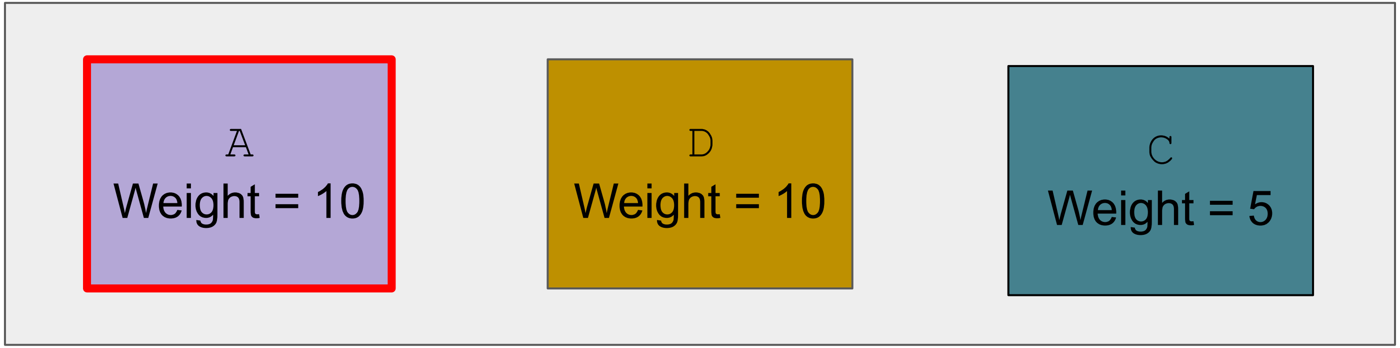
… and `C` starts running



WRR Scheduling Example

t = 330ms (Δt = 50ms)

`C` is stopped and is moved to the tail. `A` starts running again



Load Balancing

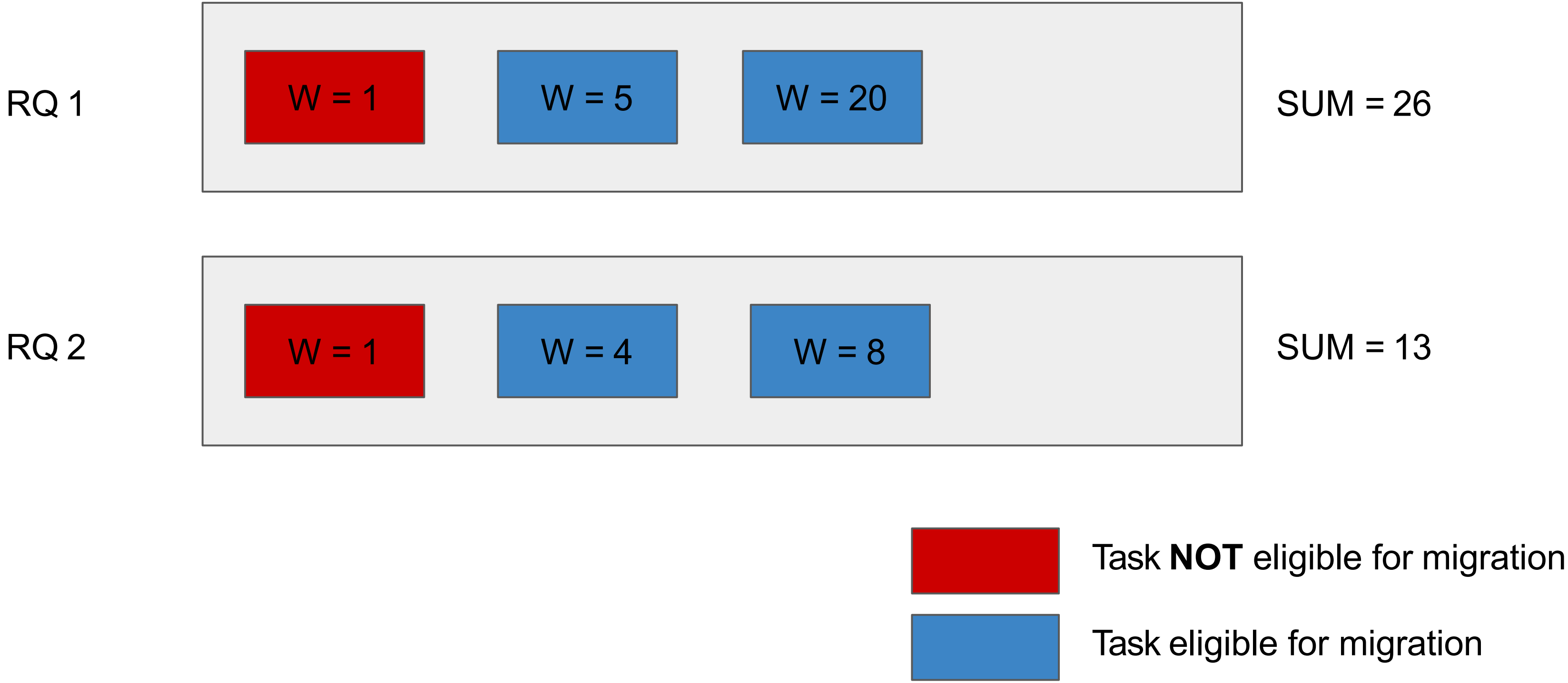
- Balance load among the run queue of each CPU
- Make sure that it only works when more than one CPU is active
  - CPU hotplug
  - `for_each_online_cpu(cpu)`
- **Leave one run queue empty!**
- Should be attempted every 2000ms

Load Balancing Algorithm

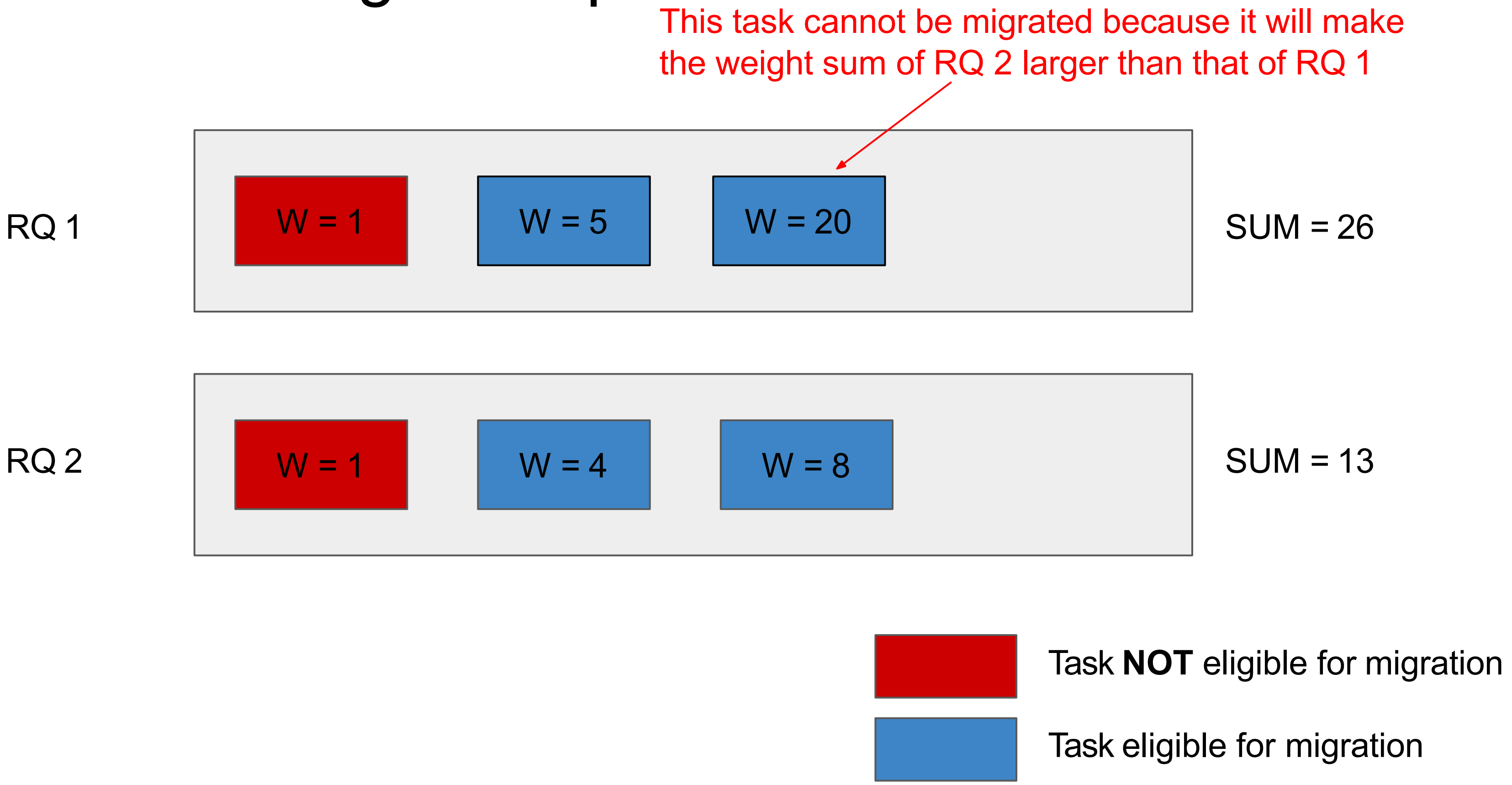
- Pick two run queues with the minimum weight sum and the maximum weight sum
  - Call them `RQ_MIN` and `RQ_MAX` respectively
- Pick a task with the largest weight among tasks that satisfy the following conditions:
  - The picked task should be able to be migrated to `RQ_MIN`
  - Migration should not cause weight of `RQ_MIN` to become **bigger than or equal** to `RQ_MAX`
  - Tasks currently running are not eligible for migration
- Migrate if an eligible task exists
  - There may be no eligible task

Load Balancing Example

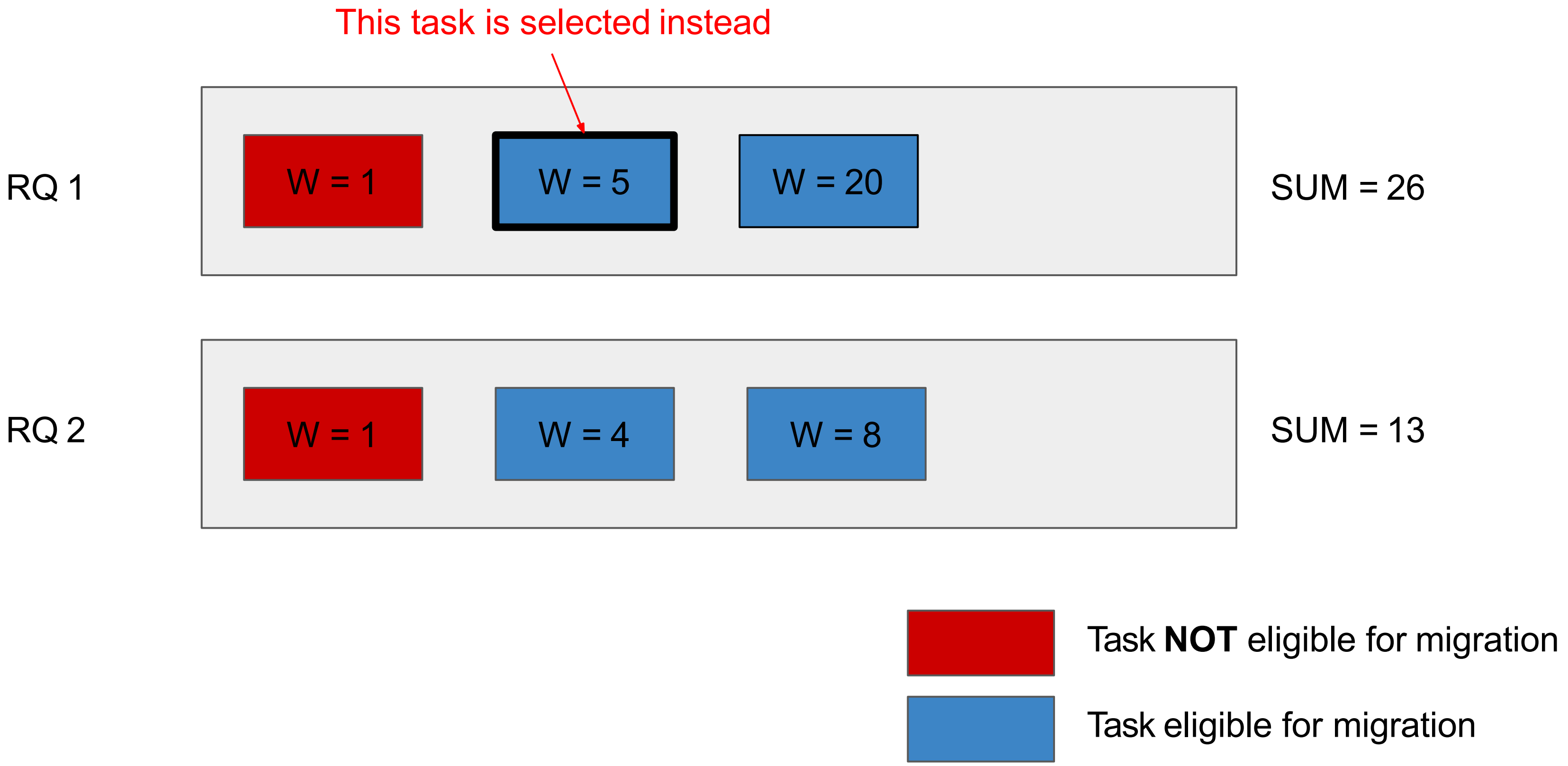
Migrating a task from RQ 1 to RQ 2



Load Balancing Example

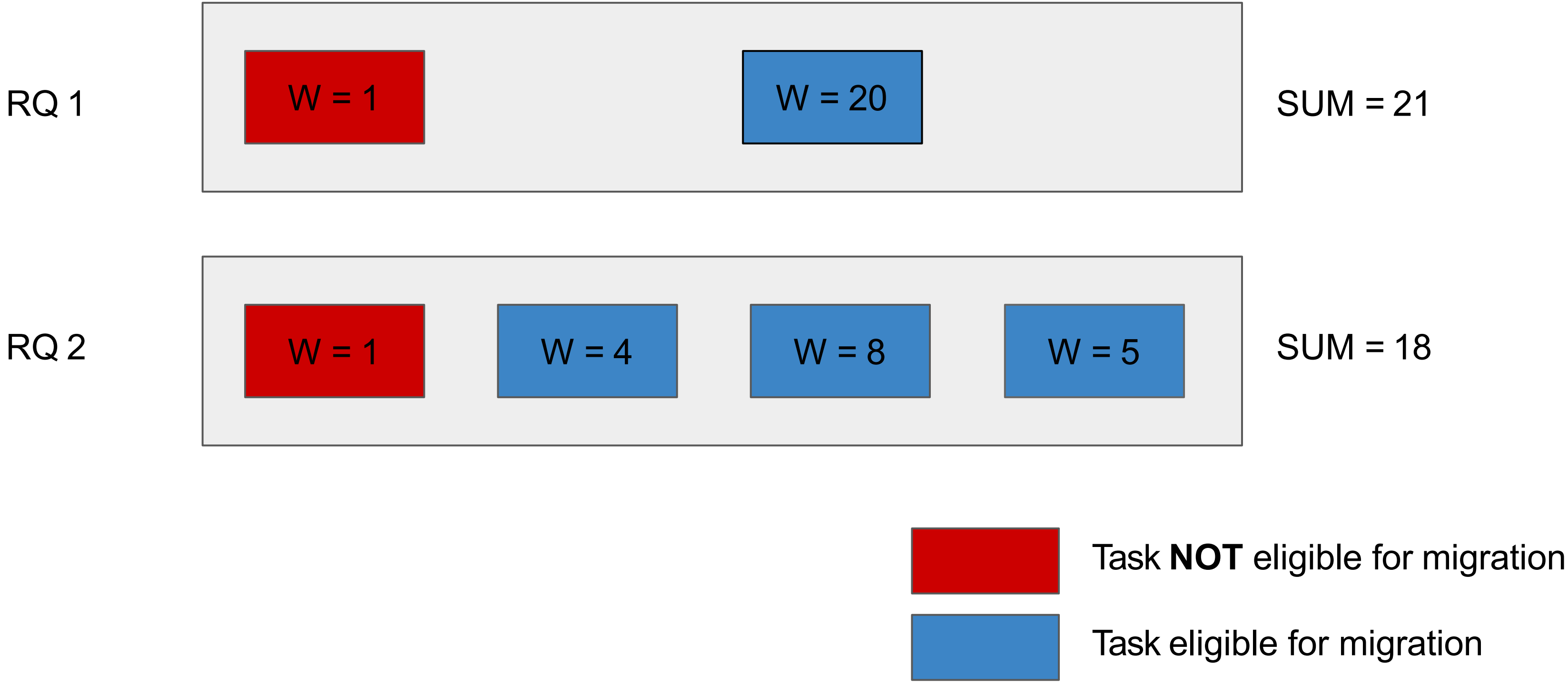


Load Balancing Example



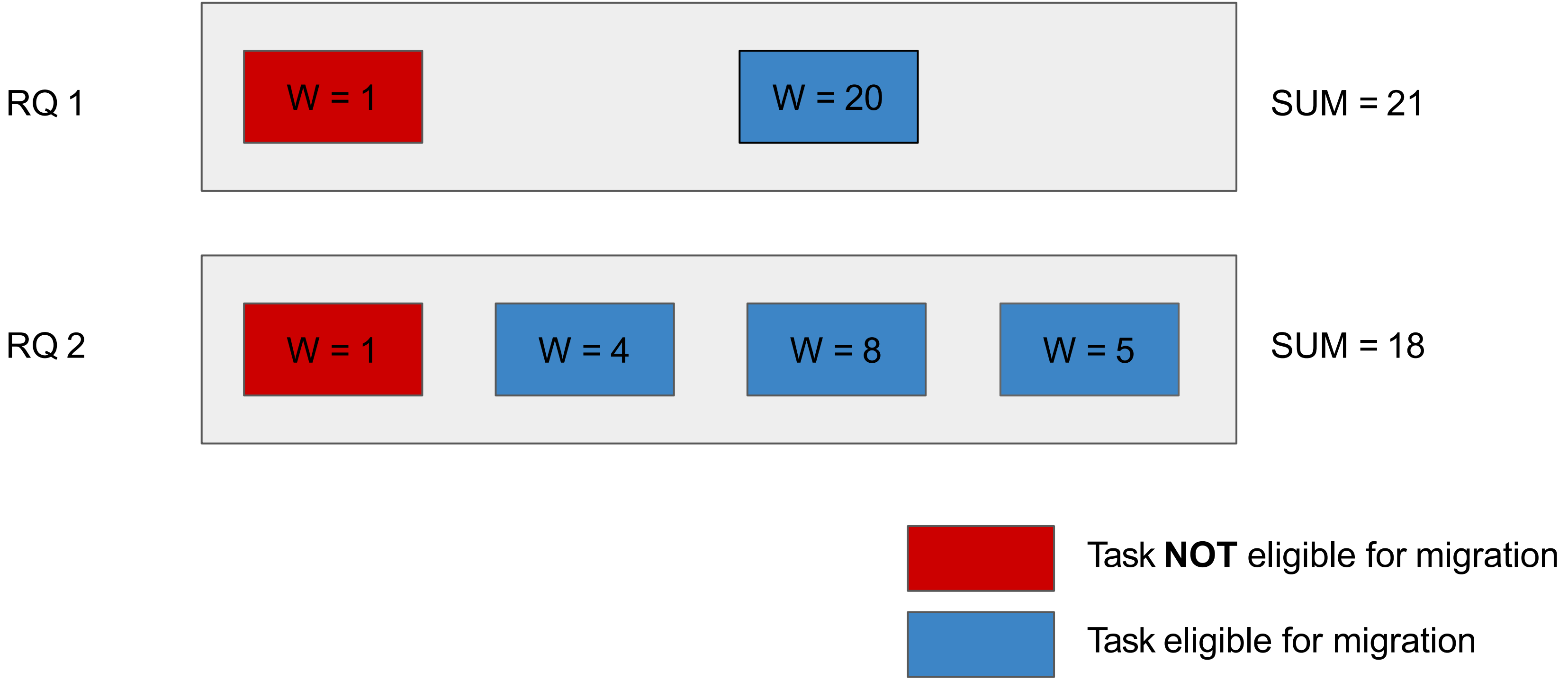
Load Balancing Example

After migration

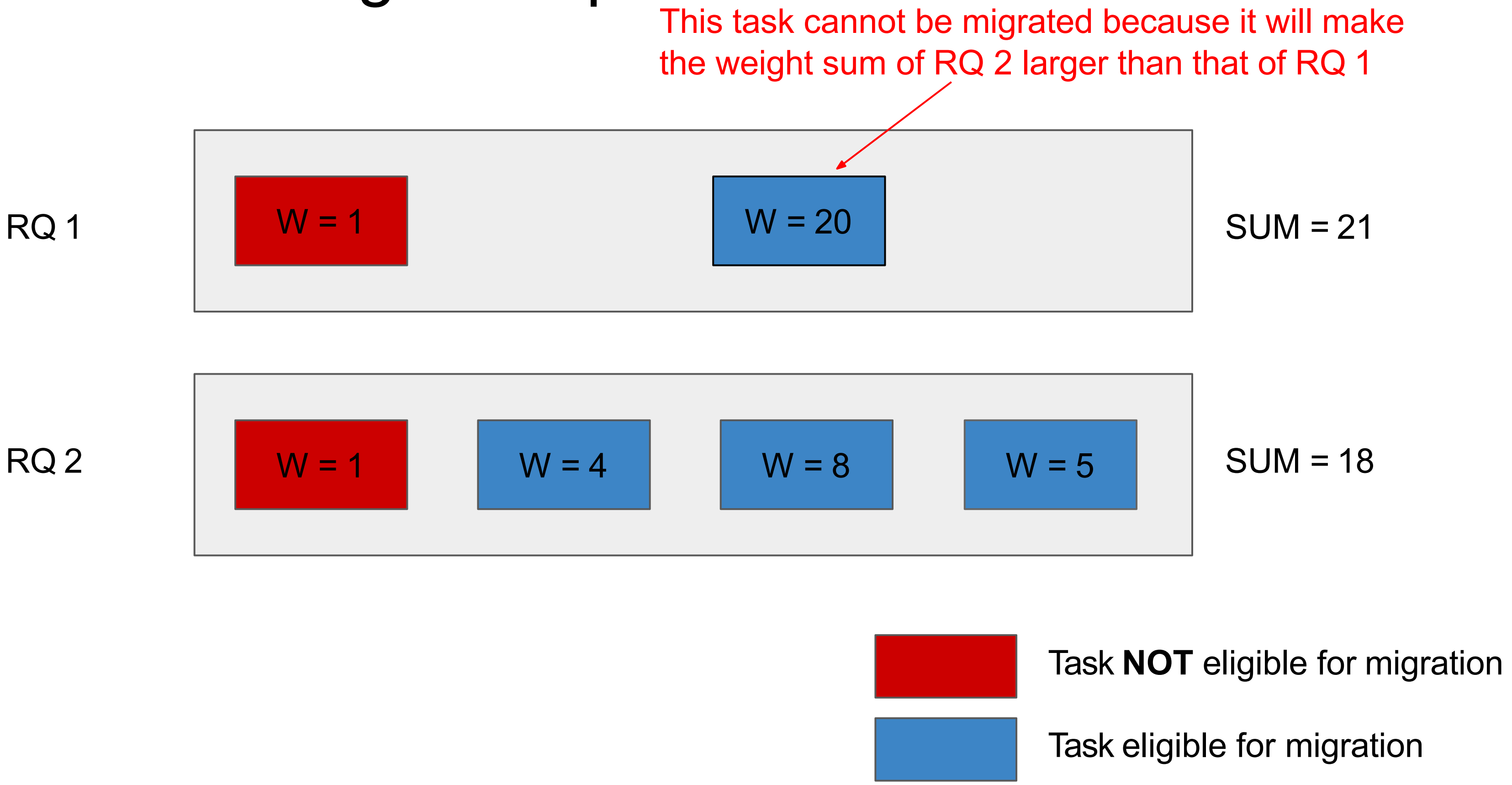


Load Balancing Example

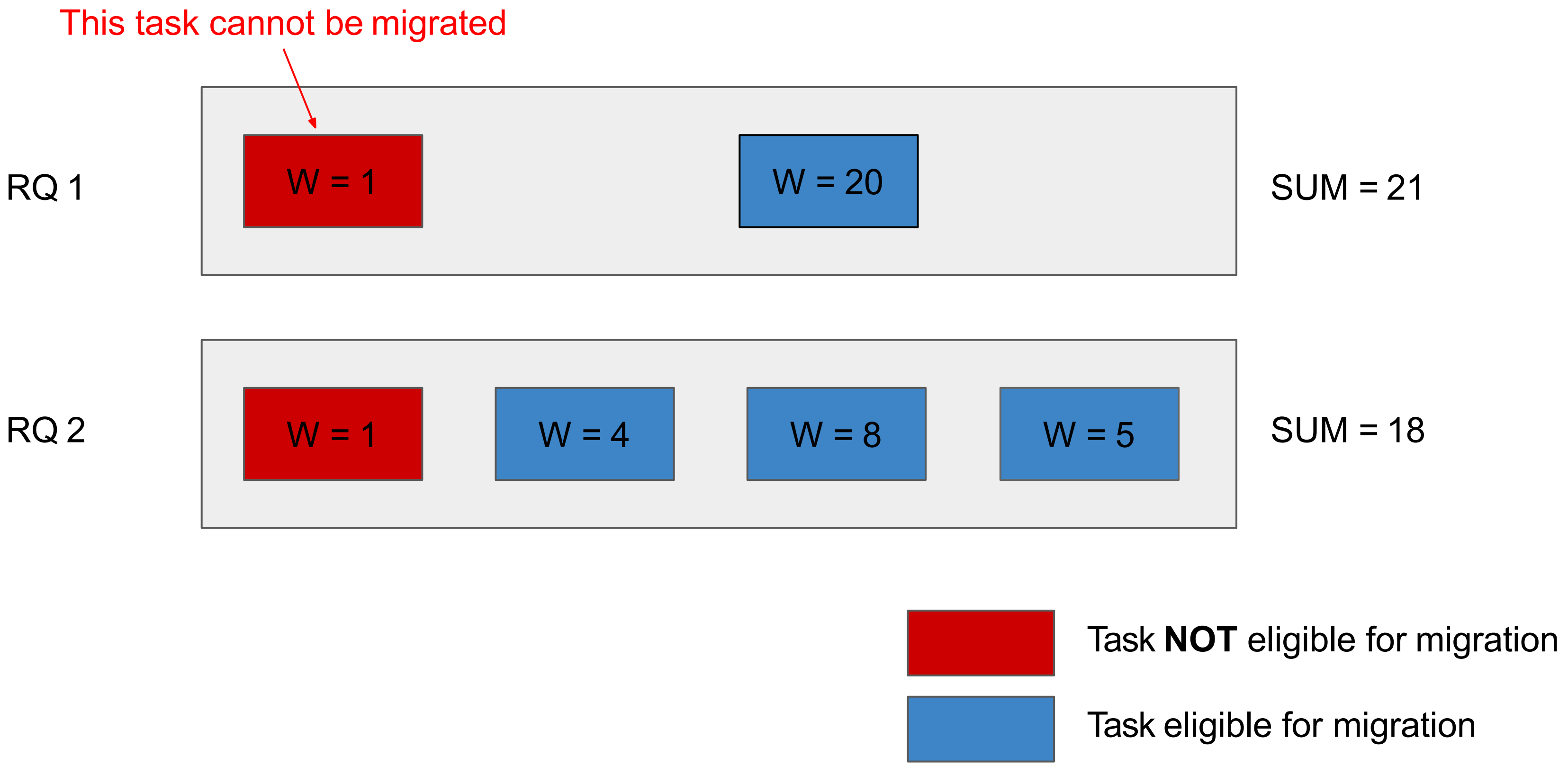
Migrating a task from RQ 1 to RQ 2 again



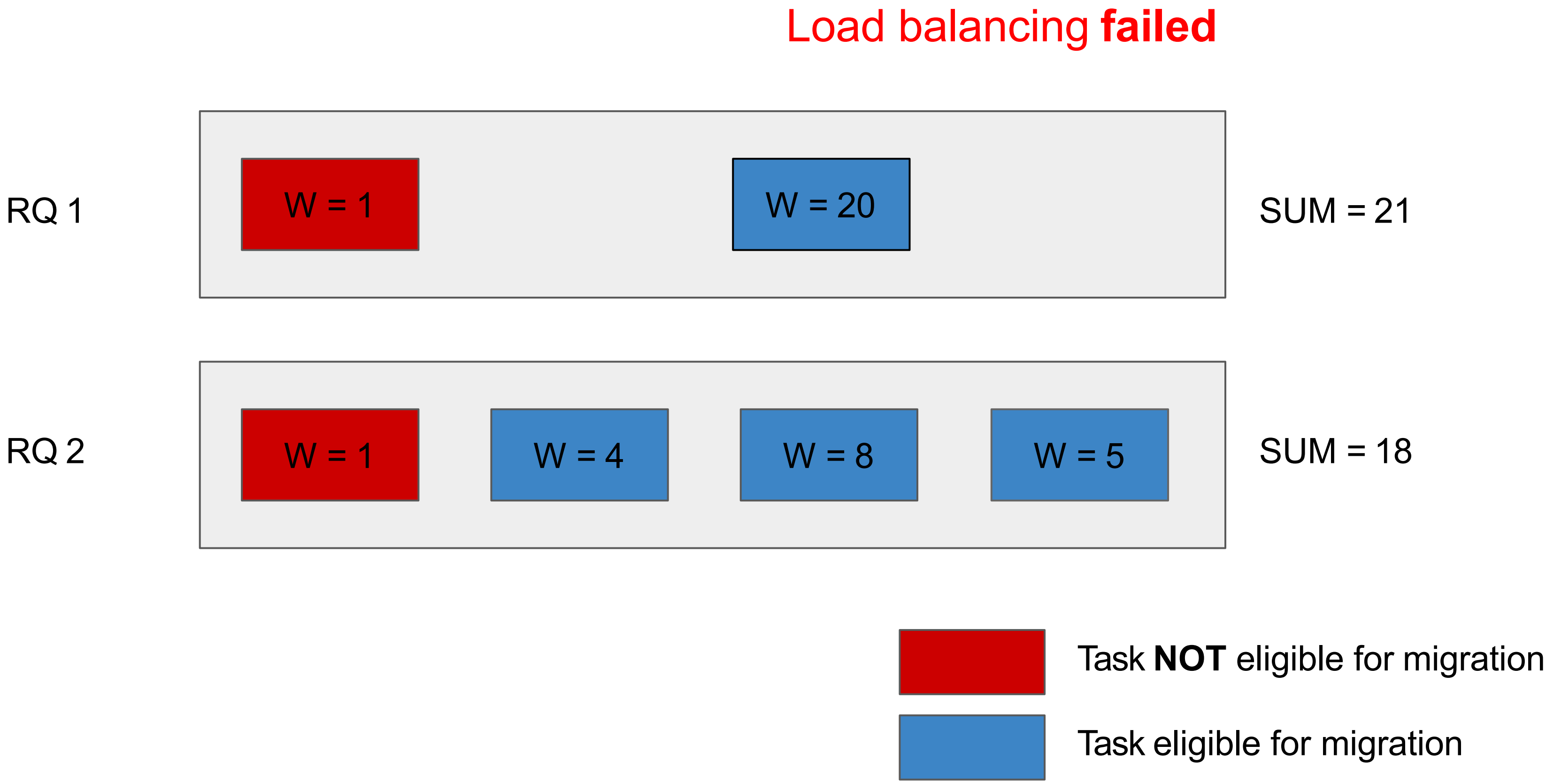
Load Balancing Example



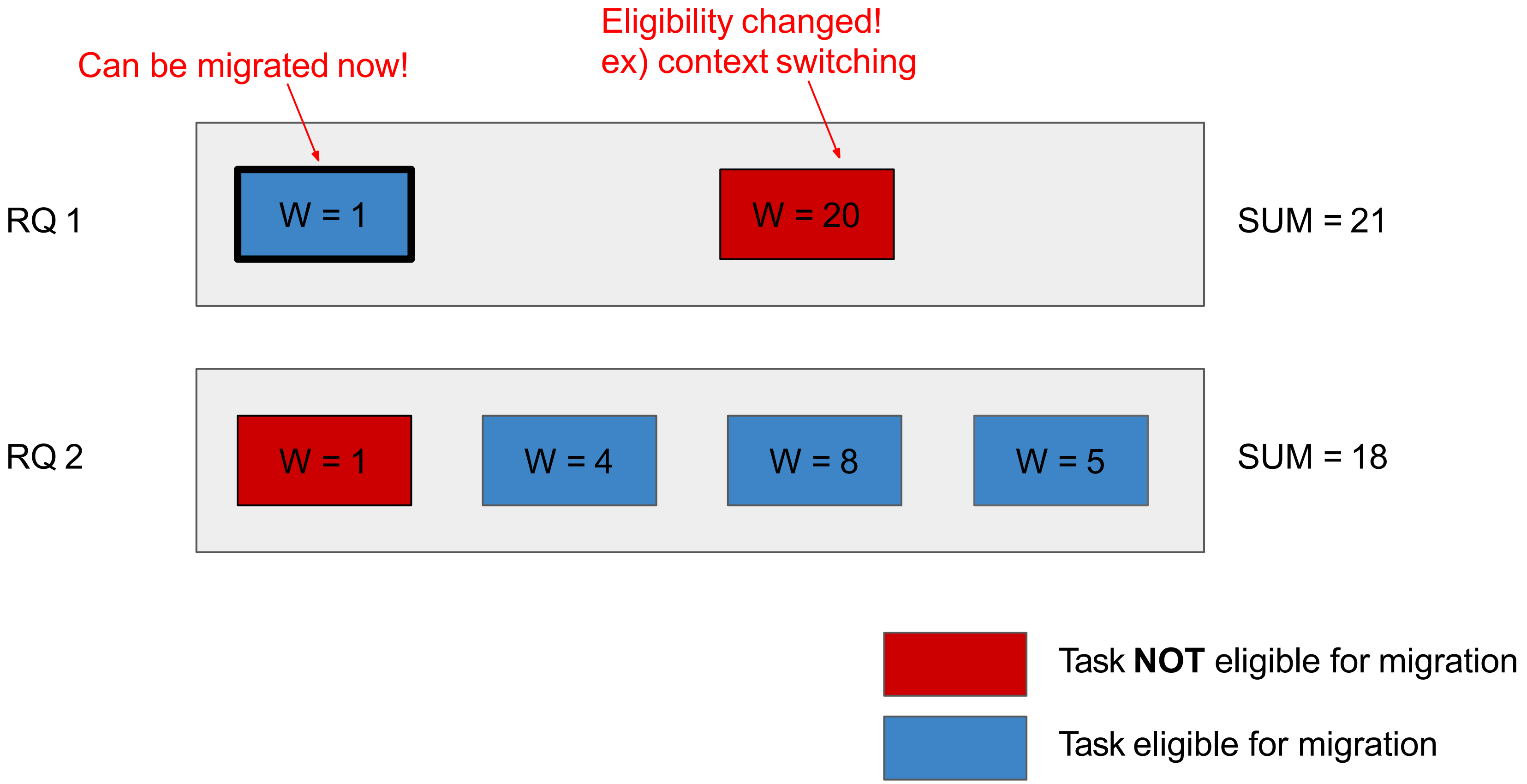
Load Balancing Example



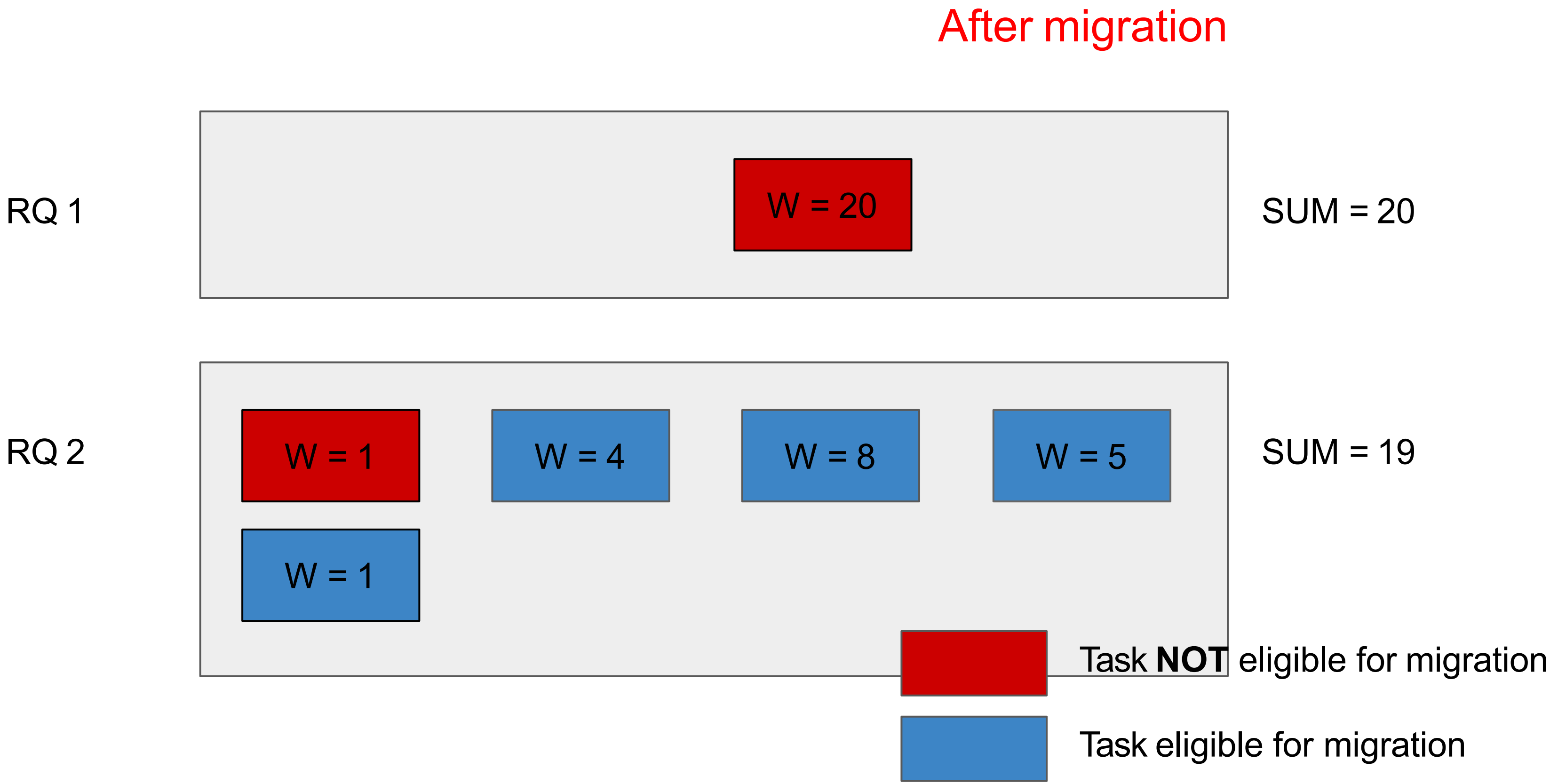
Load Balancing Example



Load Balancing Example



Load Balancing Example



Scheduler Implementation

Preliminaries

- Modify `arch/arm64/configs/tizen_bcmrpi3_defconfig`
  - `CONFIG_SCHED_DEBUG=Y`
    - You need this option to debug your scheduler
    - Possible performance degradation
  - (Optional) Enable `CONFIG_SCHEDSTATS` for more detailed debugging

Implementation Overview (1)

- Define necessary constants and data structures
  - `include/linux/sched.h`
  - `include/uapi/linux/sched.h`
  - ...
- Register a new scheduler class for WRR and implement necessary functions in **`kernel/sched/wrr.c`**
- Modify **`kernel/sched/debug.c`** to print additional necessary information about your scheduler
  - Optionally `kernel/sched/stats.c` too

Implementation Overview (2)

- Modify **`kernel/sched/core.c`** to support WRR
  - Trigger load balancing function, …
  - You might need to register some function signatures in `kernel/sched/sched.h`
- Implement necessary system calls
  - `sched_setweight`, `sched_getweight`
- Check that your scheduler is working with `sched_setscheduler`
  - One CPU run queue empty
  - Load balancing

Constants & Data Structures

- Define `SCHED_WRR` as **7**
  - `include/`**`uapi`**`/linux/sched.h`
- Define fields for WRR scheduler in `struct task_struct`
  - See how other schedulers like RT, CFS, … are implemented
  - `list_head` for WRR run queue
  - Weight, time slice, …
- Define a run queue for tasks under WRR scheduler
  - `struct rq` also needs information about WRR run queue
    - `struct rq`: CPU run queue
  - What kind of information should be stored here?
  - Should this have a locking mechanism?

Registering Scheduler

- Declare and define **`wrr_sched_class`** in `kernel/sched/sched.h` and in `kernel/sched/wrr.c`
  - Take a look at `kernel/sched/fair.c` & `kernel/sched/rt.c`
  - The next scheduler class (priority-wise) should be `fair_sched_class`
  - Similarly, the next scheduler class of `rt_sched_class` should be `wrr_sched_class`
- Implement necessary functions for `wrr_sched_class`
  - `enqueue_task`, `dequeue_task`, …
  - You don’t need to implement all the functions
- Define other necessary functions for load balancing or debugging

Modifying `kernel/sched/core.c`

- Problem: it assumes that there are only classes predefined in the kernel, such as `rt_sched_class`, `fair_sched_class`, ...
- We need to make sure that they are aware of `wrr_sched_class` too!
  - Initialize WRR run queue
  - Make `SCHED_WRR` policy valid
  - Manage forked tasks
    - The child should follow the same scheduler policy of its parent
  - ...

Debugging

- Reminder: You should turn on `CONFIG_SCHED_DEBUG`
- You might want to modify `kernel/sched/debug.c` to check whether your WRR scheduler works properly or not
- Scheduling information is written to `/proc/sched_debug`

System Calls

- You all know how to implement system calls!
- Authentication is important in `sched_setweight`
  - Increasing weight: administrator only
  - Decreasing weight: process owner & administrator only
  - Check uid and euid
- Nothing hard here :)

Load Balancing (1)

- How do I check the remaining time slice or figure out when to trigger load balancing?
- `scheduler_tick`
  - `kernel/sched/core.c`
  - Called every tick
- Tick frequency: HZ
  - A macro which represents the number of ticks in a second

Load Balancing (2)

- How do I check the remaining time slice or figure out when to trigger load balancing? (cont'd)
- `scheduler_tick`
- Tick frequency: HZ
- `jiffies`
    - A global variable containing the number of ticks after system boot
    - unsigned long - beware of overflow!
    - There are macros for comparing time
        - `time_after`, `time_before`, `time_after_eq`, `time_before_eq`
    - More things: http://www.makelinux.net/ldd3/chp-7.shtml

Load Balancing (3)

- How do I determine if a task can be migrated?
- Tasks that are currently running cannot be migrated
- Some tasks may have some restrictions on cores they can run on
  - How can we know it? / Why do they have restrictions?
  - Refer to existing load balancing code to find the answer

Load Balancing (4)

- How do I prevent race condition while load balancing?
- `scheduler_tick` is called for every available CPU!
  - You need to make sure that only one thread is working on load balancing at any time!
- One seemingly simple & plausible solution
  - Make only a certain CPU can do load balancing
  - But, because `CONFIG_HOTPLUG_CPU` is on by default, the designated CPU could be turned off anytime…
  - What happens if the designated CPU is turned off? How can we prevent it?
- Think carefully about synchronization issues and CPU hotplug!

Experiment

- Main question: how the weight affects the performance
- Measure the time for `trial` to finish for varying:
  - weights
  - number of processes
  - …
- You should make sure that all cores (except the one that should be left empty) are active when you start your experiment!
  - Initially, it is very likely that only one core is active
  - You can make a number of processes run for some time to make all cores active

More Things...

- CFS is highly optimized, while your scheduler is not: slow!
  - When the shell is not responding, just wait for a while
  - Do not create way too many processes at once (ex: 100 `fork`s)
  - Always leave one core empty (no WRR tasks running)
- `rcu_read_lock` when iterating over CPU cores
- This is the hardest project so far, and the only project that you may not be able to finish on time, so start early!

About Submission (IMPORTANT!)

- **Due: 2020-11-05 Thursday 17:00:00 KST**
- Make sure your branch name is *proj3*
- Don't be late!
  - We will not grade the commits after the deadline
- Slides and Demo
  - Send it to the TA (**os-tas@dcslab.snu.ac.kr**) before the deadline
  - Title: **[OS-ProjX] TeamX slides&demo submission**
  - File name: **TeamX-slides.pptx(.ppt, .pdf)**, **TeamX-demo.mp4(.avi, …)**
- Please submit only one video

Q &A