Workqueue insights

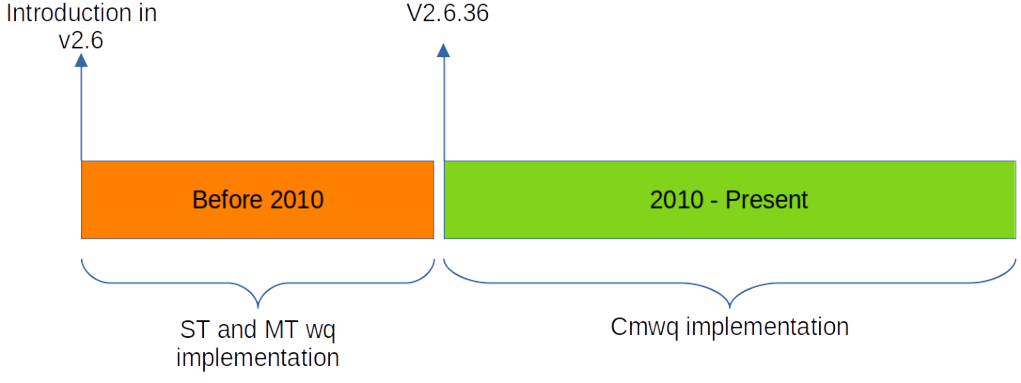
Prathu Baronia & Neeraj Upadhyay

29-07-2023

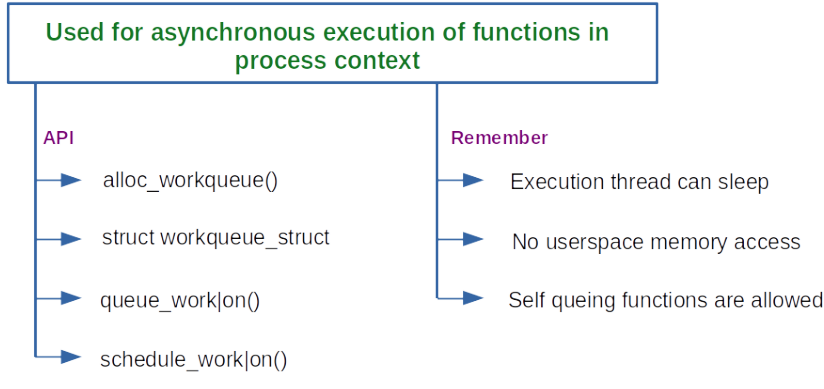
Before we begin

- wq: workqueue
- cmwq: concurrency managed workqueue
- Kernel version considered: v6.5-rc3

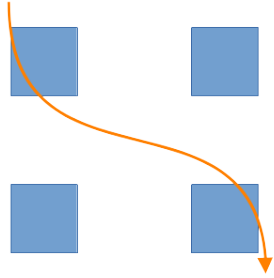
History


Introduction in v2.6
V2.6.36
Before 2010
2010 - Present
ST and MT wq implementation
Cmwq implementation

WQ API


Used for asynchronous execution of functions in process context
API
alloc_workqueue()
struct workqueue_struct
queue_work|on()
schedule_work|on()
Remember
Execution thread can sleep
No userspace memory access
Self queing functions are allowed

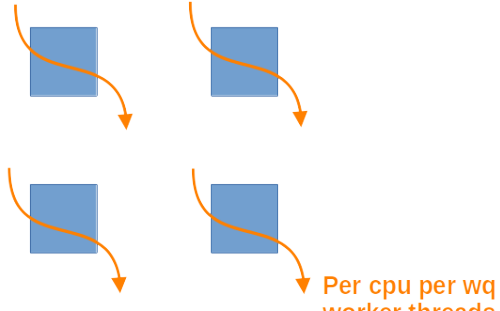
Legacy implementation: ST



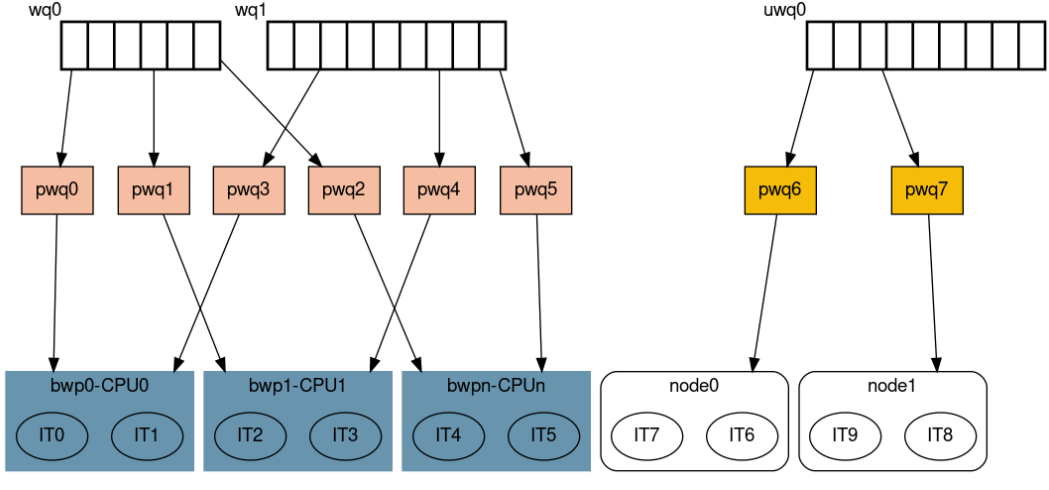
ST wq
implementation

System wide
per wq worker
thread

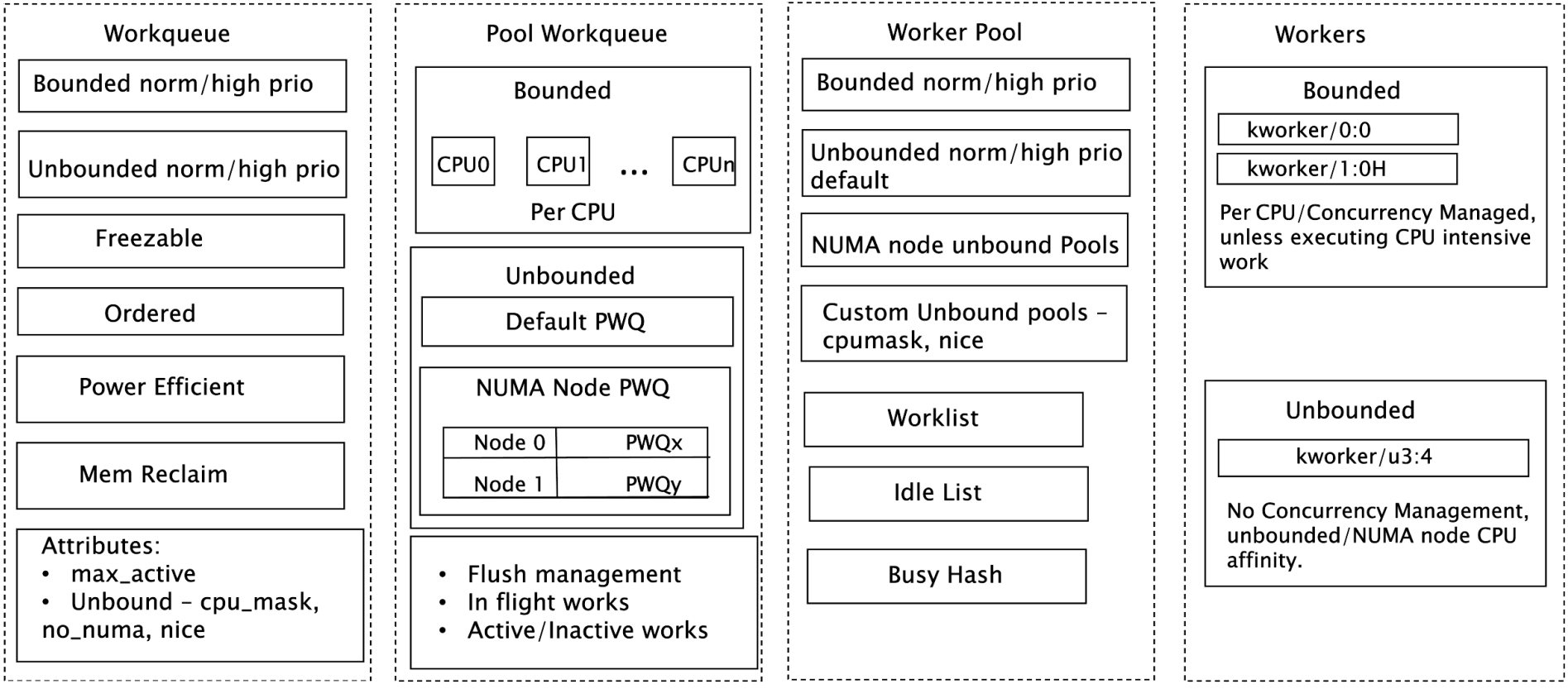
Legacy implementation: MT



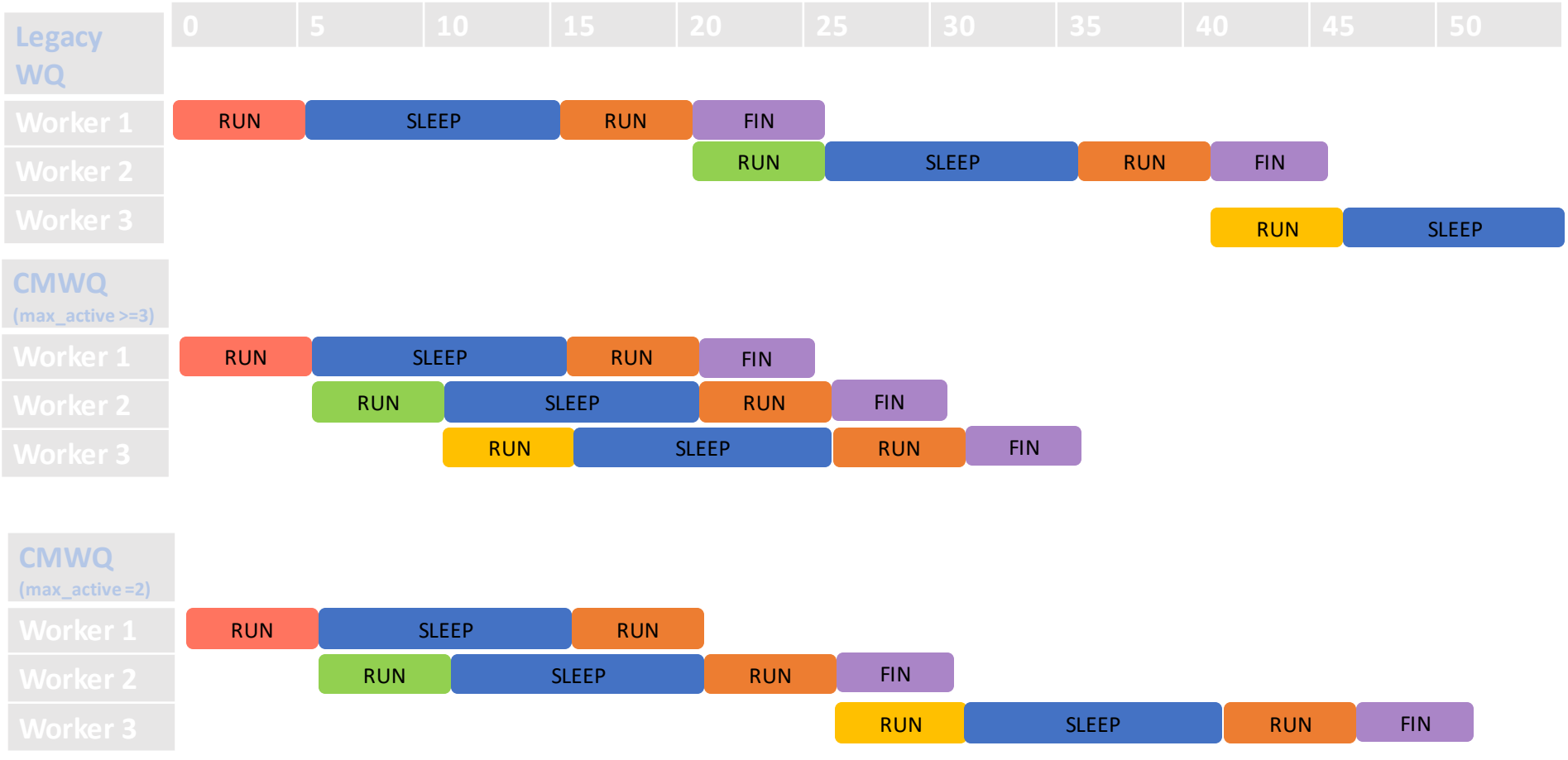
cmwq

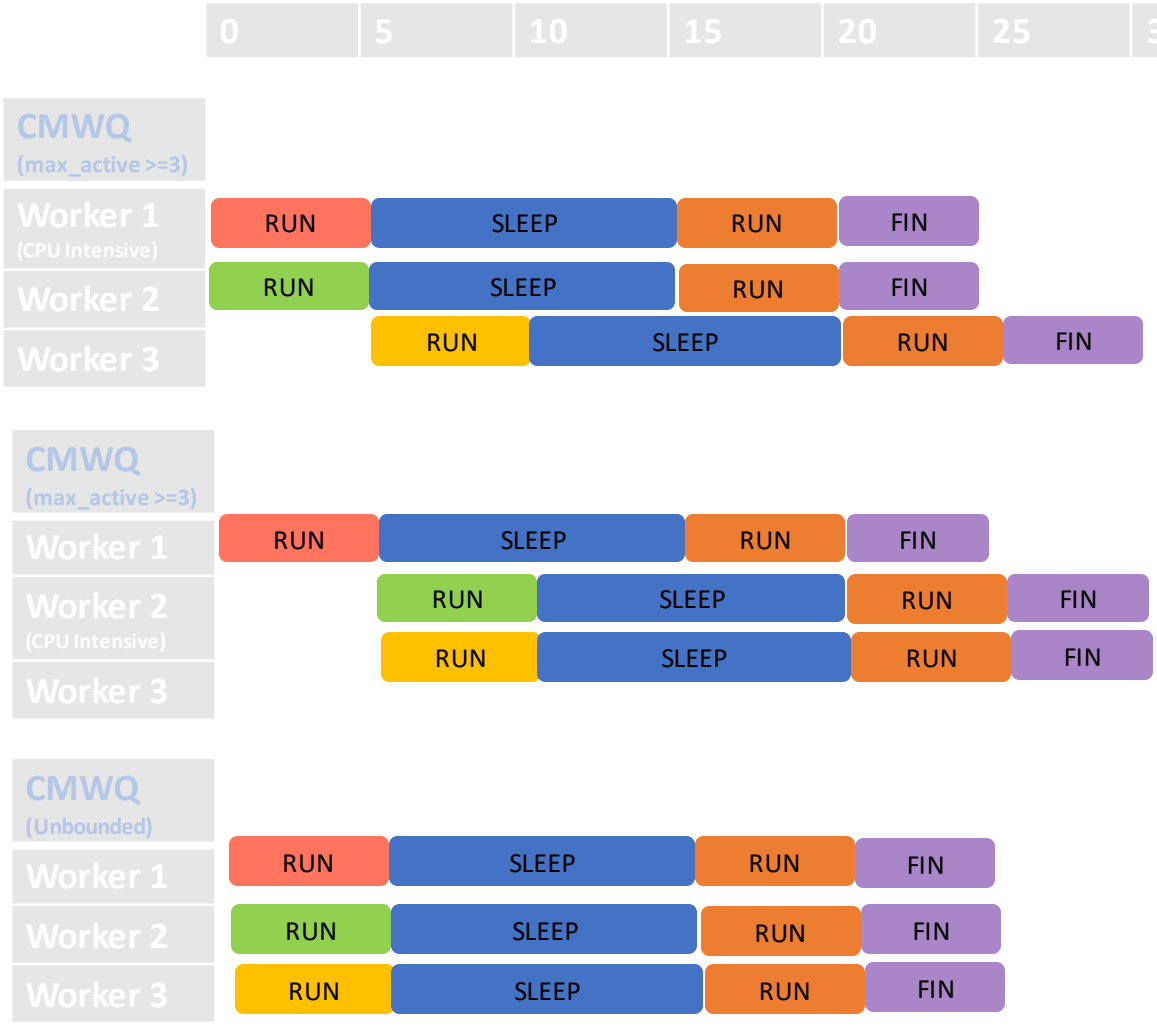

wq0
wq1
uwq0
pwq0
pwq1
pwq3
pwq2
pwq4
pwq5
pwq6
pwq7
bwp0-CPU0
bwp1-CPU1
bwpn-CPUn
node0
node1
IT0
IT1
IT2
IT3
IT4
IT5
IT7
IT6
IT9
IT8

Workqueue software layers

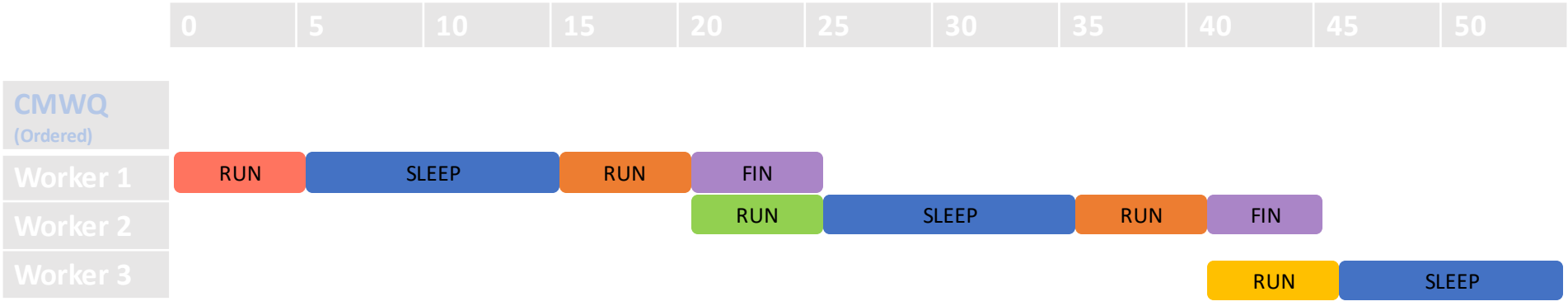


WQ EXECUTION SCENARIOS

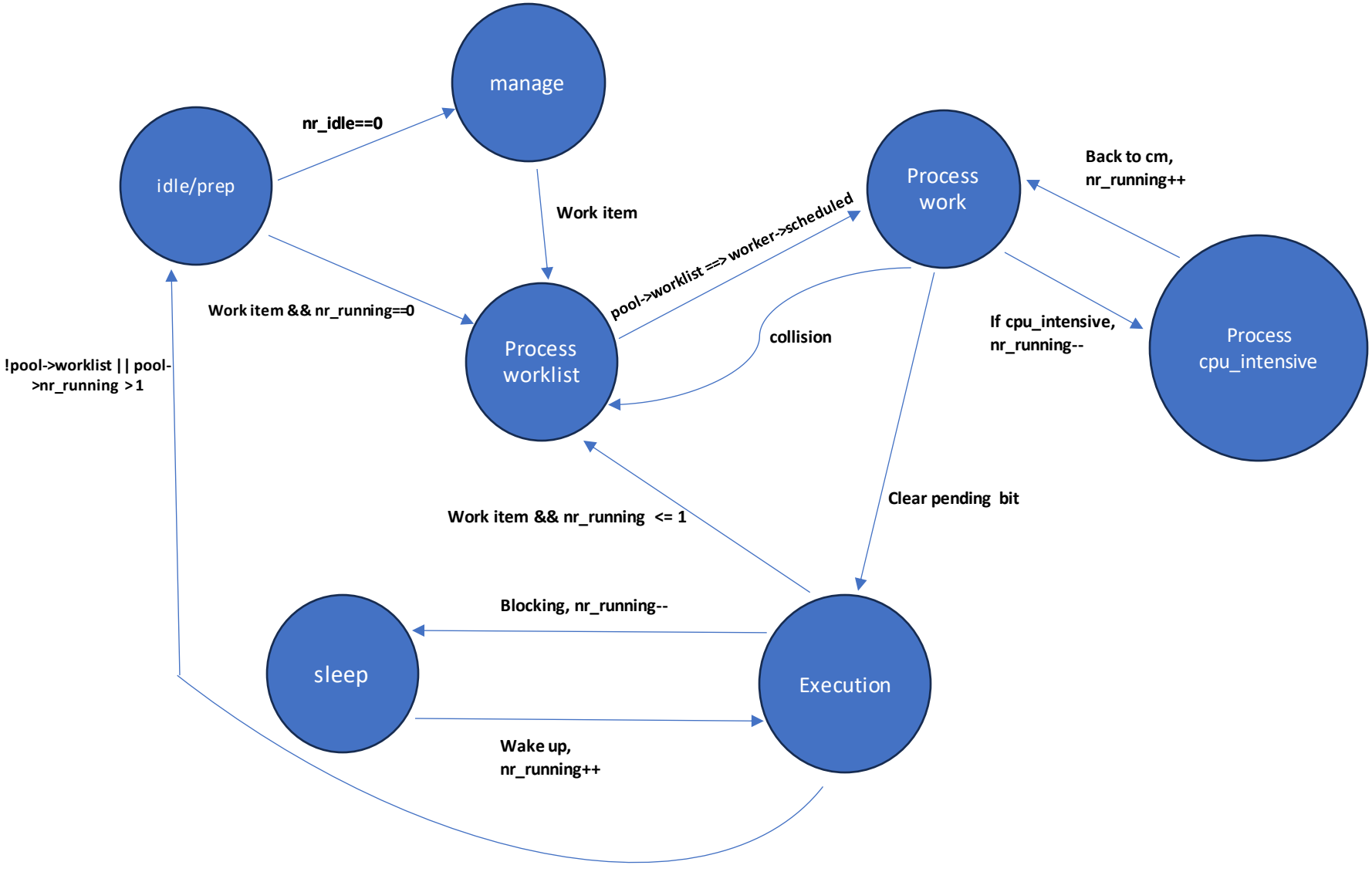


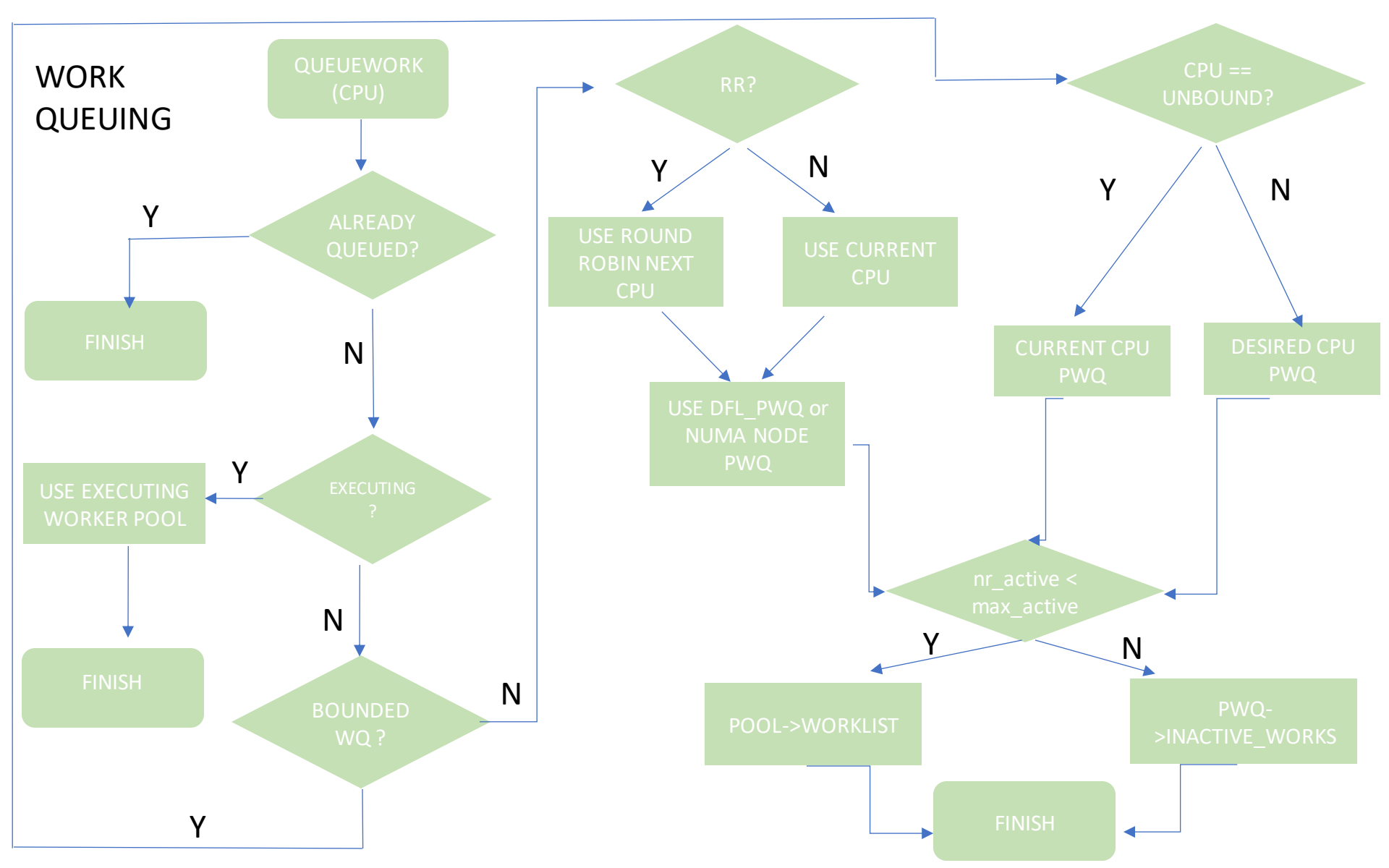


WQ EXECUTION SCENARIOS CONTD.



WQ EXECUTION SCENARIOS CONTD.


manage
idle/prep
nr_idle==0
Work item
Process work
Back to cm, nr_running++
Work item && nr_running==0
pool->worklist ==> worker->scheduled
Process worklist
collision
If cpu_intensive, nr_running--
Process cpu_intensive
!pool->worklist || pool->nr_running >1
Work item && nr_running <= 1
Clear pending bit
Blocking, nr_running--
sleep
Execution
Wake up, nr_running++


WORK QUEUING
QUEUEWORK (CPU)
ALREADY QUEUED?
Y
FINISH
N
EXECUTING ?
Y
USE EXECUTING WORKER POOL
FINISH
N
BOUNDED WQ ?
N
Y
RR?
Y
USE ROUND ROBIN NEXT CPU
N
USE CURRENT CPU
USE DFL_PWQ or NUMA NODE PWQ
CPU == UNBOUND?
Y
CURRENT CPU PWQ
N
DESIRED CPU PWQ
nr_active < max_active
Y
POOL->WORKLIST
N
PWQ->INACTIVE_WORKS
FINISH

System wide wqs

- `system_wq`
- `system_highpri_wq`
- `system_long_wq`
- `system_unbound_wq`
- `system_freezable_wq`
- `system_power_efficient_wq`
- `system_freezable_power_efficient_wq`

FAQs

- Which system wq to use in my code?
- When to use unbounded wqs?
- How to decide `max_active`?

Usecases of system wide wqs

- system_wq:
  - kernel/smp.c: smp_call_on_cpu()

```
queue_work_on(cpu, system_wq, &sscs.work);
```

- system_highpri_wq:
  - drivers/gpu/drm/radeon/radeon_display.c

```
radeon_x->flip_queue = alloc_workqueue("radeon-x", WQ_HIGHPRI, 0);
```

- system_long_wq:
  - drivers/base/core.c: devlink_dev_release()

```
queue_work(system_long_wq, &link->rm_work);
```

Usecases of system wide wqs cont.

- system_unbound_wq:
  - drivers/base/dd.c: driver_deferred_probe_trigger()

```
queue_work(system_unbound_wq, &deferred_probe_work);
```

- system_freezable_wq:
  - drivers/virtio/virtio_balloon.c: update_balloon_size_func()

```
queue_work(system_freezable_wq, work);
```

- system_power_efficient_wq:
  - sound/soc/codecs/rt5645.c: rt5645_irq() irq handler

```
queue_delayed_work(system_power_efficient_wq,
            &rt5645->jack_detect_work, msecs_to_jiffies(250));
```