Linaro CONNECT
ASIA 2013 (LCA13)

Scheduler training

Power Management Working Group - v0

Vincent Guittot / Morten Rasmussen

Content

- Topology
  - sched_domain
- Scheduler class
- CFS
- vruntime
- Nice priority
- Load Balance
  - Wake up
  - Newly Idle LB
  - Periodic LB
  - Idle LB
- Load average

Topology: How to describe a system ?

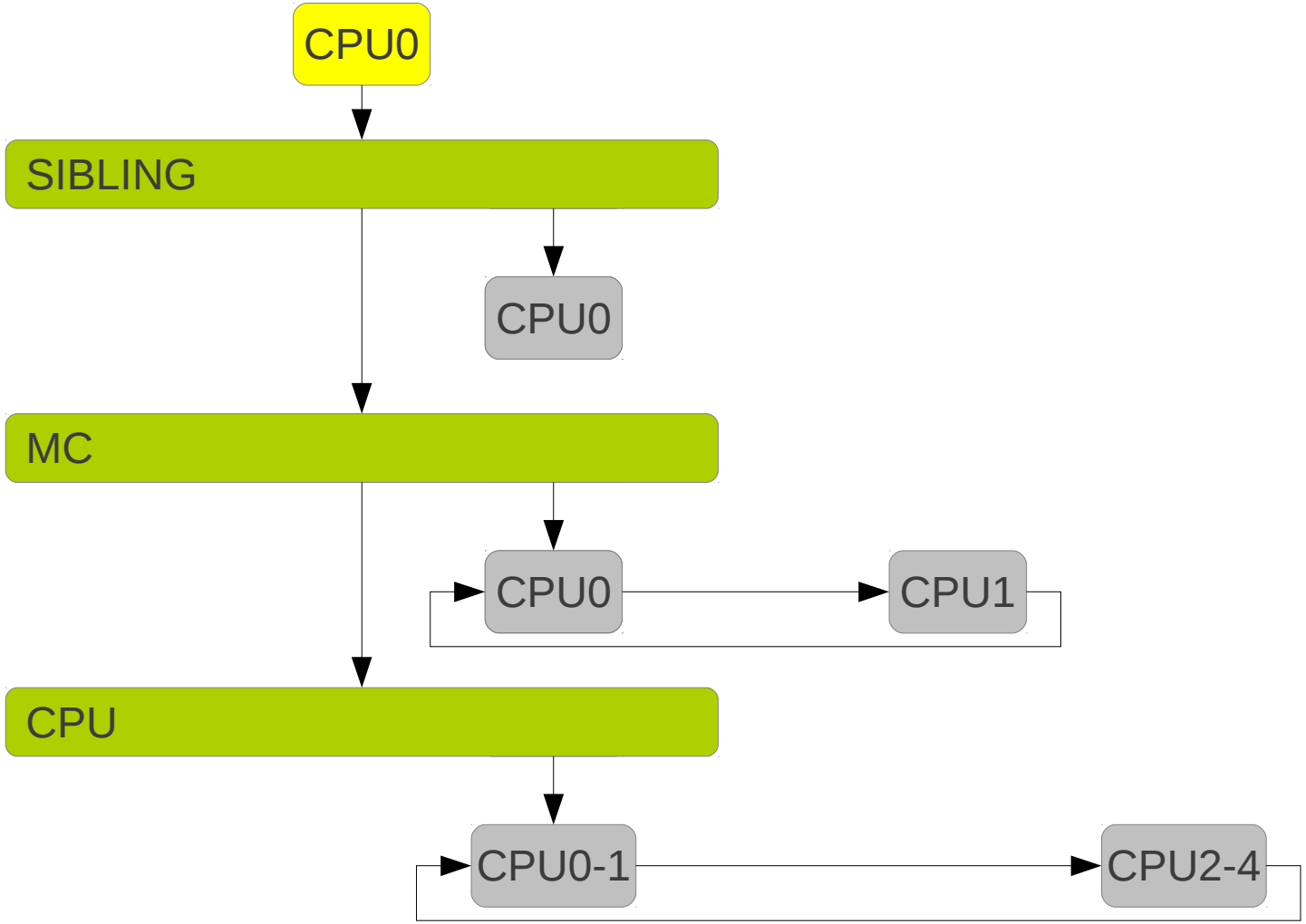
- Thanks to
  - struct sched_domain
  - and struct sched_group

- 4 sched_domain levels:
  - SIBLING level (hyper threading)
  - MC level (multi cores)
  - CPU level
  - NUMA level (several numa level)

- each level defines
  - its scheduling policy
  - for some groups of CPUs

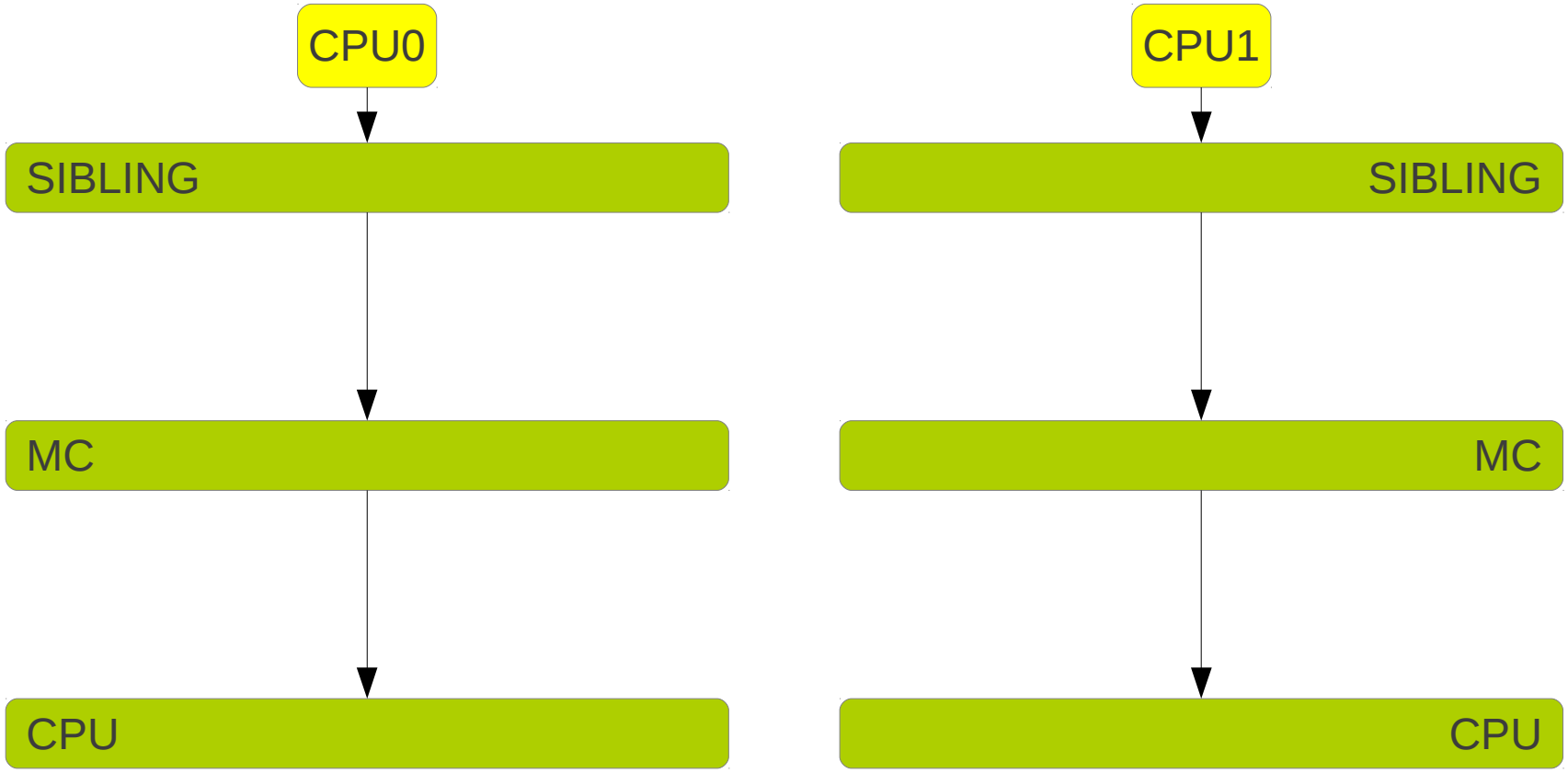
Topology: example

- CPU0 on a TC2



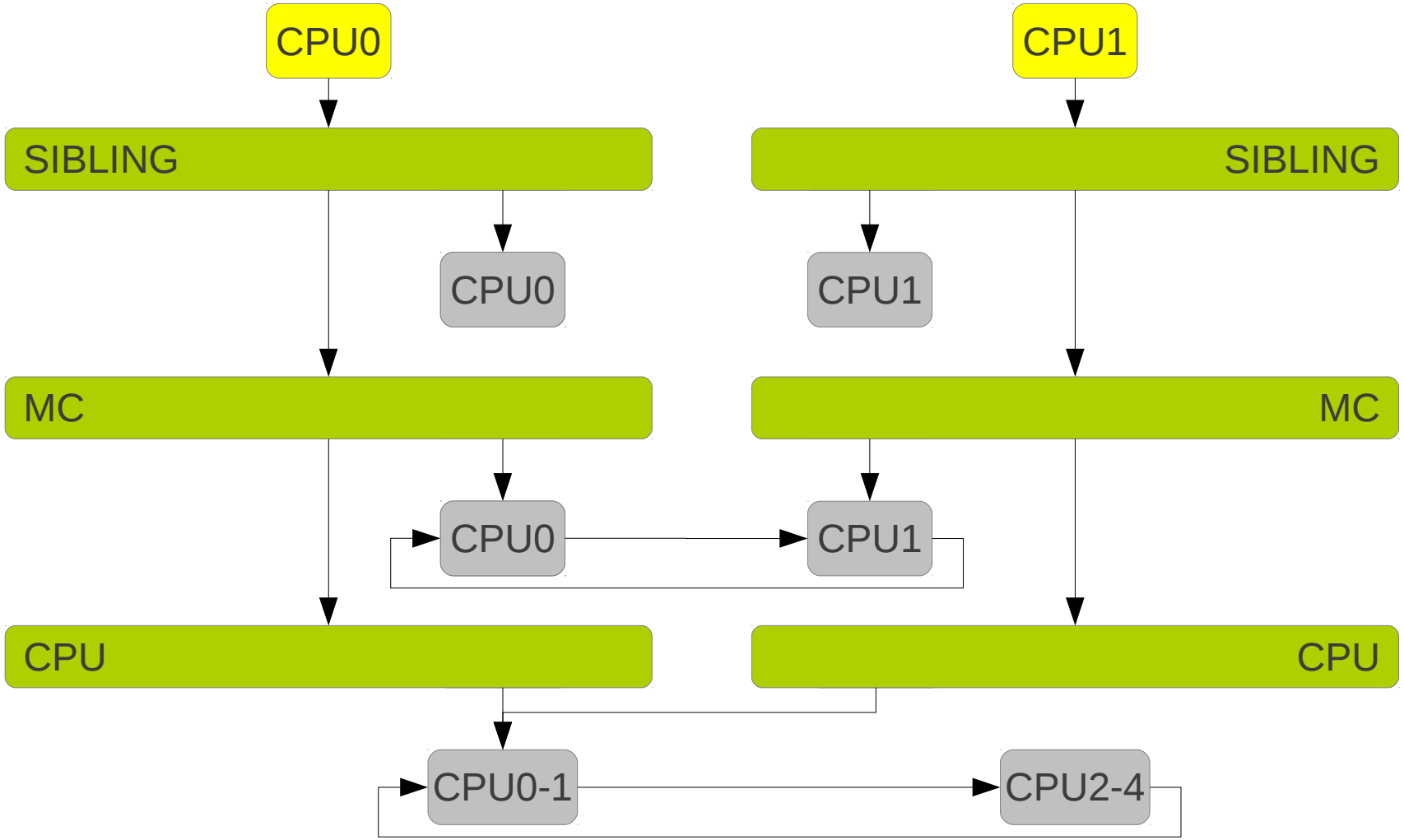
Topology: example

- Per CPU sched_domain



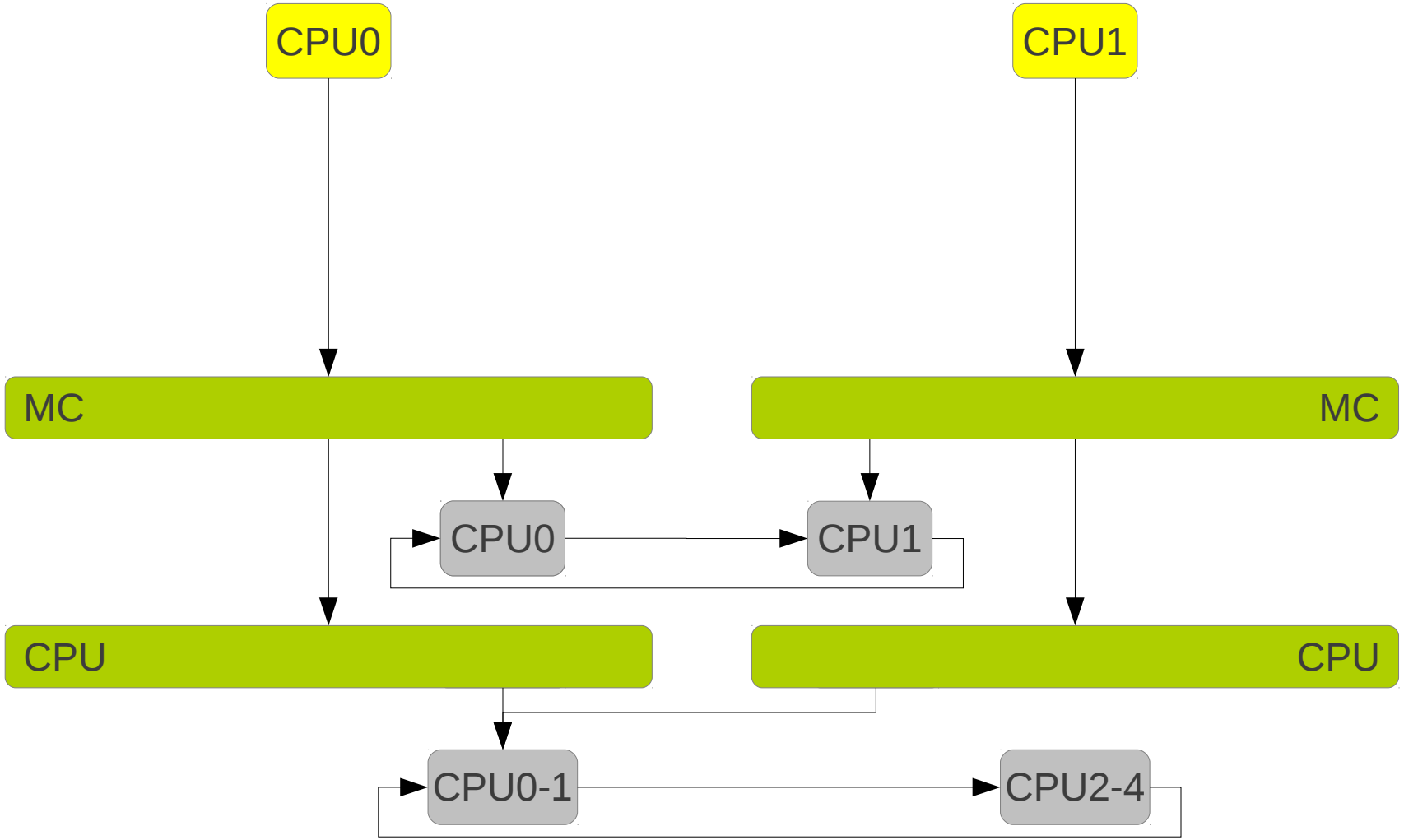
Topology: example

- Share sched_group



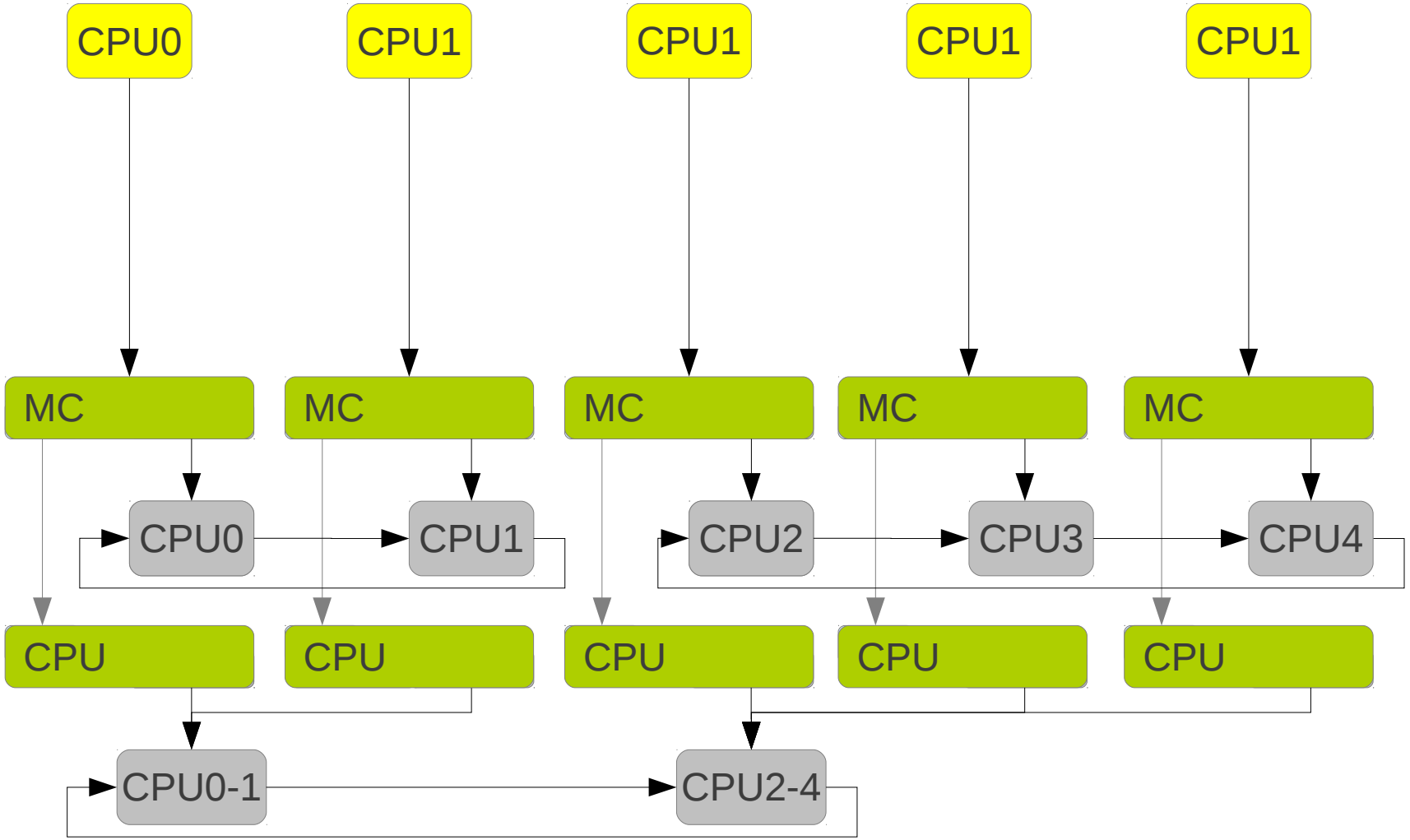
Topology: degenerate

- Remove useless sched_domain



Topology: example

- Complete view for TC2



sched_domain's fields

- sched_group->sched_group_power->power
  - Capacity of CPU and group of CPUs
- min/max/balance_interval and busy_factor
- idle/busy/wakeup/forkexec_idx
  - Interaction with cpu_load
  - Interaction with load_contrib and runnable_load_avg
- imbalance_pct
  - Threshold of load balance

sched_domain's fields

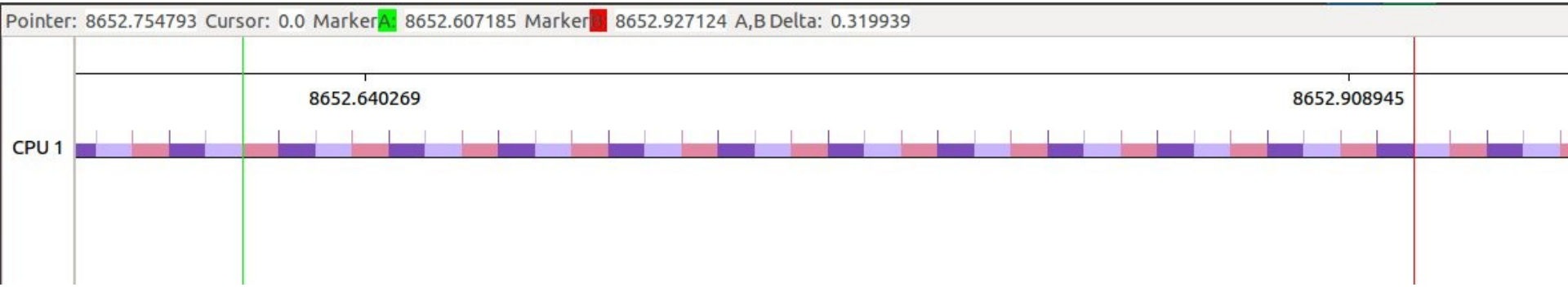
- Flag:
  - SD_LOAD_BALANCE: Do load balancing on this domain
  - SD_BALANCE_NEWIDLE: Balance when about to become idle
  - SD_BALANCE_EXEC: Balance on exec
  - SD_BALANCE_FORK: Balance on fork, clone
  - SD_BALANCE_WAKE: Balance on wakeup
  - SD_WAKE_AFFINE: Wake task to waking CPU

Scheduler class

- 4 scheduler class
  - stop: migration thread
  - rt: real-time scheduler
  - cfs: Completely Fair Scheduler
  - idle: Idle thread
- We will focus on CFS scheduler

Completely Fair Scheduler

- Give the same amount of runtime to all tasks
  - Track runtime of tasks
  - Sort tasks in a rb tree using runtime
  - Select the task with lowest runtime
- 3 tasks pinned on CPU1

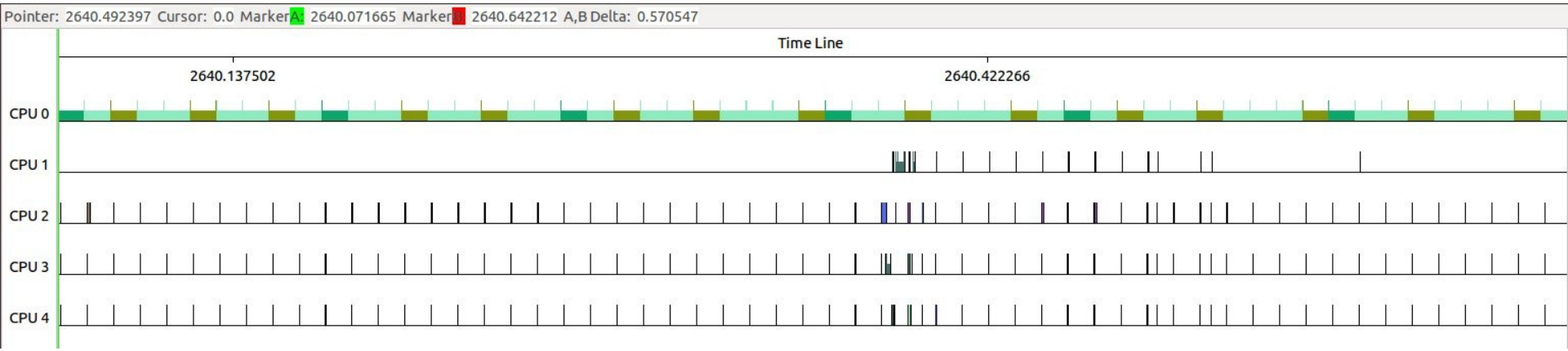


Nice priority

- Priority weights the runtime in a virtual runtime (vruntime)
- Time runs slower for lower nice priority (up to -20)
  - ~ 10% per nice level increase
- vruntime is the time that is used in rb tree

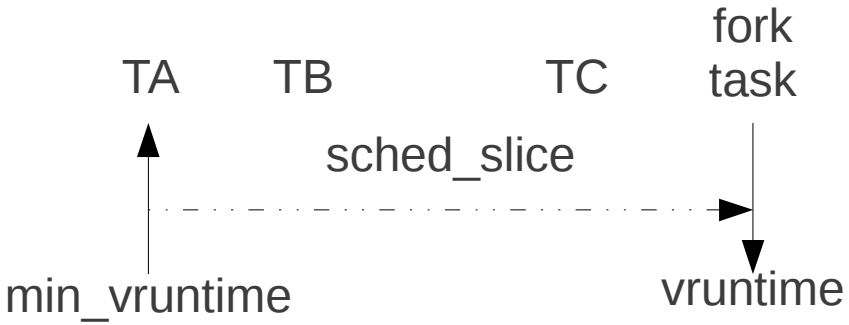
Nice priority: example

- 3 tasks pinned on CPU0 with prio -4, 0, +4
  - Task A with nice prio -4: 36 ticks
  - Task B with nice prio 0: 15 ticks
  - Task C with nice prio +4: 6 ticks

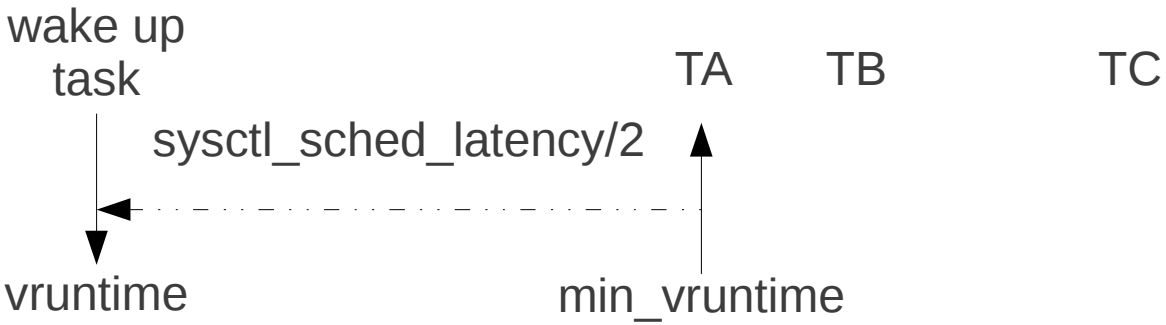


Task wakeup

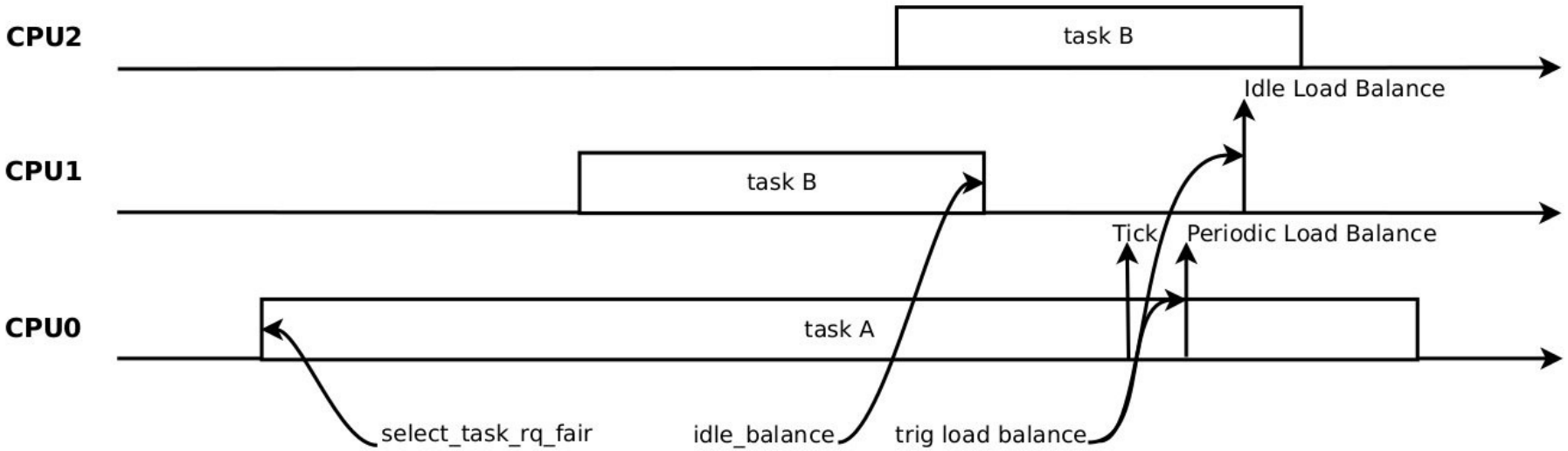
- Task's vruntime is updated at fork and wakeup
- At fork
  - after the current sched_slice



- At wake up:
  - Before the min_vruntime of the rq
  - sysctl_sched_latency / 2



Load balance



- @ wake up
- Newly idle CPU
- Periodically: busy / idle

@ wakeup: select target CPU

- Waking up tasks
  - Wake affine to current CPU if possible
- New task
  - Select the idlest CPU
- Have a look at select_task_rq_fair

Newly idle CPU

- Filter short idle time
  - Prevent to pull task on a not really idle system
  - Prevent to waste time
- Do a “normal” load balance
  - On each sched_domain
  - Try to pull task from an other imbalance group
- Have a look at idle_task

Periodic load balance

- At each tick:
  - Evaluate if a load balance is needed
  - On current CPU
    - Other CPU might be even more loaded
  - On Idle CPU
    - Local CPU might be overloaded
    - The local group might be imbalanced
- Have a look at trigger_load_balance

load balance

- Have a look at load_balance on a sched_domain
  - Find a busiest group at this level
  - Find a busiest queue in this group
  - Try to move waiting tasks
  - Try to move current task
- Have a look at load_balance

Per task load tracking

- Track average runnable of each task and CPU
- Use a geometric series

$$U_{n+1} = usage_{n+1} + Y \cdot U_n$$

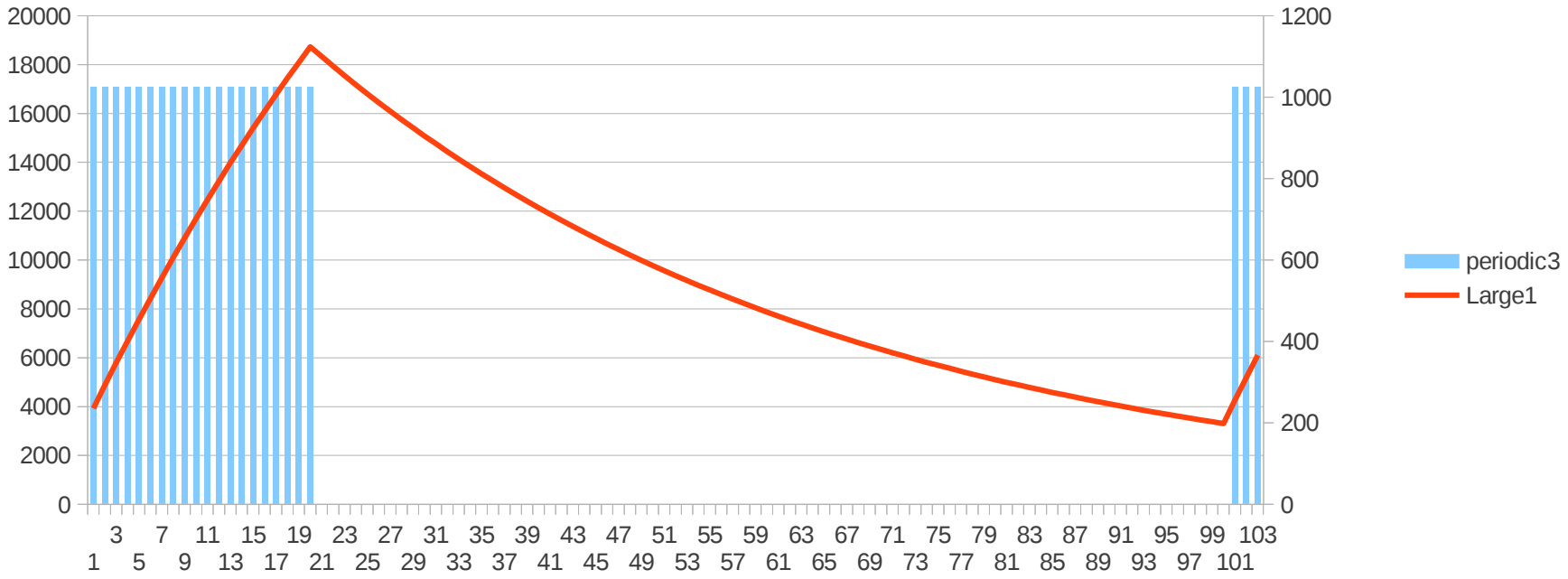
$$Y^{32} \simeq 0.5$$

$$usage \in [0 ; 1024]$$

$$U_{max} = \frac{1}{1-Y} \cdot usage_{max} \simeq 47742$$

A simple example

- Run 20 ms each 100ms



- Have a look at some runnable average example

Kernelshark

- Use trace-cmd to get trace
- Use the following sequence

```
# trace-cmd start <option>
# Do something or just sleep a bit
# trace_cmd stop
# trace_cmd extract -o <file>
```

- Use kernelshark viewer
- Have a look at an example

More about Linaro Connect: www.linaro.org/connect/
More about Linaro: www.linaro.org/about/
More about Linaro engineering: www.linaro.org/engineering/