CFS 스케줄러 파라메터

국민대학교 임베디드 연구실
경 주 현

2017 LG MC 인재육성팀 교육

Outline

- **CFS 스케줄러 파라메터 설명**

- **파라메터 변경 후 Trace**

Why talk about parameters?

- **CFS scheduler depends on CFS parameters.**

- **Parameters prevent context switch overhead.**

- **Tuning Value**
  - sched_nr_latency
  - sysctl_sched_latency
  - sysctl_sched_min_graularity
  - sysctl_sched_wakeup_granularity

CFS time slice

time_slice =

(sched_period * se.load.weight) / cfs_rq.load.weight;

Time slice and task load

```
time_slice =
(sched_period * se.load.weight) / cfs_rq.load.weight;
```

- nr_running : number of threads in runqueue

```
if nr_running <= nr_latency
    sched_period = sched_latency
else
        sched_period = min_granularity x nr_running
```

Parameter values

- **rq->nr_running**
- **sched_nr_latency**
  - sched_latency_ns / sched_min_granularity
- **sched_latency_ns**
  - 예 : 6000000 * factor(3) = 18000000(ns)
  - “/proc/sys/kernel/sched_latency_ns”
- **factor**
  - 스케일링 정책에 따른 비율 (예 - factor=3)
- **sched_min_granularity**
  - 최소 스케쥴 기간
  - 예) 750000 * factor(3) = 2250000(ns)
  - “/proc/sys/kernel/sched_min_granularity_ns”
- **sched_wakeup_granularity_ns**
  - 태스크의 wakeup 기간으로 디폴트 값(1000000) * factor(3) = 3000000(ns)
  - “/proc/sys/kernel/sched_wakeup_granularity_ns”

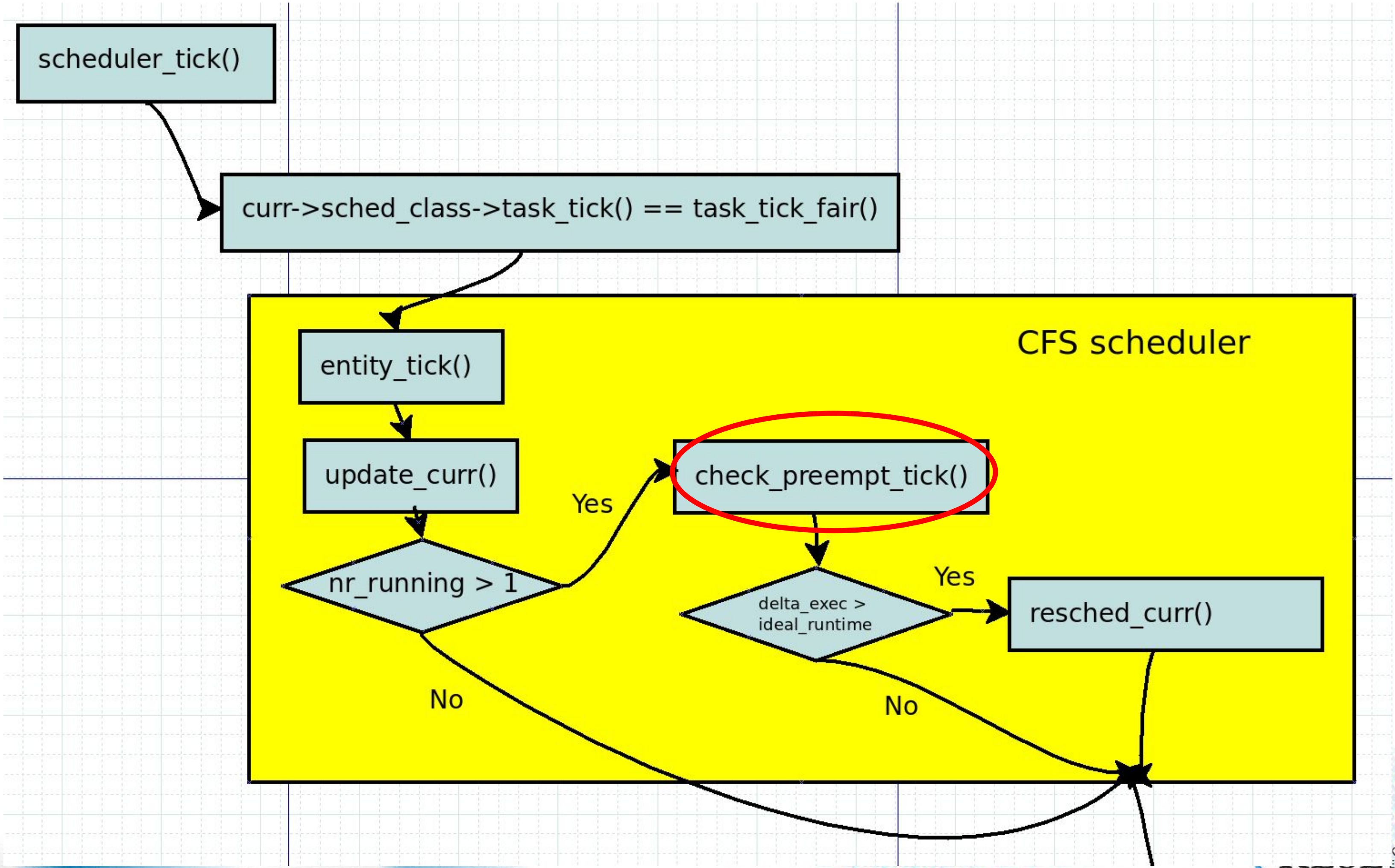
vruntime

vruntime +=
delta_exec * (NICE_0_LOAD / curr->load.weight);

- **delta_exec**
  - the time spent by the task since the last time vruntime was updated

When schedule a task

- **Scheduler Tick**



check_preempt_tick

- **delta_exec**
  - the time spent by the task since the last time vruntime was updated

```
if (delta_exec > time_slice)
    reched_task();
else if ((delta_exec < min_granularity)
    retrun; // To prevent context switch overhead
```

wakeup_preempt_entity

```
gran =
sched_wakeup_granularity * (NICE_0_LOAD / se>load.weight);
```

- **cur : wake-up task, se : running task**

```
if (cur->vruntime <= se->vruntime)
    return;
else ((cur->vruntime - se->vruntime) > gran)
    resched_task();
```

Next Step.

Energy-aware scheduling: EAS

1. **Load Balancer(Group Scheduling, Bandwidth Control, PELT)**
2. **EAS features**

Reference

- https://pdos.csail.mit.edu/6.828/2016/schedule.html
- http://web.mit.edu/6.033
- http://www.rdrop.com/~paulmck/
- “**Is Parallel Programming Hard, And If So, What Can You Do About It?”**
- Davidlohr Bueso. 2014. Scalability techniques for practical synchronization primitives. *Commun. ACM* 58

http://queue.acm.org/detail.cfm?id=2698990

- **"CPUFreq and The Scheduler Revolution in CPU Power Management", Rafael J. Wysocki**
- **https://sites.google.com/site/embedwiki/oses/linux/pm/pm-qos**
- **https://intl.aliyun.com/forum/read-916**
- **User-level threads : co-routines**

  http://www.gamedevforever.com/291

  https://www.youtube.com/watch?v=YYtzQ355_Co

- **Scheduler Activations**
  - https://cgi.cse.unsw.edu.au/~cs3231/12s1/lectures/SchedulerActivations.pdf
- **https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)**
- **http://jake.dothome.co.kr/**
- **http://www.linuxjournal.com/magazine/completely-fair-scheduler?page=0,0**
- **https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html**
- **"Energy Aware Scheduling", Byungchul Park, LG Electronic**
- **“Update on big.LITTLE scheduling experiments”, ARM**
- **“EAS Update” 2015 september ARM**
- **"EAS Overview and Integration Guide”, ARM TR**
- **“Drowsy Power Management”, Matthew Lentz, SOSP 2015**
- **https://www.slideshare.net/nanik/learning-aosp-android-hardware-abstraction-layer-hal**
- https://www.youtube.com/watch?v=oTGQXqD3CNI
- https://www.youtube.com/watch?v=P80NcKUKpuo
- https://lwn.net/Articles/398470/
- “SCHED_DEADLINE: It’s Alive!”, ARM, 2017