

ITP 30002 Operating System

Proportional Share Scheduler

OSTEP Chapter 9

Shin Hong

Lottery Scheduler

2

- Proportional-share scheduler aims to guarantee that a process is given a certain portion of the CPU time
- Lottery scheduling: hold a lottery to determine a process for the next turn
 - a process holds **tickets**. The number of tickets a process hold represents the resource-share that a process should receive
 - for each scheduling decision:
 - counts the total number of tickets held by the runnable processes
 - picks a winning ticket from the whole tickets
 - dispatches the owner process of the winning ticket
 - example: Process-A with 75 tickets (0 to 74), Process-B with 25 tickets (75 to 99)

63 85 70 39 76 17 29 41 36 39 10 99 68 83 63 62 43 0 49 12

A A A A A A A A A A A A A A A A

B

B

B

B

Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13

Lottery Scheduler - Good Points

3

- The algorithm is simple and does not incur much computation



- By managing the total number of tickets, the system can easily and reliably control the scheduling priorities of different processes
 - mechanisms
 - ticket currency
 - ticket transfer
 - ticket inflation
- It achieves good fairness unless processes run for a short time period

Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13

Stride Scheduling

4

- A deterministic fair-share scheduler
- A process is given with a **stride** value which is inverse in proportion to the number of tickets
 - the shorter a stride, the more time slices (steps) the process have in a time unit
- A **pass** value is maintained for each process, which is increased by the stride value when the process receives a time slice
- At a scheduling decision, the scheduler selects a ready process with the smallest pass value

```
curr = remove_min(queue);    // pick client with min pass
schedule(curr);              // run for quantum
curr->pass += curr->stride;    // update pass using stride
insert(queue, curr);         // return curr to queue
```

Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13

Example

5

Pass(A) (stride=100)	Pass(B) (stride=200)	Pass(C) (stride=40)	Who Runs?
0	0	0	A
100	0	0	B
100	200	0	C
100	200	40	C
100	200	80	C
100	200	120	A
200	200	120	C
200	200	160	C
200	200	200	...

Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13

The Linux Completely Fair Scheduler (CFS)

6

- The cost of running scheduler is a significant factor of system performance
 - Around 5% of overall CPU time of Google datacenter is consumed by computation for scheduling
- The approach of CFS
 - reduce the number of context-switches
 - divide a period of CPU time (`sched_latency`) evenly among all ready processes while keeping a time-slice from getting too fine-grained (`min_granularity`)
 - enable users to give time-slice weighting to control process priority
 - share CPU time fairly
 - use a periodic timer interrupt to identify the expiration of a current time slice
 - count virtual runtime to trace the accumulated CPU time of each process
 - select the runnable process of the minimum virtual runtime

Proportional Share
Scheduler

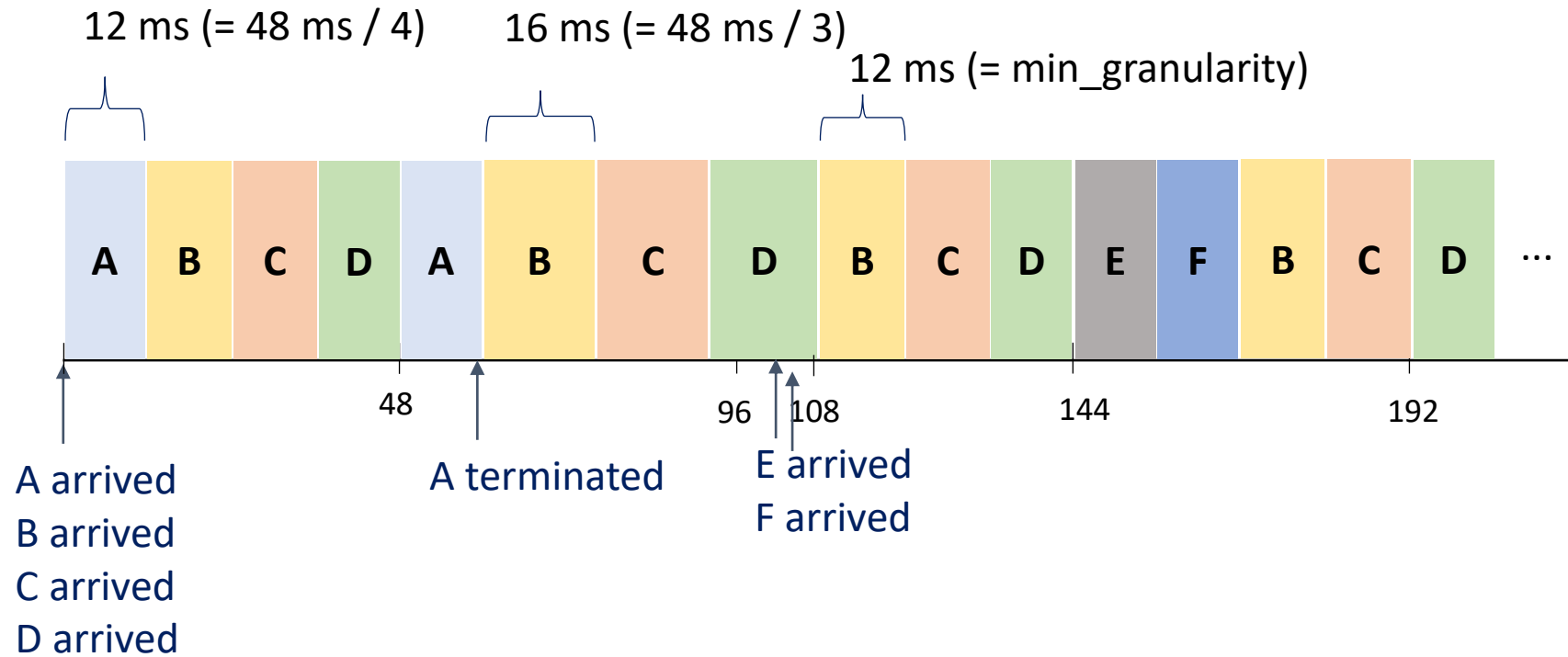
ITP 30002
Operating System

2023-04-13

Example

7

- sched_latency: 48 ms
- min_granularity: 12 ms
- timer interrupt: every 4 ms



Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13

Process Weighting

8

- The nice level in Unix represents the priority of a process
 - the lower a nice value, the higher priority is (the greater the share is).
- The time slice of a process is proportional to its weight

$$\text{time_slice}_k = \frac{\text{weight}_k}{\sum_{i=0}^{n-1} \text{weight}_i} \cdot \text{sched_latency}$$

```
static const int prio_to_weight[40] = {  
    /* -20 */ 88761, 71755, 56483, 46273, 36291,  
    /* -15 */ 29154, 23254, 18705, 14949, 11916,  
    /* -10 */ 9548, 7620, 6100, 4904, 3906,  
    /* -5 */ 3121, 2501, 1991, 1586, 1277,  
    /* 0 */ 1024, 820, 655, 526, 423,  
    /* 5 */ 335, 272, 215, 172, 137,  
    /* 10 */ 110, 87, 70, 56, 45,  
    /* 15 */ 36, 29, 23, 18, 15,
```

- accumulated CPU runtime is measured with respect to process weighting scheme
 - otherwise, a high priority process can starve

$$\text{vruntime}_i = \text{vruntime}_i + \frac{\text{weight}_0}{\text{weight}_i} \cdot \text{runtime}_i$$

Proportional Share
Scheduler

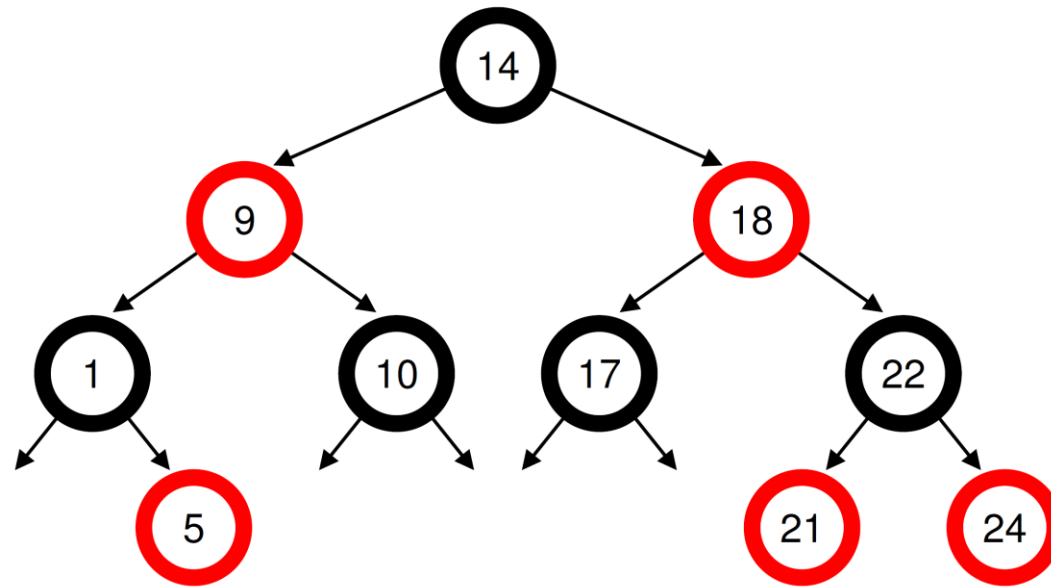
ITP 30002
Operating System

2023-04-13

Process Management

9

- A ready queue is formed as a red-black tree having vruntime as a key
 - balanced binary tree
 - $\log(n)$ for search, insertion, removal



- When a process is awoken from a sleep, the scheduler sets its vruntime as the minimum vruntime found in the ready queue to avoid monopoly

Proportional Share
Scheduler

ITP 30002
Operating System

2023-04-13