



Completely Fair Scheduling (CFS)

Pérez Palomeque José Mauricio

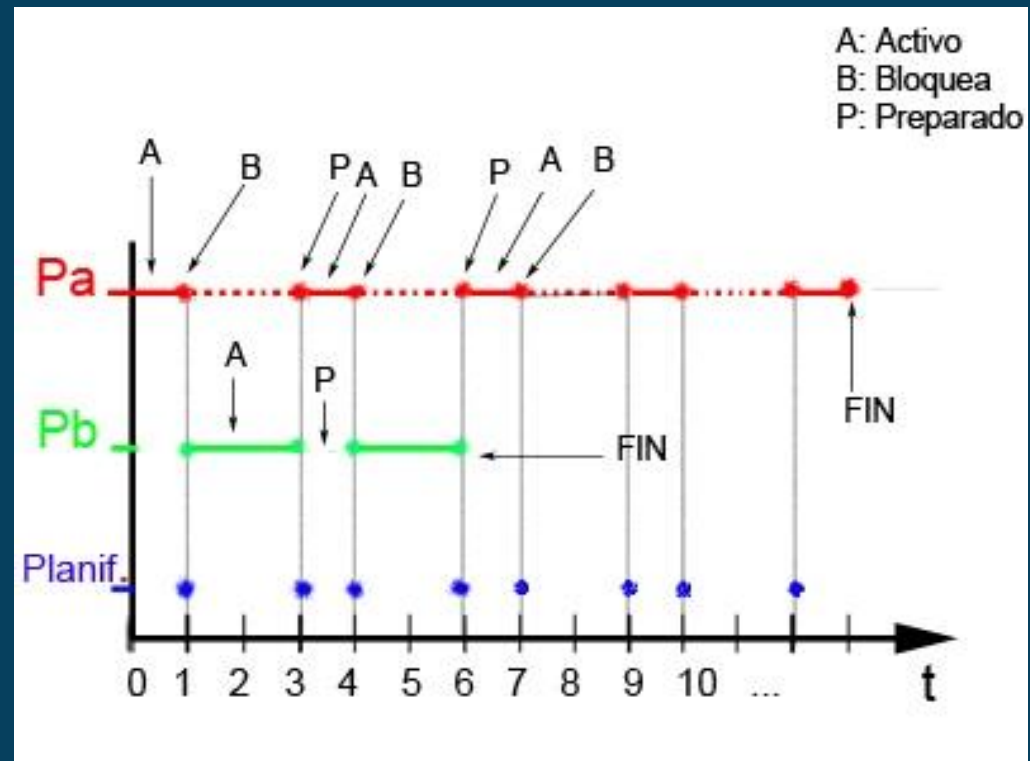
¿Qué es un planificador?

- Herramienta para que el sistema operativo determine el orden en que se adecua el procesador a los procesos que lo vayan necesitando
- Según las políticas que se utilizarán en la eficiencia del tiempo esperado en el sistema.



Algoritmos vistos previamente

- Prioridad al proceso más corto (SFJ)
- Primero en entrar – Primero en salir (FCFS)
- Round Robin



¿Qué es CFS?

- Planificador de Linux desde la versión 2.6.23
- Creado por Ingo Molnar
- Basado en el algoritmo Rotating Staircase Deadline Scheduler (RSDL)
- Incorporado en Linux Kernel desde año 2007
- Maneja la asignación de recursos de la CPU para ejecutar procesos
- Tiene como objetivo maximizar la utilización general de la CPU al mismo tiempo que maximiza el rendimiento interactivo.



Idea general

- Divide el tiempo de procesador de forma justa entre los procesos
- Si hay n procesadores listos para ser planificados, entonces cada uno de ellos recibe el mismo porcentaje del procesador
- Cada proceso debe tener $(100/n)\%$ del tiempo del procesador

Process	burst time
A	8ms
B	4ms
C	16ms
D	4ms

Ideal Fairness													
A	1	2	3	4	6	8							
B	1	2	3	4									
C	1	2	3	4	6	8	12	16					
D	1	2	3	4									

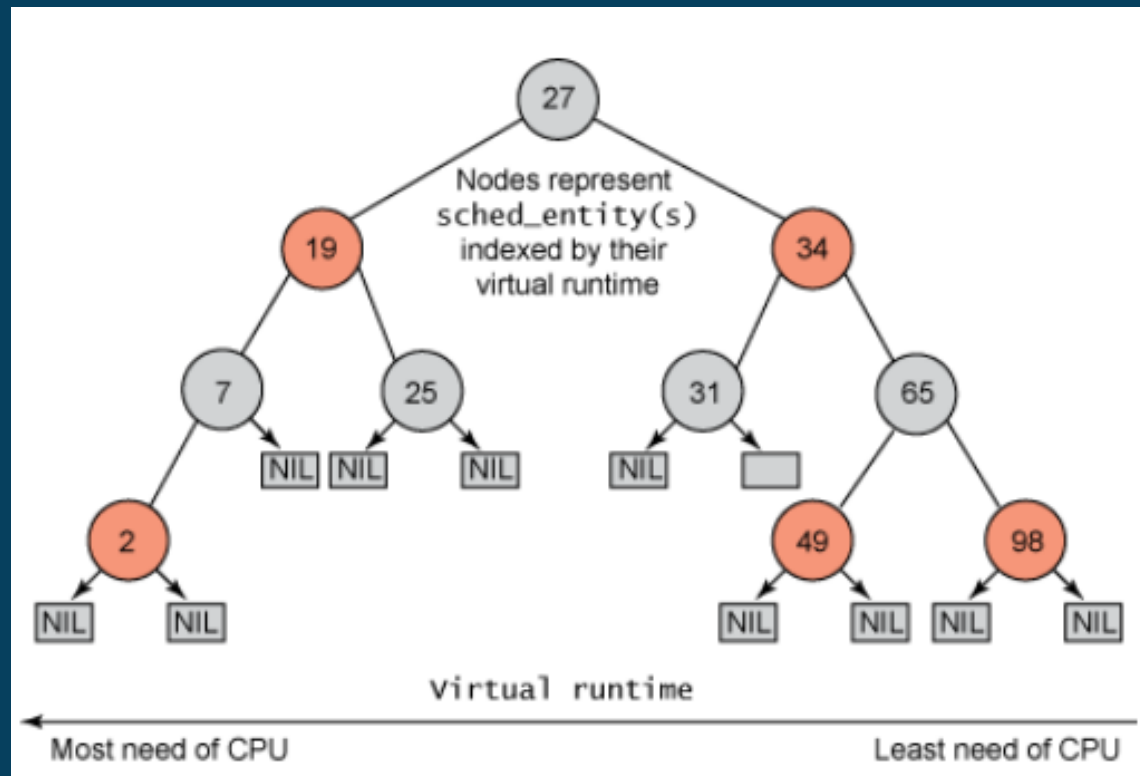
Virtual Runtimes

- A cada proceso se le asigna un virtual runtime (vruntime)
- Se define como el tiempo que se ha ejecutado una tarea en la CPU
- Vruntime aumenta en cada “*punto de planificación*”, si un proceso ha corrido por un tiempo t en el procesador. $Vruntime += t$
- Se actualiza determinando el tiempo transcurrido desde la ultima llamada a la tarea/proceso

Ideal Fairness													
A	1	2	3	4	6	8							
B	1	2	3	4									
C	1	2	3	4	6	8	12	16					
D	1	2	3	4									

Uso del árbol rojo negro en CFS

- Las propiedades de un árbol rojo-negro son muy útiles a la hora de implementar CFS
- Es capaz de equilibrarse así mismo, lo cual quiere decir que ningún camino en el árbol será más del doble de cualquier otro
- Las operaciones en el árbol ocurren en tiempo $O(\log n)$ donde n es el número de nodos en el árbol. Lo cual significa que se pueden insertar o eliminar una tarea de manera rápida y eficiente.



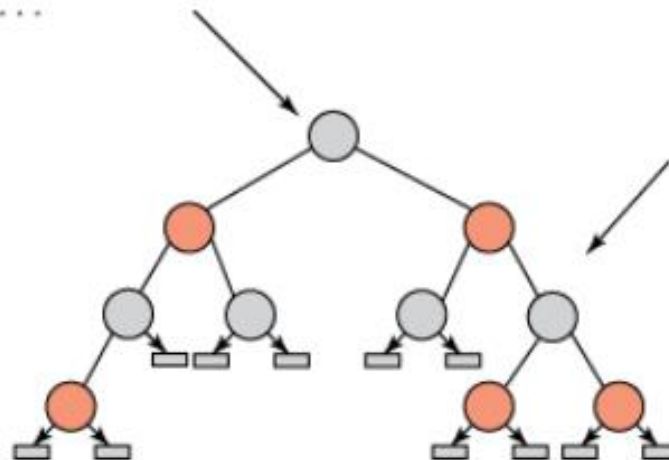
Jerarquía de estructuras en árbol rojo-negro para CFS

```
struct task_struct {  
    volatile long state;  
    void *stack;  
    unsigned int flags;  
    int prio, static_prio normal_prio;  
    const struct sched_class *sched_class;  
    struct sched_entity se;  
    ...  
};
```

```
struct ofs_rq {  
    ...  
    struct rb_root tasks_timeline;  
    ...  
};
```

```
struct sched_entity {  
    struct load_weight load;  
    struct rb_node run_node;  
    struct list_head group_node;  
    ...  
};
```

```
struct rb_node {  
    unsigned long rb_parent_color;  
    struct rb_node *rb_right;  
    struct rb_node *rb_left;  
};
```



Prioridades y CFS

- CFS no usa las prioridades directamente, sino que las usa como un factor de disminución para el tiempo que se permite ejecutar una tarea.
- Las tareas de menor prioridad tienen factores de deterioro más altos, mientras que las tareas de mayor prioridad tienen factores de retraso más bajos.



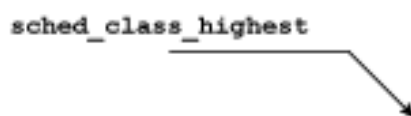
Planificación de grupos CFS

- La planificación de grupos es otra forma de hacer que la programación sea justa, particularmente frente a tareas que generan muchas otras tareas.
- Considere un servidor que genere muchas tareas para paralelizar las conexiones entrantes (una arquitectura típica para servidores HTTP).
- En lugar de que todas las tareas se traten de manera bastante uniforme, CFS introduce grupos para dar cuenta de este comportamiento.
- El proceso del servidor que genera tareas comparte sus tiempos de ejecución virtuales en todo el grupo (en una jerarquía), mientras que la tarea individual mantiene su propio tiempo de ejecución virtual independiente.




Clases de planificación

- También se introdujo con CFS la idea de planificar clases.
- Cada tarea pertenece a una clase de planificación, que determina cómo se planificará una tarea.
- Una clase de planificación define un conjunto común de funciones (a través de sched_class) que definen el comportamiento del planificador.



	kernel/sched_rt.c	kernel/sched_fair.c	kernel/sched_idletask.c
sched_class	rt_sched_class	fair_sched_class	idle_sched_class
	next	next	NULL
enqueue_task	enqueue_task_rt	enqueue_task_fair	NULL
dequeue_task	dequeue_task_rt	dequeue_task_fair	dequeue_task_idle
yield_task	yield_task_rt	yield_task_fair	NULL
check_preempt_curr	check_preempt_curr_rt	check_preempt_wakeup	check_preempt_curr_idle
pick_next_task	pick_next_task_rt	pick_next_task_fair	pick_next_task_idle
put_prev_task	put_prev_task_rt	put_prev_task_fair	put_prev_task_idle
...
	SCHED_FIFO /SCHED_RR	SCHED_OTHER	



Gracias por su
atención

Referencias

- *IBM Developer*. (s. f.). <https://developer.ibm.com/tutorials/l-completely-fair-scheduler/>
- *CFS Scheduler — The Linux Kernel documentation*. (s. f.). <https://www.kernel.org/doc/html/latest/scheduler/sched-design-CFS.html>
- Huang, J. (2020, 27 abril). *Linux Container CPU: How to Optimize Real-Time and I/O-Intensive Environments*. dzone.com. <https://dzone.com/articles/linux-container-cpu-performance-how-you-can-optimi>
- *Completely Fair Scheduler | Linux Journal*. (s. f.). <https://www.linuxjournal.com/node/10267>