

Fair Share Scheduling in Linux

Group: 4

Institute of Engineering and Technology, Ahmedabad University

April 30, 2015

Anmol Anubhai - 121004

Pooja Shah - 121035

Rahul Patel - 121040

Shashwat Sanghavi - 121049

Overview

What are Schedulers?

Linux Schedulers

- Multilevel Feedback queue

- O(1) Scheduler

- Completely Fair Share Scheduling

Fair share scheduling

Implementation

- Background

- Algorithm

- Results

Conclusion

References

What are Schedulers?

- ▶ The main goals of a scheduler is to increase the system utilization while not compromising on the wait time and response time of jobs.
- ▶ Performs key role for operating system efficiency

Linux Schedulers

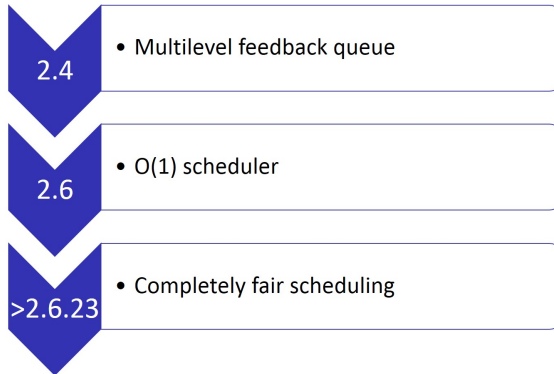


Figure : Linux kernels and their schedulers

Multilevel Feedback Queue

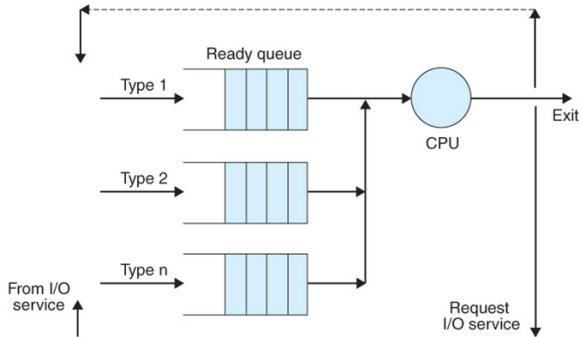


Figure : Multilevel Priority Topics

O(1) Scheduler

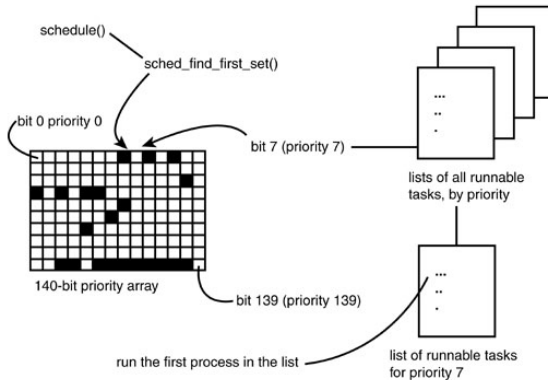


Figure : O(1) Scheduler

Completely Fair Share Scheduling

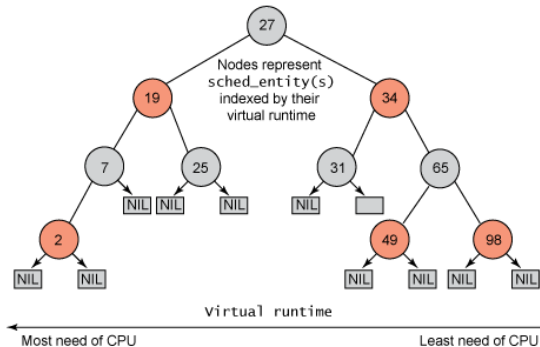


Figure : CFS RB Tree

Fair share scheduling

- ▶ Every user gets an equal share of the CPU.
- ▶ Every processes of one user gets equal amount of CPU resources.

| User | Processes | CPU time per user | CPU time per process |
|------|-----------|-------------------|----------------------|
| A | 3 | 33% | 11% |
| B | 2 | 33% | 16.5% |
| C | 4 | 33% | 8.25% |

Figure : FSS example

Background

- ▶ Choice of Linux kernel 2.6.38
- ▶ File of interest: sched.c
- ▶ Some useful functions
 - ▶ `schedule()`
 - ▶ `sched_fork()`
 - ▶ `finish_task()`
 - ▶ `task_cred()`
- ▶ Useful structure: `task_struct`
- ▶ Multiple priority runqueue

Block diagram-New process is forked

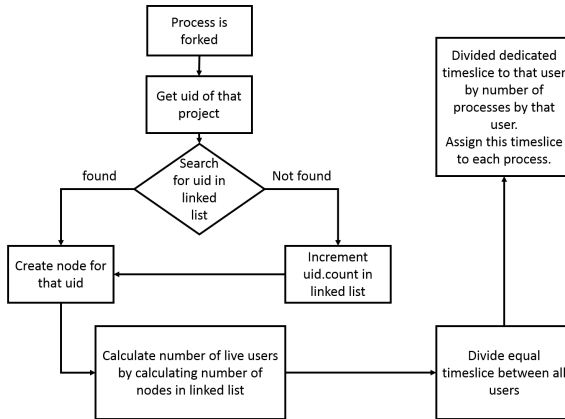


Figure : Algorithm which should be followed when new process is forked

Block diagram-Process is terminated

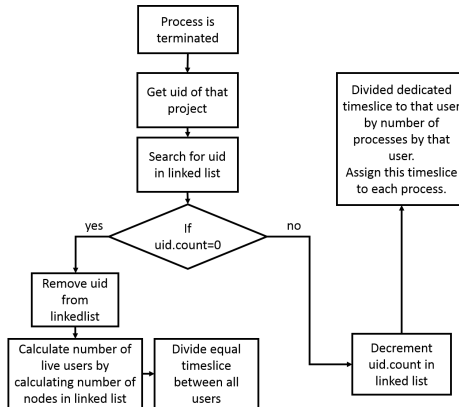


Figure : Algorithm which should be followed when process is terminated

Results

```

user@ubuntu:~$ dmesg | tail
[ 211.687389] total timeslices-100
[ 211.687390] **** uid->0 *** processes->225 *** timeslice-0 ****
[ 211.687391] **** uid->1000 *** processes->145 *** timeslice-0 ****
[ 211.687392] **** uid->1001 *** processes->2 *** timeslice-16 ****
[ 211.687922]
[ 211.687922] total timeslices-100
[ 211.687924] **** uid->0 *** processes->225 *** timeslice-0 ****
[ 211.687925] **** uid->1000 *** processes->145 *** timeslice-0 ****
[ 211.687926] **** uid->1001 *** processes->2 *** timeslice-16 ****
user@ubuntu:~$

user@ubuntu:~$ dmesg | tail
[ 323.325580] **** uid->1000 *** processes->150 *** timeslice-0 ****
[ 323.325581] **** uid->1001 *** processes->4 *** timeslice-6 ****
[ 323.325582] **** uid->1002 *** processes->2 *** timeslice-12 ****
[ 323.326172]
[ 323.326173] total timeslices-100
[ 323.326174] **** uid->0 *** processes->193 *** timeslice-0 ****
[ 323.326176] **** uid->1000 *** processes->150 *** timeslice-0 ****
[ 323.326177] **** uid->1001 *** processes->4 *** timeslice-6 ****
[ 323.326178] **** uid->1002 *** processes->2 *** timeslice-12 ****
user@ubuntu:~$

user@ubuntu:~$ dmesg | tail
[ 367.777101] **** uid->1002 *** processes->2 *** timeslice-10 ****
[ 367.777102] **** uid->1003 *** processes->6 *** timeslice-3 ****
[ 367.778290]
[ 367.778291] total timeslices-100
[ 367.778292] **** uid->0 *** processes->197 *** timeslice-0 ****
[ 367.778293] **** uid->1000 *** processes->151 *** timeslice-0 ****
[ 367.778294] **** uid->1001 *** processes->4 *** timeslice-5 ****
[ 367.778295] **** uid->1002 *** processes->2 *** timeslice-10 ****
[ 367.778296] **** uid->1003 *** processes->6 *** timeslice-3 ****
user@ubuntu:~$

```

Figure : Results showing the effect of fork exit and user login over the timeslice division

Conclusion

- ▶ Our algorithm gives desired output for FSS.
- ▶ But, it is $O(n)$ algorithm where n is the number of users.
- ▶ Hence, cannot be integrated in linux kernel since it strictly follows $O(1)$ scheduler policy.

Thank You.

REFERENCES



Scheduling in Linux http://www.cs.montana.edu/~chandrima.sarkar/AdvancedOS/CSCI560_Proj_main/



Process Scheduling in Linux

<http://www.ittc.ku.edu/~kulkarni/teaching/EECS678/projects/scheduling/materials/scheduling.pdf>



Implementing a new real-time scheduling policy for Linux: Part 1

<http://www.embedded.com/design/operating-systems/4204929/Real-Time-Linux-Scheduling-Part-1>



Background Process Scheduling <http://web.cs.wpi.edu/~claypool/courses/3013-A05/projects/proj1/>



Linux Kernel Map <http://www.makelinux.net/books/lkd2/ch04lev1sec2>



Modify the Linux Scheduler to limit the CPU usage of a process family

http://www.csd.uoc.gr/~hy345/assignments/2013/cs345_front4.pdf



LinSched: The Linux Scheduler Simulator <http://www.cs.unc.edu/~jmc/linsched/>



Understanding the Linux 2.6.8.1 Process Scheduler

<http://cs.boisestate.edu/~amit/teaching/597/scheduling.pdf>

REFERENCES



CFS Scheduler <https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt>



Fair-Share Scheduling <http://dasl.mem.drexel.edu/~ducNguyen/courses/cs370-operating-systems/cs370-p4-fair-share-scheduling/>



Linux Kernel 2.6.22.19 Scheduler

<http://dasl.mem.drexel.edu/~ducNguyen/2013/08/15/linux-kernel-2-6-22-19-scheduler/>