Linux (Fedora or Slackware) CPU Scheduling

DHEYAULDEEN MAHMOOD
Student no. 163104411
Department: IT

Submit to Prof.Dr.Hassan Huseyin Balik

Outline

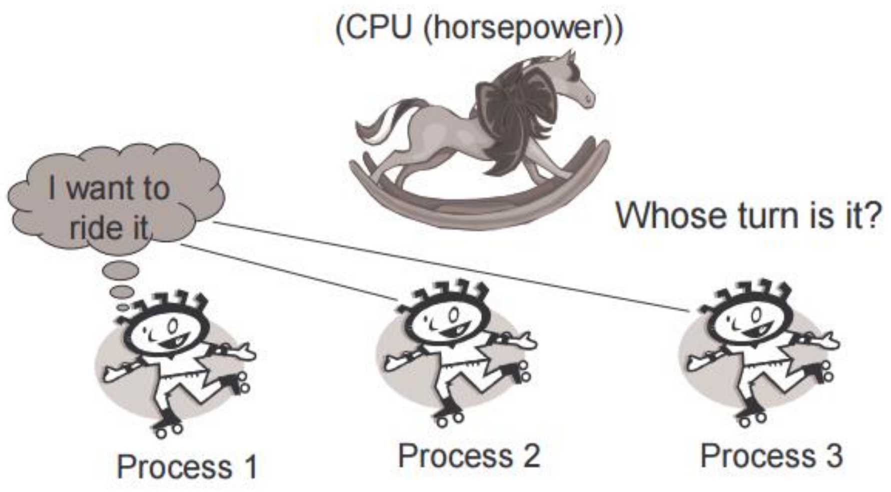
Introduction

- **CPU Scheduling**

CPU scheduling is a process which allows one process to use the CPU while the execution of another process is on hold(in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair [1].

- **Why Scheduling?**

Deciding which process/ thread should occupy the resources (CPU, disk, etc)[2]



➢ Scheduling Criteria Consisting of:

- **CPU utilization** make out the best use of CPU and not to waste any CPU cycle.
- **Throughput** It is the total number of processes completed per unit time.
- **Turnaround time** It is the amount of time taken to execute a particular process.
- **Waiting time** The sum of the periods spent waiting in the ready queue.
- **Load average** It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.
- **Response time** Amount of time it takes from when a request was submitted until the first response is produced.[3]

➢ CPU scheduling happen in Four cases?

1- A process switches from the running state to waiting state (e.g. I/O request)

2- A process switches from the running state to the ready state.

3- A process switches from waiting state to ready state (completion of an I/O operation)

4- A process terminates

➢ scheduling Objectives

1- Fairness

2- Priority

3- Efficiency Encourage good behavior

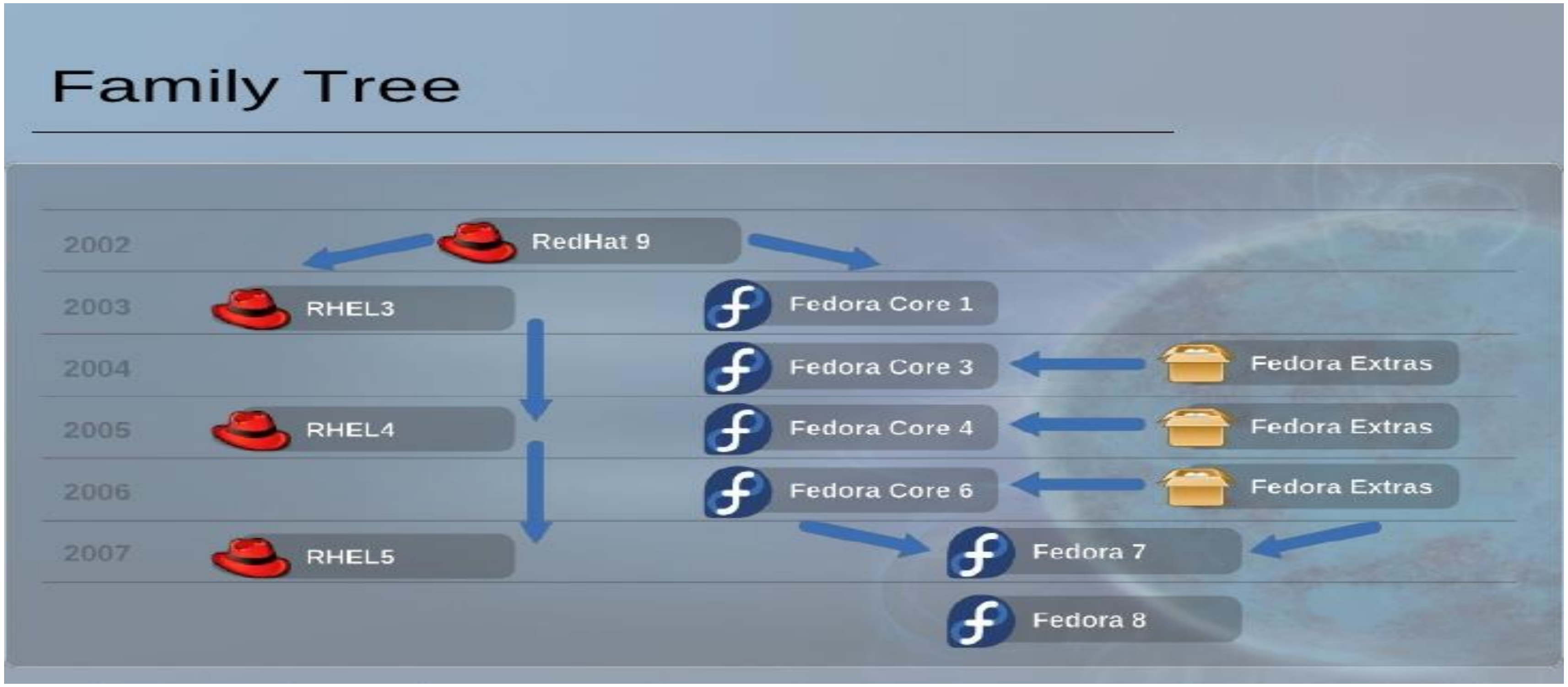
4- Support heavy loads

5- Adapt to different environments (interactive, real time, multi-media)

➢ Linux Fedora

- Fedora is a Linux-based operating system that showcases the latest in free and open source software from **“ Red Hat”** company**.**
- Fedora is always free for anyone to use, modify and distribute.
- It is built by people across the globe who work together

as a community: The Fedora Project



➢ Linux Fedora CPU Scheduler

Although CPU has many sharing algorithms to dealing with task scheduling in both Windows and Linux OS.There are some algorithms have been designed to used uniquely by Linux OS. For example :

- **An O(n) scheduler** was default in 2.4
- **An O(1) scheduler** was default in 2.6 before 2.6.23
- **Completely Fair Scheduler** is default scheduler since 2.6.23 onwards
- **Brain Fuck Scheduler** is a popular third-party scheduler available as a set of patches
- **Earliest Deadline-first** scheduler is available since 3.14 designed for real-time workload[4]

Completely Fair Scheduler (CFS)

Since Kernel 2.6.23 CFS Aiming at:

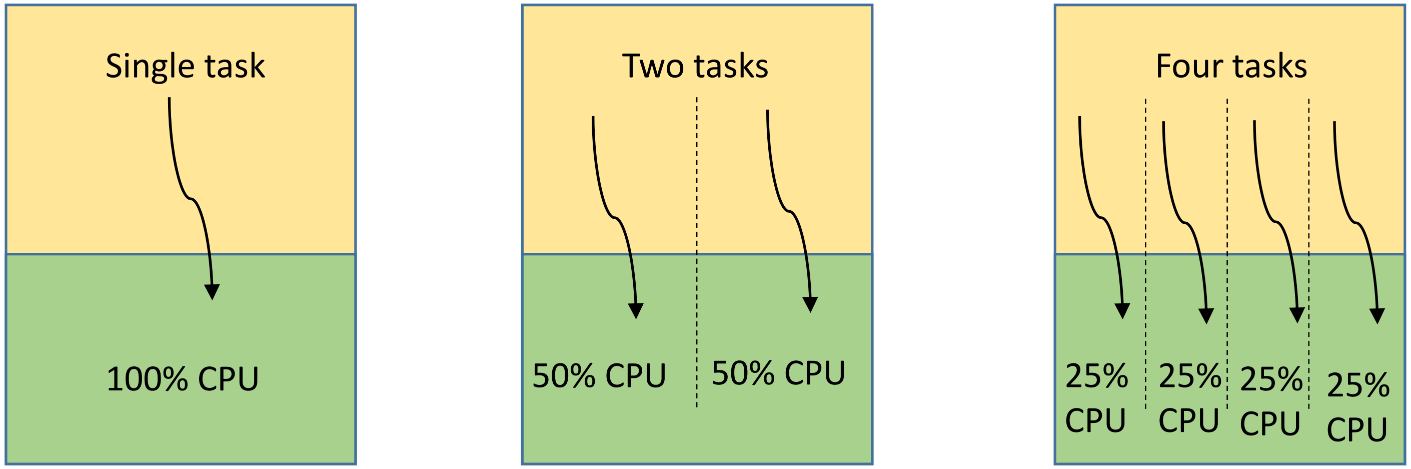
- Giving each task a fair share (portion) of the processor time (Completely Fair)
- Improving the interactive performance of scheduler for desktop.
- Introduces simple/efficient algorithmic approach (red-black tree) with O(log N).

CFS Features (cont)

- No time slices!... sort of
  - Uses wait_runtime (individual) and fair_clock (queue-wide)
  - Processes build up CPU debt
  - Different priorities “spend” time differently
  - Half priority task sees time pass twice as fast
- O(log n) complexity
  - Only marginally slower than O(1) at very large numbers of inputs

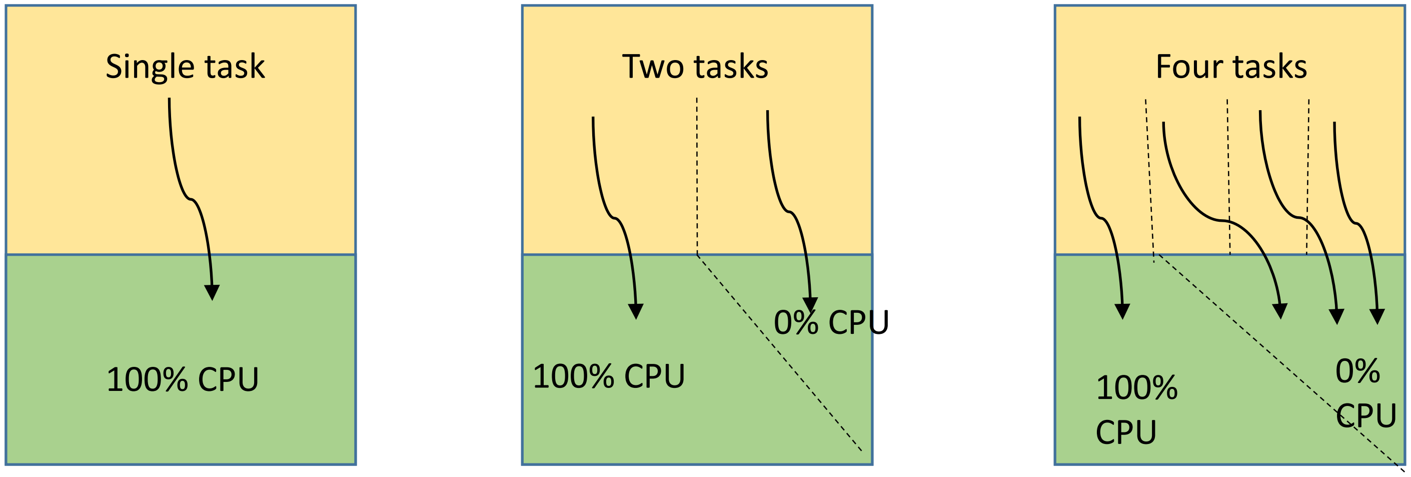
➢ Completely Fair Scheduler (CFS)

- CFS basically models an ideal, precise multitasking CPU'[5] on real hardware, that is, is a CPU that can run multiple processes at the same time (in parallel), giving each process an equal share of processor power(not time, but power).
- If a single process is running, it wold receive 100% of the processor's power. With two processes, each would have exactly 50% of the physical power (in parallel). Similarity, with four processes running, each would get 25% of physical CPU power in parallel and so on. Therefore, the CPU will be fair with all running processes



Completely Fair Scheduler (CFS)

- Obviously, this ideal CPU is not existent, but the CFS tries to emulate such a processor in software.
- On an actual real world processor, only one task can be allocated to a CPU at a particular time.
- Therefore, all other tasks wait during this period. So, while the currently running task gets 100% of the CPU power, all other tasks get 0% of the CPU power. This is obviously Not Fair



➢ CFS ,Red-Black tree

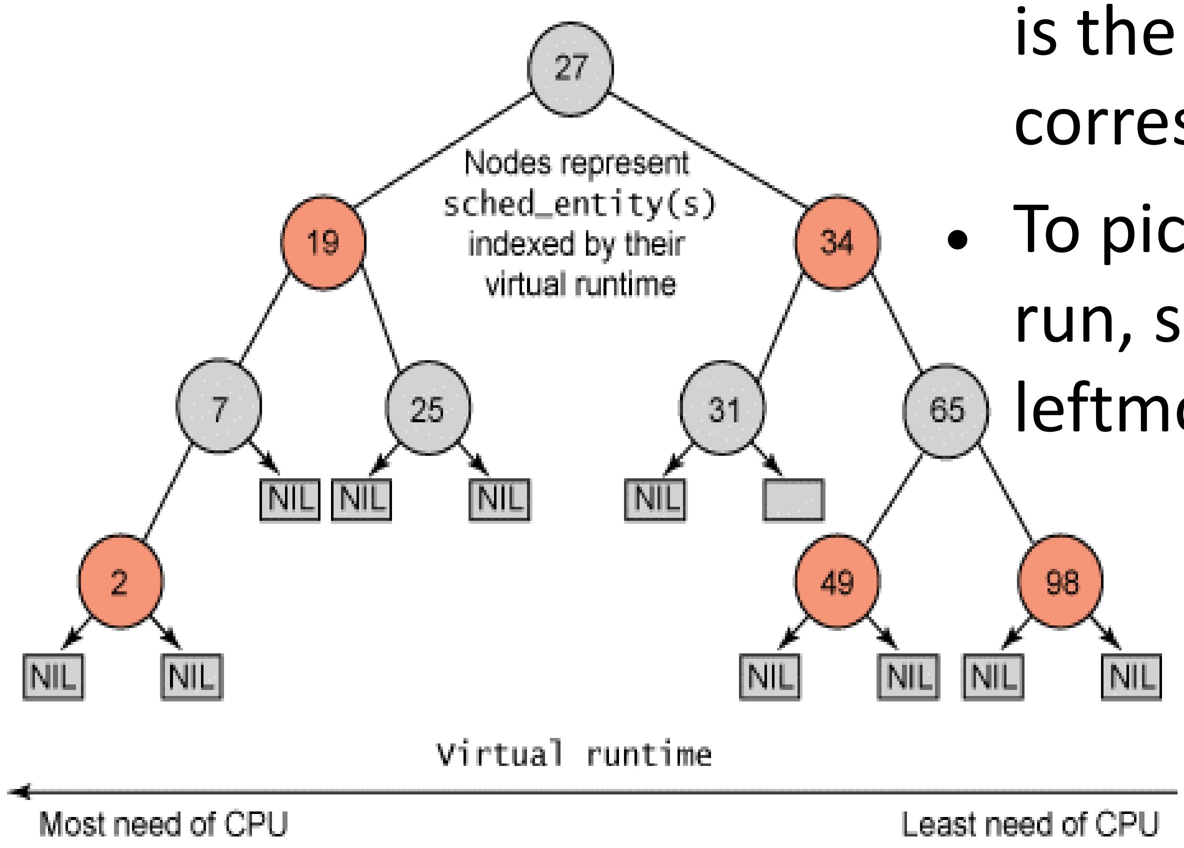
- Tasks are maintained in a time-ordered (i.e. *vruntime*) red-black tree for each CPU
- Red-Black Tree: Self-balancing binary search tree

  Balancing is preserved by painting each node with one of two colors in a way to satisfy certain properties. When the tree is modified , the new tree is rearranged and repainted to restore the coloring properties.

  The balancing of the tree can guarantee that no leaf can be more than twice as deep as others and the tree operations (searching/insertion/deletion/recoloring) can be performed in O(log N) time

- CFS will switch to the leftmost task in the tree, that is, the one with the lowest *virtual runtime* (most need for CPU) to maintain fairness.

Fedora CFS use Red-Black tree



- The key for each node is the vruntime of the corresponding task.
- To pick the next task to run, simply take the leftmost node.

➢ Earliest Deadline-first

- **An important class of scheduling algorithms is the class of dynamic priority algorithms**
  - In dynamic priority algorithms, the priority of a task can change during its execution
  - Fixed priority algorithms are a sub-class of the more general class of dynamic priority algorithms: the priority of a task does not change.
- **The most important (and analyzed) dynamic priority algorithm is Earliest Deadline First (EDF)**
  - The priority of a job (istance) is inversely proportional to its absolute deadline;
  - In other words, the highest priority job is the one with the earliest deadline;
  - If two tasks have the same absolute deadlines, chose one of the two at random (ties can be broken arbitrarily).
  - The priority is dynamic since it changes for different jobs of the same task.

Earliest Deadline-first

- Theorem Given a task set of periodic or sporadic tasks, with relative deadlines equal to periods, the task set is schedulable by EDF if and only if

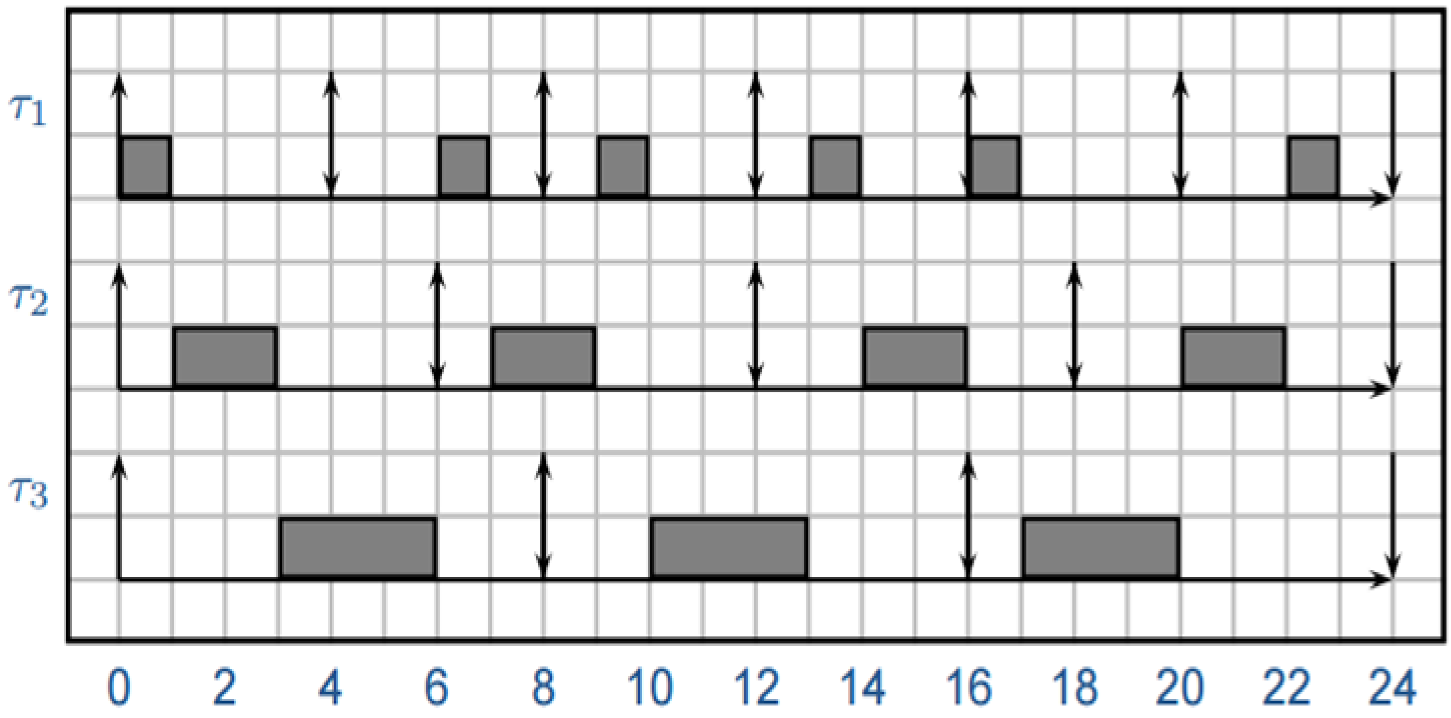
$$U = \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1$$

Where the C_i are the worst-case computation time of the n and the T is inter arrival periods and U is the utilization

- Corollary EDF is an optimal algorithm, in the sense that if a task set if schedulable, then it is schedulable by EDF.
  - In fact, if U > 1 no algorithm can successfully schedule the task set;
  - if U ≤ 1, then the task set is schedulable by EDF (and maybe by other algorithms).
- In particular, EDF can schedule all task sets that can be scheduled by FP, but not vice versa.
- Notice also that offsets are not relevant!

➢ Example: scheduling with EDF

- $\tau_1 = (1, 4)$, $\tau_2 = (2, 6)$, $\tau_3 = (3, 8)$.
- $U = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} = \frac{23}{24}$
- Again, the utilization is very high. However, no deadline miss in the hyperperiod.



➢ Summery

The sum up, of CPU Scheduling algorithms in any OS, is to make task processing more efficiency and systematically through, giving each task a fair time to do the process. Thus, make the OS users more comfortable though feeling that all tasks done at the same time.

The future of CPU Scheduling development is still opening in order to face the challenges of computer and digital world growing.

➢ References

- http://www.studytonight.com/operating-system/cpu-scheduling
- http://web.cse.ohio-
- https://lwn.net/Articles/240474/
- state.edu/~agrawal.28/660/Slides/jan16.pdf
- http://www.ayomaonline.com/academic/cpu-scheduling/
  - https://superuser.com/questions/414604/difference-between-the-windows-and-linux-thread-scheduler
- https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt