【OS】Day15(2)

【Ch9】Scheduling: Proportional Share Scheduling

9.7 The Linux Completely Fair Scheduling(2)

Using Red-Black Trees

For a scheduler, there are many facets of efficiency, but one of them is as simple as this: when the scheduler has to find the next job to run, it should do so as quickly as possible.

CFS addresses this by keeping processes in a red-black tree.

CFS does not keep all process in this structure rather, only running(or runnable) processes are kept therein. If a process goes to sleep(say, waiting on an I/O to complete), it is removed from the tree and kept track of elsewhere.

Keeping the same values in a red-black tree makes most operations more efficient. Processes are ordered in the tree by `vruntime`, and most operations are logarithmic in time(O(log n)). When n is in the thousands, logarithmic is noticeably more efficient than linear.

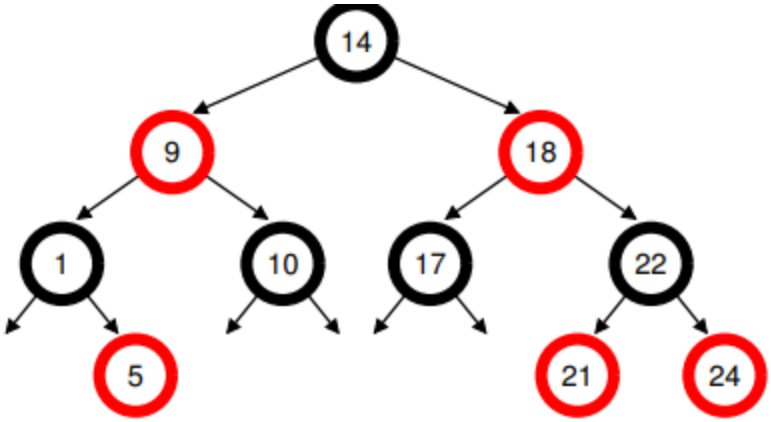

Figure 9.5: **CFS Red-Black Tree**

Dealing With I/O And Sleeping Processes

One problem with picking the lowest `vruntime` is to run next arises with jobs that have gone to sleep for a long period of time. Imagine we have a process who has slept a long period of time, when it wakes up, its `vruntime` will be far behind other processes, and thus will monopolize the CPU.

CFS handles this case by altering the `vruntime` of a job when it wakes up. Specifically, CFS sets the `vruntime` of that job to the minimum value found in the tree. In this way, CFS avoids starvation, but not without a cost: jobs that sleep for short periods of time frequently do not ever get their fair share of the CPU.