COMP124

Computer Systems

10 | Process Scheduling | Average Wait Times

Dr Stuart Thomason

Process Scheduling

- The processor manager uses one or more scheduling algorithms to despatch processes
  - Looks at all processes in the ready state
  - Decides which one gets the CPU
  - Good algorithm will give impression of speed and responsiveness

- Non-preemptive scheduling…
  - Process stays on the CPU until it terminates or blocks for I/O
  - Implements multiprogramming
  - Problems with compute-bound and I/O-bound processes

- Preemptive scheduling…
  - Process can be interrupted and replaced with another process
  - Implements multitasking

Scheduling Policies

- Operating system will have a scheduling policy according to some criteria
  - Maximise throughput – Complete as many processes as possible in a given period
  - Minimise turnaround time – Execute entire process as quickly as possible
  - Minimise waiting time – Reduce time processes spend in the ready state
  - Maximise CPU efficiency – Keep the CPU busy at all times
  - Minimise latency – Reduce time taken between request and response
- It's not possible to meet all criteria all the time, as some will conflict with each other
- The processor manager tries to ensure fairness for all processes
  - Give each process an equal amount of CPU and I/O time
  - Ensure processes don't hog the CPU and cause others to starve

Scheduling Algorithms

- This module will look at some simple scheduling algorithms
  - Advantages and disadvantages
  - Calculations to show how the scheduler picks processes
- In this lecture…
  - First come first served (FCFS)
  - Shortest job first (SJF)
  - Shortest remaining time first (SRTF)
- In the next lecture…
  - Priority scheduling
  - Round robin scheduling

First Come First Served

- This is the easiest algorithm to implement and understand
- Non-preemptive algorithm that deals with processes according to their arrival time
  - Uses a FIFO queue to store ready processes
  - The sooner a process arrives, the sooner it gets the CPU
  - Once a process gets the CPU, it stays there until it terminates or blocks for I/O

- When a new process is created, its PCB is added to the back of the ready queue

- When CPU becomes free, the process at front of queue is despatched to it
  - Its PCB is restored to the CPU
  - Instruction pointer will be restored as part of this context switch
  - Fetch-execute cycle continues as usual

First Come First Served – Example

- The burst time of a process is the time it takes on the CPU before it blocks or terminates
- The arrival time is the time a process is first put into the ready state

- We have three processes with the following burst times, and they arrive in this order

  - P1 with burst of 13 ms
  - P2 with burst of 5 ms
  - P3 with burst of 1 ms

- We can view the process schedule as a Gantt chart showing cumulative execution time

First Come First Served – Average Wait Time

- We can calculate the average wait time for this schedule
  - Use the Gantt chart to find out how long each process had to wait
  - Remember to include the first process that waited for 0 ms

  - P1 waited for 0 ms
  - P2 waited for 13 ms
  - P3 waited for 18 ms

- Average wait time = (0 + 13 + 18) / 3 = 10.3 ms

First Come First Served – Different Arrival Order

- Order of arrival can have a dramatic impact on the average wait time
- If the same three processes arrived in a different order…

  - P2 with burst of 5 ms
  - P3 with burst of 1 ms
  - P1 with burst of 13 ms

- Average wait time = (0 + 5 + 6) / 3 = 3.7 ms
- Much lower because the biggest process arrived last

First Come First Served – Advantages & Disadvantages

- Advantages…
  - Very simple to implement as a FIFO queue
  - Scheduling overhead is minimal during a context switch
  - No starvation of processes

- Disadvantages…
  - Throughput can be low due to long processes staying on CPU
  - Average wait time is not minimal and can vary substantially
  - Turnaround time and latency are hard to predict
  - No ability to prioritise processes

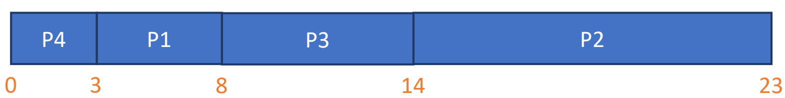
- We could achieve better all-round performance if we could order the schedule so that large processes always go last

Shortest Job First

- Non-preemptive algorithm that deals with processes according to their burst time
  - When CPU becomes free, scheduler chooses process with shortest burst time
  - If two processes have same burst time, choose according to arrival time (FCFS)
  - Once a process gets the CPU, it stays there until it terminates or blocks for I/O

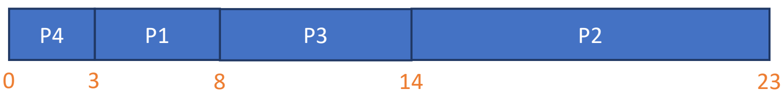
- We have four processes in the ready state, with these burst times

  - P1 with burst of 5 ms
  - P2 with burst of 9 ms
  - P3 with burst of 6 ms
  - P4 with burst of 3 ms



Shortest Job First – Average Wait Time

- The average wait time will be minimised because short jobs are scheduled first
  - P1 waited for 3 ms
  - P2 waited for 14 ms
  - P3 waited for 8 ms
  - P4 waited for 0 ms
- Average wait time = (3 + 14 + 8 + 0) / 4 = 6.25 ms



Compare SJF with FCFS

- What would the average wait time have been if we used first come first served?
- Assume the processes arrived in the order listed
  - P1 with burst of 5 ms
  - P2 with burst of 9 ms
  - P3 with burst of 6 ms
  - P4 with burst of 3 ms
- Average wait time = (0 + 5 + 14 + 20) / 4 = 9.75 ms
- So we can see that shortest job first gives a better average wait time

Shortest Job First – Advantages & Disadvantages

- Advantages…
  - Reduces overall average wait time
  - Provably optimal in giving minimal average wait time for a given set of processes

- Disadvantages…
  - Can lead to process starvation
  - Difficult to estimate burst times for new processes (often relies on prior history)

- In general, non-preemptive algorithms are not suitable in modern operating systems
  - Many processes will be in the ready state
  - All need fair access to the CPU
  - Large processes would tie up the CPU and reduce overall system performance
  - Remember the OS itself has many processes that need time on the CPU

Shortest Remaining Time First

- Remember that processes are being created all the time
  - The ready state is not a fixed set of processes
  - New processes will appear and impact the scheduling algorithm

- Shortest remaining time first is a preemptive version of shortest job first
  - CPU is allocated to process that is closest to finishing
  - If a new process arrives that is shorter than current running process, interrupt it
  - Not easy to draw Gantt chart for preemptive algorithms

- Remember the OS can only make a best guess about burst times and remaining times
  - Can be based on previous performance and estimates
  - Won't always be accurate

Shortest Remaining Time First

- Advantages…
  - Short processes are handled very quickly
  - Gives maximum throughput in most situations
  - Requires little overhead during context switch

- Disadvantages…
  - Starvation is still possible
  - Introduces extra context switches
  - Waiting time and latency increase for longer processes

- A better approach is to use a preemptive scheduling algorithm with a fixed time slice
  - Implement multitasking with a fixed quantum and clock interrupts
  - Still need to decide which process to schedule next