CS153: Scheduling

Chengyu Song

Slides modified from
Harsha Madhyvasta, Nael Abu-Ghazaleh, and Zhiyun Qian
Administrivia

- **Lab**
  - Session enrollment in progress
  - Groups forming in progress
  - Lab1 will be released after class
    - Walkthrough: **April 23**
    - All deliverables: **April 30**
Overview

• Scheduler runs during context switching to pick which process/thread runs next
  • Under what situations does this occur?
  • What should it do? Does it matter?
• Making this decision is called **scheduling**
• Today's content
  • The goals and challenges of scheduling
  • Various well-known scheduling algorithms
Recall: multiprogramming

- Multiprogramming overlaps I/O and CPU execution to increase CPU utilization and job throughput
  - Doing this requires a combination of mechanisms and policy
- We have covered the mechanisms
  - Context switching, how and when it happens
  - Process queues and process states
- Now we’ll look at the policies
Goals

• Long-term scheduling: the number of jobs loaded into memory, i.e., multiprogramming level
  • Swapping: moving jobs to/from memory
  • Infrequent
• **Short-term scheduling**: which job to run next to guarantee "good service"
  • Good service could be one of many different criteria
  • Frequent
Good service metrics

- Q: what are criteria of good service?
  - Resources utilization: CPU
  - Job throughput: \((\text{number of jobs}) / (\text{unit time})\)
  - Turnaround time: \(T_{\text{finish}} - T_{\text{start}}\)
  - Waiting time: average time spent in wait queues \((\text{Avg}(T_{\text{wait}}))\)
  - Response time: average time spent in ready queue \((\text{Avg}(T_{\text{ready}}))\)
  - Fairness
  - etc.
Good service metrics (cont.)

• Batch systems
  • Strive for job throughput, turnaround time (supercomputers)

• Interactive systems
  • Strive to minimize response time for interactive jobs (PC, smartphones)
    • iPhone vs. older Android (< 4.1)
Workload characteristic

• How much do we know about the workload?
  • Types: I/O, UI, ...
  • Priority
  • Past: how long has a task been running, ...
  • Present: how much memory used, ...
  • Future?: percentage of completion
When

- In general, the scheduler runs
  - When a job switches from running to waiting
  - When an interrupt occurs
    - Software interrupt (e.g., syscalls)
    - Hardware interrupt (e.g., timer)
  - Why?
  - When a job is created or terminated
Styles

• We’ll discuss scheduling algorithms in two contexts
  • In **preemptive** systems, the scheduler can interrupt a running job
    (involuntary context switch)
  • In **non-preemptive** systems, the scheduler waits for a running job to
    explicitly block (voluntary context switch)

• **Note**: difference between **preemptive kernel**
  • Preemptive kernel means whether the kernel itself can be interrupted
    during events handling
Starvation

• Starvation is a scheduling "**non-goal**"

• Starvation is a situation where a process is prevented from making progress because some other process has the resource it requires
  • Resource could be the CPU, or a lock (recall readers/writers)

• **Starvation usually a side effect of the scheduling algorithm**
  • A "high priority" process prevents a "low priority" process from running
  • One thread always beats another when acquiring a lock

• **Starvation can be a side effect of synchronization**
First In First Out (FIFO)

- Schedule tasks in the order they arrive
  - Continue running them until they complete or give up the processor (i.e., non-preemptive)
- Example: queues
  - Supermarket, banks, drive-through, etc.
- On what workloads is FIFO particularly bad?
  - Imagine being at supermarket to buy a drink of water, but get stuck behind someone with a huge cart (or two!) ... and who pays in pennies!
Shortest Job First (SJF)

• Always do the task that has the shortest remaining amount of work to do
  • Preemptive SJF is called shortest remaining time first (SRTF)
  • Express lane in the supermarket

• **Assumption**: we know how much amount of work is left
FIFO vs. SJF

• Suppose we have five tasks arrive one right after each other, but the first one is much longer than the others
  • Which completes first in FIFO? Next?
  • Which completes first in SJF? Next?
FIFO vs. SJF (completion)
FIFO vs. SJF (ATT)

- What's the big deal? Don’t they finish at the same time?
- How about other metrics like **Average Turnaround Time**?

**FIFO:**

\[ ATT = \frac{8 + (8+4)+(8+4+2)}{3} = 11.33 \]

**SJF:**

\[ ATT = \frac{2 + (2+4)+(2+4+8)}{3} = 7.33 \]
FIFO vs. SJF (AWT)

• How about **Average Wait Time**?

FIFO:

\[
ART = \frac{(0 + 8 + (8+4))}{3} = 6.67
\]

\[
ART = \frac{(0 + 4 + (4+8))}{3} = 5.33
\]

\[
ART = \frac{(0 + 4 + (4+2))}{3} = 3.33
\]

SJF:

\[
ART = \frac{(0 + 2 + (2+4))}{3} = 2.67
\]
FIFO vs. SJF (questions)

- Claim: SJF is optimal for average response time
  - Why?
- For what workloads is FIFO optimal?
  - For what is it pessimal (i.e., worst)?
- Does SJF have any downsides?
  - Does it work in a supermarket?
SJF (problems)

- Relies on the assumption to know the amount of work left
  - Impossible to know size of CPU burst
    - Like choosing person in line without looking inside basket/cart
    - How can you make a reasonable guess?
- Can potentially starve
Round Robin

• Each task gets resource for a fixed period of time (time quantum)
  • If task doesn't complete, it goes back in line

• Need to pick a time quantum
  • What if time quantum is too long?
    - Infinite?
  • What if time quantum is too short?
    • One instruction?
Round Robin (cont.)

Tasks

Round Robin (1 ms time slice)

(1) 

rest of task 1

(2)

(3)

(4)

(5)

Round Robin (100 ms time slice)

(1)

rest of task 1

(2)

(3)

(4)

(5)

Time
RR vs. FIFO vs. SJF

• Many context switches can be costly
• Other than that, is Round Robin always better than FIFO and SJF?
  • Completion time, fairness, ATT, AWT, ...
RR vs. FIFO vs. SJF

Tasks
(1)  
(2)  
(3)  
(4)  
(5)  

Round Robin (1 ms time slice)

FIFO and SJF

(1)  
(2)  
(3)  
(4)  
(5)  

Time
Summary

- Mechanisms and policy
- Good services metrics
  - Utilization, throughput, ATT, AWT, fairness, etc
- Styles
  - Preemptive, non-preemptive
- Simple scheduling algorithms
  - FIFO, SJF, RR
Additional question to ponder

- What about mixed workload
For next class ...

- More scheduling
- Xv6 book
  - Chapter 5