

CS 370: OPERATING SYSTEMS
[CPU SCHEDULING]

Shrideep Pallickara
Computer Science
Colorado State University

October 9, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.1

Frequently asked questions from the previous class survey

- Could we record burst times in the PCB and use this to inform scheduling?
- Could a higher priority process starve?
- How is SJF implemented if we don't know burst times?

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.2

Topics covered in this lecture

- Scheduling Algorithms
 - Wrap-up of SJF
 - Priority Scheduling
 - Round robin scheduling
 - Lottery scheduling
 - Multilevel feedback queues
 - Multiprocessor/core Environments

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.3

SHORTEST JOB FIRST (SJF)

October 9, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.4

Use of SJF in long term schedulers

- Length of the process time limit
 - Used as CPU burst estimate
- Motivate users to accurately estimate time limit
 - Lower value will give faster response times
 - Too low a value?
 - Time limit exceeded error
 - Requires resubmission!

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.5

The SJF algorithm and short term schedulers

- **No way to know** the length of the next CPU burst
- So try to **predict** it
- Processes scheduled *based on predicted* CPU bursts

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.6

Prediction of CPU bursts: Make estimates based on past behavior

- t_n : Length of the n^{th} CPU burst
- τ_n : Estimate for the n^{th} CPU burst
- α : Controls weight of recent and past history
- $\tau_{n+1} = \alpha t_n + (1-\alpha) \tau_n$
- Burst is predicted as an exponential average of the measured lengths of previous CPU bursts

October 9, 2018
 Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.7

α controls the relative weight of recent and past history

- $\tau_{n+1} = \alpha t_n + (1-\alpha) \tau_n$
- Value of t_n contains our most recent information, while τ_n stores the past history
- $\tau_{n+1} = \alpha t_n + (1-\alpha) \alpha t_{n-1} + \dots + (1-\alpha)^j \alpha t_{n-j} + \dots + (1-\alpha)^{n+1} \alpha t_0$
- α is less than 1, $(1-\alpha)$ is also less than one
- Each successive term has less weight than its predecessor

October 9, 2018
 Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.8

The choice of α in our predictive equation

- If $\alpha = 1/2$
 - Recent history and past history are **equally weighted**
- With $\alpha = 1/2$; successive estimates of τ

$$t_0/2 \quad t_0/4 + t_1/2 \quad t_0/8 + t_1/4 + t_2/2 \quad t_0/16 + t_1/8 + t_2/4 + t_3/2$$
 - By the 3rd estimate, weight of what was observed at t_0 has dropped to $1/8$.

October 9, 2018
 Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.9

An example: Predicting the length of the next CPU burst

October 9, 2018
 Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.10

The choice of α in our predictive equation

- $\tau_{n+1} = \alpha t_n + (1-\alpha) \tau_n$
- If $\alpha = 0$, $\tau_{n+1} = \tau_n$
 - Current conditions are transient
- If $\alpha = 1$, $\tau_{n+1} = t_n$
 - Only most recent bursts matter
 - History is assumed to be old and irrelevant

October 9, 2018
 Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.11

PRIORITY SCHEDULING

October 9, 2018

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.12

Priority Scheduling

- Priority associated with each process
- CPU allocated to process with **highest** priority
- Can be preemptive or nonpreemptive
 - If preemptive: Preempt CPU from a lower priority process when a higher one is ready

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.13

Depiction of priority scheduling in action

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Here: Lower number means higher priority

Wait time = $(6 + 0 + 16 + 18 + 1) / 5 = 8.2$

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.14

How priorities are set

- Internally defined priorities based on:
 - Measured quantities
 - Time limits, memory requirements, # of open files, ratio (averages) of I/O to CPU burst
- External priorities
 - Criteria outside the purview of the OS
 - Importance of process, \$ paid for usage, politics, etc.

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.15

Issue with priority scheduling

- Can leave lower priority processes waiting indefinitely
- Perhaps apocryphal tale:
 - MIT's IBM 7094 shutdown (1973) found processes from 1967!

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.16

Coping with issues in priority scheduling: Aging

- Gradually increase priority of processes that wait for a long time
- Example:
 - Process starts with a priority of 127 and decrements every 15 minutes
 - Process priority becomes 0 in no more than 32 hours

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.17

Can SJF be thought of as a priority algorithm?

- Priority is **inverse** of CPU burst
- The larger the burst, the lower the priority
 - Note: The number we assign to represent priority levels may vary from system to system

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.18

ROUND ROBIN SCHEDULING

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.19

Round-Robin Scheduling

- Similar to FCFS scheduling
 - ▣ **Preemption** to enable switch between processes
- Ready queue is implemented as **FIFO**
 - ▣ Process Entry: PCB at *tail* of queue
 - ▣ Process chosen: From *head* of the queue
- CPU scheduler goes around ready queue
 - ▣ Allocates CPU to each process *one after the other*
 - CPU-bound up to a maximum of 1 **quantum**

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.20

Round Robin: Choosing the quantum

- Context switch is **time consuming**
 - ▣ Saving and loading registers and memory maps
 - ▣ Updating tables
 - ▣ Flushing and reloading memory cache
- What if quantum is 4 ms and context switch overhead is 1 ms?
 - ▣ 20% of CPU time thrown away in administrative overhead

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.21

Round Robin: Improving efficiency by increasing quantum

- Let's say quantum is 100 ms and context-switch is 1ms
 - ▣ Now wasted time is only 1%
- But what if 50 concurrent requests come in?
 - ▣ Each with widely varying CPU requirements
 - ▣ 1st one starts immediately, 2nd one 100 ms later, ...
 - ▣ The last one may have to wait for 5 seconds!
 - ▣ A shorter quantum would have given them better service

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.22

If quantum is set longer than mean CPU burst?

- **Preemption will not happen very often**
- Most processes will perform a blocking operation before quantum runs out
- Switches happens only when process blocks and cannot continue

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.23

Quantum: Summarizing the possibilities

- Too short?
 - ▣ Too *many* context switches
 - ▣ **Lowers** CPU efficiency
- Too long?
 - ▣ **Poor** responses to interactive requests

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.24

A round-robin analogy

- Hyperkinetic student studying for multiple exams simultaneously
 - If you switch between paragraphs of different textbooks? [Quantum is too short]
 - You won't get much done
 - If you never switch? [Quantum is too long]
 - You never get around to studying for some of the courses

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.25

LOTTERY SCHEDULING

October 9, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.26

Lottery scheduling

- Give processes **lottery tickets** for various system resources
 - E.g. CPU time
- When a scheduling decision has to be made
 - Lottery ticket is *chosen at random*
 - Process holding **ticket gets** the resource

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.27

All processes are equal, but some processes are more equal than others

- More important processes are given **extra tickets**
 - Increase their odds of winning
- Let's say there are 100 outstanding tickets
 - 1 process holds 20 of these
 - Has 20% chance of winning each lottery
- A process holding a fraction f of tickets
 - Will get about a fraction f of the resource

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.28

Lottery Scheduling: Properties [1/2]

- Highly **responsive**
 - Chance of winning is proportional to tickets
- Cooperating processes may **exchange** tickets
 - Process **A** sends request to **B**, and then hands **B** all its tickets for a faster response
- Avoids starvation
 - Each process holds at least one ticket Is guaranteed to have a non-zero probability of being scheduled

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.29

Lottery Scheduling: Properties [2/2]

- Solves problems that are *difficult to handle* in other scheduling algorithms
- E.g. video server that is managing processes that feed video frames to clients
 - Clients need frames at 10, 20, and 25 frames/sec
 - Allocate processes 10, 20 and 25 tickets
 - CPU divided into approximately 10:20:25

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.30

Most commercial OS including Windows, MacOS,
and Linux use this scheduling algorithm

MULTI-LEVEL FEEDBACK QUEUES (MFQ)

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.31

MFQ is designed to achieve several simultaneous goals

- **Responsiveness:** Run short tasks quickly as in SJF
- **Low Overhead:** Minimize number of preemptions, as in FIFO
 - Minimize time spent making scheduling decisions
- **Starvation-Freedom**
 - All tasks should make progress, as in Round Robin
- **Background tasks**
 - Defer system maintenance tasks, such as defragmentation, so they do not interfere with user work
- **Fairness**

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.32

Does MFQ achieve all of these?

- As with any real system that must **balance** several, conflicting goals ...
 - MFQ does not perfectly achieve any of these goals
- MFQ is intended to be a **reasonable compromise** in most real-world cases

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.33

MFQ

- Extension of round robin
- Instead of only a single queue, MFQ has **multiple round robin queues**
 - Each queue has a **different priority level** and **time quanta**

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.34

Tasks and priorities

- Tasks at a higher priority **preempt** lower priority tasks
- Tasks at the same priority level are scheduled in **round robin** fashion
- Higher priority tasks have **shorter** time quanta than lower priority tasks

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.35

MFQ: Example with 4 priority levels

Priority	Time Slice (ms)	Round Robin Queues
1	10	
2	20	
3	40	
4	60	

October 9, 2018 CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University L15.36

Task movements and priority

- Tasks are moved between priority levels to favor short tasks over long ones
- Every time a task uses up its time quantum?
 - It **drops** a priority level
- Every time task yields the processor because it is waiting on I/O?
 - It **stays** at the same level, or is **bumped up** a level
- If the task completes ... it leaves

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.37

Impact on CPU and I/O bound processes

- A **new** CPU bound process will start as **high priority**
 - But it will quickly exhaust its time quantum and fall to the next lower priority, and then the next ...
- An I/O bound process with a modest amount of computing
 - Will always be **scheduled quickly**
 - Also, keeps the disk busy
- Compute bound tasks **run with a long time quantum** to minimize switching overhead while sharing processor

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.38

What about starvation and fairness?

- If there are too many I/O bound tasks, the compute bound tasks may receive no time on the processor
- MFQ scheduler **monitors every process** to ensure that it is receiving its fair share
 - For e.g. at each level, Linux actually maintains two queues
 - Tasks whose processes have already reached their fair share are only scheduled if other processes at the same level have also received their fair share
- Periodically, processes not receiving their fair share have their tasks increased in priority
 - Tasks that receive more than their fair share have their priority reduced

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.39

Adjusting priority addresses strategic behavior

- From a selfish point of view, a task can keep its priority high by doing a short I/O request just before its quantum expires
 - With MFQ this will be detected, and its priority reduced to its fair level

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.40

IDLE THREADS

October 9, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.41

Dispatcher in Windows XP

- Use a **queue** for each scheduling priority
- **Traverse** the queues from highest to lowest
 - **Until** it finds a thread that is ready to run
- If no ready thread is found?
 - Dispatcher will execute a special thread: **idle thread**

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.42

Idle thread in Windows

- Primary purpose is to **eliminate a special case**
 - Cases when no threads are runnable or ready
 - Idle threads are always in a **ready** state
 - If not already running
- Scheduler can always find a thread to execute
- If there are other eligible threads?
 - Scheduler will never select the idle thread

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.43

Idle threads in Windows

- Windows thread priorities go from 0-31
 - Idle thread priority can be thought of as -1
- Threads in the system idle process can also implement CPU power saving
 - On x86 processors, run a loop of **halt** instructions
 - Causes CPU to **turn off internal components**
 - Until an interrupt request arrives
 - Recent versions also **reduce the CPU clock speed**

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.44

Time consumed by the idle process

- It may seem that the idle process is monopolizing the CPU
 - It is merely acting as a **placeholder during free time**
 - Proof that no other process wants that CPU time

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.45

MULTIPROCESSOR/CORE ENVIRONMENTS

October 9, 2018

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.46

Load balancing: Migration based approaches

- Push migration
 - Specific task periodically checks for **imbalance**
 - Balances load by **pushing** processes from overloaded to less-busy processors.
- Pull migration
 - Idle processor pulls a waiting task from busy processor
- Schemes **not mutually exclusive**: used in parallel
 - Linux: Runs a load-balancing algorithm
 - Every 200 ms (**PUSH** migration)
 - When processor run-queue is empty (**PULL** migration)

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.47

Multicore processors place multiple processor cores on same physical chip

- Each core has its own register set
 - Appears to the OS as a separate physical processor
- Recent designs implement 2 or more hardware threads per core
 - If there is a memory stall (due to cache miss) on one thread, **switch** to another hardware thread

October 9, 2018
Professor: SHRIDEEP PALLICKARA

CS370: Operating Systems [Fall 2018]
Dept. Of Computer Science, Colorado State University

L15.48

Coping with memory stalls

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.49

Multithreading a processor

- **Coarse** grained
 - Thread executes on processor till a memory stall
 - Switch to another thread
- Switching between threads
 - **Flush** the instruction pipeline
 - **Refill** pipeline as new thread executes
- **Finer** grained (or interleaved)
 - Switch between threads at the boundary of an instruction cycle
 - Design includes logic for thread switching: overheads are low

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.50

Tiered scheduling on multicore processors

- **First-level: OS**
 - OS chooses which software thread to run on each hardware thread
- **Second-level: Core**
 - Decides which hardware thread to run
- **UltraSPARC T1**
 - 8 cores, and 4 hardware threads/core
 - Round robin to schedule hardware threads on core

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.51

The contents of this slide-set are based on the following references

- *Avi Silberschatz, Peter Galvin, Greg Gagne. Operating Systems Concepts, 9th edition. John Wiley & Sons, Inc. ISBN-13: 978-1118063330. [Chapter 6]*
- *Andrew S Tanenbaum and Herbert Bos. Modern Operating Systems. 4th Edition, 2014. Prentice Hall. ISBN: 013359162X/ 978-0133591620. [Chapter 2]*
- *Thomas Anderson and Michael Dahlin. Operating Systems Principles and Practice. 2nd Edition. ISBN: 978-0985673529. [Chapter 7]*

October 9, 2018
 Professor: SHRIDEEP PALICKARA

CS370: Operating Systems [Fall 2018]
 Dept. Of Computer Science, Colorado State University

L15.52