

Last Class: Threads and Scheduling

- **Thread:** sequential execution stream within a process
- Kernel threads versus user-level threads
- **Goals for Scheduling:**
 - Minimize average response time
 - Maximize throughput
 - Share CPU equally
 - Other goals?
- **Scheduling Algorithms:**
 - Selecting a scheduling algorithm is a policy decision
 - FCFS: simple, but typically fails to meet above goals

Today: More on Scheduling Algorithms

- Round Robin
- SJF
- Multilevel Feedback Queues
- Lottery Scheduling

Round Robin Scheduling

- Variants of round robin are used in most time sharing systems
- Add a timer and use a preemptive policy.
- After each time slice, move the running thread to the back of the queue.
- Selecting a time slice:
 - Too large - waiting time suffers, degenerates to FCFS if processes are never preempted.
 - Too small - throughput suffers because too much time is spent context switching.
- ⇒ Balance these tradeoffs by selecting a time slice where context switching is roughly 1% of the time slice.
- Today: typical time slice = 10-100 ms, context switch time = 0.1-1ms
- **Advantage:** It's fair; each job gets an equal shot at the CPU.
- **Disadvantage:** Average waiting time can be bad.

Round Robin Scheduling: Example 1

- 5 jobs, 100 seconds each, time slice 1 second, context switch time of 0

Job	length	FCFS	Round Robin	Wait Time	FCFS	Round Robin
1	100					
2	100					
3	100					
4	100					
5	100					
Average						

- **Disadvantages:**
 - Impossible to predict the amount of CPU time a job has left
 - Long running CPU bound jobs can starve
- **Advantages:**
 - Provably optimal with respect to minimizing the average waiting time
 - Works for preemptive and non-preemptive schedulers
 - Preemptive SJF is called SRTF - shortest remaining time first
 - ⇒ I/O bound jobs get priority over CPU bound jobs
- Schedule the job that has the least (expected) amount of work (CPU time) to do until its next I/O request or termination.

SJF/SRTF: Shortest Job First

- 5 jobs, of length 50, 40, 30, 20, and 10 seconds each, time slice 1 second, context switch time of 0 seconds

Job	length	FCFS	Round Robin	FCFS	Round Robin	Wait Time	Round Robin
1	50						
2	40						
3	30						
4	20						
5	10						
Average							

Round Robin Scheduling: Example 2

- This policy is **adaptive** because it relies on past behavior and changes in behavior result in changes to scheduling decisions.
 - To exploit this behavior, the scheduler can favor jobs that have used the least amount of CPU time, thus approximating SJF.
 - If a process is I/O bound in the past, it is also likely to be I/O bound in the future (programs turn out not to be random.)
- ⇒ overcome the prediction problem in SJF
- Multilevel feedback queues use past behavior to predict the future and assign job priorities

Multilevel Feedback Queues (MLFQ)

- 5 jobs, of length 50, 40, 30, 20, and 10 seconds each, time slice 1 second, context switch time of 0 seconds

Job length	Completion Time			Wait Time		
	FCFS	RR	SJF	FCFS	RR	SJF
1	50					
2	40					
3	30					
4	20					
5	10					
Average						

SJF: Example

- Job starts in highest priority queue.
 - If job's time slices expires, drop its priority one level.
 - If job's time slices does not expire (the context switch comes from an I/O request instead), then increase its priority one level, up to the top priority level.
- ⇒ CPU bound jobs drop like a rock in priority and I/O bound jobs stay at a high priority.

Adjusting Priorities in MLFQ

- Multiple queues with different priorities.
 - Use Round Robin scheduling at each priority level, running the jobs in highest priority queue first.
 - Once those finish, run jobs at the next highest priority queue, etc. (Can lead to starvation.)
 - Round robin time slice increases exponentially at lower priorities.
- | Priority | Time Slice |
|----------|------------|
| 1 | 1 |
| 2 | 2 |
| 3 | 4 |
| 4 | 8 |

Approximating SJF: Multilevel Feedback Queues

Multilevel Feedback Queues: Example 2

- 3 jobs, of length 30, 20, and 10 seconds, the 10 sec job has 1 sec of I/O every other sec, initial time slice 2 sec, context switch time of 0 sec, 2 queues.

Job	length	RR	MLFQ	RR	MLFQ	Wait Time
1	30					
2	20					
3	10					
Average						

Queue	Time Slice	Job
1	2	
2	4	

Multilevel Feedback Queues: Example 1

- 3 jobs, of length 30, 20, and 10 seconds each, initial time slice 1 second, context switch time of 0 seconds, all CPU bound (no I/O), 3 queues

Job	length	RR	MLFQ	RR	MLFQ	Wait Time
1	30					
2	20					
3	10					
Average						

Queue	Time Slice	Job
1	1	
2	2	
3	4	

- Degradates gracefully as load changes. Adding or deleting a job affects all jobs proportionately, independent of the number of tickets a job has.
- Assign tickets by giving the most to short running jobs, and fewer to long running jobs (approximating SJF). To avoid starvation, every job gets at least one ticket.
- On average, CPU time is proportional to the number of tickets given to each job.
- On each time slice, randomly pick a winning ticket.
- Give every job some number of lottery tickets.

Lottery Scheduling

- Possible solutions:
- Give each queue a fraction of the CPU time. This solution is only fair if there is an even distribution of jobs among queues.
 - Adjust the priority of jobs as they do not get serviced (Unix originally did this.) This ad hoc solution avoids starvation but average waiting time suffers when the system is overloaded because all the jobs end up with a high priority.
- Since SJF is optimal, but unfair, any increase in fairness by giving long jobs a fraction of the CPU when shorter jobs are available will degrade average waiting time.

Improving Fairness

FCFS: Not fair, and average waiting time is poor.
Round Robin: Fair, but average waiting time is poor.
SJF: Not fair, but average waiting time is minimized assuming we can accurately predict the length of the next CPU burst. Starvation is possible.
Multilevel Queuing: An implementation (approximation) of SJF.
Lottery Scheduling: Fairer with a low average waiting time, but less predictable.
 ⇒ Our modeling assumed that context switches took no time, which is unrealistic.

Summary of Scheduling Algorithms:

# short jobs / % of CPU each	# long jobs / % of CPU each	1/1	0/2	2/0	10/1	1/10
91%	9%					

- Short jobs get 10 tickets, long jobs get 1 ticket each.

Lottery Scheduling: Example

Job length	FCFS	Round Robin	Completion Time	FCFS	Round Robin	Wait Time
1	50	50	150	0	100	100
2	40	90	140	50	100	100
3	30	120	120	90	90	90
4	20	140	90	120	70	70
5	10	150	50	140	40	40
Average		110	110	80	80	80

- **Example:** 5 jobs, of length 50, 40, 30, 20, and 10 seconds each, time slice 1 second, context switch time of 0 seconds

Round Robin Scheduling: Example 2

Job length	FCFS	Round Robin	Completion Time	FCFS	Round Robin	Wait Time
1	100	100	496	0	396	396
2	100	200	497	100	397	397
3	100	300	498	200	398	398
4	100	400	499	300	399	399
5	100	500	500	400	400	400
Average		300	498	200	398	398

- **Example:** 5 jobs, 100 seconds each, time slice 1 second, context switch time of 0

Round Robin Scheduling: Example 1

# short jobs / # long jobs	1/1	91% (10/11)	9% (1/11)
% of CPU each short job gets	2/0	50% (10/20)	50% (1/2)
% of CPU each long job gets	10/1	10% (10/101)	> 1% (1/101)
	1/10	50% (10/20)	50% (1/20)

- Short jobs get 10 tickets, long jobs get 1 ticket each.

Lottery Scheduling Example

Job length	Completion Time		Wait Time	
	RR	MLFQ	RR	MLFQ
1	30	60	30	30
2	20	50	30	30
3	10	30	20	8
Average	46 2/3	45	26 2/3	25 1/3

Queue	Time Slice	Job	Job	
			1	2
1	1	Job	1	1
			2	1
			3	1
			4	1
			5	1
			6	1
			7	1
			8	1
			9	1
			10	1
2	2	Job	1	1
			2	1
			3	1
			4	1
			5	1
			6	1
			7	1
			8	1
			9	1
			10	1

- 3 jobs, of length 30, 20, and 10 seconds, the 10 sec job has with 1 sec of 1/0 every other sec, initial time slice 1 sec, context switch time of 0 sec, 2 queues
- This is one possible solution. Other solutions are possible depending on when the 10 sec job starts its 1/0.

Multilevel Feedback Queues: Example 2