1 – Intro to O/S Scheduling

List of Slides

1  Intro to O/S Scheduling
4  What is Scheduling?
5  Computer Systems Scheduling
6  O/S Scheduling Categories
7  O/S Scheduling and Process State
8  O/S Scheduling Layers
9  O/S Scheduling Layers
10 Scheduling Algorithm Criteria
11 Scheduling Algorithm Criteria
12 Introduction to Performance
13 Time as a Performance Metric
15 Rate as a Performance Metric
16 Response Time and Throughput
17 Utilization as a Performance Measurement
18 Queuing Notation
19 How Preemption Fits In
20 FCFS Defined
21 An FCFS Example
23 FCFS Analyzed
24 SJF (also called SPN)
25 SJF Example
27 SJF Features
28 Estimating Service Time
29 SRT
30 HRRN Scheduling
31 RR Scheduling
32 RR Example
33 RR $T_Q$ and System Turnaround Time
34 RR $T_Q$ and $E[T_q]$
35  RR Example $T_Q = 1$
36  RR Example $T_Q = 10$
37  RR Example $T_Q = 100$
38  RR Example $T_Q = 30$
39  RR Features Summary
40  Real Time Scheduling
41  OS Support For Real Time Scheduling
42  EDF Scheduling
43  Rate Monotonic Scheduling
2 – What is Scheduling?

Many definitions exist, our on-line Webster server came up with:

1. to appoint, assign, or designate for a fixed time

2. a procedural plan that indicates the time and sequence of each operation

Scheduling is an open research area, studied in engineering, decision sciences, operations research, and mathematics.

This lecture focuses on O/S level scheduling in uniprocessor computer systems.
3 – Computer Systems Scheduling

In a computer system typically there are:

1. *Servers* which respond to processes’ *requests* for service (e.g. I/O, CPU, Data Communication, adders, bus, etc.).

2. Requests are serviced in order of their *Priority*, with highest priority jobs going first.

3. A *preemption* occurs if a server:
   (a) Partially services a request, say *A*
   (b) Preserves partial results of service for *A*
   (c) Begins/resumes servicing another request with higher priority, say *B*
   (d) The server will later resume servicing *A*.

The computer systems scheduling primitives address these issues.
4 – O/S Scheduling Categories

Stallings [3] and Finkel [1] use similar O/S scheduling categories:

1. Long Term Scheduling — Which Jobs to admit for Processing
2. Medium Term Scheduling — Which process to load into memory
3. Short Term Scheduling — Which ready process gets the CPU
4. I/O Term Scheduling — Which pending I/O request does an I/O device service.
5 – O/S Scheduling and Process State

Figure 1: Scheduling in a Blocked/Suspended process model
6 – O/S Scheduling Layers

The long term scheduling primitives are supported by the “low level” short term scheduling primitives.

Figure 2: Scheduling Layers Transput denotes I/O
7 – O/S Scheduling Layers

Queuing discipline reflects scheduling.

Figure 3: Queuing models and Scheduling
8 – Scheduling Algorithm Criteria

The following issues impact scheduling algorithm selection:

1. Feasibility of computation
2. Feasibility of schedule
3. Efficiency of computation
4. Efficiency of schedule
5. Ease of implementation

Both qualitative and quantitative issues matter.
9 – Scheduling Algorithm Criteria

Stallings [3] gives the following scheduling criteria:

1. User Oriented - Performance
   (a) Response Time
   (b) Turnaround Time
   (c) Deadlines

2. User Oriented — Other Criteria — Predictability

3. Systemic Performance
   (a) Throughput
   (b) Processor Utilization

4. Systemic - Other Performance
   (a) Fairness
   (b) Enforcing Priorities
   (c) Balancing Resources
10 – Introduction to Performance

Performance denotes computational speed measurements and is sensitive to scheduling efficiency.

Jain [2] categorizes classic performance measures:

1. Time (e.g. response time)
2. Rate (e.g. throughput)
3. Resource (e.g. utilization)
11 – Time as a Performance Metric

There are intervals in user/computer interaction:

1. Response time — Time until either:
   (a) System Begins Response, or
   (b) System Ends Response

2. Reaction time — User Latency

3. Turnaround Time — Process Duration

4. Think Time — User Latency

5. Stretch Factor — Reaction of system to a fixed increased load
Figure 4: Definitions of Response Time
12 – Rate as a Performance Metric

Owners of computers want to know the \( \frac{\text{Operations}}{\text{time}} \) of the system:

1. Processing — MIPS, MFLOPS, GFLOPS, TFLOPS, Sustained, Peak
2. I/O — \( \frac{\text{Transactions}}{\text{Second}} \), Bandwidth
3. Network/Data Communications — Bandwidth, bps, fps, Mbps, pps
4. Efficiency — Measures \( \frac{\text{Achieved Throughput}}{\text{Available Throughput}} \).

Figure 5: Efficiency of a Multiprocessor System
13 – Response Time and Throughput

Figure 6: Response Time vs. Throughput
14 – Utilization as a Performance Measurement

Expensive computing resources should be used.

1. Idle Time — How long the resources are unused

2. Utilization — The ratio: \( \frac{\text{Time Used}}{\text{Total Time}} \)
15 – Queuing Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_q )</td>
<td>Turn Around Time</td>
</tr>
<tr>
<td>( T_s )</td>
<td>Service Time</td>
</tr>
</tbody>
</table>

Table 1: Stallings [3] Notation

![Diagram showing queuing notation with symbols and equations]

\[ T_w = T_q - T_s \]

Figure 7: Meaning of Time Measurements
16 – How Preemption Fits In

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_a$</td>
<td>Arrival Time in the system</td>
</tr>
<tr>
<td>$T_c$</td>
<td>Context Switch Cost</td>
</tr>
<tr>
<td>$T_w$</td>
<td>Time Waiting Queue</td>
</tr>
</tbody>
</table>

Table 2: Extended notation

*Preemption*

![Diagram](image)

Figure 8: Preemption and Time Measurements
First Come First Serve (FCFS) scheduling:

1. is nonpreemptive
2. uses a FIFO queue
18 – An FCFS Example

Figure 9: FCFS Example Time Line
<table>
<thead>
<tr>
<th>Jobs</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>$T_s$</td>
<td>$T_a$</td>
</tr>
<tr>
<td>A</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>210</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Example FCFS Job Schedules
The throughput is \( \frac{1}{E[T_q]} \) solving for \( E[T_q] \):

- Case 1: \( E[T_q] = \frac{280+260+210}{3} = 250 \)
- Case 2: \( E[T_q] = \frac{30+80+280}{3} = 130 \)

Some characteristics of FCFS scheduling:

1. Penalizes short jobs
2. Rewards long jobs
3. Large variance in throughput
4. Sensitive to order of arrival
5. Is starvation free
6. Easy to implement
Shortest Job First (SJF) is also called Shortest Process Next (SPN):

1. Services jobs in ascending $T_s$ order
2. Is nonpreemptive
21 – SJF Example

Consider the same jobs again:

Case 1

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A Runs</th>
<th>B Runs</th>
<th>C Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
<td>210</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Case 2

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A Runs</th>
<th>B Runs</th>
<th>C Runs</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
<td>210</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 10: SJF Example Time Line
<table>
<thead>
<tr>
<th>Jobs</th>
<th>Case 1</th>
<th>Case 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>$T_s$</td>
<td>$T_a$</td>
</tr>
<tr>
<td>A</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>210</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4: Example SJF Job Schedules
Recall that throughput is $\frac{1}{E[T_q]}$ solving for $E[T_q]$:

- Case 1: $E[T_q] = \frac{220 + 290 + 210}{3} = 240$
- Case 2: $E[T_q] = \frac{30 + 80 + 280}{3} = 140$

Some characteristics of SJF scheduling:

1. Penalizes long jobs
2. Rewards short jobs
3. Somewhat sensitive to order of arrival
4. Gives optimal nonpreemptive throughput
5. Permits Starvation
6. Knowing $T_s$ difficult
Let $S_n$ be the estimate for a job’s $n$th service time, $T_n$, for a process.

Let $\alpha, 0 \leq \alpha \leq 1$ be a tuning parameter.

\[
S_{n+1} = (1 - \alpha)S_n + \alpha T_n
\]

\[
= (1 - \alpha)^n S_0 + \sum_{i=1}^{n+1} (1 - \alpha)(n - i)\alpha T_{i-1}
\]

When computing the estimate $S_n$:

- $\alpha = 0$ disregard $T_n$ in estimate.

- $\alpha = 1$ only $T_n$ impacts estimate.

- $0 < \alpha < 1$ older $T_n$ and initial guess $S_0$ eventually get discounted (at an exponential rate).
24 – SRT

In SJF servicing long jobs forces shorter newly arrived jobs to wait.

Shortest Remaining Time (SRT) is a preemptive version of SJF.

1. Service Times may be estimated.
2. Favor jobs with smaller $T_Q$.

The heuristic presented may be used to estimate $T_n$. 
25 – HRRN Scheduling

Highest Response Ratio Next (HRRN):

1. is non preemptive
2. is starvation free
3. jobs are serviced in descending Response Ratio \(R_R\) order.
4. estimating service time, \(T_s\), may be required.
5. Requires Frequent Recomputation of Priorities
6. is more fair than FCFS

Response Ratio for a given processes \(n\)th service of duration \(T_n\) after waiting \(w\) duration for service is:

\[
R_R(n) = \frac{w + T_n}{T_n}
\] (1)

Estimating \(T_n\) can be done as shown earlier.
Round Robin (RR) scheduling:

1. Is Preemptive
2. Services jobs in FIFO order

The max time to preemption is the *quantum*, $T_Q$. 
Let the context switch time be $T_c = 5$.

<table>
<thead>
<tr>
<th>Id</th>
<th>$T_s$</th>
<th>$T_a$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>60</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>210</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5: RR Job Example

See how $0 < T_Q \leq 210$ impacts performance.
28 – RR $T_Q$ and System Turnaround Time

Preemption increases system turnaround time ($\max\{T_q\}$) rewards large $T_Q$ (FCFS approximation).

Figure 11: System Turnaround Time as a function of $T_Q$
The mean turnaround time for a job $E[T_q]$
30 – RR Example $T_Q = 1$

$T_Q = 1$ is far too small a time slice, usually $T_Q \gg T_c$.

![Graph showing RR, Quantum = 1](rr1.dat)

Figure 13: RR Scheduling Example for $T_Q = 1$
$T_Q = 10$ is still a bit too small a time slice.

Figure 14: RR Scheduling Example for $T_Q = 1$
$T_Q = 100$ is a bit too large to be a good time slice, typically $T_Q \leq E[T_q]$.

Figure 15: RR Scheduling Example for $T_Q = 100$
$T_Q = 30$ seems to be a good time slice.

Figure 16: RR Scheduling Example for $T_Q = 30$
34 – RR Features Summary

Some characteristics of RR scheduling:

1. Preemptive (at quantum $T_Q$).
2. Less Sensitive to order of arrival.
3. Quantum should not be too small $T_Q \gg T_c$.
4. Sensitive to overly large $T_Q$ (approximates FCFS).
5. Low overhead.
Some Notation:

- Jobs which must be completed at a specific time for correct completion have a \textit{deadline}.

- \textit{Real time scheduling} is the task of scheduling jobs in a deadline oriented system.

Categories of Real Time Scheduling:

- Hard — Guarantees completions of tasks by their deadlines (or fails), may require:
  - \textit{Admission control} — restrict access to jobs that will not violate scheduling constraints
  - Advance estimates of resource requirements of jobs.

- Soft — Ensures high priority tasks are not preempted by lower priority processes.
Real time systems require a more complete approach than just replacing the scheduler. Some approaches include:

- Cooperative Multitasking — Application programmer explicitly relinquishes CPU at *preemption points*.
  - Error prone (e.g. a hung job can hang the system).

- Make the kernel preemptible, which is hard because:
  - The kernel needs to be made *reentrant*.
  - The stalled process may have its resources reallocated to another (higher priority job) and needs to reacquire them at wakeup.
37 – EDF Scheduling

Earliest Deadline First (EDF) is a real time scheduling algorithm:

1. can be preemptive
2. is starvation free
3. Deadlines are user defined.
4. jobs are serviced in ascending deadline order.
5. completion after the deadline is wrong.
6. is not fair
often repetitive jobs have *periodic* deadlines and:

\[
\text{Rate} = \frac{1}{\text{Period}} \quad (2)
\]

Rate monotonic scheduling schedules jobs with the highest rate first.

Using the notation [3]:

<table>
<thead>
<tr>
<th>( n )</th>
<th>Number of jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_i )</td>
<td>Cost (Time) to service the ( i )th process</td>
</tr>
<tr>
<td>( T_i )</td>
<td>Period of the ( i )th process</td>
</tr>
</tbody>
</table>

It can be shown that RMS will generate a feasible uniprocessor schedule if:

\[
\sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \ldots + \frac{C_n}{T_n} = n\left(2^{\frac{1}{n}} - 1\right) \quad (3)
\]
References

