SWENG3043

Operating Systems

Software Engineering Department

AASTU

Nov 2018

Chapter Four

Scheduling

Objectives

- To introduce CPU scheduling, which is the basis for multiprogrammed operating systems.
- To describe various CPU-scheduling algorithms.
- To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system.

Topics

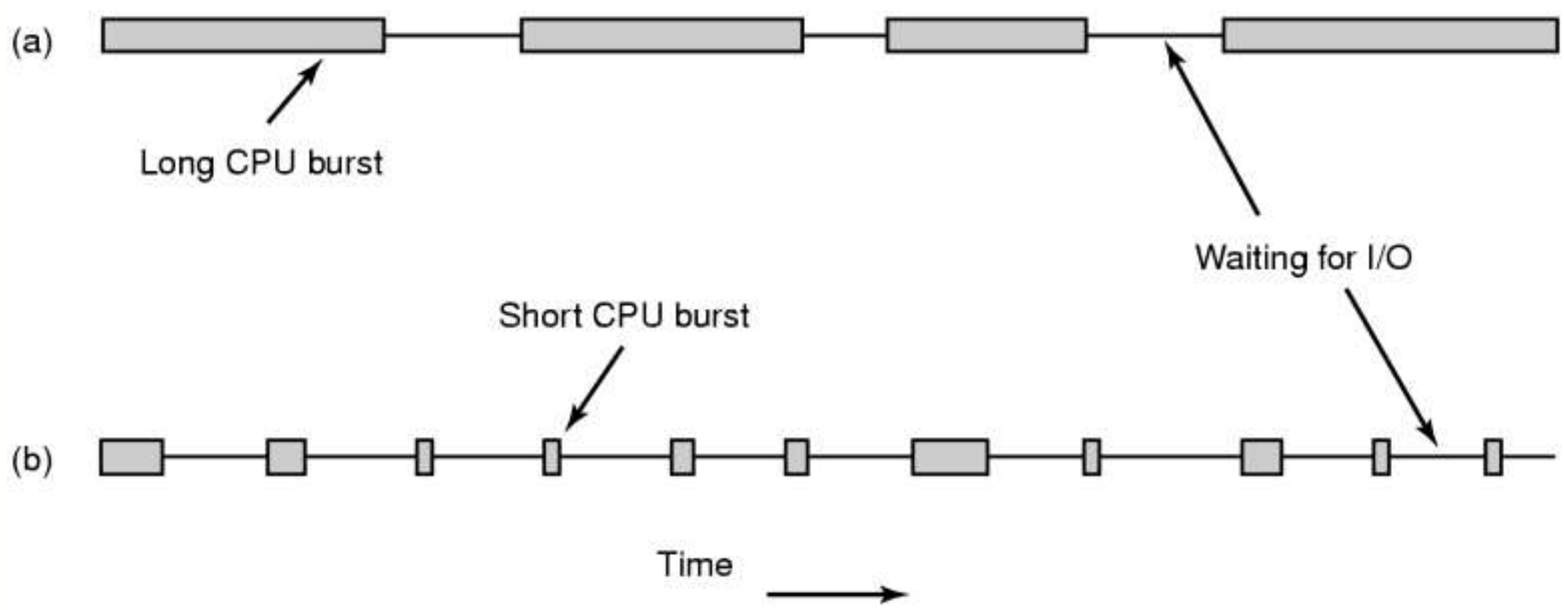
- Basic concepts
- Scheduling Criteria
- Scheduling Algorithms
- Algorithm Evaluation

Basic Concepts

- **Scheduling** refers to a set of policies and mechanisms to control the order of work to be performed by a computer system.
- Part of the operating system that makes scheduling decision is called scheduler and the algorithm it uses is called scheduling algorithm.
- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler; this involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program
- **Dispatch latency** – time it takes for the dispatcher to stop one process and start another running

Basic Concepts(cont'd)

Process Behavior



Bursts of CPU usage alternate with periods of I/O wait

a CPU-bound process

an I/O bound process

Types of Scheduling

- **Non-Preemptive**
  - Run to completion method: once a process is in running state, it continues to execute until it terminates or blocks itself to wait for some event
  - Simple and easy to implement
  - Used in early batch systems
  - It may be well reasonable for some dedicated systems
- **Preemptive**
  - The strategy of allowing processes that are logically runnable to be temporarily suspended and be moved to the ready state.
  - Events that may result pre-emption are *arrival of new processes*, occurrence *of an interrupt* that moves blocked process to ready state and *clock interrupt*
  - Suitable for general purpose systems with multiple users

Scheduling Criteria

- Efficiency/CPU Utilization: The percentage of time that the CPU is busy should be maximized.

- Throughput: The number of processes completed per unit time should be maximized.
- Turn Around Time: The interval from the time of submission of a process to the time of completion.
  - The sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/0.

- Waiting time: is the sum of the periods spent waiting in the ready queue.

- Response Time: is the time from the submission of a request until the first response is produced.
- It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time.

First-Come-First-Served Scheduling

- **First-Come-First-Served Scheduling (FCFSS)**
- **Basic Concept**
  - The process that requested the CPU first is allocated the CPU and keeps it until it released it.
  - The process that has been in the ready queue the longest is selected for running.
  - Its selection function is waiting time and it uses non preemptive scheduling/decision mode.

FCFSS: Example1

- **Illustration**
  - Consider the following processes arrive at time 0

Process	P1	P2	P3
CPU Burst/Service Time (in ms)	24	3	3

<u>Case i. If they arrive in the order of P1, P2, P3</u>

Process	P1	P2	P3
Service Time (T_s)	24	3	3
Response time	0	24	27
Turn around time (T_r)	24	27	30

Average response time = (0+24+27)/3 = 17

Average turn around time = (24+27+30)/3=27

Throughput = 3/30= 1/10

FCFSS(cont'd)

Case ii. If they arrive in the order of P3, P2, P1

Process	P3	P2	P1
Service Time (T_s)	3	3	24
Response time	0	3	6
Turn around time (T_r)	3	6	30

Average response time = (0+3+6)/3 = 3 (Much better than previous case)

Average turn around time = (3+6+30)/3=13

Throughput = 3/30= 1/10

FCFSS: Example2

- Consider the following processes arrive at time 0, 1, 2, 4 respectively

Process	P1	P2	P3	P4
Arrival Time (T_a)	0	1	2	4
Service Time (T_s)	1	100	1	100
Response time	0	0	99	98
Turn around time (T_r)	1	100	100	198

Average response time = (0+0+99+98)/4 = 49.25

Average turn around time = (1+100+100+198)/4=99.75

Throughput = 4/202

FCFSS: Exercise1

Consider the following processes arrive at time 0, 3, 8, 11 respectively

Process	P1	P2	P3	P4
Arrival Time (T_a)	0	3	8	11
Service Time (T_s)	3	5	3	100
Response time	?	?	?	?
Turn around time (T_r)	?	?	?	?

Average response time = ?

Average turn around time = ?

Throughput =?

FCFSS: Exercise1(Answer)

Process	P1	P2	P3	P4
Arrival Time (T_a)	0	3	8	11
Service Time (T_s)	3	5	3	100
Response time	0	0	0	0
Turn around time (T_r)	3	5	3	100

Average response time = (0+0+0+0)/4 = 0

Average turn around time = (3+ 5+3+100)/4=27.75

Throughput = 4/111

FCFSS: Exercise2

Consider the following processes arrive at time 0, 2, 3, 4 respectively

Process	P1	P2	P3	P4
Arrival Time (T_a)	0	2	3	4
Service Time (T_s)	4	16	50	1
Response time	?	?	?	?
Turn around time (T_r)	?	?	?	?

Average response time = ?

Average turn around time = ?

Throughput = ?

FCFSS: Exercise2(Answer)

- Consider the following processes arrive at time 0, 2, 3, 4 respectively

Process	P1	P2	P3	P4
Arrival Time (T_a)	0	2	3	4
Service Time (T_s)	4	16	50	1
Response time	0	2	17	66
Turn around time (T_r)	0	18	67	67

Average response time = (0+2+17+66)/4 = 20.5

Average turn around time = (4+ 18+67+67)/4=39

Throughput = 4/71

FCFSS: Advantage and Drawbacks

- **Advantages**
  - It is the simplest of all non-preemptive scheduling algorithms: process selection & maintenance of the queue is simple.
  - There is a minimum overhead and no starvation.
  - It is often combined with priority scheduling to provide efficiency.
- **Drawbacks**
  - Poor CPU and I/O utilization: CPU will be idle when a process is blocked for some I/O operation.
  - Poor and unpredictable performance: it depends on the arrival of processes.
  - Unfair CPU allocation: If a big process is executing, all other processes will be forced to wait for a long time until the process releases the CPU. It performs much better for long processes than short ones.

Shortest Job First Scheduling

- **Shortest Job First Scheduling (SJFS)**
- **Basic Concept**
  - Process with the shortest expected processing time (CPU burst) is selected next.
  - Its selection function is execution time and it uses non preemptive scheduling/decision mode.
- SJF scheduling is used frequently in long-term scheduling.

SJFS: Example1

- **Illustration**
  - Consider the following processes arrive at time 0

Process	P1	P2	P3
CPU Burst (in ms)	24	3	3

  - **Case i. FCFSS**

Process	P1	P2	P3
Turn around time	24	27	30
Response time	0	24	27

Average response time = (0+24+27)/3 = 17

Average turn around time = (24+27+30)/3=27

Throughput = 3/30

SJFS(cont'd)

- **<u>Case ii. SJFS</u>**

Process	P3	P2	P1
Response time	0	3	6
Turn around time	3	6	30

Average response time = (0+3+6)/3 = 3

Average turn around time = (3+6+30)/3=13

Throughput = 3/30

SJFS: Example

- **Consider the following processes arrive at time 0, 2, 4, 6, 8 respectively**

Process	P1	P2	P3	P4	P5
Arrival Time (T_a)	0	2	4	6	8
Service Time (T_s)	3	6	4	5	2
Response time	0	1	7	9	1
Turn around time (T_r)	3	7	11	14	3

Average response time = (0+1+7+9+1)/5 = 3.6

Average turn around time = (3+7+11+14+3)/5=7.6

Throughput = 5/20

SJFS: Exercise

- **Consider the following processes arrive at time 0, 2, 3, 8, 10 respectively**

Process	P1	P2	P3	P4	P5
Arrival Time (T_a)	0	2	3	8	10
Service Time (T_s)	7	10	5	1	2
Response time	?	?	?	?	?
Turn around time (T_r)	?	?	?	?	?

Average response time = ?

Average turn around time = ?

Throughput = ?

SJFS: Exercise(Answer)

- **Consider the following processes arrive at time 0, 2, 3, 8, 10 respectively**

Process	P1	P2	P3	P4	P5
Arrival Time (T_a)	0	2	3	8	10
Service Time (T_s)	7	10	5	1	2
Response time	0	13	4	4	3
Turn around time (T_r)	7	23	9	5	5

Average response time = (0+13+4+4+3)/5 = 4.8

Average turn around time = (7+23+9+5+5)/5=9.8

Throughput = 5/25

SJFS: Advantage and Drawback

- **Advantages**
  - It produces optimal average turn around time and average response time
  - There is a minimum overhead
- **Drawbacks**
  - Starvation: some processes may not get the CPU at all as long as there is a steady supply of shorter processes.
  - Variability of response time is increased, especially for longer processes.

Shortest Remaining Time Scheduling

- **Shortest Remaining Time Scheduling (SRTS)**
- **Basic Concept**
  - The process that has the shortest expected remaining process time.
  - If a new process arrives with a shorter next CPU burst than what is left of the currently executing process, the new process gets the CPU.
  - Its selection function is remaining execution time and uses preemptive decision mode

SRTS: Example1

- **Illustration**
  - **Consider the following processes arrive at time 0, 2, 4**

Process	P1	P2	P3
Arrival Time (T_a)	0	2	4
Service Time (T_s)	3	8	4
Response time	0	1	0
Turn around time (T_r)	3	13	4

Average response time = (0+1+0)/3= 0.33

Average turn around time = (3+13+4)/3= 20/3

Throughput = 3/15

SRTS: Example2

- **Illustration**
  - **Consider the following processes arrive at time 0, 2, 4, 6, 8 respectively**

Process	P1	P2	P3	P4	P5
Arrival Time (T_a)	0	2	4	6	8
Service Time (T_s)	3	8	4	5	2
Response time	0	1	0	4	0
Turn around time (T_r)	3	20	4	9	2

Average response time = (0+1+0+4+0)/5 = 1

Average turn around time = (3+20+4+9+2)/5=7.6

Throughput = 5/20

SRTS(cont'd)

- **Advantages**
  - It gives superior turnaround time performance to SJFS, because a short job is given immediate preference to a running longer process
- **Drawbacks**
  - There is a risk of starvation of longer processes
  - High overhead due to frequent process switch
- **Difficulty with SJFS**
  - Figuring out required processing time of each process

Round Robin Scheduling

- **Round Robin Scheduling (RRS)**
- **Basic Concept**
  - A small amount of time called a ***quantum*** or time slice is defined. According to the quantum, a clock interrupt is generated at periodic intervals. When the interrupt occurs, the currently running process is placed in the ready queue, and the next ready process is selected on a FCFS basis.
  - The CPU is allocated to each process for a time interval of upto one quantum. When a process finishes its quantum it is added to the ready queue, when it is requesting I/O it is added to the waiting queue.
  - The ready queue is treated as a circular queue
  - Its selection function is based on quantum and it uses preemptive decision mode

RRS(cont'd)

- **Illustration**
  - Consider the following processes arrive at time 0, 2, 4 respectively; quantum=4

Process	P1	P2	P3
Arrival Time (T_a)	0	2	4
Service Time (T_s)	3	7	4
Response time	0	1	3
Turn around time (T_r)	3	12	7

Average response time = (0+1+3)/3

Average turn around time = (3+12+7)/3

Throughput = 3/14

RRS: Exercise

- **Illustration**
  - Consider the following processes arrive at time 0, 3, 4, 7 respectively; quantum=4

Process	P1	P2	P3	p4
Arrival Time (T_a)	0	3	4	7
Service Time (T_s)	7	5	9	10
Response time	?	?	?	?
Turn around time (T_r)	?	?	?	?

Average response time = ?

Average turn around time = ?

Throughput = ?

RRS: Exercise(Answer)

- **Illustration**
  - Consider the following processes arrive at time 0, 3, 4, 7 respectively; quantum=4

Process	P1	P2	P3	p4
Arrival Time (T_a)	0	3	4	7
Service Time (T_s)	7	5	9	10
Response time	0	1	4	5
Turn around time (T_r)	19	17	25	24

Average response time = (0+1+4+5)/4

Average turn around time =(19+17+25+24)/4

Throughput = 4/31

RRS(cont'd)

- **Features**
  - The oldest, simplest, fairest and most widely used preemptive scheduling
  - Reduces the penalty that short processes suffer with FCFS
- **Drawbacks**
  - CPU-bound processes tend to receive unfair portion of CPU time, which results in poor performance for I/O bound processes.
  - It makes implicit assumption that all processes are equally important. It does not take external factors into account
  - Maximum overhead due to frequent process switch

RRS(cont'd)

- **Difficulty with RRS**
  - The length of the quantum should be decided carefully
  - E.g.1) quantum =20ms, context switch =5ms, % of context switch = 5/25 *100=20%
    - Poor CPU utilization
    - Good interactivity
  - E.g.2) quantum =500ms, context switch =5ms, % of context switch = 5/505 *100<1%
    - Improved CPU utilization
    - Poor interactivity
  - Setting the quantum too short causes
    - Poor CPU utilization
    - Good interactivity

RRS(cont'd)

- Setting the quantum too long causes
  - Improved CPU utilization
  - Poor interactivity
- A quantum around 100ms is often reasonable compromise
- To enable interactivity, a maximum response time (MR) can be specified, and the quantum (m) can be computed dynamically as follows
  - m = MR/n , where n is number of processes
    - A new m is calculated when the number of processes changes

Priority Scheduling

- **Basic Concept**
  - Each process is assigned a priority and the runnable process with the highest priority is allowed to run i.e. a ready process with highest priority(smallest integer = highest priority) is given the CPU.
  - Equal-priority processes are scheduled in FCFS order
  - Priorities defined in two ways:
    - Internal-use some measurable quantity or quantities to compute the priority of a process
      - E.g. time limits, memory requirements, the number of open files, and the ratio of average I/O burst to average CPU burst have been used in computing priorities
    - External-set by criteria outside the operating system.
      - E.g. importance of the process, the type and amount of funds being paid for computer use, the department sponsoring the work, and other, often political, factors

Priority Scheduling(cont'd)

- Illustration
  - Consider the following processes arrive at time 0

Process	P1	P2	P3	P4	P5
Priority	2	4	5	3	1
Service Time (T_s)	3	6	4	5	2
Response time	2	10	16	5	0
Turn around time (T_r)	5	16	20	10	2

Average response time = (2+10+16+5+0)/5 = 6.6

Average turn around time = (5+16+20+10+2)/5=10.6

Throughput = 5/20= 0.25

Priority Scheduling(cont'd)

- **Advantages**
  - It considers the fact that some processes are more important than others
- **Drawbacks**
  - A high priority process may run indefinitely and it can prevent all other processes from running. This creates starvation on other processes.
    - **Solution: Aging** – as time progresses increase the priority of the process
      - E.g. if priorities range from 127 (low) to 0 (high), we could increase the priority of a waiting process by 1 every 15 minutes.

Multilevel Queues Scheduling

- **Multilevel Queues Scheduling (MLQS)**
  - It is used for situations for which processes can be classified into different groups:
    - System processes
    - Interactive processes
    - Batch processes
  - The ready queue is partitioned into several separate queues
  - Each processes is assigned to one queue permanently, based on some property of the process such as memory size, process priority, process type, etc

MLQS(cont'd)

- Each queue has its own scheduling algorithm: RRS, FCFSS, SJFS, PS etc
  - System processes............... PS
  - Interactive processes............RRS
  - Batch processes..................FCFSS
- There must be some kind of scheduling between the queues
  - Commonly implemented as ***fixed-priority preemptive scheduling***, i.e. system processes have absolute priority over the interactive processes and interactive processes have absolute priority over the batch processes. Lowest priority processes may be starved.

Reading Assignment

- Read about “Algorithm Evaluation” from Operating System Concepts. Page 300-304