Assignment 4

**Task 1:**

**Task 2:**

1.Try to adjust the time quantum Q for RR and α value for SJF, give the optimal value of Q and α to minimize average waiting\_time for the above particular input. Include the output of your tests in the report.

Assumption:

In the event when a context switch and a new process is added to the Runnable queue the preference is given to the new process.

Observations:

Process 0 mostly has a large burst time, except once

Process 1 has balance of large and small burst times

Process 2 bust time is ascending though the time line

Process 3 similar to process 2 it is in ascending order

|  |  |  |
| --- | --- | --- |
| Process | Number of Short bust | Number of long bust |
| 0 | 1 | 3 |
| 1 | 2 | 2 |
| 2 | 1 | 3 |
| 3 | 1 | 3 |

Short bust = burst time < 5

Long bust = burst time > 5

So in our example we will treat Process 1 as a O/I bound process and the others as CPU bound processes.

Complete the simulation for

Round Robin (RR)

average waiting time **8.56**

Full log at APPENDIX.1

Quantum was set to **2**

To get the best result the quantum was set to **10**,

average waiting time **6.44**

Full log at APPENDIX.7

We were able to the best average waiting time.However when the Quantum is set to large number the RR algorithm becomes similar to FCFS

Shortest remaining time first (SRTF) with given CPU burst time

average waiting time **4.50**

APPENDIX.2

Shortest job first (SJF) with future prediction

average waiting time **7.12**

APPENDIX.3

Changing the alpha did not have much impact on the average waiting time.

Even though it is an non pre-emptive design the result is better than RR with quantum equals to 2

Test with all the three implemented scheduling schemes (Task 1.1-1.3) and compare them. Which one gives the least average\_waiting\_time?

**Shortest remaining time first (SRTF) gives the least average\_waiting\_time**

2. Which of the evaluated schemes in this assignment (Task 1.1-1.3, 2.1) generates the optimal schedule (gives minimum average waiting time) for a system with:

a) All short processes

Inputs

0 0 2

1 1 1

2 2 2

3 5 1

3 30 2

1 31 1

2 32 1

0 38 2

2 60 1

0 62 2

1 65 1

3 66 1

1 90 2

0 95 2

2 98 1

3 99 2

Round Robin (RR)

Logs at Appendix.8

average waiting time **0.25**

Shortest remaining time first (SRTF) with given CPU burst time

Logs at Appendix.9

average waiting time **0.25**

Shortest job first (SJF) with future prediction

Logs at Appendix.10

average waiting time **0.25**

When given short jobs all three algorithms perform the same

b) Very short and very long processes interleave each other with unpredictable pattern.

0 0 30

1 1 1

2 2 30

3 5 1

3 30 1

1 31 1

2 32 30

0 38 30

2 60 30

0 62 30

1 65 1

3 66 1

1 90 1

0 95 30

2 98 30

3 99 1

Process 1 and 3 are short process

Process 0 and 2 are long process

Round Robin (RR)

Logs at Appendix.11

average waiting time **63.69**

Shortest remaining time first (SRTF) with given CPU burst time

Logs at Appendix.12

average waiting time **31.75**

Shortest job first (SJF) with future prediction

Logs at Appendix.13

average waiting time **44.31**

Round Robin preformed the worse in the mix process test. Because it treats both the I/O bound and CPU bound process equally hence this results in larger waiting time.

Shortest Remaining time first preform the best because it is able to pirotize the short jobs first over the long once hence most of short job are able to get the CPU much earlier.

Shortest job first performance was not as good because firstly it was an non pre-emptive design, so once a CPU bound process running it will only return the CPU to other after its done hence it makes the waiting time larger.

Secondly this was too short of a test to conclude, just testing this algorithm with 12 data point is not meaningful enough to come to any conclusion.

3. State a different scheduling mechanism that has not been mentioned in the lecture notes and explain the intuition behind the using this scheme.

The scheduling algorithm that was decided to be use was **Foreground-background**

<https://en.wikipedia.org/wiki/Foreground-background>

This is very similar to the rotating stairs algorithm discussed in the lecture however this is much simpler.

The algorithm has 2 queues one for I/O bound and the other for CPU bound.And this algorithm will always let the new process preempt the current process.

**This algorithm in action:**

We will use first 4 sequence of the same inputs provided in the assignment.

0 0 9

1 1 8

2 2 2

3 5 2

...

...

FQ = Foreground queue

BQ = Background queue

FQ has a quantum of 2

BQ has a quantum of 4

Rules:

When a new process is added it preemptive the current process.

If a process in foreground queue’s quantum expires it is sent to background queue.

When FQ empty then task are queried from BQ

TICK 0

CPU = P0:9 FQ={} BQ={}

TICK 1

CPU = P1:8 FQ={P0:8} BQ={}

P0 was not able to finish its quantum hence it was send to the foreground queue

TICK 2

CPU = P2:2 FQ={P0:8,P1:7} BQ={}

New process once enter always preempt the current process

TICK 4

CPU = P0:8 FQ={P1:7} BQ={}

P0 will run for 2 quantum again

P2 finished execution

TICK 5

CPU = P3:2 FQ={P1:7,P0:7} BQ={}

TICK 7

CPU = P1:7 FQ={P0:7} BQ={}

P3 finished execution

TICK 9

CPU = P0:7 FQ={} BQ={P1:5}

P1 expires its quantum hence it is now sent to background queue

TICK 11

CPU = P1:5 FQ={} BQ={P0:5}

P0 expires its quantum hence it is now sent to background queue

P1 will now run because FQ is empty

P1 quantum is now 4

TICK 15

CPU = P0:5 FQ={} BQ={P1:1}

P0 will now run because FQ is empty

P0 quantum is now 4

TICK 19

CPU = P1:1 FQ={} BQ={P0:1}

TICK 20

CPU = P0:1 FQ={} BQ={}

P1 finished execution

TICK 21

CPU = idle FQ={} BQ={}

P0 finished execution

Using the example above the average waiting time is

P0 : 21 - 0 - 9 = 12

P1 : 20 - 1 - 8 = 11

P2 : 4 - 2 - 2 = 0

P3 : 7 - 5 - 2 = 0

(12 + 11)/4 = **5.75**

4.Assume your system has N CPU cores, and each process only requires burst time on 1 core. Will it make the scheduler more complicated? Suggest how to extend the current scheduler to multiprocessor system.

If we extend the current example we can use the a single queue that is shared across all the CPUs. In order for that to work we would need some kind of locking mechanism. Fortunately locking seriously reduce the performance.

Secondly this approach is not salable. If we have when one process locks the queue, this will result in N-1 number of process to wait for the queue to be unlocked for it to pick the next task.

Another disadvantage with this is as we are taking a task from the queue one by one. It will bounce from one CPU to another randomly resulting in a lot of cache misses.

To resolve the issue raised above we will have multiple queues. One for each CPU.

Given that a process is coupled with the CPU it ran on so as to reduce the number of cache miss, We have to be careful to no over load a CPU. We will need to ensure all the CPU are sharing the workload evenly distributed.

Each CPU’s queue should have a good mixture of IO bound and CPU bound processes.

One might feel that multiple queues is the best approach here, But using Linux scheduler as reference we can prove otherwise.

There are 3 different scheduler that are popular in linux, Completely fair Scheduler , O(1) scheduler and BFS scheduler.

CFS and O(1) schedulers use multiple queues and BFS is single queue scheduler.

Reference : <http://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-multi.pdf>

Section B

Consider the following compare-and-exchange atomic instruction (abstracted as a function – in other words, assume that the following function is atomic):

int cas(int \*ptr, int oldval, int newval);

If the integer pointed to by ptr is equal to oldval, then the content pointed to by ptr will be replaced with newval, and a ‘1’ will be returned by cas. Otherwise, a ‘0’ is returned. Using this function/instruction, show how a spinlock can be implemented by giving the C code for the spin\_lock() and spin\_unlock() functions. Write down any assumptions that you make.

**APPENDIX**

APPENDIX.1

((0, 0)

(2, 1)

(4, 2)

(6, 0)

(8, 1)

(10, 3)

(12, 0)

(14, 1)

(16, 0)

(18, 1)

(20, 0)

(30, 3)

(32, 1)

(34, 2)

(36, 3)

(38, 2)

(40, 0)

(42, 3)

(43, 2)

(45, 0)

(47, 0)

(49, 0)

(60, 2)

(62, 0)

(64, 2)

(66, 1)

(68, 3)

(70, 2)

(72, 1)

(73, 3)

(75, 2)

(76, 3)

(78, 3)

(90, 1)

(92, 1)

(94, 1)

(96, 0)

(98, 1)

(100, 2)

(102, 0)

(104, 3)

(106, 1)

(108, 2)

(110, 0)

(112, 3)

(114, 2)

(116, 0)

(118, 3)

(120, 2)

(122, 0)

(124, 3)

(126, 2)

average waiting time 8.56

APPENDIX.2

(0, 0)

(2, 2)

(4, 0)

(5, 3)

(7, 0)

(13, 1)

(30, 3)

(31, 1)

(33, 3)

(37, 2)

(43, 0)

(60, 2)

(62, 0)

(64, 2)

(65, 1)

(68, 2)

(72, 3)

(90, 1)

(100, 3)

(108, 2)

(117, 0)

average waiting time 4.50

APPENDIX.3

(0, 0)

(9, 1)

(17, 2)

(19, 3)

(30, 3)

(35, 2)

(41, 1)

(43, 0)

(60, 2)

(67, 1)

(70, 3)

(78, 0)

(90, 1)

(100, 0)

(110, 2)

(119, 3)

average waiting time 7.12

APPENDIX.4

RR with mixture of large and small process

(0, 0)

(2, 1)

(4, 2)

(6, 0)

(8, 3)

(10, 2)

(12, 0)

(14, 2)

(16, 0)

(18, 2)

(20, 0)

(22, 2)

(24, 0)

(26, 2)

(28, 0)

(30, 2)

(32, 3)

(34, 0)

(36, 1)

(38, 2)

(40, 2)

(42, 0)

(44, 0)

(46, 2)

(48, 2)

(50, 0)

(52, 0)

(54, 2)

(56, 2)

(58, 0)

(60, 0)

(62, 2)

(64, 2)

(66, 2)

(68, 0)

(70, 0)

(72, 0)

(74, 2)

(76, 1)

(78, 3)

(80, 2)

(82, 2)

(84, 0)

(86, 0)

(88, 0)

(90, 2)

(92, 2)

(94, 2)

(96, 0)

(98, 0)

(100, 1)

(102, 0)

(104, 2)

(106, 2)

(108, 0)

(110, 2)

(112, 2)

(114, 0)

(116, 3)

(118, 0)

(120, 0)

(122, 2)

(124, 2)

(126, 0)

(128, 2)

(130, 2)

(132, 0)

(134, 0)

(136, 2)

(138, 0)

(140, 2)

(142, 2)

(144, 0)

(146, 0)

(148, 2)

(150, 0)

(152, 2)

(154, 2)

(156, 0)

(158, 0)

(160, 2)

(162, 0)

(164, 2)

(166, 2)

(168, 0)

(170, 0)

(172, 2)

(174, 0)

(176, 2)

(178, 2)

(180, 0)

(182, 0)

(184, 2)

(186, 0)

(188, 2)

(190, 2)

(192, 0)

(194, 0)

(196, 2)

(198, 0)

(200, 2)

(202, 2)

(204, 0)

(206, 0)

(208, 2)

(210, 0)

(212, 2)

(214, 2)

(216, 0)

(218, 0)

(220, 0)

(222, 2)

(224, 2)

(226, 0)

(228, 0)

(230, 2)

(232, 2)

(234, 0)

(236, 0)

(238, 2)

(240, 2)

(242, 0)

(244, 0)

(246, 2)

(248, 0)

(250, 2)

(252, 0)

(254, 2)

average waiting time 68.75

APPENDIX.5

SRTF with mixture of large and small process

(0, 0)

(1, 1)

(2, 0)

(5, 3)

(6, 0)

(30, 3)

(31, 1)

(32, 0)

(34, 2)

(64, 0)

(65, 1)

(66, 3)

(67, 0)

(90, 1)

(91, 0)

(99, 3)

(100, 0)

(128, 2)

(158, 0)

(188, 2)

average waiting time 31.75

APPENDIX.6

SJF with mixture of large and small process

(0, 0)

(30, 3)

(31, 1)

(32, 2)

(62, 1)

(63, 3)

(64, 2)

(94, 1)

(95, 1)

(96, 3)

(97, 0)

(127, 3)

(128, 2)

(158, 0)

(188, 2)

(218, 0)

average waiting time 44.31

APPENDIX.7

RR q =10

(0, 0)

(9, 1)

(17, 2)

(19, 3)

(30, 3)

(35, 1)

(37, 2)

(43, 0)

(60, 2)

(67, 0)

(69, 1)

(72, 3)

(90, 1)

(100, 0)

(110, 2)

(119, 3)

average waiting time 6.44

APPENDIX.8

Short jobs RR

(0, 0)

(2, 1)

(3, 2)

(5, 3)

(30, 3)

(32, 1)

(33, 2)

(38, 0)

(60, 2)

(62, 0)

(65, 1)

(66, 3)

(90, 1)

(95, 0)

(98, 2)

(99, 3)

average waiting time 0.25

APPENDIX.9

Short jobs for SRTF

(0, 0)

(2, 1)

(3, 2)

(5, 3)

(32, 1)

(33, 2)

(38, 0)

(60, 2)

(62, 0)

(65, 1)

(66, 3)

(90, 1)

(95, 0)

(98, 2)

(99, 3)

average waiting time 0.25

APPENDIX.10

Short jobs for SJF

(0, 0)

(2, 1)

(3, 2)

(5, 3)

(30, 3)

(32, 1)

(33, 2)

(38, 0)

(60, 2)

(62, 0)

(65, 1)

(66, 3)

(90, 1)

(95, 0)

(98, 2)

(99, 3)

average waiting time 0.25

APPENDIX.11

Mix jobs RR

(0, 0)

(10, 1)

(11, 2)

(21, 3)

(22, 0)

(32, 2)

(42, 3)

(43, 1)

(44, 2)

(54, 0)

(64, 0)

(74, 2)

(84, 2)

(94, 2)

(104, 0)

(114, 1)

(115, 3)

(116, 0)

(126, 1)

(127, 2)

(137, 0)

(147, 2)

(157, 3)

(158, 2)

(168, 0)

(178, 0)

(188, 0)

(198, 2)

(208, 2)

(218, 0)

(228, 0)

(238, 2)

average waiting time 63.69

APPENDIX.12

Mix jobs SRTF

(0, 0)

(1, 1)

(2, 0)

(5, 3)

(6, 0)

(30, 3)

(31, 1)

(32, 0)

(34, 2)

(64, 0)

(65, 1)

(66, 3)

(67, 0)

(90, 1)

(91, 0)

(99, 3)

(100, 0)

(128, 2)

(158, 0)

(188, 2)

average waiting time 31.75

APPENDIX.13

Mix jobs SJF

(0, 0)

(30, 3)

(31, 1)

(32, 2)

(62, 1)

(63, 3)

(64, 2)

(94, 1)

(95, 1)

(96, 3)

(97, 0)

(127, 3)

(128, 2)

(158, 0)

(188, 2)

(218, 0)

average waiting time 44.31