# CS 620: Homework 6

Carmen St. Jean

October 11, 2012

1. *(3) Give 3 advantages of a multi-threaded file server design as opposed to a single threaded file server design.*

   If a process has multiple threads of control, it can perform more than one task at a time. With a file server, this is critical, because there will most likely be multiple users connected to the file server at once. This means that at any given moment there may be multiple requests which a multi-threaded server could process simultaneously. A single-threaded server would only process one request from one user at a time, so it would be slow for all of the users. Furthermore, if an operation must be blocked, then this may slow things down for all of the users even more.

2. *(2) Threads share the address space of their parent including the heap, but each thread has its own stack. Explain why threads do not share a common stack.*

   Say threads shared a common stack. Say also that one thread calls a function which is added to the stack and another call the same function but with different parameters to be added to the stack. Then the first thread would have to stop executing the contents of that function because only the top-most function of the stack can be executed. This would defeat the purpose of having threads in the first place. It would also become unclear which thread has access to the stack at any given time and you might have one thread corrupting the values of another thread in the stack.

3. *(9) Calculate the mean response time (R) for the following scheduling policies using the workload given in Table 1.*

| Process | Arrival Time | CPU Burst |
|---|---|---|
| P1 | 0 | 3 |
| P2 | 2 | 5 |
| P3 | 3 | 4 |
| P4 | 6 | 7 |
| P5 | 9 | 3 |

   *Assume that context switch time is 0. For RR, if a job arrives at the same time that another job finishes its quantum, assume that the arriving job gets into the CPU queue ahead of the job that just finished its quantum.*

   (a) *SRTN (Shortest Remaining Time Next)*

| SRTN | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | E | E | E | | | | | | | | | | | | | | | | | | | | | 3 - 0 = 3 |
| P2 | | | W | W | W | W | W | E | E | E | E | E | | | | | | | | | | | | 12 - 2 = 10 |
| P3 | | | | E | E | E | E | | | | | | | | | | | | | | | | | 7 - 3 = 4 |
| P4 | | | | | | | W | W | W | W | W | W | W | W | W | E | E | E | E | E | E | E | | 22 - 6 = 16 |
| P5 | | | | | | | | | | W | W | W | E | E | E | | | | | | | | | 15 - 9 = 6 |

   $R_{SRTN} = \frac{3+10+4+16+6}{5} = \frac{39}{5} = 6.5$ time units

   At time unit nine, both P2 and P5 have three time units of CPU burst remaining. In the table above, I assumed that P2 is allowed to continue executing under these circumstances. In case execution should have gone to the newer process, P5, I have drawn up the table for that as well:

| SRTN | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | E | E | E | | | | | | | | | | | | | | | | | | | | | 3 - 0 = 3 |
| P2 | | | W | W | W | W | W | E | E | W | W | W | E | E | E | | | | | | | | | 15 - 2 = 13 |
| P3 | | | | E | E | E | E | | | | | | | | | | | | | | | | | 7 - 3 = 4 |
| P4 | | | | | | | W | W | W | W | W | W | W | W | W | E | E | E | E | E | E | E | | 22 - 6 = 16 |
| P5 | | | | | | | | | | E | E | E | | | | | | | | | | | | 12 - 9 = 3 |

$R_{SRTN} = \frac{3+13+4+16+3}{5} = \frac{39}{5} = 6.5$ time units

In this case, the average response time is still 6.5 time units.

(b) *LIFO preemptive*

| LIFO pre | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | E | E | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | W | E | | 22 - 0 = 22 |
| P2 | | | E | W | W | W | W | W | W | W | W | W | W | W | W | W | W | E | E | E | E | | | 21 - 2 = 19 |
| P3 | | | | E | E | E | W | W | W | W | W | W | W | W | W | W | E | | | | | | | 17 - 3 = 14 |
| P4 | | | | | | | E | E | E | W | W | W | E | E | E | E | | | | | | | | 16 - 6 = 10 |
| P5 | | | | | | | | | | E | E | E | | | | | | | | | | | | 12 - 9 = 3 |

$R_{LIFOpre} = \frac{22+19+14+10+3}{5} = \frac{68}{5} = 13.6$ time units

(c) *RR with Qt = 2 time units*

| RR | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | E | E | W | W | E | | | | | | | | | | | | | | | | | | | 5 - 0 = 5 |
| J2 | | | E | E | W | W | W | E | E | W | W | W | W | W | W | E | | | | | | | | 16 - 2 = 14 |
| J3 | | | | W | W | E | E | W | W | W | W | E | E | | | | | | | | | | | 13 - 3 = 10 |
| J4 | | | | | | | W | W | W | E | E | W | W | W | W | W | E | E | W | E | E | E | | 22 - 6 = 16 |
| J5 | | | | | | | | | | W | W | W | W | E | E | W | W | W | E | | | | | 19 - 9 = 10 |

$R_{RR} = \frac{5+14+10+16+10}{5} = \frac{55}{5} = 11$ time units

4. *(3) In class, I mentioned that LIFO preemptive performs closest to SRTN (the fastest policy). Could FIFO perform better than LIFO preemptive? If yes, give an example. If no, justify.*

   Yes, FIFO can perform better than LIFO preemptive. Take the example we used in class, except change J3's CPU burst time to two time units and J4's burst time to three time units:

| Process | Arrival Time | CPU Burst |
|---|---|---|
| J1 | 1 | 4 |
| J2 | 3 | 1 |
| J3 | 4 | 2 |
| J4 | 5 | 3 |

| FIFO | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| J1 | E | E | E | E | | | | | | | | 5 - 1 = 4 |
| J2 | | | W | W | E | | | | | | | 6 - 3 = 3 |
| J3 | | | | W | W | E | E | | | | | 8 - 4 = 4 |
| J4 | | | | | W | W | W | E | E | E | | 11 - 5 = 6 |

$R_{FIFO} = \frac{4+3+4+6}{5} = \frac{17}{5} = 3.4$ time units

| LIFO pre | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| J1 | E | E | W | W | W | W | W | W | E | E | | 11 - 1 = 10 |
| J2 | | | E | | | | | | | | | 4 - 3 = 1 |
| J3 | | | | E | W | W | W | E | | | | 9 - 4 = 5 |
| J4 | | | | | E | E | E | | | | | 8 - 5 = 3 |

$R_{LIFOpre} = \frac{10+1+5+3}{5} = \frac{19}{5} = 3.8$ time units

5. *(4) Assume that context switch overhead is 0. Let the average CPU burst (service time) of a long job be $ST_{long}$; and let the average CPU burst (service time) of a short job be $ST_{short}$; What is the impact of*

(a) *a long quantum time ($Qt \approx ST_{long}$) on the performance of a CPU when the workload consists of a mix of long jobs and short jobs;*

The short jobs will not be switched out, while the longer long jobs will be switched out. Since $ST_{long}$ is an average, some long jobs will be shorter than the quantum time and some will be longer. Only the long jobs with service time equal to or longer than the quantum time will be switched.

(b) *a long quantum time ($Qt \approx ST_{long}$) on the performance of a CPU when the workload consists of all short jobs;*

Every job whose service time is shorter than the quantum time will not be switched out. Since all jobs are small and the burst time is long, then most likely no jobs will be switched out.

(c) *a short quantum time ($Qt \approx ST_{short}$) on the performance of a CPU when the workload consists of all long jobs;*

Every job whose service time is longer than the quantum time will be switched out. Since all jobs are long and the quantum time is small, most likely all jobs will be switched out.

(d) *a short quantum time ($Qt \approx ST_{short}$) on the performance of a CPU when the workload consists of a mix of long jobs and short jobs.*

All long jobs will be switched out at least once. Some of the longer short jobs may also be switched out if their burst time is longer than the quantum time.

6. *(4) Consider a system running 100 I/O-bound jobs and 10 CPU-bound job. Assume that the I/O-bound jobs issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context switching overhead is 0.1 millisecond and that all processes are long-running tasks. What percentage of CPU time is wasted in context switch for a round-robin scheduler when:*

The formula to find the percentage of CPU time wasted will be $\frac{c}{c+Qt}$ where $c$ is the context switch overhead and $Qt$ is the quantum time.

(a) *the time quantum is 1 millisecond;*
Time wasted $= \frac{c}{c+Qt} = \frac{0.1}{0.1+1} = \frac{0.1}{1.1} = 9.1\%$

(b) *the time quantum is 10 milliseconds.*
Time wasted $= \frac{c}{c+Qt} = \frac{0.1}{0.1+10} = \frac{0.1}{10.1} = 0.99\%$