Johnny Li

COP4600

HW 2.2

10. In the text it was stated that the model of Fig. 2-11(a) was not suited to a file server using a cache in memory. Why not? Could each process have its own cache?

In the text, it was stated that the model of Fig. 2-11(a) was not suited to a file server using a cache in memory because every process has its own address space since each process uses its own address space that differs from other processes’ address space, they cannot share the same cache memory.

14. In Fig. 2-12 the register set is listed as a per-thread rather than a per-process item. Why? After all, the machine has only one set of registers.

The register set is listed as a per-thread rather than a per-process item because the registers save the thread information, containing state information of the threads, used in context switching.

17. In this problem you are to compare reading a file using a single-threaded file server and a multithreaded server. It takes 12 msec to get a request for work, dispatch it, and do the rest of the necessary processing, assuming that the data needed are in the block cache. If a disk operation is needed, as is the case one-third of the time, an additional 75 msec is required, during which time the thread sleeps. How many requests/sec can the server handle if it is single threaded? If it is multithreaded?

Process Time = 12msec

Disk Time = 75msec

Disk Operation occurrence = 1/3 of the time process requests

Single Thread Server:

Time Cache Hit = 12msec

Time Cache Miss = 12msec+75msec = 87msec

Average Cache Time = (2/3)\*(12)+(1/3)\*(87) = 37msec

Number of requests = (1000msec/1s)\*(1 request/37msec) ~= 27 requests/sec

Multithreaded Server:

Threads are still processed while time is spent waiting for the disk operation, therefore every request takes 12msec.

Number of requests = (1000msec/1s)\*(1 request/12msec) ~= 83 requests/sec

(around 3x as much as Single Thread Server)

25. Can the priority inversion problem discussed in Sec. 2.3.4 happen with user-level threads? Why or why not?

Preemption is not allowed in user-level threads therefore when a low priority thread is running it will stay running till it finishes even with a high priority thread is ready in queue. Thus, the priority inversion problem cannot happen with user-level threads.

43. Measurements of a certain system have shown that the average process runs for a time T before blocking on I/O. A process switch requires a time S, which is effectively wasted (overhead). For round-robin scheduling with quantum Q, give a formula for the CPU efficiency for each of the following:

(a) Q = ∞ (b) Q > T (c) S < Q < T (d) Q = S (e) Q nearly 0

CPU Efficiency = CPU time/Total time = T/Q

Time switching = CPU Efficiency \* S = (T/Q)\*S

Total CPU Time = Running time + Wasted time = T + W = T+(T/Q)\*S

Final CPU Efficiency = T/(T+(T/Q)\*S)

a) Q = ∞

Final CPU Efficiency = T/(T+(T/Q)\*S) = T/(T+(T/∞)\*S) = T/(T+0\*S) = T/(T+0) = T/T = 1

b) Q > T, No switching occurs as there is enough time to finish execution.

S = 0, so Final CPU Efficiency = T/(T+(T/Q)\*S) = T/(T+(T/Q)\*0) = T/(T+0) = T/T = 1

c) S < Q < T

Final CPU Efficiency = T/(T+(T/Q)\*S), CPU efficiency will vary depending on the difference of T and Q.

d) Q = S

Final CPU Efficiency = T/(T+(T/Q)\*Q) = T/(T+T) = T/(2T) = ½

e) Q nearly 0

Final CPU Efficiency = T/(T+(T/Q)\*S) = T/(T+(∞)\*S) = T/(T+∞) = T/∞= 0

44. Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5, and X. In what order should they be run to minimize average response time? (Your answer will depend on X.)

The five jobs should be running in shortest job first to minimize average response time. Depending on the value of X, it can be in the following orders:

X,3,5,6,9

3,X,5,6,9

3,5,X,6,9

3,5,6,X,9

3,5,6,9,X

45. Five batch jobs. A through E, arrive at a computer center at almost the same time. They have estimated running times of 10, 6, 2, 4, and 8 minutes. Their (externally determined) priorities are 3, 5, 2, 1, and 4, respectively, with 5 being the highest priority. For each of the following scheduling algorithms, determine the mean process turnaround time. Ignore process switching overhead.

(a) Round robin. (b) Priority scheduling. (c) First-come, first-served (run in order 10, 6, 2, 4, 8). (d) Shortest job first.

For (a), assume that the system is multiprogrammed, and that each job gets its fair share of the CPU. For (b) through (d), assume that only one job at a time runs, until it finishes. All jobs are completely CPU bound.

Process Time Priority

A 10 3

B 6 5

C 2 2

D 4 1

E 8 4

a) Round robin: (Assuming 1min quantum)

Process Turnaround time (min)

A 30

B 23 Mean process turnaround time = (30+23+8+17+28)/5

C 8 = 106/5 = 21.2min

D 17

E 28

b) Priority:

Process Turnaround time (min)

A 24

B 6 Mean process turnaround time = (24+6+26+30+14)/5

C 26 = 100/5 = 20min

D 30

E 14

c) FCFS:

Process Turnaround time (min)

A 10

B 16 Mean process turnaround time = (10+16+18+22+30)/5

C 18 = 96/5 = 19.2min

D 22

E 30

d) SJF

Process Turnaround time (min)

A 30

B 12 Mean process turnaround time = (30+12+2+6+20)/5

C 2 = 70/5 = 14min

D 6

E 20

46. A process running on CTSS needs 30 quanta to complete. How many times must it be swapped in, including the very first time (before it has run at all)?

Swap Turnaround time (min) Remaining Time (min)

1 1 29

2 2 27

3 4 23

4 8 15

5 16 0 Complete

Number of swap = 5