EECE7095 - Homework 4

Wayne Stegner

March 17, 2021

1 Question 1

$5k jobs/month * \$2 = \$10k/month$

With deadlock: $2 deadlocks/month * 10/2 jobs/deadlock * \$2 = \$20 loss/month$

Deadlock avoidance: $\$10k/month * 0.1 overhead = \$1k overhead/month$

Part a.

Using the deadlock avoidance program could ensure that deadlock never happens, and all 5k jobs/month will still be able to run. This also removes some manual work, which reduces the possibility of human-error.

Part b.

In terms of CPU time, deadlocks only cost \$10 each time (10 jobs half way done @ \$2 each full job), costing only \$20/month. Assuming the program cost scales linearly with execution time, the deadlock avoidance program will add 10% to the monthly bill, which will cost \$1k/month. The deadlock avoidance costs many times more than just restarting processes when a rare deadlock occurs. Additionally, the 20% increase in turnaround time will probably upset whatever clients are waiting for the jobs, which may cause the clients to use another service with better turnaround time.

2 Question 2

Part a.

Deadlock cannot occur in this system because it utilized pre-emption. If a process is using resources, it will either complete and free the resources, or it will block and then other processes may pre-empt its current resource allocation if needed.

Part b.

An indefinite block can occur if a process is continuously pre-empted. In the given example, P2 takes the first resource from P0, causing P0 to be blocked by both the first and third resources of the system. If processes continuously pre-empt P0 in this way, it cannot finish and will block forever.

3 Question 3

Thread	Allocation				Max				Need			
	A	B	C	D	A	B	C	D	A	B	C	D
T0	3	0	1	4	5	1	1	7	2	1	0	3
T1	2	2	1	0	3	2	1	1	1	0	0	1
T2	3	1	2	1	3	3	2	1	0	2	0	0
T3	0	5	1	0	4	6	1	2	4	1	0	2
T4	4	2	1	2	6	3	2	5	2	1	1	3

Part a.

Available				Possible	Remaining
A	B	C	D	Threads	Threads
0	3	0	1	T2	T0, T1, T3, T4
3	4	2	2	T1	T0, T3, T4
5	6	3	2	T3	T0, T4
5	11	4	2	None	T0, T4

The system is in deadlock because neither T0 nor T4 can run. They are both waiting for more of resource D to become available.

Part b.

Available				Possible	Remaining
A	B	C	D	Threads	Threads
1	0	0	2	T1	T0, T2, T3, T4
3	2	1	2	T2	T0, T3, T4
6	3	3	3	All	None

After running T1 and T2, enough resources become available to finish any of the tasks. Because completing a task will only allocate more resources and not take away resources, T0, T3, and T4 can be run in any order with no deadlock.

4 Question 4

Allowing two page table entires to point to the same page frame allows multiple processes to share the same data while saving space. This allows large amounts of memory to be "copied" without actually copying the whole memory page, and instead making a new page table entry point to the old page. If a shared page must be modified, the Copy on Write strategy can be used, so that the page will actually be copied if any of the page table entries must update some byte of the shared page frame.

5 Question 5

Part a.

Logical memory size is $number\_of\_pages * size\_of\_each\_page$. With 64 pages, each containing 1024 bytes, the logical memory size is $64 * 1024 = 2^6 * 2^{10} = 2^{16} bytes$. Thus, the logical address must contain 16 bits.

Part b.

Physical memory size is $number\_of\_frames * size\_of\_each\_frame$. With 32 frames, each also containing 1024 bytes, the physical memory size will be half of the logical memory size, or $2^{15} bytes$. Thus, the physical address must contain 15 bits.

6 Question 6

First fit:

Each process is placed in the first partition large enough for it:

Partition	300KB	600KB	350KB	750KB	125KB
Process	115KB	500KB	200KB	358KB	(empty)
Waste	185KB	100KB	150KB	392KB	125KB

Unable to allocate: 375KB.

Total waste: 952KB.

Best fit:

Each process is placed in the available memory partition which wastes the least amount of space:

Partition	300KB	600KB	350KB	750KB	125KB
Process	200KB	500KB	(empty)	358KB	115KB
Waste	100KB	100KB	350KB	392KB	10KB

Unable to allocate: 375KB.

Total waste: 952KB.

Worst fit:

Each process is placed in the available memory partition which wastes the most amount of space:

Partition	300KB	600KB	350KB	750KB	125KB
Process	(none)	500KB	200KB	115KB	(none)
Waste	300KB	100KB	150KB	635KB	125KB

Unable to allocate: 358KB and 375KB.

Total waste: 1310KB.

First fit and Best fit tie for wasting the least amount of space because they are both unable to place the same process. Worst fit is the least efficient, as it wastes the most space.

7 Question 7

Page number is obtained by $floor(address/page\_size)$. Page offset is found as $address\%page\_size$.

Part a.

$Page\_number = floor(3085/1024) = 3$

$Page\_offset = 3085\%1024 = 13$

Part b.

$Page\_number = floor(42095/1024) = 41$

$Page\_offset = 42095\%1024 = 111$

Part c.

$Page\_number = floor(215201/1024) = 210$

$Page\_offset = 215201\%1024 = 161$

Part d.

$Page\_number = floor(650000/1024) = 634$

$Page\_offset = 650000\%1024 = 784$

Part e.

$Page\_number = floor(200000/1024) = 195$

$Page\_offset = 200000\%1024 = 320$

8 Question 8

Part a.

$Virtual\_memory\_size = number\_of\_pages * page\_size$

$Number\_of\_pages = 2^{21}/2^{11} = 2^{10}$

There are 1024 entries in the conventional page table.

Part b.

$Number\_inverted\_pages = number\_physical\_frames$

$Number\_inverted\_pages = physical\_memory/frame\_size = 2^{16}/2^{11} = 2^5 = 32entries$

There are 32 entries in the inverted page table.

Part c.

$Maximum\_physical\_memory = 2^{16}bytes = 65536bytes = 64KB$

9 Question 9

Part a.

$Virtual\_address\_bits = log_2(4KB * 256) = log_2(2^{12} * 2^8) = 20bits$

It takes 20 bits to address the logical memory space.

Part b.

$Physical\_memory\_bits = log_2(4KB * 64) = log_2(2^{12} * 2^6) = 18bits$

It takes 18 bits to address the physical memory space.

10 Question 10

Part a.

$Virtual\_memory\_size = number\_of\_pages * page\_size$

$Number\_of\_pages = 2^{32}/2^{12} = 2^{20} = 1048576entries$

There are 1048576 entries in the conventional page table.

Part b.

$Number\_inverted\_pages = number\_physical\_frames$

$Number\_inverted\_pages = physical\_memory/frame\_size = 2^{29}/2^{12}$

$= 2^{17}entries = 131072entries$

There are 131072 entries in the inverted page table.