Nicholas Garrett

Professor Fouda

CS 4461

2/27/2022

Homework 4

Chapter 20

20.1

I think we only need 1 register in either a two-level or a three-level table. This is because all levels before the last level store pointers.

20.2

seed	virtual address	translate
0	611c	08
0	3da8	not valid
0	17f5	1c
0	7f6c	not valid
0	0bad	not valid
0	6d60	not valid
0	2a5b	not valid
0	4c5e	not valid
0	2592	1b
0	3e99	1e

seed	virtual address	translate
1	6c74	06
1	6b22	1a
1	03df	0f
1	69dc	not valid
1	317a	1e
1	4546	not valid
1	2c03	16
1	7fd7	not valid
1	390e	not valid
1	748b	not valid

seed	virtual address	translate
2	7570	not valid
2	7268	16
2	1f9f	not valid
2	0325	0b
2	64c4	not valid
2	0cdf	00
2	2906	not valid
2	7a36	09
2	21e1	not valid
2	5149	1b

There were three memory lookups.

20.3

I think that because of the extra memory lookups required for the page table, a cache would be somewhat slower using a table.

Chapter 21

21.1

While running this command, the number of interrupts increases significantly. Similarly, the time spent running non-kernal code also increased, this is code that us run by the "user". I think it it makes sense that a program initiated by the user increases the user time, escpecially since it does not perform many (if any) kernal operations.
And when multiple instances of mem are run, the user time seems to share a linear relation to the number of instances running.

21.2

Running the mem script, the free memory decreased significantly, by about 1046786. This change indicates that the memory reported is in units of kb and is pretty close to the memory allocated to mem.c's program. I think this seems reasonable. Then, when I ended the program, the free memory returned to its original amount. As for the swpd, it was 0 regardless of whether the program was active or not.

21.3

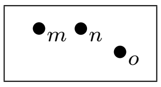
On my machine, a VM, I have 3 Gb of memory dedicated to it. When I ran mem.c with about 1 Gb allocated to it, the "si" terms would occasionally hit 12 or 28, but "so" was effectively static at zero. The same is true for 2 Gb allocated for mem.c. When I pushed it to 2.5 Gb, the "si" and "so" values went into the 40,000 - 60,000 range., though they dropped back down to lower values after some time. "si" came to the general range of 50 - 150 and "so" centered around 0. Adding it up, the data swapped looks to be about 420000 with 60,000 being swapped each cycle for about 7 seconds. Unfortunately, I do not understand where these numbers came from.

21.4

Performing the same experiment, at 1024 Mb allocated to mem.c, most of the changes were in free space. At 2000 Mb, the free space drop was similarly present, the block i/o statistics seem unstable, fluctuating between numbers of base 2 at a maximum of 120.
At 2.5 Gb, the free space dropped significanlty, as did the buffer and cache. However, both I/O operations “bi” and “bo” increased significantly. And the cpu’s time spent in kernal mode increased slightly.

21.5

For the first set, the input I chose was 1024. This is about 33% of the my RAM setting. The first loop takes almost 800 ms to complete, but drops to a bit over 600 secodnds and then 675ms in subsequent loops. For the second se, the allocation clearly beyond the available memory returned a memory allocation failure. Between the two, having space to allocate alows for us to work with it fairly easily.
Graph x(1024, 2048, 3048) and y(1294, 1291, 678):



I have no idea how to get the labels into this plot. But, the first two points, m and n, are for memory allocations well within the space, and o is for a memory allocation fairly close to the available space. The first loop typically had a higher bandwidth thatn the second loop, in the case of memory cases with memory very close to the maximum, the difference in bandwidth was $\frac{678}{49}$.

21.6

As the allocations for mem.c are increased, the availale space decreases. Around the meomory size of the machine, memory allocation fails.

21.7

Chapter 22

22.1

seed	access	FIFO H/M	LRU H/M	OPT H/M
0	8	miss	miss	miss
0	7	miss	miss	miss
0	4	miss	miss	miss
0	2	miss	miss	miss
0	5	miss	miss	miss
0	4	hit	hit	hit
0	7	miss	miss	hit
0	3	miss	miss	miss
0	4	miss	hit	hit
0	5	miss	miss	hit

seed	access	FIFO H/M	LRU H/M	OPT H/M
1	1	miss	miss	miss
1	8	miss	miss	miss
1	7	miss	miss	miss
1	2	miss	miss	miss
1	4	miss	miss	miss
1	4	hit	hit	hit
1	6	miss	miss	miss
1	7	miss	miss	hit
1	0	miss	miss	hit
1	0	hit	hit	hit

seed	access	FIFO H/M	LRU H/M	OPT H/M
2	9	miss	miss	miss
2	9	hit	hit	hit
2	0	miss	miss	miss
2	0	hit	hit	hit
2	8	miss	miss	miss
2	7	miss	miss	miss
2	6	miss	miss	miss
2	3	miss	miss	miss
2	6	hit	hit	hit
2	6	hit	hit	hit

.

22.2

Worst case for FIFO: repeating sequence of 6 accesses. (1,2,3,4,5,6, 1,2,3,4,5,6,...). For this worst case, no hits are possible. However, if we add even a single term to the cache limit, then the worst case breaks and all accesses after the first 6 will hit.
Worst case for LRU: the same squence as is the worst case for FIFO will (1,2,3,4,5,6, 1,2,3,4,5,6, ...) cause a worst-case scenario in LRU. Though, as with FIFO, increasing the cache by 1 fixes this case.
Worst case for MRU is where there are the maximum number of accesses before repetition. This would cause the efficiency to be $\frac{cachsize}{maxpage}$ for an infinitely long trace. Though, this has the same hit/miss rate as OPT, so I am not sure how better to design a trace.

22.3

I decided to limit the generation to a trace of 10 accesses. The return was [4, 10, 10, 6, 10, 3, 0, 6, 1, 0]. I would expect FIFO with a cache size of 3 to hit four of the ten accesses. I would expect LRU with a cache size of 3 to hit three of the ten accesses, as 6 would be dropped in favor of 0 on its access call. I would expect OPT with a cache size of 3 to hit four of the ten accesses, as they are repeated and the cache can hold them. I would expect MRU with a cache size of 3 to hit three of the ten accesses, as 10 would be dropped after the second access.

22.4

An easy trace to generate is [1,2,3,4,3,2,1,2,3,4,3,2,1,2,3,4,3,2,1], as each term is ± 1 compared to the previous term.
LRU made 10 hits and 9 misses.
RAND made 9 hits and 10 misses, giving LRU A +1 success.
CLOCK made 7 hits and 12 misses. However, with bits of anything besides zero, there was no change in CLOCK's return. When the defined clock bits was zero, the number of hits climbed to 8 with 13 misses.

22.5