CSC369H
Assignment 3

1 Simleloop

Trace with simple loop of size 50

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	7247	2977	136	2791	2927	70.8822
fifo	7261	2979	128	2801	10240	70.9082
lru	7456	2784	65	2669	10240	72.8125
clock	7417	2799	73	2676	10216	72.6018
opt	7571	2669	18	2601	10240	73.9355

Trace with simple loop of size 100

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	7443	2781	37	2644	2681	72.7993
fifo	7483	2757	32	2625	2657	73.0762
lru	7555	2685	2	2583	2585	73.7793
clock	7549	2691	5	2586	2591	73.7207
opt	7597	2643	0	2543	2543	74.1895

Trace with simple loop of size 150

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	7512	2712	11	2551	2562	73.4742
fifo	7523	2717	8	2559	2567	73.4253
lru	7557	2683	0	2533	2533	73.7988
clock	7555	2685	0	2535	2535	73.7793
opt	7597	2643	0	2493	2493	74.1895

Trace with simple loop of size 200

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	7512	2712	11	2551	2562	73.4742
fifo	7531	2709	6	2503	2509	73.5449
lru	7557	2683	0	2483	2483	73.7988
clock	7556	2684	0	2484	2484	73.7981
opt	7577	2639	0	2439	2439	74.1680

2 Matmul

Using tr-matmul of size 50

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	1892322	995582	478236	517296	995532	65.5258
fifo	1760642	1127270	541683	585537	1127220	60.9659
lru	1846682	1041230	520102	521078	1041180	63.9452
clock	1846645	1041267	520117	521100	1041217	63.9439
opt	2300455	587457	293414	293993	587407	79.6581

Trace with tr-matmul memsize 100

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2564985	322919	157859	164960	322819	88.8182
fifo	1804360	1083552	530671	552781	1083452	62.4797
lru	1881456	1006456	502791	503565	1006356	65.1794
clock	1846895	519983	520934	520934	1040917	63.9526
opt	2795114	34410	46008	46690	92698	96.7867

Trace with tr-matmul memsize 150

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2791163	96741	47273	49318	96591	96.6501
fifo	2853502	34410	16665	17595	34260	98.7987
lru	2855025	32887	16018	16719	32737	98.8612
clock	2854702	33210	16130	16930	33060	98.8500
opt	2861297	26615	12924	13541	26465	99.0784

Trace with tr-matmul memsize 200

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2831491	56413	27482	28731	56213	98.0466
fifo	2854023	33889	16250	17439	33689	98.8265
lru	2855036	32876	15985	16691	32676	98.8616
clock	28550069	32903	15988	16715	32703	98.8607
opt	286847	19265	9238	9287	19065	99.3329

3 Blocked

Trace with block with memsize 50

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2409843	8285	3003	5232	8235	99.6574
fifo	2411683	6501	2116	4335	6451	99.7312
lru	2412965	5219	1435	3734	5169	99.7842
clock	2412917	5267	1467	3750	5217	99.7822
opt	2414475	3709	1302	2339	3659	99.8466

Trace with block with memsize 100

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2412881	5247	1840	3307	5147	99.7830
fifo	2413846	4338	1397	2841	4238	99.8206
lru	2414398	3789	1328	2358	3686	99.8434
clock	2414144	4040	1344	2606	3940	99.8329
opt	241574	3010	1043	1867	2910	99.8755

Trace with block with memsize 150

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2413698	4430	1536	2744	4280	99.8168
fifo	2413958	4226	1366	2710	4076	99.8252
lru	2414415	3769	1318	2301	3619	99.8441
clock	2414249	3935	1326	2459	3785	99.8373
opt	2415657	2527	829	1548	2377	99.8955

Trace with block with memsize 200

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	2414273	3855	1338	2317	3655	99.8406
fifo	2415010	3174	1001	1973	2974	99.8687
lru	2414488	3696	1281	2215	3496	99.8472
clock	2414996	3188	1065	1923	2988	99.8682
opt	2415907	2277	655	1422	2077	99.9058

4 Fourth program (submitted to markus as fourth.c)

Brief expalantion of what we are doing here: we tried to implement what is referred to as "looping sequential", the worst case program for fifo and lru, making them perform much worse than rand algorithm in some cases. What this program does it that we refer to 101 pages in sequence, starting at 0,1, ..., up to page 100, and then we loop, repeating those accesses, for a total of 10,000 accesses to 101 unique pages. The reason why it is bad for fifo and lru is that as soon as the physical memory is full, these 2 algorithms start to evict pages that will be referred to soon while keeping pages that will be needed longer in the future. As we can see in the table, since the program refer to 101 unique pages, when the physical is of size 50 and 100, fifo and lru(also clock, its approximation) perform very poorly with a hit rate below 50%, while the hit rate of rand algorithm in these two cases are as high as 54.2411% and 94.4975% respectively. Of course, as soon as the memsize becomes larger than the number of unique pages we are referring to in the program, the hit rate in fifo, clock and lru are tremendously increased and are larger than the hit rate of rand.

Trace with block with memsize 50

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	10449	8815	127	8638	8765	54.2411
fifo	8657	10607	150	10407	10557	44.9387
lru	9901	10173	30	10093	10123	47.1917
clock	9076	10188	38	10100	10138	47.1138
opt	13777	5487	13	5424	5437	71.5168

Trace with block with memsize 100

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	18204	1060	9	951	960	94.4975
fifo	8926	10338	54	10184	10238	46.3351
lru	9129	10135	1	10034	10035	47.3889
clock	9123	10141	5	10036	10037	47.3578
opt	18733	531	7	424	431	97.2436

Trace with block with memsize 150

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	18985	279	1	128	129	98.5517
fifo	19032	232	0	82	82	98.7957
lru	19048	216	0	66	66	98.8787
clock	19033	231	0	81	81	98.8009
opt	19050	214	0	64	64	98.8891

Trace with block with memsize 200

Algo	Hit count#1	Miss count#2	Clean evictions#3	Dirty eviction#4	Total eviction#5	Hit rate#6
rand	19031	233	0	33	33	98.7905
fifo	19042	222	0	22	22	98.8476
lru	19050	214	0	14	14	98.8891
clock	19043	221	0	21	21	98.8528
opt	19050	214	0	14	14	98.8891

5 Comparing various algorithms

When memsize is fixed and same tracefile is used, the opt algorithms has the highest hit rate, the lru algorithm has a hit rate between that of fifo and opt. In some cases, rand has higher hit rate than fifo, lru and clock.

As a brief explanation, since the opt algorithm can see the future(the future memory reference), it can evict the page that is referenced furthest which minimizes the miss rate. The lru algorithm evicts the least recently referenced page while fifo evicts the page that is introduced first regardless of if the page has been referenced recently or not. Since lru is based on spatial and temporal locality, in the case that data are not randomly accessed, the lru algorithm tends to behave better than rand, otherwise rand will behave equally well or even better. As for the clock algo, it is an approximation of the lru algorithm and so it tends to have similar hit rate as the lru, more precisely, less hit rate in almost every case, except with tr-blocked.ref when memsize is 200.

When tracing the simple loop, since the data is likely to be random, the algorithms do not behave very well.
When tracing matmul, spatial and temporal locality applies and algorithms behaves well when memsize is large enough. When mensize is not large enough, thrashing prevents them from reaching a high hit rate.
When tracing blocked, spatial and temporal locality applies here and so the algo behaves well. Since we are not referring to too many unique pages in this case, the problem of thrashing won't arise, as did in the case of matrix multiplication.

6 Data for LRU as size of memory increases

Question: comment on what you notice about the hit rate (does it increase or decrease and explain why). How does it compare to other algorithms? What do you notice for a large trace like matmul?
Generally speaking, the hit rate of lru algorithm increases as the memory size increases, unlike fifo, lur won't suffer from belady's anomaly.

Compared to other algorithms, the hit rate of lru is always between that of fifo and opt, as we discussed above, and as long as there is locality and thrashing doesn't happen, lru will behave better than rand.

Since the clock algorithm is an approximation of lru algorithm, the two algorithms have really close hit rates.

While tracing the memory usage for matmul, the hit rate of lru increases sharply as memsize increases from 100 to 150. This is because when memsize if less than or equal to 100, the lru algorithm suffers from thrashing, and when memsize increases to 150, the problem of thrashing is gone.