TABLES

The fourth program is mean-var.c, which sums the numbers 1 to 1000 and prints the mean and variance of this sum. We added it to observe how the algorithms perform for a simple program that does not use up all the memory (case memory size = 200).

Trace file = tr-simpleloop.ref
Memory size = 50

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	70.8756	7269	2987	2937	345	2592
FIFO	71.0998	7292	2964	2914	318	2596
LRU	72.9037	7477	2779	2729	202	2527
CLOCK	72.8159	7468	2788	2738	209	2529
OPT	74.0250	7592	2664	2614	108	2506

Trace file = tr-simpleloop.ref
Memory size = 100

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	73.1377	7501	2755	2655	161	2494
FIFO	73.1572	7503	2753	2653	158	2495
LRU	73.8787	7577	2679	2579	113	2466
CLOCK	73.8592	7575	2681	2581	114	2467
OPT	74.2882	7619	2637	2537	38	2499

Trace file = tr-simpleloop.ref
Memory size = 150

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	73.5862	7547	2709	2559	134	2425
FIFO	73.5569	7544	2712	2562	129	2433
LRU	73.8885	7578	2678	2528	112	2416
CLOCK	73.8690	7576	2680	2614	112	2418
OPT	74.2882	7619	2637	2487	2	2485

Trace file = tr-simpleloop.ref
Memory size = 200

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	73.6252	7557	2705	2505	130	2375
FIFO	73.6349	7552	2704	2504	125	2379
LRU	73.8885	7578	2678	2478	112	2366
CLOCK	73.8787	7577	2679	2479	112	2367
OPT	74.2882	7619	2637	2437	2	2435

Trace file = tr-matmul.ref
Memory size = 50

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	65.5531	1893136	994808	994758	955756	39002
FIFO	60.9666	1760680	1127264	1127214	1083353	43861
LRU	63.9458	1846720	1041224	1041174	1040195	979
CLOCK	63.9454	1846706	1041238	1041188	1040204	984
OPT	79.6584	2300491	587453	587403	586440	963

Trace file = tr-matmul.ref
Memory size = 100

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	88.7958	2564373	323571	323471	316133	7338
FIFO	62.4805	1804402	1083542	1083442	1061337	22105
LRU	65.1499	1881494	1006450	1006350	1005389	961
CLOCK	63.0468	1846932	1041012	1040912	1039949	963
OPT	96.7869	2795151	92793	92639	91732	961

Trace file = tr-matmul.ref
Memory size = 150

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	96.6819	2792120	95824	95674	93427	2247
FIFO	98.8087	2583539	34405	34255	33059	1196
LRU	98.8614	2855061	32883	32733	31772	961
CLOCK	98.8503	2854740	33204	33054	32090	964
OPT	99.0786	2861334	26610	26460	25499	961

Trace file = tr-matmul.ref
Memory size = 200

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	98.0349	2831192	56752	56552	55046	1506
FIFO	98.8267	2854060	33884	33684	32549	1135
LRU	98.8618	2855073	32871	32401	31710	691
CLOCK	98.8609	2855047	32897	32697	31736	961
OPT	99.3331	2868684	19260	19060	18100	960

Trace file = tr-blocked.ref

Memory Size = 50

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	99.6520	2409769	8415	8365	5988	2377
FIFO	99.7323	2411710	6474	6424	4299	2125
LRU	99.7844	2412971	5213	5163	2943	2220
CLOCK	99.7831	2412940	5244	5194	2979	2215
OPT	99.8467	2414478	3706	3656	2695	961

Trace file = tr-blocked.ref

Memory Size = 100

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	99.7827	2412929	5255	5155	3544	1611
FIFO	99.8208	2413850	4334	4234	2876	1358
LRU	99.8436	2414403	3781	3681	2720	961
CLOCK	99.8339	2414167	4017	3917	2731	1186
OPT	99.8757	2415178	3006	2906	1956	950

Trace file = tr-blocked.ref

Memory Size = 150

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	99.8161	2413736	4448	4298	2902	1396
FIFO	99.8254	2413962	4222	4072	2770	1302
LRU	99.8443	2414419	3765	3615	2674	941
CLOCK	99.8372	2414246	3938	3788	2691	1097
OPT	99.8956	2415660	2524	2374	1424	950

Trace file = tr-blocked.ref

Memory Size = 200

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	99.8404	2414324	3860	3660	2436	1224
FIFO	99.8688	2415012	3172	2972	1995	977
LRU	99.8473	2414492	3692	3492	2551	941
CLOCK	99.8675	2414979	3205	3005	2063	942
OPT	99.9060	2415910	2274	2074	1129	945

Trace file = myprogram.ref

Memory Size = 50

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	94.2274	6513	399	349	303	46
FIFO	94.8640	6557	355	305	254	51
LRU	95.8478	6625	287	237	214	23
CLOCK	95.4282	6596	316	266	237	29
OPT	96.9763	6703	209	159	144	15

Trace file = myprogram.ref

Memory Size = 100

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	96.9039	6698	214	114	94	20
FIFO	97.2656	6723	189	99	73	16
LRU	97.5550	6743	169	69	65	4
CLOCK	97.4971	6739	173	73	69	4
OPT	97.7575	6757	155	55	53	2

Trace file = myprogram.ref

Memory Size = 150

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	97.7431	6756	156	6	4	2
FIFO	97.7431	6756	156	6	3	3
LRU	97.7575	6757	155	5	5	0
CLOCK	97.7575	6757	155	5	3	2
OPT	97.7575	6757	155	5	4	1

Trace file = myprogram.ref

Memory Size = 200

Algorithm	Hit Rate	Hit Count	Miss count	Overall Eviction count	Clean Eviction count	Dirty Eviction Count
rand	97.7575	6757	155	0	0	0
FIFO	97.7575	6757	155	0	0	0
LRU	97.7575	6757	155	0	0	0
CLOCK	97.7575	6757	155	0	0	0
OPT	97.7575	6757	155	0	0	0

Comparison of algorithms

To compare the algorithms, we will be comparing the hit rate and the overall eviction count when running the algorithm against traces of different programs with different memory size. OPT always has the highest hit rates and lowest eviction count because it reads the trace file ahead, and always picks the page that will never be used again or the page that will not be used for the longest period (next best thing). Generally, LRU is the second most accurate algorithm, followed extremely closely by CLOCK. This happens because both algorithms try to find an “old” page to evict. However, while LRU actually looks up the oldest page, CLOCK only looks for “old enough” page, which is a tradeoff between accuracy and speed. There is a good chance that a page that hasn’t been used for a while might not be needed anymore, which is why evicting an old page would result in higher hit rates and lower future evictions. Lastly, FIFO and RAND have the worst accuracy and most evictions, with FIFO being better in most cases. RAND does seem to do better than others (not including OPT) in tr-matmul because the data in matmul is randomly/dynamically generated, so RAND gets a chance to perform better, otherwise picking a page randomly to evict would not increase hit rates. Lastly, if you look at the last table, you’ll see that all algorithms do the same. This is because if memory is big enough to store everything, then there are no evictions, and the misses simply occur when we first load the data for the first time.

LRU description

LRU replaces pages by evicting the page that has not been used for the longest time in the past. The idea behind LRU is that pages that have been used recently are more likely to be used again in the near future. The hit rate of LRU generally increases as memory size increases. This is because as memory size increases, there are more pages that are considered for eviction during replacement, and thus the probability that the least recently used page out of the group is one that will not be used in the near future increases.

For example, with a memory size of 100, a program that repeatedly uses the same 150 pages to perform a set of instructions will have to repeatedly replace several pages in the page table in order to execute. However, with a memory size of 150, the program can simply store the entirety of the 150 pages in memory at once, reducing the need to replace pages, and thereby increasing the hit rate and reducing the eviction count.

This is what happens when LRU is used with matmul. The hit rate jumps from around 65% with a memory size of 100 to around 98% with a memory size of 150. As well, the total eviction count drastically reduces from around 1,000,000 (with memory size 100) to around 37,000 (with memory size 150). This suggests that matmul uses about the same 150 pages of memory to execute certain sections of code. By having access to 150 pages of memory as opposed to 100, the number of evictions drastically reduces when executing these sections.