CSC369-A2

Tables

1. Memory size: 50

Table 1: tr-simpleloop.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	70.9050	7255	2977	2927	342	2585
FIFO	70.8855	7253	2979	2929	333	2596
Clock	72.6642	7435	2797	2747	218	2529
LRU	72.7912	7448	2784	2734	208	2526
OPT	73.9152	7563	2669	2619	112	2507

Table 2: tr-matmul.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	65.5208	1892179	995725	995675	956307	39368
FIFO	60.9658	1760634	1127270	1126920	1083360	43860
Clock	63.9438	1846637	1041267	1041217	1040234	983
LRU	63.9451	1846674	1041230	1041180	1040201	979
OPT	79.6580	2300447	20.3420	587407	586445	962

Table 3: tr-blocked.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	99.6530	2409737	8391	8341	5962	2379
FIFO	99.7318	2411642	6486	6436	4310	2126
Clock	99.7819	2412855	5273	5223	3011	2212
LRU	99.7841	2412908	5220	5170	2949	2221
OPT	99.8466	2414418	3710	5170	2700	960

Table 4: tr-interesting.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	11.2583	34	268	218	145	73
FIFO	16.5563	50	252	202	129	73
Clock	16.5563	50	252	202	129	73
LRU	16.5563	50	252	202	129	73
OPT	16.5563	50	252	202	140	62

2. Memory size: 100

Table 5: tr-simpleloop.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	72.8206	7451	2781	2681	180	2501
FIFO	73.0551	7475	2757	2657	163	2494
Clock	73.7002	7541	2691	2591	123	2468
LRU	73.7588	7547	2685	2585	120	2465
OPT	74.1693	7589	2643	2543	43	2500

Table 6: tr-matmul.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	88.8183	2564987	322917	322817	315492	7325
FIFO	62.4796	1804352	1083552	1083452	1061345	22107
Clock	63.9525	1846887	1041017	1040917	1039953	964
LRU	65.1493	1881448	1006456	1006356	1005396	960
OPT	96.7867	2795106	92798	92698	91738	960

Table 7: tr-blocked.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	99.7831	2412882	5246	5146	3539	1607
FIFO	99.8206	2413790	4338	4238	2881	1357
Clock	99.8329	2414088	4040	3940	2746	1194
LRU	99.8434	2414341	3787	3687	2727	960
OPT	99.8755	2415117	3011	2911	1963	948

Table 8: tr-interesting.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	25.1656	76	226	126	81	45
FIFO	33.1126	100	202	102	65	37
Clock	33.1126	100	202	102	65	37
LRU	33.1126	100	202	102	59	43
OPT	33.1126	100	202	102	70	32

3. Memory size: 150

Table 9: tr-simpleloop.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	73.4949	7520	2712	2562	137	2425
FIFO	73.4461	7515	2717	2567	135	2432
Clock	73.7588	7547	2685	2535	118	2417
LRU	73.7783	7549	2683	2533	118	2415
OPT	74.1693	7589	2643	2493	2	2491

Table 10: tr-matmul.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	96.6502	2791164	96740	96590	94334	2256
FIFO	98.8085	2853494	34410	34260	33065	1195
Clock	98.8500	2854694	33210	33060	32096	964
LRU	98.8612	2855017	32887	32737	31777	960
OPT	99.0784	2861289	26615	26465	25505	960

Table 11: tr-blocked.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	99.8168	2413699	4429	4279	2910	1369
FIFO	99.8252	2413901	4227	4077	2776	1301
Clock	99.8369	2414184	3944	3794	2698	1096
LRU	99.8441	2414358	3770	3620	2680	940
OPT	99.8954	2415599	2529	2379	1427	952

Table 12: tr-interesting.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	49.6689	150	152	2	2	0
FIFO	49.6689	150	152	2	1	1
Clock	49.6689	150	152	2	1	1
LRU	49.6689	150	152	2	1	1
OPT	49.6689	150	152	2	1	1

4. Memory size: 200

Table 13: tr-simpleloop.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	73.4949	7520	2712	2512	134	2378
FIFO	73.5242	7523	2709	2509	131	2378
Clock	73.7686	7548	2684	5368	2684	2684
LRU	73.7783	7549	2683	2483	118	2365
OPT	74.1693	7589	2643	2443	2	2441

Table 14: tr-matmul.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	98.0466	2831493	56411	56211	54725	1486
FIFO	98.8265	2854015	33889	33689	32555	1134
Clock	98.8607	2855001	32903	32703	31743	960
LRU	98.8616	2855028	32876	32676	31716	960
OPT	99.3329	2868639	19265	19065	18105	960

Table 15: tr-blocked.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	99.8405	2414272	3856	3656	2434	1222
FIFO	99.8686	2414951	3177	2977	2001	976
Clock	99.8681	2414938	3190	4098	2049	2049
LRU	99.8471	2414431	3697	3497	2557	940
OPT	99.9058	2415849	2279	2079	1139	940

Table 16: tr-interesting.ref

Algorithms	Hit Rate	Hit Count	Miss Count	Overall Eviction Count	Clean Eviction Count	Dirty Eviction Count
Rand	50.00	151	151	0	0	0
FIFO	50.00	151	151	0	0	0
Clock	50.00	151	151	0	0	0
LRU	50.00	151	151	0	0	0
OPT	50.00	151	151	0	0	0

Analysis & Comparison

Regardless of size, the **OPT** algorithm has the highest hit rate (and hence the lowest overall eviction count) of all the algorithms tested, which is consistent with what we would expect in theory since we have absolute knowledge of the future. The **Rand** algorithm usually has the lowest hit rate, except for the traces of the matmul program with memory sizes 50 and 100 because unlike other programs the matmul program has poor locality, meaning that pages are scattered more sparsely across the memory, and therefore **Rand,** has a chance to have a better performance than the algorithms that depend on locality. The rest of the algorithm generally has the hit rates following: **Rand** < **FIFO** < **LRU** < **CLOCK** << **OPT**. **FIFO** occasionally has slightly worse hit rate than **Rand**, and this might be due to the Belady's anomaly which means that the fault rate might actually increase with more memory. The blocked.ref has overall much better hit rate than the rest of the ref files because the block matrix multiplication is efficient in terms of memory allocation. The matrices are divided into smaller blocks so that these blocks ideally fit in the same page so when we do the matrix arithmetic in the same page, it requires less memory access; also, with the same reason **Rand** does poorly in comparison to other algorithms since such programs exploit the locality of programs. Additionally, **Clock's** hit rate is very close to **LRU's** hit rate since clock essentially **approximates** the **LRU** algorithm and it does not do the overhead of manipulating any bits or shifting the stack.

On the other hand, as the memory size increases, LRU performs better and better. On average, the hit rate for this algorithm increases from 63% for a memory size of 50 to about 80% for a memory size of 200. The overall eviction count also decreases significantly, from about 25,000 on average for memory sizes of 50 and 100 to about 9,700 for memory sizes of 150 and 200. The clean eviction count, on average, follows a similar pattern as the overall eviction count: from about 25,000 for memory sizes of 50 and 100 to about 8,600 for memory sizes of 150 and 200. The dirty eviction count, on the contrary, although it decreases like the overall eviction count, it does so more slowly: from about 1500 for a memory size of 50 to about 1000 for a memory size of 200. LRU in all cases has consistently the best performance among all non-random algorithms and the closest hit rate to OPT. Because unlike FIFO, LRU has complete knowledge of the past accesses and accounts for their time frames, this is used very effectively to predict the future and minimize page faults.

Our custom trace file is interesting because it accesses the memory in a very predictable pattern. It uses one contiguous chunk of memory and accesses it from the top to bottom and back again, precisely hitting every page twice. (e.g. 0-1-2-3-3-2-1-0) With this pattern all four non-random algorithms achieved the same hit rate, meaning all algorithms are performing on par with the optimal, only difference being the page replacement order indicated by the different clean/dirty counts. This is because the FIFO assumption is true in our case with the older pages to less likely be the next used page.
