README

Shoaib Shaikh and Tanuj Kumar

November 2017

1 Tables

Tables will be located at the end of the document due to size reasons.

2 Fourth Program Description

Our fourth program is **randomize.c**, which is a simple algorithm that does the following:

- Initializes a generally large-sized array on both the stack (stack array) and the heap (heap array) consisting of integers equal to zero.
- Uses a randomizer twice. First, if the randomizer chooses 1, it will visit the stack array, and if 0, it will visit the heap array.
- With the second randomizer it will take a random index and increment the integer in that specific array at that random index.

Why was this simple program chosen for interest?

- **Heap and stack**. Because the heap and stack are located and start in separate places in the virtual address space, and grow towards each other, it was thought that we would get behaviour that would involve virtual address calls in drastically different non-local locations.
- **Randomization**. On top of the above, we hoped that the randomization of adding to either the heap or the stack, and to a random index on top of that, would remove locality and lessen any kind of patterns in the virtual address accesses.

Interestingly, the hit rate of blocked generally outperformed randomize up until the saturation point of 121 misses and no evictions, which happened because the memory size of the coremap exceeded 121. Despite this, the hit rate was higher than expected, but it is presumed that a different randomization seed may affect the initial hit rate but not the increasing progression of the rate as the memory size increases.

3 Algorithm Comparisons

- **RAND**. This algorithm simply evicted by randomly choosing an index in the coremap and evicting the page at that index. Therefore, its hit rate varies heavily. In the four programs, it was not once last place for hit rate due to luck: 3rd in simpleloop, amazingly 2nd in matmul, 4th in blocked, and 4th in randomize. In terms of evictions, it was 3rd in randomize, 3rd again in matmul, and last in simpleloop and blocked.

- **FIFO**. Due to the random seed for RAND, this algorithm actually performed the worst in most situations. Simply acting as a page replacement queue, it was in 5th place for simpleloop and randomize, and 4th place in matmul, 3rd place in blocked. It also made a lot of excessive evictions, coming in 5th place in matmul and randomize.

- **LRU**. This algorithm makes use of a reference time counter, and whenever an eviction happens, the least recently used page is that with the lowest time counter. In terms of hit rate, LRU was 2nd with simpleloop, 5th in matmul, 5th in blocked, 3rd in randomize. However, it does better with regards to eviction count, being 2nd in simpleloop, 3rd in matmul, 5th in blocked, and 2nd with randomize.

- **CLOCK**. This algorithm uses a circular buffer in tandem with a "clock hand" and reference bit. It is roughly average with LRU, if not outperforming it slightly. In terms of hit rate, it is 3rd with simpleloop, 3rd with matmul, 2nd place for blocked, 2nd for randomize. For eviction counts, 3rd in simpleloop, 4th in matmul, 2nd in blocked, 3rd in randomize.

- **OPT**. The ideal algorithm, as it keeps a linked list of all traces the program makes, looking into the future by traversing and observing the farthest-in-the-future address out of those currently assigned page frames. It is 1st place in all hit rates and eviction counts, as expected.

4 LRU Analysis

- **Comparisons to other algorithms**. For brevity's sake we will not re-state this here, see the paragraph above in the section **LRU**. We will note that the results of LRU are seemingly all over the place roughly tied with CLOCK for usefulness. This will be further analyzed below.

- **Memory size increases**. In all situations with the exception of blocked at memory 200, the hit rate steadily increased and the eviction count decreased proportional to the size of memory. As the memory size increases, the eviction count decreasing makes sense because there is more base memory in the coremap in which to actually store more pages corresponding to virtual addresses and connecting to page frames at once. The hit rate's steady increase

is likely because the presence of more memory means that hits are likelier to happen since pages are stored for a longer time in the coremap.

- **Large trace sizes**. Where LRU noticeably fails is with trace sizes that are extremely large. In comparison, simpleloop and randomize were about 108 KB and 68 KB respectively, while in contrast matmul and blocked were 28 MB and 24 MB respectively, a massive size difference. This may be due to the nature of the algorithm and the tracefiles. For one, massive tracefiles of this size typically involve accessing program code that is from many different disparate locations, inclusive of memory accesses, stores, reads, and writes, and so the immense size of this situation renders much less long-term locality in virtual address access. This creates a lot of "ties" in the least recently used pages, which would simply keep releasing the lowest index page, regardless of whether this is a popular page in the future or not, thus reducing the hit rate and requiring more evictions.

5 Table Figures

	SIMPLELOOP															
ALG	FIFO				LRU				CLOCK				OPT			
M	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
HIT RATE	71.05	73.23	73.62	73.7	73.15	74.11	74.17	74.22	72.89	73.9	73.94	73.94	74.08	74.34	74.34	74.34
HIT	7315	7540	7580	7588	7532	7630	7637	7642	7505	7608	7613	7613	7628	7654	7654	7654
MISS	2981	2756	2716	2708	2764	2666	2659	2654	2791	2688	2683	2683	2668	2642	2642	2642
CLEAN	134	33	8	6	54	5	0	0	68	4	0	0	20	0	0	0
DIRTY	2797	2623	2558	2502	2660	2561	2509	2454	2673	2584	2533	2483	2598	2542	2492	2442
ALL	2931	2656	2566	2508	2714	2566	2509	2454	2741	2588	2533	2483	2618	2542	2492	2442
	MATMUL															
ALG	FIFO				LRU				CLOCK				OPT			
M	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
HIT RATE	60.97	62.48	98.81	98.82	65.26	65.32	67.42	97.73	63.95	65.31	98.79	98.86	79.66	96.79	99.08	99.33
HIT	1760691	1804418	2853560	2854081	1884610	1886509	1947021	2822536	1846735	1886173	2853255	2855081	2300512	2795172	2861355	2868705
MISS	1127277	1083550	34408	33887	1003358	1001459	940947	65432	1041233	1001795	34713	32887	587456	92796	26613	19263
CLEAN	541689	530672	16664	16250	501200	500294	470092	32296	520106	500465	16941	15994	293417	46006	12925	9240
DIRTY	585538	552778	17594	17537	502108	501065	470705	32936	521077	501230	17622	16693	293989	46690	13538	9823
ALL	1127227	1083450	34258	33787	1003308	1001359	940797	65232	1041183	1001695	34563	32687	587406	92696	26463	19063
	BLOCKED															
ALG	FIFO				LRU				CLOCK				OPT			
M	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
HIT RATE	99.73	99.82	99.83	99.87	93.05	93.15	98.61	97.35	99.76	99.82	99.84	99.87	99.85	99.88	99.9	99.91
HIT	2411639	2413808	2413920	2414972	2250034	2252517	2384583	2354113	2412379	2413837	2414365	2414936	2414436	2415136	2415619	2415869
MISS	6505	4336	4224	3172	168110	165627	33561	64031	5765	4307	3779	3208	3708	3008	2525	2275
CLEAN	2120	1398	1366	1001	82696	81721	15946	31329	1679	1331	1324	1056	1323	1044	830	656
DIRTY	4335	2838	2708	1971	85364	83806	17645	32502	4036	2876	2305	1952	2335	1864	1545	1419
ALL	6455	4236	4074	2972	168060	165527	33591	63831	5715	4207	3629	3008	3658	2908	2375	2075
	RANDOMIZE															
ALG	FIFO				LRU				CLOCK				OPT			
M	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
HIT RATE	95.5	97.88	98.18	98.18	97.01	98.18	98.18	98.18	96.88	98.06	98.18	98.18	97.9	98.18	98.18	98.18
HIT	6364	6523	6543	6543	6465	6543	6543	6543	6456	6535	6543	6543	6524	6543	6543	6543
MISS	300	141	121	121	199	121	121	121	208	129	121	121	140	121	121	121
CLEAN	74	0	0	0	33	0	0	0	36	0	0	0	12	0	0	0
DIRTY	176	41	0	0	116	21	0	0	122	29	0	0	78	21	0	0
ALL	250	41	0	0	149	21	0	0	158	29	0	0	90	21	0	0

Figure 1: FIFO, LRU, CLOCK, and OPT Tables. Rows are: Hit Rate, Hits, Misses, Clean Evictions, Dirty Evictions, All Evictions

	RAND ALGO															
FILE	SIMPLELOOP				MATMUL				BLOCKED				RANDOMIZE			
M	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
HIT RATE	71	73.29	73.59	73.71	65.53	88.78	96.66	98.04	99.65	99.78	99.82	99.84	95.74	98.02	98.18	98.18
HIT	7311	7546	7577	7589	1892548	2563877	2791546	2831509	2409755	2412884	2413765	2414339	6380	6532	6543	6543
MISS	2985	2750	2719	2707	995420	324091	96422	56459	8389	5260	4379	3805	284	132	121	121
CLEAN	142	33	11	11	478209	158340	47099	27469	3067	1841	1524	1297	72	0	0	0
DIRTY	2793	2617	2558	2496	517161	165651	49173	28790	5272	3319	2705	2308	162	32	0	0
ALL	2935	2650	2569	2507	995370	323991	96272	56259	8339	5160	4229	3605	234	32	0	0

Figure 2: RAND Table. Rows are: Hit Rate, Hits, Misses, Clean Evictions, Dirty Evictions, All Evictions