NRU Optimal Refresh Rate

I determined that the optimum refresh rate for the Not Recently Used algorithm for page eviction (NRU) was approximately 30. In order to find there best rate, I constructed a table of refresh rates and sampled the page eviction rate and write to disk rate for refresh rates from one to one hundred, incremented by five. Once I had found these rates, I constructed the following formula for roughly determining the effectiveness of each refresh rate:

(Page Faults / 100) + Writes to Disk = Score

I reasoned that page faults weren't nearly as expensive as writes to disk, and therefore the measurement standard which we use should weigh writes to disk much more heavily than page faults. By dividing the page faults by one hundred, the average page fault rate falls around 2000-3000 which is consistently lower than the average write to disk rate of approximately 7000-8000. This means that, for any given measurement, the writes to disk is always approximately 2-3 larger and thus has a greater impact upon the score. From this list of scores, I picked the minimum and found 30 memory references before each refresh to be the optimum refresh rate.

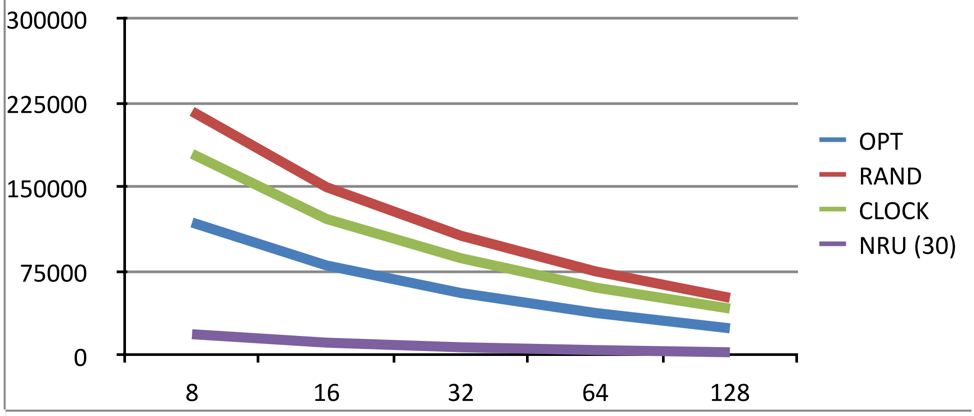
If one examines the NRU graphs, it is somewhat simple to see why thirty would be an ideal choice. The page faults generally follow a negative trend, with less and less page faults as NRU's *memory* of the past increases in size, it's able to make better decisions about what may be used again. The write to disk rate is inversely correlated with the page fault rate. This is because as NRU increases it's refresh rate, it will 'force' pages into lower categories more quickly, forcing NRU to have to evict dirty pages more often.

Thirty memory references happens to be a nice balance between the page faults and the writes to disk. More than thirty will generally leave you with an unideal number of writes to disk, and less than thirty would generally leave you with a higher amount of page faults.

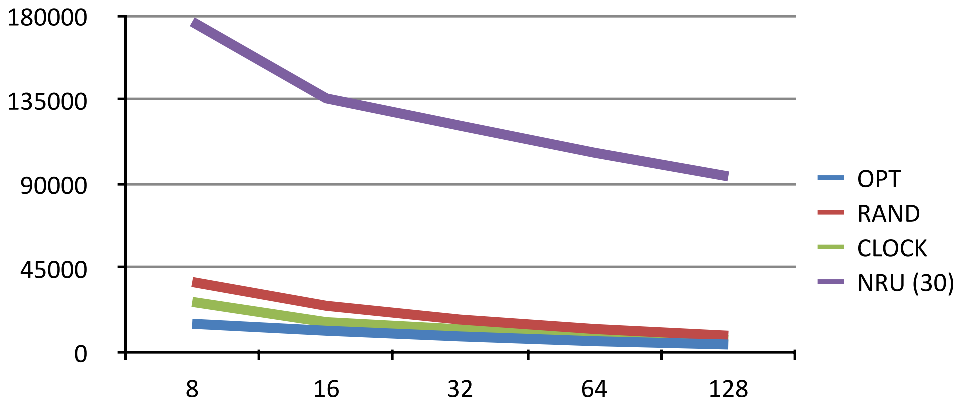
Best Page Eviction Algorithm

If I were to pick a page eviction algorithm for a modern day computer, I would probably go with the clock algorithm. The clock algorithm performs consistently better than any of the other viable algorithms in the area of writes per disk, which is our primary concern as this is the largest bottleneck that computers face during page evictions. Clock doesn't perform supremely well with page faults— it is worse than NRU, but still better than random— but the improvements that it makes in avoiding page faults make it well worth it.

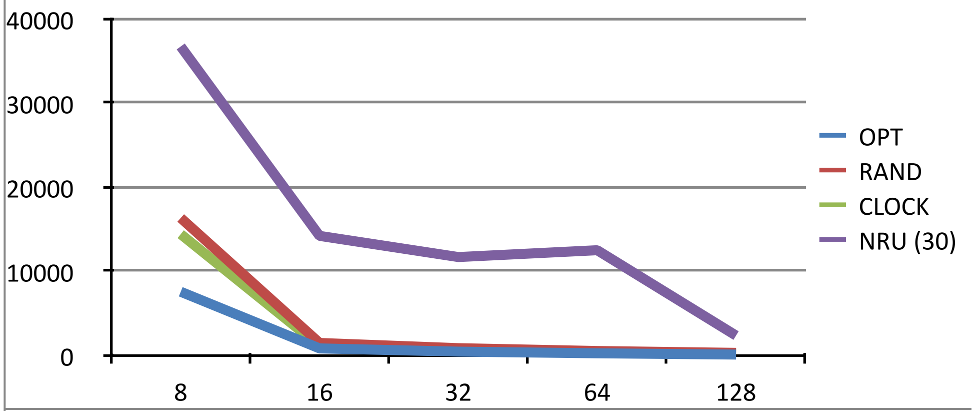
Page Faults - GCC



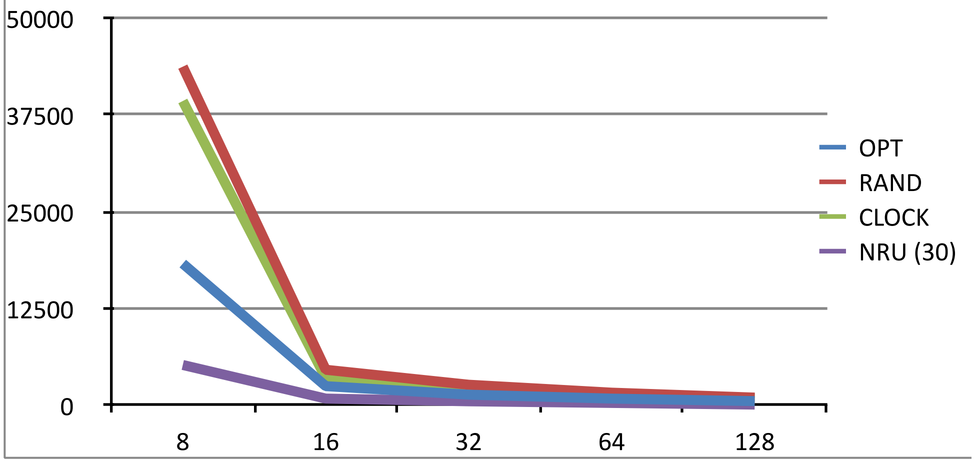
Writes to Disk - GCC



Writes to Disk - BZIP



Page Faults - BZIP



Page Faults - GCC

Frames	OPT	RAND	CLOCK	NRU
8	118480	217324	179557	19151
16	80307	149970	121675	11638
32	55802	106748	86739	7527
64	38050	75074	60664	5031
128	24391	51514	41948	3081

Page Faults - BZIP

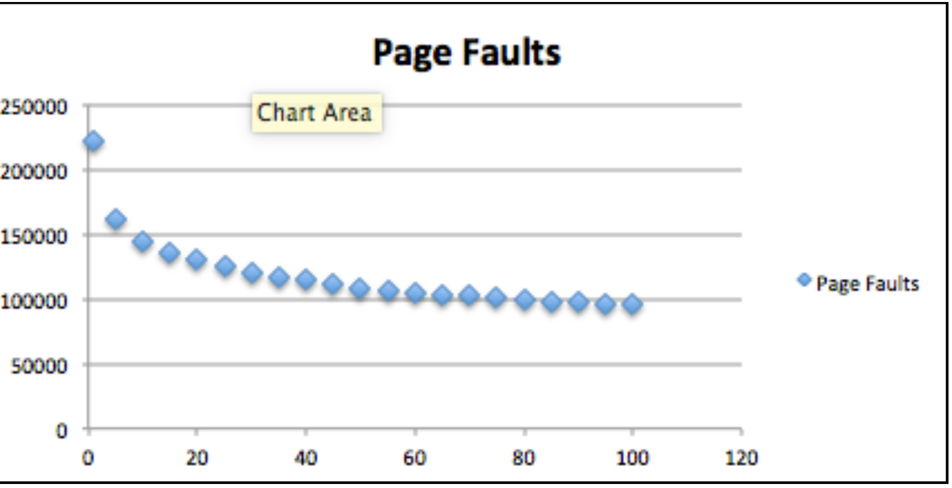
Frames	OPT	RAND	CLOCK	NRU
8	18251	43677	39243	5163
16	2427	4544	3492	824
32	1330	2611	2089	471
64	821	1587	1275	261
128	497	952	795	49

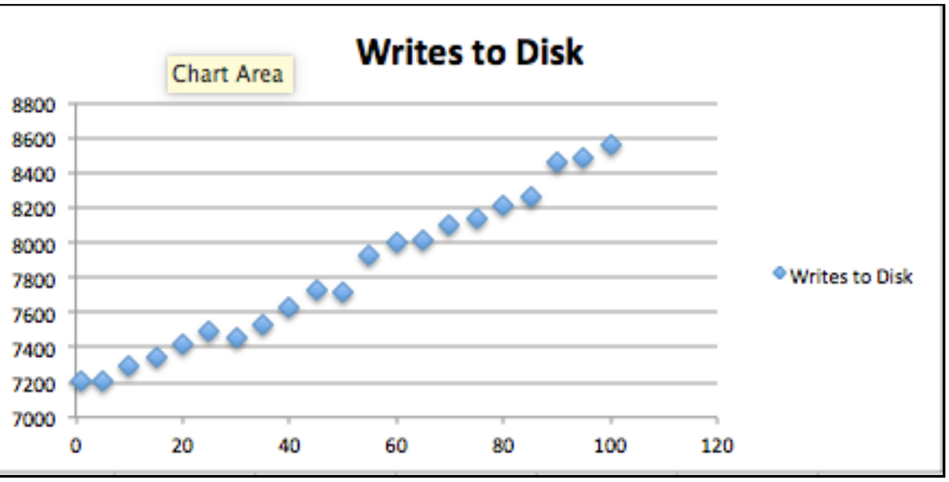
Writes to Disk - GCC

Frames	OPT	RAND	CLOCK	NRU
8	15030	37389	26746	176547
16	11314	149970	15910	135592
32	8266	17281	12066	120952
64	5725	12246	9122	106534
128	3954	8661	6345	93851

Writes to Disk - BZIP

Frames	OPT	RAND	CLOCK	NRU
8	7580	16352	14428	36744
16	845	1492	1128	14225
32	459	873	698	11706
64	264	535	438	12519
128	113	315	237	2371





NRU Refresh Rate Sampling

Refresh Rate	Page Faults	Writes to Disk	Score
1	221247	7207	9419.47
5	160718	7203	8810.18
10	144763	7292	8739.63
15	135808	7349	8707.08
20	129968	7420	8719.68
25	124856	7487	8735.56
30	120487	7452	8656.87
35	117352	7532	8705.52
40	114358	7631	8774.58
45	111489	7729	8843.89
50	108509	7711	8796.09
55	106943	7925	8994.43
60	105306	7995	9048.06
65	103358	8011	9044.58
70	102498	8099	9123.98
75	100712	8141	9148.12
80	99361	8214	9207.61
85	98486	8267	9251.86
90	97516	8454	9429.16
95	96751	8479	9446.51
100	95520	8564	9519.2