Michael Bowen

CS1550 – Tues/Thurs 2:30pm Recitation: Fri 12:00pm

Project 3 Write Up

NRU Refresh Rate Analysis

When going about finding the optimal refresh rate for the NRU algorithm, it quickly became apparent that optimal refresh rate for the given files was also dependent on the number of frames available. In other words, the best refresh rate with eight available frames was vastly different than the rate with 64 available frames. So, I tried to find a refresh rate that worked well with each of the four frame configuration. I used the gcc.trace when testing NRU’s refresh rates. Of the two files we were given, it seemed to be by far the more varied in terms of memory pages used. I felt this would give more representative results.

I began with a coarse approach, increasing the refresh rate by 50 each time and recording the number of page faults. I kept doing so until it became apparent that the number of page faults had began to rise again. Note that due to the NRU algorithm choosing eviction victims at random when multiple candidates exist within the same eviction class, results may be slightly inaccurate. I was trusting that the C random number generator would provide the necessary variablity and forwent taking an average of a large sample size. See the graphs below for the coarse results.

Once I found the refresh rate around which the number of page faults seemed to bottom out, I went through and to a more granular approach, getting page fault data with refresh rates increasing in steps of five for 8 frames and steps of ten for 16, 32, and 64 frames. See graphs below.

After comparing the statistics at each of the frame rate values, the refresh rates that I determined to use were the following:

|  |  |
| --- | --- |
| Frames | Optimal Refresh Rate |
| 8 | 20 |
| 16 | 80 |
| 32 | 200 |
| 64 | 650 |

Page Replacement Algorithm Analysis

For analysis of the four page replacement algorithms there were two results to be considered: the number of memory faults and the number of writes to disk that needed to be performed due to evictions. So I ran each algorithm at 8, 16, 32, and 64 frames, using the refresh rates above for NRU. I started out using those refresh rates for the Aging algorithm as well, but before tracking the actual stats I had been using a placeholder refresh rate of 100 and that rate had given better results overall than NRU’s refresh rates. I tested refresh rates for Aging a bit further (I did not do as complete of an analysis as with NRU) and landed on a refresh rate of 50 for the aging algorithm and I used that rate for each number of frames. Data is listed below, and graph representations follow.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gcc.trace | | | | | | | | |
|  | OPT | | | | Clock | | | |
| Frames | 8 | 16 | 32 | 64 | 8 | 16 | 32 | 64 |
| Faults | 118480 | 80307 | 55802 | 38050 | 181856 | 121682 | 87686 | 61640 |
| Writes | 15031 | 11316 | 8274 | 5730 | 29401 | 16376 | 12293 | 9346 |
| % Worse than OPT Faults | - | - | - | - | 53.49 | 51.52 | 57.14 | 62.00 |
| % Worse than OPT Writes | - | - | - | - | 95.60 | 44.72 | 48.57 | 63.11 |
|  | NRU | | | | Aging | | | |
| Frames | 8 | 16 | 32 | 64 | 8 | 16 | 32 | 64 |
| Faults | 174734 | 123058 | 89037 | 61489 | 228396 | 131684 | 84304 | 60834 |
| Writes | 18428 | 13235 | 10198 | 7406 | 27184 | 15669 | 11680 | 9069 |
| % Worse than OPT Faults | 47.48 | 53.23 | 59.56 | 61.60 | 92.77 | 63.98 | 51.08 | 59.88 |
| % Worse than OPT Writes | 22.60 | 16.96 | 23.25 | 29.25 | 80.85 | 38.47 | 41.17 | 58.27 |
| bzip.trace | | | | | | | | |
|  | OPT | | | | Clock | | | |
| Frames | 8 | 16 | 32 | 64 | 8 | 16 | 32 | 64 |
| Faults | 18251 | 2427 | 1330 | 821 | 46164 | 3468 | 2203 | 1318 |
| Writes | 7580 | 847 | 460 | 283 | 17568 | 1128 | 734 | 443 |
| % Worse than OPT Faults | - | - | - | - | 152.94 | 42.89 | 65.64 | 60.54 |
| % Worse than OPT Writes | - | - | - | - | 131.77 | 33.18 | 59.57 | 56.54 |
| bzip.trace continued | | | | | | | | |
|  | NRU | | | | Aging | | | |
| Frames | 8 | 16 | 32 | 64 | 8 | 16 | 32 | 64 |
| Faults | 43191 | 4236 | 2701 | 1732 | 28695 | 3471 | 2436 | 2037 |
| Writes | 3547 | 884 | 532 | 292 | 11473 | 1091 | 818 | 573 |
| % Worse than OPT Faults | 136.65 | 74.54 | 103.08 | 110.96 | 57.22 | 43.02 | 83.16 | 148.11 |
| % Worse than OPT Writes | -53.21 | 4.37 | 15.65 | 3.18 | 51.36 | 28.81 | 77.83 | 102.47 |

Conclusion

After looking at all the numbers, it is somewhat disheartening just how much worse they all are than OPT. If only we could always have perfect future knowledge! That said, the number of memory faults that occur for each of the algorithms tend to follow the same general trajectory, decreasing when the number of available frames increases – this is as expected. And in terms of memory faults, none of the non-OPT algorithms really stand out as a clear winner.

This leads me to think that the determining metric in choosing an algorithm for use in a real operating system might instead come down to the number of writes to disk that need to be performed. After all, that is the most expensive part of swapping pages in and out. With these files, the clear winner was NRU. In the bzip.trace file, NRU beat OPT in terms of memory writes performed when there were eight available frames and continued to pretty closely mirror OPT’s performance in this metric for the other runs. Based on these statistics, which are admittedly a small sample given we only had two files to work with, and assuming that the writing to disk is the primary source of work that needs to be done, I would choose the NRU algorithm over the other options.