Daniel Kulenkamp
CSE330 Project 3 Milestone
April 13, 2018
Professor Syrotiuk

Project 3 Milestone Submission - Design and Analysis

For the milestone submission, the Least Recently Used (LRU) page replacement algorithm was implemented. Below is a description of the implementation as well as a table including the hit ratios for several cache sizes on the several provided trace files.

The included Makefile generates an executable named lru.

Usage: ./lru cache_size file_name

The implementation was done by using a doubly linked list and a hash map. Both data structures used are provided by the standard c++ library. The list stored page references, and the map stored key value pairs of integers mapped to list iterators which pointed to the node in the list for that page reference. This list was ordered in a most recent to least recent fashion, which was accomplished by only inserting elements into the front of the list, and only removing them from the back of the list. When a page reference was found to be in the list, it was moved from its current position to the most recently used position (the front of the list) and a hit was recorded. If the reference was not in the list, it was added to the first position and a miss was recorded. By using a hash map to keep track of where all of the nodes are in the list, figuring out whether a page was already in the "cache" was accomplished in constant time as opposed to having to search through the entire list. To ensure all values were counted, the uintmax_t type was used, which ensures the largest integer size for any machine.

This implementation of the algorithm was fast. Even with very large files it completed processing in under a minute on general.asu.edu. It also appears to be complete. There were no errors in testing with any of the input files provided (full or partial), nor in any additional files generated for testing. Furthermore, hit rates for OLTP.lis were exactly what were reported in the paper, and hit rates for P3.lis were only a few percentage points different.

Below is a table of hit ratios for various trace files. Values were rounded to the nearest hundreth.

c	OLTP.lis	P3.lis	P6.lis
1024	33.21%	1.05%	0.71%
2048	42.78%	1.15%	0.86%
4096	51.24%	1.32%	1.09%

Table 1: Hit ratios for OLTP.lis, P3.lis, P6.lis for cache sizes $c = 1024, 2048, 4096$