Effectiveness of Page Replacement Algorithms

Taylor Foxhall, William Jagels

April 22, 2016

1 Introduction

We set out to examine the effectiveness of certain page replacement algorithms based on different conditions in the runtimes of several processes running on the computer. Here we test three page replacement algorithms: Most Frequently Used, Least Frequently Used, and a True Least Recently Used implementation.

2 Why it matters and why we asked it.

The demand for memory in many different processes are a critical part of any modern computer. The operating system is in control of managing those pages of memory and part of its job is to determine when to evict a page out of memory when it needs physical space for a process. This page eviction is costly, as it involves moving on the order of thousands of bytes from memory to physical storage. In essence, the more page evictions, or page faults, the more latency in the operating system. This is a critical bottleneck, so an page replacement algorithm should be employed to reduce the total number of page faults. Here test the effectiveness of such page replacement algorithms.

3 How we tested it.

We created two programs, one that generates page references and another that takes the output from that program and simulates page replacement using the three policies we decided to test. We compared the effect of page size and locality wave length on page fault rates for each of the algorithms.

3.1 Locality Phases

After a phase time expires an address can go into any one of the following phases:

- Local
- Open

- Heavy
- I/O

3.1.1 Local

The Local phase causes processes to access only local addresses. All processes start in the Local phase to help overcome cold start costs.

3.1.2 Open

The Open phase will access any page in a processes's local reach so long as it passes a locality test, which is by default has a 50% chance of passing. Otherwise it could pick any address in its address space.

3.1.3 Heavy

The Heavy phase causes a process to access many contiguous parts of memory all at once and much more frequently than it can in the Local or Open phases. This is meant to help test processes that do heavy computations and strain the page table.

3.1.4 I/O

The I/O phase is meant to replicate the behavior of a process that is waiting on I/O device responses. This will cause the process to make local memory address references but very infrequently compared to the Local and Open phases.

4 Results

We graphed the variables we tested against page fault rates to create this 3D curve. The results can be seen below.

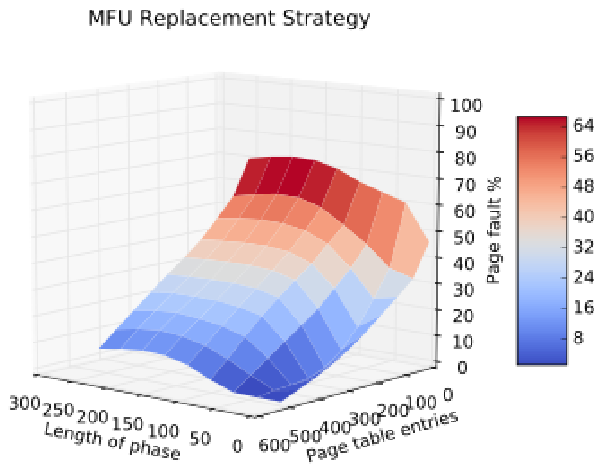

Figure 1: MFU Replacement

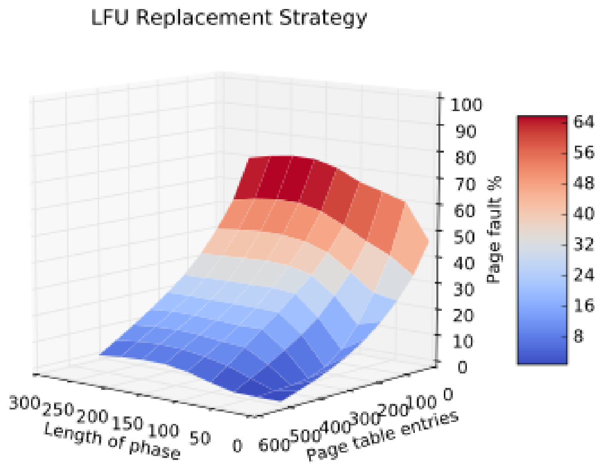

Figure 2: LFU Replacement

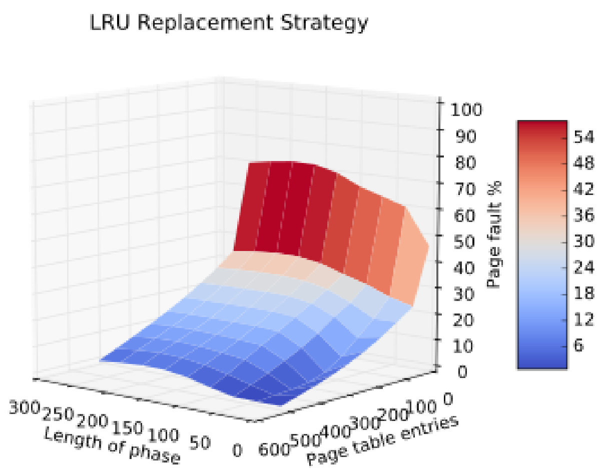

Figure 3: LRU Replacement

5 Conclusion

From our results we can clearly see that the most influential factor in page fault results is the size of the page table. This shouldn't be too surprising, as this controls how much physical memory is available. However it's important to note that we are getting diminishing returns with page table size increase. The other observation we can make is that it appears the longer the phase lengths are the worse each of the algorithms are compared to a shorter phase length. This is surprising, as this seems to imply a more consistent behavior occurs higher page fault rates. This may be because the shorter phase lengths simulate a particular condition over the life of a process, and this causes the replacement algorithms to perform better overall.

One property that is outstanding is that the LRU strategy seems to outperform the other algorithms, both getting better page fault rates in the worst conditions and improving faster than the other as page table size increases. However, we know that a true LRU page replacement algorithm is acutally very expensive for the operating system to maintain, so we suppose an approximation strategy to LRU may be the next step forward.