

Computer Science 161: Operating Systems

Section 8: Virtual Memory, II

CS161 Course Staff
cs161@fas.harvard.edu
<http://www.courses.fas.harvard.edu/~cs161/>
<http://motelab.eecs.harvard.edu/bb/>

April 9, 2007

1 Introduction

Hope you had a relaxing spring break. If you haven't started already, **start coding**.

2 Design Docs

This is the first assignment this year that you're submitting performance analysis. We expect a lot from these. As you've been designing, you've already probably noted many knobs that you could turn to evaluate performance. At the very least, there is the selection of a page eviction algorithm and the constants on the pager daemon works. Choose some knobs to turn, and turn them. We've given you a few workloads (huge, matmult, sort, triplehuge, triplemat, triplesort, parallelvm). You might want to write your own.

It is unacceptable to just give us a bunch of raw data. You should produce tables and/or graphs. For graphing, we recommend `gnuplot` or `Jgraph` (installed on `nice`, but also at <http://www.cs.utk.edu/~plank/plank/jgraph/jgraph.html>). Feel free to submit a document in text, postscript, or PDF. Run every test multiple times (3-5), and check that the standard deviation is within 5% of the mean.

You should always understand your data. Understand why something is behaving the way it is. For this purpose you have the statistics you're measuring on your VM system and `gdb`. Explain, in your performance analysis, what is happening, why something is slow and something else is fast.

It is inevitable that you will find bugs while preparing your analysis. Make sure you're comparing the same kernel on any given head-to-head analysis, but we understand if your kernel has evolved during the process as a whole.

3 Addressing some common questions

- To figure out when a page has been dirtied, set it to read-only in the TLB (using the dirty bit) when it's clean. Then, when you get an appropriate TLB fault, you know that it's going to be written to, so you can mark it dirty and reinsert it into the TLB.

4 Typical Bugs (mostly synchronization)

- Reading the page from swap before it has been completely written
- Using spl synchronization when you are going to disk

- Make sure you're paging in to the right physical address
- Paging in over the kernel will break things; check that you don't do this with assertions.
- Sharing pages inadvertently is bad
- Losing changes made to pages when you swap them out. This might happen if you clean a page using a pager daemon while it is actively mapped in the TLB.
- Are your pages actually zero-filled?
- Have you corrupted the kernel heap?

5 Incremental Approaches

- Even if nothing works, compile often to catch easy bugs and typos.
- Make things workout swapping first—just use lots of RAM.
- Then get "Hello World" to work, and then check `fork()` and `exec()`
- Test bigger programs that dirty pages
- Turn on swap, but run without pager daemon. Test.
- Turn on paging daemon.
- You can make your system run "faster" by increasing the RPM value of your swap disk, which should already be in `/tmp`. Make sure you reset it to the original value when doing performance analysis.