**How to reduce the cost of paging to disk because frequent paging is prohibitive.**

There is a steep cost in reading data from disk as compared to memory (Arpaci-Dusseau & Arpaci-Dusseau, 2012). The following table shows approximate latencies for reading from memory, disk, and faster solid-state drives (SSDs):

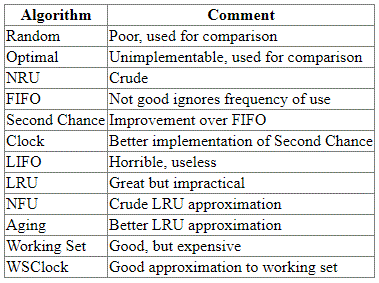
|  |  |
| --- | --- |
| Read 1 MB sequentially from memory | 250,000 nanoseconds |
| Read 1 MB sequentially from SSD | 1,000,000 nanoseconds |
| Read 1 MB sequentially from disk | 20,000,000 nanoseconds |

(adapted from Norvig, 2010; Boner, 2012)

The disparity in cost between accessing memory and accessing disk, or SSD, is only exacerbated by frequent paging, as happens when there is a heavy workload, which increases memory pressure, or insufficient free memory (Arpaci-Dusseau & Arpaci-Dusseau, 2012). Thankfully, there are a number of approaches that can reduce the latency associated with paging to a dedicated swap space on the disk (Arpaci-Dusseau & Arpaci-Dusseau, 2012). Listed here are just a few methods that can improve system performance:

As we learned last week, we can introduce the use of a Translation-Lookaside Buffer (TLB). The TLB is a hardware cache that holds page translations of frequently used pages so that they can be accessed more quickly than they would if the translation information were only held in page tables (Arpaci-Dusseau & Arpaci-Dusseau, 2012).

Another approach discussed by Hasan, et al. (2014) and Gottlieb (2009) is to choose which pages to keep in memory verses the ones we swap out to disk through the use of page replacement algorithms. With the right pages kept in memory, the number of accesses to disk will be reduced, thereby increasing performance. Here is a brief, but evaluative summary of various page replacement algorithms (Gottlieb, 2009):

.

The next method to reduce paging costs is Prefetching, where the OS tries to “predict when pages will be needed and load them ahead of time to avoid page faults” (Ousterhout, 2014). This is an effective technique if the program is accessing memory in a sequential fashion; whereas there is no benefit when programs access data in unpredictable ways (Ousterhout, 2014; Arpaci-Dusseau & Arpaci-Dusseau, 2012).

A more drastic approach to reduce paging is to control the number of processes allowed to access memory -- by not running them (Arpaci-Dusseau & Arpaci-Dusseau, 2012).

All of the aforementioned approaches can help to alleviate the cost of paging frequently to disk; However, the single most effective solution is to simply invest in more memory so that memory pressure is reduced and there is less need to page out virtual space in the first place (Arpaci-Dusseau & Arpaci-Dusseau, 2012).

References

Arpaci-Dusseau, R. & Arpaci-Dusseau, A. (2012). Operating Systems: Three Easy Pieces. Madison, WI: University of Wisconsin-Madison. Retrieved from <http://pages.cs.wisc.edu/~remzi/OSTEP//>

Boner, J. (2012). Latency.txt. *GitHubGist.* Retrieved from <https://gist.github.com/jboner/2841832>

Gottlieb, A. (2009). V22.0202 Operating Systems. *New York University*. Retrieved from <https://cs.nyu.edu/courses/spring09/V22.0202-002/class-notes.html>

Hasan, M. H. O., Munam, A. S., Abuelgasim, I.M., & Manzoor, I.T. (2014). A Comparison of Page Replacement Algorithms in Linux Memory Management. *International Journal of Computer and Information Technology*, 3(3), 565-560. Retrieved from <https://www.techrepublic.com/resource-library/whitepapers/a-comparison-of-page-replacement-algorithms-in-linux-memory-management/post/?regId=&asset=32990111&skipAutoLoad=1>

Norvig, P. (2010). Teach yourself Programming in Ten Years. Norvig.com. Retrieved from <http://norvig.com/21-days.html#answers>

Ousterhout, J. (2014). Demand Paging. *Stanford University.* Retrieved from <https://web.stanford.edu/~ouster/cgi-bin/cs140-spring14/lecture.php?topic=paging>