<u>Masters Thesis Proposal</u>
Jonathan Ledlie
October 10, 2000

Cacheable web files embody certain unique properties which can and should enable much higher performance than when they are stored in a standard UNIX file system. While various refinements have shown elegant methods to keep frequently accessed files in memory, prior studies portray that approximately 90% of all hits come from disk. In memory speedups can only go so far. Two previous studies have established that arranging files by size and domain name, again within a standard file system, can improve throughput by as much as 25 times over a simple directory hierarchy. These systems leverage the unchanging size and lack of ownership inherent in cacheable web files. One system further improves performance by buffering writes to make for more sequential access, often at the expense of reads.

We argue for two major changes in prior approaches. First, the fact that these files by definition already have a backup enables us to loosen the strict consistency semantics that a standard file system enforces. For example, we can keep our meta-data in memory and then checkpoint based on how much we are willing to lose with a crash, achieving a balance between robustness and speed. Second, we dispute that web caches should equate performance with operations per second. Fast writes are meaningless if user-perceived latency is very high. Those focused on throughput have argued: "our policies interact with disks that operate in the millisecond range, while the typical world-wide web latency is in the second range." This logic would seem to imply that virtually any work on disk I/O for web caches is dwarfed by the network cost. For caches relatively near the client, this is certainly not true and can take more than 30% of the overall transfer. Again, we argue for a balance between long streams of writes, which are not part of the user's immediate perception, and reads, which are.

To evaluate these two hypotheses – that in-memory meta-data improves performance without damaging robustness and that a balance can be achieved between latency and throughput – we will develop five flavors of caches.

1. Ext2, one URL per file
2. Ext2, buddies (keeping web files of the same size in the same local files)
3. Ext2, buddies with locality (mapping the same domain name to the same local directory)
4. Ext2, buddies-locality-meta (keep all meta-data in memory, checkpointing occasionally)
5. Raw disk, buddies-locality-meta, variable length write queue.

All caches will be multithreaded and will follow a pre-forking model. We feel that shifting to the raw disk here is acceptable because it allows us to more explicit with when disk access actually occurs and because this type of access is available on most UNIXes, including Linux.

We will evaluate the efficacy of each type of cache with the Web Polygraph benchmarking tool. This will facilitate a straightforward comparison among these five caches and, to some extent, with others (their test results are on other hardware). The Polygraph provides a synthetic client and server and outputs throughput and latency and other benchmarks.

Bibliography

1. Technical Report 111, Digital Equipment Corp. Systems Research Center. Blackwell, T., Harris, J., and Seltzer, M. (1995). *Heuristic Cleaning Algorithms in Log-Structured File Systems*. In Proceedings of the Winter 1995 USENIX Conference.

2. Ledlie, Jonathan and McCormick, Matthew. *A Fast File System for Caching Web Objects*. Working Paper, May 2000.

3. Li Fan, Pei Cao, Jussara Almeida, and Andrei Broder. *Summary Cache: A scalable wide-area Web cache sharing protocol*. In Proceedings of ACM SIGCOMM98, September 1998.

4. C. Maltzahn, K. Richardson, and D. Grunwald. *Reducing the Disk I/O of Web Proxy Server Caches*. In Proc. of the 1999 Usenix Technical Conference, 1999.

5. Evangelos P. Markatos, Manolis G.H. Katevenis, Dionisis Pnevmatikatos, and Michail Flouris. *Secondary Storage Management for Web Proxies*. In Proc. 2nd USENIX Symp. on Internet Technologies and Systems, pages 93-104. Boulder, CO, October, 1999.

6. M. Rosenblum and J. Ousterhout. *The design and implementation of a log-structured file system*. Proceedings of the 13th Symposium on Operating System Principles, pages 1--15, October 1991.

7. Alex Rousskov and Valery Soloviev. *A Performance Study of the Squid Proxy on HTTP/1.0*. World-Wide Web Journal, Special Edition on WWW Characterization and Performance Evaluation, 1999.