CSC2231 – Internet Systems and Services

Paper Review #3 – Locality Aware Request Distribution in Cluster-based Network Servers
Name: Alex Wun
Date: Sep. 25, 05

Two-tier HTTP server clusters consist of a set of front-end servers receiving incoming requests that are passed off to back-end servers, which actually serve the requests. Traditional state-of-the-art algorithms are based on Weighted Round Robin (WRR) distribution of requests to back-end servers. This provides good load balancing, but lacks performance when the total working set of requests causes frequent replacements in the caches of the back-end servers. Vivek et al. propose an algorithm to "hash" requests to specific back-end servers based on the request's contents. This method logically partitions the back-end servers according to request content, thereby improving server affinity. Additionally, the algorithm also redistributes requests between back-end servers in order to maintain a balanced load (based on the number of active TCP connections per server).

LARD has several advantages over traditional WRR request distribution methods. First, by taking advantage of locality in HTTP requests, LARD increases the cluster's performance by increasing the number of successful cache hits. Second, by partitioning the back-end servers according to request content, server affinity is increased allowing more scalable secondary storage. Third, due to the use of content-based distribution, LARD allows for the possibility of employing different specialized servers for different requests. Finally, this algorithm increases the total effective cache size from the front-end server's point of view. Since each back-end server's cache is less likely to be cleared and the front-end coordinates the use of each back-end server, the effective cache size of the back-end cluster is potentially the sum total of each individual back-end server's cache.

However, LARD has two drawbacks. First, the mapping between each "type" of request and its corresponding back-end server must be maintained. And second, every connection between the client and back-end server must be managed by the front-end servers in a "hand-off" process. Due to the large number of different possible requests, the amount of additional overhead caused by maintaining a hash and constantly managing connection hand-offs is significant. But more importantly, their implementation requires customization of the TCP stack in both the front and back-end servers. This limits the portability of LARD in addition to increasing the coupling between front and back-end servers.

Overall, this is an intuitive and effective approach to improving the performance of HTTP server clusters. If carefully implemented (as Vivek et al. have done), the benefits of server affinity for improved caching outweigh the overhead of hash maintenance and constant connection hand-offs on the front-end servers. In fact, although my hands-on experience with TCP is limited, I strongly suspect that it is possible to implement the necessary connection hand-offs through other means and avoid manipulating the TCP stacks.