James Maciej Krywin

Multicore Programming – Lab 4 Write Up

The presented graph shows the average and median thread times for a server that uses only disk writes compared against a server with an added key value store that acts as a cache. The servers were hosted on crunchy1 and both contained 64 threads. Both were tested against 1, 4, 8, 16, 32, 64, 128, 256, 512 sessions consisting of 100 requests each. As seen on the graph, the server with the implemented cache key-value store outperforms the plain server. However, this is only with the recent addition of fsync() to the server writes. Initially, the server with the cache performed comparably to its counterpart due to the fact that fclose() of a File Stream does not guarantee an immediate write to disk. Thus, the writes were happening to memory, which in turn do not have the time penalty associated with a true disk I/O.

In the presented graph we notice a time disparity of up 1 millisecond, which translates to a total of 33% speed-up for the cache-implemented server. This could be the result of my implementation of locks. Instead of relying on mutex locks for each file, my system relies on the creation of “.rdlock” and “.wrlock” files on the drive. Their presence indicates that the associated key is currently being read or written to. Thus, any conflicting actions such as a write & read or write & write cannot occur. The added I/O operations associated with this extra file creation contribute significantly to the already penalized thread times of the regular server.