Tyler Kim

P2 Exp2b

There are a couple of things that were done to the hashtable-biglock.c skeleton file such that it functions correctly and scales well. The first implementation was to create an H(k) function that automatically takes a key to find which hashtable the key-value pair must go to using the equation, . In the void \*thread\_func(void \*thread\_id) function, I called find\_which\_hashtable(int key), which gets the H(k) value for a given key, to extract the correct hash table index. I changed the arguments in the locking and unlocking mutex so that only the mutex that corresponds to the correct hash table index locks or unlocks. In addition, the key-value pairs are inserted into the correct hash table. Finally, in the main(int argc, char \*\*argv)function, numHashTables was set to 100 \* numThreads to reduce contention for hash table resources.

In terms of the methods described in exp2a, it seems as if they were already implemented in some form or another in the skeleton code. The first method which created multiple locks for multiple hash tables was implemented in the thread\_func(void \*thread\_id) function. Pooling resources were already implemented by making a single array for the keys and another for values. Evening out work distribution was already implemented using the the\_config.numParts variable.

A graph with a red line

Description automatically generated