lab3 report

To run the program, use ./lab3 -n threadnumber -v
If you choose -v, the program will use ThreadSafeKVStore with a global mutex implementation.

with read/write lock

Threads	mean (us)	max (us)	min (us)	median (us)
1	379	5474	95	185
2	439	6046	103	208
4	479	6090	104	237
5	509	6449	106	236
8	474	5750	100	214
12	479	6280	103	215

with global lock

Threads	mean (us)	max (us)	min (us)	median (us)
1	386	5603	100	189
2	440	5536	98	206
4	480	5762	98	209
5	490	5957	106	219
8	526	5985	103	234
12	480	5814	105	224

Each time there arrives a request, the complete time of request will be pushed into a thread safe list, and the list will be locked and perform a sort to judge the mean, median, min and max. The output files will be overwrite every time a request finishes. The mean, median min and max view from client can be viewed from httperf output directly.
Here I choose 1 session per 5000 requests. From the ouput files, we can see there is not much difference between an implementation of global mutex and read write lock. I guess the major impact coms from the threadsafe queue. The queue implements with a global mutex and if we can use a lock free concurrent queue instead, the difference caused by KVstore could be more clear.
Another interesting phenomenon is that when compared stats_with_4_threads and 5,8 with rw lock, the mean time of 5 threads is largest. While we suppose the mean time should decrease as there are more threads, the time

cost of context switch should be taken into considered. A trade off bewteen the speed up from concurrent and cost from context switch should be made, rather than choose as many threads as we can. It remindes me that large time cost jobs might perform better than now.
What's more, we can see the mean value is always much larger than median, because the max time of a request influence the mean value much more than influence the median. The max time of a request might be the contention of the queue (wait in queue for long time). We should consider to improve the queue efficiency, using CAS might be a good way.