Performance Evaluation Report

Team #: CD-074
Course Instructor: Ted Nolan
Shubham Manchanda - 999812225
Harsh Verma - 998734482
Mandheer Khabra - 999865374
Date Submitted: February 16, 2014

Methodology

To get a clear picture of the efficiency and transaction speed of our storage database we performed tests under three configurations:

- logging to file: writing to the logger file
- logging to stdout: printing to the screen
- logging disabled

The performance tests included measuring the following:

- Server processing time: time taken by the server to complete an operation such as inserting, modifying, deleting or retrieving a record.
- End-to-end time: time taken by the entire client-server network system to process a set of operations issued by the user and returning the output.

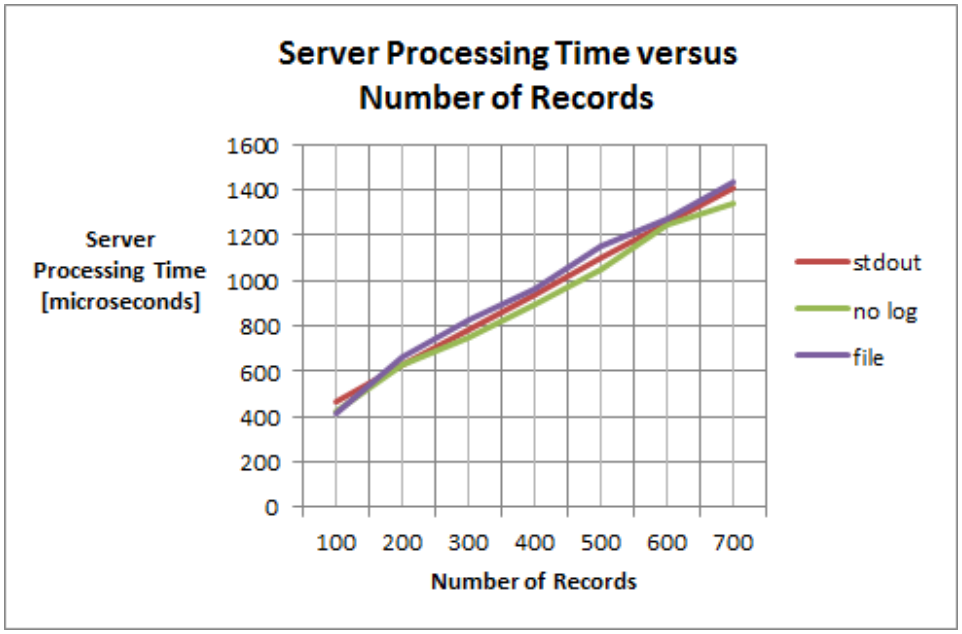
To minimize inconsistent and inaccurate results, we automated the performance testing process. In order to measure the server processing time, we created a function on the server side, which loaded the census data into our database by iterating through the key-value pairs in the input census file and calling the insert function from our hash table library.

In a similar manner to the server processing time evaluation, we obtained the timestamps before and after calling a function in our client shell, which loaded/read the workload data(census) by calling the set/get function from the client library.

For both of the above evaluation tests, we varied the the number of records to get analyze the relationship of the processing times with the number of records in our hash table.

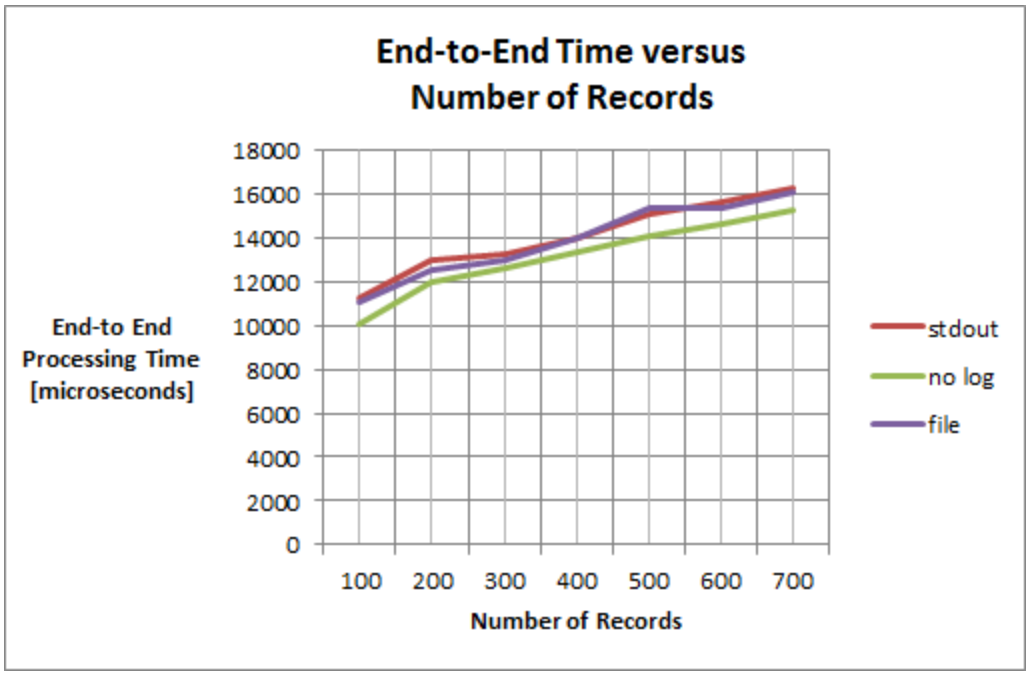
Results

Graph 1:



End-to-End Time [microseconds] versus Number of Records			
Number of Records	stdout	no log	file
100	11265	10053	11068
200	13006	11956	12539
300	13206	12623	12973
400	13996	13356	14005
500	15100	14092	15325
600	15639	14602	15350
700	16212	15256	16053

Table 1:



Graph 2:

Analysis

As illustrated in the graphs above, the processing time increases for both the server and the end-to end processing time as the number of records increases.

In regards to Graph 1, it is observable that the processing times for testing under all three configurations does not greatly vary. This may have been because the workload was loaded into the hash table as soon as the configuration parameters were authenticated, demonstrating that a very limited number of logger calls were made. As expected, Graph 2 shows that end-to-end time is greater than server processing time and follows a similar trend as seen in Graph 1. However, the end-to-end time for the configuration when logging is disabled is slightly attenuated, which may be due to the smaller number of operations in between the two time checkpoints. Although the processing time is lower for the configuration when logging is disabled, the

difference is not substantial considering that it is in microseconds. We should also take into consideration that a logger file allows us to trace the path and narrow down the breaking point if the storage system crashes.

Putting our analysis into perspective, the program is 96.6% quicker than the objective goal of having a “fast” storage system. This indicates that a great amount of speed may still be sacrificed in order to continue to achieve this objective goal, allowing for the improvement of our design in other aspects. Therefore and because there is a restriction on the size of a table, we must work towards developing an even better hash function that would improve the record mapping process.