Team CD-101 (Heindrik Bernabe and Matthew Marji)

Performance Evaluation Report

Methodology:

Note: For all our test cases we used 250 cities from the database of the 2006 Canadian Census by statistics Canada.

Average End-to-End Delay

Our team evaluated the average end-to-end delay with concurrency mechanisms' 0 (Multi-threading disabled) and 1 (POSIX Threads enabled). We measured the time (in seconds) that it would take for the client to set 250 records with and without concurrency. In addition, we also measured the average end-to-end time for multiple clients to get 250 records. We repeated this test with 1 through 10 clients performing 250 the get function simultaneously.

Transaction Abort Rate

To evaluate the abort rates our team performed a set, delete and set again on the same variable. This allowed us to force an abort within the server. We tested with multiple clients simultaneously running using POSIX Threads. To count the amount of aborts that occurred we had a global variable that incremented every time an abort is called. We tested this with 1 through 10 clients performing the same task simultaneously.

Distributed evaluation

This is similar to the Average End-to-End delay test. The only difference we tested up to 3 clients and each of the clients was connected on different machines. In addition, this test was conducted with POSIX Threads (Concurrency 1) enabled.

Results:

\

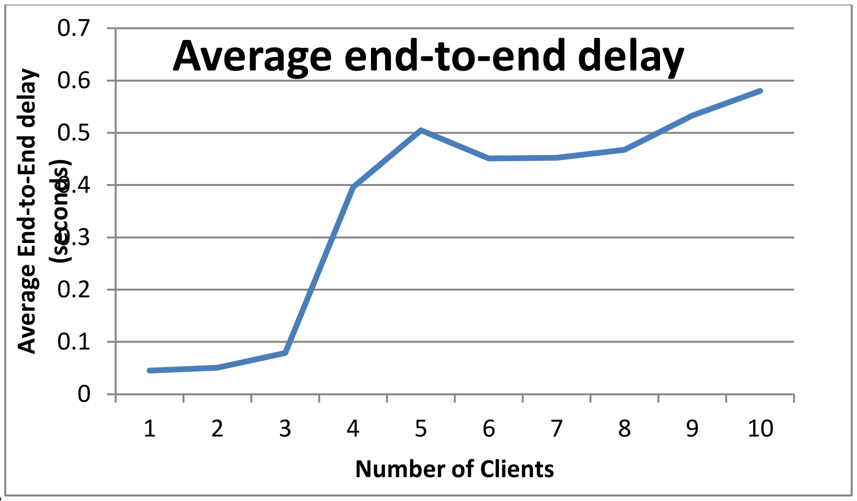

Figure 1: A graph of the Average End-to-End time

Team CD-101 (Heindrik Bernabe and Matthew Marji)

Number of Clients Running	1	2	3	4	5	6	7	8	9	10
End-to-End Time per Run	0.04621	0.050154	0.054424	0.842119	0.600822	0.52457	0.873023	0.796902	0.844034	0.888603
	0.04321	0.050896	0.129911	0.211246	0.712896	0.572667	0.216643	0.276212	0.485576	0.468381
	0.045575	0.049941	0.051414	0.134528	0.200123	0.256031	0.267031	0.329898	0.269482	0.383974

Table 1: Table of the results of the End-to-End times versus number of Clients Running

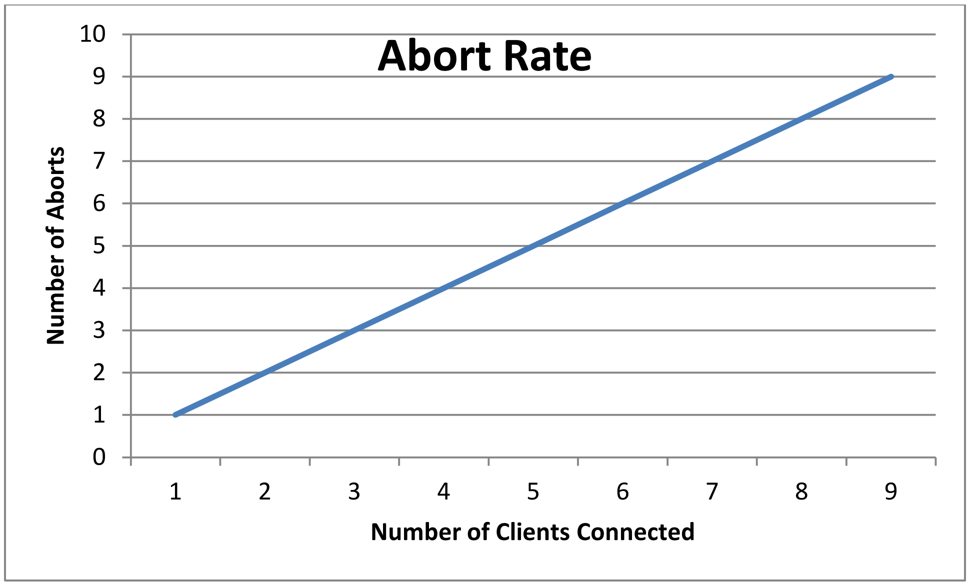

Figure 2: A Graph showing the results Abort rate

Number of Clients	1	2	3	4	5	6	7	8	9	10
Number of Aborts	0	1	2	3	4	5	6	7	8	9

Table 2: Shows the results of the abort rate versus the number of clients running.

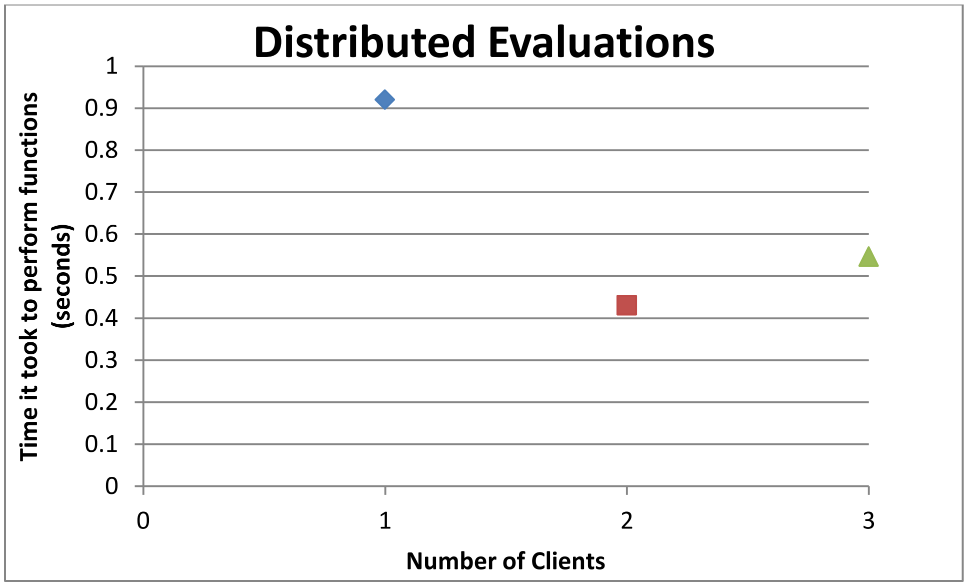

Figure 3: A graph showing the results of the Distributed Evaluation

NOTE: Although not depicted on the graph there is an error bar of around 0.2 seconds.

Number of Clients Running	Try 1	Try 2	Try 3
1	0.919677	0.486299	0.478903
2	0.430729	0.480478	0.878903
3	0.546787	0.646787	0.543893

Table 3: Show the timing results of each run versus the number of clients running

Analysis:

In Conclusion, our team is very happy with the results as the server is able to run multiple clients with no real significant loss in performance. If we look at the average end-to-end time performance we see the graph beginning to settle and level off at about 5 clients. This is good as there will be minimal performance loss if several clients are simultaneously performing functions on the server. Furthermore, the difference in performance between the Distributed Evaluation versus the Average End-to-End results are so close that we are confident that our Server would perform effectively in a real life scenario. Finally there is a major issue with the Abort Rate as the number of aborts increases with the number of clients connected. Fortunately, there is a fix to this issue by updating the current value before setting the new value when there are version mismatches within a transaction.