18-749: Fault-Tolerant Distributed Systems

Phase III
Experiments
Team 1

Saul Jaspan
Lucia de Lascurain
Yudi Nagata
Christopher Nelson
Luis Rios

Apr 17th, 2006

Lessons Learned from the Experiments

Error identification takes time

We ran into situations where the experiments had been running for hours just to realize at the end that we were having problems with the experiments. We detected several errors in the scripts, and in the system that cost us a considerable amount of time and effort. Most of this time elapsed between the start of the experiments and the identification of the problem.

Experimentation resulted in a good test tool

The experiments resulted in a very good tool for testing our system. One of the obvious benefits was that the system was tested under new conditions as for example the simultaneous interaction of several clients. We detected a concurrency issue causing more than one client to have the same client id and thus making the results of the experiments useless.

Order of method calls matter

Given the restrictions on the environment on which the experiments were executed, it was important to measure methods that could have a similar latency each time they were invoked. As an example of a bad method sequence we could cite our createGame method which creates a new record in the database and then called our listGames method which returns the existing games in the database. As the number of games grew in the database, the execution of listGames turned into a huge overhead for the experiments.

Be proactive in checking results

Just waiting for the experiments to finish may be counterproductive. If we had waited for the experiments to be done we would have spent much more time. We were able to have the experimentation done on time in part because we realized at a good time that our experimentation scripts were buggy, and that our system was not ready for the experimentation. So, whenever the results were contradictory or made no sense we attacked the problem as soon as possible. Otherwise we would have wasted some precious time.

System Findings

Performance

Our system appears to scale well in regards to reply size. Within sets of experiments where the number of clients was held constant, we did not see much (if any) change in latency with an increase in reply message size. However, our system does not scale quite as well with an increase of clients. Compared to our one client case, our four, seven, and ten client experiments had respective latency increases of approximately 100%, 500%, and 700%.

Dependability

During the multiple executions of our experiments (we had to restart our experiments a couple of times), our application only failed to provide service due to situations that were outside of our control. One service outage was caused by a disk quota over-run caused by the multiple teams collecting data in a shared volume in the class' AFS space. A second service outage was caused by a kerberos ticket that expired while our experiments were running and kept our shell scripts from being able to ssh into other machines in the clusters.

Additionally, our system showed a fairly constant maximum latency (when taking away the outliers) regardless of the changes in the number of clients, reply size, or inter-request time. However, the outliers showed some large peaks in maximum latency – on the order of 2000% increase in latency compared to the mean latency.

Robustness

While our system did show some large spikes in latency, the mean latency throughout all the experiments stayed relatively constant. Additionally, our runs with ten clients and no inter-request time showed a max latency approximately equivalent to the max latency of the runs with one, four, and seven clients with no inter-request time.

Hypothesis of non-expected results

We had many more outliers (using the 3 sigma approach) than we expected. It also appears that the majority of the latency for outliers comes from the server, rather than the middleware and network, and the majority of the time server-side is likely the time to access the database. We think the number of outliers and their component breakdown show the effects of multiple teams demanding too much from the same mysql installation at the same time.

Experiments for verifying hypothesis

To verify our hypothesis, we would need to add a couple more probes to our server to determine what portion of the server-side time is logic versus database access. We could also re-run the experiments against an installation of mysql that is only available to our team to see if we get similar outliers in quantity, component breakdown, and magnitude.

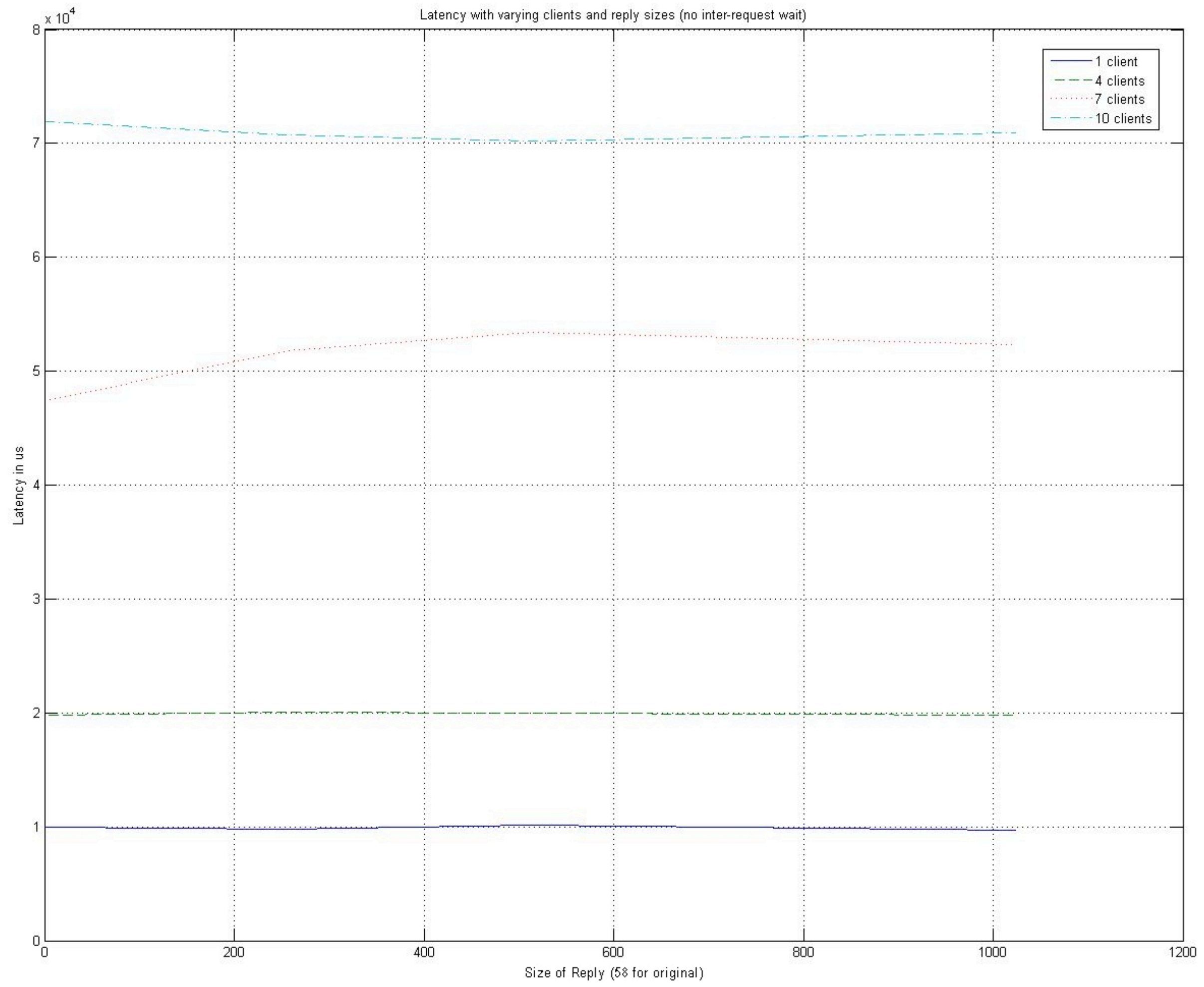
Magical 1% theory
When using the 3 sigma approach to outliers, we marked approximately 190,000 data points as outliers. This is approximately 7% of our 2,640,000 data points. After we remove the 1% of the data points that were the furthest from the mean, the max value line follows the same curve as the mean value line, but it is approximately .2 to .3 seconds higher on the y-axis (latency).

Graphs
Original images available at:
http://www.ece.cmu.edu/~ece749/teams-06/team1/html/FaultTolerantApp/ft_baseline_eval_data/graphs/

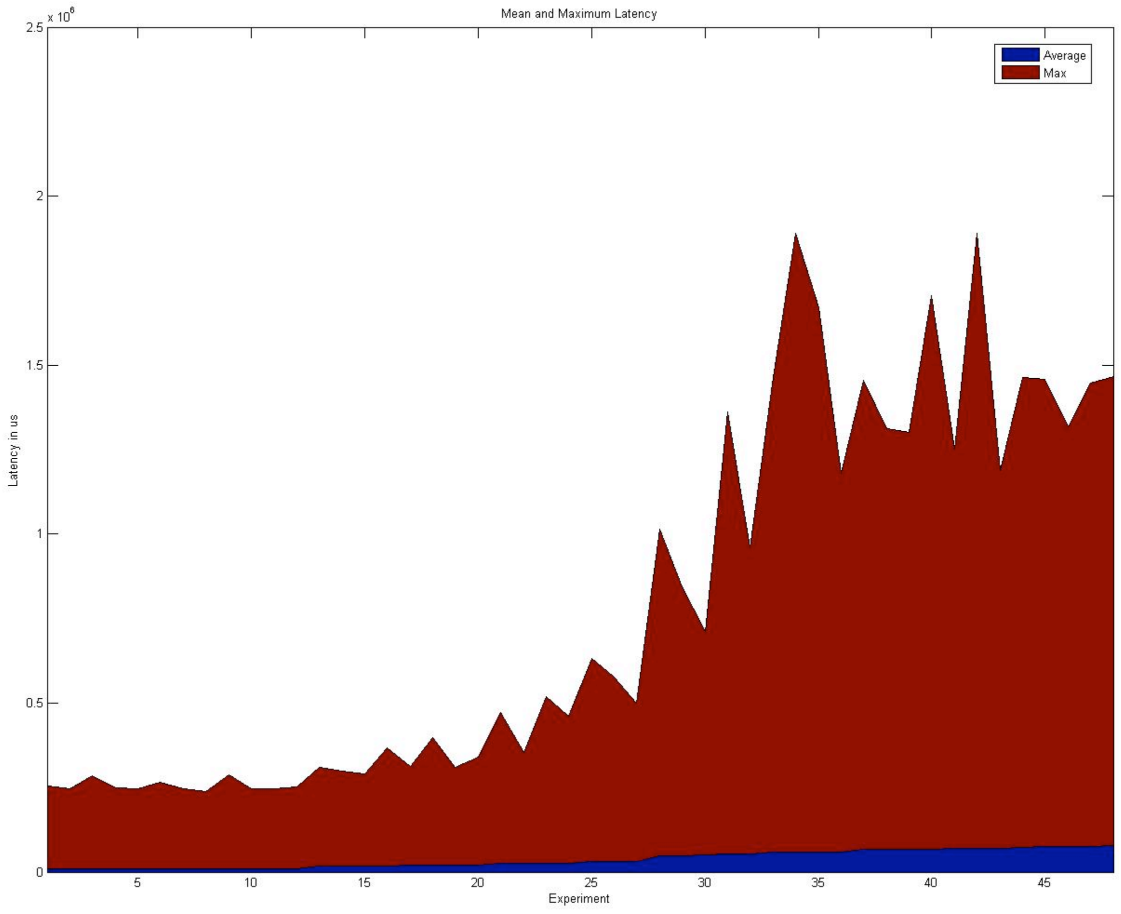
Experimentation numbering in graphs

Experiment No.	Number of Clients	Reply size	Inter request time
1	1	58	0
2	1	256	0
3	1	512	0
4	1	1024	0
5	4	58	0
6	4	256	0
7	4	512	0
8	4	1024	0
9	7	58	0
10	7	256	0
11	7	512	0
12	7	1024	0
13	10	58	0
14	10	256	0
15	10	512	0
16	10	1024	0
17	1	58	20000
18	1	256	20000
19	1	512	20000
20	1	1024	20000
21	4	58	20000
22	4	256	20000
23	4	512	20000
24	4	1024	20000
25	7	58	20000
26	7	256	20000
27	7	512	20000
28	7	1024	20000
29	10	58	20000
30	10	256	20000
31	10	512	20000
32	10	1024	20000
33	1	58	40000
34	1	256	40000
35	1	512	40000
36	1	1024	40000
37	4	58	40000
38	4	256	40000
39	4	512	40000
40	4	1024	40000
41	7	58	40000
42	7	256	40000
43	7	512	40000
44	7	1024	40000
45	10	58	40000
46	10	256	40000
47	10	512	40000
48	10	1024	40000

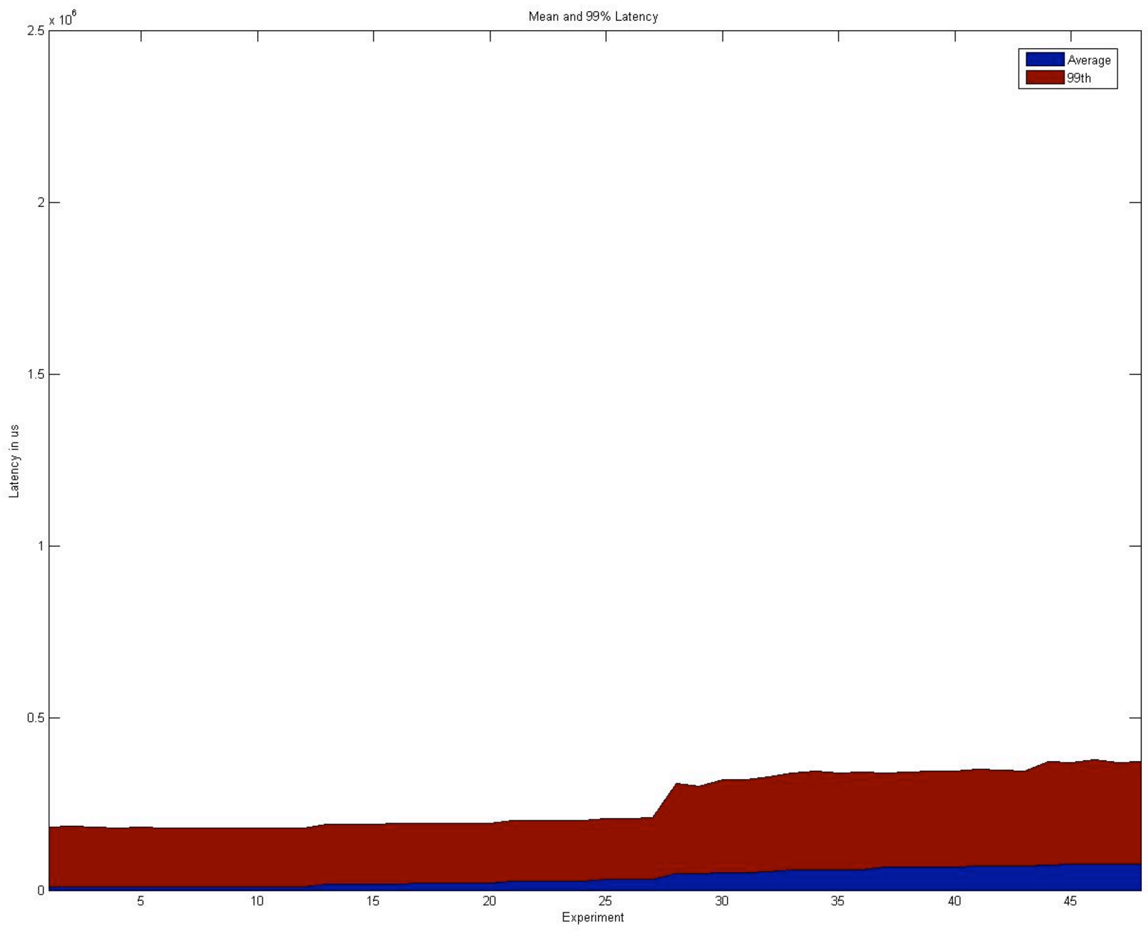
Line plots of latency for increasing number of clients and different reply sizes (no pause)



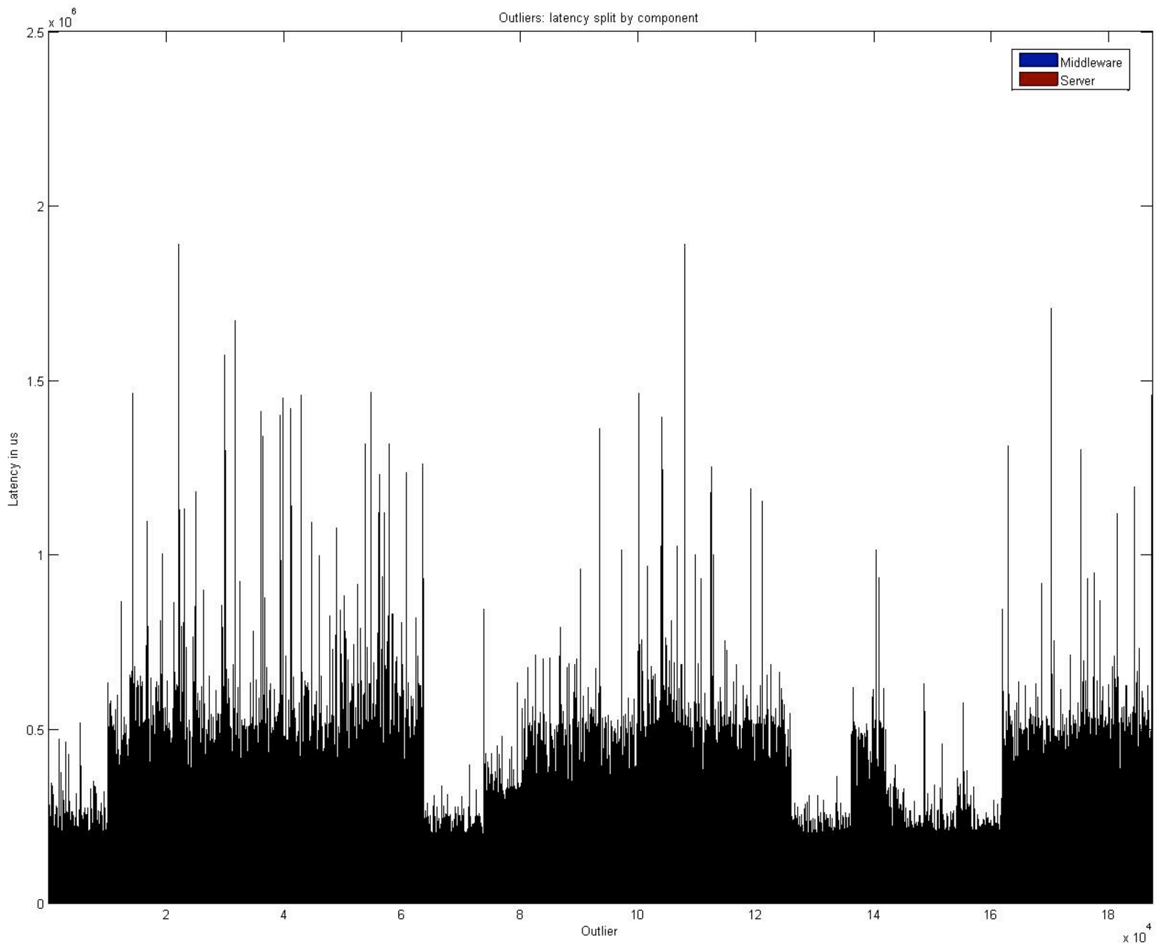
Area plot of (mean,max) latency



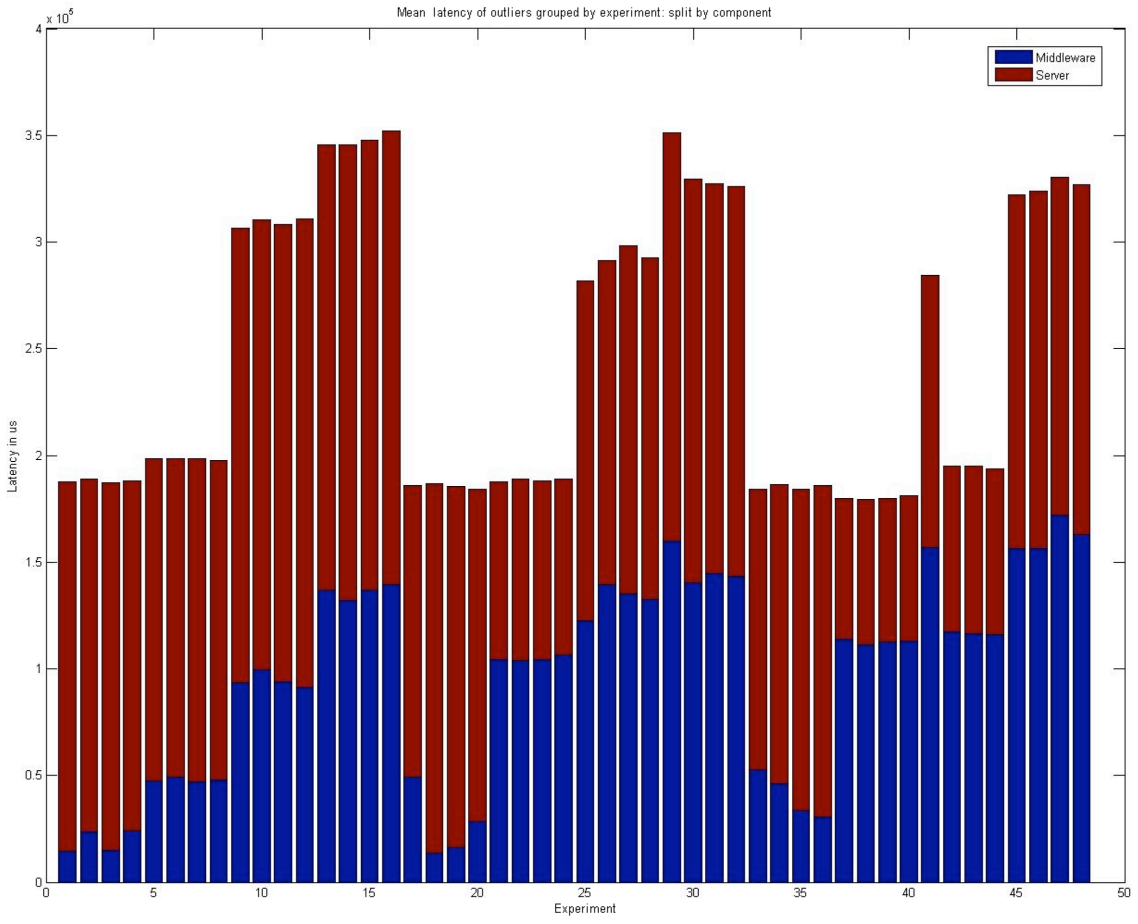
Area plot of (mean,99%) latency



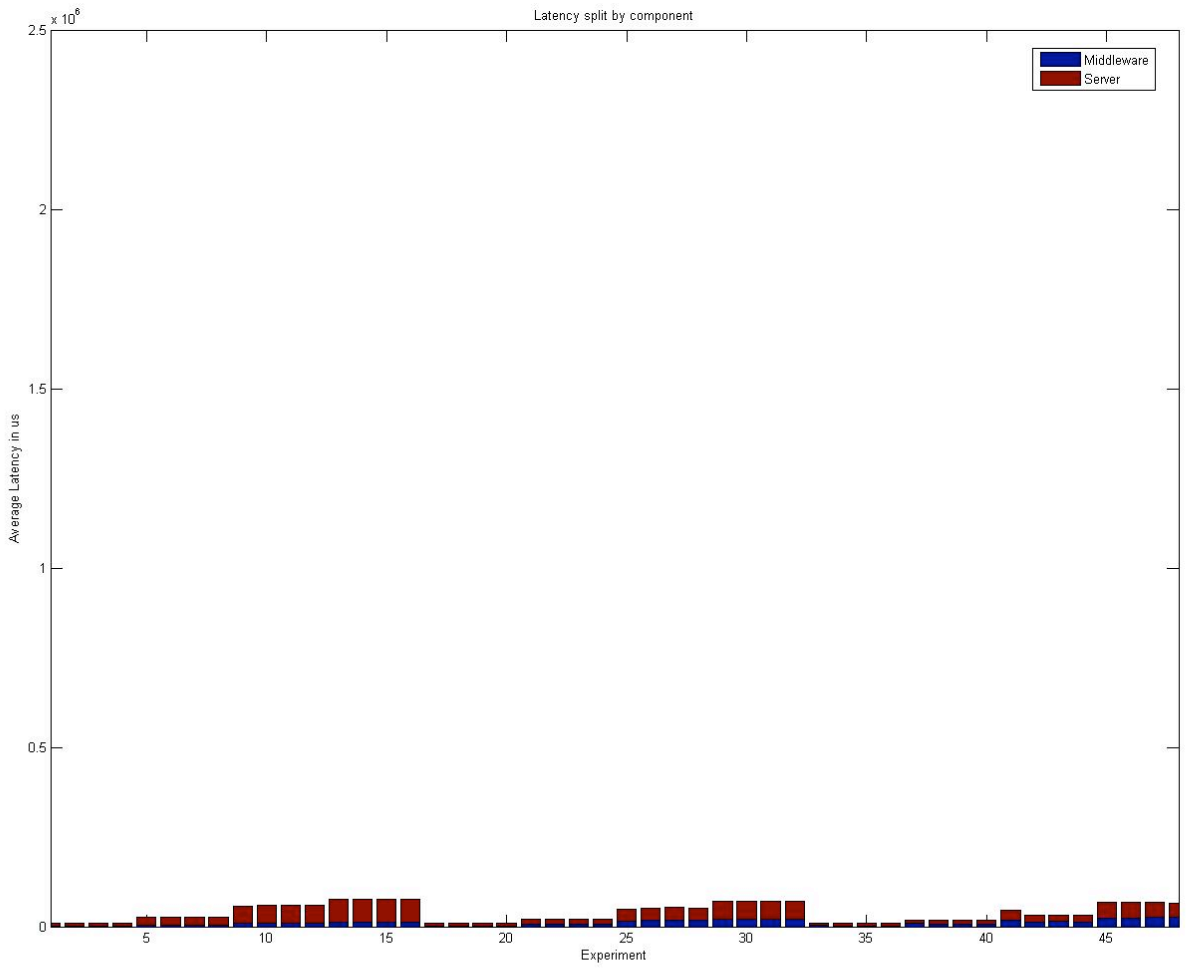
Latency component breakdown for outliers



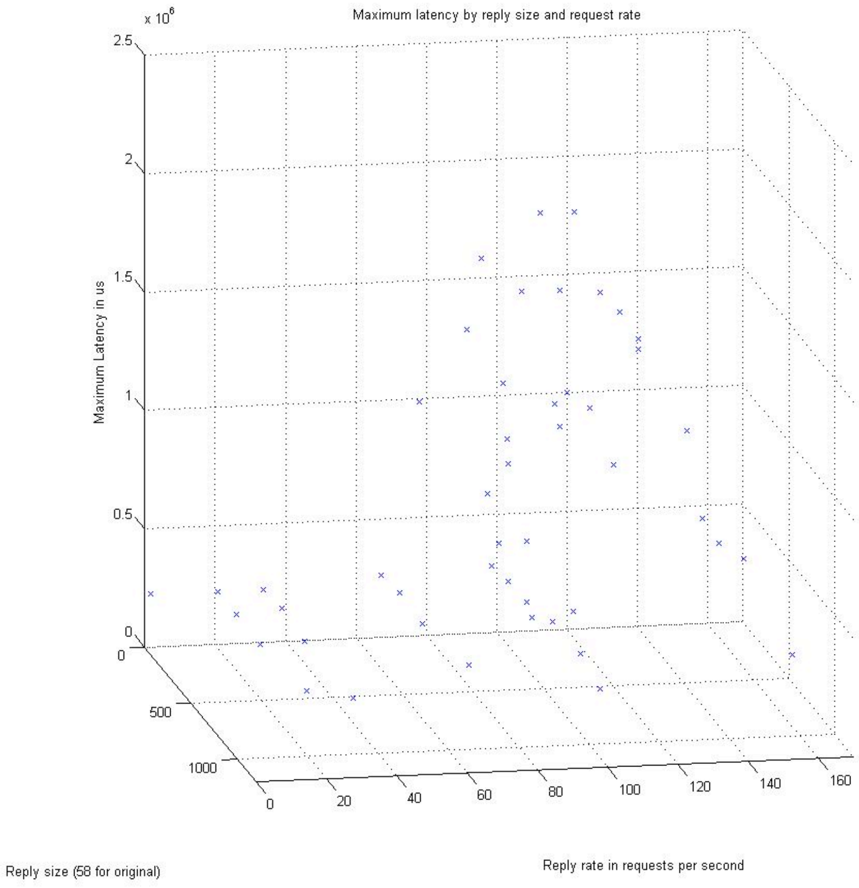
Mean latency for outliers grouped by experiment to better show component breakdown



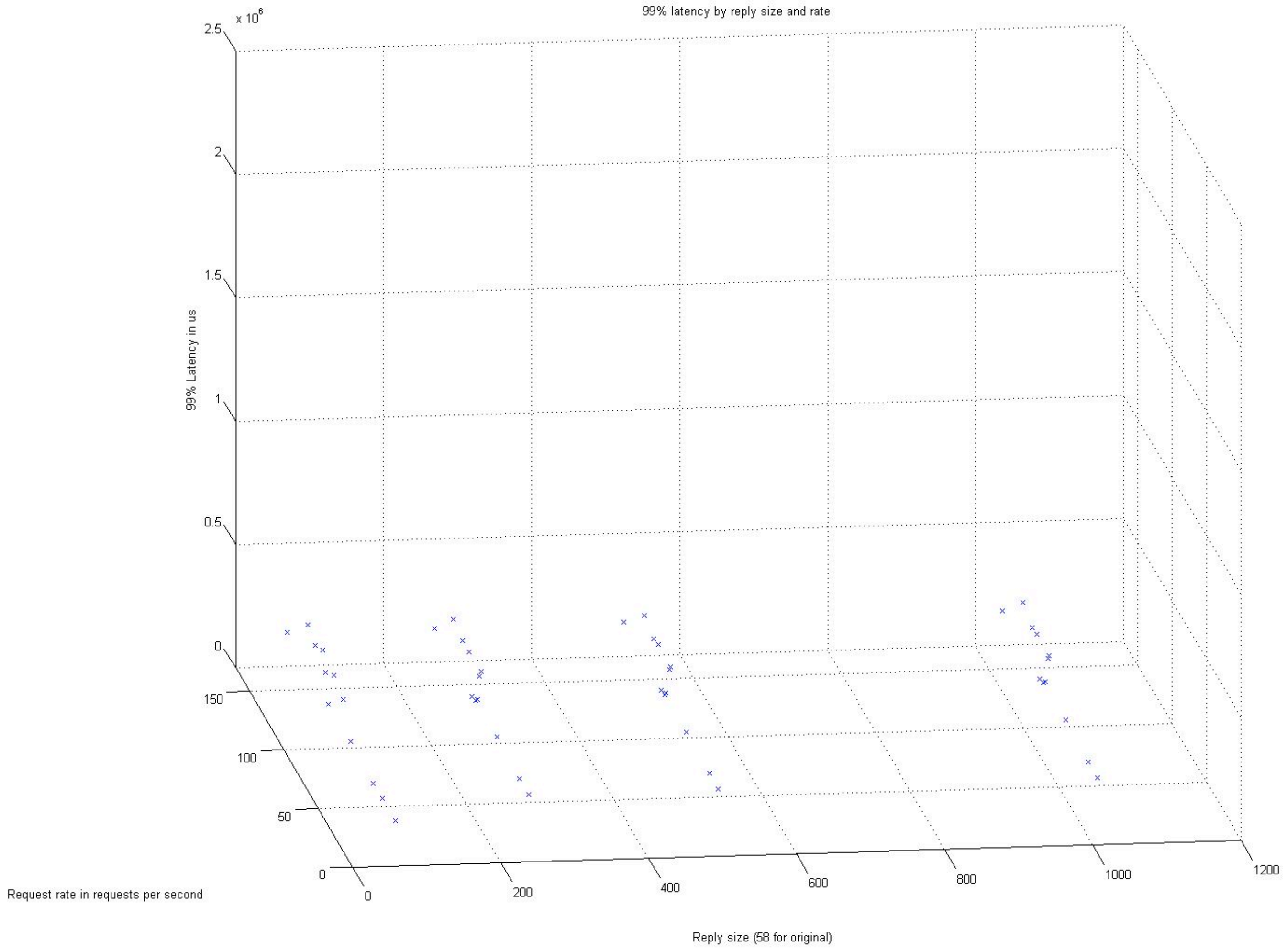
Latency component breakdown for normal request



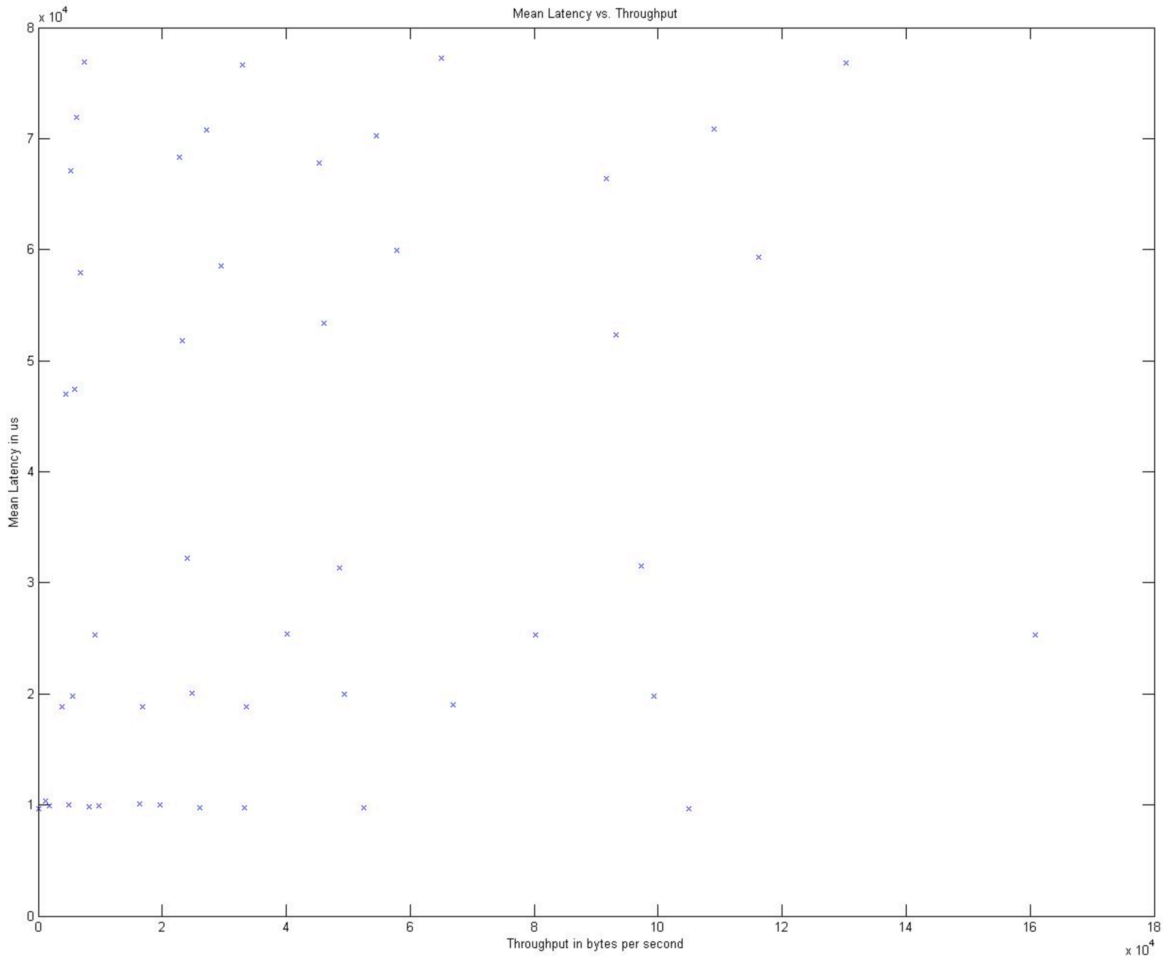
Reply size and request rate impact on max latency



Reply size and request rate impact on 99% latency



Latency vs. throughput


Mean Latency vs. Throughput
x 10^4
Mean Latency in us
Throughput in bytes per second
x 10^4