Analysis Document for Fault Tolerance Evaluation

Team 6: Slackers
Steven Lawrance
Puneet Aggarwal
Karim Jamal
Hyunwoo Kim
Tanmay Sinha

1. List of client invocations that we measured

Method	One Way?	Database Access?	Request Size (in bytes)	Reply Size* (in bytes)
enterLot	N	Y	8	4 * array length
exitLot	N	Y	4	0
getClientID	N	N	0	4
getOtherLotAvailability	N	Y	4	4 * array length
getLots	N	Y	0	4 * array length
moveUpLevel	N	Y	4	4
moveDownLevel	N	Y	4	4
getCurrentLevel	N	N	0	4
getMaxLevel	N	Y	0	4
getMinLevel	N	Y	0	4

*Size of the reply before the experiment padding is added

The following invocations were used in the experiments:

1. getClientID()
2. getLots()
3. enterLot()
4. getOtherLotAvailability()

The average size of the original replies sizes was 4 bytes.

2. Initial analysis of experimental results from our fault tolerance evaluation

The outlier calculation has been done using the 3-s test that is specified in the lecture notes.

Two 3-D Scatter plots: Max Latency vs. 99% Latency

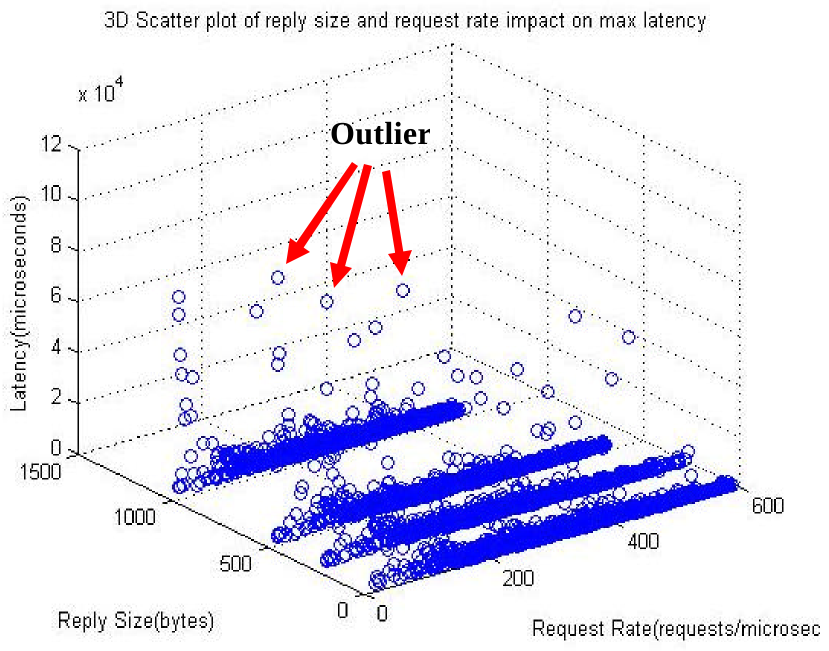

Figure 1. 3D Scatter plot of max latency

- Figure 1 shows the variation of the maximum latency as the request rate and reply size increase. As this is for the maximum latency, we can observe the outliers scattered on the plot.

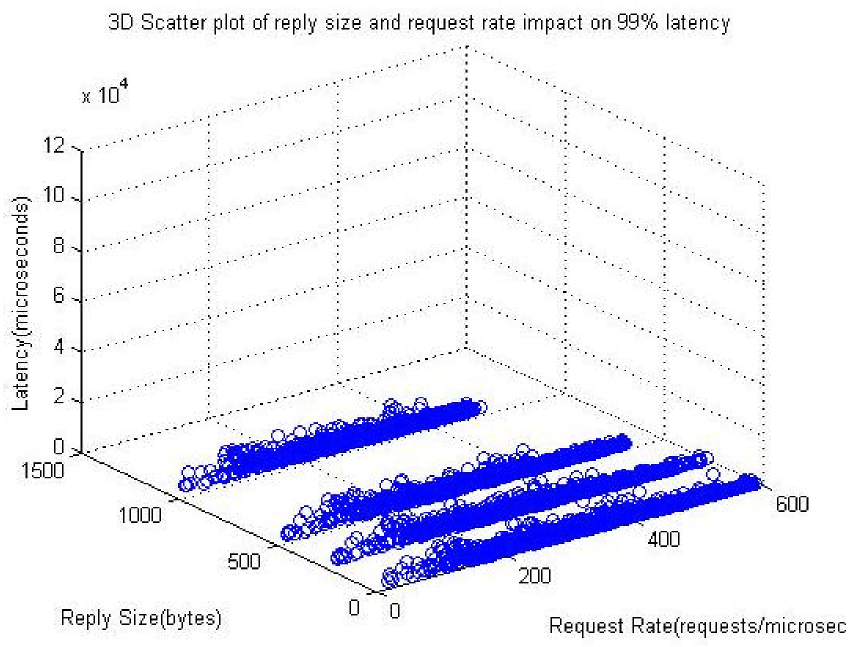

Figure 2. 3D Scatter plot of 99% latency

- Figure 2 shows the variation of the 99% latency as the request rate and reply size increase. After removing the maximum 1% of latency from the dataset in Figure 1, we are able to demonstrate the conformance with the Magical 1% theory. However, by simply observing this 99th percentile, it is not clear whether or not there is a trend that latency increases with the request rate and the reply size. This could be because our experimental configurations were within a small range in both the request rate and the reply size. In other words, we can obtain more observable data if we configure the request rate to be 1000, 1500, and 2000 milliseconds.

Two area plots: (Mean, Max) vs. (Mean, 99%) Latency

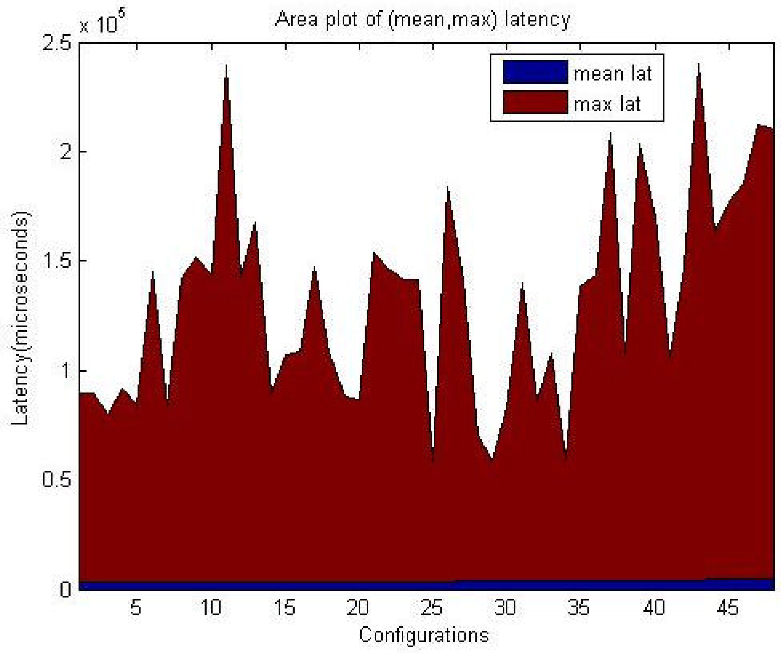

Figure 3. Area plot of mean and max latency

- Figure 3 shows the variation of the mean and maximum latencies across the 48 configurations on the X-axis, which were sorted by the corresponding mean latencies. Here, we can observe that the mean latency stayed almost constant. On the other hand, the maximum latency varied a lot. This could be because of the different combinations or due to some utilization of the servers we were using.

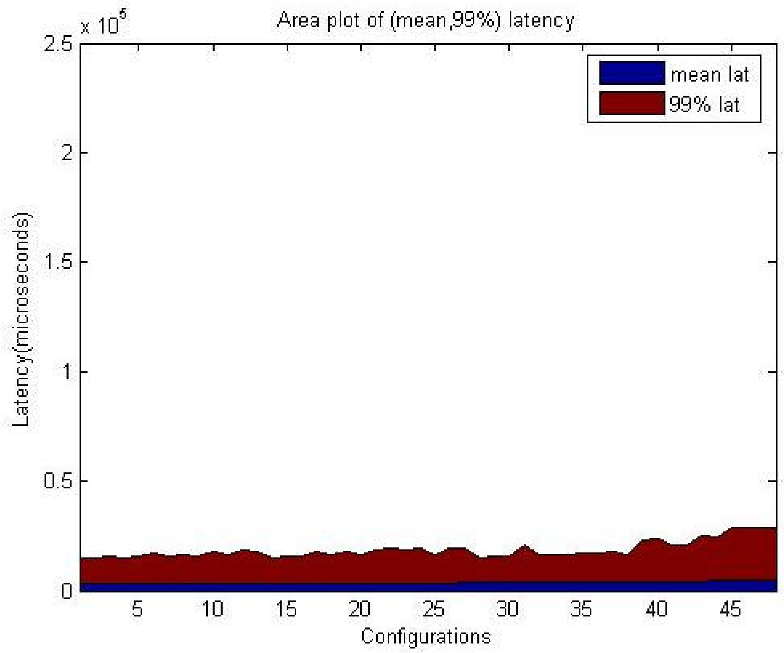

Figure 4. Area plot of mean and 99% latency

- Figure 4 shows the variation of the mean and the 99% latencies across the same 48 configurations, which were sorted by the corresponding mean latencies. After removing the maximum 1% of latency from the dataset in Figure 3, we are able to demonstrate the conformance with the Magical 1% theory. In this figure, we can observe that, even though the 99% latency values did increase towards the end of the graph, these values remained relatively constant or stable compared to the variation of the max latency in Figure 3,.

Line plot: Latency Dependence on number of clients and reply sizes

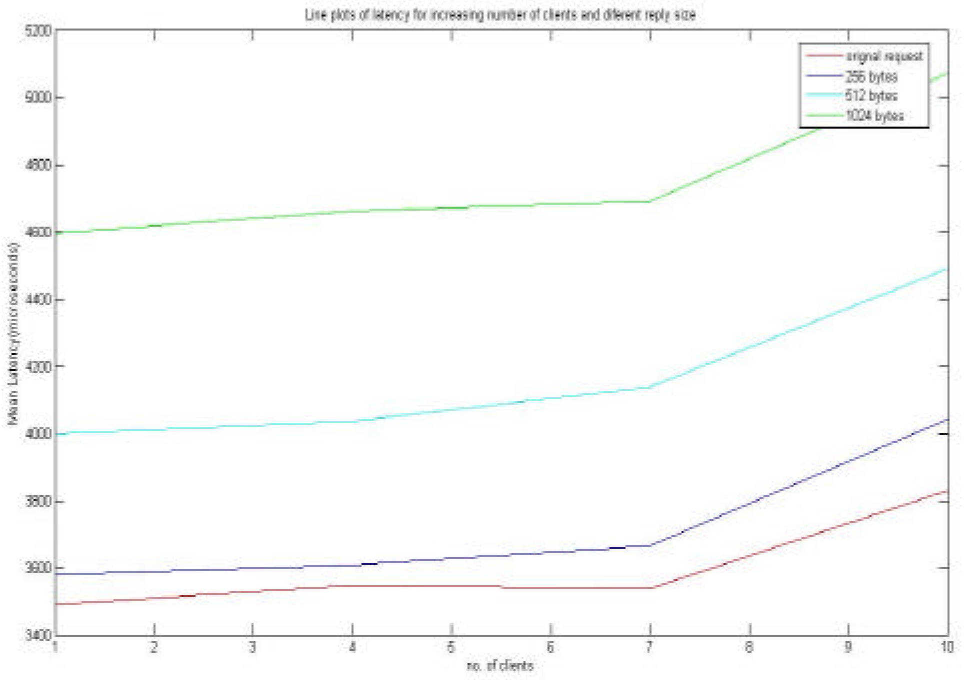

Figure 5. Line plots of latency depending on number of clients and reply sizes

- Inter-request time is 0 ms.
- Figure 5 shows the variation of the mean latencies by increasing number of clients and reply sizes.
- We can observe that the mean latency increases as the reply size increases. This may be because more data has to be transferred over the network; the larger the amount of data, the more time it will take to serialize and de-serialize that data.
- Another observation is that the mean latency keeps increasing as the number of clients increase. This may be because more of the server' s resources are consumed, and so, the server' s response time increases.
- We observed a steep rise in latency when we moved from 7 to 10 clients. The request rate and throughput increases. This might be due to following reasons
  - Simultaneous access to the database (as it has affect on the latency)
  - Serialization of the requests due to the client manager
  - Requests come in faster than what the application can handle
  - Hardware saturation; having more than 7 clients may cause the server' s hardware resources to become totally saturated, leading to increased response time

Bar graph: Configuration vs. Outliers (%)

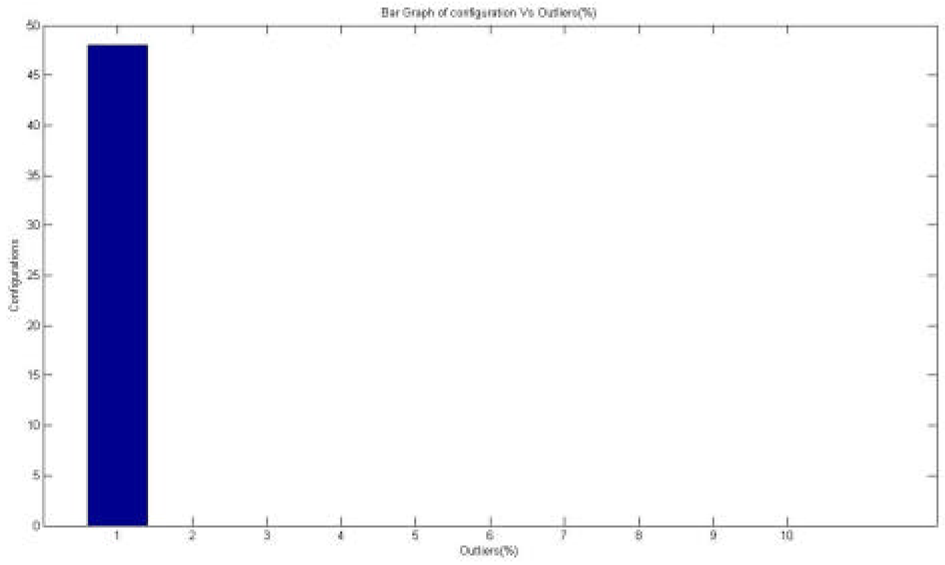

Figure 6. Bar graph of configuration vs. outliers (%)

- Figure 6 shows the percentage of outliers within each configuration.
- We can observe that, in all the 48 configurations, the measured latencies have less than 1% outliers. This means that there is no significant unusual behavior in our fault tolerance system.

Bar graph: Latency Components for Outliers

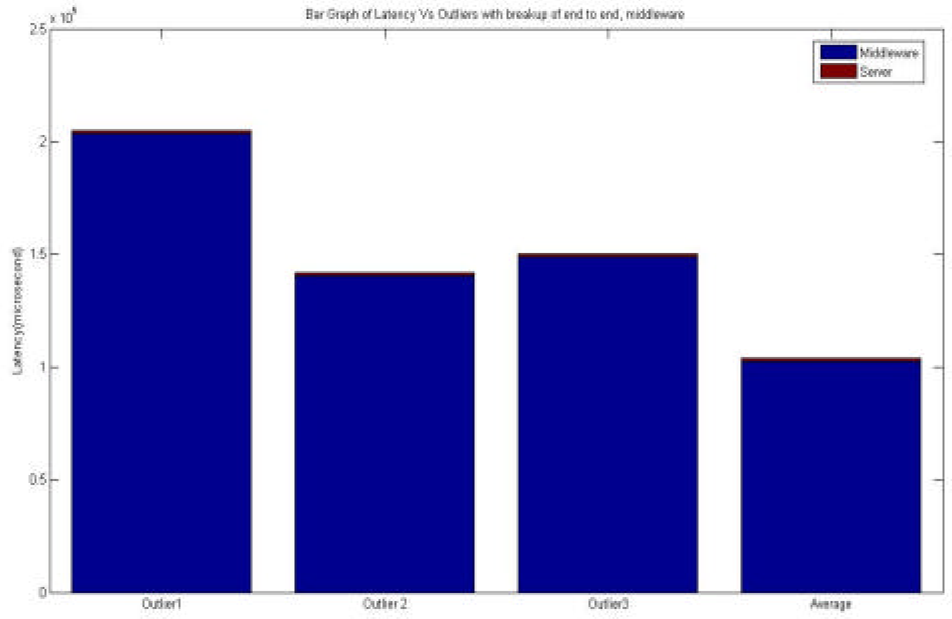

Figure 7. Bar Graph of Latency vs. Outliers with Breakup of End-To-End Application, Middleware and Server

- This graph helps to find the outliers in the configuration. We can then determine the latency within the end-to-end application, middleware, and server that corresponds to that outlier.
- This graph shows that the server has very little contribution as far as outliers are concerned. The middleware and end-to-end application are primarily responsible for the outliers.

Latency vs. Throughput

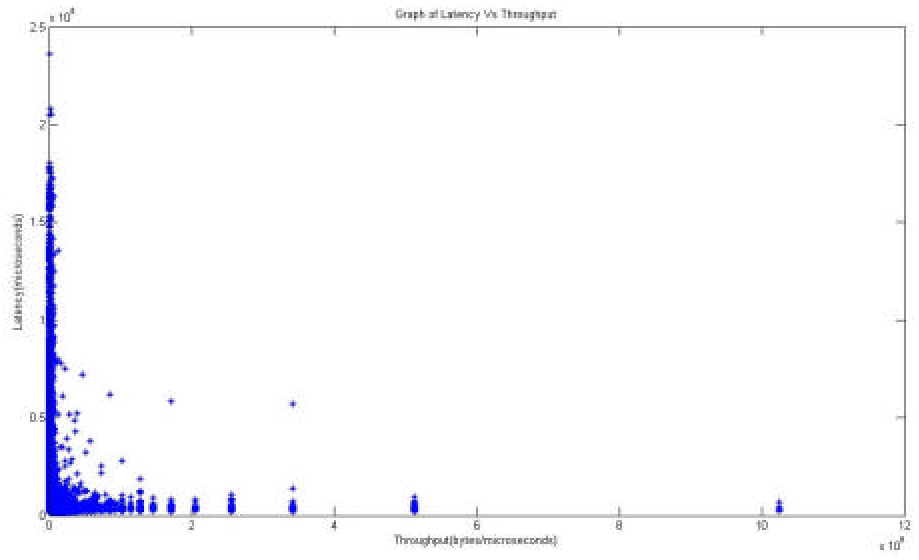

Figure 8. Graph of latency vs. throughput

- Figure 8 shows the relation between latency and throughput, in other words, how many bytes per microsecond the server can transmit to the client based on how long it takes to process the request. The longer the server takes to respond, the fewer bytes it will be able to return to the client per microsecond.
- We would say that if the latency of any service call is more than 50000 microseconds, the server' s throughput is likely to be close to zero.
  Also, we would say that if the latency of any service is less than 50000 microseconds and decreases to zero, the server' s throughput increases significantly or exponentially.

The break up of the number of outliers in the various parts of the application

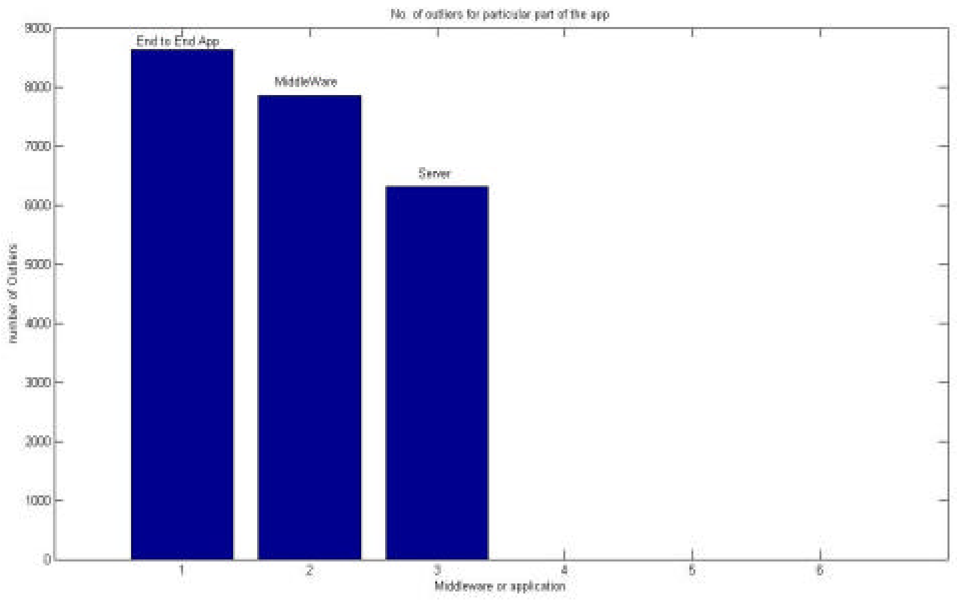

Figure 9. Number of outliers by application part

- This graph shows the break up of the number of the outliers within the various parts of the application (i.e. End-to-end, Middleware, Server).
- It depicts that the server has the least number of outliers, and so it is behaving consistently as compared to the middleware. A possible reason why the middleware has a larger number of outliers may be attributed to the fact the middleware latency also includes the latency to communicate across the network. The policy for scheduling clients through the middleware may also cause the outliers.
- The outlier in the server is because of the padding and un-padding of the reply. It also depends on the other processes running on the machine which hosts the server.

3. **Next steps for fault tolerance evaluation**
    - Implement the Fault Injector.
        - We'll implement it after/during our discussion meeting this Monday
        - For now we can say that the fault injector will be designed to only inject the fault of killing the server; we can include others later based on our discussion.
    - Run the tests again with fault injector working.
    - Add performance tweaks in the project.
    - Run the tests again and do a comparative analysis.
    - Hopefully, we're done after that ☺