Building Scalable Distributed Systems

# Assignment 1

Git Repository: <https://github.com/akksshah/building-scalable-distributed-systems>

# Client Design Description

The entry point for the application is in through the ClientApp.java class.

The clientApp verifies the config.properties properties through the ConfigParameter.java class.

Once the properties have been loaded, the ClientApp gives the configuration to the StoreFactory.

The StoreFactory simulates the opening of stores staggered across 3 time zones as required to emulate in the given spec. The storeFactory creates threads equal to the number of stores to emulate as provided in the spec. Each of the “Store” is a Runnable thread. The job of the thread is to make request to the server a total of numPurchase request per hour the store is opened.

The storeFactory provides a shared latch/countdown to each of the store opened in that time zone. Once the count reaches 0, the storeFactory launches another phase.

The storeFactory provides a shared counter (a synchronized counter) that keeps track of the total number of successful requests the store was able to make and a total of failed request.

For the client in part2

The storeFactory also maintains a list of RequestTracker objects tha store information about the request made by the store. For example, it stores the time when the request was made, when the response was returned by the server, the status code it received in the response and the type of request it made.

This list is then used to understand the system capacity and the MetricsGenerator can then be used to derive insights from the same.

The design that I chose to use was to implement something similar to Singleton design. The counter and requestTracker list are something whose references are provided to each thread/store.

# Client Part 1 Output:

32 threads:

Text

Description automatically generated

64 threads:

Text

Description automatically generated

128 threads:

Text

Description automatically generated

256 threads:

Text

Description automatically generated

1024 threads:

Text

Description automatically generated

1536 threads:

Text

Description automatically generated

1792 threads:

Text

Description automatically generated

2048 threads:Text

Description automatically generated

Note: When I tried to emulate 2048 stores, that is when my client started to throw exceptions caused as I was out of memory/resource limit

Threads vs Wall time

# Client Part 2 Output:

32 threads:

Text

Description automatically generated

64 threads:

Text

Description automatically generated

128 threads:

Text

Description automatically generated

256 threads

Text

Description automatically generated

1024 threads:

Text

Description automatically generated

Threads vs throughput

Threads vs mean response time