Prerequisites

1. Maven
2. RabbitMQ v3 +
3. Elasticsearch

Before performing testing, please make sure that your sls cache, core and rabbitmq hosts are up and running. Perform basic testing to see if the entire system is responding correctly before using any of the components in this latency measurement system.

Also, **Make sure that RabbitMQ is running in your localhost before you run any of these below classes.**

In order to generate load based on a distribution, the following two classes should be run.

Distribution Load Generator:
This is used to generate a number of requests based on the poisson distribution. The mean number of requests can be set in the distributionloadgenerator.yaml file. The various other parameters can be found in the same yaml file. Appropriate comments are left for each parameter.

The class can be run from the command line by
Sudo mvn compile
Sudo mvn exec:java

Latency Checker(needs elasticsearch to be running):
This class receives messages from the localhost rabbitmq and checks the sls cache to see if a given uri has reached the cache. It measures the latency and stores the values into an elasticsearch index. The elasticsearch index and various other parameters can be configured via the latencychecker.yaml file. Appropriate comments are left for each parameter.

The class can be run from the command line by
Sudo mvn compile
Sudo mvn exec:java

In order to generate load based on production logs, the following three classes should be run in the order they are specified

1. **Latency Checker (needs elasticsearch to be running) :**

This class receives messages from the localhost rabbitmq and checks the sls cache to see if a given uri has reached the cache. It measures the latency and stores the values into an elasticsearch index. The elasticsearch index and various other parameters can be configured via the latencychecker.yaml file. Appropriate comments are left for each parameter.

The class can be run from the command line by
Sudo mvn compile
Sudo mvn exec:java

2. Load Generator

This class receives messages from a localhost rabbitmq (from the log feeder class) and generates the request to the sls core for a record specified in the message. The number of threads and various other parameters of this class can be configured via the loadgenerator.yaml file. The higher the number of threads, the higher the number of parallel requests.

The class can be run from the command line by
Sudo mvn compile
Sudo mvn exec:java

3. Log Feeder

This class starts reading from the logs from a timeframe (24 hours before the current system time) from the production logs. It reads the number of renew and register messages per minute and sends that information to the load generator via rabbitmq. Hence starting this class should be the last step.

The log server can be configured via the logfeeder.yaml file.
The class can be run from the command line by
Sudo mvn compile
Sudo mvn exec:java