**CMSC 621 – Project 2 – Report 2**

Group Members: Rohan Gujarathi, Ajay Pal

**Problem Description:**

The chief goal of this project is to construct a distributed web-service systems. This project will describe the infrastructure and architecture used to create the system. Furthermore, it will also describe how to discover the service and will simultaneously invoke them on the basis of load distributed on the nodes. With certain assumptions involved, multiple sites will be hosting collection of web services.

**Changes in design**

As per the previous report, we have planned to use chord system for efficient service discovery and replicated load balancer for uniform distribution of load. The design still remains the same, but due to the complexity of chord, we are first considering to implement centralized web service registry for service discovery which will be replicated. After completing implementation of centralized registry we will try to implement chord system for service discovery.

**Current Progress:**

What is implemented as of now:

* Created 6 different web services for following operations

1. Add numbers
2. Subtract Numbers
3. Multiply Numbers
4. Divide Numbers
5. Square Numbers
6. Cube Numbers

Combination of these web services will be deployed on multiple servers

* Centralized service discovery is also implemented as a web service that will store the end point and WSDL location of a services provided by all the servers. It maintains a Map as follows:

Map<Key,Value> where Key is a string indicating the name of the operation and value is the list of the IP address and port numbers of all the nodes providing that service

Eg: registry = <Add:[192.168.2.1:8080, 192.168.12.1:8080], Subtract:[ 192.168.2.1:8080, 192.168.10.1:8080]>

**Technology Stack:**

1. Tomcat 9.0
2. Java 1.8
3. SOAP using JAX-WS

**What is pending:**

* Implementation of Load Balancer
* Implementation of Chord Algorithm
* Testing

**Testing Strategy for current implementation:**

Assumptions: There are 2 replicas of registry. One of the Registry is always up and running

1. Add a service to a node. The web service should be added in the registry.
2. Search a service through registry. The registry should return all the nodes implementing the services.
3. Test if the server is down, it should be removed from the registry
4. If the registry is down, requests should be transferred to replicated registry
5. When the registry comes up after it was down, it should be able to maintain the same state as the other replicated registry

**Testing Strategy for load balancer:**

Assumptions: There are 2 replicas of load balancer. One of them is always up and running. Load balancer is implemented as round robin

1. Load balancer should be able to contact registry
2. If one of the registry is down, load balancer should send the request to another registry
3. Send multiple requests to the load balancer and check the load of all the servers. The load should be equally distributed. Run this test multiple time
4. If one of the load balancer is down, client should be able to contact another load balancer

**Updated Plan of Execution and Timeline:**

|  |  |
| --- | --- |
| Tasks | Deadline |
| Creation of all the web services | 16th November |
| Creating centralized registry | 21th November |
| Implementing load balancer | 5th December |
| Implementing chord algorithm | 5th December |
| Testing | 10th December |