<u>Design Document</u>

1. Introduction :

The key aspect of this document is to explain how to implement a distributed task execution framework on Amazon EC2 using Simple Queue Service. This framework is composed of following two main components:

1. Client
    - -Local Client
    - -Remote Client
2. Worker
    - -Local Worker
    - -Remote Worker

Functionality of these two components are as under :

1. Client Functionality :

    1.1 :Local Client :
    - -The Client reads data from the workload file and generates the task.
    - -Pushing the tasks in the request queue.
    - -Polling the task execution status from the response queue.

    1.2 :Remote Client :
    - -Reads data from the workload file and generates the tasks.
    - -Pushing the task in the SQS request queue.
    - -Polling the task execution status from the SQS response queue.

2. Worker Functionality :

    2.1 :Local Worker :
    - -Polling the tasks from the request queue.
    - -Executing the task fetched from the request queue.
    - -Pushing the task execution status in the response queue.

    2.2 :Remote Worker :
    - -Polling the Task from the SQS request queue.
    - -Check the TaskId in the DyanmoDB Table.
    - -If task does not exist in the Table then add the task in the DynamoDB table and execute the Task.
    - -Push the task execution status in the SQS response queue.

2. Design Implementation :

■ Local Client-Worker Implementation :

The implementation of the Local Client-Worker is done in JAVA. JAVA gives the inbuilt libraries and functionalities to implement queue data structure, which has made this task very easy.

The flow of execution in Local Client-Worker is as under :

1. The client-worker are in the same java file for the Local implementation.
2. The task of the client is to create a queue, read data from the workload file and add it into the request queue.
3. The task of the worker is to fetch the task from the request queue, execute the task and push the task status in the response queue.
4. After all the task are executed by the worker, client fetches the task status from the response queue and check the status with 0 and 1 value. If Task status is 0 represents the success of the task, while 1 represents the failure in execution.

■ Remote Client-Worker Implementation :

The implementation of the Remote worker is done using Eclipse JAVA. Importing the AWS SDK and configured the AWS with account's region in Eclipse. Moreover, SQS and DynamoDB implementation and configurations are easy to handle in Java. Java provides libraries to work with DynamoDB and SQS.

The flow of execution in Remote Client-Worker is as Under:

1. The client-worker are two different file with Client.java and Worker.java name conventions respectively.
2. The task of the client is to create a SQS request queue, read data from the workload file and push the data into the SQS request queue. In Java, SQS generates unique ids for each task in the queue. Access to this task are done with this ids.
3. While in worker side, task are polled from the request queue and their ids are checked with the TaskId in DynamoDB. DyanamoDB had **Task-Table** which has the schema that contains the TaskID and Job in it. So, if the taskid of the request queue's task matches with the taskid in the DynamoDB, that means this task is duplicate task and should be deleted from the SQS request queue. And if there is no match, that task is added in the DynamoDB and the then it is executed.
4. If the task execution is successful, then that task is added in the response queue with the value 0. if the task execution fails, then value 1 is added in the SQS response queue.
5. At the the end, client fetches the data from the SQS response queue and checks the status of the tasks.

3. <u>Designing Trade-Offs</u> :

- As there is no such fuctinoality to invoke System call Sleep in java, execution of the Sleep task is done using Thread.sleep() function, which is not a system call. Thead.sleep() pauses the current working thread and makes the cpu available to other threads. Execution time for Thread.Sleep() is same as to call system call "Sleep()".

4. <u>Improvement and Extension</u> :

- In local Client-Worker, there should be system that make worker inactive when there is not task in the queue. And when the task is pushed in the queue the worker gets wake up call to execute the task.
- Performing the task on the first-in-first-out basis could have been changed to priority based scheduling.

5. <u>References</u> :

1) http://datasys.cs.iit.edu/publications/2014_CCGrid14_CloudKon.pdf
2) http://aws.amazon.com/ec2/
3) http://aws.amazon.com/sqs/
4) http://aws.amazon.com/dynamodb/