Aditya Kale – A20355807

Cloud Computing

Programming Assignment – 3

Design Document

- Cloudlkon is a scalable and distributed a task execution framework that builds on cloud computing building blocks.
- It comprise of different cloud components viz Amazon EC2, Simple Queuing Service and a DynamoDB.
- It's a master slave architecture where client will send the complete workload to a worker for execution of tasks.
- Below figure shows the architecture of CloudKon task execution framework.

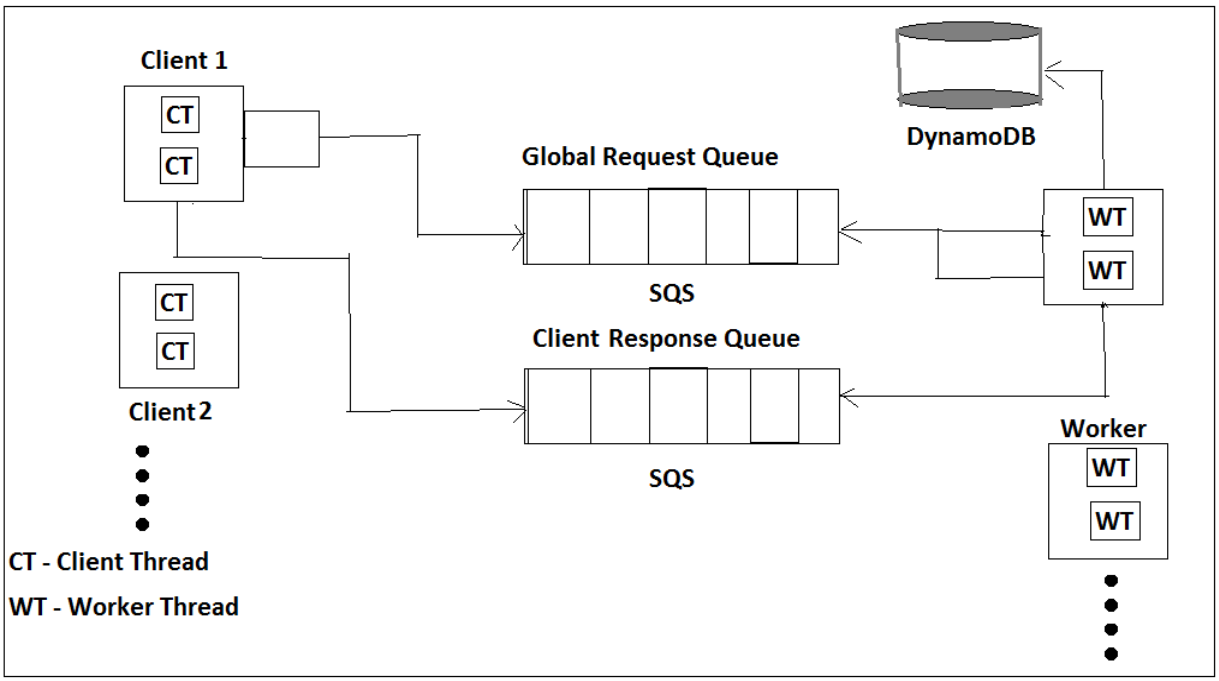

Figure 1. Architecture of CloudKon.

- The above system architecture shows multiple Clients which are multi-threaded and number of multi-threaded workers.
- There are two simple queue in this architecture, of which one is used to put the task by client on the queue from where worker can pick up tasks. The other queue is used to put the responses from a worker after execution of these particular tasks.
- Clients process the workload file and puts task on an execution task queue. It's a simple Amazon SQS service which is used to store these task values.
- Worker pick up all the tasks from this execution task queue and process these tasks. After processing this task it push all the task to a Client response queue.
- Client continuously poll this response queue for the result of execution tasks.

- It can happen that SQS can provide the same task to the different worker. To handle this case and not to execute the same task again worker will first put the data in a dynamoDB. If the task is already present in the database it will not be executed by the worker else worker will execute the task and puts its entry into a client's response queue.
- All the operations on DynamoDB are atomic operations so it handles all the concurrent operations performed by multiple workers.
- I have follow the same architecture for implementing the local system. It has two Concurrent Linked Queue for storing the in memory task and their execution result. These queues are used in place of SQS on local system.
- Also at the place of dynamoDB, Concurrent hash map is used to store and maintain the history of executed tasks.

Classes and their Implementing Methods

For Local System: QueingService

- **Task.java**
  - Task is a simple entity class which has taskId, executionTime as String variables and isExecuted as a Boolean variable.
  - This Boolean variable is used to set the status of the task.

- **TaskCreattion.java**
  - It's an utility class prepared for creation of workload file. It takes and input parameter as no of tasks and the sleep time and generates task accordingly in a **workloadFile**.

- **Worker.java**
  - This is a runnable class. Its run method checks for the task in a request queue continuously. If the queue is empty it breaks the execution.
  - As it process the task it also change the status of that task and put it into a response queue.

- **Client.java**
  - It's a also a runnable class where it just submit the workload file to a scheduler to input task in a request queue.

- **ExecuteTask.java**
  - It's a main class which invokes the Client Thread as well as worker threads.
  - It takes no of threads input from a command line and then generates that number of worker thread.
  - It display complete execution time for processing all task from task submission to task execution.

For Remote System: CloudKon

- **TaskCreattion.java**
  - It's an utility class prepared for creation of workload file. It takes and input parameter as no of tasks and the sleep time and generates task accordingly in a **workloadFile**.

- **Worker.java**
  - This is a worker class which takes input parameter as Queue names and no of worker threads.
  - It create a dynamoDB table which is used in storing the task and checking for the status of task. It checks whether task is already executed or not by any other worker.
  - It also creates two SQS global queue which are used in picking the request and putting the response of a task.
  - It is a main class which calls WorkerImpl class and creates that number of workerImpl threads as taken input from command prompt.

- **WorkerImpl.java**
  - Its an implementation class of worker class. Its run method continuously polls for the task in a request queue. It removes a task from a request queue check in a DynamoDB if entry not exist execute the task and insert response in a response queue.

- **Client.java**
  - It's a main class where it submits all the task to a SQS request queue. And then continuously check the response queue (SQS) for the submitted task responses.
  - As all the tasks are executed correctly it prints the total time required for task execution and then terminates.

- **DynamoDbImpl.java**
  - This class has all the functionalities related to a DynamoDB.
  - It has methods which creates a table, insert entry into a table, check whether an entry is present or not in a table and deletes a table
    createTable(String tableName);
    insertElementInTable(String tableName, String taskId);
    checkIfEntryExist(String tableName,String taskId);
    deleteTable(String tableName)

- **SQSImplementation.java**
  - This class has all the functionalities related to a Simple queuing service (SQS).
  - It has methods which creates a Queue, insert entry into a Queue, retrieve element from a queue and deletes an element from a queue.
    deleteQueue(String queueName);
    retrrieveElementFromQueue(String queueName);
    insertElementInQueue(String queueName, String message);

createQueue(String queueName);
getQueueURL(String queueName);