CS 553: CLOUD COMPUTING

Design Document for Programming Assignment #3

Task Execution Framework

This assignment aims to design and develop a task execution framework, which is similar to cloudKon system. It also aims to provide in-depth understanding of task execution system as well as understanding of how to calculate throughput and efficiency of task execution system, which is very crucial to understand distributed system. Assignment is divided into two parts

1. In-Memory
2. SQS

In-Memory

In this part of this assignment, i have written a "java program" that submit data to queue to in-memory as well as to SQS queue depending on parameter.

- I have written the program is a decoupled way so that it can we integrated with other utilities as well as capable to run on its on.
- Program have a Two Threads as a part of client "LocalProducer" that one for submitting the task and another "ClientResultConsumer"for receiving the task
- I have used two LinkedBlockingQueue `for storing the task.`
- inMemoryQueue for task submission and resultQueue for task result.
- On worker side I have one runnable class that takes data from submission Queue and after processing it puts the data into resultQueue.
- For Management of Thread on worker side I have implemented thread pool functionality using array of threads.
- Thread pool creates number of thread given by user and run worker to execute those task till there is task present in submission queue.
- ClientResultConsumer thread wait for 5ms once the LocalProducer thread have started because we need to have some data present in resultQueue
- Client and worker start simultaneously as different threads on same time.
- We have used hashmap on client side to check all the task completion by worker.
- Code is implemented in such a way that we have different classes doing its own functionality and a main class responsible for running all those task.

Improvements and extensions

Evaluating throughput and efficiency of system is always a challenging task. There are number of ways to solve these problem. One way to solving this problem is to create thread pool using Executor Framework of java because that takes care of creating number of threads as needed

SQS Queue

In this part of this assignment, i have written a "java program" that submit data to SQS queue and uses DynamoDB to checking the duplicate task.

- I have written the program is a decoupled way so that it can we integrated with other utilities as well as capable to run on its on.
- Program have a Two Threads as a part of client "SQSClientDataSender" that one for submitting the task and another "SQSClientDataSender" for receiving the task
- I have used two SQS queue for task submission and result of task submission .
- SQSManager class is responsible for making connection to SQS and sending and reciving data from SQS Queue.
- DynamoDB class is responsible for making connection to DynamoDB and putting and getting data from DynamoDB.
- For Management of Thread on Worker side I have used Executor framework to create threads.
- I have used 8 threads for each worker as from previous experiment I found that 8 threads is more efficient.
- Worker Thread class pulls data from SQS submission Queue and check it from DynamoDB to know is the task is already processed.
- After completion of task WorkerThread put data in result SQS queue.
- Client and worker start simultaneously as different threads on same time.
- We have used hashmap on client side to check all the task completion by worker.
- Code is implemented in such a way that we have different classes doing its own functionality and a main class responsible for running all those task.

Worker Code for pulling task

```
public void reciveMessage() {

    List<Message> messages = taskSubmitQueue.receiveMessage(this.taskSubmitQueueUrl).getMessages();

    //System.out.println(messages.size());
    if (!messages.isEmpty()) {
        for (Message message : messages) {
            if (DynamoDB.getDataFromTable(dynamoDBTblName, message.getBody()) == null) {
                DynamoDB.putDataIntoTable(dynamoDBTblName, message.getBody());
                String[] msgTokens = message.getBody().split(" ");
                try {
                    Thread.sleep(Integer.parseInt(msgTokens[2]));
                    SQSManager.sendMessage(this.taskCompletedQueue, this.taskCompletedQueueUrl, message.getBody());
                } catch (InterruptedException e) {
                    SQSManager.sendMessage(this.taskCompletedQueue, this.taskCompletedQueueUrl, message.getBody());
                }
            }
            String messageReceiptHandle = message.getReceiptHandle();
            taskSubmitQueue.deleteMessage(new DeleteMessageRequest(this.taskSubmitQueueUrl, messageReceiptHandle));
        }
    }
```

Improvements and extensions

There are various systems available in market like cloudKon, MATRIX. My program is designed for understanding the task execution functionality we can use other technique to schedule job to worker.