CloudKon clone with Amazon EC2, S3,SQS and DynamoDB

Design :

The assignment performs the implementation of a distributed task execution framework on Amazon EC2 using SQS. It involves implementation of task execution framework similar to CloudKon. The SQS service is used to handle the queue of requests to handle a queue of requests to load balance across multiple workers.

We perform the following task execution framework:

Client

Local Back-End Workers

Remote Back-End Workers

Duplicate tasks

In the client task execution framework the following tasks have been provided: Sleep 1000, sleep 100000, sleep 0, sleep 500 . Here Client acts like nothing but a command line tool which submits tasks to SQS which is a fast, reliable, scalable, fully, managed message queuing service. The tasks are being performed using WORKLOAD_FILE which will have the above mentioned tasks. The client essentially communicates with the SQS and DynamoDB performing the tasks.

Client –s QNAME –w <WORKLOAD_FILE>

The Local Back-End Workers consists of two queues a task queue and a result queue. The communication between the client and worker is through an in memory queue. The task queue adds on the workload tasks from the file and performs those tasks for 100K times for 1,2,4,8 and 16 threads. Once the sleep

tasks are completed a success return message is returned. Once this is done the tasks are passed from the task queue to the result queue. The following command is used to perform the above mentioned explanation:

Client –s LOCAL –t N <WORKLOAD_FILE>

In the remote back end user the operation performed in these frameworks are similar to that of the local back end worker with the difference being that this has the ability to run on different machines to allow large amount of computing scales. SQS is used to communicate between the client and the worker. Here the number of backend workers range from 0 to 16. Here DynamoDB is used to store that tasks and to check if the task while being executed is The design is implemented in such a way that 1 task is considered at time and only 1 task is processed at a time and multiple tasks are run concurrently at the same time. The command used here:

worker -s QNAME –t N

We also need to check whether the message being sent is being delivered only once. SQS does not guarantee that. In order to make sure of it we use the DynamoDB which keeps track of what message is being sent and if that message has already been used then it can be discarded.

User Manual:

First install aws software configuration in eclipse. Select aws toolkit while doing so. This is basically done for remote back-end as we can SQS and DynamoDB by doing so.

In order to successfully access aws tool kit in eclipse, we need to first generate the access key from the amazon aws website and provide the access key id and the secret access key when asked in eclipse.

In amazon aws launch t2.micro instance andet credentials in the path

To run the local back-end we ClientLW.java which takes in the workload file which does the sleep operations. We use threads ranging from 1 to 16 which fetches the sleep tasks from the file and executes it N number of times. N is the number of tasks assigned to it while entering in the command line tool.

Once the command line tool is entered sleep operations are performed and the time taken for the operation is displayed

The command line is

Client –s LOCAL –t 1 -100000 WORKLOAD_FILE

Complie: javac ClientLW.java

Java ClientLW

To run the Remote Back we use the ClientRW.java which takes in the number of the tasks to be performed and number of workers being used . Here the communication between the client and worker is done by the SimpleQueueService.java which transfers the tasks one after the other and performs it. It is then DynamoDb class which stores it and also takes care of the duplicate tasks issue that SQS does not guarantee to take care of. Here the number of nodes varies from 1 to 16