CS 553 Programming Assignment#3

CloudKon clone with Amazon EC2, S3, SQS, and DynamoDB

Harsh Parikh

Hparik11@hawk.iit.edu

A20338453

**Design Document**

**ABSTRACT**

Amazon Simple Queue Service offers reliable and scalable distributed queues for storing messages which can be used in distributed programming applications. By using Amazon SQS, you can move data between distributed components of our applications that perform different tasks without losing messages or requiring each component to be always available. As Amazon SQS is a distributed queue system, it enables web service applications to quickly and reliably queue messages, that one component in the application generates to be consumed by another component. It acts as a temporary repository for messages that are awaiting processing.

**DESIGN**

The system is developed using python. Boto library is used to write the system that makes use of Amazon services like SQS, S3 and EC2. This framework provides an easy to use, object-oriented API as well as low-level direct service access. Multi-threading (Multi-processing in Python) is used to parallelize the jobs among all the available cores. The performance is checked by varying the number of threads as well as the “sleep n” tasks. All the arguments can be passed using the command line tool. Python’s “argparse” library is used to parse the command line arguments. The command line arguments for each program in explained in the manual section below.

**1. The Client:**

The client basically submits all the tasks from the given workload file on the AWS SQS. The queue is first created on the AWS if not present. Then the tasks are submitted one by one on the queue. If the queue is already present then, the tasks are directly submitted. Each task is submitted with a unique identifier to identify each tasks which are similar. Now each task can be identified using this Task ID.

**2. Local Back-End Workers:**

This is a simple in-memory queue system developed in python which can be used to submit, retrieve and execute tasks to and from the queue. Multithreading is used to parallelize the tasks among various cores. The client can submit the tasks using command line arguments. This is further explained in the Manual section below.

**3. Remote Back-End Workers:**

This program is used to retrieve the tasks from the AWS SQS queue and execute those tasks. The tasks are identified using the Task ID’s created by the client program. This tasks can be submitted on different machines also. DynamoDB is used to guarantee that the duplicate tasks are not executed multiple times. This avoids duplicate tasks. AWS SQS does not guarantee that the messages are delivered only once. The number of workers are varied from 1 to 16. The performance for each worker is measured and is used to plot a graph. Here while passing argument, we need to pass the number of Workers we need to invoke. So those many worker/instances will be launched and at the same time we need to run script for running worker file. While using SQS, we may get duplicate number of tasks so that we have to use dynamodb to discard duplicate ones.

**4. Animoto**

The same system developed for submitting and retrieving the jobs is used for this jobs too. The “wget” and “ffmpeg” commands is first installed on all the systems and then the jobs to be processed are accepted. The workload file is changed accordingly.