CS553 Cloud Computing
Programming Assignment 3
Design

submitted by:
Chiranjeevi Ankamredy
A20359837

The assignment is on distributed load balancing and how to distribute work between clients and workers It involves implementing a distributed task execution framework on Amazon EC2 using the SQS and implement a task execution framework similar to Cloudko.I used EC2 to run your framework; the framework should be separated into two components, client (e.g. a command line tool which submits tasks to SQS) and workers (e.g. which retrieve tasks from SQS and executes them). The SQS service is used to handle the queue of requests to load balance across multiple workers and DynamoDb is to Eliminate Duplicate Task Execution.

The Task Execution Framework consists of two parts.

1.Local System

2.Remote System

1.Local System:

The Local System Framework code has been written in java. It consists of two parts.one is client side and another is Local BackEnd Workers.

Client: The client is a command line tool, which can submit tasks to the SQS.I implemented using java, as well as the TCP communication protocol between the client and the SQS. The client should follow this interface: client –s QNAME w The QNAME is the name of the SQS queue.I started time when client starts to send tasks. if QNAME does not exist, the client should terminate. The WORKLOAD_FILE is the local file (client side) that will store the tasks that need to be submitted to the SQS.workload file consists of sleep 1000 sleep 10000 sleep 0 sleep 500 Each task is separated by a new line character. Each line is a task description. The Task performs Sleep operation like “sleep 1000” denotes that the task is to invoke the sleep library call for 1000 milliseconds. Since the client is simply reading these task descriptions and send to SQS. And here I Submitting each task one by one.and To be able to keep track of task completion, each task should be given a unique task ID at submission time. Once all tasks had been sent, the client should wait for results to be received at the client, and every received task ID result should be matched with the corresponding submitted task ID.

Local BackEnd Workers:

Local System framework will only support sleep tasks, and therefore you can in run a large number of sleep tasks on the same resource where the client runs, as the tasks are extremely lightweight and do not require significant amount of resources. I implemented configurable size pool of threads that will process tasks from the submit queue, and when complete will put results on the result queue. The pool of threads can be extremely simple, and only have to handle the sleep functionality, of a specified length in milliseconds. When the sleep task is completed, a success returns value 0 should be returned and failure returns returns Value 1. And t the communication between the

client and the workers is via the inmemory queue like insert tasks ,remove tasks and retrieve tasks. with a client, and local workers, and you should be able to run sleep workloads.I started the local workers by: client –s LOCAL t N w Where LOCAL denotes that it is the local worker version, N denotes the number of threads in the thread pool.

2. Remote System

Remote Consists of various Parts. They are client, work reader, Sqs, Worker, DynamoDb, job Sceheduler.

Client: I Installed Amazon AWS on Eclipse and Made all the configurationsSqs,DyanamoDb,EC2Instances and S3. The client is a command line tool, which can submit tasks to the SQS.I implemented using java, as well as the TCP communication protocol between the client and the SQS. The client should follow this interface: client –s QNAME w The QNAME is the name of the SQS queue.I started time when client starts to send tasks. if QNAME does not exist, the client should terminate. The WORKLOAD_FILE is the local file (client side) that will store the tasks that need to be submitted to the SQS.workload file consists of sleep 1000 sleep 10000 sleep 0 sleep 500 Each task is separated by a new line character. Each line is a task description. The Task performs Sleep operation like “sleep 1000” denotes that the task is to invoke the sleep library call for 1000 milliseconds. Since the client is simply reading these task descriptions and send to SQS. And here I Submitting each task one by one.and To be able to keep track of task completion, each task should be given a unique task ID at submission time. Once all tasks had been sent, the client should wait for results to be received at the client, and every received task ID result should be matched with the corresponding submitted task ID. When running the client, it is important that the client also run in the cloud, in a VM.

Remote BackEnd Workers: I implemented backend workers using Java and Runs on Amazon Ec2 Instances which have the ability to run on different machines to allow large amounts of computing to scale and leverages Amazon AWS services. In order to allow a general purpose cloudkon enabled solution, i used the SQS service to communicate between the client and the backend workers, and vice versa. I assumed only 1 client, and N backend workers, where N could be 0 to 16; I useed spot instance types to run Workers. The worker should only receive 1 task at a time, and process 1 task at a time. And it allow smultiple tasks to run concurrently at the same and allow completed tasks to be returned back to the client (via SQS) as soon as they are complete. The interface to start workers with worker –s QNAME –t N and the remote workers run on different VMs than the client and other workers.

SQS:I have created JobSqs to receive and store the all the tasks from the client and workers will take the tasks from SQS and execute the Tasks and returns output into the queue.

DynamoDb: I have created table to store task id and allows read and write should be 1 since SQS does not guarantee that messages from SQS are delivered exactly once. DynamoDB is to keep track of what messages sent from the Sqs, so that if a duplicate is retrieved, it can be discarded.

Animoto Clone: I have implemented a web applicationthat converts pictures to video, which will involve both significant network I/O and significant amounts of processing. I have used all the functionalities of Remote System, but instead of running sleep task, used the real time app.S3 is used to store the video file. I have used images on the web.and workload will then consist simply of URLs to the images .Using wget is to Download all the images and all the URLs were listed. ffmpeg is used to convert images. Once the video has been created to the local disk, it should be written to S3, and the client notifies the location of the video.