Programming Assignment 3 – Report

Prasanth Bhagavatula - A20355611

Table of Contents

Design.....3
Manual.....3

Design

This section provides details about the overall program design. The program is to implement Client-Worker task execution model using two methodologies. One method uses In-memory queue and the other to use the Amazon SQS. As Amazon SQS does not guarantee one-time delivery only, Amazon Dynamo DB has been used to track the status of tasks executed and thus ensuring one-time execution of each and every task.

Since the main communication channel is different, separate classes have been written for each of the methodology implementation. For the In-memory queue implementation, 'LocalClient' and 'LocalWorker' classes have been defined. In-memory queue is implemented using 'LinkedBlockingQueue' class in Java.

For the Amazon SQS implementation, as the name itself suggests, an AmazonSQS queue is created and used by Client to store the task information. The workers retrieve data from Amazon SQS queue and check in a DynamoDB table for the task status (entry would be present if the task is complete) and then process accordingly. Classes 'RemoteClient' and 'RemoteWorker' have been defined to carry out this processing.

Manual

This section provides details about how the program works.

For the In-memory queue processing, the codes required are 'Client.java' and 'LocalQueueProcessing.java'. The 'Client.java' code will parse the options provided and if it needs to implement the In-memory queue processing, LocalQueueProcessing object will be created and it will take care of creating the different threads. The LocalQueueProcessing object creates two queues, one for requests and other for responses, and then creates a LocalClient object and starts a thread which will creating entries into the In-memory queue. Then, the LocalQueueProcessing object will create LocalWorker objects based on the number of workers to be created and then starts a thread for each of them. The LocalWorker objects reads from the In-memory queue and then executes the task. Once the task is executed, an entry is made in the response queue indicating that the task is executed.

For the Amazon SQS implementation, the required java codes are 'Client.java', 'RemoteQueueProcessing.java', 'Worker.java' and 'RemoteWorker.java'. As explained above, the 'Client.java' is used to parse the options provided and creates an object for RemoteQueueProcessing. The RemoteQueueProcessing object will inturn create an object of 'RemoteClient' which will read the input task file and create messages on the Amazon SQS queue. The 'Worker.java' code is used to create worker instances and in turn run the 'RemoteWorker' code which will read the messages from Amazon SQS and checks the status of the task in Amazon DynamoDB and then either executes the task or discards the task.

The In-memory queue implementation is done using the pre-defined LinkedBlockingQueue class in Java and for the Amazon SQS, we use 'sendMessage' and 'receiveMessage' APIs for sending and receiving messages from the queue and 'putItem' and 'getItem' APIs are used for creating and

checking items in Amazon DynamoDB table.

A 'Makefile' has been created for the compiling the codes.