Assignment 3

Structure:
The project uses the pub/sub features of Google app engine to handle distribution of messages. Topics are can be established both manually and dynamically. Each subscriber can perform two kinds of function, push or pull, and can be changed after establishment but can not perform both function together. Push method is used when subscriber traffic is low and they are often offline. Push method is more close to a real life application. But pull is used for test case like this. Subscriber can be established also in two ways and can be only subscribe to one topic at a time. Multiple subscribers can be onto one topic. Ack deadline is auto set up to 10 seconds that if no message received success code is send back to server, after certain amount of time, server will try to resend the message. When the topic is deleted, subscriber will shows in a deleted topic section instead of auto deleted. Message can be created in two ways, and will stay in the message queue until subscriber ack the pull function.

The message receivers in the pub/sub design then take the messages and submit them to a task queue to be asynchronously processed by a worker that examines the words for example stop words and updates the information for each word. The word information is stored in the Google Datastore, with each word containing information on the number of times the word has appeared (stored in the “count” field) and the number of messages it has appeared in (stored in the “sources” field). Every message sent to the worker process is split into individual words, each word is checked against example stop words, and then the datastore is queried for that word. If no result is returned, a new entity is made for that word and stored. Otherwise, the existing entity has its counts updated and saved.
The data in the datastore is interacted with through an ObjectiveService, for ease of accessing and modifying data (as the ObjectiveService handles the conversion between the entity in the datastore and the class holding the information). The word information entities in the datastore are indexed by their count field, allowing a query to sort through entities by how many times the word has occurred, in order to select the top n most common words. The query can take an integer input to determine the number of results returned.

The task queue used is the standard push queue, allowing app engine to handle the distribution of messages to worker servlets.
Scaling of instances is handled automatically by Google app engine, to respond to increases in traffic.

createPublisher/createSubscriber
Takes args[0] is input and create corresponding topics and subscribers.
messagePub:
20 threads were set to process message publishing function and each thread will perform 5% of message publish job on each topic.
messageRec:
20 threads were set to process message pulling function and each thread will perform 5% of message pulling job for each subscriber. Pulled message can be printed out by getPayloadAsStering()

Performance:

2 instances were initially used, and a 3rd one were automatically added with auto scaling rule.
Totally amount of requests can be found on instance home page.
Test1:
20 topics 100 messages each. Mean: 0.621min

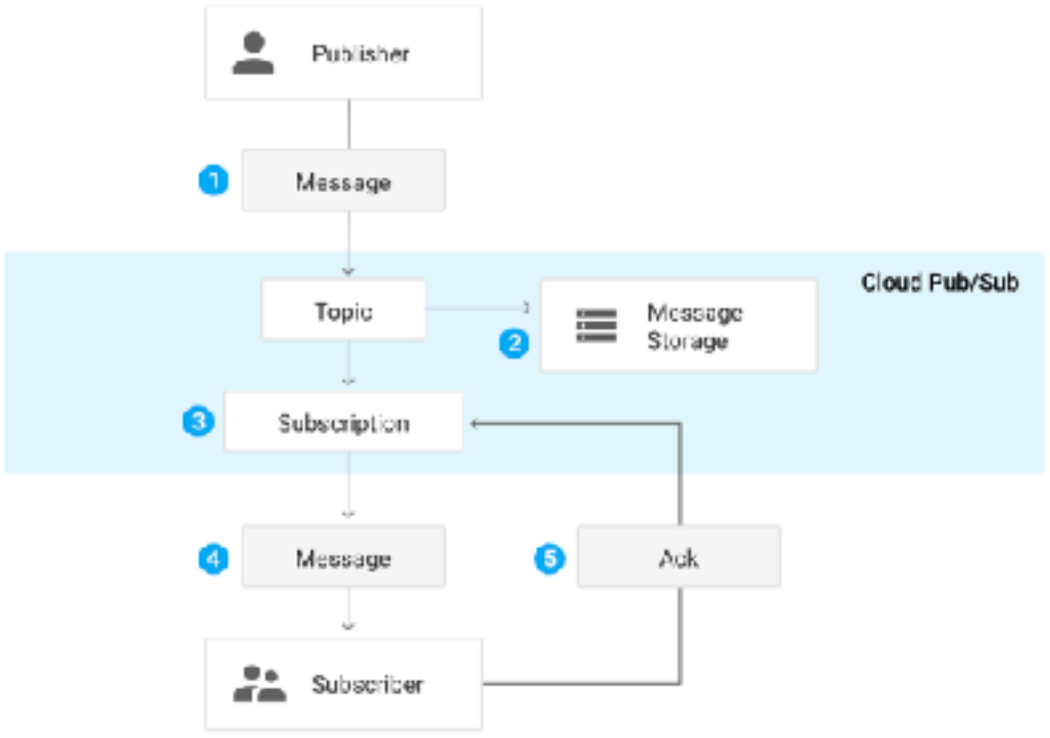
Test2:
20 topics 1k messages each. Mean: 2.9 mins

Test3:
20 topics 10k messages each. Mean: 26 mins

Test4:
20 topics 10k messages each 10 subscribers. Mean: 40 mins.

Tset5:
20 subscriber 10k messages. Mean: 12 mins.

Reference
https://github.com/GoogleCloudPlatform/google-cloud-java/tree/master/google-cloud-pubsub
https://cloud.google.com/appengine/docs/java/tools/setting-up-intellij
https://cloud.google.com/pubsub/docs/overview


Publisher
1
Message
Cloud Pub/Sub
Topic
2
Message Storage
3
Subscription
4
Message
5
Ack
Subscriber