CPSC 416: Project 2 Proposal

Distributed MapReduce:
Calculate K-Means on Large Clusters

Professor Ivan Beschastnikh

HangYee Wong - r9i0b
Kailun Hong - s2d0b
Qiushan Zhao - n0g9
Yukang (Ken) Zhang - u2z8
Zixuan Yin - p4d0b

Introduction

Inspired by a pressing demand of corporations for building a global market model without giving up their data privacy, we decide to build a distributed system based on the MapReduce programming model developed by Google and apply it to K-means clustering (Dean & Ghemawat, 2008). We treat a company providing its data and computation power as a resource node. The system will train a global model using data from all resource nodes in the network. Upon user program request, the system will classify the user provided data into k categories using that model. The system is expected to gracefully survive node failures, as well as incorporate various optimizations such as load balancing and backup tasks, in order to further boost the performance.

Background on K-Means

K-means clustering is an unsupervised machine learning method. It is used when there is data without defined categories. The goal of this algorithm is to find categories in the data, with the number of categories represented by the variable K (Trevino, 2016). The algorithm works iteratively to assign each data point to one of K groups based on the provided characteristics of the data point. Data points are clustered based on feature similarity. The results of the K-means clustering algorithm are centroids of the K clusters, which can be used to label new data (Trevino, 2016).

Types of Nodes

There are three types of nodes in our system:

- **User Programs:** Send request to master for training a new model or classifying a user provided dataset.
- **Master Server:** Monitor the states of nodes in the network and assign tasks for them.
- **Resource Nodes:** Join the network with their own training dataset that will contribute to the training of global model, and work on map and/or reduce tasks assigned by the master while connected.

Topology

Depending on the state of the system (i.e. idle, map task in progress, or reduce task in progress), it assumes two topologies : star and all-to-all.

When the system is idle, for instance, when it just completes a training phase and there is no immediate classifying request, all resource nodes send heartbeat to the server, demonstrating a star topology.

During a map task, all resource nodes work on their local data without needing to contact each other, which is a star topology as well.

When the system moves on to a reduce task, the resource nodes will establish connection with each other transfering results of previous map task, which serves as input at current stage. In this case, a all-to-all topology is present.

Key Assumptions

- We assume that the master server is always online, doesn't fail, and doesn't misbehave.
- We assume that there is a process to check nodes' credibility in the real world (See in the Security and Data Privacy section).

Detailed Workflow

There are two phases which are training and classifying in the system. We design our own map and reduce functions in each phase. The detailed distributed system processes will be described from (1) to (7) below. Please refer to figure 1 and 2 for graphical overviews.

Step (1):

- Client program specifies k (number of clusters) in the request and the location of the data to be classified in the Azure database, and sends it to the master.
- Master checks the system's availability. If the system is not available, it queues the request for later processing.
- Master checks if the model needs to be retrained. The model needs to be retrained under two conditions:
    - The user provided k does not match the number of clusters of the current model.
    - There has been nodes joining the system since last train.

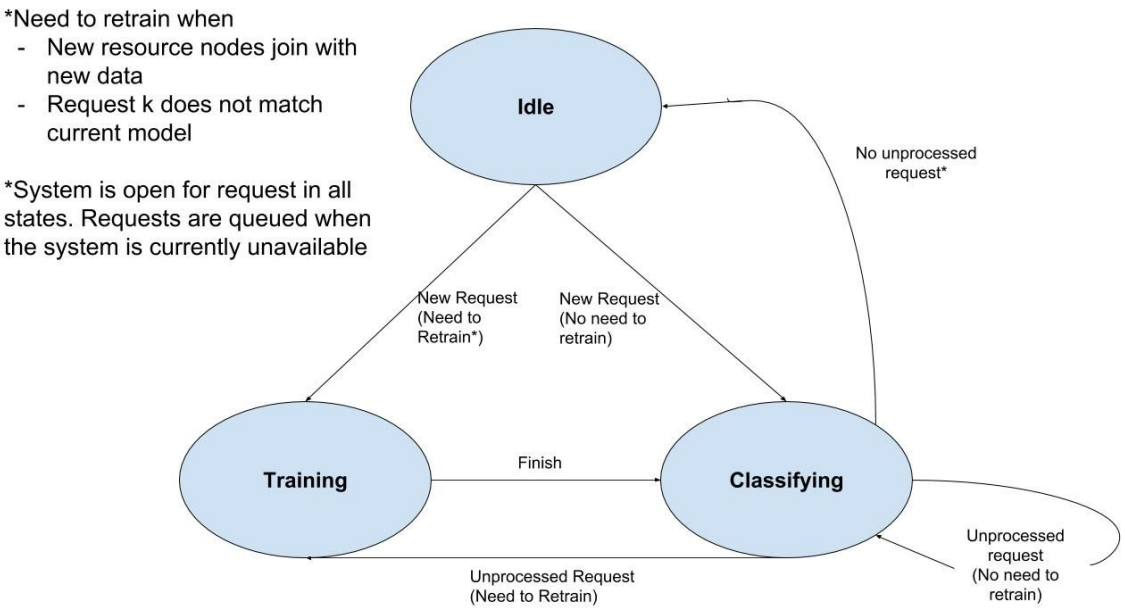

Figure 1

The workflow of training and classifying diverges from step (2).

Training Phase

Step (2):

- Server notifies resource nodes of the new map task, providing them with following information
    - Which map function to execute, and
    - The number of clusters k, and
        - Randomly initialized means if this is the first iteration, or
        - Existing means

Step (3):

- Resource nodes read their private data and apply the training phase map function to each data entry.

Step (4):

- The output of the map function is a list of key-value pairs. Resource nodes write this output to local disk, and notify the server when they have completed the writes.

Step (5):

- Server waits until it has received notifications from all resource nodes in the map phase. After that, it assigns keys to resource nodes. The keys are integers from 1 to k (k is the number of cluster). The server sends each resource node the keys they should work on and the IP addresses of the nodes they should contact in order to retrieve the corresponding key-value pairs.

Step (6):

- The resource node contacts the server provided IPs to retrieve the corresponding key-value pairs.
- For each key, the node applies reduce function and output the corresponding value, which is the global mean associated with the key.
- The node sends newly generated key-value pairs back to server.

Step (7):

- Server updates existing global means
- Server checks if new means generated from the reduce process equal to means from the previous iteration. If they are not equal, the system starts a new iteration by returning to step (2).
- In theory, new means would equal to old means eventually.

Classifying Phase

Step (2):

- Server distributes the keys to resource node. Keys are integers from 1 to n, and n is the number of data entries provided by user program from Step1.
- Server informs resource nodes of the keys they should work on, k means from the training phase, the locations of the data entries, and which map function to execute.

Step (3):

- Resource nodes retrieve the data from the designated locations.
- Resource nodes apply the map function to each data entry, which calculates the distance between the object and each global means received from the server. We complete both map and reduce locally at the step since resource nodes have k means locally and they have the data entries to complete the reduce step. Thus, we do not have to start a whole global reduce process and increase the workload of the system.
- Resource nodes write the new key-value pairs to the global Azure database, and notify the server when they have completed the writes.

Step (5):

- The server notifies the user program that it has completed the classification and the location of the result.

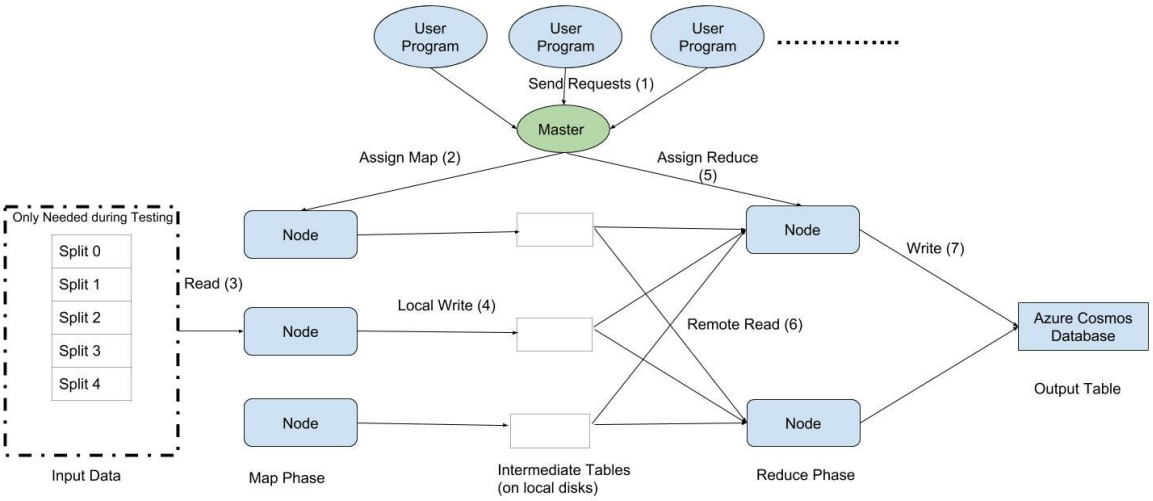

Figure 2 (Dean & Ghemawat, 2008)

API

User Programs:

- err <- AddData(data, location)
    - Upload local data to Azure Cosmos DB at the given location.
- resultLocation, err <- Classify(dataLocation, K)
    - Classify the data at dataLocation into K categories, return error when there is a network problem.

Master Server:

- resultLocation, err <- ReceiveRequest(dataLocation, K)
    - Receive user programs' request for data classification. Train a new K-means model if needed, and use the global model to classify the data at dataLocation into K categories.
- err <- Nodejoin(pubkey, signature)
    - Add new resource node with given public key into the network. Return error if the public key is not in the pre-authorized list, or the signature cannot be verified using the public key.

Resource Node:

- err <- SartTrainingMap(K, prevMeans)
    - Apply map function for training phase to local training data. K and prevMeans (means calculated from last iteration) are the inputs to the map function. Return error if data dimensions and prevMeans dimensions do not match.
- curMeans <- StartTrainingReduce(keys, listIP)
    - Contact other resource nodes in listIP (list of IP) to retrieve the key-value pairs corresponding to keys (list of keys the resource node is responsible for). Apply the reduce function for training phase to the retrieved data and calculate new means. Return the new means.
- resultLocation, err <- StartClassify(dataLocation, curMeans, keys)
    - Retrieve key-value pairs corresponding to the given keys from dataLocation. Apply map function for classifying phase to the retrieved data, which takes curMeans as the input. Store the classification result to Azure Cosmos DB and return that location.

Security and Data Privacy

We assume that there is a process to check nodes' credibility in the real world. In order ensure the credibility of training data and accuracy of trained models, the nodes that wish to join the system have submitted a request and passed the verification process, so that they have their public keys included in the pre-authorized public key list kept on the server.

Our system naturally has a data privacy feature. In the map phase of training, the STAR topology implies that nodes in the network are not directly connected with each other, and since the server never sends information about data, the nodes stay oblivious to raw data on other

nodes. And in the reduce phase, the nodes are only informed of paths to processed data, thus the raw data privacy is still preserved.

Node Joining the Network

Server has a list of pre-authorized public keys on its local disk. When a resource node joins the network, it uses its private key to sign a message and send the resulting signature along with its public key to the server.

The server will check if the public key is in the pre-authorized list. If it does, the server verifies the signature using the public key. If it passes the verification, the server broadcasts the node information to rest of the network so that others can find and communicate with the node. The network will only allow verified and broadcasted nodes by the server to join.

If there are nodes joining the network during the training phase, the model will be automatically retrained since there is more data to improve the accuracy of the model.

Fault Tolerance

- **Training**
  During the training phase, if the resource node fails, it fails with its data on the local storage. Since missing some training data in the training phase will only cause data loss and no further consequences, we won't recover any node failures during the map phase.

  During the reduce phase, if a resource node fails, server assign tasks to resource nodes that have already completed their own tasks.

- **Classifying**
  If a resource node fails, server assign tasks to resource nodes that have already completed their own tasks.

Load Balancing

In the beginning, loads will be equally distributed to all the nodes. Then, we will split the task based on the previous performances. On the issue of previous performance, we can measure the number of key-value pairs each worker is handling, as well as the time it takes for them to finish. Using this, the server regularly modifies the workload to nodes to give faster machines more work.

Backup Task

As we recognize a possible scenario where some nodes may take unusually long to process the tasks and slow down the overall process, hence significantly prolonging the overall execution time, we use backup tasks to preempt these stragglers. When the overall progress reaches a predetermined stage (e.g. 80%), the server reassigns the tasks yet to be finished to now idle

nodes. That is, these tasks in progress are potentially being executed by multiple nodes simultaneously at this stage, and the results returned first are adopted by the server. The server then notifies other nodes working on the same tasks to terminate the corresponding processes.

Testing the Correctness of the System

Once the server generates the random mean, we print it out to the console. We calculate K mean in one single machine with all data from all resource nodes with identical initialization. The global means and classification should be same as the distributed system result.

Use of Azure

There are two benefits of using Azure Cosmos DB for data storage. The first benefit is that Azure supports different query languages, which provides flexibilities for nodes to join and use our system. The second benefit is that since Azure Cosmos DB is a globally distributed database, our network can take advantage of the its locations and availability to reduce our network load and latency because the heavy data transferring will happen on the Azure distributed database system network instead of our system.

SWOT

Strengths	**Weaknesses**
● Two people did well in machine learning and parallel computing course. ● We know each other well	● Majority of team didn't take machine learning. ● All group members began learning Go this school year.
Opportunities ● Go to office hours to prove our concepts to the professor or TAs ● Enough research papers of MapReduce to be successful in doing the project	**Threats** ● People have to multitask among jobs, classes, and life ● Unpredictable grading scheme and demo requirements

Timeline:

Mar 2 : Project proposal drafts (group)
Mar 9 : Final project proposals (group)
Mar 16 :

- Calculate K-means on a single machine locally for verifying the correctness of our system output values
- Decide the query language used for the database, and get familiar with it (Yukang Zhang, Kailun Hong)
- Implement Map and Reduce Function, make sure each resource node can calculate k-means locally (Zixuan Yin, HangYee Wong)
- Set up connection to Azure Cosmos DB (HangYee Wong)
- Set up connection between Resource node, Master node, implement heartbeat protocol for maintaining states (HangYee Wong)
- Implement Master Server (Kailun Hong, Yukang Zhang, ZIxuan Yin)
    - Track the state of the resource nodes (idle, working on map, working on reduce, dead)
    - Receive requests from user program
    - Be able to assign tasks and synchronize between nodes.

Mar 23 :

- Send email to an assigned TA to schedule a meeting to discuss project status.
- Implement the whole training and classifying process (QiuShan Zhao, Yukang Zhang)
- Implement Fault Tolerance (QiuShan Zhao)

Apr 1:

- Design the formula used for Load Balancing, and split the workload base on the designed formula (Zixuan Yin)
- Display the result of classification as a graph (Kailun Hong)
- Detect straggler and start Backup task (Qiushan Zhao)
- Handle security issue (Kailun Hong)

Apr 6 : Project code and final reports (group)
Apr 8 : Practice the demo (group)
Apr 9-20 : Project demo (group)

Sources

Dean, J., & Ghemawat, S. (2008). *MapReduce: Simplified data processing on large clusters*. New York: ACM.10.1145/1327452.1327492

Trevino, A. (2016). Introduction to K-means Clustering. Retrieved March 08, 2018, from https://www.datascience.com/blog/k-means-clustering