Project Report

Railway Segment Fault Analysis using Parallel K-means

CSE4001- Parallel & Distributed Computing

Submitted by

16BCI0041- Kritika Mishra

16BCE2124- Sai Raghavendra V.

Under the guidance of

Prof. Saira Banu

Bachelor of Technology
in
Computer Science and Engineering with specialization in Information Security

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

October 2018

Contents

ABSRACT:

To explore the data processing of high-speed railway fault signal diagnosis based on MapReduce algorithm, the partitioning of the data set has been improved and implemented using K- means clustering algorithm. After which SON algorithm and Markov models have been applied to get better and improved results. In MapReduce parallelization process, the data partition matrix Tk was stored in line segmentation, the computing load was distributed in every node of cluster, and the time consumption of mobile data matrix and the consumption of partitioned matrix were calculated. Results show that the algorithm proposed could reduce the amount of computation in the execution process, greatly reduce the memory space consumption, and improve the counting speed in railway signal system.

INTRODUCTION:

With the further speed increase of China's high-speed railway and the continuous improvement of the railway information system, the conditions of collecting more railway running information are now available. At present, the running high-speed railway train, through the deployment of a large number of sensors, collects a variety of data. However, the traditional vibration data feature extraction and analysis technology is running on a single machine. This kind of technology, in the mass vibration data acquired by sensors, exposed the shortcomings of long processing time, various artificial intervention, and poor capability of processing big data file and so on. The emergence of cloud computing technology provides a way of thinking to solve the above problems. Map Reduce is an effective parallel computing framework of processing big data, which is one of the main models of cloud computing, and can automatically assign tasks and realize task balance. The working principle, operating mechanism and fault tolerance mechanism of Map Reduce calculation model are studied. In addition, combined with the characteristics of association rule generation algorithm, the traditional parallel algorithm is improved and the parallel optimization

scheme of association rules algorithm based on Map Reduce is proposed. Moreover, the improved algorithm is used in the railway quality analysis and evaluation industry.

The program can realize its distributed functions simply by Map function and Reduce function programming and deploying their own procedures to the cluster. In recent years, many researchers have proposed an improved algorithm for the shortcomings of parallel algorithms in practical applications. Hashem proposed an Apriori parallel improved algorithm introducing indexing structure. The algorithm improves the Apriori algorithm, and through MapReduce mechanism, conducts block processing of data. It also increases the data index in each data block, so as to enhance the performance of the algorithm. In the implementation process, only the part of the data object affected is updated. Although the improved algorithm can effectively enhance the efficiency, there is still the problem of low precision of mining.

RELATED WORKS:

According to [1] s in China train control system level 3 (CTCS-3), the control data transfer delay should be no larger than 500ms with greater than 99% probability. Coverage of both non-redundant networks and intercross redundant networks and cases of single Mobile terminals (MTs) and redundant MTs on one train are considered, and the corresponding vehicle-ground communication models, delay models, and fault models are constructed. The simulation results confirm that the transfer delay can meet the standard requirements under all cases. In particular, the probability is greater than 99.996% for redundant MTs and networks, and the standard of transfer delay in CTCS-3 will be improved inevitably.

In paper [2] a typical complex, multi-objective and nonlinear system is discussed. In this study, fuzzy predictive control technology is used to provide high quality control conditions for train operation, which provides great potential for the control of

complex system. It is difficult to find the accurate mathematical model and the optimal solution. First, the basic structure and function of train automatic control system are introduced, especially the coordination between automatic train operation (ATO) subsystem and other subsystems. Then, the basic principles of fuzzy logic and predictive control are introduced, and various forms of fuzzy logic [3] and predictive control are analyzed. The application and simulation of fuzzy predictive control in ATO system are deeply studied. Fuzzy predictive control for speed following system of ATO is designed. The fuzzy predictive control technology is compared with the conventional control technology. The simulation results show that the performance of train safety, comfort, parking accuracy and other performance indicators have been improved significantly by using fuzzy predictive controller.

In paper [4] they have discusses how By abstracting away the complexity of distributed systems, large-scale data processing platforms—MapReduce, Hadoop, Spark, Dryad, etc.—have provided developers with simple means for harnessing the power of the cloud. In this paper, they ask whether they can automatically synthesize MapReduce-style distributed programs from input–output examples. The ultimate goal is to enable end users to specify large-scale data analyses through the simple interface of examples.Thus a new algorithm and tool has been presented for synthesizing programs composed of efficient data-parallel operations that can execute on cloud computing infrastructure. The tool has been evaluated on a range of real-world big-data analysis tasks and general computations.

In paper [5] the authors have discussed how the existing parallel mining algorithms for frequent itemsets lack a mechanism that enables automatic parallelization, load balancing, data distribution, and fault tolerance on large clusters. As a solution to this problem, they designed a parallel frequent itemsets mining algorithm called FiDoop using the MapReduce programming model. To achieve compressed storage and avoid building conditional pattern bases, FiDoop incorporates the frequent items

ultrametric tree, rather than conventional FP trees. In FiDoop, three MapReduce jobs are implemented to complete the mining task. In the crucial third MapReduce job, the mappers independently decompose itemsets, the reducers perform combination operations by constructing small ultrametric trees, and the actual mining of these trees separately. They show that FiDoop on the cluster is sensitive to data distribution and dimensions, because itemsets with different lengths have different decomposition and construction costs. To improve FiDoop's performance, they developed a workload balance metric to measure load balance across the cluster's computing nodes. Extensive experiments using real-world celestial spectral data demonstrated that the proposed solution is efficient and scalable.

METHODOLOGY:

At present, the running high-speed railway train, through thedeployment of a large number of sensors, collects a variety of data. However, the traditional vibration data featureextraction and analysis technology is running on a singlemachine. This kind of technology, in the mass vibration data acquired by sensors, exposed the shortcomings of long processing time, various artificial intervention, and poor capability of processing big data file and so on. Map Reduce is an effective parallel computing framework of processing big data, which is one of the main models of cloud computing, and can automatically assign tasks and realize task balance.

Since, the amount of data collected every day is very large we need to segregate the data to collect the relevant information and discard the rest. In this project we have implemented a method of segregating the data into clusters based on their distance from the centroids using Eucledian distance, also known as K-means clustering algorithm. By applying this algorithm the large dataset has been segregated into 4 clusters and then we can find the difference between the different clusters along with the final output. This will help us in collecting important information about the clusters based on their centroid values. It also makes searching for any particular

information easier. After the segregation is done, a person can just look for the nearest centroid to the data he is looking for, after which thedata can besearched for in the cluster of that particular centroid only. Therefore, instead of searching in 4 clusters a person has to search only in 1 cluster which reduces the work by 75%. Therefore, this method is an efficient way of partitioning data clusters and then searching for relevant information as per the user's need, while discarding the irrelevant extra data.

DISCUSSION:

We have used k-means algorithm in our project to segregate the dataset into 4 clusters. K- means clustering algorithm is composed of 3 steps:

Step 1: Initialization

The first thing k-means does, is **randomly** choose K examples (data points) from the dataset as initial centroids and that's simply because it does not know yet where thecenter of each cluster is. (a centroid is thecenter of a cluster).

Step 2: Cluster Assignment

Then, all the data points that are theclosest (similar) to a centroid will create a cluster. If we're using theEuclideandistance between data points and every centroid, a straight line is drawn between two centroids, then a perpendicularbisector divides this line into two clusters.

Step 3: Move the centroid

Now, we have new clusters, that needcenters. A centroid's new value is going to be themean of all theexamples in a cluster.

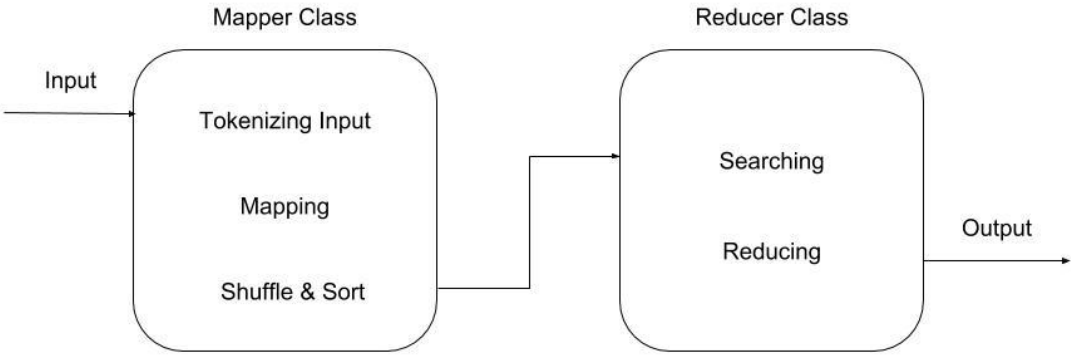
We'll keep repeating step 2 and 3 until the centroids stop moving, in other words, K-means algorithm is converged.

ALGORITHM:

TheMapReduce algorithm contains two important tasks, namely Map and Reduce.

- The map task is done by means of Mapper Class
- The reduce task is done by means of Reducer Class.

Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiplesystems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

These mathematical algorithms may include the following −

- Sorting

  Sorting is one of the basic MapReduce algorithms to process and analyse data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from themapper by their keys.

- Searching

  Searching plays an important role in MapReduce algorithm. It helps in thecombiner phase (optional) and in theReducer phase. Let us try to understand how Searching works with the help of anexample.

- Indexing

  Normally indexing is used to point to a particular data and its address. It performs batch indexing on the input files for a particular Mapper.

  Theindexing technique that is normally used in MapReduceis known as **invertedindex.** Search engines likeGoogle and Bing useinvertedindexing technique. Let us try to understand how Indexing works with the help of a simpleexample.

- TF-IDF

  TF-IDF is a text processing algorithm which is short for Term Frequency − InverseDocumentFrequency. It is one of the common web analysis algorithms. Here, the term 'frequency' refers to the number of times a term appears in a document.

MAPPER CLASS

TheMapperclass defines the Map job. Maps input key-value pairs to a set of intermediate key-value pairs. Maps are the individual tasks that transform the input records into intermediate records. Thetransformedintermediate records need not be

of the same type as the input records. A given input pair may map to zero or many output pairs.

REDUCER CLASS

TheReducerclass defines the Reduce job in MapReduce. It reduces a set of intermediate values that share a key to a smaller set of values. Reducerimplementations can access the Configuration for a job via theJobContext.getConfiguration() method. AReducer has three primary phases − Shuffle, Sort, and Reduce.

- **Shuffle** − TheReducercopies the sorted output from each Mapper using HTTP across thenetwork.
- **Sort** − The framework merge-sorts theReducer inputs by keys (since different Mappers may have output the same key). Theshuffle and sort phases occur simultaneously, i.e., while outputs are being fetched, they aremerged.
- **Reduce** − In this phase the reduce (Object, Iterable, Context) method is called for each <key, (collection of values)> in the sorted inputs.

Generally MapReduce paradigm is based on sending map-reduce programs to computers where the actual data resides.

- During a MapReduce job, Hadoop sends Map and Reduce tasks to appropriate servers in the cluster.
- The framework manages all the details of data-passing like issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on the nodes with data on local disks that reduces thenetwork traffic.
- After completing a given task, the cluster collects and reduces the data to form an appropriateresult, and sends it back to the Hadoop server.

KMeans

Randomly choose k examples as centroids

Whiletrue:

    Create k clusters by assigning each example to closest centroid

    Compute k new centroids by averagingexamples in each cluster

    If centroids don’t change:

        Break

K-means is a fast and efficient method, because thecomplexity of one iteration is k*n*d where k (number of clusters), n (number of examples), and d (time of computing theEuclidiandistance between 2 points).

We try different values of k, weevaluatethem and we choose the best k value using the following algorithm:

```
Best = kMeans(points);
For t in range(numTrials):
    C= kMeans(points);
    if dissimilarity(C) < dissimilarity(best):
        best = C;
return best
```

Dissimilarity(C) is the sum of all thevariabilities of k clusters

Variability is the sum of all Euclideandistances between the centroid and each example in the cluster.

We have used Euclediandistance to calculate thedistance between every node and the centroid of every cluster to segregate the data points into different clusters.

Euclediandistance function can be defined as:

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Where x and y are two vectors

CODE:

```
import time
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MiniBatchKMeans, KMeans
from sklearn.metrics.pairwise import pairwise_distances_argmin
from sklearn.datasets.samples_generator import make_blobs
import pandas as pd
mydata= pd.read_csv("/home/sai/Downloads/red.csv")
np.random.seed(0)
batch_size = 45
centers = [[1, 1], [-1, -1], [1, -1], [-1,1]]
n_clusters = len(centers)
X, labels_true = make_blobs(n_samples=3000, centers=centers, cluster_std=0.7)
# Compute clustering with Means
k_means = KMeans(init='k-means++', n_clusters=4, n_init=10, verbose=1)
t0 = time.time()
k_means.fit(X)
t_batch = time.time() - t0
# Compute clustering with ParallelKMeans
mbk = MiniBatchKMeans(init='k-means++', n_clusters=4, batch_size=batch_size,
                      n_init=10, max_no_improvement=10, verbose=1)
t0 = time.time()
mbk.fit(X)
t_mini_batch = time.time() - t0
# Plot result
fig = plt.figure(figsize=(8, 3))
```

```
fig.subplots_adjust(left=0.02, right=0.98, bottom=0.05, top=0.9)
colors = ['#4EACC5', '#FF9C34', '#4E9A06','#000000']
# We want to have the same colors for the same cluster from the
# ParallelKMeans and the KMeans algorithm. Let's pair the cluster centers per
# closest one.
k_means_cluster_centers = np.sort(k_means.cluster_centers_, axis=0)
mbk_means_cluster_centers = np.sort(mbk.cluster_centers_, axis=0)
k_means_labels = pairwise_distances_argmin(X, k_means_cluster_centers)
mbk_means_labels = pairwise_distances_argmin(X, mbk_means_cluster_centers)
order = pairwise_distances_argmin(k_means_cluster_centers,
                                  mbk_means_cluster_centers)
# KMeans
ax = fig.add_subplot(1, 3, 1)
for k, col in zip(range(n_clusters), colors):
    my_members = k_means_labels == k
    cluster_center = k_means_cluster_centers[k]
    ax.plot(X[my_members, 0], X[my_members, 1], 'w',
            markerfacecolor=col, marker='.')
    ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
            markeredgecolor='k', markersize=6)
ax.set_title('KMeans')
ax.set_xticks(())
ax.set_yticks(())
plt.text(-3.5, 1.8,  'train time: %.2fs\ninertia: %f' % (
    t_batch, k_means.inertia_))
# ParallelKMeans
ax = fig.add_subplot(1, 3, 2)
for k, col in zip(range(n_clusters), colors):
    my_members = mbk_means_labels == order[k]
    cluster_center = mbk_means_cluster_centers[order[k]]
    ax.plot(X[my_members, 0], X[my_members, 1], 'w',
            markerfacecolor=col, marker='.')
    ax.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
            markeredgecolor='k', markersize=6)
ax.set_title('Parallelized KMeans')
ax.set_xticks(())
ax.set_yticks(())
plt.text(-3.5, 1.8, 'train time: %.2fs\ninertia: %f' %
         (t_mini_batch, mbk.inertia_))
# Initialise the different array to all False
different = (mbk_means_labels == 4)
ax = fig.add_subplot(1, 3, 3)
for k in range(n_clusters):
    different += ((k_means_labels == k) != (mbk_means_labels == order[k]))
identic = np.logical_not(different)
ax.plot(X[identic, 0], X[identic, 1], 'w',
        markerfacecolor='#bbbbbb', marker='.')
ax.plot(X[different, 0], X[different, 1], 'w',
```

```
        markerfacecolor='m', marker='.')
ax.set_title('Difference')
ax.set_xticks(())
ax.set_yticks(())
plt.suptitle("Showing the Execution time for KMeans and Parallel KMeans")
plt.show()
import boto3

bucketName = "cse4001"
Key = "/home/sai/Downloads/red.csv"
outPutname = "screenshot"

s3 = boto3.client('s3')
s3.upload_file(Key,bucketName,outPutname)
import botocore

Bucket = "cse4001"
Key = "/home/sai/Pictures/a.png"
outPutname = "screenshot"

s3 = boto3.resource('s3')
try:
    s3.Bucket(Bucket).download_file(Key,outPutName)
except botocore.exceptions.ClientError as e:
    if e.response['Error']['Code'] == "404":
        print("The object does not exist.")
    else:
        raise
```

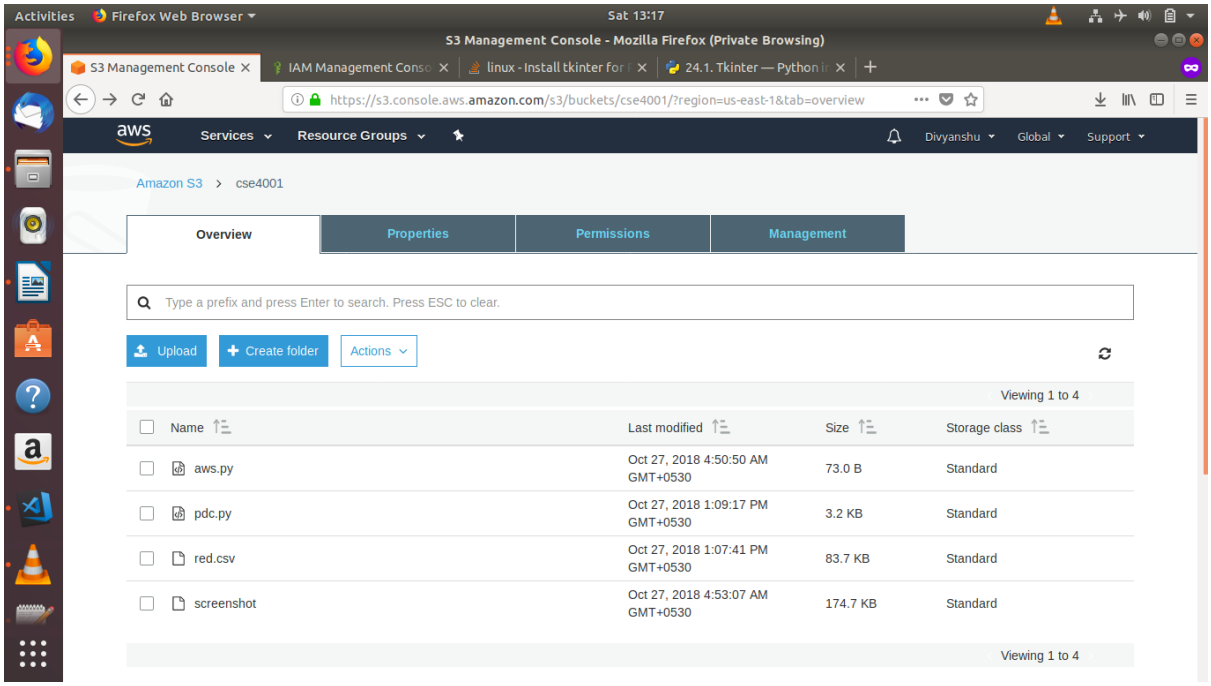
```
import boto3

s3 = boto3.client('s3')
s3.create_bucket(Bucket='cse4001')
```

RESULTS:

After applying k-means algorithm using python we get the following results:

We havedivided the dataset into 4 clusters here using k-means algorithm.



Eucledian distance has been used as a measure for finding the distance between the nodes and the clusters. As a part of our research we also ties to implement theproject using Cosine Similarity function. However, wefailed to do so because we found out that this particular problem cannot besolved using cosine similarity. This is because cosine similarity cannot be used as a metric for measuringdistance when themagnitude of the vector between two points matter. Since, in our case the magnitude is the important factor that decides thedistance from the clusters, which therebyhelps in clustering; we have implemented our project by using K- means algorithm using Euclidean distance.

CONCLUSION:

With the improvement of information level of railway system, data growth rate is fast. People's demand for using data to create value is also increasing. The value of data mining in daily business activities and social management is also being improved. Data flows are characterized by high speed, continuity and boundlessness. If there are too many different elements or requirement for processing multiple data flows at once, it is impossible to store all information of data flows in memory.

Therefore, a parallel processing algorithm is proposed for fault diagnosis of railway signal systems in this paper. Reasonable data counting strategy is made by using MapReduce, and the data flow is conducted with batch processing. As a result, it ensures that when the block data implements a large number of tests and update operations, it can only estimate the distributed separate elements, reduce the amount of calculation in the process of algorithm implementation, greatly reduce the memory consumption, and improve the counting efficiency and accuracy. The test of different data sets shows that the algorithm proposed is superior to the general algorithm in the statistical calculation of the frequent item sets of big data flows.

REFERENCES:

[1] Y. Cao, L.C. Ma, S. Xiao, et al., Standard analysis for transfer delay in CTCS-3, Chin. J. Electron. 26 (5) (2017) 1057–1063.

[2] Y. Cao, L.C. Ma, Y.Z. Zhang, Application of fuzzy predictive control technology in automatic train operation, Clust. Comput. (2018). http://dx.doi.org/10.1007/ s10586-018-2258-0.

[3] S. Xiao, W. Li, T. Shang, Fuzzy logic based high speed data transmission algorithm of sensor networks for target tracking, J. Intell. Fuzzy Systems 33 (5) (2017) 2887–2893.

[4] C. Smith, A. Albarghouthi, MapReduce program synthesis, ACM Sigplan Not. 51 (6) (2016) 326–340.

[9] Y. Xun, J. Zhang, X. Qin, Parallel mining of frequent itemsets using MapReduce, IEEE Trans. Syst. Man Cybern.: Syst. 46 (3) (2016) 313–325.