STUDY OF DATA MINING ALGORITHM IN CLOUD COMPUTING USING MAPREDUCE FRAMEWORK[1]

Vignesh V.[1], Dr. K Krishnamoorthy[2]

[1]Research Scholar, Dept of Computer Science & Engineering, Sai Nath University, Ranchi

[2] Professor, Sudarshan Engineering College, Tamil Nādu

ABSTRACT

Today's Cloud computing technology has been emerged to manage large data sets efficiently and due torapid growth of data, large scale data processing is becoming a major point of information technique. The Hadoop Distributed File System (HDFS) is designed for reliable storage of very large data sets and to stream those data sets at high bandwidth to user applications. In a large cluster, hundreds of servers both host directly attached storage and execute user application tasks. By distributing storage and computation across many servers, the resource can grow on demand while remaining economical at every size. MapReduce has been widely used for large-scale data analysis in the Cloud. Hadoop is an open source implementation of MapReduce which can achieve better performance with the allocation of more compute nodes from the cloud to speed up computation; however, this approach of "renting more nodes" isn't cost effective in a pay-as-you-go environment.

Keywords: Cloud Computing, Distributed Data Mining, Hadoop, Hadoop Distributed File System, Map Reduce

INTRODUCTION

These days large amount of data is created every day so with this rapid explosion of data we are moving towards the terabytes to petabytes. This trend creates the demand for the advancement in data collection and storing technology. Hence there is a growing need to run data mining algorithm on very large data sets. Cloud computing is a new business model containing pool of resources constituting large number of computers. It distributes the computation task to its pool of resources so that applications can obtain variety of software services on demand. Another feature of cloud computing is that it provides unlimited storage and computing power which leads us to mine mass amount of data.

Hadoop is the software framework for writing applications that rapidly process large amount of data in parallel on large clusters of compute nodes. It provides a distributed file system and a

[1] How to cite the article:
Vignesh V., Krishnamoorthy K., Study of Data Mining Algorithm in Cloud Computing Using Mapreduce Framework, *International Journal of Advances in Engineering Research*, April 2013, Vol 3, Issue 4, 33-44

framework for the analysis and transformation of very large data sets using the MapReduce

paradigm. The volume of data, collectively called data-sets, generated by the application is very large. So, there is a need of processing large data-sets efficiently.

MapReduce is a generic execution engine that parallelizes computation over a large cluster of machines. An important characteristic of Hadoop is the partitioning of data and computation across many hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers.

Big data has been used to convey the all sorts of concepts, including huge quantities of data (with respect to volume, velocity, and variety), social media analytics, next generation data management capabilities, real-time data, and much more. Now organizations are starting to understand and explore how to process and analyze a vast array of information in new ways.

Data mining is the process of finding correlations or patterns among fields in large data sets and building up the knowledge-base, based on the given constraints. The overall goal of data mining is to extract knowledge from an existing dataset and transform it into a human understandable structure for further use. This process is often referred to as Knowledge Discovery in data sets (KDD). It encompasses data storage and access, scaling algorithms to very large data sets and interpreting results. The data cleansing and data access process included in data warehousing facilitate the KDD process.

Based on the increasing demand for parallel computing environment of cloud and parallel mining algorithm,we study different mining algorithms. Association rule based algorithm, Apriori algorithm, is improved in order to combine it with the MapReduce programming model of cloud and mine large amount of data. With emerging trends in Cloud Computing, datamining enters a new era, which can have a new implementation. We can use cloud computingtechniques with Data mining to reach high capacity and efficiency by using parallel computational nature of the cloud. As MapReduce provides good parallelism for the computation,it's very suitable for us to implement data mining system based on MapReduce.

In a distributed computing environment bunch of loosely coupled processing nodes are connected by the network. Each node contributes into the execution or distribution / replication of data. It is referred as a cluster of nodes. There are various methods of setting up a cluster, one of which is usually referred to as cluster framework. Such frameworks enforce the setting up processing and replication nodes for data. Examples are Aneka and Apache Hadoop (also called Map / Reduce). The other methods involve setting up of cluster nodes on ad-hoc basis and not being bound by a rigid framework. Such methods just involve a set of API calls basically

for remote method invocation (RMI) as a part of inter-process communication.

The method of setting up a cluster depends upon the data densities and up on the scenarios listed below:

1. The data is generated at various locations and needs to be accessed locally most of the time for processing.
2. The data and processing is distributed to the machines in the cluster to reduce the impact of any particular machine being overloaded that damages its processing.

RELATED WORK

Distributed Data Mining in Peer-to-Peer Networks (P2P) [1] offers an overview of the distributed data-mining applications and algorithms for peer-to-peer environments. It describes both exact and approximate distributed data-mining algorithms that work in a decentralized manner. It illustrates these approaches for the problem of computing and monitoring clusters in the data residing at the different nodes of a peer-to-peer network. This paper focuses on an emerging branch of distributed data mining called peer-to-peer data mining. It also offers a sample of exact and approximate P2P algorithms for clustering in such distributed environments.

Architecture for data mining in distributed environments [2] describes system architecture for scalable and portable distributed data mining applications. This approach presents a document metaphor called \emph{Living Documents} for accessing and searching for digital documents in modern distributed information systems. The paper describes a corpus linguistic analysis of large text corpora based on collocations with the aim of extracting semantic relations from unstructured text.

Distributed Data Mining of Large Classifier Ensembles [3] presents a new classifier combination strategy that scales up efficiently and achieves both high predictive accuracy and tractability of problems with high complexity. It induces a global model by learning from the averages of the local classifiers output. The effective combination of large number of classifiers is achieved this way.

Map-Reduce for Machine Learning on Multi core [4] discuss the ways to develop a broadly applicable parallel programming paradigm that is applicable to different learning algorithms. By taking advantage of the summation form in a map-reduce framework, this paper tries to parallelize a wide range of machine learning algorithms and achieve a significant speedup on a dual processor cores.

STUDY OF DATA MINING ALGORITHMS

A. K-Means Clustering

The K-mean clustering algorithm [7] is used to cluster the huge dataset into smaller cluster.

In data mining, k-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. The problem is computationally difficult (NP-hard), however there are efficient heuristic algorithms that are commonly employed and converge fast to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both algorithms. Additionally, they both use cluster centers to model the data, however k-means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes. The most common algorithm uses an iterative refinement technique. Due to its ubiquity it is often called the k-means algorithm; which is also referred to as Lloyd's algorithm, particularly in the computer science community.

data **1. Algorithm:** Finding Frequent Itemset by using Apriori mining algorithm*:*

Given an initial set of k means m1 (1)… mk (1),

The algorithm proceeds by alternating between two steps:

1. **Assignment step***:* Assign each observation to the cluster with the closest mean.
2. **Update step**: Calculate the new means to be the Centroid of the observations in the cluster.

In the beginning we determine number of cluster K and we assume the centroid or center of these clusters. We can take any random objects as the initial centroids or the first K objects in sequence can also serve as the initial centroids.

Then the K means algorithm will do the three steps below until convergence. Iterate until stable:

1. Determine the centroid coordinate
2. Determine the distance of each object to the centroids
3. Group the object based on minimum distance

2. Euclidean distance:

In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. By using this formula as distance, Euclidean space (or even any inner product space) becomes a metric space. The Euclidean distance between point's p and q is the length of the line segment connecting them. In Cartesian coordinates, if p = (p1, p2, ... pn) and q = (q1, q2... qn) are two points in Euclidean n-space, then the distance from p to q or from q to p is given by:

$$d(p,q) = d(q,p) = \sqrt{(q1-p1)^2 + (q2-p2)^2 + \cdots\cdots + (qn-pn)^2}$$

B. Apriori

Apriori [7] is one of the key algorithms to generate frequent itemsets. Analysing frequent itemset is a crucial step in analysing structured data and in finding association relationship between items. This stands as an elementary foundation to supervised learning.

Association– It aims to extract interesting correlations, frequent patterns associations or casual structures among sets of items in the transaction databases or other data repositories and describes association relationship among different attributes.

Require: Items I = {i1, i2, . . . , in}, dataset D, user-defined support threshold
Ensure: F(D, _) := Frequent sets from D w.r.t. that particular threshold
1: C1 := {{i}| i 2 I} //Start with singleton sets 2: k := 1
3: while Ck 6= {} do 4: //Pruning Part
5: for all transactions (tid, I) 2 D do 6: for all candidate sets X 2 Ck do 7: if X _ I then
8: support(X) + + 9: end if
10: end for
11: end for //Computes the supports of all candidate sets
12: Fk := {X|support(X) _ _} //Extracts all frequent sets
13: //Generating Part
14: for all X, Y 2 Fk,X[j] = Y [j] for 1 _ j _ k − 1, and X[k] < Y [k] do
15: I = X [{Y [k]} //Join step 16: if 8J _ I, |J| = k : J 2 Fk then
17: Ck+1 := Ck+1 [I //Prune step 18: end if
19: end for 20: k + +
21: end while

In short we are trying to perform following steps:

1. Generate *Ck*+1, candidates of frequent itemset of size *k* +1, from the frequent itemsets of size *k*

2. Scan the database and calculate the support of each candidate of frequent itemsets.

3. Add those itemsets that satisfies the minimum support requirement to $Fk+1$.

The Apriori algorithm is shown above in line 13 generates $Ck+1$ from Fk in the following two step process:

a. Join step: Generate $RK+1$, the initial candidates of frequent itemsets of size $k + 1$ by taking the union of the two frequent itemsets of size k, Pk and Qk that have the first $k-1$ elements in common.

$RK+1 = Pk \cup Qk = \{i\ teml, \ldots, i\ temk-1, i\ temk , i\ temk\_ \}$
$Pk = \{i\ teml , i\ tem2, \ldots, i\ temk-1, i\ temk \}$ $Qk = \{i\ teml , i\ tem2, \ldots, i\ temk-1, i\ temk\_ \}$

where, $i\ teml < i\ tem2 < \cdots < i\ temk < i\ temk\_$.

b. Prune step: Check if all the itemsets of size k in $Rk+1$ are frequent and generate $Ck+1$ by removing those that do not pass this requirement from $Rk+1$. This is because any subset of size k of $Ck+1$ that is not frequent cannot be a subset of a frequent itemset of size $k + 1$.

Function subset in line 5 finds all the candidates of the frequent itemsets included in transaction t. Apriori, then, calculates frequency only for those candidates generated this way by scanning the database. It is evident that Apriori scans the database at most $k\max+1$ times when the maximum size of frequent itemsets is set at $k\max$.

The Apriori achieves good performance by reducing the size of candidate sets. However, in situations with very many frequent itemsets, large itemsets, or very low minimum support, it still suffers from the cost of generating a huge number of candidate sets.

RESEARCH METHODOLOGY

A. Cloud computing:

It consists of shared computing resources which are opposed to local servers or devices. Users [6] can pay on the basis of resource usage as timely basis. The major goal of cloud computing is to provide easily scalable access to computing resources and IT(InformationTechnology) services for achieving better performance. Cloud computing basically provides three different types of service based architectures are SaaS, PaaS, and IaaS.

i. **SaaS (Software as-a-service):** It offers application as a service on the internet.
ii. **PaaS (Platform as-a-service):** This is to be used by developers for developing new applications.
iii. **IaaS (Infrastructure as-a-service)**: It is basically deals by providers to provide featureson demand Utility.

Table1. Feature Comparison Of Commercial Offerings For Cloud Computing.

Properties	Amazon EC2	Google App Engine	Microsoft Azure	Manjras Aneka	
Service Type	IaaS	IaaS – Compute/ (web app-lications)	IaaS - PaaS Compute	PaaS Compute	
Support for (value added service provider	Yes	Yes	Yes	Yes	Web API
User Access	Command Line Tool, Web Portal	Command Line Tool, Custom	Azure Web APIs, Portal	Interface	
Virtualization	OS on Xen Container	Application Container	Service Container	Service Hypervisor	Container
(OS & runtime)	Linux, Windows	.NET on Windows	Platform Services	Linux, .NET on Mono	Window
Deployment model	Customizable	(Python, C++, VB)	Azure Applicatio VM	Web apps Java, Ruby	(C#,
If PaaS, ability to deploy on 3rd party	N.A.	No	No	Yes	IaaS

B. MapReduce:

MapReduce [5] is a programming model for processing large data sets, and the name of an implementation of the model by Google. MapReduce is typically used to perform distributed computing on clusters of computers. The model is inspired by map and reduces functionscommonly used in functional programming, although their purpose in the MapReduce framework is not the same as their original forms. MapReduce libraries havebeen written in many programming languages. A popular free implementation is Apache Hadoop.

MapReduce is a framework for processing the parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across

geographically and administratively distributed systems, and use more heterogeneous hardware). Computational processing can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce can take advantage of locality of data,

processing data on or near the storage assets to decrease transmission of data.

1) "Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. A worker node may do this again in turn, leading to a multi- level tree structure. The worker node processes the smaller problem, and passes the answer back to its master node.

2) "Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output – the answer to the problem it was originally trying to solve.

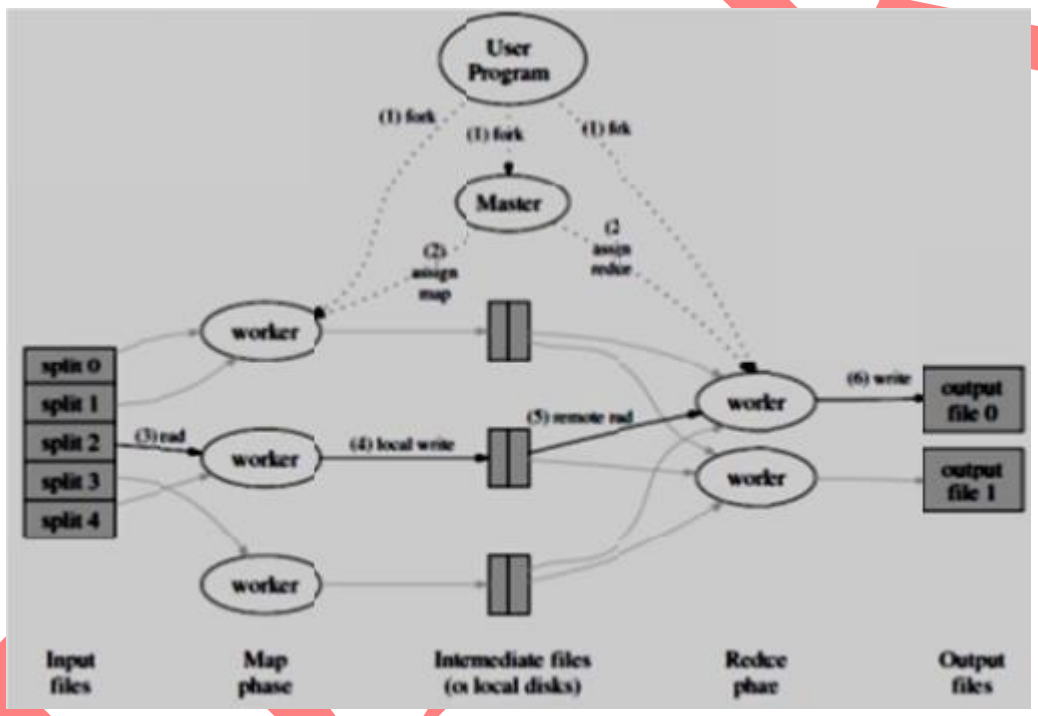

Fig. 1: EXECUTION OVER VIEW

Dataflow of the system is given below; the frozen part of the MapReduce framework is alarge distributed sort. The above figure consists of following parts:

a) Input reader: It divides the input into appropriate size (in practice typically 64 MB to 512 MB as per HDFS) and the framework assigns one split to one Map function. The input reader reads the data from stable storage (typically a in our case Hadoop distributed file system) and generates key/value pairs.

b) Map function: Each Map function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs. The input and output types of the map can be (and often are) different from each other.

c) Partition function: Each Map function output is allocated to a particular reducer by the application's partition function for sharing purposes. The partition function is given the key and the number of reducers and returns the index of desired r**e**duce.

d) Comparison function: The input for every Reduce is fetched from the machine where the Map run and sorted using the application's comparison function.

e) Reduce function: The framework calls the application's Reduce function for each unique key in the sorted order. It also iterate through the values that are associated with that key and produce zero or more outputs.

f) Output writer: It writes the output of the Reduce function to stable storage, usually a Hadoop distributed file system.

As an example, the illustrative problem of counting the average word length of every word occurrences in a large collection of documents in MapReduce is represented as following: The input key/value to the Map function is a document name, and its contents. The function scans through the document and emits each word plus the associated word length of the occurrences of that word in the document. Shuffling groups together occurrences of the same word in all documents, and passes them to the Reduce function. The Reduce function sums up all the word length for all occurrences. Then divide it by the count of that word and emits the word and its overall average word length of every word occurrences.

Example: Consider the problem of counting the average word length in a large collection of documents. The user would write code similar to the following pseudo-code:

function map(String key, String value): // key: document name
//value: document contents

```
    for each word w in value: EmitIntermediate(w, wordlength);

function reduce(String key, Iterator values) // key: word
// values: list of counts

    double sum = 0, count =0, result=0; for each v in values:
     sum += ParseInt(v); count++; result = sum / count;
    Emit(w, AsDouble(result));
```

Here, each document is split into words, and each word length is counted by the map function, using the word as the result key. The framework puts together all the pairs with the same key and feeds them to the same call to reduce, thus this function just needs to sum all of its input values to find the total appearances of that word. Then for finding average word length we divide the sum by the count of that word.

CONCLUSION

There are many new technologies emerging at a rapid rate, each with technological advancements and with the potential of making ease in use of technology. However one must be very careful to understand the limitations and security risks posed in utilizing these technologies. Neither MapReduce-like software, nor parallel databases are ideal solutions for data analysis in the cloud. Hybrid solution that combines the fault tolerance, heterogeneous cluster, and ease of use out-of- the-box capabilities of MapReduce with the efficiency, performance, and tool plug ability of shared-nothing parallel systems could have a significant impact on the cloud market. We will work on bringing together ideas from MapReduce and data mining algorithms, also to combine the advantages of MapReduce-like software with the efficiency and shared work advantages that come with loading data and creating performance enhancing data structures.

REFERENCES

[1] Souptik Datta, Kanishka Bhaduri, Chris Giannella, Ran Wolff, and Hillol Kargupta, (July 2006), Distributed Data Mining in Peer-to-Peer Networks, University of Maryland, Baltimore County, Baltimore, MD, USA, *Journal IEEE Internet Computing archive* Volume 10 Issue 4, Pages 18 – 26.

[2] Mafruz Zaman Ashrafi, David Taniar, and Kate A. Smith, (2007); A Data Mining Architecturefor Distributed Environments, pages 27-34, *Springer-Verlag* London, UK.

[3] Grigorios Tsoumakas and Ioannis Vlahavas, (11-12 April 2008), Distributed Data Mining of Large Classifier Ensembles

SETN-2008, *Thessaloniki, Greece, Proceedings, Companion* Volume, pp. 249-256

[4] Cheng-Tao Chu et. al., (2006), *Map-Reduce for Machine Learning on Multicore*, CS Department, Stanford University, Stanford, CA.

[5] Jeffrey Dean and Sanjay Ghemawat, (2004); Mapreduce: Simplified data processing on large clusters. In *OSDI*, pages 137-150.

[6] Daniel J. Abadi, (2009); Yale University, *Data Management in the Cloud: Limitations and Opportunities, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*

[7] *Top 10 algorithms in data mining*, © Springer-Verlag London Limited 2007

[8] James l. Johnson, (2009); SQL in the Clouds, *IEEE journal Cloud Computing*.