An Efficient Parallel Processing Technique For Large Text Files

B. Rama Jyothi, Dr. A. Krishna Mohan
Student, Department of Computer Science
H.O.D of Computer Science Department
UCEK, Kakinada
Andhra Pradesh, India
ramajyothi.bathula@gmail.com
krishna.ankala@gmail.com

Abstract–Now a days the data was increasing rapidly because of heterogeneous resources. The data that is useful for large organisations will required large number of computing resources for processing large data sets. there are so many well known languages and platforms are available for processing big data sets but requires large amount computing time and large number of CPU cycles. It is very difficult to organisations to spend large clock cycles on large data sets. In this paper by using Map Reduce algorithm we will count the number of words in large text file of book reviews so by analysing these words we conclude that whether the book is preferable for learners or professors.

Index Terms– Data characteristics, challenging issues, data processing, Hadoop map reduce

1. Introduction

Now a days the people is having well knowledge on social sites and search engines. The organisations stores and process large data sets. The data comes from various resources day to day. If you want process that large data it requires large number of computing resources and also time consuming.

Social networking sites like twitter, facebook data are increasing day by day because of large number of users. The servers are stores large amount of information regarding various fields like science, organisational, social networks etc. the cost is high for storing and retrieving huge data from servers. There are so many platforms and tools are developed for processing huge amount of data. Several parallel processing techniques are applied for processing they give good results in less time.

There are so many applications like Filekr pictures sharing site. Suppose when the number of images are increased the storage size also increased so it requires more cost for processing.

The above examples are tell us how the data was increased. Hence to process the data it requires more efficient parallel processing techniques to handle that data. Hadoop is one of the best platform to process large data sets that are coming from various resources.

The data is having some noise data or raw data hence to mine that data by applying data mining techniques. The redundant and unwanted data will be removed after applying the data mining algorithms on huge data sets.

Hadoop is a parallel efficient parallel processing technique for examine the large data sets in less computing resources. The data is having heterogeneous characteristics because of multiple resources.

The remaining paper will tells about the characteristics and challenging issues faced by the huge amount of data sets and section will discuss related work to know the data processing with parallel processing technique by using Hadoop platform.

2. Large Data Characteristics

Huge data starts with heterogeneous resources, autonomous sources and explore the knowledge on huge data to extract complex and evolving relationships between large data sets.

Fig: Large data characteristics

2.1 Huge Data with heterogeneous and dissimilar dimensionality

It is one of the important characteristics of huge Volume of data sets. In real world the data coming from different resources so it is having heterogeneous features. For example if we take DNA reports of the human being that report is having different types of molecules, the heterogeneous dimensionality is also exists between each molecule. different organisations
will take different schemas to process these DNA molecules then the heterogeneous and dissimilar dimensionality place a crucial role.

2.2 Independent sources with decentralised control
The large volumes of data is generated from independent sources. The search engines like google, Flickr generates large volumes of data the velocity of data flow is independent to the another server. The processing speed of server is depends on the design architecture of the system. The facebook, Twitter generates huge volumes of data because of large number of users on social sites. The processing speed of data is depends on the server..

2.3 Data complexity and evolving relationships
The data comes from different resources is having different dimensionality. the relation between data sets is depends on the flow and volume of the data. For example if we consider organisation it is having so many number of working people having various names and different genders and they get different income depending on the position of the job. so if we want to process that huge volumes of complex data it requires complex architecture platform. The patterns are discovered from large data after mining the relationships between the data sets.

3. Huge data Challenging issues
The data are coming from various resource so there may be a chance of getting redundant data or irrelevant data. so we apply some data mining algorithms to mine the huge data sets. we get the relevant patterns from the large data sets.

The multiple sources generates large volumes of data, but there may be a chance of unauthorized access of that data from different sources. To provide security we have to apply some cryptographic algorithms on large data sets. It is used to process the data by removing threats that are placed by attackers.

The large data sets are beyond the capacity of personal computers. They requires highly computational platform to mine the data sets and it requires parallel processing techniques like map reduce. then the results be displayed as useful patterns for us after removing unwanted and noise data from huge data sets.

3.2 Data Semantics and Application knowledge
The data semantics are nothing but the user knowledge, policies, regulations of the particular data sets.

The two major issues are:

3.2.1 Data privacy and information sharing
The data generally comes from heterogeneous resources, then there may be a chance of privacy issue comes because many number of users access that data sets.In case of information sharing the unauthorised user is present in between the communication and stole our useful patterns. So the security is major issue of the large data sets.

Suppose one system is sharing the data to the another system there are some situations where we have to hide the architectural designs of the server system from the client system. So the data sharing with privacy feature is important for large data sets.

3.2.2 Domain and Application Knowledge
Domain and application knowledge will help us to extract the correct features from the large data sets and also helpful for design of data mining algorithms for processing of the large data sets. for example if we consider the stock market data the data will updated from time to time. Then to analyse that data we requires domain knowledge like how the behaviour of the purchasing goods will change depending on the customers interest.

4. Experimental Results
4.1 platform
Formally speaking, Hadoop is an open source framework for writing and running distributed applications that process large amounts of data. Distributed computing is a wide and varied field, but the key distinctions of Hadoop are that it is Accessible—Hadoop runs on large clusters of commodity machines or on cloud computing services such as Amazon’s Elastic Compute Cloud (EC2)
It should be flexible and powerful enough to handle most of the targeted data processing applications. MapReduce uses lists and (key/value) pairs as its main data primitives. The keys and values are often integers or strings but can also be dummy values to be ignored or complex object types. The map and reduce functions must obey the following constraint on the types of keys and values. In the MapReduce framework you write applications by specifying the mapper and reducer. Let’s look at the complete data flow:

The input to your application must be structured as a list of (key/value) pairs, list(<k1, v1>). This input format may seem open-ended but is often quite simple in practice. The input format for processing multiple files is usually list(<String filename, String file_content>). The input format for processing one large file, such as a log file, is list(<Integer line_number, The list of (key/value) pairs is broken up and each individual (key/value) pair, <k1, v1>, is processed by calling the map function of the mapper. In practice, the key k1 is often ignored by the mapper. The mapper transforms each <k1, v1> pair into a list of <k2, v2> pairs.

Note that the (key/value) pairs are processed in arbitrary order. The transformation must be self-contained in that its output is dependent only on one single (key/value) pair. For word counting, our mapper takes <String filename, String file_content> and promptly ignores filename. It can output a list of <String word, Integer count> but can be even simpler. As we know the counts will be aggregated in a later stage, we can output a list of <String word, Integer count> with repeated entries and let the complete aggregation be done later. The output of all the mappers are (conceptually) aggregated into one giant list of <k2, v2> pairs. All pairs sharing the same k2 are grouped together into a new (key/value) pair, <k2, list(v2)>. The framework asks the reducer to process each one of these aggregated (key/value) pairs individually.

4.2 Mapreduce algorithm

The programmer writes her data processing task as processing primitives in the form of either a producer or a consumer. The timing of their execution is managed by the system. Similarly, MapReduce is also a data processing model. Its greatest advantage is the easy scaling of data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into mappers and reducers is sometimes nontrivial. But, once you write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers.

MapReduce programs are executed in two main phases, called mapping and reducing. Each phase is defined by a data processing function, and these functions are called mapper and reducer, respectively. In the mapping phase, MapReduce takes the input data and feeds each data element to the mapper. In the reducing phase, the reducer processes all the outputs from the mapper and arrives at a final result. In simple terms, the mapper is meant to filter and transform the input into something that the reducer can aggregate over. You may see a striking similarity here with the two phases we had to develop in scaling up word counting. The similarity is not accidental. This two-phase design pattern was seen in scaling many programs, and became the basis of the framework.
Hence by observing the words after running the mapreduce algorithm we say that this book is preferable for beginners and as well learners. This will done in less number of clock cycles comparing to other platforms for large text files.

6. References


