

Analyzing User Behavior Using MapReduce

Vikhyat Gupta

Department of Computer Science
University of Bridgeport
Bridgeport, USA
vigupta@my.bridgeport.edu

Tarik El Taeib

Department of Computer Science
University of Bridgeport
Bridgeport, USA
teltaeib@my.bridgeport.edu

Abstract— Big Data involves large volume of data, which cannot be processed using traditional tools. These data sets are complex and evolves continuously in real time. The challenge is to analyze and extract useful knowledge from this volume of data.

To process this large amount of data we will use a parallel programming tool, MapReduce, which can process and generate large data sets.

There are many areas which can increase their profits by catering to the user needs. Hence, we need some techniques to analyze the user behavior and accordingly generate some patterns. As the number of users in every sector of the market are exponentially increasing, resulting in the increase of the volume of the log files. These files are usually not structured and must be analyzed in a reasonable amount of time. In this paper we will propose some general ideas by which we can gather user behavior and how that information can be used to generate further information specific to a user. We will use apache Hadoop as our framework that will implement MapReduce to analyze these patterns in an acceptable time. Based on these patterns, we can suggest the user of the other related patterns.

Keywords— Big data, Data mining, Hadoop, MapReduce, user behavior analysis

I. INTRODUCTION

First we will look at the scenario describing big data, its various complexities; it's continuously evolving nature and see how normal computing processes will be insufficient to extract knowledge in least possible time. Also, we will see how parallel processing paradigms are so efficient with big data.

Researches show that, 90% of the data has been generated in the last two years itself. The average data generated every day is in quintillions of volume. This user data usually comes when a user surfs the internet, logs in to an account, buys stuff (which further generates more data) and so on, This data is usually unstructured may be in the form of log files, images, emails and other data. This amount of data cannot be processed by the traditional tools. Further, we need algorithms to extract useful information from this data. Hence, we need some framework that does this data analysis efficiently in real time.

Now this data is usually distributed (that is, not centralized), complex and keeps on evolving. For

instance, a user may keep on updating their information and the organization must keep pace with this data evolution in order to serve the user in an appropriate manner. Fig 1 depicts this scenario, where there are four blind persons and they are trying to communicate information to each other in order to form a collaborative information.[1]

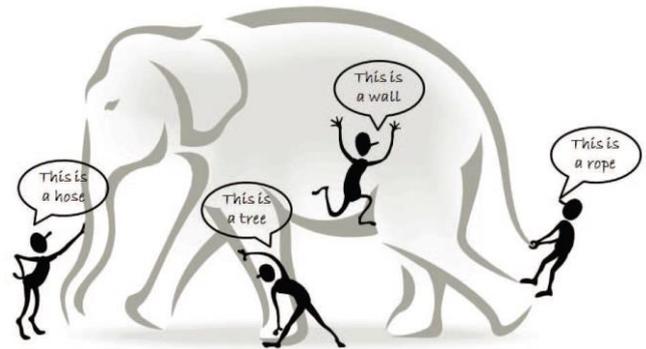


Fig.1. The blind men and the giant elephant: the localized (limited) view of each blind man leads to a biased conclusion. [1]

Here, each user may perceive information as they see it. One may perceive the leg as a pole, one may perceive the tail as a rope and so on. This is due to the fact that each person is working in their local region. Also, there could be a situation in which the elephant is not stable and is growing. This would lead to more complexities. In general, data is also collected through various heterogeneous sources and is subject to various complexities.

Why distributed data is preferred?

- If centralized data is attacked by any virus, then the complete data is at risk.
- Distributed data need not depend on any central system to gather information.
- Each system can generate their own data relative to the location in which they are situated.

II. WORKING ON BIG DATA

MapReduce is a paradigm without any actual source code and its design patterns are platform independent. A number of programming models have been developed to implement MapReduce. We will use Hadoop as our model to implement MapReduce. [2]

The *map* and the *reduce* tasks run in a distributed fashion on a cluster of machines to enable parallel processing. *Map* task generally loads, parses and

filters data. Its output are intermediate keys and values. The *reduce* task handles a subset of the map task output by merging the intermediate values. This paradigm lends high scalability for data processing. [3]

As we know, MapReduce works in two phases, map and reduce. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. [4]

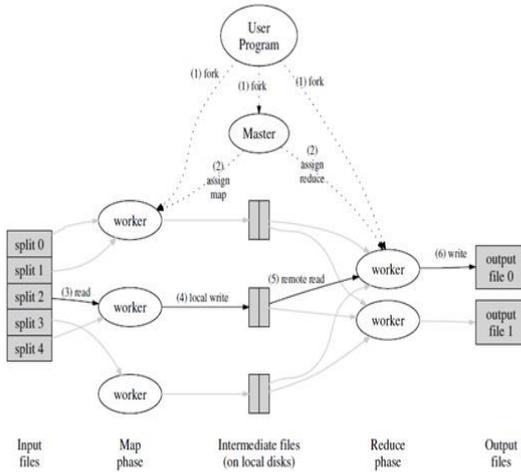


Fig.2. Execution overview.[4]

We can use MapReduce paradigm for prediction analysis. Given a large unstructured database, we can do the following from the perspective of prediction-

- We can see number of occurrence of a particular item and predict further information according to a more specialized constraint.

Example- Suppose we have a Toy database, we can see the demand of each toy according to different age group or according to different categories of toys. Hence, we can maintain stock accordingly.

- If we have a user login for a particular application that also maintains a large database, we can see the interests of the user by monitoring the frequency of the user's visit to a particular item. Hence, we can give predictions to the user according to the data generated.

Example- Suppose a user looks for cell phones on a shopping website. If the user frequently looks for a cell phone of a particular brand, we can suggest phones from the same brand; further we can give suggestions for the same price range for which the user is constantly looking for.

III. THE IDEA

This paper gives a general idea as to how we can use MapReduce paradigm to an unstructured database to extract useful information. It further discusses how we can use this extracted information.

To understand it, let us look at an example-

```
product/productId: B000LQOCHO
review/userId: ABXLMMWJIXXAIN
review/profileName: Natalia Corres "Natalia Corres"
review/helpfulness: 1/1
review/score: 4.0
review/time: 1219017600
review/summary: "Delight" says it all
review/text: This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin with nuts - in this case Filberts. And it is cut into tiny squares and then liberally coated with powdered sugar. And it is a tiny mouthful of heaven. Not too chewy, and very flavorful. I highly recommend this yummy treat. If you are familiar with the story of C.S. Lewis' "The Lion, The Witch, and The Wardrobe" - this is the treat that seduces Edmund into selling out his Brother and Sisters to the Witch.

product/productId: B000UA0QID
review/userId: A395BORC6FGVXV
review/profileName: Karl
review/helpfulness: 3/3
review/score: 2.0
review/time: 1307923200
review/summary: Cough Medicine
review/text: If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in addition to the Root Beer Extract I ordered (which was good) and made some cherry soda. The flavor is very medicinal.
```

Fig.3. A fraction of Food Review Database

Food review database represented in Fig.3., is an unstructured database, hence we cannot run SQL commands on it to extract information to our needs. Hence, we use Hadoop and put the database over the HDFS. Its scalability is improved by adding clusters to it and thus making it more distributed.

The food company can judge the user demands by analyzing their top rated and bottom rated products and hence keep the stock accordingly.

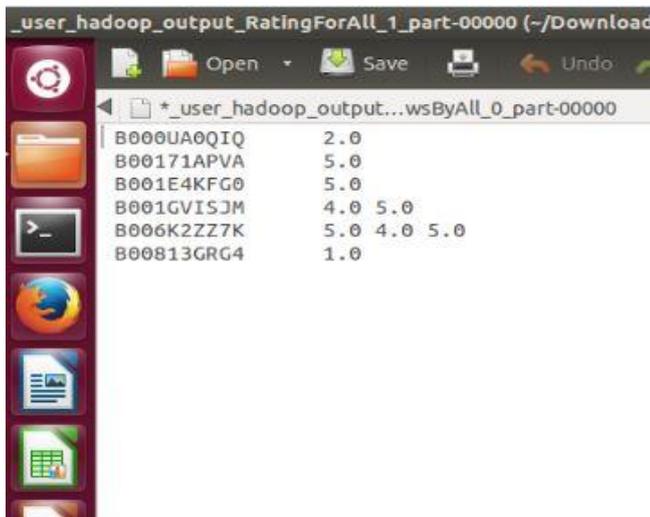
Product ID	Review Count
B00002N8SM	38
B00002NCJC	2
B00002Z754	2
B00004CI84	189
B00004CXX9	195
B00004RAMS	30
B00004RAMV	9
B00004RAMX	25
B00004RAMY	172
B00004RBDU	95
B00004RBDW	31
B00004RBDZ	59
B00004RYGX	189
B00004S1C5	18
B00004S1C6	61
B000052Y74	5
B00005344V	17
B0000535LF	1
B0000537KC	16
B00005C2JE	1
B00005C2M2	50
B00005C2M3	7
B00005IX96	16
B00005IX97	16
B00005IX98	16
B00005IX99	4

Fig.4. Number of reviews for all the products

We can get all the products corresponding to a user and see what products they use the most –*user behavior*. Based on this information we can suggest the user of some similar products.

The above scenario is possible if have a static copy of a database. In this case, other than this we can cluster the database and find number of reviews for a particular product. This helps to get a rough estimate as to which product is the most reviewed -

Fig.4. We can also see all the scores given by a user –Fig.5.



User ID	Score
B000UA0QIQ	2.0
B00171APVA	5.0
B001E4KFG0	5.0
B001GVISJM	4.0 5.0
B006K2ZZ7K	5.0 4.0 5.0
B00813GRG4	1.0

Fig.5. Scores by a user

We can perform many such tasks. All of this extraction of information and can be used in one way or the other. Having such information is beneficial for an organization as it helps them to function accordingly.

If the database is dynamic and is continuously evolving, we can analyze user behavior by any of the two ways-

- Users can login and then access their accounts. This would help maintain their activity.
- A session can be maintained. As soon as a user enters a website, a session is maintained and all his activities are stored for a session.

Applying any of the methods, we can cater to a dynamically evolving database in real time.

This is a general idea as to how we can use a database to extract information and further use that information.

Use of parallel processing mechanism is essential. The term Big data corresponds to a data that is voluminous. Parallel processing tools works in a distributed fashion and computes the result simultaneously. If some other mechanism is employed, computing this large data will be cumbersome and the results generated will take longer than such mechanisms.

In the above example, MapReduce effectively makes chunks of the database and distributes them to all the clusters. The clusters then processes the data and generate their individual results. These results are aggregated by the master machine. Hence, the amount of time is divided by the number of clusters. These clusters also provide high fault tolerance. The data is duplicated on other clusters in case a node experiences a failure.

IV. RELATED WORK

The paper "MapReduce: Simplified Data Processing on Large Clusters" by Jeffrey Dean and Sanjay Ghemawat [2] first discussed how google used MapReduce on such large volume of data. This paper helped follow many variations in MapReduce to process data. Kim et al. [5] used MapReduce to analyze user behavior in order to maximize profits of IPTV providers. Akdogan et al. [6] discussed the problem of processing parallel geospatial query using MapReduce. Tsai et al. [7] proposed how cloud services can be improved by Mapreduce to define a better service replication. Wei et al. [8] further explored the aspects of MapReduce by proposing SecureMR that assures service integrity.

V. CONCLUSION

MapReduce is fairly easy to implement as most of the work is taken care by the framework itself. Task scheduling, job scheduling, failure handling are all handled by Hadoop. MapReduce model transforms the data into sets of keys and values. As MapReduce employs batch processing model, it gives result in an ad hoc manner. RDBMS is good when data size to be processed is in gigabytes, whereas Hadoop can process data upto petabytes. Hadoop efficiently extracts knowledge from unstructured database which is possible through RDBMS.

This paper discussed how we can use the extracted information. First, we saw that the data today is not only growing, but is also complex and continuously evolving. Then we saw the need to use parallel programming tools like Hadoop to process this big data. Lastly, the rest of the paper discussed how we can analyze user behavior from the data we extract using MapReduce and use it to further make suggestions to user. These suggestions help the user to make more informed decisions. It also helps the organizations to maximize their profits by exploiting user behavior to their use.

REFERENCES

- [1] Xindong Wu; Xingquan Zhu; Gong-Qing Wu; Wei Ding, "Data mining with big data," *Knowledge and Data Engineering, IEEE Transactions on*, vol.26, no.1, pp.97,107,Jan.2014
- [2] O'Reilly – *Hadoop: The Definitive Guide* – By Tom White
- [3] O'Reilly – *MapReduce Design Patterns* – By Donald Miner and Adam Shook
- [4] Jeffrey Dean; Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", *USENIX OSDI*,2004.
- [5] Joohee Kim; Chankyoun Hwang; Eunkyoun Paik; Youngseok Lee, "Analysis of IPTV user behaviors with MapReduce," *Advanced Communication Technology (ICACT), 2012 14th International Conference on*, vol., no., pp.1199,1204, 19-22 Feb. 2012

[6] Akdogan, A.; Demiryurek, U.; Banaei-Kashani, F.; Shahabi, C., "Voronoi-Based Geospatial Query Processing with MapReduce," *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on* , vol., no., pp.9,16, Nov. 30 2010-Dec. 3 2010

[7] Tsai, W.-T.; Peide Zhong; Elston, J.; Xiaoying Bai; Yinong Chen, "Service Replication Strategies with

MapReduce in Clouds," *Autonomous Decentralized Systems (ISADS), 2011 10th International Symposium on* , vol., no., pp.381,388, 23-27 March 2011

[8] Wei Wei; Juan Du; Ting Yu; Xiaohui Gu, "SecureMR: A Service Integrity Assurance Framework for MapReduce," *Computer Security Applications Conference, 2009. ACSAC '09. Annual* , vol., no., pp.73,82, 7-11 Dec. 2009