A Framework for Event Log Analysis using Hadoop MapReduce

Shweta Thankur*, Prof. Sampada Vishwas Massey†
Dept. of Computer Science and Engineering, Shri Shankaracharya Group of Institutions
Bhilai, India 490024
Email: * Shwetathakur181291@gmail.com, † sampada.satav@gmail.com

Abstract

This Event log file is the most common data-sets exploited by many companies for customer behavior analysis. Oftentimes these records are unordered, and need to be grouped by certain key for effective analysis. One such example is to group similar user with different session ID to facilitate further analysis. This kind of analysis is known as User Sessionization. In this paper, we propose a distributed framework in combination of Hadoop and MapReduce to analyze event log file and sessionize user based on IP-address and timestamp.

Keywords**:** *User Sessionization, Hadoop, MapReduce, Event Log Files.*

I. INTRODUCTION

Companies like Flipkart, Snapdeal and Amazon routinely produces a huge amount of logs on a daily basis. They continually improve their operations and services by analyzing the data. Analyzing these huge amounts of data in a very short period of time is a crucial task for any business analyst.

The problem of log files analysis is complicated because of not only its volume but also its disparate structure [1][5]. In 2015, Hemant Hingave [2] compared the Hadoop MapReduce and RDBMS suggested that Hadoop MapReduce is capable of processing job faster. Hadoop MapReduce [6] used in many areas for processing big data. A large data-sets which are used to gain business intelligence has disparate structure. The datasets need to be group via some data points of a certain feature together for effective analysis. Such process is essential in job like User Sessionization [3].

Usually, the data-sets for User Sessionization are the event log files [10]. The event file has some attributes like the type of event, origin of the event, date and time of happened event. These events are periodically generated by the web server and can scale up to petabytes in size. In general, the data-sets are the collection of records. Each record consists of primary key, secondary key and a value. The records in the data-set are need not to be sorted by time-stamp. For such an input data-set, we define Re-Organization of User Session or Re-Organization, as a processing that generates an output which contains all the similar primary key at one location with sorted by secondary key. The Re-Organization task requires a large amount of data for processing. MapReduce programming model is used for processing the Re-Organization task. MapReduce allows focusing on business logic rather than dealing with the complexity of the Parallel computation.

In this paper, we propose a Re-Organization or Re-Organization of User Session for efficient analysis of event log files in distributed environment. Our proposed mechanism effectively sessionize the activity performed by the user based on primary key, secondary key and a value.

We have built a distributed framework which supports the Re-Organization mechanism by extending Hadoop [11]. Hadoop enables applications to work in a distributed environment. There may be thousands of distributed component working together to accomplish a single task.

II. PROBLEM DEFINITIONS

In this section, we will first describe the application that our framework targets. Then we present the Mathematical Model for MapReduce jobs.

A. Click Stream Analysis

Many companies such as Flipkart, Snapdeal, Amazon have web services and generates a large amount of event log files. These companies are interested in analyzing the click stream [13] generated by their website by the customer which provide very useful information. One can detect customer click pattern which can be used for the product recommendation, advertisement, promotion and personalization of service. The arrangement of the users based on their session is a very important task. Usually, a session consists of user behavior in that particular session. Various intelligence can be gathered from click stream patterns like:

- Killer Pages: The page in which user has no interest. Generally, a user spends less time and log-out after watching.
- Interesting Pages: The page in which user shows interest and spent more time on that particular page.

```
237.198.29.13 - - [13/July/2015:22:00:01 ] "GET /about HTTP/1.0" 200 2097
192.145.2.1 - - [13/July/2015:22:00:01 ] "GET /abc.gif HTTP/1.0" 200 778
234-122-12.21 - - [13/July/2015:22:00:02 ] "GET /login.jpg HTTP/1.0" 200 980
146.199.167.12 - - [13/July/2015:22:00:03 ] "GET /graph HTTP/1.0" 200 899
198.198.29.13 - - [13/July/2015:22:00:03 ] "GET /contact HTTP/1.0" 200 100
```

Fig. 1: Sampled click stream data

III. EXISTING MAPREDUCE SUPPORT OF RE-ORGANIZATION TASKS

The In this section, we discuss the implementation of ReOrganization tasks on Hadoop framework. We begin our discussion with an overview of Hadoop MapReduce programming model. Then we present existing mechanism for solving ReOrganization on MapReduce/Hadoop framework.

A. MapReduce/ Hadoop

The Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage [4].

Hadoop enables the application to work in a distributed environment. There may be thousands of distributed component working together to accomplish a single task. Generally, the huge log files are distributed over various clusters known as HDFS cluster [12] (Hadoop distributed file system). Hadoop breaks up the records into the number of blocks. These blocks are distributed over various clusters and processed in each system in a parallel fashion. The performance of the Hadoop system is gained by operating files in a parallel environment.

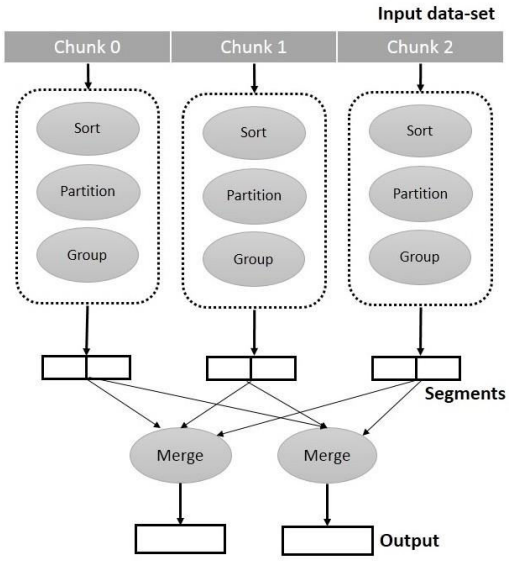

Fig. 2: Proposed MapReduce work flow

B. Existing MapReduce approach for Re-Organization tasks

The For input data-set $(Rec)_{in}$ the existing MapReduce approach is as follows. A map() function read each record one after another and transform primary key, secondary key and a value ($r_i = p_i, s_i, v_i$) into an intermediate key-value pair, where the key is IP-address and value is rest of the attributes. In general the output of map phase is a set of intermediate key-value pair $[p_i,(s_i, v_i)]$. But this approach has a problem that Hadoop sorts the values based on primary key (IP-address). Hence, sessionizing user clicks with this approach is not relevant.

Now, suppose reducer reads all of the keys and buffer it in its array data structure performing an in-reducer sort on the values. This method is not scalable because the reducer will receiving all the values for a key, and might it run out of memory. This method is only applicable to a small amount of data that will not cause an out of memory error.

IV. MAPREDUCE SOLUTION FOR RE-ORGANIZATION TASKS

Fig. 2 shows the proposed work flow of Hadoop/MapReduce jobs. Sometimes we have to sort the values coming into the reducer of Hadoop MapReduce job. We can indirectly sort the values by using the composite key. The Re-Organization tasks consist of 3 Phases: the sort phase, the partition phase, and the group phase.

Instead of emitting the IP-address as the key from the mapper, we need to emit a key which has multiple parts i.e. IP-address and time-stamp. So the intermediate key-value pair is IP-address, time-stamp as key and other attributes as value $[p_i, s_i, v_i]$. The similar IP-address are grouped together with the different unsorted time-stamp. The composite key comparator compares similar IP-address and sort time-stamp in order of event happened first. Now we are able to sort based on IP-address and time-stamp. The output data point consists of those records which have similar IP-address sorted by the timestamp. Finally, the reducer merges multiple segments from different mappers to produce a single output, where the records with similar IP-address are stored into one data point.

V. FRAMEWORK IMPLEMENTATION

In this section, we present the implementation of our Hadoop framework for efficiently executing Re-Organization tasks. Our proposed mechanism extends Hadoop to implement Sort, Partition and Group scheme.

A. Sorting Phase

To accomplish Re-Organization task, we need to handle the sort order of intermediate keys and order in which reducer process keys. First, we inject a value (event log records) into the composite key, and then we handle the sort order of intermediate keys. Here, IP-address is the natural key and combination of IP-address and timestamp is the composite key. Now, we add the timestamp field to the natural key for accomplishing secondary sort because we want the reducer value to be sorted by timestamp. Sorting of the IP-address and timestamp is done by overriding compareTo() method of Hadoop and WritableComparable interface is implemented for comparing custom data types (IP-address and timestamp).

B. Partitioning Phase

Based on the mapper output key-value pair, the partitioner class of Hadoop decides which mapper's output goes to which reducer. For this, we have incorporated two classes namely a custom partitioner and custom comparator. Custom partitioner class decides which reducer has to process which keys and custom comparator class sort the IP-address so that it groups the data when it arrives at reducer.

C. Grouping Phase

Place In the grouping phase, we incorporate comparator (IpTimestampGroupingComparator class) that groups the similar Ip-address for a single call to the reduce() function. Hadoop provides the various mechanism for implementing the grouping comparator code into the framework. One of the ways is to set the Grouping Comparator class (job.setGroupingComparatorClass).

VI. EVALUATION

We have performed experiments to evaluate the performance of Re-Organization mechanism on a distributed environment.

A. Experiment Setup

To evaluate our framework we use pseudo distributed mode. It consists of to 1 instances [8]. Instance has 1 CPU, 3.75 GB of RAM, 10 GB of Hard Disk Drive. Apache Web Server log [9] are used in our evaluation. We have taken the varying number of records to analyze.

B. Result

Apache web server log is used in our evaluation. The records are collected and place it in Hadoop distributed file system (HDFS) for processing. The records contain IP-address, timestamp, and other values. The IP-address and timestamp are unordered and need to be grouped by similar IP-address with increasing order of timestamp with 15 minutes of session timeout threshold. We run Re-Organization tasks with varying number of records. Our task user sessionization groups click performed by the user and divide the session of each user based on timeout threshold. (e.g. 15 min). Fig. 4 shows the user sessions which are grouped by the similar IP-address and sorted by the timestamp. The session ID is appended at the end of each record. Session ID represents the session for each successful login. If the same IP-address has different session ID then the user might be had logged out or reached the timeout threshold of 15min. User sessionization tasks help in figuring out the session duration of each user in all sessions and most interesting and killer page of each session.

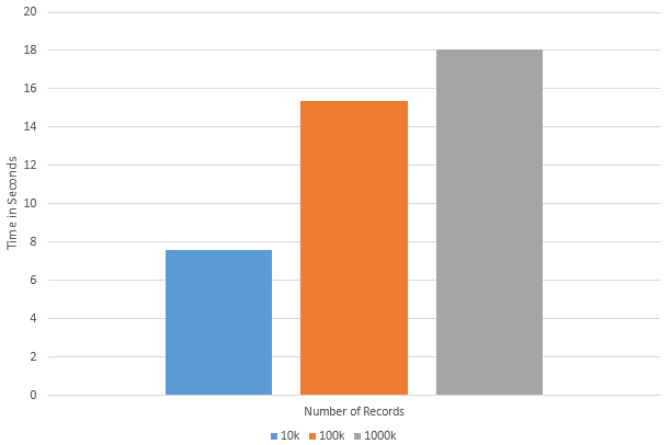

Fig. 3. Running time of jobs

```
192.145.2.1 - - [13/July/2015:14:33:15 ] "GET /about HTTP/1.0" 200 778 +7899000
192.145.2.1 - - [13/July/2015:14:33:18 ] "GET /product1 HTTP/1.0" 200 600 +7899000
192.145.2.1 - - [13/July/2015:14:55:18 ] "GET /login HTTP/1.0" 200 100 +8823400
198.198.29.13 - - [13/July/2015:15:00:03 ] "GET /contact HTTP/1.0" 200 456 +9099922
198.198.29.13 - - [13/July/2015:15:50:00 ] "GET /locate HTTP/1.0" 200 23 +1028780
198.198.29.13 - - [13/July/2015:15:55:03 ] "GET /pro2 HTTP/1.0" 200 45 +1028780
```

Fig. 4: Screenshot of user sessions

The User Sessionization task is tested over different nodes with varying size of files. Fig. 3 represents the running time of job as the number of nodes and records increases. We can see that the CPU time taken to perform map and reduce tasks decrease smoothly as the number of node increases.

VII. CONCLUSION

User Sessionization is an important task in event log file analysis. Hadoop MapReduce framework analyzes event log file and generate output as a collection of similar IP-address sorted by time-stamp. From the output generated company's can evaluate in which part of their website user is most and least interested. The company's

may also evaluate the click pattern and interest of the user and based on this they recommend products to their customer. We identified that the big data tool, Hadoop MapReduce can efficiently analyze ReOrganization tasks. The performance of Hadoop MapReduce was evaluated via experiments with real-world data-sets.

REFERENCES

[1] Sayalee Narkhede, Tripti Baraskar, "HMR log Analyzer: Analyze Web Application logs over Hadoop MapReduce", International Journal of UbiComp (IJU), Vol.4, No.3, July 2013

[2] Hemant Hingave, Prof. Rasika Ingle, "*An approach for MapReduce based Log analysis using Hadoop*", IEEE Sponsored 2'nd International Conference on Electronics and Communication Systems (ICECS'2015)

[3] B. Berendt, B. Mobasher, M. Nakagawa, M. Spiliopoulou,"*The impact of site structure and user environment on session reconstruction in web usage analysis* ", WEBKDD 2002 Mining Web Data for Discovering Usage Patterns and Profiles, 2002, pp. 159179

[4] " Apache Hadoop ", http://hadoop.apache.org/ [accessed March 06, 16]

[5] Bharat Parte, Umesh Jamdade, Pranita Sonavane, Sheetal Jadhav, "*SQUID Log Analyzer Using Hadoop Framework*", Journal of Innovative Research in Engineering & Management (IJIREM), Volume-2, Issue-1, January-2015

[6] J. Dean, S. Ghemawat, "*MapReduce: simplified data processing on large clusters*", The 6th Symposium on Operating Systems Design and Implementation, OSDI04, USENIX Association, 2004, pp. 107113

[7] " Amazon EC2 ", http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html [accessed March 06, 16]

[8] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C.Krintz, "*See spot run: using spot instances for mapreduce workflows*", USENIX Conference on Hot Topics in Cloud Computing (HotCloud), 2010.

[9] " Apache Web Log Format ", https://httpd.apache.org/docs/1.3/logs.html [accessed March 06, 16]

[10] R. Vaarandi, "*Mining event logs with SLCT and LogHound*", Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, 2010, pp. 1071-1074

[11] M. Bhandarkar, "*MapReduce programming with apache Hadoop*", IEEE International Symposium on Parallel & Distributed Processing (IPDPS), 2010, pp. 1

[12] J.Shafer, S.Rixner, A.L.Cox, "*The Hadoop distributed filesystem: Balancing portability and performance*", IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS), 2010, pp.122-133

[13] A.Surya, D.K.Sharma, "*A comparative analysis of clickstream as web page importance metric*", IEEE Conference on Information & Communication Technologies (ICT), 2013, pp. 776-781