POLİTEKNİK DERGİSİ

JOURNAL of POLYTECHNIC

ISSN: 1302-0900 (PRINT), ISSN: 2147-9429 (ONLINE)
URL: http://dergipark.org.tr/politeknik

Equi-depth histogram construction methodology for big data tools

Büyük veri araçları için eş derinlikli histogram oluşturma metodolojisi

***Yazar(lar) (Author(s))**: Tolga BUYUKTANIR[1], Ahmet Ercan TOPCU[2]*

ORCID[1]: 0000-0001-5317-0028

ORCID[2]: 0000-0003-1929-5358

Bu makaleye şu şekilde atıfta bulunabilirsiniz(To cite to this article): Büyüktanır T. ve Topcu A. E., "Equi-depth histogram construction methodology for big data tools", ***Politeknik Dergisi***, 23(3): 859-865, (2020).

Erişim linki (To link to this article): http://dergipark.org.tr/politeknik/archive

DOI: **10.2339/politeknik.620198**

Equi-Depth Histogram Construction Methodology for Big Data Tools

Highlights

- *Equi-depth histogram construction with quality guarantees for big data*
- *Creating histograms and elimination the complex implementation*
- *Applying new techniques to create histograms using big data*
- *The capability of writing multiple jobs including histogram construction using Apache Pig*

Graphical Abstract

The system provides the capability of writing multiple jobs using Apache Pig and eliminates the complex implementation of Map-Reduce jobs by creating histograms.

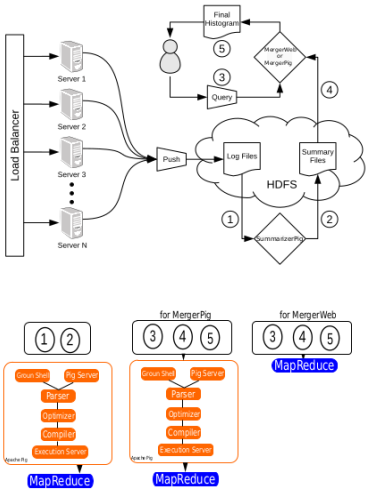

Figure. Flowchart of demonstrations

Aim

To build a fast and effective equi-depth histogram with quality guarantees for big data.

Design & Methodology

Approximate equi-depth histogram construction for big data and provide easy execution of data processing and data management.

Originality

Contribute the error-guaranteed equi-depth histogram construction method for big data tools by facilitating high-level languages.

Findings

A successful model is established to provide an easy execution of data processing and to create the histogram of unstructured big data.

Conclusion

The system provides the capability of writing multiple jobs using Apache Pig, and programmers can make use of the advantages of Apache Pig to create histograms and eliminate the complex implementation of Map-Reduce jobs.

Declaration of Ethical Standards

The author(s) of this article declare that the materials and methods used in this study do not require ethical committee permission and/or legal-special permission.

Equi-Depth Histogram Construction Methodology for Big Data Tools

Araştırma Makalesi / Research Article

Tolga BUYUKTANIR[1, 3*], Ahmet Ercan TOPCU[2, 3]
[1]Idea Technology Solutions, R&D Center, Turkey
[2]College of Engineering and Technology, American University of the Middle East, Kuwait
[3]Faculty of Engineering and Natural Sciences Department of Computer Engineering, Ankara Yildirim Beyazit University, Turkey

(Geliş/Received : 13.09.2019 ; Kabul/Accepted : 01.04.2020)

ABSTRACT

In recent decades, countless data sources such as social media, machines, and networks are constantly pushing data into the digital world. The size of the data has been growing exponentially. To understand the statistical information of data query optimization, equi-depth histograms are essential. In this paper, we present approximate equi-depth histogram construction for big data using both Apache Pig Scripts and Java Web Interface interacting with Apache Hadoop. We use equi-depth histogram construction with quality guarantees for big data approaches and implement them with Apache Hadoop Map-Reduce and Apache Pig user-defined functions. We introduce a prototype implementation of the construction of the approximate equi-depth histogram from the Java Server Face page using Apache Hadoop jobs and the Hadoop Distributed Files System, and we evaluate these methods using the demonstration. We explain Apache Pig Scripts techniques to create equi-depth histograms using big data. The results indicate that our system provides the capability of writing multiple jobs using Apache Pig, and programmers can make use of the advantages of Apache Pig to create histograms and eliminate the complex implementation of Map-Reduce jobs.

Keywords: Approximate histogram, merging histograms, big data, log files, hadoop distributed file system.

Büyük Veri Araçları için Eş-Derinlikli Histogram Oluşturma Metodolojisi

ÖZ

Son yıllarda, verileri sürekli olarak dijital dünyaya aktaran ağlar, makinalar ve sosyal medya gibi bir çok data kaynağı vardır. Bu kaynaklardan üretilen datanın boyutu eksponansiyel olarak artmaktadır. Hali hazırda elde bulunan datanın istatistiki bilgisini anlamak ve sorgu optimizasyonu sağlamak için eş-derinlikli histogram vazgeçilmez bir araçtır. Bu makalede, büyük veriler için hem Apache Pig betiklerini hem de Apache Hadoop ile etkileşimli Java Web Arayüzü kullanılarak yaklaşık eş-derinlikli histogram oluşturulması gösterilmektedir. Büyük veriler için, kalite garantisiyle birlikte, eş-derinlikli histogram oluşturma metotları kullanılmakta, bu metotların teknik yönlerden deneysel sunumları ortaya konulmakta ve yine bu metotlar Apache Hadoop Map-Reduce ve Apache Pig User Defined Functions ile uygulanmaktadır. Arka planda Apache Hadoop Map-Reduce işleri (Apache Hadoop Map-Reduce jobs) ve Hadoop Distributed Files System kullanılarak Java Server Face sayfasından kalitesi garantilenmiş eş-derinlikli histogram oluşturulmasının prototip uygulaması ve bu uygulamaların kullanılmasıyla metotların değerlendirilmesi sunulmaktadır. Ayrıca büyük verileri kullanarak eş-derinlikli histogram oluşturmak için Apache Pig betiklerinin teknikleri izah edilmektedir. Sonuçlar gösteriyor ki; sistemimiz Apache Pig kullanılarak, histogram kullanımını da gerektiren çoklu iş yazma yeteneğini basit bir şekilde sağlamaktadır. Programcılar, histogram oluşturmak ve Map-Reduce işlerinin karmaşık uygulamalarından kaçınmak için Apache Pig'in avantajlarından faydalanabilmektedir.

Anahtar Kelimeler: Yaklaşık histogram, histogram birleştirme, büyük veri, log dosyaları, hadoop distributed file system.

1. INTRODUCTION

The size of generated and stored data in industry is of terabytes or even petabytes and is growing bigger every day [1-3]. These data are obtained from machines (logs), social media, and medical and wearable devices. The processing of data and the deriving of meaningful results are crucial to improve the decision-making skills of the data owner. This means that there is a clear need for efficiently handling obtained datasets. There are several tools and technologies to meet this need, and almost all of them are based on the Map-Reduce paradigm, which is a type of distributed computing that splits input data into chunks and processes them independently. Apache Hadoop is the most popular big data tool that comes with the Map-Reduce framework and a distributed system called the Hadoop Distributed File System (HDFS). Apache Hadoop [5, 6] has standard methods to process enormous datasets in industry and academia [3, 7-11]. Other reasons for its popularity are its ease of use, open-source nature, failover, high performance, scalability, and reliability. Many researchers and academics have focused on improving Hadoop's Map-Reduce job performance in different high-level languages [3, 8, 11, 12], together with query optimization and indexing techniques [7, 13-18] and join algorithms and data structures [9, 19, 20]. In today's business environment,

**Sorumlu Yazar (Corresponding Author)*
e-posta : tolga.buyuktanir@ideateknoloji.com.tr

where needs and desires change rapidly, it is necessary to get results quickly from big data for big data analytics [9]. To obtain fast results from large datasets, sampling from datasets or computing previous results provides approximate results within the predefined rate of accuracy. These are the most efficient ways to solve problems. For example, consider big companies such as Google, Yahoo, or Wikipedia. The Google web engine processes over 40000 search queries every second on average, which translates to 3.5 billion searches per day [21]. The Yahoo website has one billion active users per month on average, which translates to nearly 40000 requests per second [22]. Wikipedia has a maximum of 30000 requests per second. Therefore, there are different data with different data sources, and they require different algorithmic approaches that take into consideration performance and accuracy criteria.

Histogram construction is one way to obtain statistical information from data, as histograms give information about data distribution and represent a summary of all of the data. Moreover, a histogram provides quick query optimization, approximate results, distribution fitting, and parallel data partitioning [23]. The most used histogram type is the equi-depth histogram, also known as a height-balanced histogram. In an equi-depth histogram generated by β buckets of data chunks containing N elements, every bucket contains N/β elements. Yildiz et al. [4] proposed a histogram merging algorithm with quality guarantees. According to the algorithm, they merged pre-calculated equi-depth histograms of partitioned data with user-specified error bounds. More specifically, pre-computed equi-depth histograms of data with T buckets are merged to build a β-buckets histogram with the desired error bound and quality guarantees. Apache Hadoop is an easily scalable distributed tool, but the solution to a big data problem requires multiple skills and tasks, and it may be necessary to write more than one Hadoop Map-Reduce job. This may be troublesome. Instead of writing multiple jobs, it makes sense to use a high-level language such as Apache Pig, which is an SQL-like scripting language [12, 24], which translates scripts into optimized sequences of Map-Reduce jobs. It also saves time for developers in terms of writing a lot of code and the trouble of Hadoop tuning. In this paper, we apply the histogram construction method proposed in [4] to Apache Pig and explain how to use the implementation of it in Hadoop Map-Reduce in web applications. Accordingly, we present two demonstrations. The first one is a web interface that merges equi-depth histograms and builds a new one using Apache Hadoop Map-Reduce jobs. The purpose of this first application is to show that the quality-guaranteed histogram generation method can be used in web tools. Moreover, it facilitates the connection of big data and enterprise applications. The second application is written in Pig Latin with the same goals because of its ease of use. In the s econd application, the quality-guaranteed approximate histogram generation method is presented as a contribution to the Apache Pig tool. The purpose of this application is to show that this generation method is easily implemented for big data applications by taking advantage of Apache Pig.

The rest of this paper is organized as follows: In Section 2, we define the histogram construction problem for big data. We explain our demonstrations with implementation details and give future works in Section 3, followed by conclusions in Section 4.

2. PROBLEM DEFINITION

Machine-generated data, also known as logs, are automatically produced by machines. To be informed of machine activity, logs are traditionally stored daily in W3C format [25]. When we consider the web servers of big systems, all servers write requests and responses data to log files. Some engineers may need to analyze these. For example, business intelligence specialists derive intelligence (top-rated pages, top users, gender of top users, time spent online). To respond to user demands, companies need to develop log management and analytical tools. A medium-sized Internet business may have thousands of web servers logging all activity. Moreover, there are ETL processes incrementally collecting, cleaning, and storing the logs in big data storage. The quantity of data to the ETL and to be analyzed is enormous and rapidly growing. Therefore, it is important to get as accurate intelligence from it as possible. In this regard, the equi-depth histogram is a quick and reliable way to understand data statistics. In this paper, we create two applications to create an on-demand equi-depth histogram of desired data within a time interval. Daily logs are kept on web servers. At the end of the day, all logs of this day are pushed to the HDFS. As soon as new log files are available in the HDFS, an exact or daily equi-depth histogram is constructed with the Summarizer and stored in the HDFS. If a user wants to construct an approximate equi-depth histogram of any time interval, such as the last week, Merger gets daily equi-depth histograms in a time interval as input, merges them, and creates an approximate equi-depth histogram with quality guarantees. A more detailed problem definition is available in [4].

3. OVERVIEW OF DEMONSTRATIONS

In this section, we will explain our demonstrations with implementation details. We have two demonstrations. One of them consists of two Pig Latin scripts, which are called SummarizerPig and MergerPig, respectively. SummarizerPig runs offline and is used to construct an exact equi-depth histogram of new data coming to the HDFS with a number of buckets pre-specified by the user. The output of SummarizerPig, which can be called the summary, is also stored in the HDFS. The other

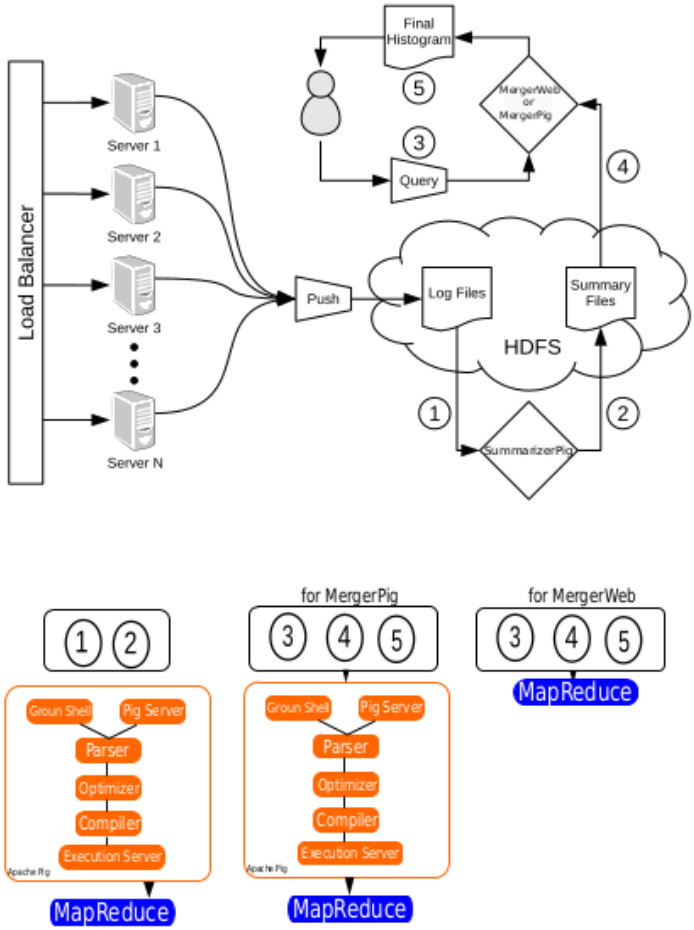

Figure 1. Flowchart of demonstrations.

script, MergerPig, runs online with the desired query request by the user. It merges SummarizerPig's desired outputs considering a specified query and builds the final approximate equi-depth histogram. The second demonstration is used for the same goals as MergerPig, but there are minor differences. We call this application MergerWeb because a Map-Reduce job works in the background of the Java Server Face (JSF) web page. In MergerWeb, the user prepares a query and sends it to the Map-Reduce job. The job gets inputs, which are summaries from the HDFS according to the query, and builds an approximate equi-depth histogram as MergerPig. This scenario is explained in Figure 1. In Figure 2, an overview of the demonstration processing is given. The online and offline parts can be seen on the right side of the figure. Raw data, summary files, and final approximate histograms constructed according to user demands stored in the HDFS are displayed on the left of the figure. When new data are pushed to the HDFS, SummarizerPig works, constructing its summary and storing it in the HDFS again. The constructed T-buckets equi-depth histogram is the exact histogram. Whenever a user requests an equi-depth histogram of desired data partitions, MergerPig gets summaries of related data partitions, computes the approximate β-buckets equi-depth histogram, and saves the output, which is the final merged histogram, to the HDFS. In this research, we have two Hadoop Map-Reduce job alternatives of SummarizerPig and MergerPig. The first job can be used instead of SummarizerPig, which works offline. The second job, which is an alternative of MergerPig, works on demand. In our work, we provide a web application to get queries from the user and run the Map-Reduce jobs using this query. As a result, it merges exact histograms and builds an approximate equi-depth histogram.

3.1. Bucket Number Calculations

In this research, we demonstrate the quality-guaranteed approximate equi-depth histogram construction by merging exact equi-depth histograms. The bucket number of the exact equi-depth histogram T is defined by the user. The bucket number of the approximate histogram β is constructed by merging exact histograms. This is also determined by the user. However, there is a relationship between T and β for the quality guarantee. The relationship between T and β is shown in Equation 1.

$$e_{max} < 2\beta / T \leq tolerable\ percentage\ error \quad (1)$$

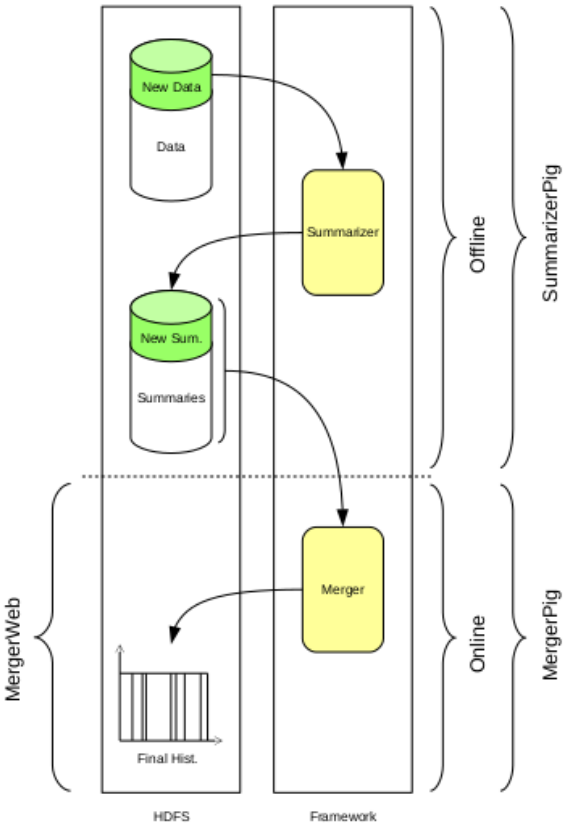

Figure 2. Overview of demonstrations.

For example, we assume that we have 10% tolerable error using Equation 1. We calculate the relationship between T and β as follows:

$$e_{max} < 2\beta / T \leq 0.1$$
$$20\,\beta \leq T$$

As a result, the required bucket size of exact histograms T should be at least 20 times more than the required bucket size of approximate histogram β. The Oracle database uses the number 254 as a histogram bucket size [26], so in this study, we also use 254. The T values are shown in Table 1 according to the desired β and the tolerable maximum percentage error using Equation 1.

Table 1. Sample β and T values according to the tolerable maximum percentage error rate.

e_{max} (%)	β	T
5	10	400
5	254	10160
10	5	100
10	254	5080
20	254	2540

3.2. Demonstration Examples

We now present our applications using hourly page view statistics data of Wikipedia in this section. We also tested our application on skewed data produced by the Gumbel distribution, and we described the operational logic of the demonstrations in Section 2. Here we explain our demonstration examples. In Table 2 and Table 3, the first application scripts are displayed. We implement the HistogramSummarizer user-defined function (UDF) and register it in line 1 of Table 2 to create an exact equi-depth histogram. The number of buckets shown as 'T' is defined in line 2 of Table 2. In the other part of the script, data are loaded, the histogram is generated, and the daily histogram is stored in the HDFS under the 'Summaries' directory.

MergerPig 2 merges the desired daily exact equi-depth histograms. To use this method in Apache Pig, we write another UDF registered in 1 of Table 3. As we can see in this script, the number of buckets of the approximate histogram is defined in line 2 of Table 3. The merged histogram data of the desired time interval (time interval is a week in the following example) are loaded in line 3 of the following table. In the rest of the script, the approximate equi-depth histogram is built and stored in the HDFS.

Table 2. Script of SummarizerPig.

```
register HistogramSummarize.jar;
define Hist HistogramSummarize('254');
A = load 'pagestat/20160101' using PigStorage(' ')
as (lang:chararray,url:chararray,req:long,byte:long);
B = foreach A generate byte as (byte:long);
C = group B all;
D = foreach C {
sorted = order B by byte;
generate Hist(sorted);
};
store D into 'Summaries/20160101';
```

Table 3. Script of MergerPig.

```
register HistogramEstimate.jar;
define Hist HistogramEstimate('254');
A = load 'Summaries/2016010[1-7]/*' as
(B: bag {T:
tuple(bound:long,numberOfTuple:long)});
B = foreach A generate flatten(B);
C = group B all;
D = foreach C{
sorted = order B by bound;
generate Hist(sorted);
};
store D into output;
```

We implement two Hadoop Map-Reduce jobs for the second demonstration. The first job constructs the exact equi-depth histogram as SummarizerPig. However, we

do not present it in Figure 2 because SummarizerPig has enough working logic to construct an exact equi-depth histogram. The second job works as MergerPig. The required command to run this job is shown in Table 4.

Table 4. Command of Hadoop histogram merger.

```
hadoop jar HistogramEstimate.jar
edu.tou.HistogramEstimate
$numberOfBuckets($\beta$)
$startdate $enddate $Summaries $output
```

uncompressed 60 GB and consists of language, pagename, pageviews, and pagesize columns. It includes the first 6 days of January 2016 and approximately 952686335 tuples. We operate pagesize for histogram constructions.

In order to obtain the comparison results, we run SummarizerPig and SummarizerWeb for the data of each day and we construct daily exact histograms. The Hadoop History Server was run at the same time while running the tests. In Figure 5, we display elapsed times (taken from the History Server) to construct a daily exact equi-depth histogram with SummarizerPig (blue line) and Apache Hadoop Summarizer (orange line). After

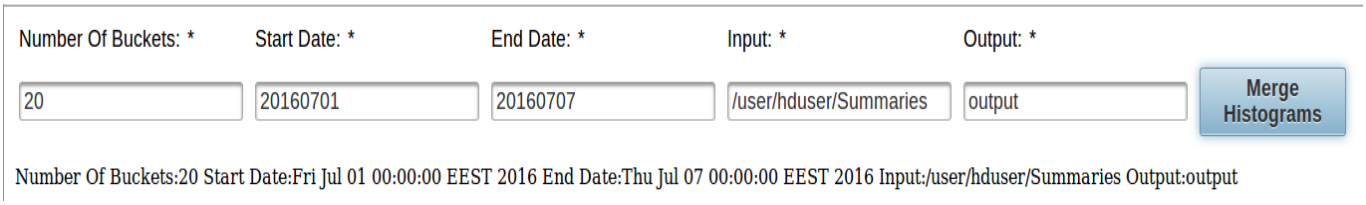

Figure 3. MergerWeb form.

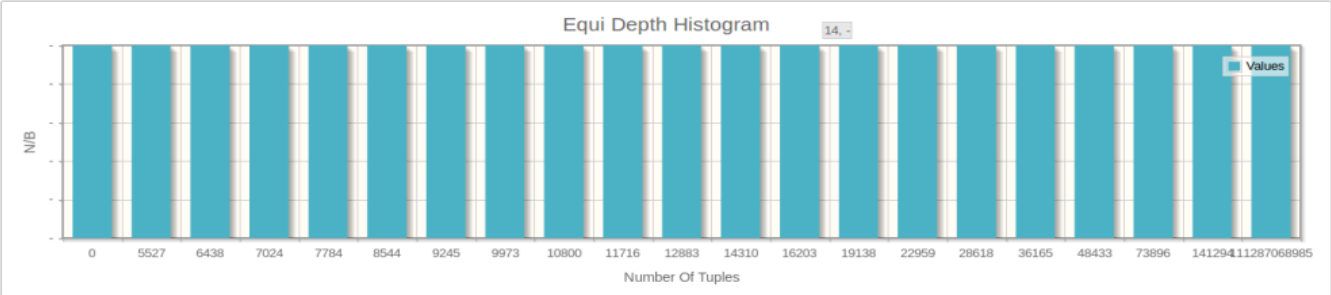

Figure 4. Equi-depth histogram constructed by MergerWeb.

We only use this job in MergerWeb, which is written in the JSF and Maven frameworks. The user interface is shown in Figure 3. In order to use this application, input fields, which are the number of buckets (β), start date, end date, input files path, and output file path, must be filled in, and then the 'Merge' button is clicked. As the job is finished, the constructed approximate equi-depth histogram of the time interval is obtained as demonstrated as Figure 4.

3.3. Comparison of Demonstrations

In this section, we present the comparison results of applications implemented with Apache Pig UDFs and Apache Hadoop Map-Reduce. For the testing, we created a Hadoop ecosystem with 2 machines running on DigitalOcean standard droplets. One of them has 8 GB memory, 4 vCPUs, and 160 GB SSD. We composed a master and slave on it. The other machine, which is a slave, has 4 GB memory, 2 vCPUs, and 80 GB SSD. We utilized Hadoop 2.7 and Apache Pig 0.16.0 for our experiments. We used 6 days of data from the hourly page view statistics dataset of Wikipedia, which is an

daily histograms are obtained, MergerWeb and MergerPig are run to create an approximate histogram of one-day, two-day, three-day etc. intervals. One-day histogram merging may be confusing, which is related to the number of buckets. There is no merging, but the bucket number is reduced. In Figure 6, a chart of elapsed time is provided to compare MergerWeb (orange line) and MergerPig (blue line).

The graphs depicted in Figure 5 show that Apache Hadoop Map-Reduce is nearly 2.5 times faster than Apache Pig when exact histogram constructions are considered. If the number of daily equi-depth histograms to be merged is increased, the difference between the elapsed times for MergerWeb and MergerPig increases.

The results presented here are not different from other big data problems implemented in Apache Pig and Apache Hadoop. Figures 5 and 6 illustrate the characteristics of Apache Hadoop Map-Reduce and Apache Pig. When comparing Apache Hadoop and Apache Pig, Apache Hadoop is more efficient. However, Apache Pig requires less effort when developing software.

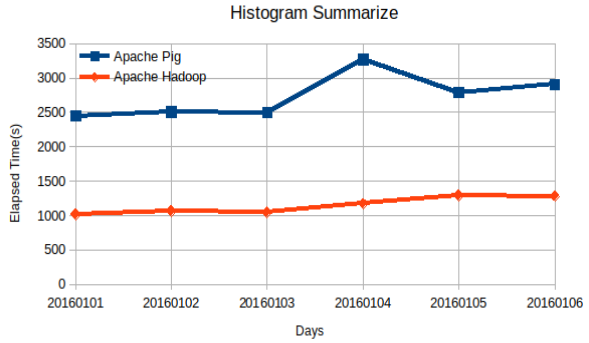

Figure 5. Comparison of the histogram summarization jobs.

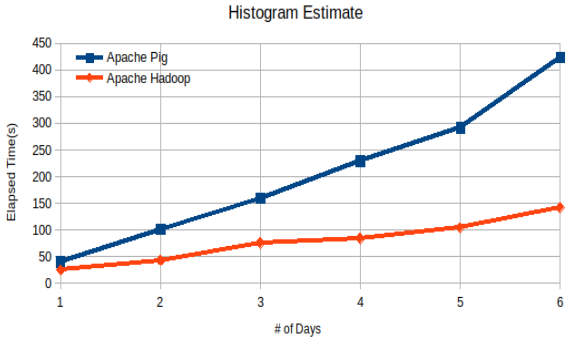

Figure 6. Comparison of the histogram estimation jobs.

3.4. Future Works

We have presented our implementations for Apache Pig and Apache Hadoop with demonstrations in this paper. However, this method can also be implemented in other big data ecosystem tools such as Apache Spark and Apache Hive with UDFs.

4. CONCLUSION

In this paper, we have implemented a novel approximate equi-depth histogram method on Apache Pig using Apache Pig UDFs and have developed a web interface to create approximate histograms using Hadoop Map-Reduce jobs, which is also an implementation of this method on the back-end. The proposed method is more accurate and faster than the tuple-level sampling method. We have shown that the advantages of Hadoop jobs, i.e. Hadoop Map-Reduce, and the quality-guaranteed equi-depth histogram construction method can be used in web applications with our demonstration of MergerWeb. By utilizing the extensibility property of Apache Pig, we showed that our Apache Pig Scripts (SummarizerPig and MergerPig) can be used to construct exact and quality-guaranteed equi-depth histograms. Although Apache Hadoop is an easily scalable distributed big data tool, the solutions of big data problems usually require multiple skills and tasks, such as creating multiple jobs. Instead of these job operations, using high-level languages such as Apache Pig allows for the easy execution of big data processing and data management for unstructured data models. Hence, programmers have advantages in using Apache Pig to create histograms that eliminate the need for the implementation of Map-Reduce jobs. Consequently, the same approximate histogram construction method has been implemented in a web interface using Apache Hadoop Map-Reduce in the background and has also been implemented in Apache Pig with UDFs. It was found that the method implemented in Apache Hadoop Map-Reduce yields faster results than the method implemented in Apache Pig UDFs.

REFERENCES

[1] Logothetis D., Olston C., Reed B., Webb K. C., and Yocum K. , "Stateful bulk processing for incremental analytics", ***In Proceedings of the 1st ACM symposium on Cloud computing***, 51-62, (2010).

[2] Thusoo A., Shao Z., Anthony S., Borthakur D., Jain N., Sen Sarma J., ... and Liu H. , "Data warehousing and analytics infrastructure at facebook", ***In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data***, 1013-1020, (2010).

[3] Thusoo A., Sarma J. S., Jain N., Shao Z., Chakka P., Zhang N., ... and Murthy R. , "Hive-a petabyte scale data warehouse using hadoop", ***In 2010 IEEE 26th international conference on data engineering (ICDE 2010)***, 996-1005, (2010).

[4] Yıldız B., Büyüktanır T., and Emekci F. , "Equi-depth histogram construction for big data with quality guarantees", ***arXiv preprint arXiv:1606.05633***, (2016).

[5] https://hadoop.apache.org, "A. S. Foundation Apache Hadoop", (2008).

[6] Dean J., and Ghemawat S., "MapReduce: a flexible data processing tool", ***Communications of the ACM***, 53(1): 72-77, (2010).

[7] Dittrich J., Quiané-Ruiz J. A., Jindal A., Kargin Y., Setty V., and Schad J., , "Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing)", ***Proceedings of the VLDB Endowment***, 3(1-2): 515-529, (2010).

[8] Gates A. F., Natkovich O., Chopra S., Kamath P., Narayanamurthy S. M., Olston C., ... and Srivastava U., "Building a high-level dataflow system on top of Map-Reduce: the Pig experience", ***Proceedings of the VLDB Endowment***, 2(2): 1414-1425, (2009).

[9] Jindal A., Quiané-Ruiz J. A., and Dittrich J. , "Trojan data layouts: right shoes for a running elephant", ***In Proceedings of the 2nd ACM Symposium on Cloud Computing***, 1-14, (2011).

[10] Zaharia M., Konwinski A., Joseph A. D., Katz R. H., and Stoica I., "Improving MapReduce performance in heterogeneous environments", ***In Osdi***, 8(4): 7, (2008).

[11] Isard M., Budiu M., Yu Y., Birrell A., and Fetterly D., "Dryad: distributed data-parallel programs from sequential building blocks", ***In Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007***, 59-72, (2007).

[12] Schumacher A., Pireddu L., Niemenmaa M., Kallio A., Korpelainen E., Zanetti G., and Heljanko K., "SeqPig: simple and scalable scripting for large sequencing data sets in Hadoop", ***Bioinformatics***, 30(1): 119-120, (2014).

[13] Wu S., Li F., Mehrotra S., and Ooi B. C., "Query optimization for massively parallel data processing", ***In Proceedings of the 2nd ACM Symposium on Cloud Computing***, 1-13, (2011).

[14] Babu S., "Towards automatic optimization of MapReduce programs", ***In Proceedings of the 1st ACM symposium on Cloud computing***, 137-142, (2010).

[15] Herodotou H., and Babu S., "Profiling, what-if analysis, and cost-based optimization of mapreduce programs", ***Proceedings of the VLDB Endowment***, 4(11): 1111-1122, (2011).

[16] Jahani E., Cafarella M. J., and Ré C., "Automatic optimization for MapReduce programs", ***arXiv preprint arXiv:1104.3217***, (2011).

[17] Jiang D., Ooi B. C., Shi L., and Wu S., "The performance of mapreduce: An in-depth study", ***Proceedings of the VLDB Endowment***, 3(1-2): 472-483, (2010).

[18] Dittrich J., Quiané-Ruiz J. A., Richter S., Schuh S., Jindal A., and Schad J., "Only aggressive elephants are fast elephants", ***arXiv preprint arXiv:1208.0287***, (2012).

[19] Floratou A., Patel J., Shekita E., and Tata S., "Column-oriented storage techniques for MapReduce", ***arXiv preprint arXiv:1105.4252***, (2011).

[20] Lin Y., Agrawal D., Chen C., Ooi B. C., and Wu S., "Llama: leveraging columnar storage for scalable join processing in the MapReduce framework", ***In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data***, 961-972, (2011).

[21] ***http://www.internetlivestats.com/google-search-statistics***, Google search statistics.

[22] ***https://advertising.yahoo.com/yahoo-sites/Homepage/index.htm***, Yahoo advertising.

[23] Ioannidis Y., "The history of histograms (abridged)", ***In Proceedings 2003 VLDB Conference,*** 19-30, (2003).

[24] Olston C., Reed B., Srivastava U., Kumar R., and Tomkins A., "Pig latin: a not-so-foreign language for data processing", ***In Proceedings of the 2008 ACM SIGMOD international conference on Management of data***, 1099-1110, (2008).

[25] Hallam-Baker P. M., "Extended log file format", ***WWW Journal***, (1996).

[26] ***https://docs.oracle.com/database/121/TGSQL/tgsql_histo.htm#TGSQL380***, Oracle Database Histograms.