IRSTI 20.53.19

U.T. Nurkey[1], A. Bogdanchikov[1]
[1]Suleyman Demirel University, Kaskelen, Kazakhstan

TEACHING BIG DATA ANALYTICAL PLATFORMS IN HIGHER EDUCATION FOR GRADUATE DEGREE STUDENTS

Abstract. An extensive archive of petabytes of data has been generated from modern information systems and digital technologies such as scientific data analysis, social data analysis, reference systems and Internet services journals. To investigate and extract knowledge from this enormous data much effort is needed. Due to this, Big Data Management Systems need to be integrated as part of the computing curriculum. In this article, we present examples of analysed tasks that can be processed as large data projects using Apache Hadoop, and it's Map – Reduce, Apache Spark, Hive and Pig by demonstrating how each type of system can be integrated via sample datasets and data analysis tasks. The aim of this paper is to show how Big data analyzing tools can be educated through sample tasks, their solution and implementation.

Keywords: big data, education of data mining, hadoop, spark.

***

Андатпа. Деректердің петабайттарының үлкен репозиторийі күн сайын заманауи ақпараттық жүйелерден және ғылыми деректерді талдаудан, әлеуметтік медиа деректерін өңдеуден, ұсыныс жүйесінен және веб-қызмет журналдарынан талдау сияқты цифрлық технологиялардан жасалады. Бұл массивтік деректерді талдап маңызды деректерді табу және соған сәйкес шешімдер қабылдау – көп күш жұмсауды талап етеді. Бұл факт есептеудің бір бөлігі ретінде үлкен деректерді басқару жүйелерін зерттеуді интеграциялаудың қажеттілігін тудырады. Бұл мақалада Apache Hadoop, HDFS және Map-Reduce көмегімен үлкен деректер жобаларына бастама ретінде шешілген аналитикалық мәселелердің мысалдары келтірілген; Apache Spark; Apache Hive және Apache Pig арқылы келешекте мысалдар келдіріледі; жүйенің әрбір түрі деректер жиынтығымен және деректерді талдау тапсырмаларын орындау арқылы жоғарғы оқу процессіне қалай енгізілуіне болатынын көрсетеді. Осы мақаланың мақсатыҮлкен деректерді талдау құралдарын үлгілік тапсырмалар мен оларды шешу және іске асыру әдістерімен оқытуға болатындығын көрсету болып табылады.

Түйін сөздер: үлкен деректер, жоғары білім беру жүйесі үшін деректерді өңдеу, hadoop, spark.

***

Аннотация. Огромное хранилище размерами в петабайты данных генерируется каждый день из современных информационных систем и цифровых технологий, таких как анализ научных данных, анализ данных в социальных сетях, системы рекомендаций и анализ журналов веб-служб. Анализ этих массивных данных требуют много усилий на разных уровнях для извлечения знаний и дальнейшего принятия решений. Этот факт создает необходимость интегрировать изучение систем управления большими данными как часть компьютерной учебной программы. В этой статье представлены примеры аналитических проблем, которые могут быть решены в виде введения в проекты больших данных с использованием ApacheHadoop и Map-Reduce; ApacheSpark, Hive, Pig в будущих работах; демонстрируя, как каждый тип системы может быть интегрирован с помощью выбранных наборов данных и задач анализа данных. Цель этой статьи – показать, как инструментам анализа больших данных можно обучить студентов с помощью примеров, задач, их решения и реализации.

Ключевые слова: большие данные, обучение анализу данных, hadoop, spark.

Introduction

Big Data is modifyinghealthcare, engineering, science, finance and business, and eventually our society itself. Common use cases are:

- Analyzing logs of web pages;
- Risk modeling with unstructured data;
- Social sentiment analysis;
- Image classification for health care, for example;

Thus, the integration of this course aims to expose students to various tools and techniques by developing amodel for decision-making to select the right tool for any big data problem. An important contribution of this work is description of sample projects and class exercises, their solutions and how these resources support the learning outcomes by analyzing survey provided at the end of the semester, allowing practical experience with big data technologies.

Literature review

Since large data is still developed as an important research area and new findings and tools are constantly evolving, this chapter does not include all possibilities and focuses on providing a list of present realistic tools, techniques and methods for parallelization of big data by the work of other researches on these issue.

Paritosh Goldar, Yogesh Rai, Santosh Kushwaha, 2016 made research on real time analytical and comprehensive analysis of big data parallelization techniques and methods. They investigated its unique characterizations based on the analytical methods and experiments. The advanced and future approaches of big data were reviewed[1]. There are several further future analysis of big data parallelization when it is integrated with other core implementation of programming and algorithms.

Summary from Bhandarkar, M. explains how to design and developHadoop applications and advanced application systems running between 4 and 4000 computers for terabytes of data. The idea of general problems in expanding performance of Hadoop is examined[2].

Researchers from Arizona state university shared their experience in integrating Big data analysing tools in their curriculum[3]. They defined challenges for computer science educators that raise while organizing different BDMS technologies(MapReduce, NoSQL with HBase, NewSQL) into learning units when including into the computing curricula. What is handful in this paper, authors provide class resources like in-class exercises, sample projects and programs to implement firsthand experience with big data analysing tools. For consistent and durable data processing database systems are preferable according to the paper, while Map-Reduce is an appropriate tool for analyzing and receiving useful information like aggregations, trends and correlations (for selling operations).

As well as this, use cases for HBase is discussed: in the messaging system, it is the right tool to run a query that pulls the user's message quickly when it is randomly accessed. In the same case, framework Map-Reduce considered to be the best tool to run a job which perform oncomplete dataset and finds total number of messages by location of sender.

Big data projects included in curriculum of Stetson University by using Apache Hadoop, HDFS and Map-Reduce; Apache Hive; Apache Spark; and other tools are discussed in Joshua Eckroth’s work [4]. The main learning aim there is that students show the ability to select which tool is the most convenient for particular data analysis tasks and build model to choose right tool for each dataset. First, students were given 3GB dataset to analyze failure rate by plotting graphics, and find unique characteristics that have a higher proportion of failures: where students had to use RStudio, which was a painful process for R because of slowing down it. Next, students were given more than 116GB XML files and needed to summarize a dataset by producing jobs with Java/Python in Map-Reduce. After that they were asked to write straightforward queries by using Hive, where they found that Hive is simpler to use than Map-Reduce for these kind of tasks. As well as this, machine learning tasks were given to be solved by using Spark Mahout VS Weka workbench. In decision matrix Hive was preferable in SQL-like queries whereas Spark was selected for image processing for big data.

Theoretical differences alongside with comparison of performances of Hadoop and Spark is explained in I-SMAC 2017 as well [5]. In terms of batch and iterative processing, data storage and access rate, real time processing, working principles, and differences in computation are provided. As a practical performance evaluation authors used two data sets: wordcount problem by changing number of words for each iteration; problem related with predicting by using logistic regression method. For both cases they used single node cluster. As was expected, performance ratio of Spark over Hadoop was 2.82 whereas in iterative queries Spark was faster by 2.17 ratio. Even if Spark is faster for iterative queries, since it saves it's data in a cache, and the size of that cache is limited, that is why second experiment showed fewer performance rate.

Comparison of Hadoop and Spark was provided in many papers, for example, full-text search with more than 50GB on-disk Wikipedia data on a 20 node Amazon cluster took 20 seconds with Hadoop, whereas with an in-memory RDD the same query took only 0.8 seconds [6]. Advantage of storage model in Spark (RDD) guarantees fault tolerance that minimizes network I/O.[7]

Figure 1 compares performance of HadoopversusSpark for fluctuating numbers of iterations. In this experiment, 100 GB of data on a 50-node Hadoopclusterstakes approximately 110 secondsper each iteration. However, Spark takes for the first iteration 80 secondsto load the data intoRAM, but for each nextiterations 6 seconds! [7]

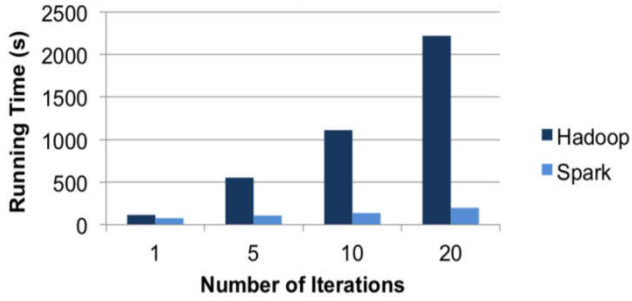

Fig.1. Performance of logistic regression in Hadoop VS Spark [7].

By taking account all provided reviews of other researches, Big data course was planned carefully to fulfill graduate degree student's knowledge and some practical exercises were selected for each tool.

Data processing withhadoop map-reduce

For the following in-class exercises was used pseudo-distributed mode of Hadoop which allows Map-Reduce jobs on a single node where each Hadoop daemon runs on a different Java process. Total number of tasks was 15

for one semester, which is the duration of this course. The Map-Reduce programs that will be presented in the next sections make use of three datasets:

- *sdu-portal.log file;*
- *2 files with information about vectors;*
- *twitter dataset*;
- *2 files about airlines : 2008 (csv) , airports.csv.*

Sdu.portal.log file contains which pages are being accessed, by whom, and when (date, by using which browser) from my.sdu.edu.kz web server.

Each line of the Twitter dataset contains the following comma separated values about twit ID, date and time, sender username, twit itself .

Airline on-time performance dataset hasdetails about flight arrivals and departuresfor all commercial flights in the USA, from October 1987 to April 2008. Total dataset contains 120 million records, and takes up 1.6 Gb of compressedspace.

1. Implementing Map-Reduce(in stand alone mode) program to:

- Count URL access frequency of sdu-portal.log

Mapper and Reducer is executed in one java file, so that output of mapper goes to reducer directly. In mapper stage by using Pattern and Matcher classes, all HTTP codes are found:

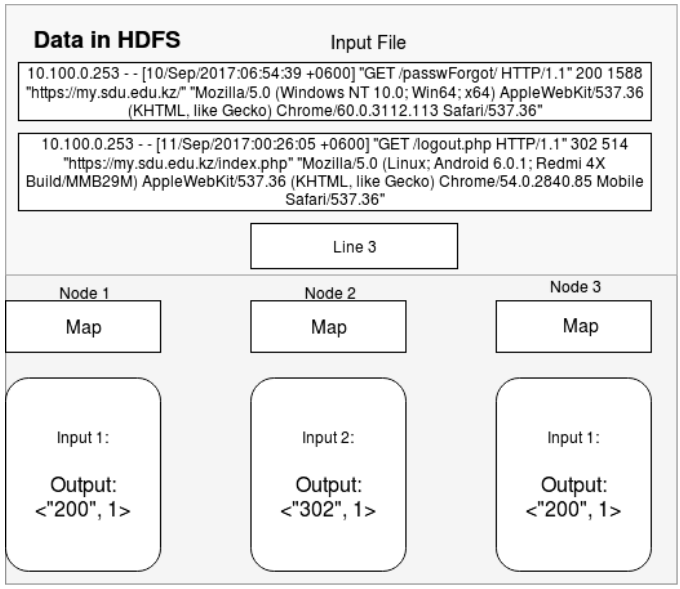

Fig. 2. Input and Output of Map task, shuffle stage

In Shuffle and Reduce stages all same keys are collected and their values summed up. Since frequency should be counted, second Reducer task is to divide each counted value to the total URL numbers.

- Find in which period of time portal is loaded most, output is a list of all hours from most accessed to least
- After extracting Date, hours are sent to Reducer with their values, where each hours value will be counted.

2. Inner product of two sparse vectors

Input files: two vectors data (A.txt, B.txt)

Output of the first Map-Reduce job should be multiplications of corresponding elements.

To solve this problem, the first step is preprocessing in mapper step by joining two sparse vectors data into one file as key and value. In Reducer, list is used where it checks if there exists key:

```
if result.get(key):
        result[key]=result[key]*value
else:
        result[key]=value
```

Fig.3 Reducer code for vector

Next task is to calculate sum of all elements by using aggregate function.

3. Twitter dataset [8].

Join trending word count and total word count and percentage for each day from twitter dataset. Here Streaming API is used to write in Python. Hadoop Streaming enables commands to be used as mappers and reducers.

- Given twitter data needs to be extracted by each word in that date, so that each line would contain date and word from twit separately, and number 1(to count in reducer).
- The job of Reducer will be to count the occurrence of word for each date(*out1*).
- Next mapper's task is to take output from previous reducer (date, word, occurence of word for that day) as input; and map only words with their values. Reducer is the same, however counts total occurrence of that word without date(*out2*).
- Last mapper sends two inputs (*out1, out2*) to reducer, and by joining by key as words, divide the number of each word by days to the total number of that word.

4. task 9: By using Pig LOAD sdu-portal.log data and find link with maximum number of accesses.

First by using *load* command in Pig SQL-like scripting language we load log file separating by blank spaces. Then from each line we take url address name (10'th field) and *group* line*by* value of url address field. In Pig it is needed to *generate* group (as group_name) where *count* function can be applied to taken url address names. Final step is sorting: *order* last output by

result from *count* function above in descending order. There are total 5 lines of code.

5. task 9.2 :LOAD 2008_flights.csv and output distinct flights sorted by flight distance

By using Pig LOAD command 2008.csv file is loaded with comma delimiter. From all fields only need to select origin of flight, destination of flight and distance of flight converted to integer. Pig has built in function DISTINCT to show only unique flights. The output of DISTINCT is just ordered by distance field in descending order. Total 5 lines are used to solve this problem, including loading and showing(dump) the results.

6. task 10.1 by using Hive- Create Table of Airlines and Load Hive 2008.csv; Find number of outgoing flights for each airport with airport name.

First part of task is - sql query to create new external tables with all fields from 2008.csv and airports.csv. with correct data types. These two tables can be listed in one sql, which then goes to python code to be parsed and joined through *iata*field and saved in dictionary.

```
for line in sys.stdin:
    line=line.strip().split('\t')
    if len(line)>7:
        Origin,FlightNum=line[16], line[9]
        dict_lict[Origin]=dict_lict.get(Origin,0)+1
    else:
        iata, airport=line[0],line[1]
        if dict_lict.get(iata):
            dict_lict[airport]=dict_lict.pop(iata)

for key, value in sorted(dict_lict.items(), key=lambda item:(item[1], item[0]))[::1]:
    print("%s:%s" % (key, value))
```

Fig.3. Sample pyspark code

As a key & value pair name of airport & outgoing flights from this airport goes.

7. Implementing similar tasks like 5,6 by using Spark (+ Find all unique flights and their distances + average number of flights per day.)

After loading 2008.csv file into pyspark we group all flights by origin and count number of flights (N) for each origin place:

- To find average number of flights that flew every day we divided total amount of flights for each origin place(N) into 365.
- To find unique flights and their distance we need to select Origin , destination and distance from all data, then use function distinct()

8. By using apiHiPi, find RGB histograms for each images, sort histograms by color intensity

Discussion

To discover how much learning aims were reached and to assess the integration of proposed tools to curriculum of master and PhD students, we

took survey from participants in Big Data analytics course. While this was a small class, the results are encouraging. The survey provided students with aseveral of specific questions about the module. First, we asked about their levels of competence on Java, Linux, and Python to understand level of undergraduate studies on Hadoop Map-Reduce. As given in Table 1 below, students enrolled the course with mostly intermediate knowledge to understand the programming aspects of Map-Reduce and Linux file system for HDFS data location.

Table I

Level of Proficiency

Java	Python	Linux
60% intermediate 15% advanced	46% intermediate 15% advanced	77% intermediate

Next, the survey asked to measure the time students spent on finishing each assignment to determine whether the workloads of the assignments were appropriate.

Table II

Time to Complete (1: less than 3 hours, 2: 4 to 8 hours, 3: 1 to 2 days, more than 3 days).

Activity	**Time taken**
1. LetterCount based on WordCount	100% -less than 8 hours;
2. Count URL access frequency with Java; 3. Inner product of two sparse vectors	75%-less than 8 hours(1,2);
4. Period of time when portal is loaded most	84%-less than 8 hours
5. Working with twitter dataset with Python	67%-less than 8 hours
6. RGB histograms for each images, sort histograms	25%-less than 8 hours; 33%-1-2 days; 50%-more than 3 days
7. Counting URL access	84%-less than 8 hours

frequency using Pig; 8. Distinct flights sorted by flight distance using Pig; 9. Number of outgoing flights for each airport with airport name using Hive; 10. Average number of flights per day using Spark	
11. All unique flights and their distances using Spark	100%-less than 8 hours(67% within 3 hours + 33% 4-8 hours)
12. Find the shortest backup flight for each unique flight using Spark	58% less than 8 hours; 42%-more than 1 day

As well as this, for each task was a question about complexity level of tool they needed to use exact for that task and complexness of task implementation itself. First three tasks were easy relatively, tool usage might took more time for implementing. For task 4 and 5, students (85%) responded this task as moderate/difficult, however tool usage was easy. Most difficult assignment for students was task 6, where 92% of them agreed with complexness of tool and assignment itself. The score dispersion for the next 4 assignments were very similar: simplicity of assignment and tool usage in tasks 7,8,9,10,according to students, was 92%. Assignment where students were asked to find the shortest backup flight for each unique flight using Spark was considered complex for implementing (83%), however easy for tool usage(92%).

In the open ended questions, we asked them to write their opinion about which of the given assignments would be preferable to solve in Hadoop Map-Reduce, Apache Pig or Apache Hive, or Apache Spark based on their experience. As a preferable assignments for Hadoop Map-Reduce questions students voted mostly for tasks 1-6; also for the questions «In your opinion, which task is better to solve with Apache Pig/Hive/Spark?», we received similar responses like: «all tasks which related to joining and counting huge data set», «Data preprocessing", "When you want to get faster computing», «Almost all», «When you have tables with data», «Where data is structured».

The last option in survey was to evaluate their knowledge on each learned big data tool after passing this course. Overall student's feedback about the course was consistently high based onimportance and practicality, quality of the lecture materials and suitableness of assignments. More interestingly, students expressed strong growth during the course: half of them felt themself as intermediate user ofHadoop and Spark.

Conclusion

Examples are used in this paper were solved by master and PhD students in standalone and pseudo-distributed mode of Hadoop. Since the size of datasets were different, comparison by time is not efficient. Discussed examples gives good opportunity to learn big data analytic tools.

Graduates can work for companies that already use BDMS or want to use them to get better insights from the large data sets they manage. Therefore, it is important to integrate research into these systems as part of curriculum.

However, course suffers from some limits like: absence of truly big datasets of TB or larger; resources like multiple nodes.

Despite this, we believe that students will gain rich experience in a big data tools, and their feedback show that they agree.

This course differs from similar courses at other institutions in itsbroad range of subjectsfrom small data analysis and visualization to big data processing. For a more focused course including data analysis components, review Linh B. Ngo [9], developed a course focused on Map-Reduce.

References:

1 Paritosh Goldar, Yogesh Rai, Santosh Kushwaha. A Review on Parallelization of Big Data Analysis and Processing //IJETCSE - 2016. -Vol. 23 Issue 4. - P. 60-65.

2 Srinivasa, K.G., Siddesh, G.M., Srinidhi H. Network Data Analytics: A Hands-On Approach for Application Development. *Springer International Publishing*, 2018. - 398 p.

3 Joshua Eckroth. Teaching future big data analysts: Curriculum and experience report //*IEEE International Parallel and Distributed Processing Symposium Workshops*. - 2017. - pp. 346-351

4 Silva, Y.N., Dietrich, S.W., Reed, J.M., Tsosie L.M. Integrating big data into the computing curricula //*45th ACM Technical Symposium on Computer Science Education*. - 2014. - pp. 139–144.

5 Akaash Vishal Hazarika, Jagadeesh Sai Raghu Ram, Eeti Jain. Performance Comparision of Hadoop and Spark Engine //*I-SMAC*. - 2017. - pp. 671-674.

6 https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

7 Simpson, H. *Fast and Interactive Analytics over Hadoop Data with Spark*. Dumb Robots, 3rd ed., Springfield: UOS Press. - 2012. - pp. 6-9.
https://www.kaggle.com/kazanova/sentiment140

8 Ngo, L.B., Duffy, E.B., Apon, A.W. Teaching HDFS/MapReduce systems concepts to undergraduates //*IEEE International Parallel & Distributed Processing Symposium Workshops* (IPDPSW). -2014. - pp.1114–1121.