CSE4/587 Data-intensive Computing Spring 2017

LAB 5: PROGRAMMING THE DATAFLOW FOR BIG DATA ANALYTICS USING APACHE SPARK: B. RAMAMURTHY

OVERVIEW:

The hands-on practical learning components of the course comprises two types of activities: labs covering one or two knowledge units (skills, competencies) of data-intensive computing and a single term project serving as a capstone covering the entire data pipeline. In the first half of the course we learned data analytics and quantitative reasoning using R language. In the second half we focused on big data approaches for data analytics with Hadoop MapReduce and Apache Spark. In this lab, we will work on understanding concepts related to data analysis using Apache Spark [1].

In Lab1 we wrote a data client and very simple information server. In Lab2 we worked on data cleaning and data munging. In lab (Lab 3) we applied machine learning algorithms and statistical models to data with the ultimate goal of being able to predict the outcome for a certain query or classify data according to certain criteria. More specifically, we explored algorithms discussed in Chapter 3 of Doing Data Science textbook [2]: linear regression, k-nearest neighbors, k-means. In lab4, we are exploring approaches that deal with big data, especially text data, using the Google's MapReduce algorithm [3]. In Lab 5 we will explore processing graph data using Spark [1].

GOALS:

Major goals of the lab5 are to:

1. **Understand** the components of the Apache Spark framework and programming: spark context (sc), dataflow operations in **transformations** and ***actions***
2. ***Study the scalability provided by Spark. Analyze the performance improvement*** achieved by Apache Spark vs. traditional Hadoop MR by repeating the analysis with Project Tesserae [4] data.
3. ***Format*** the output according to the format shown in the Project Tesserae page [4].

OBJECTIVES:

The lab goals will be accomplished through these specific objectives based on the needs of the stakeholders of the project:

1. Open and read the data of Latin texts provided. These are available at UBBox [5] and also in the instructions for Lab4. Understand the input data format and analyze how you may be able to reuse most of the data in a line about the author, chapter number etc.
2. Work on a simple but well-known data set to understand Apache Spark environment and programming.

3. Repeat the word co-occurrence (for n-grams, n = 2, 3) with "line" as the scope or context for co-occurrence. You will not only pick adjacent words but also all words (forward) co-occurring with the current word but within the line. This is same as Activity 4 of Lab4.
4. Compare the performance as you increase the number of documents as well as n in n-grams.

LAB DESCRIPTION:

Introduction: In this age of analytics, data science process plays a critical role for many organizations. Several organizations including the federal government (data.gov) have their data available to the public for various purposes. Social network applications such as Twitter and Facebook collect enormous amount of data contributed by their numerous and prolific user. For other businesses such as Amazon and NYTimes data is a significant and valuable byproduct of their main business. Nowadays everybody has data. Most of these data generator businesses make subset of their data available for use by registered users for free. Some of them as downloadable data files (.csv, .xlsx) as a database (.db, .db3). Sometimes the data that needs to be collected is not in a specific format but is available as a web page content. In this case, typically a web crawler is used to crawl the web (pages) and scrap the data from these web pages and extract the information needed. Data generating organizations have realized the need to share at least a subset of their data with users interested in developing applications. Entire data sets are sold as products. Very often data collected is not in the format required for the downstream processes such as EDA, analysis, prediction and visualization. The data needs to be cleaned, curated and munged before modeling and algorithmic processing.

Also we will follow the pedagogical pattern

- Preparation before lab (pre-lab)
- Learn from working on some of the solved problems.
- Apply the knowledge to meet the goals of this project to study scalability.

PREPARATION: Here are the preliminary requirements for the lab. **Time needed: 2 to 3 hours (Day 1)**

1. Work in groups: You will work in groups of 1 or 2. Find partner who will share the work with you equally and also have complementary resources and skills so that you can learn from each other. No more than two per group is allowed.
2. Choose an Apache Spark environment: VM [6], docker, Jupyter, Windows/mac/linux native installations, google app engine [7], amazon EMR[8]
3. Make sure the environment you have chosen works well with Apache-Spark sample programs such as word count. Try it out and get familiarized with the dataflow and commands.

Lab 5: What to do?

1. **(15 points) Understand Apache Spark with Titanic data analysis. (Day 2,3: Time Needed: 3-4 hours per day)**
   This activity is similar to an R-vignette. In this case you learn features of Apache Spark. On Jupyter's [9] start page the "Try it in your browser" and once you are inside the notebook, click on the Python or Scala version of the Spark notebook. Run the same analysis on the environment you created in the Pre-lab and make sure it works.

2. **(35 points) Featured activity: Analysis of Latin documents for word-co-occurrence (Day 3,4 : 3-4 hours each day)**

Program a word-co-occurrence program in Spark and test it for n=2 (n-grams) and just 1 or 2 documents. Scale it up for as many documents as possible. Instructions from Lab4 are repeated here.

This problem was provided by researchers in the Classics department at UB. They have provided two classical texts and a lemmatization file to convert words from one form to a standard or normal form. In this case you will use several passes through the documents. The documents needed for this process are available in in the UBbox [7].

Pass 1: Lemmetization using the lemmas.csv file

Pass 2: Identify the words in the texts by <word <docid, [chapter#, line#]> for two documents.

Pass 3: Repeat this for multiple documents.

Here is a rough algorithm (non-MR version): Apply all the experience from lab4 in deciding the data flow.

for each word in the text

normalize the word spelling by replacing j with i and v with u throughout

check lemmatizer for the normalized spelling of the word

if the word appears in the lemmatizer
obtain the list of lemmas for this word
for each lemma, create a key/value pair from the lemma and the location where the word was found
else
create a key/value pair from the normalized spelling and the location where the word was found

From word co-occurrence that deals with just 2-grams (or two words co-occurring) increase the co-occurrence to n=3 or 3 words co-occurring. Create a table of n-grams and their locations. Discuss the results and plot the performance and scalability.

n-gram (n =2)	Location
{wordx, wordy}	Document.chap#.line#.position#
Etc.	
n-gram (n=3)	Location
{wordx, wordy, wordz}	Document.chap#.line#.position#
…	…

In this activity you are required to "scale up" the word co-occurrence by increasing the number of documents processed from 2 to n. Record the performance of the Apache Spark infrastructure and plot it as discussed in Chapter 3 of Lin and Dyer's text [10]. Also see the performance evaluation

charts on p.56 (60) of the same text. A table similar to the above will have thousands of entries. Add the documents incrementally and not all at once.

SUBMISSION DETAILS:

1. Readme file containing your name (First and last) and your partner's name at the top with all the other details such as environment you have chosen etc. how to run your program and explanation of the output.
2. Other details of format of data outputs will be given to you later.
3. Online submission: do not email me: (-10 points) if you email me the code.

DUE DATE: 5/13/2017 BY 11.59PM. ONLINE SUBMISSION.

REFERENCES:

[1] Apache Spark. http://spark.apache.org/, last viewed 2017.

[2] C. ONeil and R. Schutt, Doing Data Science, ISBN:978-1-4493-5865-5. Oreilly Media, Doing Data Science, http://shop.oreilly.com/product/0636920028529.do, 2013.

[3] Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.

[4] Project Tesserae. http://tesserae.caset.buffalo.edu/, last viewed 2017.

[5] UBBox data set for Lab4: https://buffalo.box.com/s/s4v5zsod0djle5l8xa5ml9imeei28n71 last viewed 2017.

[6] Virtual Machine for Hadoop MapReduce on UBbox: https://buffalo.box.com/s/52did77hn2vjoje7iguf19btgs6vhvsc, last viewed 2017.

[7] Amazon AWS Elastic MapReduce. https://aws.amazon.com/emr/ , Last Viewed April 2017.

[8] MapReduce on Google App Engine. https://cloud.google.com/appengine/docs/standard/python/dataprocessing/, last viewed April 2017.

[9] Project Jupyter. Jupyter.org, last viewed 2017.

[3] J. Lin & C. Dyer. Data-intensive text processing with MapReduce, https://lintool.github.io/MapReduceAlgorithms/