Big Data Ecosystems Programming Assignment 2: Implement PageRank on Spark

This project is to implement the PageRank to find the most important Wikipedia pages on the provided Wikipedia dataset using Spark/GraphX on the cloud platform AWS Elastic MapReduce (EMR) or Amazon EC2. Programming languages: Scala, Java, Python.

1 Data Set

1.1 Data Description

We choose the Freebase Wikipedia Extraction (WEX), a processed dump of the English version of Wikipedia. The complete WEX dump is approximately 62GB uncompressed, and contains both XML and text versions of each page. We only need use the "articles" files from the Data Set (32G), which contains one line per wiki article in a tab-separated format. Each line in WEX data set (freebase-wex-2009-01-12-articles.tsv) has five fields, separated by tabs:

- page id
- page title
- last date the page was modified
- XML version of the page
- plain text version of the page

Please refer to `http://wiki.freebase.com/wiki/WEX/Documentation` for more information.

1.2 Data Download

The WEX data set is made public available. There are two ways to download:

- Access the data from Public Data Sets of AWS (`http://aws.amazon.com/datasets/2345`). If you are not familiar with how to use the public data set, please refer to:

  `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-public-data-sets.html`

- Access the data through S3. There is one file in:

  `s3://bigdata2015-uf/wikidata/freebase-wex-2009-01-12-articles.tsv`

Please make sure you export the correct AWS_SECRET_ACCESS_KEY and AWS_ACCESS_KEY_ID when you use AWS services. In order to deeply share data and improve the parallelismplease move the input data into HDFS for data pre-processing and data analysis.

2 Software/Hardware Environments

2.1 AWS Account

You need to first creat an AWS account if you haven't done so. You might find following link helpful: https://s3.amazonaws.com/prod_object_assets/assets/25445975006398/AWSsetup.pdf?AWSAccessKeyId=AKIAIXM6FRIC5QVSA63Q&Expires=1453482459&Signature=%2B9noFQIvOZFy3D#_=_

2.2 Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. You might find following links helpful when you create the Spark EMR:

- http://blogs.aws.amazon.com/bigdata/post/Tx15AY5C50K70RV/Installing-Apache-Spark-on-an-Amazon-EMR-Cluster
- https://aws.amazon.com/articles/Elastic-MapReduce/4926593393724923
- https://github.com/awslabs/emr-bootstrap-actions/tree/master/spark

2.3 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. We hope you might following links helpful when you create Spark EC2:

- http://spark.apache.org/docs/latest/ec2-scripts.html
- https://aws.amazon.com/amazon-linux-ami/2014.09-release-notes/

Even though you are required to run your program on EC2/EMR, we suggest you might prefer to develop and debug your code in your local machine, then deploy them into EC2/EMR.

3 Submission

Your submission contains three parts: basic, advanced, and bonus. Their specific requirements and corresponding marks are as below:

- (50%) Basic: Get the top 100 Pages.
- (50%) Advanced: Performance analysis, comparison using pure Spark and GraphX
- (20%) Bonus: Get the top 100 universities and their properties (PageRank/weights, and others: e.g., top alumni etc.

3.1 Job Jar Files

Besides the results, you are required to provide the executable Job jar files, which can launch applications by spark-submit along with the customizable Wikidata Location and the master URL for the cluster, for example, spark://23.195.26.187:7077. For details, please refer to: http://spark.apache.org/docs/1.2.0/submitting-applications.html

3.2 Submission Package

In summary, your report should contains following contents:

- PageRank results (Top 100 pages).
- Provide the procedures to configure, compile, package, execute code/jar and input/output files.
- Performance tuning results: iteration time analysis, scalability (performance versus number and instance types), HDFS configuration, PageRank algorithm optimization, compare performance using pure Spark and GraphX.
- In your submission, please include the zipped directory of all source codes and link to your GitHub repository

4 Some Useful Links

During this assignment, google is your best friend. Here we provide several links that you might find useful.

- GraphX Programming Guide: `https://spark.apache.org/docs/1.1.0/graphx-programming-guide.html`
- `http://blog.argteam.com/coding/university-ranking-wikipedia/`
- `http://ilpubs.stanford.edu:8090/573/1/2002-6.pdf`