Pig PageRank Project

Abderrezak Allalen

Varun Patil

Indiana University

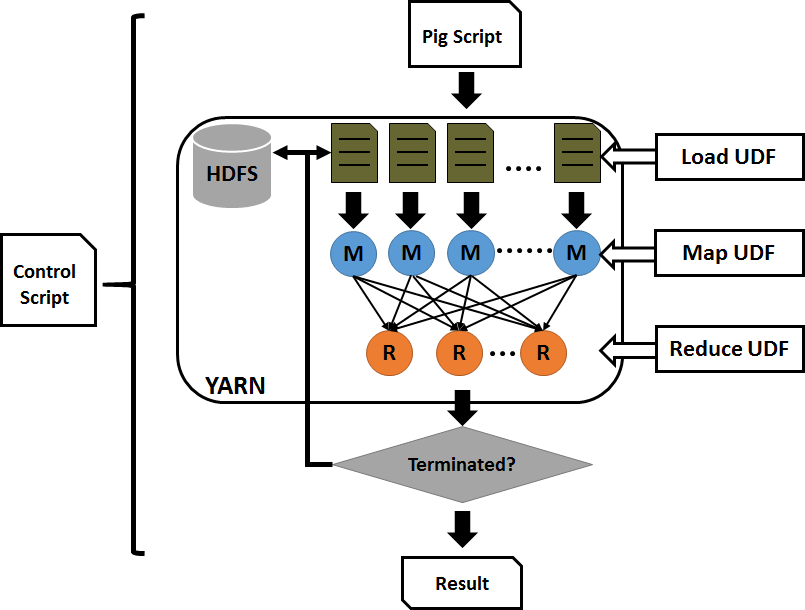
April, 22nd 2015

**Project Goal**

The purpose of this project is to write a PageRank program in Pig Latin dataflow language and report our observations on its overall performance vs. MapReduce.

## Introduction:

## Pig Latin is a dataflow language. Unlike other high level programing languages, it does not have control flow constructs such as if, else, for and while. Pig Latin provides a solution to this lack by the ability to write an external wrapper program to support writing iterations and control statements. The external wrappers can be written in Java, Python and JavaScript. In this project we use Java a control flow and we run PageRank using Pig. Figure 1 below is a schematic of iterative computing process using Pig Latin

  
  
Figure 1: Iterative computing using Pig Latin [1]

**Pig PageRank Process**

The source code in this project consists of a Java wrapper class PageRank.java, a customized loader user defined function CustomLoader.java and a pig PageRank script. The Java wrapper class links data between iterations, the user defined function serves as data loader for clueweb09 data format and the pig scripts contains the PageRank algorithm.

The following pig script calculates the new pagerank.

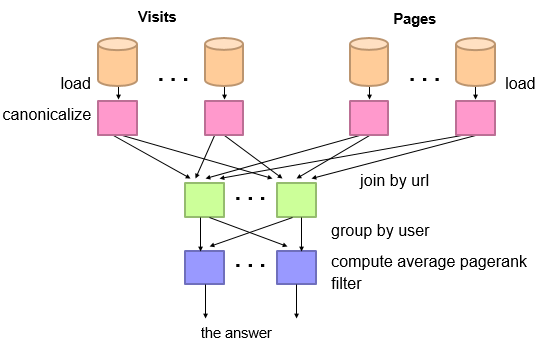
*(1-$dampingFactor) / $noOfURLs + $dampingFactor \* (SUM(previous\_pagerank.pagerank) is null ? 0 : SUM(previous\_pagerank.pagerank)) as pagerank*

The code above also checks if the SUM(previous\_pagerank.pagerank)is null. If it is it sets it to 0 as such *(SUM(previous\_pagerank.pagerank) is null ? 0 : SUM(previous\_pagerank.pagerank))* to avoid arithmetic error.

**Program compilation and execution:**

After compiling the program and putting input into HDFS, we executed the program passing 50000 as total number of urls, 0.15 as dumping factor, 2 as number of iterations, 50kinput as input directory, 50krun1 as output directory and finally 4 as number of running reducers.

**Pig Script Dataflow**

****

1. REGISTER PigPageRank.jar;  
   The above script loads Pig UDF for loading the data.
2. In the next step, we complete the Pig configuration using the following commands:  
   SET default\_parallel $parallelism;  
   SET pig.noSplitCombination true;
3. original\_data = LOAD '$inputFile' USING CustomLoader('$noOfURLs','$iteration') as (source:chararray,pagerank:double,out:bag{});  
   The above command then loads the adjacent matrix into original\_data outer data bag with specified data structure. In the project the source is chararray,pagerank:double,out:bag{}.
4. previous\_pagerank = FOREACH original\_data GENERATE FLATTEN(out) as source,pagerank/SIZE(out) as pagerank;  
   The above command prepares an outgoing link pagerank in records by getting the last column of original\_data as (source, self pagerank/SIZE(out)).
5. cogroup\_graph = COGROUP original\_data by source , previous\_pagerank by source OUTER;  
   This command groups all the same url into a (source, {original\_data#m, previous\_pagerank#n})
6. new\_pagerank = FOREACH cogroup\_graph GENERATE group as source, (1-$dampingFactor) / $noOfURLs + $dampingFactor \* (SUM(previous\_pagerank.pagerank) is null ? 0 : SUM(previous\_pagerank.pagerank)) as pagerank, FLATTEN(original\_data.out) as out;  
   This command will generate source, calculated new pagerank, out in comma list foreach cogroup\_graph. Normally, the formula to calculate pagerank is:  
   1-$dampingFactor) + $dampingFactor \* SUM(previous\_pagerank.pagerank)  
   But in this case, we check if the SUM(previous\_pagerank.pagerank) is NULL, if true the value is set to 0, otherwise it is set as SUM(previous\_pagerank.pagerank).
7. The calculated pagerank is stored into the outputFile by using the following command.  
   STORE new\_pagerank INTO '$outputFile' ;

**Observations:**

The following is a summary of some observations we have noticed when running Pig PageRank vs. MapReduce PageRank:

* Pig Latin easy to read
* Development with pig is faster
* Fewer lines of code when using pig
* We have noticed a slightly faster running time with pig. Thus better performance.