# CIS 833 Programming Assignment 2: PageRank in Spark

Jordan DeLoach

November 18, 2016

## 1. Architecture

### 1.1. Functionality

The programming assignment is completed in one Scala file, PageRank.scala. It leverages Spark, and namely the Resilient Distributed Dataset-based APIs. The only requirements to run the code is providing an input folder of Wikipedia pages, with each page being a single line contains a title and body. The body contains links of, or similar to, the format "[[]]." These, as the edges, alongside with their corresponding title pages, as the vertices, form the graph upon which the PageRank algorithm is ran.

### 1.2. Map Reduce Jobs

While several "map" or "reduce" jobs are leveraged, I will describe the major components that would correspond to the operations in a Apache MapReduce-like execution environment.

Within `loadWikipedia`, the input files are loaded into an RDD. That RDD is then flat mapped, removing pages with semi-colons in the title, emitting a tuple of `(title,body)`.

Next, `generateLinkGraph` has what would be akin to two map-reduce operations. First, the RDD of titles and bodies from above is used as the input to a flat map operation which emits many tuples of the form `(outgoing,INTEGER,title)`, where INTEGER varies between 0 and 1, as described in the Cloud9 algorithm in the homework description. That map operation is then reduced via a `groupByKey` operation, which acts as an input to a second flatMap which takes the list of form `(pageKey,(INTEGER,incomingLink),...)`. This flat map operation ensures that the destination page of each link exists in the link graph, and emits all distinct (page,outgoingLink) combinations from each list. This map operation is then reduced by a second `groupByKey` operation which creates for a second time the adjacency lists for each page.

The actual execution of the PageRank algorithm is just a singular set of Map Reduce-like operations. An input of a page, existing PageRank score, and outgoing links are used as input to a mapper which emits tuples corresponding to the link contribution, $\frac{PR(Y)}{n}$, for every outgoing link of this page. That mapper output is then reduced by summing up the individual contributions and the PageRank equation is then used to generate the next iteration of the PageRank score for each page.

### 1.3. Data Types

Various different data types are used throughout the process. In `loadWikipedia`, the key/value pair of title/body is used. In `generateLinkGraph`, the input of title/body is used, with the intermediate key of `(linkDestination,(int,linkSource))`, and the method output of `(page,outgoingLinks)`. In the execution of the PageRank algorithm, the key/value pairs of of `(title,(prY,outgoing))` are taken as inputs to a mapper which emits tuples of `(outgoingLink,contribution)`. Those contributions, for each of the same outgoing links, are reduced by summing, and then mapped into pairs of the page and the associated page rank.

### Cluster Execution (Beocat)

*jdeloach@gremlin00: $ spark-submit –class pa2.PageRank –master yarn ... pageRank.jar WikiProject/*

```
jdeloach@gremlin00: $ hadoop fs -get pageRank_14790366884982614.out
jdeloach@gremlin00: /pageRank_14790366884982614.out$ cat part*

(France,48.04075450266202)
(Germany,46.56285477707166)
(India,41.897947813729346)
(Latin,40.536699826478454)
(Italy,36.834885686174545)
(Europe,28.27378084005861)
(China,25.755054111341174)
(Islam,25.62228620746836)
(Netherlands,25.246141063079293)
(Canada,24.190459009789095)
```

## 2. Reflections

### 2.1. Reflections

This assignment forced me to be very cognizant of the differences in execution strategy between MapReduce and Spark, and translate the description of how things should work in MapReduce as given in the homework description, to the corresponding paradigm for Spark. Additionally, this assignment forced me to be more focused on the specific benefits and drawbacks of certain transformations and actions available in Spark, for instance leveraging flatMaps to either emit multiple results (as done in `generateLinkGraph`), or leveraging it to emit one or zero results as in `loadWikipedia`. This project was also the first time I had leveraged the RDD `join` operation. Open issues with this project include finding where the slight inaccuracies in final PageRank scores that exist emanate from. My PageRank scores are generally a couple of percentage points off of the samples provided in the project description, generally falling a bit higher than the sample values (e.g. 277 versus 250 for Europe).

### 2.2. Time Spent

The amount of time taken to complete this assignment was around 8-10 hours. The initial version of the code took around 3-4 hours, with the last several hours going towards refinements and documentation. The PageRank algorithm itself was quite easy to construct, while the most time for the initial version was spent researching the way links are used in Wikipedia and deciphering which links I need to include, and under what title.

The most challenging part of this assignment was the implementation of the logic to remove dead links without ever fully caching the index of pages. It took me several attempts to understand how the algorithm worked, as well as how to translate from the MapReduce steps into a paradigm that would be applicable in Spark.

### 2.3. Testing

I tested my code namely by check pointing at different parts and ensuring it met with intermediary values as provided in the homework description or as in the online discussion forms, including checking the correct corpus size and ensuring the correct adjacency lists that were provided. After that, I namely just kept running the sample `scowiki` dataset, and attempting to match my results to as close as possible with the solution, eventually finding mistakes like forgetting to divide the contribution by the number of outgoing links, e.g. originally I forgot the denominator in $\frac{PR(Y)}{n}$.

## 2.4. Results

**Scowiki**:
```
(Europe,277.39474199008475)
(Unitit Kinrick,168.4234053083549)
(Inglis leid,144.98180784494696)
(Unitit States,138.2098593343058)
(Ingland,137.472983250761)
(Germany,130.57942301188467)
(The Yird,130.40532729797079)
(Asie,127.83470018888599)
(European Union,124.53953566758108)
(Scotland,123.9619069615278)
```

**Afwiki**:
```
(Suid-Afrika,96.94151635249288)
(Verenigde State van Amerika,88.04329215968755)
(Frankryk,73.82254816363663)
(Gregoriaanse kalender,53.456076187890034)
(Engels,53.219869642276905)
(Duitsland,51.25295371379678)
(Verenigde Koninkryk,49.84045742855917)
(Rome,49.25925279888253)
(Itali,49.010156693169925)
(Rooms-Katolieke Kerk,44.31794464458442)
```

The results for the `scowiki` make sense as those are the most general concepts, which both link to many other pages and are linked to by many, making them pages that can be reasonably considered trustworthy. The results for `afwiki` are a bit harder to decipher due to a further language barrier, but from some lookups of terms, they match intuitive expectations. The 15GB dataset ran on Beocat are a few sections up.

## 2.5. Acknowledgements

The Apache Spark API was heavily utilized as well as StackOverflow for niche one-liner solutions, but other than that, only class slides were utilized.