Giraph implementation of Nutch LinkRank Algorithm - Ahmet Emre Alada

Introduction

Current Development

[a] https://github.com/AGMLab/giraph/
[b] https://github.com/AGMLab/giraph/tree/trunk/giraph-examples/src/main/java/org/apache/giraph/examples/LinkRank
[c] https://github.com/AGMLab/giraph/blob/trunk/giraph-examples/src/test/java/org/apache/giraph/examples/LinkRankVertexTest.java

[d] https://issues.apache.org/jira/browse/GIRAPH-584

Problem

LinkRank Scoring mechanism in Apache Nutch 1.x currently works in pure map-reduce pattern. Moreover, Apache Nutch is not optimized for graph-processing operations. Due to this nature of Nutch, scoring calculation could have been more efficient if done by a graph-processing library that runs with Bulk Synchronous Parallel model. Moreover, Apache Nutch 2.x which has slightly different architecture than 1.x, lacks LinkRank scoring. So rather than porting it to the new architecture, a cross-version solution would be nice to have.

Solution

My objective is to implement parallelized LinkRank via Apache Giraph library so that:

1. The scores can be computed with Bulk-synchronous processing model more efficiently and suitable to the nature of the algorithm.
2. Nutch code will be lesser and clearer.
3. It's modular
4. LinkRank would be re-usable across many applications including Solr. This way we can also integrate LinkRank to Apache Nutch 2.x which doesn't have LinkRank implemented yet.

Why my solution is different

1. It's simpler since it will be implemented with bulk-synchronous processing mode, not the basic map-reduce.
2. You will be able to run LinkRank from any application you want.
3. You will be able to get both Nutch 1.x and 2.x to use LinkRank with appropriate proxy plugins.

Why is Scoring Important to Nutch?

Scoring has been one of the most important modules of Apache Nutch project. When crawling the web, Nutch will have to decide which links to fetch in the next cycles. To achieve a good crawling, Nutch has to choose the links leading to the most important pages so that the quality of the crawl database is higher.

Scoring Mechanisms in Nutch

In Nutch 1.x, there are three scoring methods implemented: LinkRank, OPIC and Depth-Limited Scoring. LinkRank is the favored one with its relatively high accuracy. It is a derivation of the popular PageRank Algorithm by Google. The differences between the two algorithms are:

1. LinkRank has an optional Loop Detection & Removal phase. This is important in detecting and discarding spam pages but very expensive so it's not widely used.
2. You can remove the duplicate links and assume they are just one link.
3. In the PageRank, damping factor is fixed as 0.85. In LinkRank it's the same but it can be configured via link.analyze.damping.factor parameter.
4. PageRank runs until it converges. LinkRank will run a pre-defined number of iterations (default is 10).

In Nutch 2.x, only OPIC has been implemented and LinkRank is the next step in scoring algorithms. Since writing LinkRank for this branch requires a great deal of effort, we thought it would be cleaner and more efficient to write LinkRank in Apache Giraph library.

In Nutch 1.x, we know that LinkRank reads from the WebGraphDB, makes a clone of it. Then it inverts outgoing links, creating the inlink database. Then at each iteration, each vertex sums the score from its neighbors: score(neighbor_i)/outdegree(neighbor_i), resulting in its new score. Then it updates the clone WebGraphDB with its new score. After 10 iterations, the resulting clone WebGraphDB is replaced with the original WebGraphDB and the it's done.

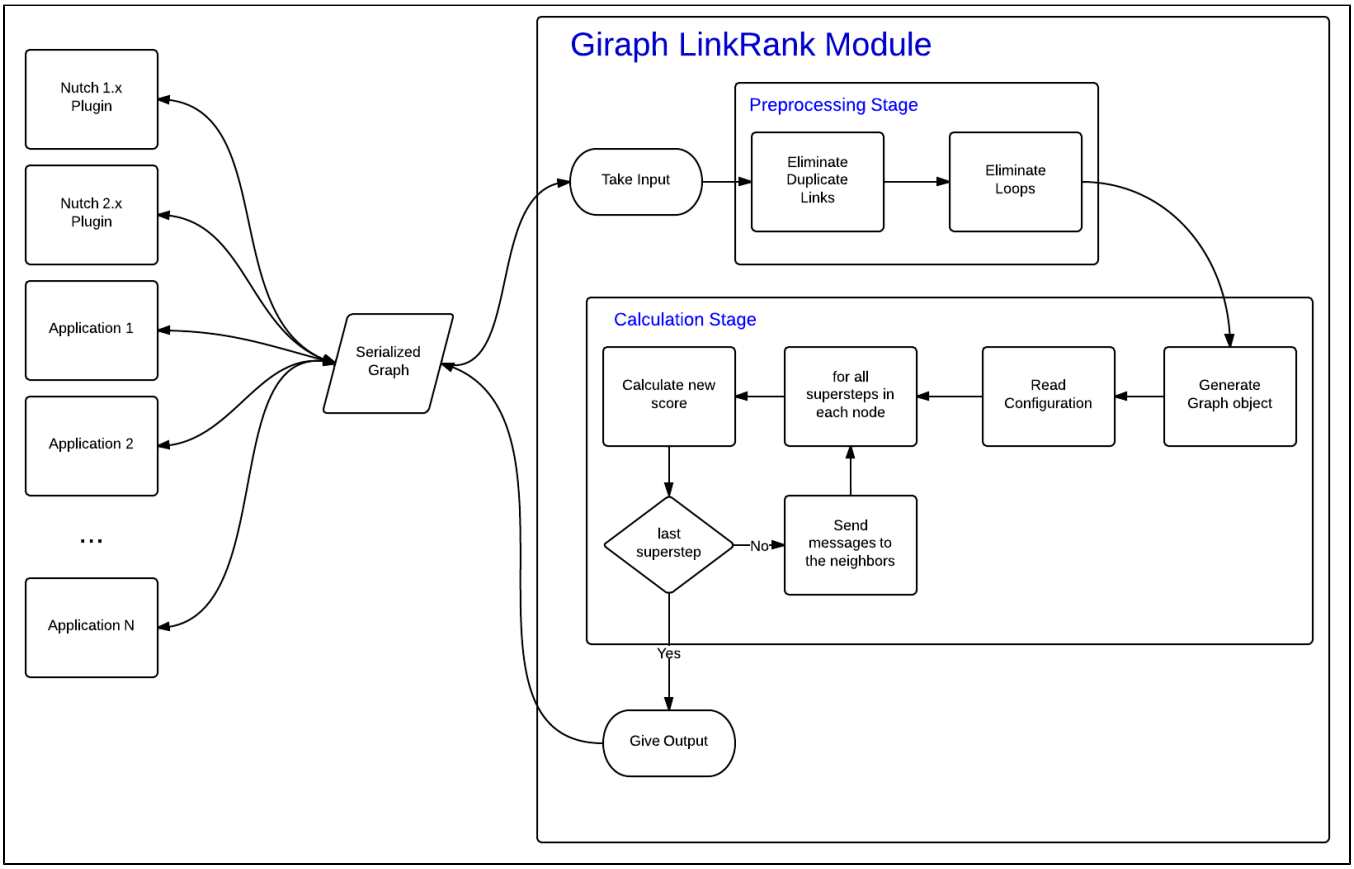
Apache Giraph Library

Apache Giraph is a distributed graph processing library in incubating stage. Since graph algorithms may take a great deal of running time, running them in a parallel fashion would save us lots of time. So that's where Apache Giraph comes in. Giraph follows bulk-synchronous parallel mode where each vertex in the graph performs a task and sends a message to its neighbors at each superstep. Graphs are partitioned and distributed to Hadoop nodes and each node performs the calculations assigned to their vertices.

Giraph implementation of Nutch LinkRank Algorithm

LinkRank is suitable for running parallel in supersteps. In each superstep, each node will calculate its new score and send messages to its neighbors. By partitioning the graph on Hadoop, we can implement LinkRank to run parallel.

In my implementation, there will be a Giraph LinkRank Module which will be a generic module that can be used by several applications. It will take input in a serialized graph format and assign scores to the nodes in the graph. Then will return the resulting graph in a serialized format which will be evaluated by each application that is using this module. There will be Nutch 1.x and 2.x proxy plugins which take the responsibility of converting the crawl database into /from the serialized graph format and calling the Giraph LinkRank module to perform scoring on the graph. So it's modular. Below a diagram of the overall architecture can be seen.



Outline of the tasks

In order to get Giraph perform these calculations I will be completing these tasks:

1. Find sample popular graphs to work on.
2. Run PageRank on these sample graphs.
3. Design the input/output pipeline
4. Write a Nutch-independent LinkRank implemetation
    a. Duplicate Links should be avoided in the serialized graph reading stage.
    b. Loops will be eliminated optionally.
    c. We don't need to invert the links since the architecture doesn't require us to do so. Each vertex sends the score(v)/outdegree(v) to its neighbors. All a vertex has to do is to sum up these messages coming from its neighbors and calculate the new score for the next superstep.
5. Write tests with the sample graphs to make sure it's working properly.
6. Write a plugin that works as a proxy to Nutch 1.x. It will convert the Nutch WebGraphDB to a serialized form which will be taken as input to Giraph Graph class. This class will contain vertices holding URL, current score and ID metadata. Then let Giraph calculate LinkRank scores and update WebGraphDB with the new scores outputted by Giraph.
7. Compare the existing LinkRank results with the new LinkRank Results and make sure they are close enough. If not, investigate the reasons and find an appropriate solution.
8. Write a plugin that works as a proxy to Nutch 2.x via Apache Gora.

Problems I may encounter

1. New LinkRank implementation might not produce the scores of the existing implementation and it may require some effort to obtain the same results. I will be debugging for solving this problem.

2. Existing Nutch scoring filter architecture doesn't seem to be compatible with bulk-synchronous processing model so I will be writing a generic plugin instead of scoring plugin. Scoring filter interface forces you to work in a specific manner but in my case we will be flexible since the deciding mechanism will be Giraph LinkRank module.
3. Defining a generic serialization format might need some thought. I will consult the list for their suggestions.
4. Incompatibility between Nutch 1.x and 2.x will be problem. Nutch 1.x uses HDFS so I can read through it but in Nutch 2.x, I will have to use Gora to access Hbase. This will require different proxy plugins but it's easier to write different serializer plugins than writing the whole scoring mechanism with different APIs.

Working Schedule

Stage	Date Range	Task	Status
1	18.03.2013 - 27.03.2013	Practice crawling with Nutch 1.x and 2.x and Searching through SolrCloud.	DONE
2	28.03.2013 - 01.04.2013	Practice Hadoop and Mapreduce	DONE
3	02.04.2013 - 06.04.2013	Read on WebGraph, LinkRank and ScoringFilter mechanism	DONE
4	07.04.2013 - 15.04.2013	Write sample scoring plugins for Nutch 1.x and 2.x and debugging	DONE
5	16.04.2013 - 24.04.2013	Practice with Giraph. Write sample PageRank code from scratch and modify it	DONE
6	06.05.2013 - 01.06.2013	Run PageRank on sample graphs, practice more with Giraph	DONE
	Milestone 1	**Discovering & Learning**	
7	03.06.2013 - 07.06.2013	Design Graph Metadata Design	DONE
8	10.06.2013 - 14.06.2013	Duplicate Link Removal	DONE
9	17.06.2013 - 21.06.2013	Design input/output pipeline and serialization	DONE
10	24.06.2013 - 28.06.2013	Write Tests to make sure it's working properly.	DONE
	Milestone 2	**Generic LinkRank with Giraph**	
11	01.07.2013 - 05.07.2013	Read more on Nutch 1.x plugin mechanism	IN PROGRESS
12	08.07.2013 - 11.07.2013	Write Nutch 1.x proxy plugin	
13	15.07.2013 - 19.07.2013	Test Nutch 1.x - Giraph Integration	
14	22.07.2013 - 26.07.2013	Make sure original LinkRank and mine produces the same results	
	Milestone 3	**Nutch 1.x Integration**	
15	28.07.2013 - 02.08.2013	Learn how to use Gora for accessing the scores in Nutch 2.x	
16	05.08.2013 - 09.08.2013	Read more on Nutch 2.x plugin mechanism	
17	12.08.2013 - 16.08.2013	Write Nutch 2.x proxy plugin	
18	19.08.2013 - 30.08.2013	Test Nutch 2.x - Giraph Integration	
	Milestone 4	**Nutch 2.x Integration**	
19	02.09.2013 - 06.09.2013	Loop Elimination	
20	09.09.2013 - 13.09.2013	Testing Loop Elimination	
21	16.09.2013 - 20.09.2013	Community Testing & Review & Writing Report	
22	23.09.2013	Final Deliverable	
	Milestone 5	**Final Milestone**	
	Future Work	Improvements on LinkRank: similar and better versions of LinkRank.	

About Me

I'm Ahmet Emre Alada, a 4th semester PhD Student in Boaziçi University, Istanbul, Turkey. My research interests are Complex Network Analysis (Ranking algorithms, Influence networks, Information Spread, Finding the most influential person/page), Information Retrieval (Crawling, search engines, ranking the web pages via graph-theoretic measures and pattern recognition methods given implicit feedback.). I have taken Complex Networks, Information Retriveal, Aritficial Intelligence, Machine Learning courses that could be related to this project.

In the Masters (GPA 4.00), I had 1 conference publication [1] on Visualization of Protein Interaction Networks and 2 journal publications on highly reputable Oxford Bioinformatics Journal on the topics Clustering, Aligning and Visualizing Protein Interaction Networks [2] [3]. I have also taken Advanced Algorithms on graphs and Parallel Programming courses.

I had my (non-GSoC) internship in the Pardus Linux project (which was also involved in GSoC) and developed a Package-Content Search Engine and Multi-System Installation system for Pardus Linux. I have been a Linux user and Free Software Contributor since 2006. I contributed several Django applications and developed open source projects on github/bitbucket. I used mostly Python, Java and some C for my projects.

I've been reading Hadoop and MapReduce for a while and will keep on learning and coding for this project. I have some initial practice with several Apache projects including Nutch, Solr, HBase, Hadoop, Giraph.

More about me

About Me

My Resume

Github Repo

Bitbucket Repo

My Commitment

Currently, I'm working for a R&D company where I'm given the position for developing an efficient and precise ranking algorithm. We will be using Nutch 2.x and it's to-be-implemented LinkRank scoring so they support me in contributing Nutch community. I will be working on this project in my working hours at the office and also at home. My company and our partners have been contributors to the Nutch project for some years. Moreover, my research area in my PhD studies is detecting the most important person/page on a network. So it will be very convenient and joyful for me to work on this project. Contributing to a project of Apache foundation is an honour for me.

[1] A.E. Aladag, C. Erten, M. Sozdinler, ”An integrated model for visualizing biclusters from gene expression data and PPI networks”, Proc. International Symposium on Biocomputing, no.24, 2010.
[2] A.E. Aladag, C. Erten, M. Sozdinler, ”Reliability Oriented Bioinformatic Networks Visualization”, Bioinformatics, vol 27, pp. 1583-1584, 2011.
[3] A.E. Aladag, C. Erten, ”SPINAL: Scalable Protein Interaction Network Alignment”, Bioinformatics, vol 29, pp. 917-924, 2013.