Compare Graph Management Systems for Various Queries

Final Report

Ziran Li
Boston University
zrli@bu.edu

Zulin Liu
Boston University
liuzulin@bu.edu

Zhiyi Yang
Boston University
yzy123@bu.edu

Project GitHub link: https://github.com/yzy123/CS562

Abstract

In our project, three graph-processing algorithms including will be conducted on the tree platforms (Graphlab, Giraph and GraphLite) to evaluate the performance of different graph-process database systems from certain perspectives. The statics retrieved will be utilized to draw tables and graph for more direct comparison. However, there are also some test failure in the process which will be listed in details and the analysis of what had caused those failure. Finally, a summary conclusion about the project will be drawn.

1. Introduction and Motivation

A number of graph-parallel processing frameworks have been proposed to address the needs of processing complex and large-scale graph structured datasets in recent years. Since significant performance improvement made by those frameworks were reported, our motivation is to compare advantages and disadvantages of each of these frameworks over the others, and to find the best utilization of those frameworks for a specific graph computing task and setting. Specifically, we learned GraphLab, Giraph and GraphLite on Hadoop and compared GraphLab with GraphLite.

2. Problem Definition

In order to achieve the performance comparison of the frameworks over the others, several problems will be defined and answer as follows.

2.1. What resources (i.e. database platforms, datasets) are available?

Database Platform: Three existing offline graph analytic systems including Graphlab, Giraph [1] and GraphLite [2] will be utilized to perform an iterative, batch processing over the entire graph dataset until the computation satisfies a fixed-point or stopping criterion. Efficient algorithms on top of them to query and analyze large datasets, processing complex and large scale datasets shall be implemented.

Datasets: Datasets comes from Stanford Large Network Dataset Collection [3], including Enron email network, Wikipedia vote network, YouTube social network and ground-truth communities, Friendster social network and ground-truth communities. The specific information of dataset is shown in section 3.2.

2.2. What specific algorithms should we utilize on top of systems to analyze datasets?

All three important graph-processing algorithms including PageRank, Shortest Path and Cluster Coefficient (Triangle Counting) will be conducted on three platforms mentioned above respectively to check whether there is a prior graph management system among them or they would have their own advantage by applying certain performance measurement methods.

2.3. What parameters can we use to evaluate performance of each target system?

In this project three factors including time complexity, resource utilization and scalability will take into consideration to evaluate the performance of three platforms [1].

Time complexity: the ability of a platform to (quickly) process large-scale graphs.

Resource utilization: the ability of a platform to efficiently utilize the resources it has.

Scalability: the ability of a platform to maintain its performance behavior when resources are added to its infrastructure.

2.4. How to control variable among different database platforms, different test datasets and different algorithms?

We made some small modification on the three graph-processing algorithm embedded in the different graph management systems to keep the consistency among our evaluation.

2.5. What factors can reflect the differences between test datasets?

Since all our datasets are graphs, the size of graph (number of edges and vertices) and whether the edge is weighted or unweighted can reflect the difference between test datasets.

3. Method Description

As mentioned in the problem definition, 3 graph-processing algorithms including PageRank, Shortest Path and Cluster Coefficient (Triangle Counting) will be conducted on the tree platforms (Graphlab, Giraph and GraphLite) to evaluate the performance from time complexity, resource utilization, and scalability respectively. The statics retrieved will be utilized to draw tables and graph for more direct comparison , which will reflect whether those database systems have efficient ability to process graph.

3.1. Experiment platform

GraphLab: The GraphLab framework is a parallel programming abstraction targeted for sparse iterative graph algorithms. It provides a high level programming interface, allowing a rapid deployment of distributed machine learning algorithms.

Giraph: Apache Giraph is an Apache project to perform graph processing on big data. Giraph utilizes Apache Hadoop's MapReduce implementation to process graphs. Facebook used Giraph with some performance improvements to analyze one trillion edges using 200 machines in 4 minutes. Giraph is based on a paper published by Google about its own graph processing system called Pregel. It can be compared to other Big Graph processing libraries such as Cassovary.

3.2. Experiment environment

The experiment environment is a virtual environment on MacBook Pro OS 10.13.4.

Processor: 3.1 GHz Intel Core i5
Memory: 8 GB 2133 MHz LPDDR3
Graphics: Intel Iris Plus Graphics 650 1536 MB

3.3. Experiment datasets

We intentionally chosen four datasets with significant different on their size to demonstrate the scalability of graph management systems. Our datasets are come from Stanford Large Network Dataset Collection [3]:

1. Enron email network: (Nodes: 36692 Edges: 367662):

Enron email communication network covers all the email communication within a dataset of around half million emails. [4]

2. Wikipedia vote network (Nodes: 7115 Edges: 103689):

The network contains all the Wikipedia voting data from the inception of Wikipedia till January 2008. Nodes in the network represent Wikipedia users and a directed edge from node i to node j represents that user i voted on user j. [5]

3. YouTube social network and ground-truth communities (Nodes: 1134890 Edges: 2987624):

In the YouTube social network, users form friendship each other and users can create groups which other users can join. We consider such user-defined groups as ground-truth communities. [6]

4. Friendster social network and ground-truth communities (Nodes: 65608366 Edges: 1806067135):

Friendster was a social networking site where users can form friendship edge each other. [7]

3.4. Data Pre-process

GraphLab: One advantage of GraphLab is it can take the popular 2-Colomn dataset (<v1,v2>) as input. But it can only deal with undirected, unweighted data. In our experiment, since data we chose are already 2-Colomn data, there is no pre-process needed.
Giraph: The advantage of Giraph is it provide a wide range of class including a lot of different input format.
GraphLite: We use MATLAB to partition data into 4 part, so that we can apply MapReduce over the dataset.

4. Results

4.1. GraphLab

For the 4 different datasets, not all datasets can be handled in the experiment. As the size of data increase to (Nodes: 65608366 Edges: 1806067135) just like Friendster social network and ground-truth communities, Graphlab return error that indicates time out during downloading target dataset. Therefore, Graphlab exhibit not that good scalability over different datasets of all the algorithms.

Table 1~3 shows the computing time results of Graphlab for three selected graph algorithms (Over three rest different datasets):

Table1

Enron email	PageRank	Shortest Path	Triangle Counting
Computing Time /second	1.830144	1.6589	0.945635

Table 2

Wikipedia vote network	PageRank	Shortest Path	Triangle Counting
Computing Time /second	1.412713	1.3592	0.3981445

Table 3

YouTube social network	PageRank	Shortest Path	Triangle Counting
Computing Time /second	4.351234	11.9762	18.288515

Table 4~6 shows the resource utilization results of Graphlab:

Table 4

Enron email	PageRank	Shortest Path	Triangle Counting
CPU Percentage	20.5	67.1	43.2
Virtual Memory Percentage	70.2	72.8	72.9

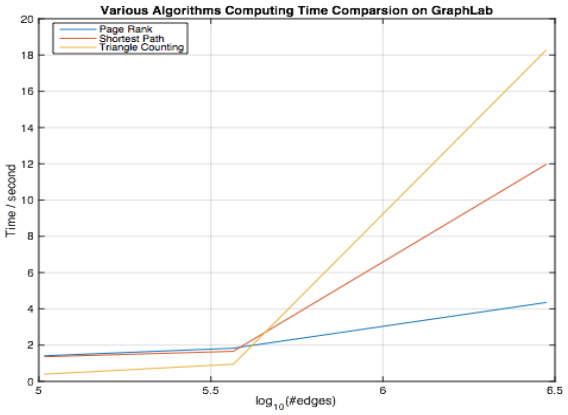
Table 5

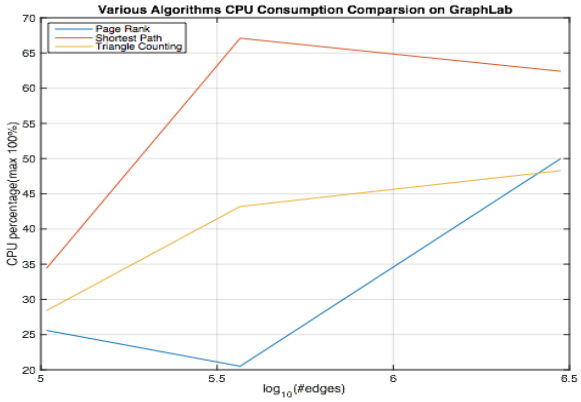
Wikipedia vote network	PageRank	Shortest Path	Triangle Counting
CPU Percentage	25.6	34.4	28.4
Virtual Memory Percentage	67.7	67.9	67.5

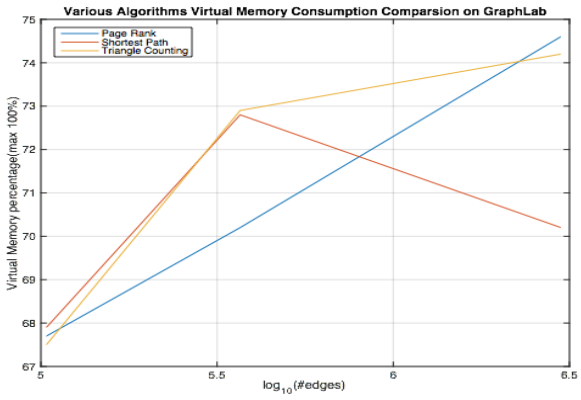
Table 6

Youtube social network	PageRank	Shortest Path	Triangle Counting
CPU Percentage	50.0	62.4	48.3
Virtual Memory Percentage	74.6	70.2	74.2

From the results above, Graphlab exhibit good scalability over different datasets of all the algorithms.







The key findings of Graph Lab are that Shortest Path will consume most CPU resource which is followed by Triangle Counting. Page Rank will consume least CPU resource.

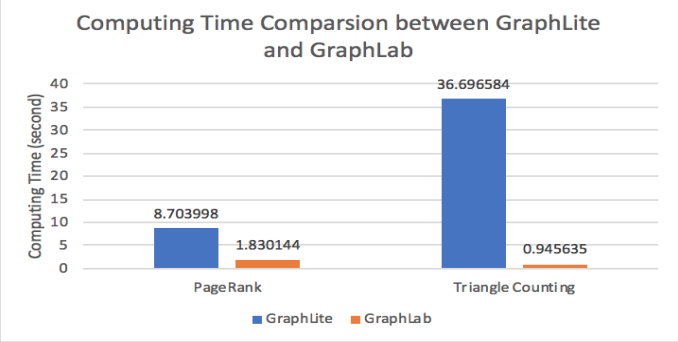
The difference of virtual memory among different graph processing algorithms is not significant.

4.2. GraphLite

For the 4 different datasets, only small size dataset can be hold, like Enron Email (nodes: 36692 Edges: 367662) as the sizes of dataset increase, the graph processing program will crush, which means that the scalability of GraphLite is not as good as GraphLab.

Enron email	Page Rank	Triangle Counting
Computing Time /second	8.703998	36.696584

4.3. GraphLab VS GraphLite



Key Findings: in terms of computing time on Page Rank and Triangle Counting Algorithm, the speed of GraphLite is much slower than that of GraphLab.

4.4. Failure on Giraph

Pregel is a most popular and representative bulk synchronous parallel (BSP) computing system, while since it is not easy to access, in our project we are using an open-source alternative: Giraph.

We deployed a single-node,pseudo-distributed hadoop cluster on our MAC. This node will act as both master and slave. However, we encountered several problem when we deployed Giraph over Hadoop.

1. We followed the instruction strictly, however, problem occurred when we are trying a SimpleShortestPathsComputation example. We found, instead of using command in the instruction:

Figure: https://giraph.apache.org/quick_start.html

If user want to use a single-node cluster, we should use:

```
$HADOOP_HOME/bin/hadoop jar $GIRAPH_HOME/giraph-examples/target/giraph-examples-1.2.0-SNAPSHOT-for-hadoop-0.20.203.0-jar-with-dependencies.jar org.apache.giraph.GiraphRunner org.apache.giraph.examples.SimpleShortestPathsComputation -vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip /user/hduser/input/tiny_graph.txt -vof org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /user/hduser/output/shortestpaths -w 1
```

```
dhcp-wifi-8021x-155-41-54-66:hadoop liuzulin$ bin/hadoop jar $GIRAPH_HOME/giraph-examples/target/giraph-examples-1.3.0-SNAPSHOT-for-hadoop-1.2.1-jar-with-dependencies.jar org.apache.giraph.GiraphRunner org.apache.giraph.examples.SimplePageRankComputation -vif org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputFormat -vip /Users/liuzulin/Downloads/hadoop/input -vof org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /Users/liuzulin/Downloads/hadoop/output/shortestpaths -w 1 -ca giraph.SplitMasterWorker=false
```

Figure: Command line copy from our terminal

Which is nothing but add "-ca giraph.SplitMasterWorker=false" .

2. The second problem is when we do the job, terminal will return error "map 100% reduce 0%". It seems it only do the map task but not reduce task. We have three speculations about this issue: Giraph runs workers as map-only jobs on Hadoop. Memory assigned can only satisfy "map" task, and there is no room for the reduce task.

We made some mistakes when we do it step by step. Since the time constraint, we are not be able to apply three algorithms on Giraph. However, from the process. We learned a lot about Hadoop and Giraph, as well as HDFS knowledge.

3. We found the error from part2 is because we deployed two different version of Hadoop with their HDFS opening. We solved the problem by stopping HDFS of one of them. However, we meet a new problem:

```
18/05/02 19:23:45 INFO job.GiraphJob: Waiting for resources... Job will start only when it gets all 2 mappers
```

Figure: error from terminal

We are still working on it.

5. Conclusion

Through our method, a basic empirical performance evolution benchmark suite including datasets, algorithms selection is developed. Our method focus 3 performances aspects on the resource: utilization, scalability, and time complexity with domain of 4 graphs whose size up to 1.8 billion edges representing graph structures for multiple application.

Use the benchmark suite, a basic performance analysis among different graph-processing platform. is conducted.

References

[1] Y. Guo, M. Biczak, A. L. Varbanescu, A. Iosup, C. Martella, and T. L. Willke. How Well Do Graph-Processing Platforms Perform? An Empirical Performance Evaluation and Analysis. In Proceedings of the IEEE 28th International Parallel and Distributed Processing Symposium, pages 395–404. IEEE Computer Society, 2014.

[2] https://github.com/schencoding/GraphLite/tree/master/GraphLite-0.20

[3] Stanford Large Network Dataset Collection: http://snap.stanford.edu/data/

[4] Dataset: Enron email network: http://snap.stanford.edu/data/email-Enron.txt.gz

[5] Dataset: Wikipedia vote network: Wiki-Vote.txt.gz

[6] Dataset: Youtube social network and ground-truth communities: com-youtube.ungraph.txt.gz

[7] Dataset: Friendster social network and ground-truth communities: com-friendster.ungraph.txt.gz