DBMS Group Project

A case study on performance of Graph Processing Algorithms

LARGE SCALE GRAPH PROCESSING

GROUP NO :- 18

14CS10013 | Pradeep Dogga
14CS10038 | Rajarshi Haldar
14CS30005 | Arijit Panigrahy
14CS30041 | Ritam Dutt
14CS30044 | Sohan Patro

Objective

The main objective of the project was large scale graph processing.

To achieve that end, our procedure was two-fold namely:

1. Explored a few graph datasets and compared different properties across different models.
2. The models / platforms were compared on both stand-alone and distributed modes.
3. Profile performance for each of the models were denoted.

Datasets

The three datasets used in the calculation and performance evaluation of different metrics are :

1) Test dataset - A small set of 5 nodes to verify the correctness of the codes.
2) Hyves dataset - A Dutch online social network to show the relationships between users. Users represent the nodes and friendships denote the edges. Url :http://socialnetworks.mpi-sws.org/data-wosn2008.html
3) Flickr dataset - The social network of Flickr users and their friendship connections, where users represent the nodes , friendships denote the edges. Url :http://socialnetworks.mpi-sws.org/data-wosn2008.html

Network properties

The three networks chosen have the following properties :

1. Undirected
2. Unweighted

Dataset	No of nodes	No of edges
Test Data	5	6
Hyve	1402673	2777419
Flickr	2302925	33140017

Graph Properties

The graph properties which were evaluated for comparison across the 3 models:

1. Degree Distribution
2. Clustering Coefficient or Transitivity
    a. Local Clustering Coefficient
    b. Global Clustering Coefficient
3. Number of triangles
4. Number of connected components
5. Page Rank

Degree Distribution

The degree distribution $P(k)$ of a network is defined as the fraction of nodes in the network with degree k. Thus if there are n nodes in total in a network and n_k of them have degree k, we have $P(k) = n_k/n$.

Algorithm:

1. Create the graph g .
2. Count the number of degrees per node.
3. Tabulate the occurrence of degrees of a particular size.

Number of triangles

The number of triangles correspond to the number of closed paths of length 3.

Algorithm:

1. Create the graph g.
2. Represent the graph in an adjacency matrix format , say A.
3. Compute B= A^3.
4. Compute the trace of B , c= tr(B).
5. c/6 gives us the total number of triangles .

Clustering Coefficient

The global clustering coefficient is defined as :

$$C = \frac{3 \times \text{number of triangles}}{\text{number of connected triplets of vertices}}$$

Algorithm:

1. Create the graph g.
2. Find the number of closed triplets (number of cycles of length 3) call a.
3. Find the number of connected triplets (number of paths of length 3).
4. Compute the value of a/b.

Connected Components

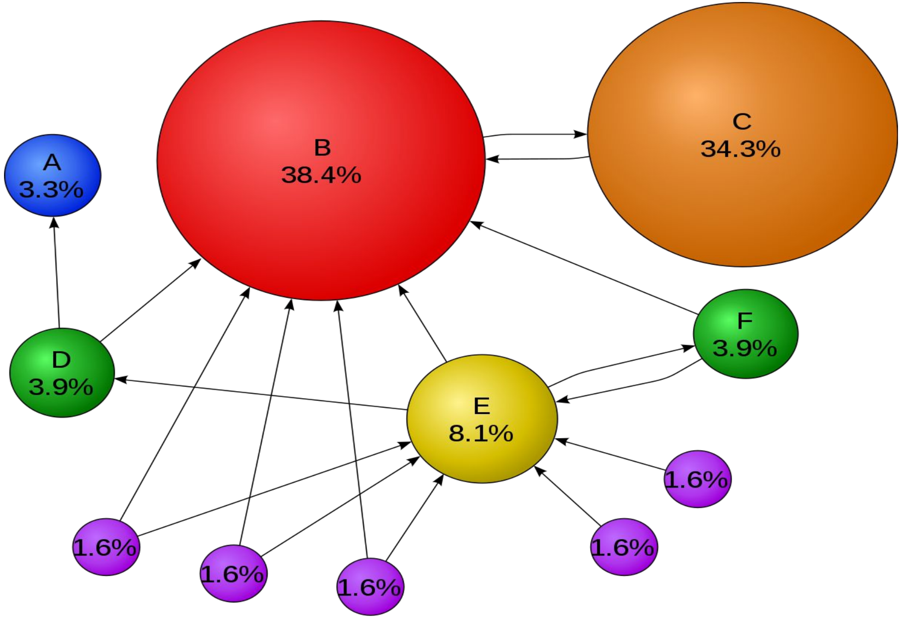
A connected component of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph.

Algorithm:

1. Create the graph g.
2. Enlist the vertices of the graph.
3. Perform BFS traversals so that all the vertices in g are included in at least one traversal.
4. The number of BFS traversals indicate the number of connected components.

Pagerank

1. In undirected graphs, pagerank is statistically close to the degree distribution.
2. Counts the number and quality of links to a node to ascertain its importance.



Formulation of Pagerank PR(A)

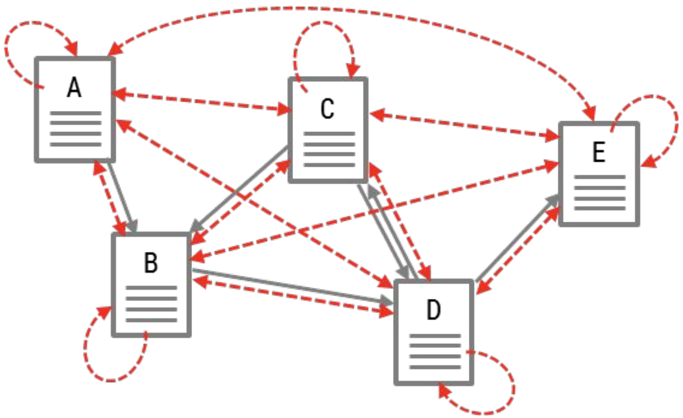
PR(A) = (1-d) / N + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

Where N is the total number of all pages on the web.

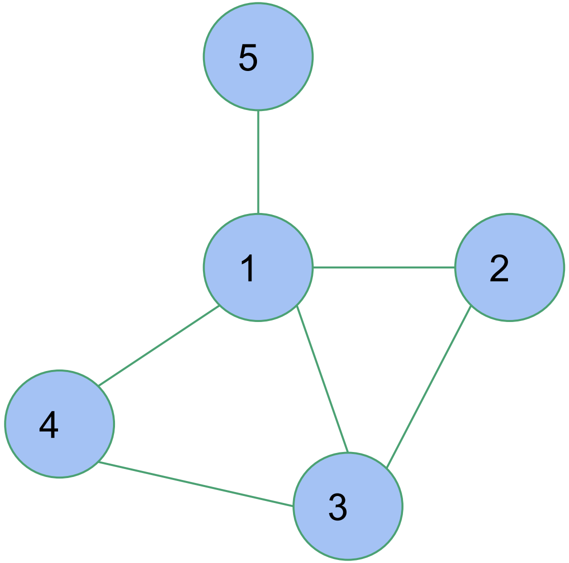
It is calculated iteratively.

The PageRank of the world wide web can be calculated in 100 iterations.

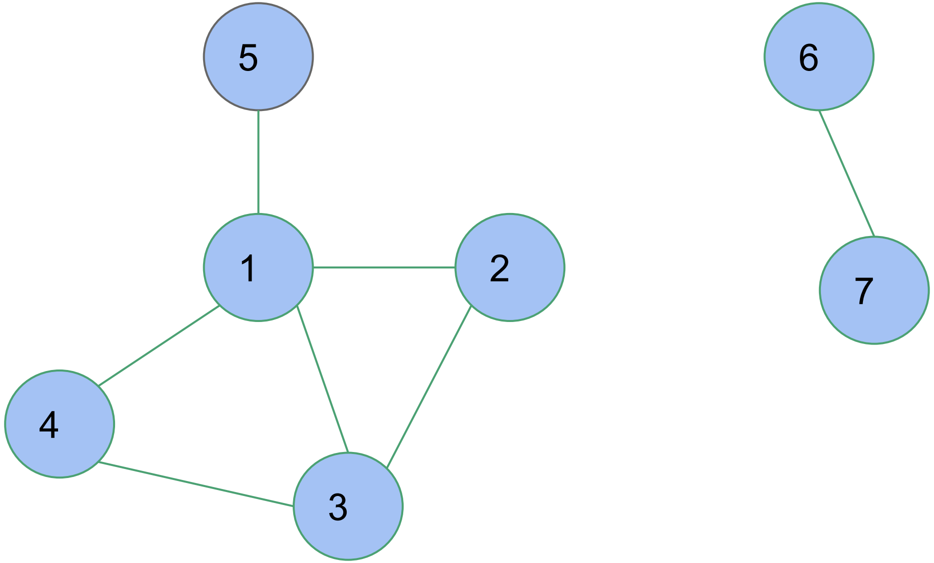
This model is called the random surfer model.



Test Network 1



Test Network 2



I. Tools Used

The requisite tools used are:

1. Python 2.7.13 using GCC 4.4.7
2. Igraph 0.7.1

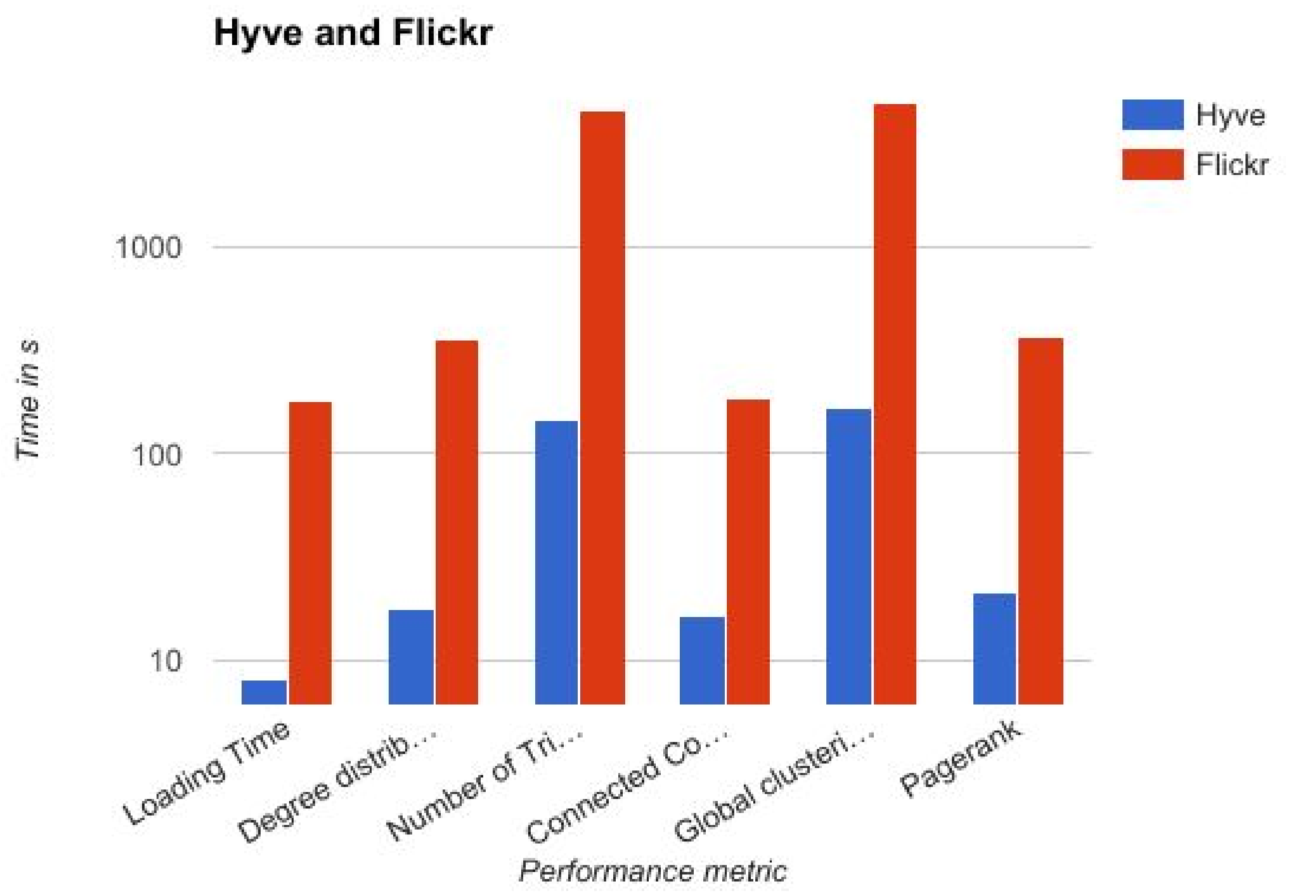
The codes were run on the server on standalone mode

Observations for Hyve

Performance metric	Value	Time taken in (ms)
Loading Time	N/A	8.29983305931
Degree distribution	3.96018031287	17.7917730808
Number of Triangles	752401	144.554925919
Connected Components	1	16.7130410671
Global clustering coefficient	0.00155978113422	164.474163285
PageRank	N/A	21.5329020023

Observation for Flickr

Performance metric	Value	Time taken in (ms)
Loading Time	N/A	180.280133009
Degree distribution	28.7808044118	360.331094987
Number of Triangles	76485	4557.89106173
Connected Components	30769	187.049488068
Global clustering coefficient	0.107647853584	4819.01817897
Pagerank	N/A	361.937635183



Hyve and Flickr
Hyve
Flickr
1000
100
10
Time in s
Loading Time
Degree distrib…
Number of Tri…
Connected Co…
Global clusteri…
Pagerank
Performance metric

Conclusion

1. Python Igraph is a wonderful application for graph processing, even on a large scale in reasonable amount of time.
2. It is highly optimized in computing extensive queries owing to the efficient use of data structures
3. However, the application runs simply on stand-alone mode and not distributed. Consequently, it is not scalable.

II. Tools Used

Hadoop 2.3.0-cdh5.1.0

Hive 0.12.0-cdh5.1.0

Why Hive over Hadoop MapReduce?

- Hive is a query language that loads the data from the HDFS and converts the queries into a series of MapReduce jobs and executes it.
- Then why Hive? Because it simplifies the task.
  For ex:- A single JOIN may take hundreds of MapReduce queries whereas its just a single line of code in Hive..
- Performance wise even Hive is better than Hadoop MapReduce. Because, the queries are optimised for the specific jobs.
  For ex: When you write the MapReduce queries for say diference, you may not be doing it in an optimal way, while in Hive, it has already been Optimised. Thus, we choose HIVE to demonstrate the MapReduce queries.

Properties Computed using HiveQL

The graph properties computed in HiveQL are :

1. Degree Distribution
2. Number of triangles
3. Global clustering coefficient

We have a table “edges(a int, b int)” to represent all the edges (between nodes ‘a’ and ‘b’) of the input graph.

- <u>Degree Distribution</u>

```
> select temp.cntb , count(temp.a) from (select a, count(b) as
cntb from edges group by a)temp group by temp.cntb
```

- Number of Triangles in the Graph

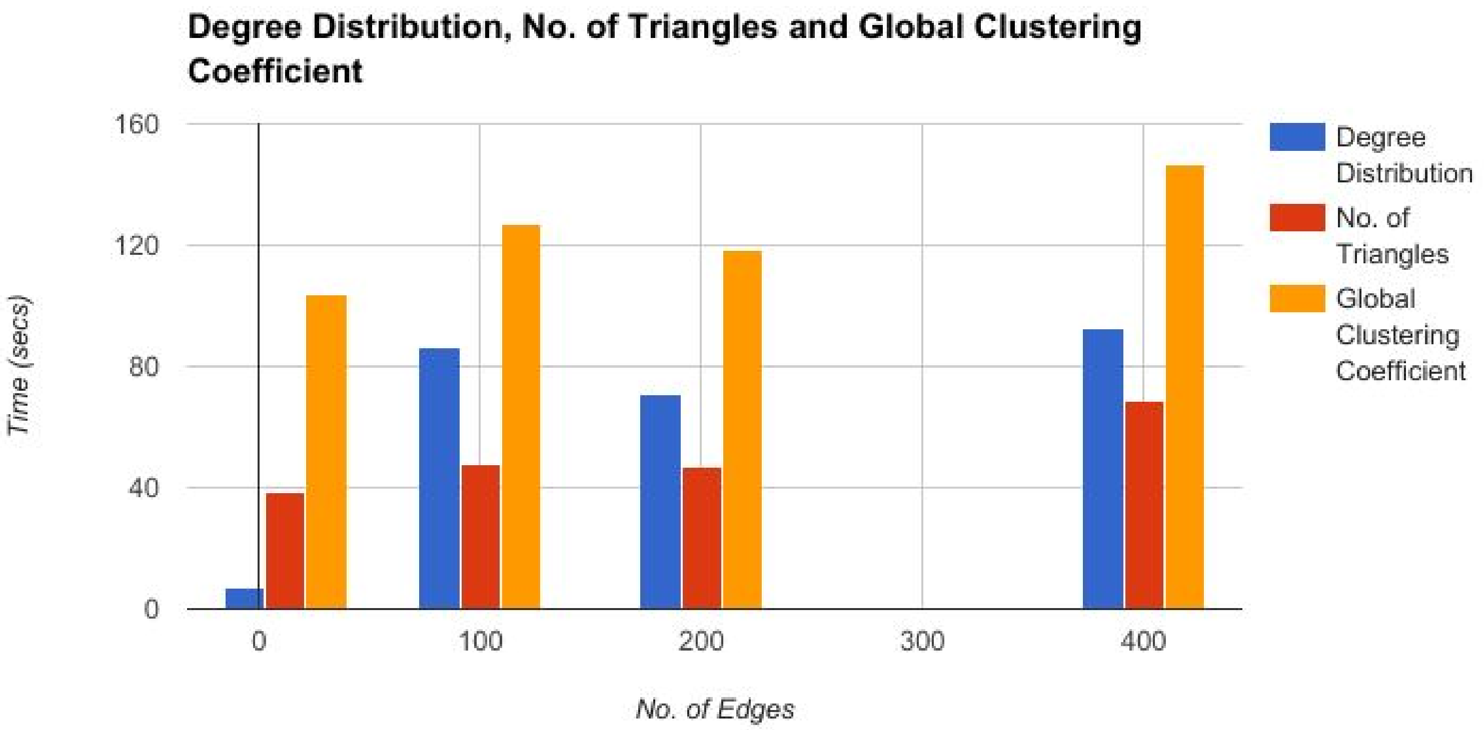
> select count(*)/6 from edges join edges as e2 join edges as e3 where edges.b = e2.a and e2.b = e3.a and e3.b = edges.a;

- The Global Clustering Coefficient

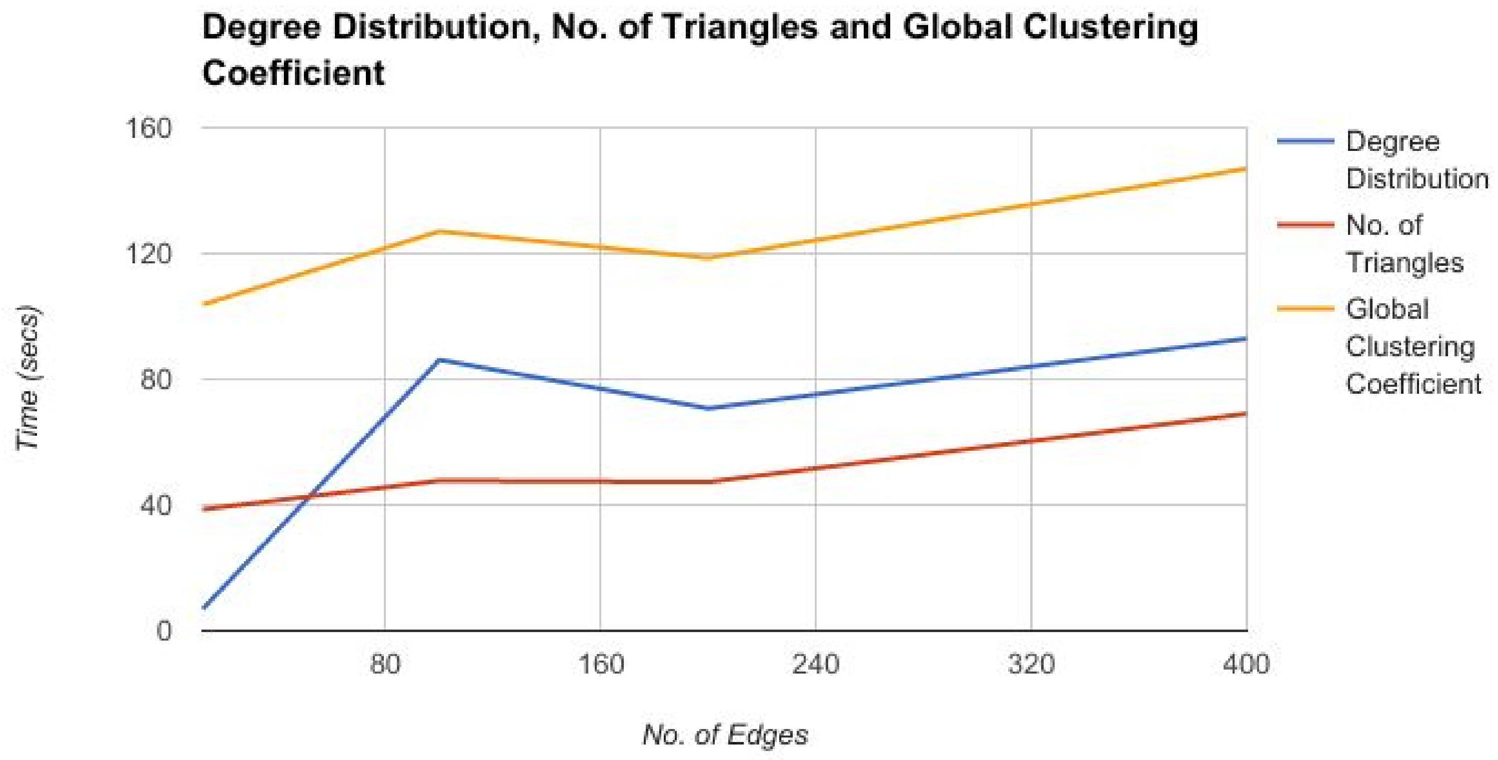
> select 3.0*n.res/d.res from (select count(*)/6 as res from edges join edges as e2 join edges as e3 where edges.b = e2.a and e2.b = e3.a and e3.b = edges.a) as n, (select count(*)/2 as res from edges join edges as e2 where edges.b = e2.a and edges.a <> e2.b) as d;

Observation

No of edges	Degree Distribution (in s)	No. of Triangles (in s)	Global Clustering Coefficient (in s)
12	6.899	38.588	103.723
100	86.163	47.726	127.045
200	70.726	47.301	118.557
400	92.906	69.013	146.997



Degree Distribution, No. of Triangles and Global Clustering Coefficient
160
120
80
40
0
Time (secs)
0
100
200
300
400
No. of Edges
Degree Distribution
No. of Triangles
Global Clustering Coefficient


Degree Distribution, No. of Triangles and Global Clustering Coefficient
Time (secs)
160
120
80
40
0
80
160
240
320
400
No. of Edges
Degree Distribution
No. of Triangles
Global Clustering Coefficient

Conclusion

1. HiveQL doesn’t support recursion or recursive queries, as of yet.
2. Consequently queries like pagerank or finding the number of connected components cannot be computed via this method.
3. Not being a Graph- Processing Tool, Hive’s processing time is seen to be very large as opposed to other Graph- Processing tools like igraph.

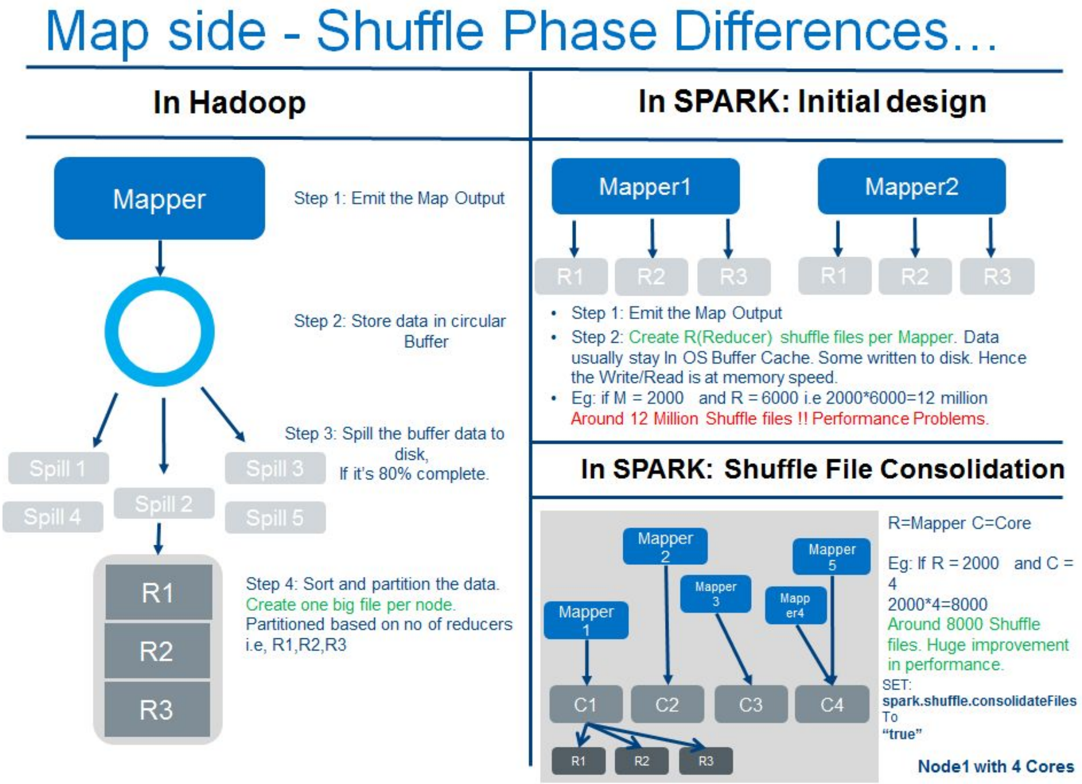
III. Tools Used

The requisite tools used are:

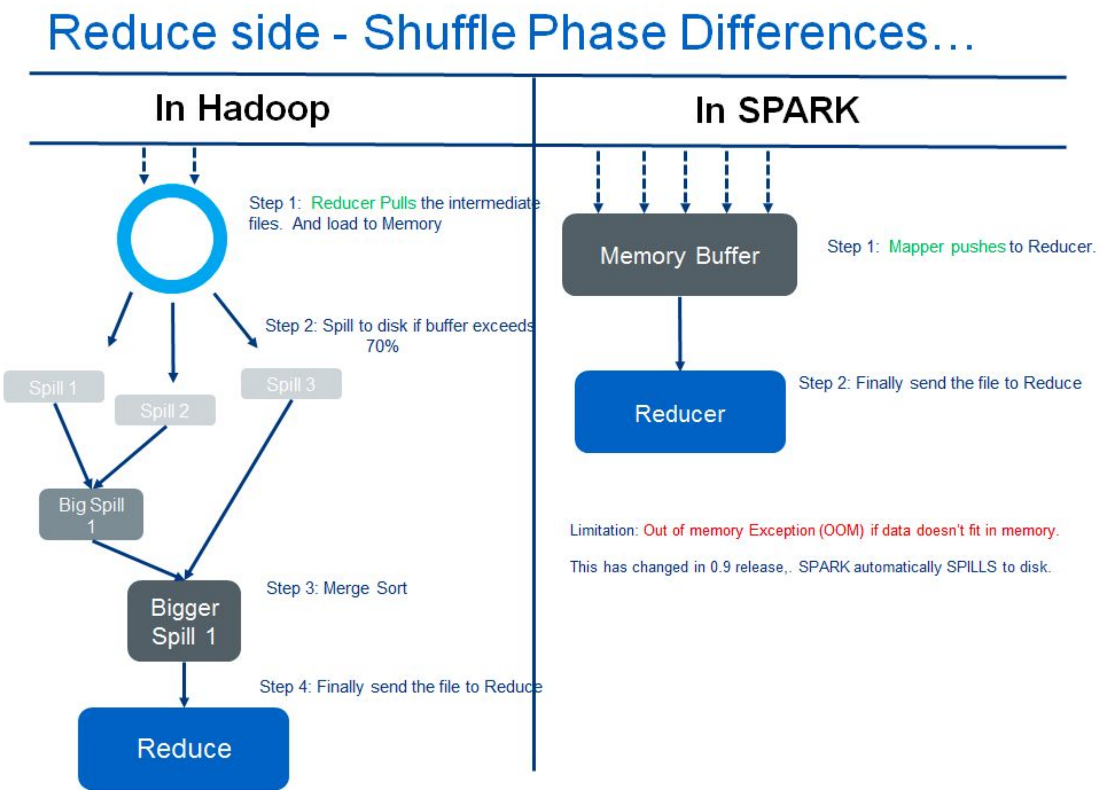
1. Scala 2.11
2. Spark 2.1.0
3. GraphX library

The codes were run on the server on distributed mode.

Map Side difference



Reduce Side Difference



Spark vs Hive

- Performance:
  Spark processes data in-memory while MapReduce persists back to the disk after a map or reduce action.
- Easy to write queries:
  It has a lot of comfortable APIs and thus is more user friendly.
- Data Processing:
  It is highly suitable for graph processing (thanks to its high performance) and thus is for us.

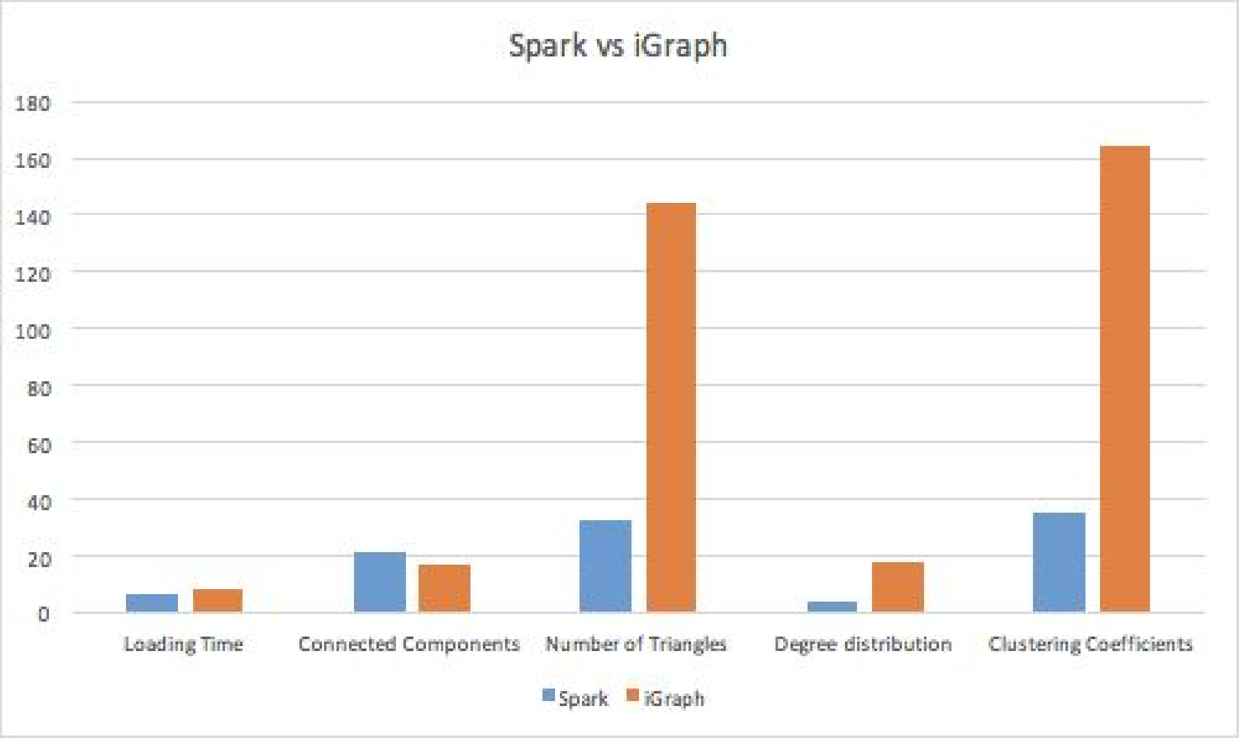
Procedure

- We tried to capture the effects of cores as well as increase in data-set size on the time taken for the queries.
- The queries were written in scala with the help of GraphX library.
- They were run on the DBMS server using the following configurations:-
  - Hyve.txt
    - Cores: 2, 4, 8, 16 (To study the effect of cores)
    - Queries: Degree Distribution, Pagerank, Global Clustering coefficient , Connected Components, Number of Triangles, Loading time
  - Flickr.txt
    - Cores: 16 (To study the effect of various databases)
    - Queries: Same as above

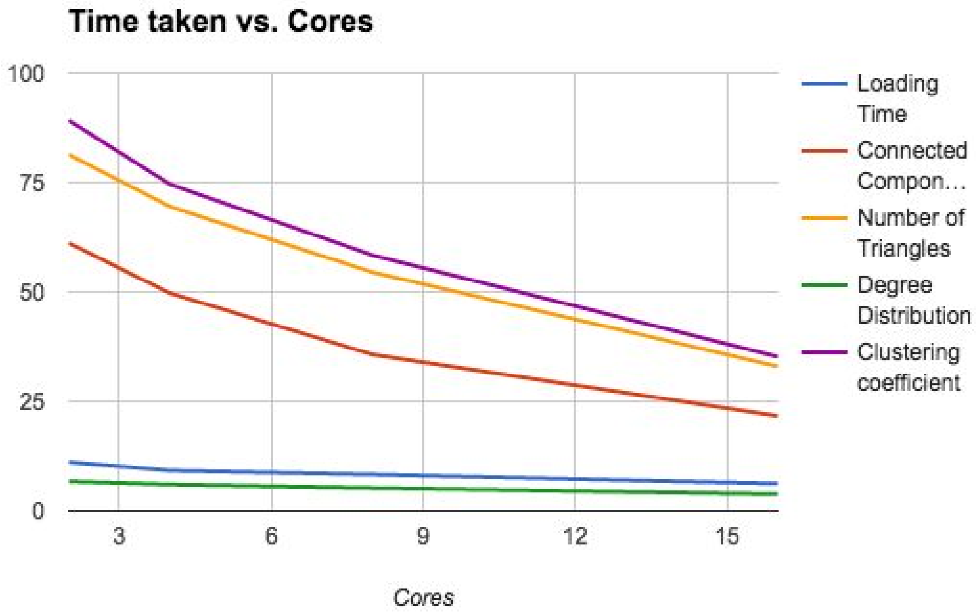
Results on Hyve dataset

Time (in s)	16 CORES	8 CORES	4 CORES	2 CORES
Loading Time	6.2	8.3	9.2	11.1
Pagerank	26.3	39.2	53.2	69.5
Connected Components	21.7	35.7	49.7	61.2
Number of Triangles	33	54.5	69.5	81.4
Degree distribution	3.8	5.2	6	6.7
Clustering Coefficients	35.2	58.4	74.6	89.2

Hyve Dataset



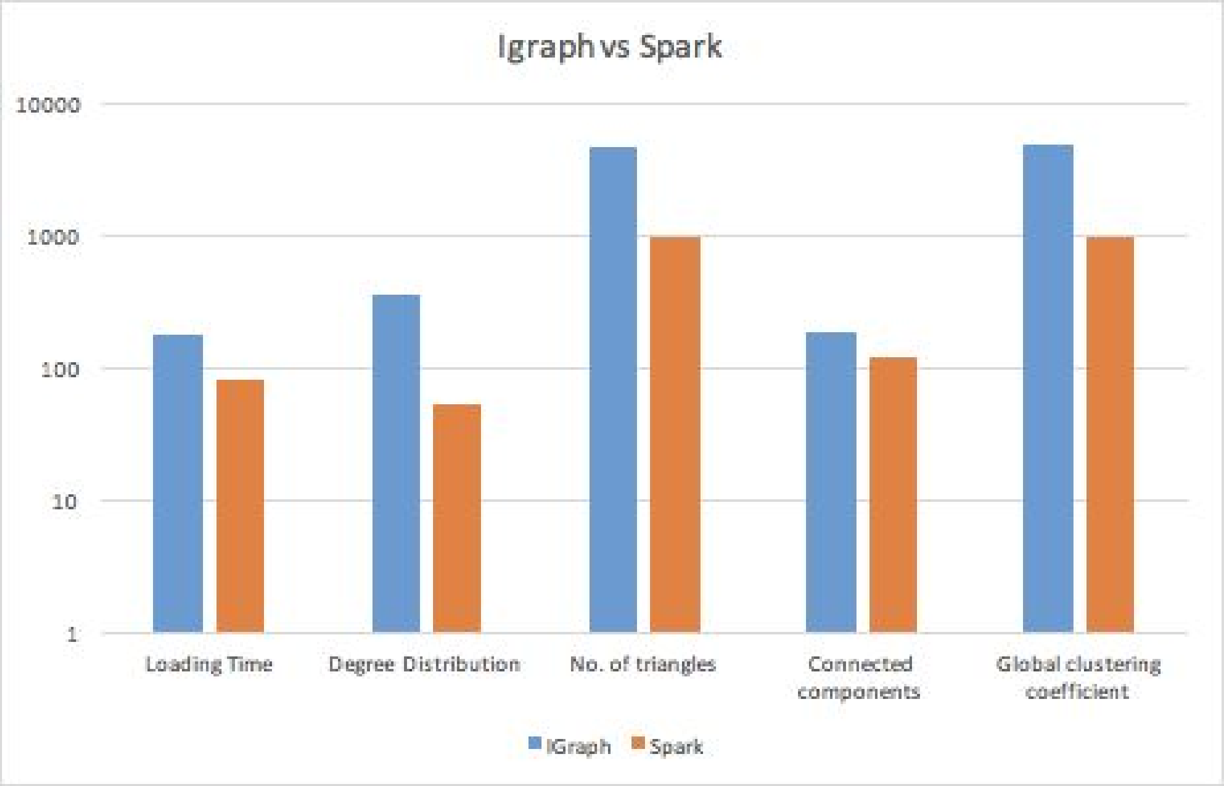
Hyve Dataset (Time taken vs. Cores)



Results on Flickr dataset

Time (in s)	16 Cores
Loading Time	82.5
Pagerank	1005.2
Connected Components	118.6
Number of Triangles	961.5
Degree distribution	53.8
Global Clustering Coefficients	984.2

FlickR Dataset



Conclusion

1. As it is evident from previous graph, Apache Spark performs far better than both Hive and Igraph. Thus, we can safely conclude that MapReduce distributed systems outshines single node systems.
2. Also increasing the number of cores, improves the performance.
3. Apache Spark is far more optimized than Hadoop MapReduce (that Hive uses).