Project:

Benchmarking tool for graph algorithms

Team id-22
Project id -19

By:
Abhinaba Sarkar 201405616
Malavika Reddy 201201193
Nikita Kad 201330030
Yash Khandelwal 201302164

HIGH-LEVEL DESIGN:

We will build various algorithms using Hadoop Mapreduce, GraphLab and Giraph then create a benchmarking to compare the performance of the algorithms to determine which graph algorithm works better for which kind of graphs and graph related queries.
We will implement a 3-tier architecture for managing data (in our case graphs), processing data (computation) and comparing and analyzing performance.

APPROACH:

We will be testing following algorithms:

- ➢ BFS - Breadth First Search: broken into two tasks

  - Map Task : In each Map task, we discover all the neighbors of the node currently in queue (we used color encoding GRAY for nodes in queue) and add them to our graph.
  - Reduce Task : In each Reduce task, we set the correct level of the nodes and update the graph.

- ➢ Page Rank: broken into two tasks

- Map task: Each page emits its neighbours and current pagerank.
- Reduce task: For each key (i.e page) new page rank is calculated using pagerank emitted in the map task.

➢ Dijkstra:

- Map task : In each of the map tasks, neighbors are discovered and put into the queue with color coding gray.
- Reduce task : In each of the reduce tasks, we select the nodes according to the shortest distances from the current node

PROJECT PLAN:

We will implement a basic 3 tier architecture for the project:

- Data management and representation (in Neo4j : DB for graphs)
- Distributed Computing unit (Hadoop cluster: test a graph for various algorithms and note the performance mainly time and memory used).
- Benchmarking unit (performance analysis of different types of graphs for various algorithms)

We need to analyse the runtime of graph algorithms on distributed systems. Performing computation on a graph requires processing at each node.

REQUIREMENTS:

Tools required for the implementation are:

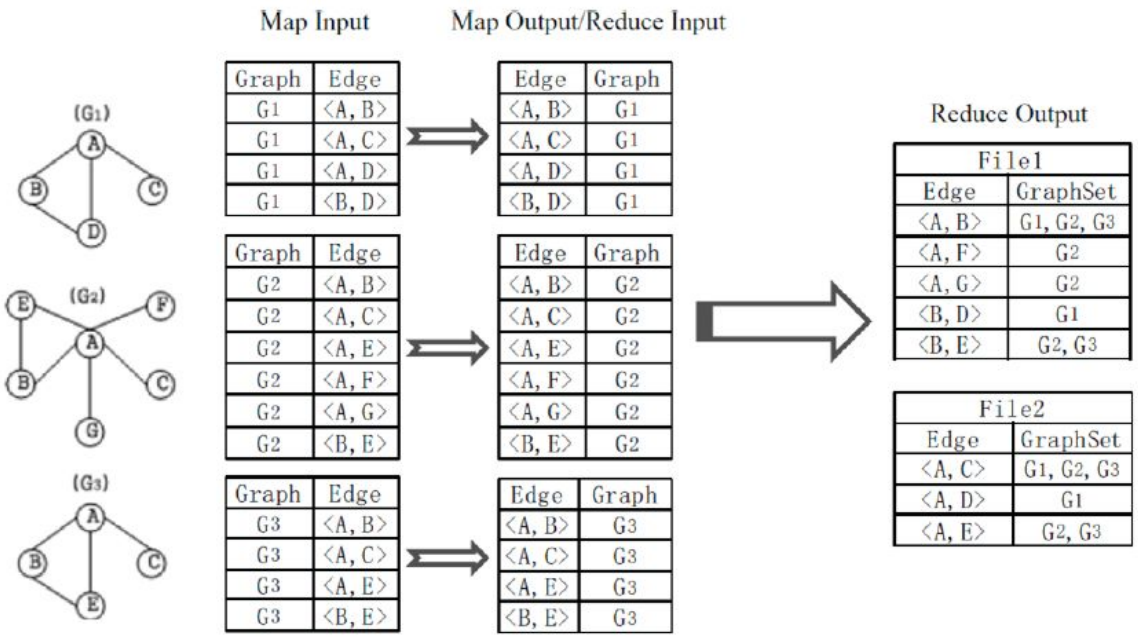
- Hadoop Mapreduce Engine

- GraphLab - GraphLab is a graph-based, high performance, distributed computation framework written in C++. It was originally developed for Machine Learning tasks. GraphLab is an asynchronous distributed shared-memory abstraction in which graph vertices share access to a distributed graph with data stored on every vertex and edge. In this

programming abstraction, each vertex can directly access information on the current vertex, adjacent edges, and adjacent vertices — irrespective of edge direction.

- Giraph - Apache Giraph is an iterative graph processing system built for high scalability. For example, it is currently used at Facebook to analyze the social graph formed by users and their connections. In Giraph, graph-processing programs are expressed as a sequence of iterations called *supersteps*. During a superstep, the framework starts a user-defined function for each vertex, conceptually in parallel. The user-defined function specifies the behaviour at a single vertex *V* and a single superstep *S*. The function can read messages that are sent to *V* in superstep *S-1*, send messages to other vertices that are received at superstep *S+1*, and modify the state of *V* and its outgoing edges. Messages are typically sent along outgoing edges, but you can send a message to any vertex with a known identifier. Each superstep represents atomic units of parallel computation.

- Neo4j - Neo4j is an open-source graph database, implemented in Java. Neo4j is an embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables.



EXPECTED OUTCOME:

By implementing this basic 3-tier architecture we will be able to analyze the performance (mainly time and memory usage) of various graph algorithms in Hadoop Mapreduce and will compare the performance of the same with GraphLab and Giraph and thus to determine which graph algorithm works better for which kind of graphs and graph related queries.