Graph Mining on Big Data System

Presented by Hefu Chai, Rui Zhang, Jian Fang

Outline

* Overview
* Approaches & Environment
* Results
* Observations
* Notes
* Conclusion

Overview

* What we have done?
  * Compared different platforms to process graph mining algorithm
  * Evaluated the usability of each platform
  * Analyzed the results of these experiments

Approaches & Environment

* Approaches
  * Algorithms: Degree Distribution, Weakly Connected Component, PageRank
  * Dataset: 2.7G Kronecker graph produced by method introduced in [1]
* Environment
  * AWS EC2 (us-east-1/m3.x2large)
  * 1, 3, 6 nodes
  * Ubuntu 12.04 LTS 64bit
  * Systems tested: HP Vertica, SciDB, Apache Hadoop, PostgreSQL

[1] "Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication." by Leskovec, Jurij, et al.

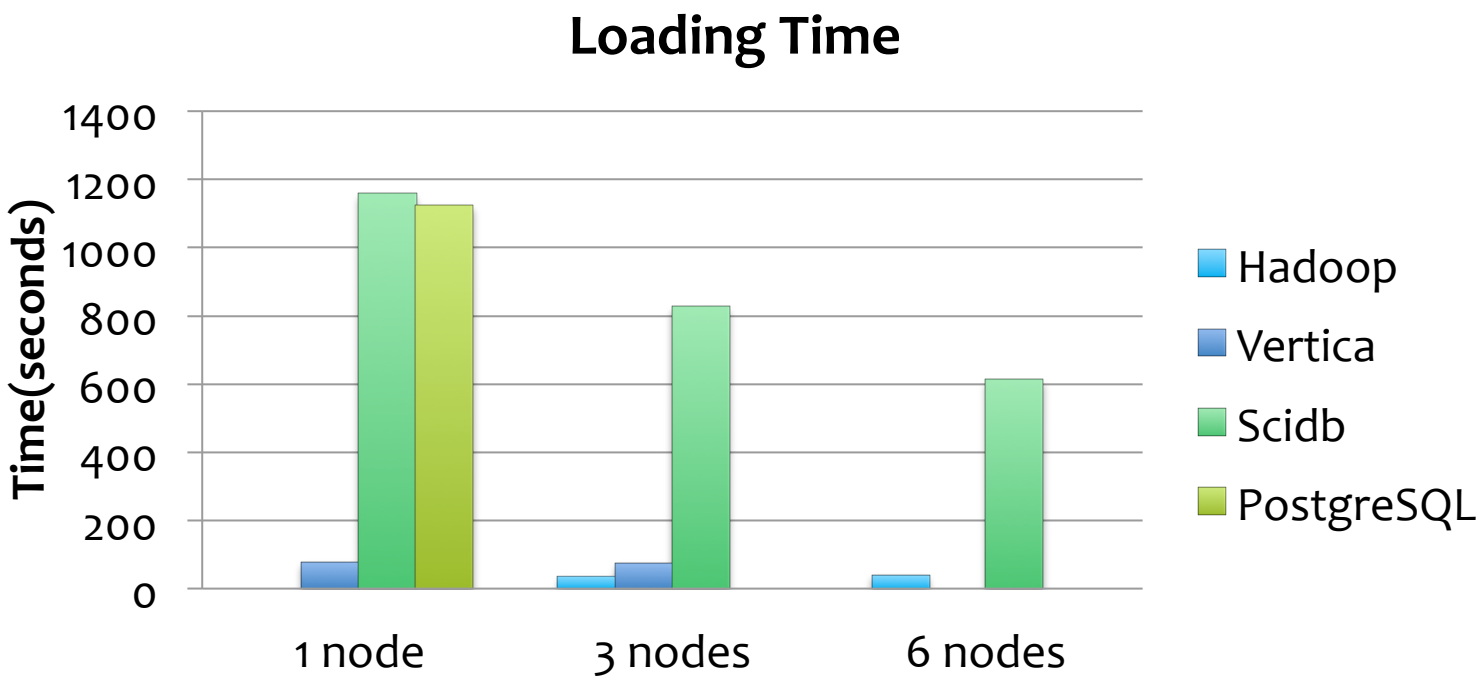
Approaches & Environment

* How we implemented these algorithms?
  * Vertica:
    * Use Java as host language(Vertica's UDF does not support loops)
  * PostgreSQL:
    * Use plpgsql language to define UDFs (Logics and main sql languages are the same as the vertica version)
  * Hadoop:
    * Degree Distribution: one MapReduce Job
    * Weakly Connected Component: three phases[initialization, computation(iterations), final]
    * PageRank: similar to weakly connected component
  * SciDB:
    * Array-based matrix multiplication

Approaches & Environment

* Special Notes
  * Why we only implement one node version on PostgreSQL?
  * Why we don't implement one node version on Hadoop?
  * Why we only implement one node and three node version on Vertica?
  * Why we choose Datasets with 2.7 GB?

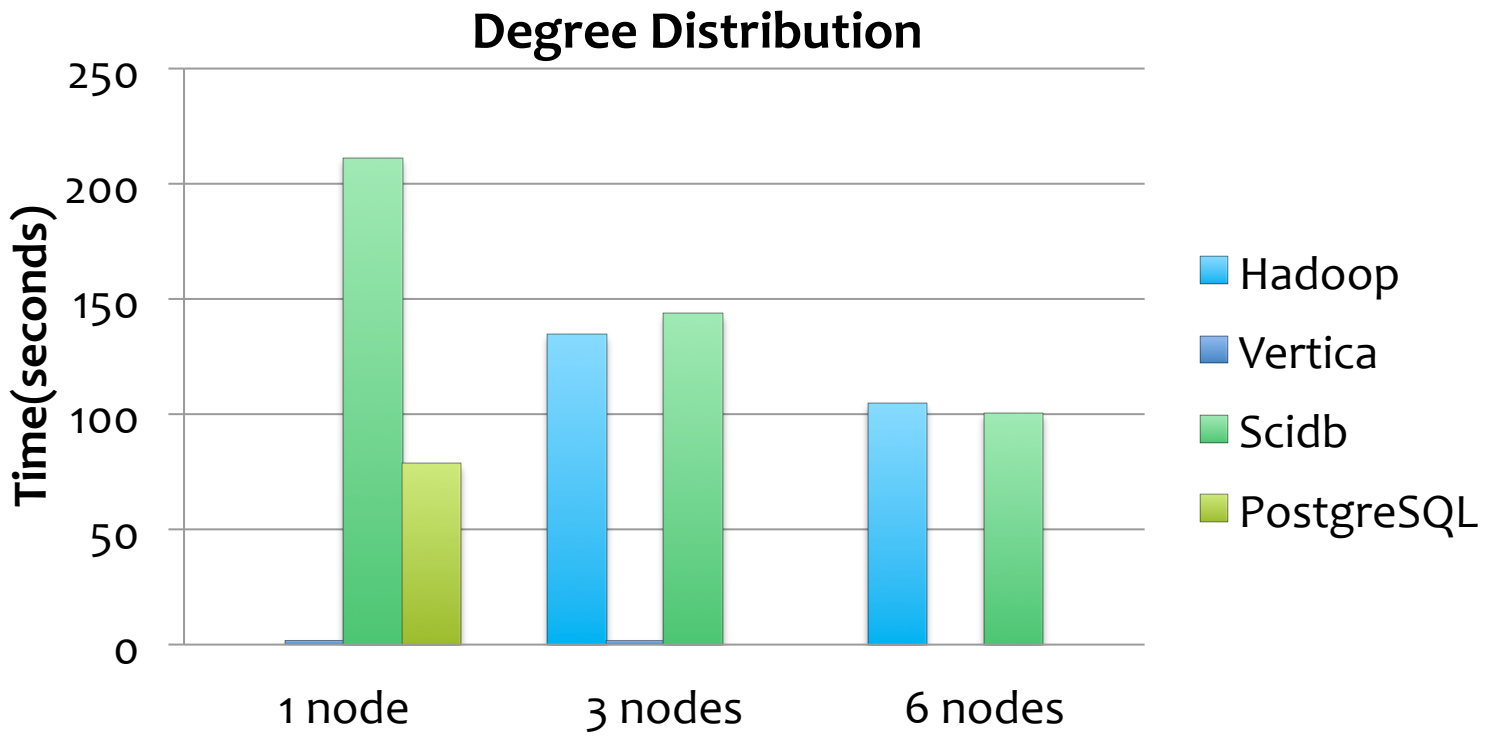
Results


Loading Time
Time(seconds)
1400
1200
1000
800
600
400
200
0
1 node
3 nodes
6 nodes
Hadoop
Vertica
Scidb
PostgreSQL

Observations

* Loading time
  * Hadoop has the most efficient loading efficiency.
  * Vertica also has a huge advantage over others
  * SciDB and PostgreSQL performs really bad
  * Analysis:
    * Hadoop can load data without extra operation; It simply divides the file into chunks and replicates them .
    * Vertica now has a Hybrid Storage Model (WOS, TM and ROS) in which WOS is optimized for data updating.
    * PostgreSQL manages a temporary buffer for data loading which is very small by default so it incorporates many disk writes.
    * SciDB supports parallel loading with multiple instances, but it is quite slow. The user guide does not provide in-situ data processing.

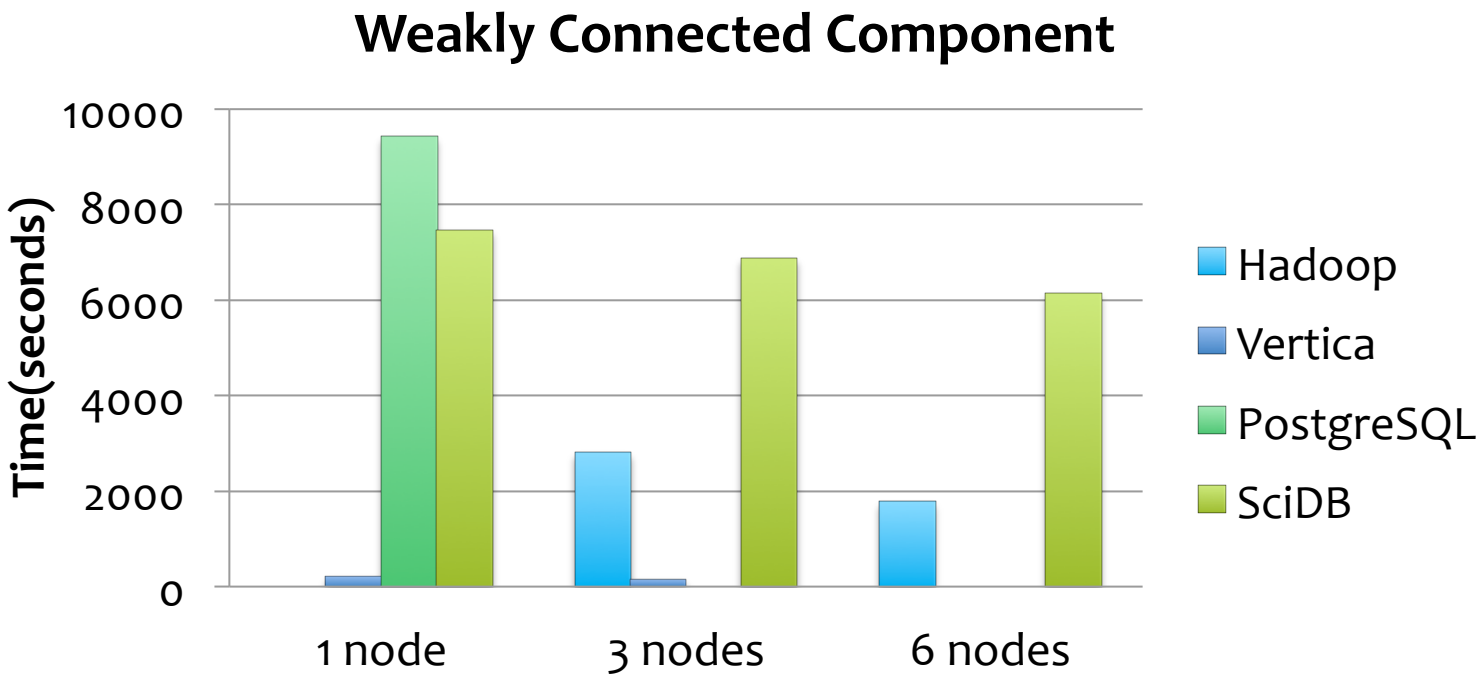
Results


Degree Distribution
Time(seconds)
250
200
150
100
50
0
1 node
3 nodes
6 nodes
Hadoop
Vertica
Scidb
PostgreSQL

Observations

* Degree Distribution
  * Hadoop performs badly and the performance does not increase linearly in this case
  * Vertica can finish this task within seconds
  * Although having the same logic, postgreSQL is less efficient than Vertica
  * SciDB takes minutes to finish the work due to its logic storage manner.
  * Analysis:
    * Hadoop has a huge launching overhead and it involves many disk writes and net work communications to do any simple job
    * Vertica is a read optimized column store. (compression strategy, only need to access the first column)
    * PostgresSQL needs to access the whole row
    * SciDB has to conduct redimension on data and store the temp data.

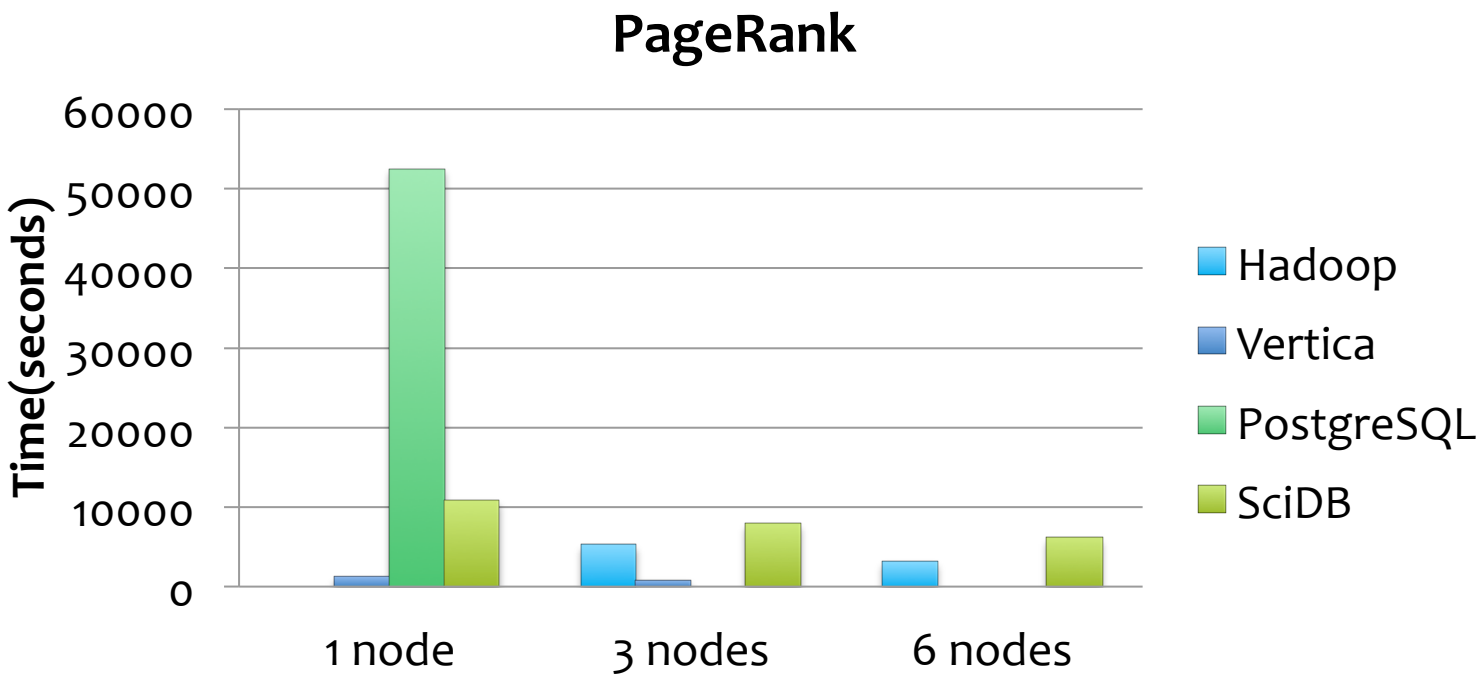
Results


Weakly Connected Component
Time(seconds)
10000
8000
6000
4000
2000
0
1 node
3 nodes
6 nodes
Hadoop
Vertica
PostgreSQL
SciDB

Observations

* Weakly Connected Component
  * Hadoop takes hours(3 nodes) or nearly an hour(6 nodes) to finish the job
  * Vertica finishes this task in 200 seconds(1 node) or 140 seconds(3 nodes) to finish the job
  * PostgreSQL takes nearly 3 hours to finish the job
  * SciDB also takes 2 hours to finish the job
  * Analysis:
    * Hadoop has many iterations and each iteration will start a new MapReduce job. The launching, disk I/O and network communication overhead is huge.
    * The SQLs for doing this job has many deletes, inserts and updates. Vertica performs well because of WOS which is a memory-resident data structure. WOS has no compression and indexing so it enables fast data update
    * The PostgreSQL performs bad because of the disk I/Os even if we increase the temp buffer size. It is still not sufficient to store the huge intermediate temporary table.
    * In SciDB, we implemented WCC based on adjacent matrix, and conduct join operations on dimensions. It performs better than PostgreSQL

Results


PageRank
Time(seconds)
60000
50000
40000
30000
20000
10000
0
1 node
3 nodes
6 nodes
Hadoop
Vertica
PostgreSQL
SciDB

Observations

* PageRank
  * Hadoop performs bad but scales well
  * Vertica finishes this task in 20 minutes(1 node) or 10 minutes(3 nodes) to finish the job
  * PostgreSQL takes more than half a day to finish
  * SciDB takes near 3 hours to finish the tasks
  * Analysis:
    * Hadoop scales well because of the independence among Mappers and Reducers
    * The SQLs for doing this job has mostly create and drop tables and join operations. Vertica does not arrange data as tables but projections so this task it perform less efficient than weakly connected component. But the pre-join projection provides improvements for join operation.
    * The PostgreSQL performs bad because of the same reason: insufficient temp buffer to contain the whole temporary table, many disk I/Os.
    * Similar to WCC, we implemented PageRank based on adjacent matrix, and conduct matrix and vector multiplication to update new pagerank value in each iteration.

Observations

* Usability
  * Hadoop
    * easy to setup and program
  * Vertica
    * has full documentation but also has some minor inaccuracies.
    * It use almost the same SQL semantics as other traditional relational DBMSs
    * It supports simple UDFs and UDXs which are also user-defined functions defined by other languages through Vertica SDK
  * PostgreSQL
    * It supports complex UDFs
  * SciDB
    * It gives user more freedom and more challenge on designing arrays
    * The system are decoupled so it is easy to add and remove nodes
    * It's tricky to implement an efficient graph mining algorithm with array based schema.

Notes

* We tried Giraph & Hama, but failed
  * Documents have bugs and are not complete
  * Community is small
  * Very hard to start a cluster
* Usability is important

Conclusion

* Hadoop performs bad when working with graph related problems (iterations)
* Vertica has strong power to process massive updates through its hybrid data model
* PostgreSQL, as a traditional relational database, is not suitable for large graph related problems (single node restriction)
* SciDB has specific design in array-based data, but not expressive enough in complex graph mining algorithm.