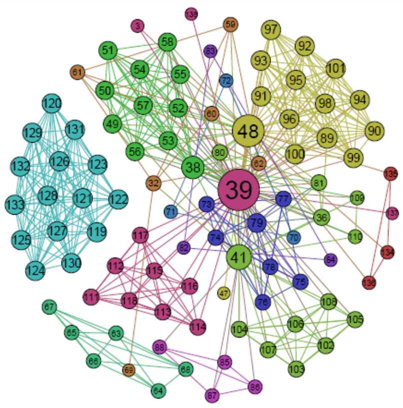


Large Scale Social Network Analysis

DATA ANALYTICS 2013

TUTORIAL

Rui Sarmento
email@ruisarmento.com

João Gama
jgama@fep.up.pt

IARIA

Outline

PART I

1. Introduction & Motivation
    - Overview & Contributions
2. Software Tools
    - Algorithms Provided
    - Advantages And Disadvantages
    - Metrics Calculations and Results
        - ➢ Case Studies
        - ➢ Practical Examples
        - ➢ Results - Processing Time
        - ➢ Example Results

Outline

PART II

1. Algorithm Developments
   - Green-Marl Language
   - Community Detection Algorithm
   - Similarity Ranking Algorithm
   - Metrics Calculations and Results
     - Case Studies
     - Practical Examples
     - Results – Modularity & Processing Time
2. Summary & Conclusions

Part Ⅰ

Outline

1. **Introduction & Motivation**
    - Overview & Contributions
2. Software Tools
    - Algorithms Provided
    - Advantages And Disadvantages
    - Metrics Calculations and Results
        - Case Studies
        - Practical Examples
        - Results - Processing Time
        - Example Results

Introduction & Motivation

Generic Problem:

Nowadays, the huge amounts of data available pose problems for analysis with regular hardware and/or software.

Solution:

Emerging technologies, like modern models for parallel computing, multicore computers or even clusters of computers, can be very useful for analyzing massive network data.

Tutorial Overview & Contributions

1. **Aggregation of information**:
   a. What tools to use for analyzing large social networks
   b. What algorithms are already implemented with these tools
   c. Several Tools - Advantages and Disadvantages

2. **Implementation Example** of algorithms for large scale Social Network analysis and some results:
   a. Community Detection algorithm implementation with Green-Marl language
   b. Similarity Ranking algorithm implementation also with Green-Marl language

Outline

1. Introduction & Motivation
   - Overview & Contributions
2. **Software Tools**
   - **Algorithms Provided**
   - **Advantages And Disadvantages**
   - **Metrics Calculations and Results**
     - ➢ **Case Studies**
     - ➢ **Practical Examples**
     - ➢ **Results - Processing Time**
     - ➢ **Example Results**

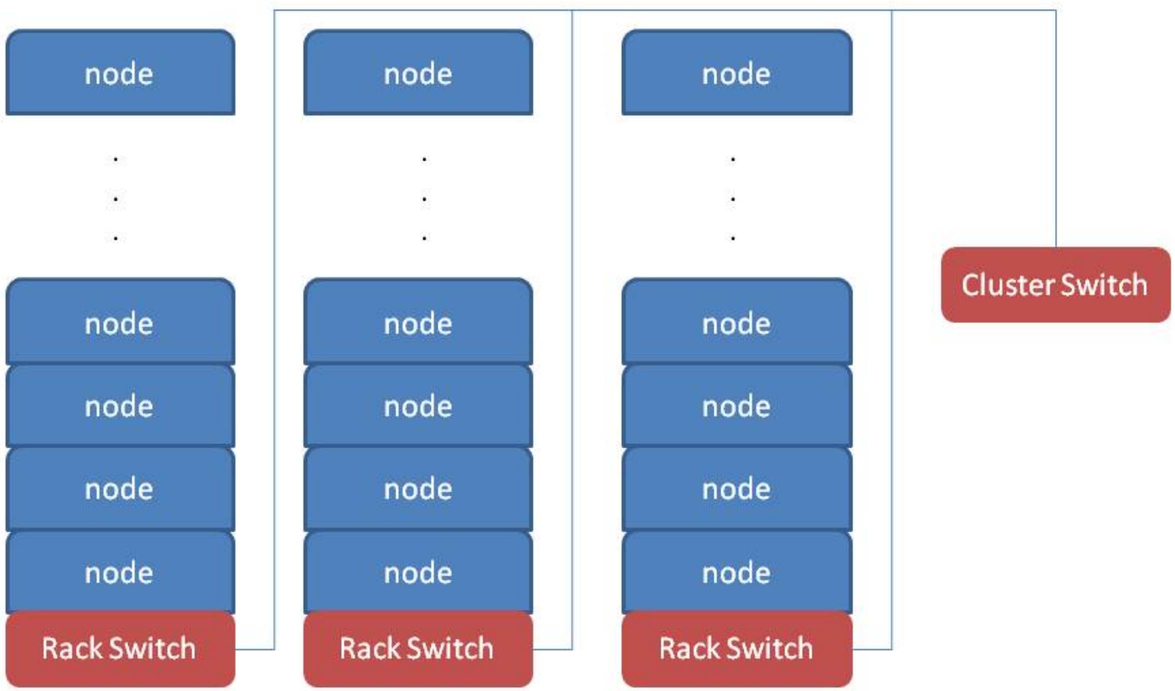
Software Tools

– **To list a few:**

1. **Hadoop Map/Reduce**
2. **Giraph**
3. **Graphlab**
4. **Pegasus**
5. **Green-Marl**

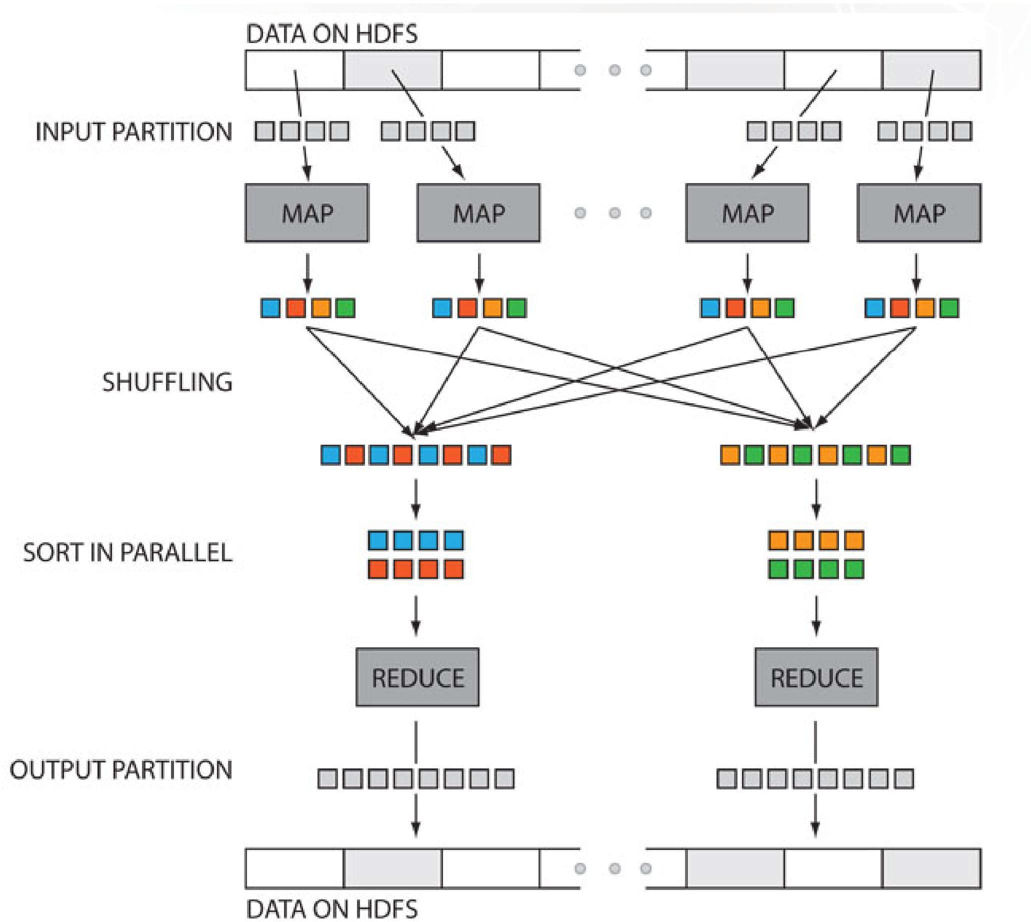
Software Tools

Hadoop HDFS – Architecture of Compute Nodes



Software Tools

Hadoop Map-Reduce



Software Tools

Hadoop MapReduce Example – Counting terms in documents

Let start with something really simple. The code snippet below shows Mapper that simply emit "1" for each term it processes and Reducer that goes through the lists of ones and sum them up:

```
class Mapper
   method Map(docid id, doc d)
      for all term t in doc d do
         Emit(term t, count 1)

class Reducer
   method Reduce(term t, counts [c1, c2,...])
      sum = 0
      for all count c in [c1, c2,...] do
          sum = sum + c
      Emit(term t, count sum)
```

Software Tools

Hadoop MapReduce Advantages & Disadvantages

Tool	Hadoop MR
Advantages	• Ability to write MapReduce programs in Java, a language which even many non computer scientists can learn with sufficient capability to meet powerful data-processing needs • Ability to rapidly process large amounts of data in parallel • Can be deployed on large clusters of cheap commodity hardware as opposed to expensive, specialized parallel-processing hardware • Can be offered as an on-demand service, for example as part of Amazon's EC2 cluster computing service Washington (2011)
Disadvantages	• One-input two-phase data flow rigid, hard to adapt - Does not allow for stateful multiple-step processing of records • Procedural programming model requires (often repetitive) code for even the simplest operations (e.g., projection, filtering) • Map Reduce nature is not specially directed to implement code that presents iterations or iterative behavior • Opaque nature of the map and reduce functions impedes optimization from Zinn (2010)

Software Tools

Hadoop Map-Reduce Algorithms (Online Resources):

Highly Scalable Blog

- **Log Analysis, Data Querying**
- **Graph Analysis, Web Indexing**
- **Text Analysis, Market Analysis**

atbrox.com website

- **Ads Analysis**
- **Bioinformatics/Medical Informatics**
- **Information Extraction and Text Processing**
- **Artificial Intelligence/Machine Learning/Data Mining**
- **Statistics**
- **Numerical Mathematics**
- **Graphs**

Software Tools

Algorithms Provided – Other tools

Software	Pegasus	Graphlab	Giraph	Snap
Algorithms available from software install	• Degree • PageRank • Random Walk with Restart (RWR) • Radius • Connected Components	• approximate diameter • kcore • pagerank • connected component • simple coloring • directed triangle count • simple undirected triangle count • format convert • sssp • undirected triangle count	• Simple Shortest Path (available from) • Simple In Degree Count • Simple Out Degree Count • Simple Page Rank • Connected Components	• cascades • centrality • cliques • community • concomp • forestfire • graphgen • graphhash • kcores • kronem • krongen • kronfit • maggen • magfit • motifs • ncpplot • netevol • netinf • netstat • mkdatasets • infopath
Parallel computing	YES	YES	YES	NO
Can user configure number of cores or machines?	YES	YES	YES	NO

Software Tools

Advantages & Disadvantages

Tool	Pegasus	Graphlab	Giraph	Snap
Advantages	• Similar positive points to Hadoop MR	• Algorithms can be described in a node-centric way; same computation is repeatedly performed on every node. • Significant amounts of computations are performed on each node. • Can be used for any Graph as long as their sparse.	• Several advantages over Map Reduce: - it's a stateful computation - Disk is hit if/only for checkpoints - No sorting is necessary - Only messages hit the network as mentioned from Martella (2012)	• Optimized for Graph processing. • Written with C++ which is intrinsically considered a fast language
Disadvantages	• Similar negative points to Hadoop MR	• Programmability: user must restructure his algorithm in a node centric way. • There is an overhead of runtime system when the amount of computation performed at each node is small. • Small world graphs: Graphlab lock scheme may suffer from frequent conflicts for such graphs.	• Still in a very immature phase of development • Lack of a complete offered algorithm library	• Not developed to take advantage of parallel or distributed processing of tasks • Some algorithms can be time consuming even for relatively small graphs due to the number of graph characteristics covered (eg. "centrality" algorithm)

Software Tools

Metrics Calculations and Results – Use Case Studies

- Network A – Relationships Between Tech. Companies and Financial Institutions.
  - 16.339 vertexes and 30.313 edges.
  - Retrieved from Crunchbase API

- Network B – Relationships Between Personalities and Companies.
  - 107.033 vertexes and 128.746 edges.
  - Retrieved from Crunchbase API

- Network C – Amazon co-purchased products.
  - 334.863 vertexes and 925.872 edges.
  - Retrieved from Stanford Large Network Dataset Collection

- Network D – Youtube online social network.
  - 1.134.890 vertexes and 2.987.624 edges.
  - Retrieved from Stanford Large Network Dataset Collection

- Network E – Live Journal online social network.
  - 3.997.962 vertexes and 34.681.189 edges.
  - Retrieved from Stanford Large Network Dataset Collection

Software Tools

Practical Example with Graphlab – Triangle Counting

110414015@hpcgrid-centos6:~/graphlabapi/release/toolkits/graph_analytics

```
[110414015@hpcgrid-centos6 graph_analytics]$ ./undirected_triangle_count --graph
=/home/110414015/Dados/com-lj.ungraph.tsv --format=tsv
This program counts the exact number of triangles in the provided graph.

INFO:     mpi_tools.hpp(init:63): MPI Support was not compiled.
TCP Communication layer constructed.
INFO:     metrics_server.cpp(launch_metric_server:219): Metrics server now liste
ning on http://hpcgrid-centos6:8090
INFO:     distributed_graph.hpp(load_from_posixfs:1823): Loading graph from file
: /home/110414015/Dados/com-lj.ungraph.tsv
INFO:     distributed_ingress_base.hpp(finalize:166): Finalizing Graph...
INFO:     distributed_ingress_base.hpp(exchange_global_info:493): Graph info:
         nverts: 3997962
         nedges: 34681189
         nreplicas: 3997962
         replication factor: 1
Number of vertices: 3997962
Number of edges:    34681189
Counting Triangles...
INFO:     synchronous_engine.hpp(start:1248): Iteration counter will only output
 every 5 seconds.
INFO:     synchronous_engine.hpp(start:1263): 0: Starting iteration: 0
INFO:     synchronous_engine.hpp(start:1312):   Active vertices: 3997962
INFO:     synchronous_engine.hpp(start:1361):    Running Aggregators
INFO:     synchronous_engine.hpp(start:1263): 0: Starting iteration: 1
INFO:     synchronous_engine.hpp(start:1312):   Active vertices: 0
INFO:     synchronous_engine.hpp(start:1373): 1 iterations completed.
Updates: 3997962
Counted in 14.4743 seconds
177820130 Triangles
Metrics server stopping.
[110414015@hpcgrid-centos6 graph_analytics]$
```

Software Tools

Case Studies - Metrics and their practical use

- Triangles – involved in the computation of one of the main statistical property used to describe large graphs met in practice and that is the clustering coefficient of the node.

- K-Core – The concept of a k-core was introduced to study the clustering structure of social networks from and to describe the evolution of random graphs. It has also been applied in bioinformatics and network visualization.

- Friends of Friends – this algorithm is of good application in the commercial data networks where the results could serve as basis for a recommender system.

- Centrality Measures – The centrality measures algorithms have large application in several areas including Psychology, Anthropology, Business and communications, Ecology among many others.

Software Tools

Processing Time

Processing Time	**Hadoop MR** “Friends of Friends”	**Pegasus** Degree Measures	**Graphlab** Triangles Counting	**Snap** Centrality Measures
Network **A**	16,040s	5,380s	0,048s	374s (06m14s)
Network **B**	23,880s	7,070s	0,103s	17400s(4h50m)
Network **C**	138,980s	11,050s	0,305s	-[1]
Network **D**	430,420s	23,330s	1,211s	-[1]
Network **E**	1516,257s	35,680s	16,211s	-[1]

[1] Value too high

Software Tools

Example Results

1. **Pegasus Degree**

2	**30**
4	224
6	59
8	13
10	48
12	113
14	12

2. **Friends of Friends**

10077	8507:2,17745:1,11077:1,24814:1,85008:1,24937:1,2569:1,2599:1,15721:1,26176:1
1008	73285:1,1469:1,35600:1,247:1,213:1,58475:1,51474:1,7522:1,1991:1,1010:1
1009	14833:1,35600:1,2050:1,11160:1,184:1,2474:1,7313:1,142:1,247:1,73285:1
10099	7613:1,7466:1,109:1,2474:1,12:1,357:1,27658:1,15:1,1135:1,26915:1
101	36:8,15:3,7293:3,26:2,7434:2,513:2,53:2,87:2,6:1,6319:1
1010	7490:4,1875:2,607:2,247:1,35509:1,100:1,1:1,57:1,1008:1,1009:1
1011	939:3,15:3,54:2,7279:2,7377:2,51820:1,5136:1,507:1,5:1,483:1
10116	55775:2,2870:2,39005:2,18924:2,72017:2,26185:1,25966:1,25866:1,25794:1,24768:1
1012	10996:1,1523:1
10120	35585:1,3192:1,31255:1,30752:1,30748:1,30663:1,27754:1,26857:1,26789:1,2665:1
10121	13289:1,11617:1,671:1,18956:1
10127	81082:1,9417:1,813:1,7542:1,7541:1,7227:1,27141:1,24898:1,15759:1,12134:1
10128	59502:1,5822:1,5739:1,56896:1,5344:1,4746:1,4410:1,43497:1,43350:1,4314:1

Software Tools

Example Results

3. Centrality Measures with Snap

#NodeId	Degree	Closeness	Betweennes	EigenVector	Network Constraint	Clustering Coefficient	PageRank	HubScore	Authority Score
3	80.00	0.233747	1139257.192383	0.000461	0.016776	0.000633	0.001181	0.000094	0.029831
843	14.00	0.193071	164648.965528	0.000028	0.083915	0.000000	0.000798	0.000000	0.000021
844	16.00	0.207691	287289.050309	0.000061	0.071393	0.000000	0.000907	0.000000	0.001772
9	33.00	0.213657	310964.724490	0.000223	0.039056	0.000000	0.000361	0.000008	0.015517
1352	9.00	0.181062	96242.573356	0.000015	0.118590	0.000000	0.000539	0.000000	0.000147

Part II

Outline

1. **Algorithm Developments**
   - Green-Marl Language
   - Community Detection Algorithm
   - Similarity Ranking Algorithm
   - Metrics Calculations and Results
     - Case Studies
     - Practical Examples
     - Results – Modularity & Processing Time
2. Summary & Conclusions

Algorithm Developments

Green-Marl Language

•Green-Marl, a DSL in which a user can describe a graph analysis algorithm in a intuitive way. This DSL captures the high-level semantics of the algorithm as well as its inherent parallelism.

•The Green-Marl compiler which applies a set of optimizations and parallelization enabled by the high-level semantic information of the DSL and produces an optimized parallel implementation targeted at commodity SMP machines.

•An interdisciplinary DSL approach to solving computational problems that combines graph theory, compilers, parallel programming and computer architecture.

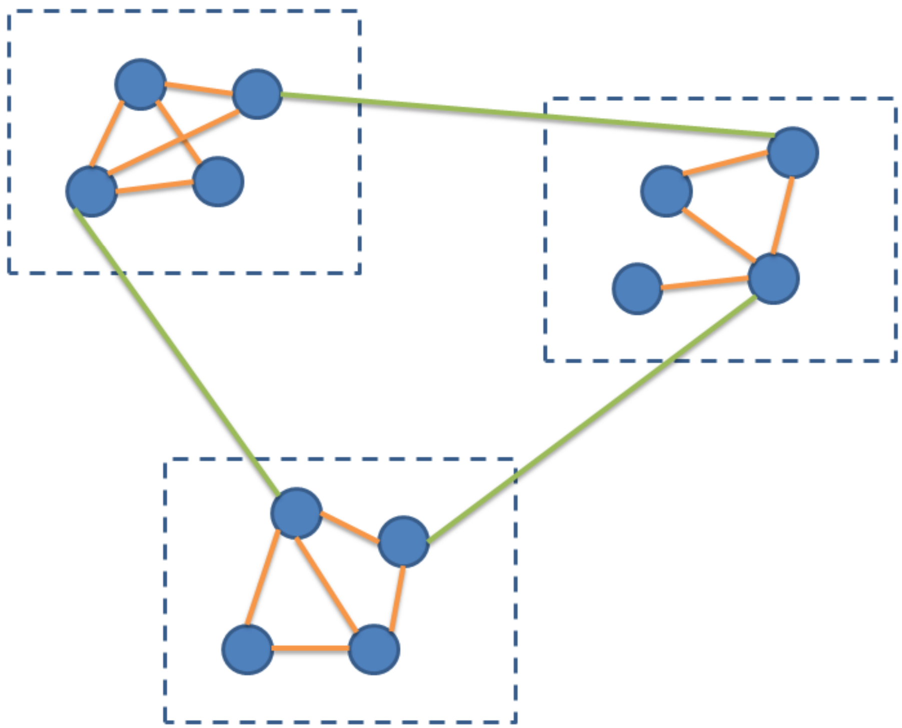
Algorithm Developments

Green-Marl Language - Available Algorithms

Green-Marl Software Algorithms	Brief Description	OpenMP C++ compatible	Giraph/GPS compatible
avg_teen_count	Computes the average teen count of a node	YES	YES
bc	Computes the betweenness centrality value for the graph	YES	NO
bc_random	Computes an estimation for the betweenness centrality value for the graph	YES	YES
communities	Computes the different communities in a graph	YES	NO
kosaraju	Finds strongly connected components using Kosaraju's Algorithm	YES	NO
pagerank	Computes the pagerank value for every node in the graph	YES	YES
potential-friends	Computes a set of potential friends for every node using triangle closing	YES	NO
sssp	Computes the distance of every node from one destination node according to the shortest path	YES	YES
sssp_path	Computes the shortest paths from one destination node to every other node in the graph and returns the shortest path to a specific node.	YES	NO
triangle_counting	Computes the number of closed triangles in the graph	YES	NO

Algorithm Developments

Community Detection



Simple Graph with 3 communities surrounded with dashed squares.

Algorithm Developments

Community Detection

•Community detection is known to be a NP-complete problem.

•Community detection can be related to graph partitioning and there are good parallel algorithms for graph partitioning but for community detection it is a usual problem that relies on parallelism achievable from sequential algorithms.

•The top-down approach (divisive approach) or bottom-up approach (agglomerative approach) have inherent sequential flow with possibility of being parallelized on a higher amount on the first stages than the later stages.

•Because of the high computational overhead of community detection algorithms one cannot usually apply such algorithms to networks of hundreds of millions of nodes or edges. Thus, an efficient and high quality algorithm (modularity) for community detection is hard to achieve and a challenging problem as mentioned by Soman and Narang (2011).

Algorithm Developments

Similarity Ranking Algorithm

•SimRank proposed by Jeh and Widom (2002) has become a measure to compare the similarity between two nodes using network structure.

•Although SimRank is applicable to a wide range of areas such as social networks, citation networks, link prediction and others, it suffers from heavy computational complexity and space requirements.

•The basic recursive intuition behind SimRank approach is "two objects are similar if they are referenced by similar objects."

•Being an algorithm with $O(n^2)$ time complexity where n is the number of nodes in the graph, it is a good choice to develop it in distributed computing environments.

Algorithm Developments

Results – Case Studies

1. **Community Detection Algorithm**

Networks for Algorithms Modularity Comparison

- Zachary's Karate Club with 34 vertexes and 78 edges.
- Dolphin Social Network with 62 vertexes and 159 edges.
- American Colleague Football with 115 vertexes and 615 edges.

Networks for Algorithms Processing Time Comparison

- Network A with 16.339 vertexes and 30.313 edges.
- Network B with 107.033 vertexes and 128.746 edges.
- Network C with 334.863 vertexes and 925.872 edges.

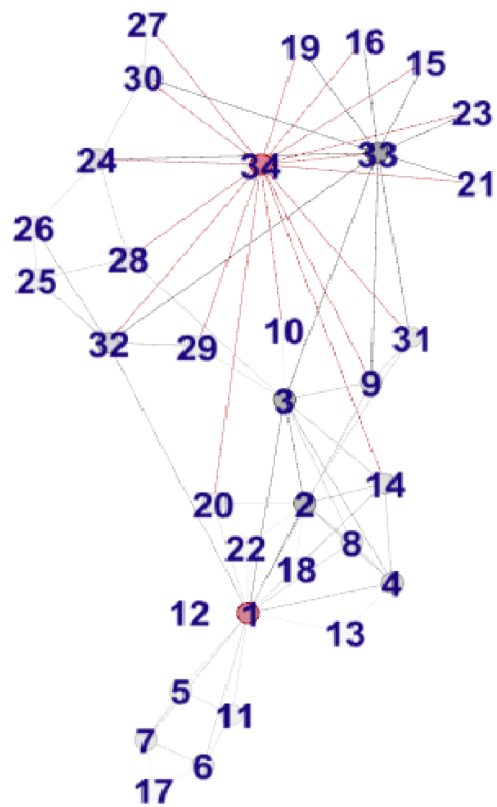
2. **Similarity Ranking Algorithm**

Networks for Sequential vs Parallel Comparison

- Network F with 471 vertexes and 250 edges.
- Network G with 892 vertexes and 500 edges.
- Network H with 1.659 vertexes and 999 edges.

Algorithm Developments

Practical Example - Community Detection Algorithm



Zachary's Karate Club with 34 vertexes and 78 edges.

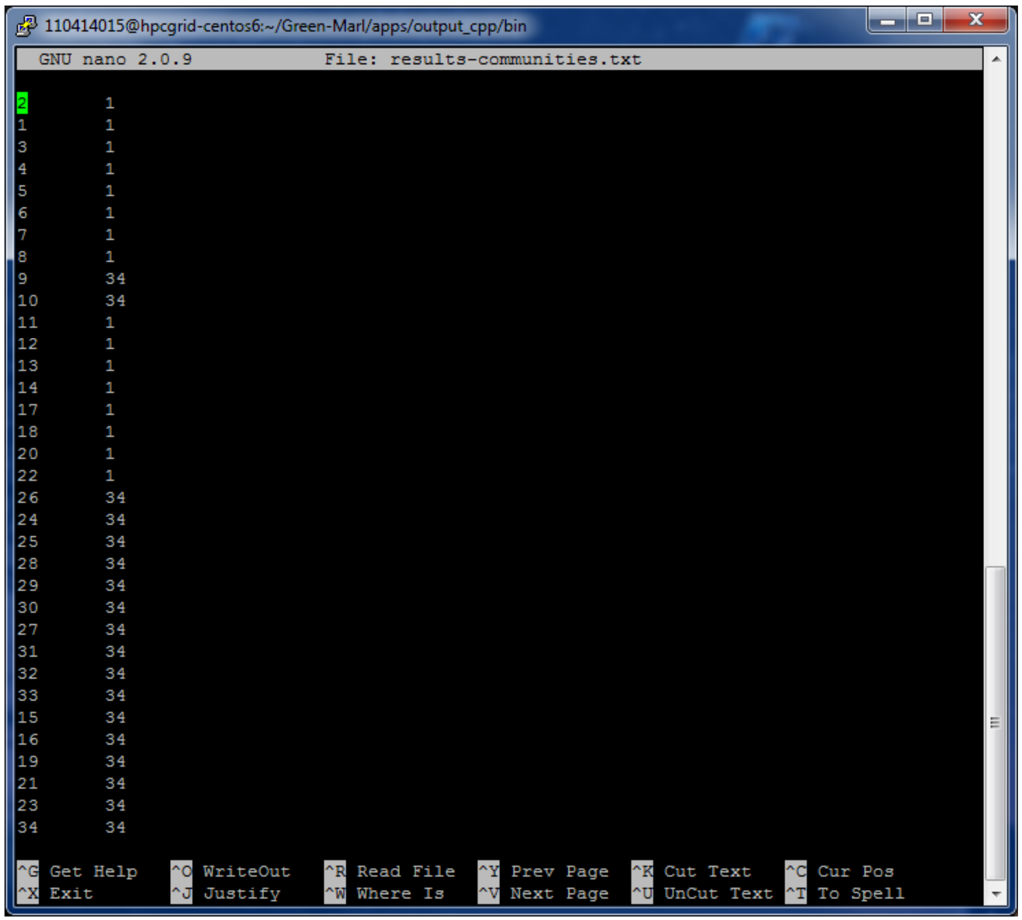
Algorithm Developments

Practical Example - Community Detection Algorithm

```
110414015@hpcgrid-centos6:~/Green-Marl/apps/output_cpp/bin

[110414015@hpcgrid-centos6 bin]$ ./communities-algo

############################################
#####  Community Detection Algorithm   #####
############################################

1st Cleaning Task Successfully Done
2nd Cleaning Task Successfully Done
Is the graph directed? Answer y (yes) or n (no): n
Input graph file name (only unweighted edge list is accepted!!): karate-edge.txt
Do you want to calculate Modularity? It can make the algorithm slow! Answer y (yes)
or n (no): y
Started Computation of Communities Algo at: 15:26
Loading Edge List...
Opened File karate-edge.txt
Initializing Variables...
Graph is undirected!
Begining While Loop to read the edge list file...
Graph has 34 Nodes!
Graph has 78 Edges!
Closing Edge List file!
Calculating Communities Labels for every node...
Processing...Computing Graph Edges Weight!
Processing...Setting Initial Community Labels for every node!
Processing...Computing Graph Maximum Neighbor's Edge Weight and for every node!
Processing...Computing/Creating Auxiliary Graph Edges!
Processing...Computing Auxiliary Graph Kosaraju Strong Connected Components!
Processing...Labeling each node in component with its lower label!
Algorithm Iteration 1
Algorithm Iteration 2
Algorithm Iteration 3
Algorithm Iteration 4
Calculating Modularity. Please Wait...
Modularity: 0.435897
Ended Computation of Communities Algorithm at: 15:26
Processing Time - 0.000833333 hours, 0.05 minutes OR 3 seconds
Compiling Results...
Opening Raw Results File
Begining While Loop to read the Raw file...
File results-communities.txt has the algorithm results! Enjoy!

############################################
```

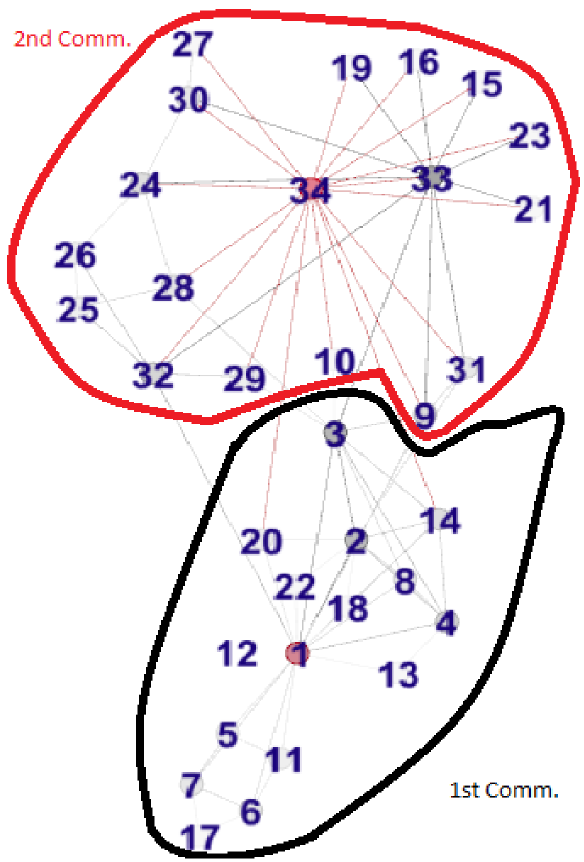
Algorithm Developments

Practical Example - Community Detection Algorithm



Algorithm Developments

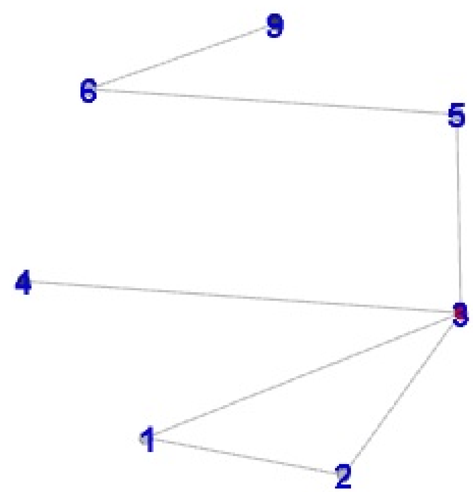
Practical Example - Community Detection Algorithm



Zachary's Karate Club with 34 vertexes and 78 edges, divided in 2 Communities by the developed algorithm.

Algorithm Developments

Practical Example - Similarity Ranking Algorithm



Test Network used in the development of the similarity algorithm.

Algorithm Developments

Practical Example - Similarity Ranking Algorithm

110414015@hpcgrid-centos6:~/Green-Marl/apps/output_cpp/bin

```
[110414015@hpcgrid-centos6 bin]$ ./simrank

################################
#####  SimRank Algorithm   #####
################################

1st Cleaning Task Successfully Done
2nd Cleaning Task Successfully Done
Is the graph directed? Answer y (yes) or n (no): n
Input graph file name (only unweighted edge list is accepted!!): Tests.txt
Started Computation of Similarity Ranking (Simrank): 15:29
Loading Edge List...
Opened File Tests.txt
Initializing Variables...
Graph is undirected!
Begining While Loop to read the edge list file...
Graph has 7 Nodes!
Graph has 7 Edges!
IMPORTANT NOTE: With this Graph, memory use will be approximately around 0MB MAX
PLEASE MAKE SURE YOUR MACHINE'S MEMORY IS ENOUGH TO RUN THE ALGORITHM!!
Closing Edge List file...
Closed Edge List file!
Calculating Simrank for every node...
Ended Computation of Simrank at: 15:29
Processing Time - 0.000833333 hours, 0.05 minutes OR 3 seconds
Compiling Results...
Opening Raw Results File
Begining While Loop to read the Raw file...
File results-simrank.txt has the algorithm results! Enjoy!

################################
#####  SimRank Algorithm   #####
################################

[110414015@hpcgrid-centos6 bin]$
```

Algorithm Developments

Practical Example - Similarity Ranking Algorithm

	1	2	3	4	6	5	9
1	1.000000	0.235798	0.168164	0.350434	0.051199	0.209529	0.068624
2	0.235798	1.000000	0.168164	0.350434	0.051199	0.209529	0.068624
3	0.168164	0.168164	1.000000	0.066980	0.177689	0.043468	0.019956
4	0.350434	0.350434	0.066980	1.000000	0.018981	0.353290	0.106580
6	0.051199	0.051199	0.177689	0.018981	1.000000	0.012027	0.005073
5	0.209529	0.209529	0.043468	0.353290	0.012027	1.000000	0.353290
9	0.068624	0.068624	0.019956	0.106580	0.005073	0.353290	1.000000

Algorithm Developments

Community Detection Algorithm – Sequential vs Parallel

Modularity	**Girvan – Newman Algorithm with Snap**	**Clauset-Newman-Moore Algorithm with Snap**	**Developed Algorithm with GM**
Zachary's Karate Club	0.401	0.381	**0.436**
Dolphin Social Network	**0.519**	0.515	0.333
American College Football	**0.599**	0.549	0.339

Processing Time	**Girvan – Newman Algorithm with Snap**	**Clauset-Newman-Moore Algorithm with Snap**	**Developed Algorithm with GM**
Network **A**	288 (hours)	6s	**4s**
Network **B**	300+ (hours)	**53s**	133s
Network **C**	400+ (hours)	*	**45659s**

[1] Failed with *segmentation fault (core dumped)* error

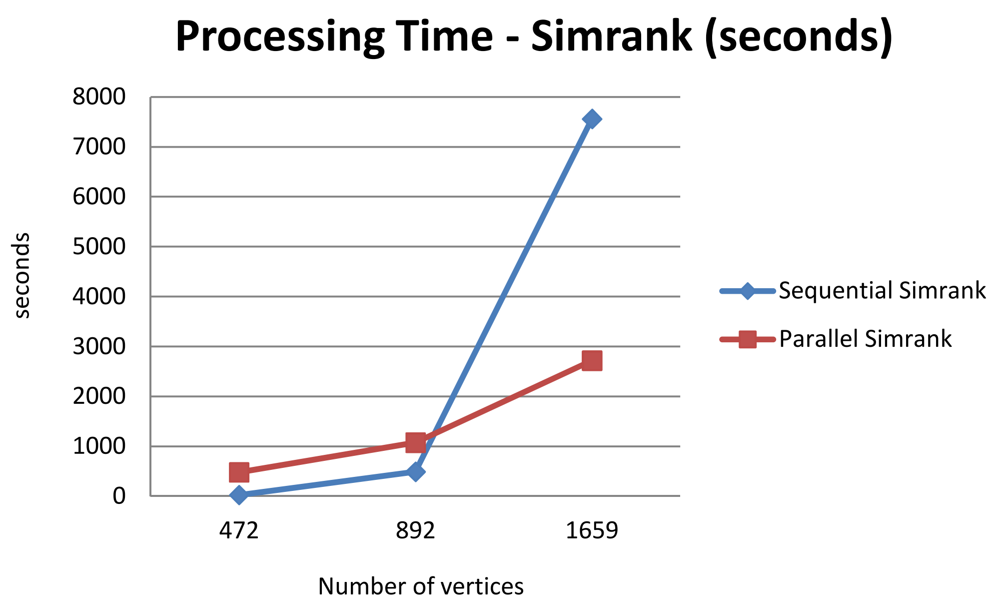
Algorithm Developments

Similarity Ranking Algorithm – Sequential vs Parallel

Processing Time	**Parallel Simrank with Green-Marl**	**Sequential Simrank with R**
Network **F**	480s	**25s**
Network **G**	1073s	**491s**
Network **H**	**2716s**	7560s
Network **A**	**26851s**	1022000+ s

Algorithm Developments

Similarity Ranking Algorithm – Sequential vs Parallel



Outline

1. Algorithm Developments
   - Green-Marl Language
   - Community Detection Algorithm
   - Similarity Ranking Algorithm
   - Metrics Calculations and Results
     - Case Studies
     - Practical Examples
     - Results – Modularity & Processing Time
2. **Summary & Conclusions**

Summary & Conclusions

❖One of this part of the tutorial goals was to expose which tools to look for when dealing with big graphs studies.

❖We made the introduction to the tools used nowadays for distributed graph analysis

❖We wrote some practical examples of computing algorithms that leverage the tools potential for big scale graphs studies

❖Other tutorial goal was to prove the utility and diversity of the tools and algorithms available for graph studies.

❖We learned also that the increasing number of SDLs for big graph analysis make the choice of languages for programming tasks between two generic languages, C++ and Java.

❖The Green-Marl language was also a great tool in the set of tools available and some implementation results are given in this tutorial.

Summary & Conclusions

Support Documents

- “Large Scale Social Networks Analysis” – Thesis
  - Document available for download on:
    - http://www.ruisarmento.com/uploads/Large_Scale_Social_Networks_Analysis_-_2013_-_Aftermath.pdf
  - Code available for download:
    - http://www.ruisarmento.com/uploads/Code.zip

Summary & Conclusions

Some References

- Alvarez-Hamelin, J. I., L. Dall'Asta, A. Barrat and A. Vespignani (2005). "k-core decomposition: a tool for the visualization of large scale networks". CoRR.
- Apache. (2012). "Apache Giraph." from http://incubator.apache.org/giraph/.
- Backstrom, L., D. Huttenlocher, J. M. Kleinberg and X. Lan (2006). "Group Formation in Large Social Networks: Membership, Growth, and Evolution". KDD, page 44-54. ACM.
- Bader, G. D. and C. W. Hogue (2003). "An automated method for finding molecular complexes in large protein interaction networks". BMC Bioinformatics.
- Clauset, A., M. E. J. Newman and C. Moore (2004). "Finding community structure in very large networks". Physical review E 70(6):066111.
- Fortunato, S. (2010). "Community detection in graphs". Physics Reports 486(3–5):75 - 174, Physics Reports.

Summary & Conclusions

Some References

- Girvan, M. and M. E. J. Newman (2002). "Community structure in social and biological networks". Proceedings of the National Academy of Sciences 99(12):7821-7826.
- Graphlab. (2012). "Graph Analytics Toolkit." 2012, from http://graphlab.org/toolkits/graph-analytics/.
- Graphlab. (2012). "Graphlab The Abstraction." 2012, from http://graphlab.org/home/abstraction/.
- Holmes, A. (2012). Hadoop In Practice, Manning.
- Hong, S., H. Chafi, E. Sedlar and K. Olukotun (2012). "Green-Marl: A DSL for Easy and Efficient Graph Analysis". ASPLOS, page 349-362. ACM.
- Jeh, G. and J. Widom (2002). "SimRank: A Measure of Structural-Context Similarity". Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, page 538--543. New York, NY, USA, ACM.

Summary & Conclusions

Some References

- Kang, U. (2012). "PEGASUS: Peta-Scale Graph Mining System." Retrieved 11-2012, from http://www.cs.cmu.edu/~pegasus/.
- Kang, U., D. H. Chau and C. Faloutsos (2010). "PEGASUS User's Guide", Carnegie Mellon University.
- Kang, U. and C. E. Tsourakakis (2009). "PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations". Proceeding ICDM '09 Proceedings of the 2009 Ninth IEEE International Conference on Data Mining.
- Latapy, M. (2008). "Main-memory Triangle Computations for Very Large (Sparse (Power-Law)) Graphs". Theor. Comput. Sci. 407(1-3):458-473.
- Leo, S. (2012, 2012-12-20 16:00:03). "Hadoop Wiki." Retrieved 16-01-2013, 2013, from http://wiki.apache.org/hadoop/PoweredBy.

Summary & Conclusions

Some References

- Leskovec, J. (2009). "Stanford Large Network Dataset Collection." Retrieved 25-02-2013, 2013, from http://snap.stanford.edu/data/index.html.
- Leskovec, J. (2012). "Stanford Network Analysis Platform." Retrieved 12-2012, 2012, from http://snap.stanford.edu/snap/.
- Lizorkin, D., P. Velikhov, M. Grinev and D. Turdakov (2008). "Accuracy Estimate and Optimization Techniques for SimRank Computation". VLDB J. 19(1):45-66.
- Luczak, T. (1991). "On the size and connectivity of the k-core of the random graph".
- Malewicz, G., M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser and G. Czajkowski (2010). "Pregel: A System for Large-Scale Graph Processing". Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, page 135--146. New York, NY, USA, ACM.

Summary & Conclusions

Some References

- Martella, C. (2012). "Apache Giraph: Distributed Graph Processing in the Cloud". FOSDEM 2012, Graph Processing Room.
- Mazza, G. (2012, 2012-11-30 19:22:49). "FrontPage - Hadoop Wiki." Retrieved 11-2012, from http://wiki.apache.org/lucene-hadoop/.
- Newman, M. (2006). "Modularity and community structure in networks". Proceedings of the National Academy of Sciences of the United States of America 103(23):8577--82.
- Newman, M. (2013). "Network Data." Retrieved 04-2013, from http://www-personal.umich.edu/~mejn/netdata/.
- Noll, M. G. (August 5, 2007, June 29, 2012). "Running Hadoop On Ubuntu Linux (Single-Node Cluster)." Retrieved 06-11-2012, from http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/.
- Owens, J. R. (2013). "Hadoop Real-World Solutions Cookbook", PACKT Publishing.

Summary & Conclusions

Some References

- Rajaraman, A., J. Leskovec and J. D. Ullman (2012). "Mining of Massive Datasets". Cambridge University Press, Cambridge.
- Science, C. M. U.-S. o. C. (2012). "Getting Started with PEGASUS." Retrieved 11-2012, from http://www.cs.cmu.edu/~pegasus/getting%20started.htm.
- Seidman, S. B. (1983). "Network structure and minimum degree". Social Networks 5(3):269 - 287.
- Sharir, M. (1981). "A strong-connectivity algorithm and its applications in data flow analysis", NEW YORK UNIVERSITY.
- Society, I. C. (1990). "System Application Program Interface (API) [C Language]. Information technology—Portable Operating System Interface (POSIX)", IEEE Press, Piscataway,NJ.
- Soman, J. and A. Narang (2011). "Fast Community Detection AlgorithmWith GPUs and Multicore Architectures". 2011 IEEE International Parallel & Distributed Processing Symposium.

Summary & Conclusions

Some References

- Thanedar, V. (2012). "API Documentation."   Retrieved 04-2012, 2012, from http://developer.crunchbase.com/docs.
- Washington, U. o. (2011). "What is Hadoop?"   Retrieved 05-03-2013, 2013, from http://escience.washington.edu/get-help-now/what-hadoop.
- Zinn, D. (2010). "MapReduce". Amazon Cloud Computing Workshop in conjunction to the Bioinformatics Next Generation Sequencing Data Analysis Workshop.