

Families of distributed graph algorithms

Divide and conquer

Márton Balassi¹

`mbalassi@ilab.sztaki.hu`

¹Hungarian Academy of Sciences – Institute for Computer Science and Control
Data Mining & Search Group

June 24, 2014

Table of contents

Distributing data-intensive algorithms

- Motivation

- MapReduce & Pregel

- Counting the number of triangles in a graph

Families of distributed graph algorithms

- Local algorithms

- Graph traversal based algorithms

- Matrix multiplication based algorithms

Experiments

- Representative algorithms

- Results

Table of contents

Distributing data-intensive algorithms

Motivation

MapReduce & Pregel

Counting the number of triangles in a graph

Families of distributed graph algorithms

Local algorithms

Graph traversal based algorithms

Matrix multiplication based algorithms

Experiments

Representative algorithms

Results

A bit about myself

My background

- ▶ BSc, MSc in Computer Science, Eötvös University Budapest
- ▶ BA in Economics, TU Budapest
- ▶ Distributed algorithms
- ▶ Big data architecture

A bit about myself

My background

- ▶ BSc, MSc in Computer Science, Eötvös University Budapest
- ▶ BA in Economics, TU Budapest
- ▶ Distributed algorithms
- ▶ Big data architecture

A bit about myself

My background

- ▶ BSc, MSc in Computer Science, Eötvös University Budapest
- ▶ BA in Economics, TU Budapest
- ▶ Distributed algorithms
- ▶ Big data architecture

A bit about myself


My background

- ▶ BSc, MSc in Computer Science, Eötvös University Budapest
- ▶ BA in Economics, TU Budapest
- ▶ Distributed algorithms
- ▶ Big data architecture

Motivation

Let's do a PageRank on this graph...


- ▶ A large Portuguese webcrawl¹
- ▶ $3.1 \cdot 10^9$ nodes
- ▶ $1.1 \cdot 10^{11}$ edges
- ▶ 80 GB of compressed data
- ▶ Divide and conquer is almost mandatory

¹a large Portuguese crawl of the Portuguese Web Archive obtained from  Daniel Gomes

Motivation

Let's do a PageRank on this graph...


- ▶ A large Portuguese webcrawl¹
- ▶ $3.1 \cdot 10^9$ nodes
- ▶ $1.1 \cdot 10^{11}$ edges
- ▶ 80 GB of compressed data
- ▶ Divide and conquer is almost mandatory

¹a large Portuguese crawl of the Portuguese Web Archive obtained from  Daniel Gomes

Motivation

Let's do a PageRank on this graph...


- ▶ A large Portuguese webcrawl¹
- ▶ $3.1 \cdot 10^9$ nodes
- ▶ $1.1 \cdot 10^{11}$ edges
- ▶ 80 GB of compressed data
- ▶ Divide and conquer is almost mandatory

¹a large Portuguese crawl of the Portuguese Web Archive obtained from  Daniel Gomes

Motivation

Let's do a PageRank on this graph...


- ▶ A large Portuguese webcrawl¹
- ▶ $3.1 \cdot 10^9$ nodes
- ▶ $1.1 \cdot 10^{11}$ edges
- ▶ 80 GB of compressed data
- ▶ Divide and conquer is almost mandatory

¹a large Portuguese crawl of the Portuguese Web Archive obtained from  Daniel Gomes

Motivation

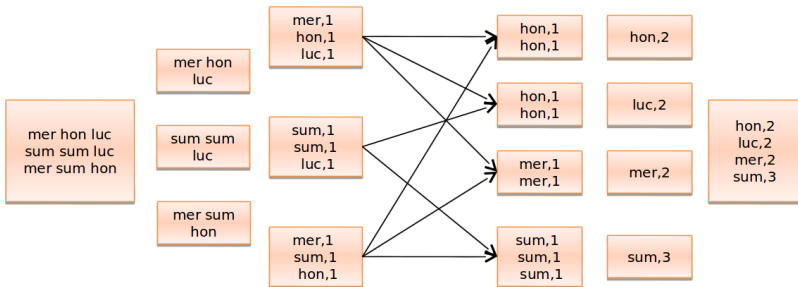
Let's do a PageRank on this graph...

- ▶ A large Portuguese webcrawl¹
- ▶ $3.1 \cdot 10^9$ nodes
- ▶ $1.1 \cdot 10^{11}$ edges
- ▶ 80 GB of compressed data
- ▶ Divide and conquer is almost mandatory

¹a large Portuguese crawl of the Portuguese Web Archive obtained from  Daniel Gomes

MapReduce [DG04]

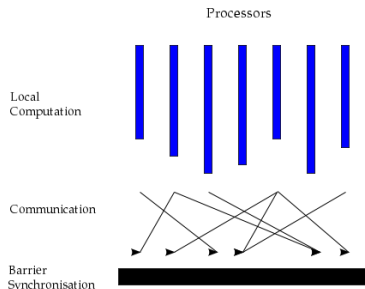
Input Splitting Mapping Shuffling Reducing Output



Pregel [MAB⁺10]

Traits

- ▶ Bulk Synchronous Parallel [Val90]
- ▶ „Think like a vertex”
- ▶ Graph kept in memory

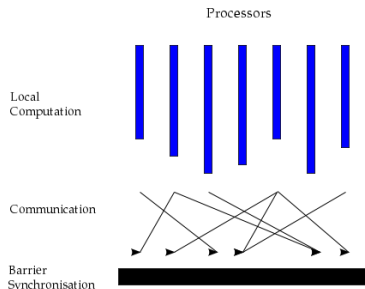


Scheme of the BSP system
Wikipedia, public domain

Pregel [MAB⁺10]

Traits

- ▶ Bulk Synchronous Parallel [Val90]
- ▶ „Think like a vertex”
- ▶ Graph kept in memory

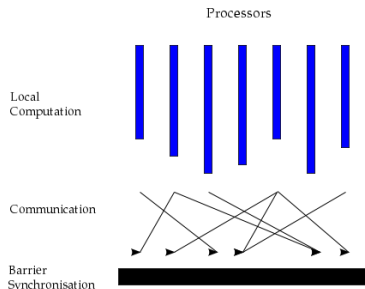


Scheme of the BSP system
Wikipedia, public domain

Pregel [MAB⁺10]

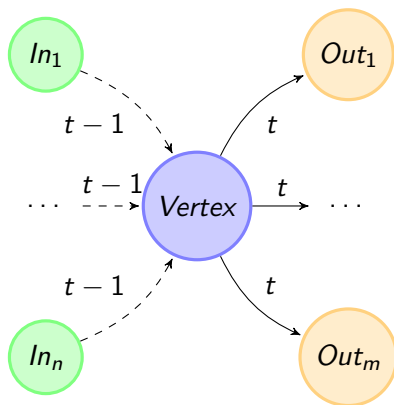
Traits

- ▶ Bulk Synchronous Parallel [Val90]
- ▶ „Think like a vertex”
- ▶ Graph kept in memory



Scheme of the BSP system
Wikipedia, public domain

Pregel [MAB⁺10]



Pregel schema as perceived from a vertex

Triangle Counter – Sequential algorithm

Sequential algorithm

Every vertex executes a search of itself bounded in depth of three. Thus every triangle is counted three times.

Triangle Counter – Sequential algorithm

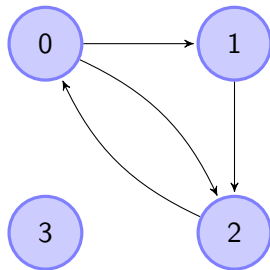
Sequential algorithm

Every vertex executes a search of itself bounded in depth of three.
Thus every triangle is counted three times.
You can do better by making use of the ordering on the vertices.

Triangle Counter – distributed algorithm

Representation

```
0 1 2  
1 2  
2 0  
3
```



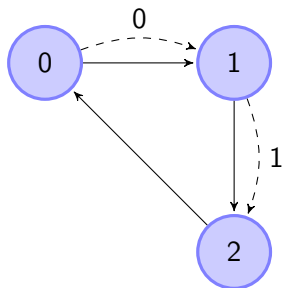
Triangle Counter – distributed algorithm

First Map

Let's send our ID to all of our neighbours possessing a higher ID than ours. Let's send our neighbours to ourselves.

First Reduce

Let's write out the information received.



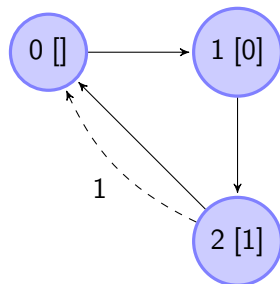
Triangle Counter – distributed algorithm

Second Map

If the ID received is smaller than ours let's pass it on to our neighbours.
Let's send our neighbours to ourselves.

Second Reduce

If the ID received is our neighbour then let's increment a global counter.



Triangle Counter – distributed algorithm

Second Map

If the ID received is smaller than ours, let's pass it on to our neighbours.

Let's send our neighbours to ourselves.

Second Reduce

If the ID received is our neighbour, then let's increment a global counter.

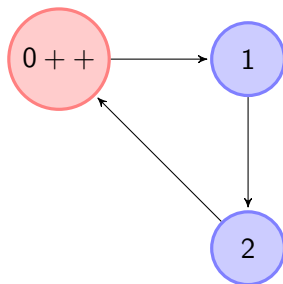


Table of contents

Distributing data-intensive algorithms

Motivation

MapReduce & Pregel

Counting the number of triangles in a graph

Families of distributed graph algorithms

Local algorithms

Graph traversal based algorithms

Matrix multiplication based algorithms

Experiments

Representative algorithms

Results

Local algorithms

Traits

- ▶ Dependant on a small environment of the given vertex or edge.
- ▶ „Trivial” candidates for parallel computing.
- ▶ Examples are fingerprint computation, local clustering coefficient and the number of triangles.

Local algorithms

Traits

- ▶ Dependant on a small environment of the given vertex or edge.
- ▶ „Trivial” candidates for parallel computing.
- ▶ Examples are fingerprint computation, local clustering coefficient and the number of triangles.

Local algorithms

Traits

- ▶ Dependant on a small environment of the given vertex or edge.
- ▶ „Trivial” candidates for parallel computing.
- ▶ Examples are fingerprint computation, local clustering coefficient and the number of triangles.

Graph traversal based algorithms

Traits

- ▶ Dependant on taking long routes in the graph.
- ▶ Difficult to implement in a distributed environment.
- ▶ The distributed algorithm can be less effective than the sequential as the representation is less powerful.
- ▶ Examples could be accessibility, betweenness centrality and strongly connected components.

Graph traversal based algorithms

Traits

- ▶ Dependant on taking long routes in the graph.
- ▶ Difficult to implement in a distributed environment.
- ▶ The distributed algorithm can be less effective than the sequential as the representation is less powerful.
- ▶ Examples could be accessibility, betweenness centrality and strongly connected components.

Graph traversal based algorithms

Traits

- ▶ Dependant on taking long routes in the graph.
- ▶ Difficult to implement in a distributed environment.
- ▶ The distributed algorithm can be less effective than the sequential as the representation is less powerful.
- ▶ Examples could be accessibility, betweenness centrality and strongly connected components.

Graph traversal based algorithms

Traits

- ▶ Dependant on taking long routes in the graph.
- ▶ Difficult to implement in a distributed environment.
- ▶ The distributed algorithm can be less effective than the sequential as the representation is less powerful.
- ▶ Examples could be accessibility, betweenness centrality and strongly connected components.

Matrix multiplication based algorithms

Traits

- ▶ Are basically power iterations of matrix-vector multiplications with typically fast convergence traits.
- ▶ Suitable to implement in Pregel-based systems and not that straight-forward in plain MapReduce. [KBEK14]
- ▶ Representatives could be eigenvalue centrality, PageRank or LineRank.

Matrix multiplication based algorithms

Traits

- ▶ Are basically power iterations of matrix-vector multiplications with typically fast convergence traits.
- ▶ Suitable to implement in Pregel-based systems and not that straight-forward in plain MapReduce. [KBK14]
- ▶ Representatives could be eigenvalue centrality, PageRank or LineRank.

Matrix multiplication based algorithms

Traits

- ▶ Are basically power iterations of matrix-vector multiplications with typically fast convergence traits.
- ▶ Suitable to implement in Pregel-based systems and not that straight-forward in plain MapReduce. [KBK14]
- ▶ Representatives could be eigenvalue centrality, PageRank or LineRank.

Table of contents

Distributing data-intensive algorithms

Motivation

MapReduce & Pregel

Counting the number of triangles in a graph

Families of distributed graph algorithms

Local algorithms

Graph traversal based algorithms

Matrix multiplication based algorithms

Experiments

Representative algorithms

Results

Representative algorithms

Local

Graph traversal based

Matrix multiplication based

Representative algorithms

Local

TriangleCounter already presented ...

Graph traversal based

Matrix multiplication based

Representative algorithms

Local

Graph traversal based

LineRank Construct the *linegraph*, where vertices represent the edges of the original graph and a directed edge points from e_1 to e_2 if the target of e_1 is the source of e_2 .

Matrix multiplication based

Representative algorithms

Local

Graph traversal based

LineRank Construct the *linegraph*, where vertices represent the edges of the original graph and a directed edge points from e_1 to e_2 if the target of e_1 is the source of e_2 . Run a PageRank on this. [KPST11, PBMW98]

Matrix multiplication based

Representative algorithms

Local

Graph traversal based

Matrix multiplication based

SCC Run a label propagation to detect connected components. Remove edges with different labels, then do a reverse label propagation. [[MIHPR05](#)]

Representative algorithms

Local

Graph traversal based

Matrix multiplication based

SCC Run a label propagation to detect connected components. Remove edges with different labels, then do a reverse label propagation. [[MIHPR05](#)] Remove the sccs found and iterate.

Representative algorithms

Local

Graph traversal based

Matrix multiplication based

SCC Run a label propagation to detect connected components. Remove edges with different labels, then do a reverse label propagation. [MIHPR05] Remove the sccs found and iterate. The sequential algorithm is more efficient. [BCVDP11]

Results [EBKK14]

Name	Vertices	Edges	Source
web-Google	$8,7 \cdot 10^5$	$5,0 \cdot 10^7$	[LLDM08]
wiki-Talk	$2,4 \cdot 10^6$	$5,1 \cdot 10^7$	[LHK10]
soc-LiveJournal	$4,8 \cdot 10^6$	$6,9 \cdot 10^8$	[BHKL06]
forest10M	10^7	$2,4 \cdot 10^8$	generated ²
forest20M	$2 \cdot 10^7$	$4,8 \cdot 10^8$	generated ²

Summary of the graphs used

²A slightly modified version of the ForestFire model. [LF06, EBKK14]

Results [EBKK14]

Implementations

sequential Plain Java.

MapReduce Apache Hadoop on 20 cores, 40 reducers.

Pregel Apache Giraph on 17 cores, 3 occupied by Zookeeper.

Results [EBKK14]

Implementations

sequential Plain Java.

MapReduce Apache Hadoop on 20 cores, 40 reducers.

Pregel Apache Giraph on 17 cores, 3 occupied by Zookeeper.

Results [EBKK14]

Implementations

sequential Plain Java.

MapReduce Apache Hadoop on 20 cores, 40 reducers.

Pregel Apache Giraph on 17 cores, 3 occupied by Zookeeper.

Results [EBKK14]

Implementations

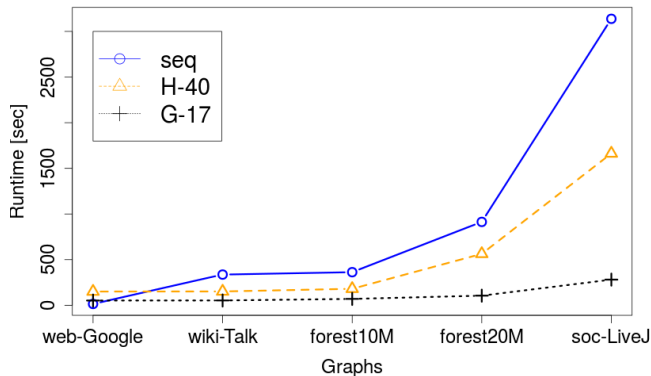
sequential Plain Java.

MapReduce Apache Hadoop on 20 cores, 40 reducers.

Pregel Apache Giraph on 17 cores, 3 occupied by Zookeeper.

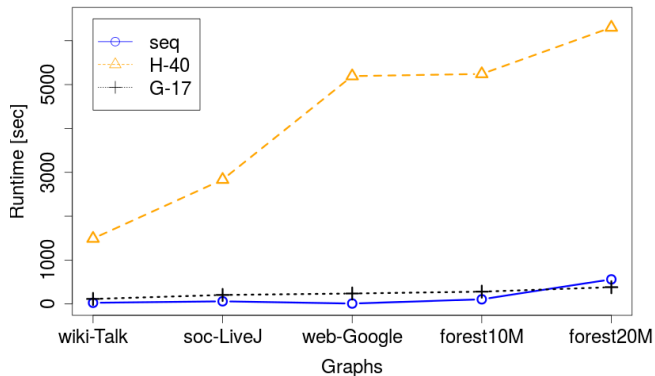
Results [EBKK14]

TriangleCounter performance

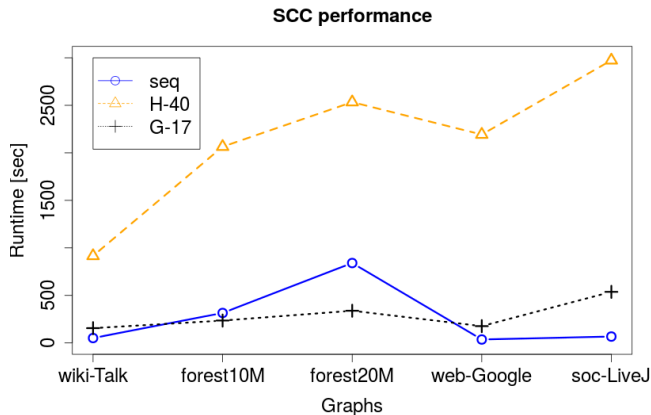


Results [EBKK14]

Linerank performance



Results [EBKK14]



Summary

Take home messages

- ▶ If your data is „big” you have to go multi-core/machine
- ▶ If multi-machine, then try MapReduce
- ▶ For distributed graph algorithms it is useful to distinguish three families
- ▶ The families behave differently
- ▶ It is worth to distribute even for data not that „big”

Summary

Take home messages

- ▶ If your data is „big” you have to go multi-core/machine
- ▶ If multi-machine, then try MapReduce
- ▶ For distributed graph algorithms it is useful to distinguish three families
- ▶ The families behave differently
- ▶ It is worth to distribute even for data not that „big”

Summary

Take home messages

- ▶ If your data is „big” you have to go multi-core/machine
- ▶ If multi-machine, then try MapReduce
- ▶ For distributed graph algorithms it is useful to distinguish three families
 - ▶ The families behave differently
 - ▶ It is worth to distribute even for data not that „big”

Summary

Take home messages

- ▶ If your data is „big” you have to go multi-core/machine
- ▶ If multi-machine, then try MapReduce
- ▶ For distributed graph algorithms it is useful to distinguish three families
- ▶ The families behave differently
- ▶ It is worth to distribute even for data not that „big”

Summary

Take home messages

- ▶ If your data is „big” you have to go multi-core/machine
- ▶ If multi-machine, then try MapReduce
- ▶ For distributed graph algorithms it is useful to distinguish three families
- ▶ The families behave differently
- ▶ It is worth to distribute even for data not that „big”

Márton Balassi

`mbalassi@ilab.sztaki.hu`

Hungarian Academy of Sciences – Institute for Computer Science and Control
Data Mining & Search Group



Jiří Barnat, Jakub Chaloupka, and Jaco Van De Pol.
Distributed algorithms for scc decomposition.
J. Log. and Comput., 21(1):23–44, February 2011.



Lars Backstrom, Dan Huttenlocher, Jon Kleinberg, and Xiangyang Lan.
Group formation in large social networks: Membership, growth, and evolution.
In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 44–54, 2006.



Jeffrey Dean and Sanjay Ghemawat.
Mapreduce: Simplified data processing on large clusters.
In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.



Péter Englert, Márton Balassi, Balázs Kósa, and Attila Kiss.
Efficiency issues of computing graph properties of social networks.

In Proceedings of the 9th International Conference on Applied Informatics, page to appear, 2014.



Balázs Kósa, Márton Balassi, Péter Englert, and Attila Kiss.
Betweenness versus linerank, 2014.

sent to the 6th International Conference on Computational Collective Intelligence Technologies and Applications,
download from: <http://people.inf.elte.hu/balhal/publications/BetweennessVsLinerank.pdf>.



U Kang, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong.

Centralities in large networks: Algorithms and observations.
In *SDM*, pages 119–130. SIAM / Omnipress, 2011.



Jure Leskovec and Christos Faloutsos.

Sampling from large graphs.

In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 631–636, 2006.



Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg.

Signed networks in social media.

In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370, 2010.



Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney.

Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters, 2008.




[cite arxiv:0810.1355](#) Comment: 66 pages, a much expanded version of our WWW 2008 paper.



Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski.

Pregel: a system for large-scale graph processing.

In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 135–146, 2010.

-  William McLendon III, Bruce Hendrickson, Steven J. Plimpton, and Lawrence Rauchwerger.
Finding strongly connected components in distributed graphs.
Journal of Parallel and Distributed Computing, 65(8):901–910, August 2005.
-  L. Page, S. Brin, R. Motwani, and T. Winograd.
The pagerank citation ranking: Bringing order to the web.
In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, 1998.
-  Leslie G. Valiant.
A bridging model for parallel computation.
Commun. ACM, 33(8):103–111, August 1990.