PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs

Summary Presentation for R244 Seminar 3

Samuel Stark

25/10/2021

University of Cambridge

Background

- Graph-Parallel Computation
  - Run a *vertex-program* on all vertices on graph
  - Vertex-program communicates with adjacent vertices
  - Each vertex ends up with a value (eg. rank in PageRank, distance in SSSP)
  - Many data dependencies => MapReduce isn't suitable[Low+10]
- In 2012, main system is Google's Pregel[Mal+10] + similar implementations
  - Piccolo[PL10], Giraph[21b]
- GraphLab[Low+10] also released in 2010
  - Prequel to PowerGraph, shares most authors

Problems...

Pregel, GraphLab did not split vertices between nodes.
Gonzalez et al. observe challenges for *asymmetric* graphs...

- Work Imbalance
- Scalability Issues
- Partitioning Difficulties eg. [Lan04]
- Communication Bottlenecks
- Storage Requirements

Natural graphs have a *skewed power-law* degree distribution, so we need to deal with asymmetry...

Can we split vertices between nodes?
We need to parallelize vertex programs!

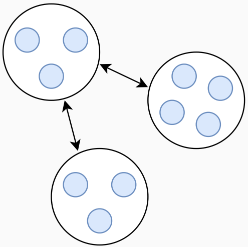

Figure 1: Example partitioning for symmetric node distribution

Problems...

Pregel, GraphLab did not split vertices between nodes.
Gonzalez et al. observe challenges for *asymmetric* graphs...

- Work Imbalance
- Scalability Issues
- Partitioning Difficulties eg. [Lan04]
- Communication Bottlenecks
- Storage Requirements

Natural graphs have a *skewed power-law* degree distribution, so we need to deal with asymmetry...

Can we split vertices between nodes?
We need to parallelize vertex programs!

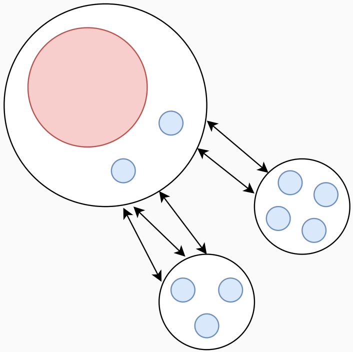

Figure 2: Example partitioning for *asymmetric* node distribution

Problems...

Pregel, GraphLab did not split vertices between nodes.
Gonzalez et al. observe challenges for *asymmetric* graphs...

- Work Imbalance
- Scalability Issues
- Partitioning Difficulties eg. [Lan04]
- Communication Bottlenecks
- Storage Requirements

Natural graphs have a *skewed power-law* degree distribution, so we need to deal with asymmetry...

Can we split vertices between nodes?
We need to parallelize vertex programs!

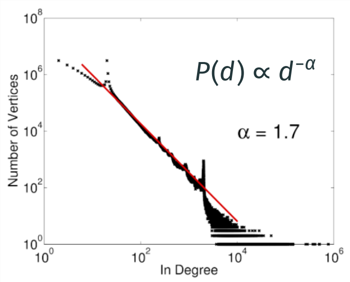

Figure 3: In-degree distributions for Twitter follower network[Gon+12]

Problems...

Pregel, GraphLab did not split vertices between nodes.
Gonzalez et al. observe challenges for *asymmetric* graphs...

- Work Imbalance
- Scalability Issues
- Partitioning Difficulties eg. [Lan04]
- Communication Bottlenecks
- Storage Requirements

Natural graphs have a *skewed power-law* degree distribution, so we need to deal with asymmetry...

Can we split vertices between nodes?
We need to parallelize vertex programs!

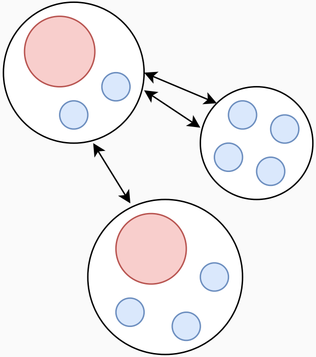

Figure 4: Example split-vertex partitioning for *asymmetric* node distribution

PowerGraph Parallelization

Gather-Apply-Scatter

Gonzalez et al. observe that (most) vertex programs have three distinct phases:

1. Gather
2. Apply
3. Scatter

```
Message combiner(m1, m2) :
   return Message(m1.value() +
      m2.value());

void PregelPageRank(msg) :
   float total = msg.value();
   vertex.val = 0.15 + 0.85*total;
   foreach(nbr in out_neighbors) :
      SendMsg(nbr, vertex.val);
```

Figure 5: PageRank in Pregel[Gon+12]

```
void GraphLabPageRank(Scope scope) :
   float accum = 0;
   foreach (nbr in scope.in_nbrs) :
      accum += nbr.val;

   vertex.val = 0.15 + 0.85*accum;

   // No explicit message passing
```

Figure 6: PageRank in GraphLab[Gon+12]

Gather-Sum-Apply-Scatter

PowerGraph adds an extra stage:

1. Gather
2. Sum
3. Apply
4. Scatter

Gather + Sum parallelized across nodes, eventually get a single sum-of-gathers

Apply on one node

Scatter parallelized across nodes

```
gather(D_u, D(u,v), D_v):
  return Dv.rank

sum(a, b): return a + b

apply(D_u, acc):
  rnew = 0.15 + 0.85*acc
  D_u.delta = (rnew - D_u.rank)
  D_u.rank = rnew

scatter(D_u, D(u,v), D_v):
  if(|D_u.delta| > ε) Activate(v)
  return delta
```

Figure 7: PageRank in PowerGraph[Gon+12]

Delta Caching

PowerGraph also allows for *delta-caching*.

It remembers the last sum value, and if your neighbours have changed, they'll apply a delta to it.

If a vertex's value hasn't changed, you don't need to talk to it.

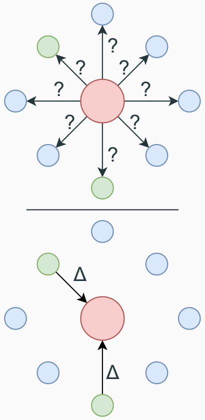

Figure 8: Comparison of no delta caching vs delta caching

Parallelization

Great, now we can split vertex computation across multiple nodes!

But how do we partition vertices effectively?

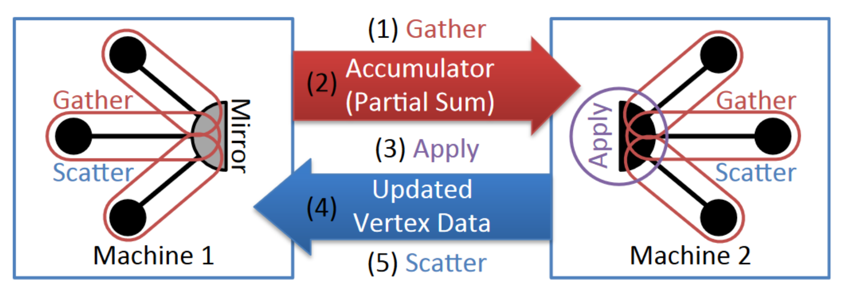

Figure 9: How PowerGraph splits computation across nodes[Gon+12]

PowerGraph Partitioning

Improving Edge Cut

Balanced *p*-way Edge Cut

Assign vertices to nodes, balance the number of cut edges

Overhead $\propto n_{edgecuts}$

Used by Pregel, GraphLab

Bad for power-law graphs

Falls back to random vertex placement, which is bad[Gon+12]

Balanced *p*-way Vertex Cut

Randomly assign edges to nodes, *should* balance cut vertices

Overhead $\propto n_{vertexcuts}$

Good for regular and power-law graphs[Gon+12]

Balanced edges bring balanced communication and storage

Proven to be strictly better than edge cuts

Greedy Vertex Cut

Instead of randomly assigning edges...

assign the next edge to the least bad node!

Track the assignment of each vertex, use a ruleset to determine where to place the next edge.

Guaranteed to be no worse (and usually better) than random placement...

but it's not embarrassingly parallel!

Greedy Vertex Cut

Instead of randomly assigning edges...

assign the next edge to the least bad node!

Track the assignment of each vertex, use a ruleset to determine where to place the next edge.

Guaranteed to be no worse (and usually better) than random placement...

but it's not embarrassingly parallel!

Greedy Vertex Cut

Instead of randomly assigning edges...

assign the next edge to the least bad node!

Oblivious

Cheat!

Just track your own assignments, don't check anyone else's.

Coordinated

Maintain a distributed database of assignments

Local caching reduces communication, but decreases accuracy

Greedy Vertex Cut

Instead of randomly assigning edges...

assign the next edge to the least bad node!

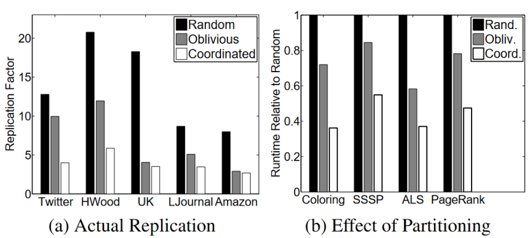

Figure 10: Impact of Greedy Vertex Cuts on vertex cuts and runtime

Implementations

Synchronous

Run every v-program once, waits for others to finish, starts again.

Cannot execute some programs

eg. Graph Coloring

Asynchronous

Run v-programs in parallel, don't wait for other programs

Allow arbitrary interleaving.

Non-deterministic, can lead to divergence

Async+Serialized

Run v-programs in parallel, except for vertices on the same edge.

All parallel executions have an equivalent serial execution.

Deterministic

Pregel is Synchronous, GraphLab is Async+Serialized

Evaluations

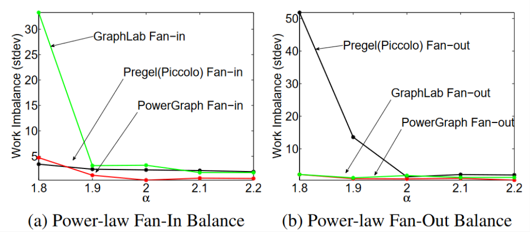

Figure 11: Work Imbalance on power-law graphs[Gon+12]

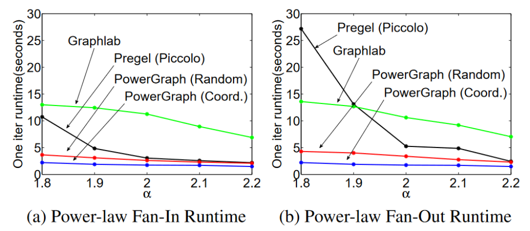

Figure 12: Runtime on power-law graphs[Gon+12]

Evaluations

PageRank	Runtime	$\lvert V \rvert$	$\lvert E \rvert$	System
Hadoop	198 s	-	1.1 B	50x8
Spark	97.4 s	40M	1.5 B	50×2
Twister	36 s	50M	1.4 B	64x4
PowerGraph (Sync)	3.6 s	40M	1.5 B	64x8

Table 1: Relative performance of PageRank vs other systems[Gon+12]

Where are they now?

- GraphLab[Low+10] -> PowerGraph[Gon+12], GraphChi[KBG12]
  - PowerGraph basically deprecated since 2015
  - GraphX implemented PowerGraph on Spark[Xin+13], now merged into Spark[21a]
- Prof. Carlos Guestrin started GraphLab, Inc -> Dato, Inc -> Turi
  - Turi was bought by Apple in 2016[Sop16], Prof. Guestrin now Head of ML at Apple
  - Main product is (GraphLab|Turi) Create, built for generic ML.
- PowerGraph was still influential!
  - 400+ citations
  - eg. Liu et al. observes the partitioning methods were used in GraphBuilder[JLW13] and then built upon in PowerLyra[Che+15], LightGraph[Zha+14]

Summary

Pros

- Splitting vertex computation across nodes is cool
- Parallelizing GAS is very cool
- Paper seems very foundational
- Paper has had lasting impact

Cons

- Limited to single vertex computations, less well suited to multi-stage or global computations (GPS is a Pregel-based system that attacked this[SW13])
- Gather-Apply-Scatter isn't always intuitive, as observed by [SW14]
- Combined Implementation+Evaluation section leads to lack of clarity.

Questions/Comments?

References

- *Apache Spark*. The Apache Software Foundation, 24th October 2021. URL: https://github.com/apache/spark (visited on 24/10/2021).
- *Apache/Giraph*. The Apache Software Foundation, 18th October 2021. URL: https://github.com/apache/giraph (visited on 24/10/2021).

Rong Chen et al. 'PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs'. In: *Proceedings of the Tenth European Conference on Computer Systems*. EuroSys '15: Tenth EuroSys Conference 2015. Bordeaux France: ACM, 17th April 2015, pp. 1–15. ISBN: 978-1-4503-3238-5. DOI: 10/gm7pv7. URL: https://dl.acm.org/doi/10.1145/2741948.2741970 (visited on 24/10/2021).

Joseph E. Gonzalez et al. 'PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs'. In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. OSDI'12. USA: USENIX Association, 8th October 2012, pp. 17–30. ISBN: 978-1-931971-96-6.

Nilesh Jain, Guangdeng Liao and Theodore L. Willke. 'GraphBuilder: Scalable Graph ETL Framework'. In: *First International Workshop on Graph Data Management Experiences and Systems*. GRADES '13. New York, NY, USA: Association for Computing Machinery, 23rd June 2013, pp. 1–6. ISBN: 978-1-4503-2188-4. DOI: 10/gm7pz3. URL: https://doi.org/10.1145/2484425.2484429 (visited on 24/10/2021).

Aapo Kyrola, Guy Blelloch and Carlos Guestrin. 'GraphChi: Large-Scale Graph Computation on Just a PC'. In: *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*. OSDI'12. USA: USENIX Association, 8th October 2012, pp. 31–46. ISBN: 978-1-931971-96-6.

Kevin Lang. 'Finding Good Nearly Balanced Cuts in Power Law Graphs'. In: (2004), p. 10.

- Ning Liu et al. 'Large-Scale Graph Processing Systems: A Survey'. In: *Frontiers of Information Technology & Electronic Engineering* 21.3 (March 2020), pp. 384–404. ISSN: 2095-9184, 2095-9230. DOI: 10/ghszds. URL: http://link.springer.com/10.1631/FITEE.1900127 (visited on 24/10/2021).
- Yucheng Low et al. *GraphLab: A New Framework for Parallel Machine Learning*. 25th June 2010. arXiv: 1006.4990 [cs]. URL: http://arxiv.org/abs/1006.4990 (visited on 24/10/2021).
- Grzegorz Malewicz et al. 'Pregel: A System for Large-Scale Graph Processing'. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD '10. New York, NY, USA: Association for Computing Machinery, 6th June 2010, pp. 135–146. ISBN: 978-1-4503-0032-2. DOI: 10/fh62wr. URL: https://doi.org/10.1145/1807167.1807184 (visited on 24/10/2021).

References v

Russell Power and Jinyang Li. ‘Piccolo: Building Fast, Distributed Programs with Partitioned Tables’. In: *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. OSDI’10. USA: USENIX Association, 4th October 2010, pp. 293–306.

Taylor Soper. *Exclusive: Apple Acquires Turi in Major Exit for Seattle-Based Machine Learning and AI Startup*. GeekWire. 5th August 2016. URL: https://www.geekwire.com/2016/exclusive-apple-acquires-turi-major-exit-seattle-based-machine-learning-ai-startup/ (visited on 24/10/2021).

Semih Salihoglu and Jennifer Widom. ‘GPS: A Graph Processing System’. In: *Proceedings of the 25th International Conference on Scientific and Statistical Database Management - SSDBM*. The 25th International Conference. Baltimore, Maryland: ACM Press, 2013, p. 1. ISBN: 978-1-4503-1921-8. DOI: 10/gm7pzz. URL: http://dl.acm.org/citation.cfm?doid=2484838.2484843 (visited on 24/10/2021).

Semih Salihoglu and Jennifer Widom. ‘HelP: High-Level Primitives For Large-Scale Graph Processing’. In: GRADES Workshop on Graph Data Management Experiences and Systems. Utah: Stanford InfoLab, June 2014. URL: http://ilpubs.stanford.edu:8090/1085/ (visited on 22/10/2021).

- Reynold S. Xin et al. 'GraphX: A Resilient Distributed Graph System on Spark'. In: *First International Workshop on Graph Data Management Experiences and Systems*. GRADES '13. New York, NY, USA: Association for Computing Machinery, 23rd June 2013, pp. 1–6. ISBN: 978-1-4503-2188-4. DOI: 10/gfs42m. URL: https://doi.org/10.1145/2484425.2484427 (visited on 24/10/2021).
- Yue Zhao et al. 'LightGraph: Lighten Communication in Distributed Graph-Parallel Processing'. In: *2014 IEEE International Congress on Big Data*. 2014 IEEE International Congress on Big Data. June 2014, pp. 717–724. DOI: 10/gm7pz4.