PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs

Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, Carlos Guestrin

Edmund Goodman

October 30, 2024

Motivation

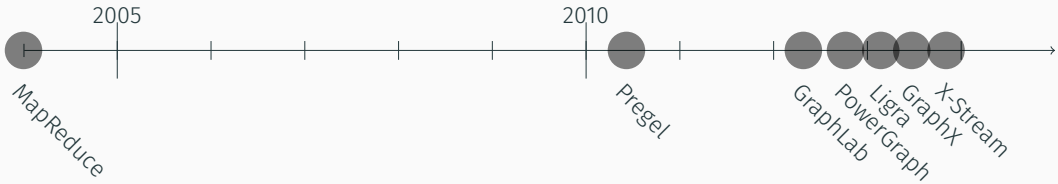
Large-scale graph-structured computation is needed for many tasks[1]

- PageRank on the web topology
- Characterising trends in social networks
- Targeted advertising

- Very similar to Pregel's[2] motivations 2 years before

[1] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs".
[2] Malewicz et al., "Pregel: a system for large-scale graph processing".

Trends in graph processing frameworks



- Google building distributed systems in the 2000s
- Followed by work in quick succession 2012–2013 particularly at Carnegie Mellon then Berkeley (GraphLab[3], PowerGraph[4], GraphX[5])
- Less frequent publications after this

[3] Low et al., *Distributed GraphLab: A Framework for Machine Learning in the Cloud.*
[4] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs".
[5] Gonzalez, Xin, et al., "GraphX: Graph Processing in a Distributed Dataflow Framework".

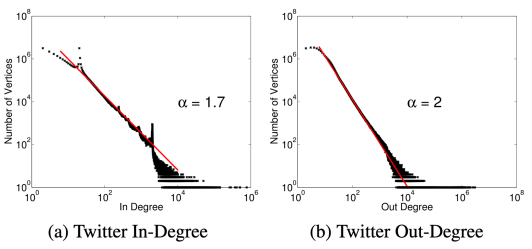
Key idea

Leveraging the internal structure of real-world graphs

- Graphs in the real world are often of a specific shape
- Existing solutions don't optimise for this, so sacrifice possible performance gains

$\Rightarrow$ Design graph-parallel abstractions with real-world graph shapes in mind

Natural graphs



(a) Twitter In-Degree (b) Twitter Out-Degree

Figure 1: "The in and out degree distributions of the Twitter follower network plotted in log-log scale"[a].

- "Natural" graphs are common in real-world data
  - Defined as skewed power law distributions $P(d) \propto d^{-\alpha}$
  - A few "celebrities" with many in-edges
- The internet topology graph is $\alpha \approx 2.2$[a]

[a] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", Figure 1.

[a] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology".

Graph-parallel abstractions

- A vertex program Q over a sparse graph $G = \{V, E\}$ is[6]:
  - Executed concurrently on each vertex $v \in V$
  - Able to interact with adjacent instances $Q(u), \exists(u, v) \in E$
- Constrains communications between vertices
- Basis of existing work by Pregel[7] and GraphLab[8]

[6] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", p. 2.
[7] Malewicz et al., "Pregel: a system for large-scale graph processing".
[8] Low et al., *Distributed GraphLab: A Framework for Machine Learning in the Cloud*.

Gather-Apply-Scatter model

PowerGraph then proposes a programming model combining desirable properties of Pregel[9] and GraphLab[10]:

1. **G**ather – Aggregate adjacent vertex and edge values at end of previous superstep
   - `gather(`D_u`, `$D_{(u,v)}$`, `D_v`) → Accum`
   - `sum(Accum left, Accum right) → Accum` *(must be associative and commutative)*
2. **A**pply – Apply function to value of vertex
   - `apply(`D_u`,Accum) →` D_{unew} *(must be sub-linear for natural graphs performance)*
3. **S**catter – Update adjacent edge values with vertex value at end of superstep
   - `scatter(`D_{unew}`, `$D_{(u,v)}$`, `D_v`) → (`$D_{(u,v)new}$`, Accum)`

Can emulate previous approaches with linear `apply`, but loses performance uplift

[9] Malewicz et al., "Pregel: a system for large-scale graph processing".
[10] Low et al., *Distributed GraphLab: A Framework for Machine Learning in the Cloud.*

Distributed graph placement

- Key factor in performance is how work is distributed across machines
- Pregel and GraphLab uses a balanced p-way edge cut, falling back to random

- Instead, PowerGraph uses a balanced p-way **vertex** cut
  - ”Shatters” graph by few popular vertices, minimising traffic and unbalanced work
  - This is approximated with a greedy algorithm to assign edges across machines
  - This greedy algorithm is then run separately or together across machines

$$\underset{k}{\operatorname{argmin}}\, \mathbb{E}\left[\sum_{v \in V} |A(v)| \,\middle|\, A_i, A(e_{i+1}) = k\right]$$

Figure 2: Greedy heuristic: the lowest expected number of replicas given current assigned edges[11]

[11] Gonzalez, Low, et al., “PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs”, Equation 5.13.

Performance – work balance

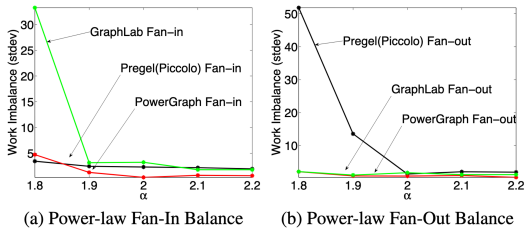


Figure 3: "Standard deviation of worker computation time across 8 distributed workers for each abstraction on power-law fan-in and fan-out graphs"[a].

- Aim to spread computation evenly
- Variance in computation measures placement effectiveness
- GraphLab and Pregel are imbalanced for either highly skewed fan-in or fan-out respectively
- PowerGraph is (comparatively) balanced for both fan-in and fan-out

[a] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", Figure 9.

Performance – communication

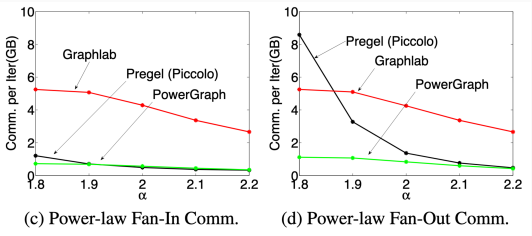


Figure 4: "Bytes communicated per iteration for each abstraction on power-law fan-in and fan-out graphs"[a].

[a] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", Figure 9.

- Aim to minimise communication between machines
- Data transferred per iteration measures graph placement effectiveness
- GraphLab has consistently high traffic
  - Asynchronous shared memory coherence?
- Pregel has high traffic for highly skewed fan-out only
- PowerGraph is (comparatively) low for both fan-in and fan-out

Performance – runtime

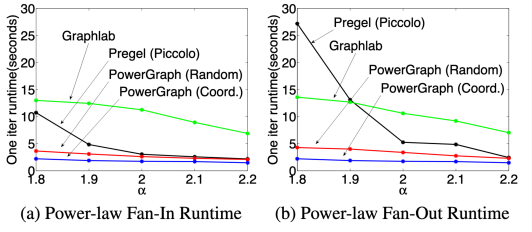


Figure 5: "Per iteration runtime of each abstraction on synthetic power-law graphs"[a].

[a] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", Figure 10.

- PowerGraph outperforms previous approaches for highly skewed ($\alpha \approx 1$)
  - Co-ordinated placement is faster
  - Approaches converge for larger α
  - Note convergence of Pregel is $\alpha \approx 2.2$, the skew value of the internet topology graph cited ealier...
- Runtime dominated by communication rather than compute
  - PageRank is not very computationally intensive

Criticism

Strengths:

+ Novel idea to leverage information about structure of real-world graphs
+ Theoretically justified and empirically measured performance uplift

Weaknesses:

– Weak justification for why Piccolo[12] can proxy for Pregel
– Incomparable rows in Table 2 of performance results » appendix
– Motivation weakened by real-world graph skewness being approximately the convergence point of performance

Takeaway message:

- Drawing information from your problem domain can yield signicant benefits

[12] Power and Li, "Piccolo: Building fast, distributed programs with partitioned tables".

Conclusions

Existing work processes large graphs, we want to process real-world large graphs for which these approaches struggle:

- ⇒ New **G**ather-**A**pply-**S**catter programming model
- ⇒ New graph placement algorithm using vertex cuts and a greedy heuristic
- ⇒ Theoretically and empirically show new approach is more performant and efficient than previous work

Future work and impact

Possible avenues of future work:

- More directly comparable performance measurements between technologies
- Any common graph structures other than skewed power-law? How do these perform?
- Improve supported for serialisable asynchronous execution

Historical impact:

- PowerGraph later included back into GraphLab project, from which the Turi company was formed – co-founded by first author Gonzalez
- Follow-up project GraphX, also by Gonzalez, later included in Apache Spark[13]
- Turi acquired by Apple Inc. in 2016 and developed into TuriCreate[14], used until late 2023 for developing machine learning models

[13] *GraphX - Spark 3.5.3 Documentation.*
[14] *apple/turicreate.*

PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs

Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, Carlos Guestrin

Edmund Goodman

October 30, 2024

PageRank	Runtime	$\|V\|$	$\|E\|$	System
Hadoop [22]	198s	–	1.1B	50x8
Spark [37]	97.4s	40M	1.5B	50x2
Twister [15]	36s	50M	1.4B	64x4
PowerGraph (Sync)	3.6s	40M	1.5B	64x8

Triangle Count	Runtime	$\|V\|$	$\|E\|$	System
Hadoop [36]	423m	40M	1.4B	1636x?
PowerGraph (Sync)	1.5m	40M	1.4B	64x16

LDA	Tok/sec	Topics	System
Smola et al. [34]	150M	1000	100x8
PowerGraph (Async)	110M	1000	64x16

Figure 6: "Relative performance of PageRank, triangle counting, and LDA on similar graphs. PageRank runtime is measured per iteration. Both PageRank and triangle counting were run on the Twitter follower network and LDA was run on Wikipedia. The systems are reported as number of nodes by number of cores."[15].

[15] Gonzalez, Low, et al., "PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs", Table 2.

References i

[1] ***apple/turicreate.*** original-date: 2017-12-01T00:42:04Z. Oct. 2024. URL: https://github.com/apple/turicreate (visited on 10/29/2024).

[2] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. **"On power-law relationships of the Internet topology".** In: *SIGCOMM Comput. Commun. Rev.* 29.4 (Aug. 1999), pp. 251–262. ISSN: 0146-4833. DOI: 10.1145/316194.316229. URL: https://dl.acm.org/doi/10.1145/316194.316229 (visited on 10/23/2024).

[3] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. **"PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs".** en. In: 2012, pp. 17–30. ISBN: 978-1-931971-96-6. URL: https://www.usenix.org/conference/osdi12/technical-sessions/presentation/gonzalez (visited on 10/23/2024).

References ii

[4] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. **"GraphX: Graph Processing in a Distributed Dataflow Framework".** en. In: 2014, pp. 599–613. ISBN: 978-1-931971-16-4. URL: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/gonzalez&sa=U&ei=iqWOVKmxBqafygPrqILQCA&ved=0CC4QuAIwA1AB&usg=AFQjCNHibMBth4tAjIBaqIgher5itftGmA (visited on 10/28/2024).

[5] ***GraphX - Spark 3.5.3 Documentation.*** URL: https://spark.apache.org/docs/latest/graphx-programming-guide.html (visited on 10/29/2024).

[6] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. ***Distributed GraphLab: A Framework for Machine Learning in the Cloud.*** arXiv:1204.6078. Apr. 2012. DOI: 10.48550/arXiv.1204.6078. URL: http://arxiv.org/abs/1204.6078 (visited on 10/23/2024).

References iii

[7] 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. New York, NY, USA: Association for Computing Machinery, June 2010, pp. 135–146. ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807184. URL: https://dl.acm.org/doi/10.1145/1807167.1807184 (visited on 10/08/2024).

[8] Russell Power and Jinyang Li. **“Piccolo: Building fast, distributed programs with partitioned tables”.** In: *9th USENIX symposium on operating systems design and implementation (OSDI 10)*. 2010.