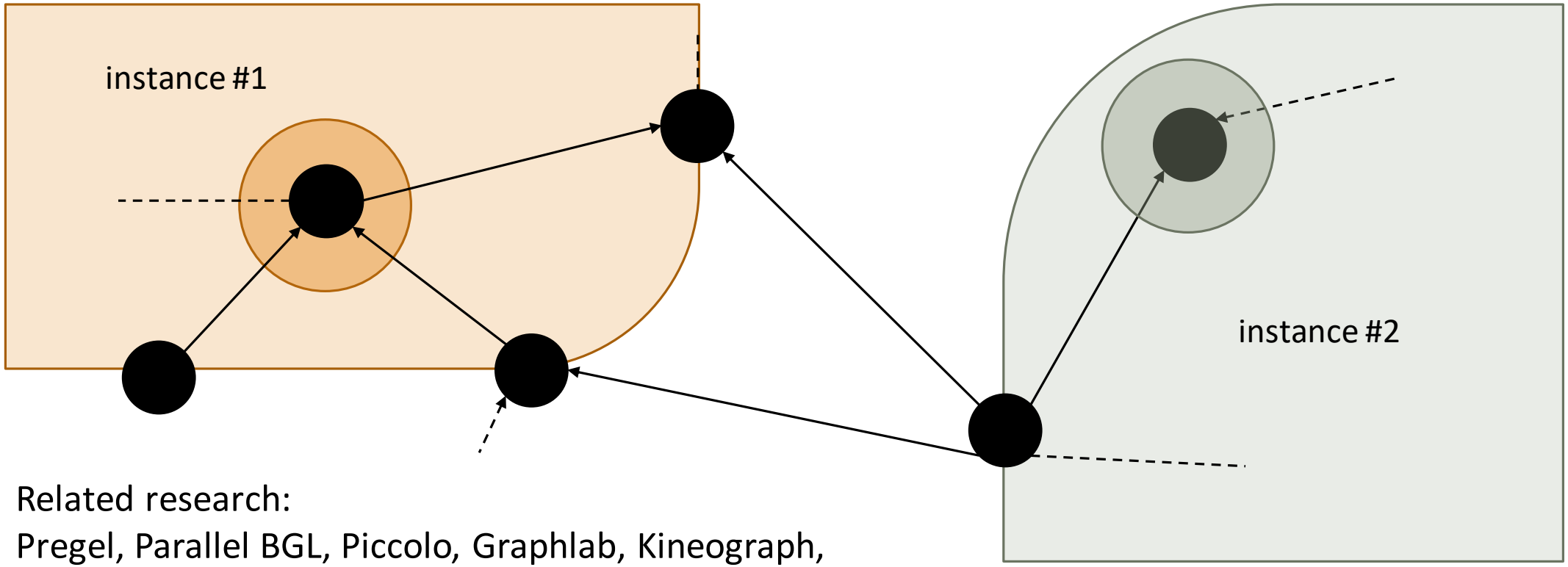
PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs

JOSEPH E. GONZALES, YUCHENG LOW, HAIJIE GU, DANNY BICKSON, CARLOS GUESTRIN

10TH USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, 2012

PRESENTED BY JULIUS LISCHEID

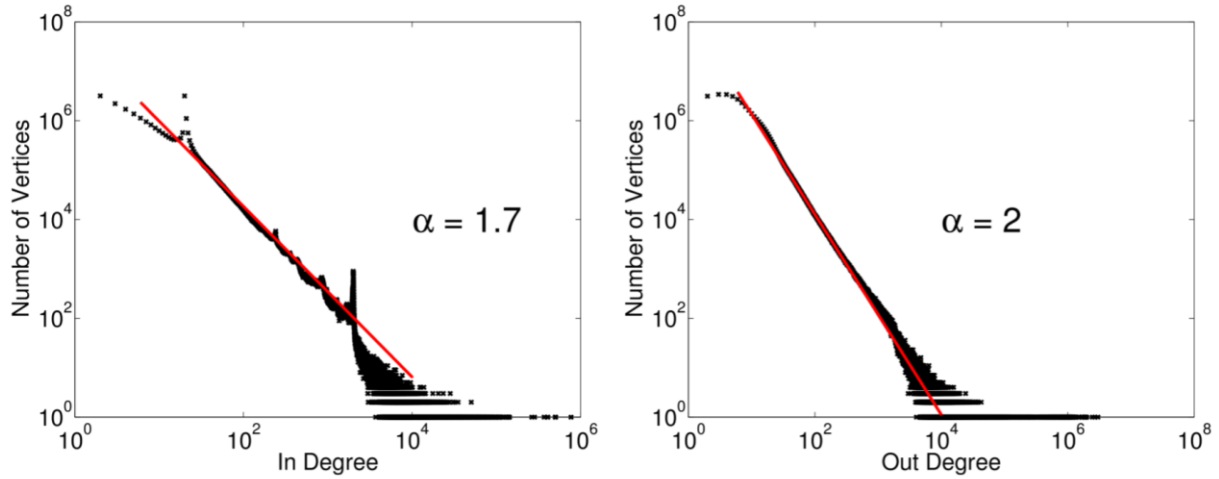
Context: What is a Vertex-Program?



Related research:
Pregel, Parallel BGL, Piccolo, Graphlab, Kineograph, GraphChi

Context: What is a Natural Graph?

Natural graphs have a skewed power-law degree distribution.

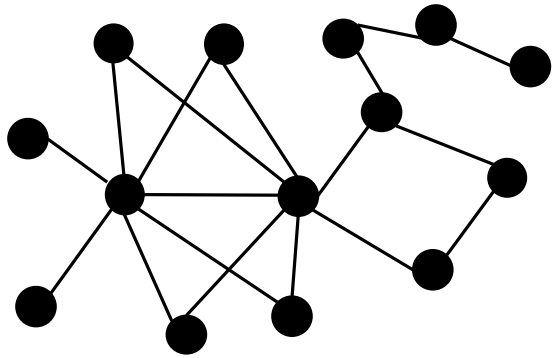


(a) Twitter In-Degree (b) Twitter Out-Degree

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

Vertex-program problems:

1. Work balance
2. Partitioning
3. Communication
4. Storage
5. Computation



The GAS Model

commutative, associative

1. Gather

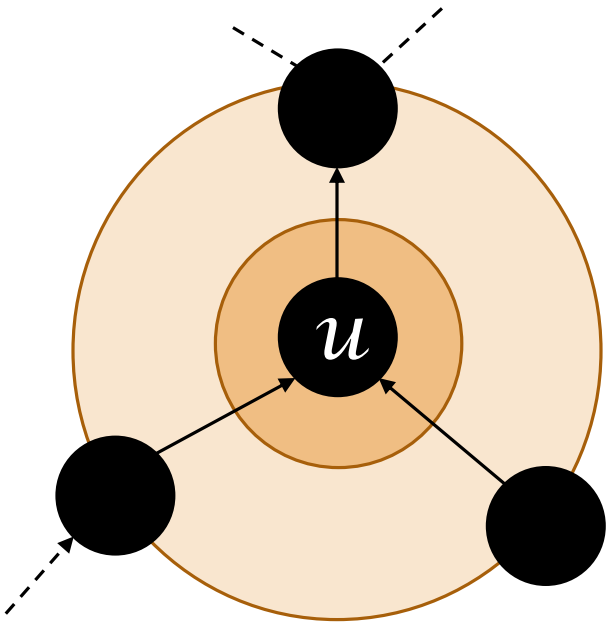
$$\Sigma \leftarrow \bigoplus_{v \in \mathbf{Nbr}[u]} g\left(D_u, D_{(u,v)}, D_v\right)$$

2. Apply

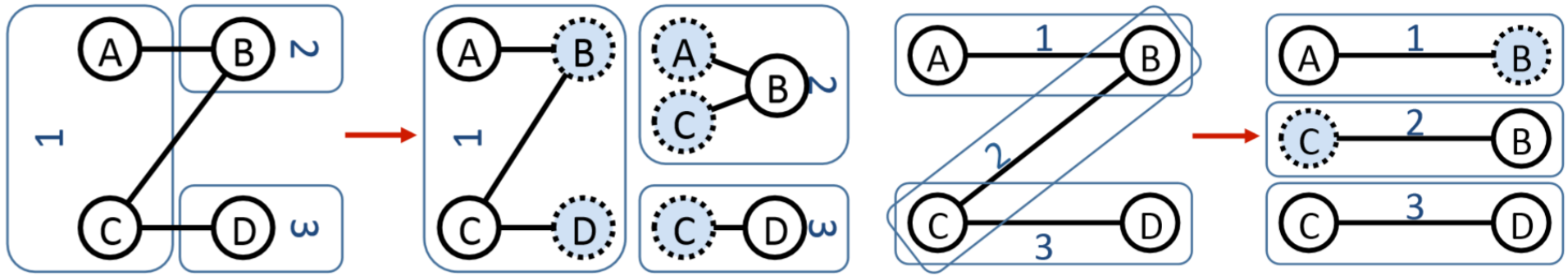
$$D_u^{\text{new}} \leftarrow a\left(D_u, \Sigma\right)$$

3. Scatter

$$\forall v \in \mathbf{Nbr}[u]: \quad \left(D_{(u,v)}\right) \leftarrow s\left(D_u^{\text{new}}, D_{(u,v)}, D_v\right)$$



Distributed Graph Placement: Edge-Cuts vs Vertex-Cuts



(a) Edge-Cut (b) Vertex-Cut

Figure 4: **(a)** An edge-cut and **(b)** vertex-cut of a graph into three parts. Shaded vertices are ghosts and mirrors respectively.

Distributed Graph Placement: Vertex-Cuts

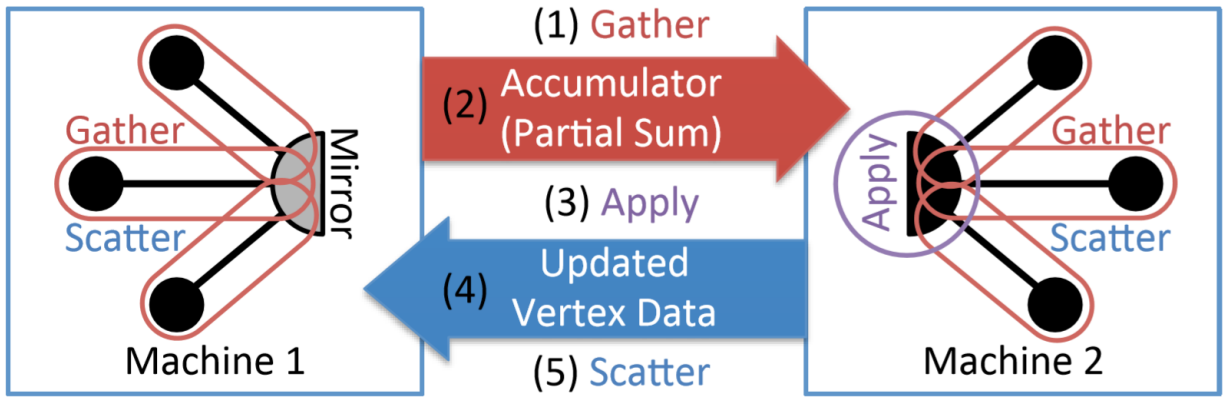
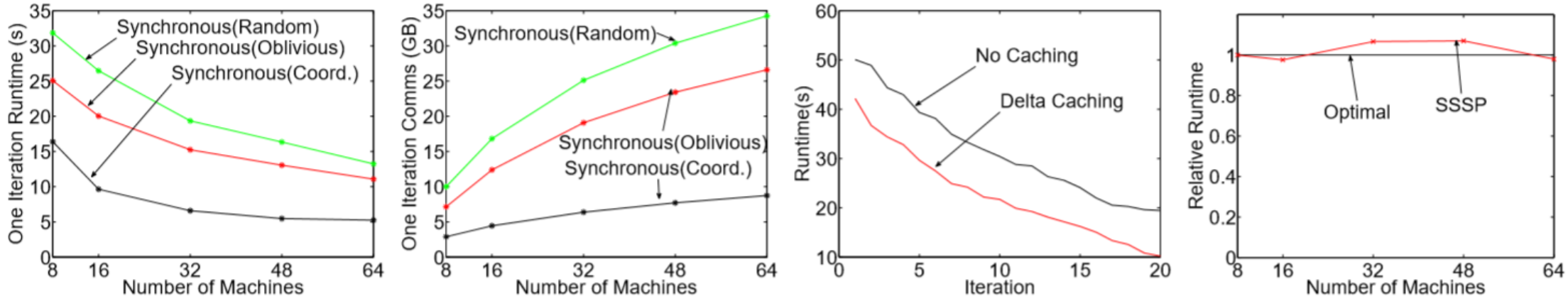
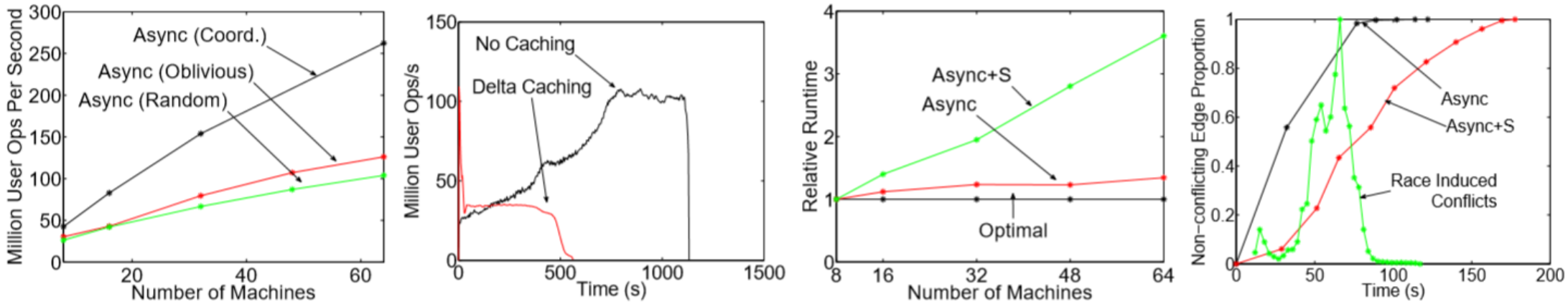


Figure 5: The communication pattern of the PowerGraph abstraction when using a vertex-cut. Gather function runs locally on each machine and then one accumulators is sent from each mirror to the master. The master runs the apply function and then sends the updated vertex data to all mirrors. Finally the scatter phase is run in parallel on mirrors.



(a) Twitter PageRank Runtime (b) Twitter PageRank Comms (c) Twitter PageRank Delta Cache (d) SSSP Weak Scaling

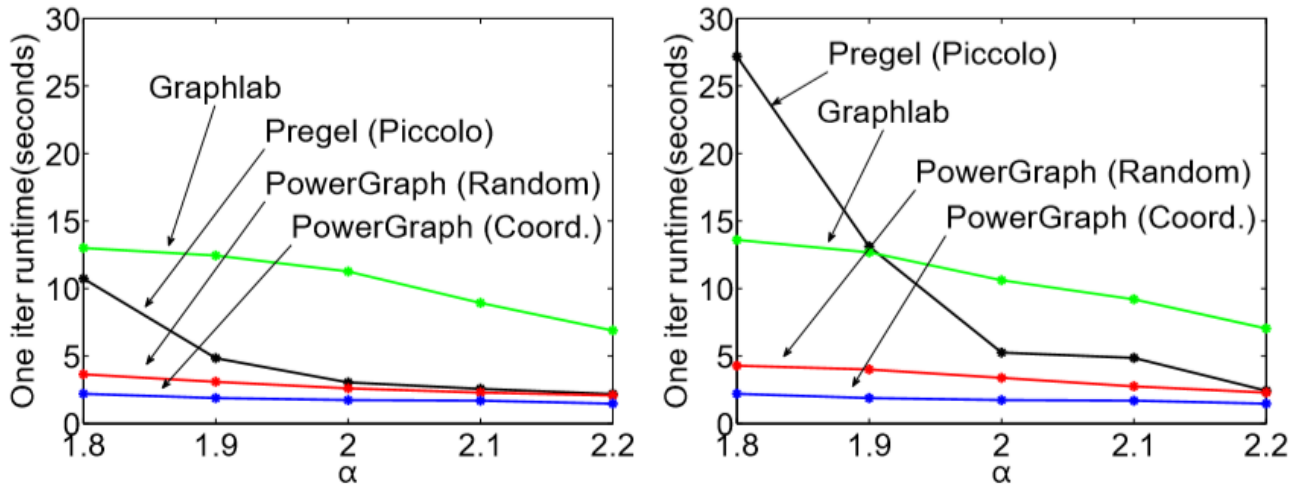
Figure 11: **Synchronous Experiments (a,b)** Synchronous PageRank Scaling on Twitter graph. **(c)** The PageRank per iteration runtime on the Twitter graph with and without delta caching. **(d)** Weak scaling of SSSP on synthetic graphs.



(a) Twitter PageRank Throughput (b) Twitter PageRank Delta Cache (c) Coloring Weak Scaling (d) Coloring Conflict Rate

Figure 12: **Asynchronous Experiments (a)** Number of user operations (gather/apply/scatter) issued per second by Dynamic PageRank as # machines is increased. **(b)** Total number of user ops with and without caching plotted against time. **(c)** Weak scaling of the graph coloring task using the Async engine and the Async+S engine **(d)** Proportion of non-conflicting edges across time on a 8 machine, 40M vertex instance of the problem. The green line is the rate of conflicting edges introduced by the lack of consistency (peak 236K edges per second) in the Async engine. When the Async+S engine is used no conflicting edges are ever introduced.

Comparison to Pregel (Piccolo) and Graphlab



(a) Power-law Fan-In Runtime (b) Power-law Fan-Out Runtime

Figure 10: **Synthetic Experiments Runtime. (a, b)** Per iteration runtime of each abstraction on synthetic power-law graphs.

Comparisons are only run on synthetic graphs.

PageRank, Triangle Count, LDA

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

Table 2: 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.

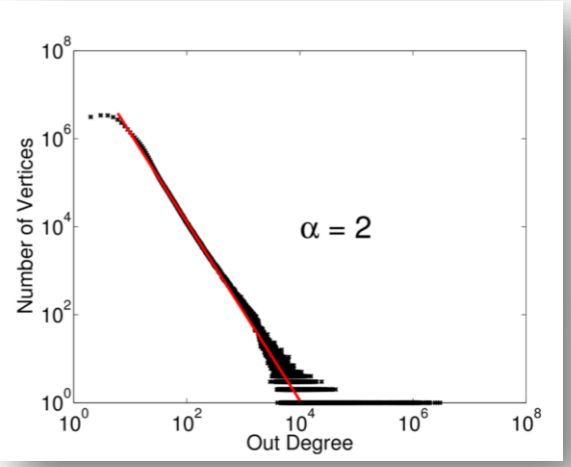
Conclusions

KEY TAKEAWAYS

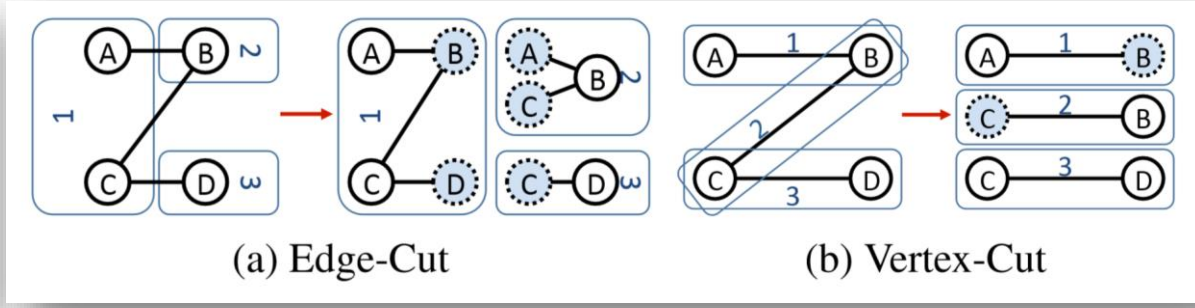
- GAS model allows for vertex-cut distribution of graphs across machines
- This can improve performance and scalability of distributed graph processing

IMPACT

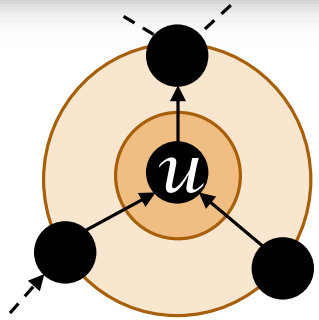
- > 1100 citations
- PowerGraph was integrated into GraphLab, sold for $200M to Apple in 2016
- Lead author moved on to GraphX, built on top of Spark

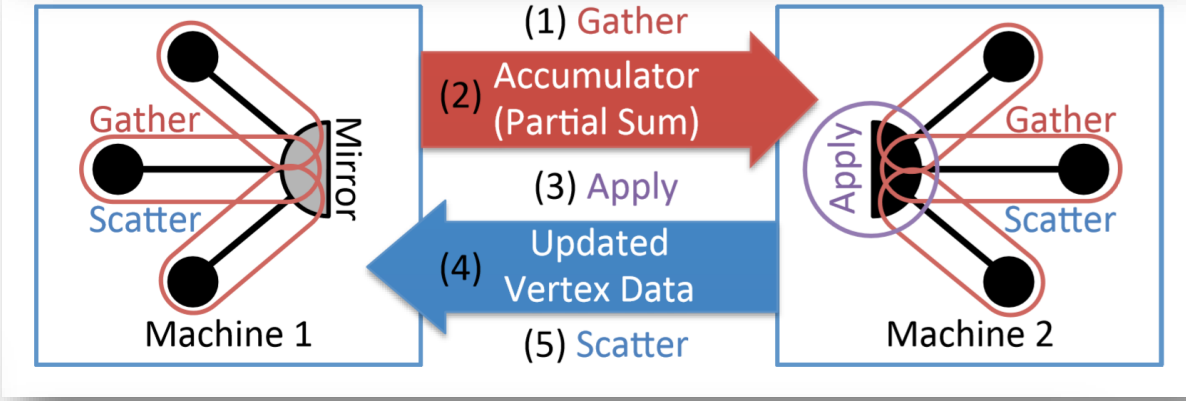


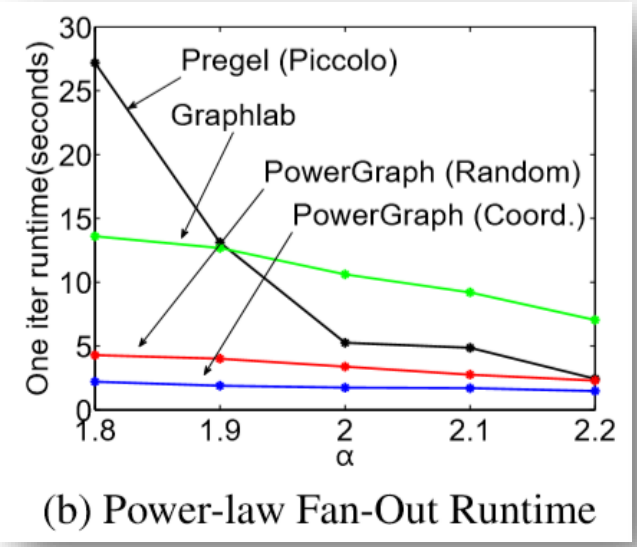

```
interface GASVertexProgram(u) {
  // Run on gather_nbrs(u)
  gather(D_u, D_(u,v), D_v) → Accum
  sum(Accum left, Accum right) → Accum
  apply(D_u, Accum) → D_u^new
  // Run on scatter_nbrs(u)
  scatter(D_u^new, D_(u,v), D_v) → (D_(u,v)^new, Accum)
}
```

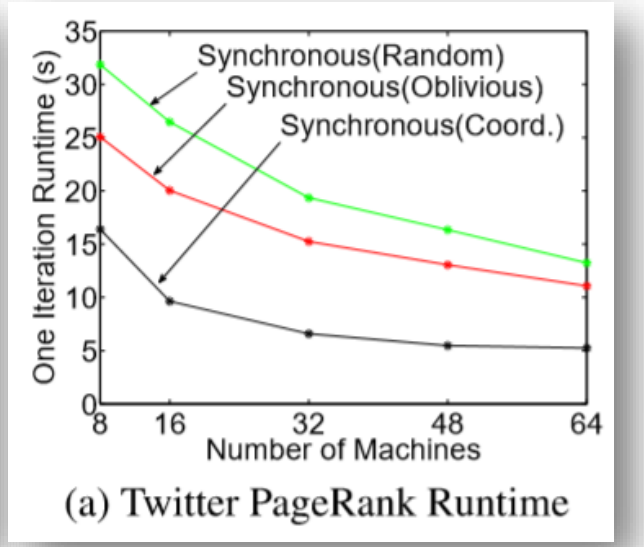


Questions?









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

Reference: Gonzalez, Joseph E and Low, Yucheng and Gu, Haijie and Bickson, Danny and Guestrin, Carlos: Powergraph: Distributed graph-parallel computation on natural graphs. Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), 2012.