Jaccard Coefficients as a Potential Graph Benchmark

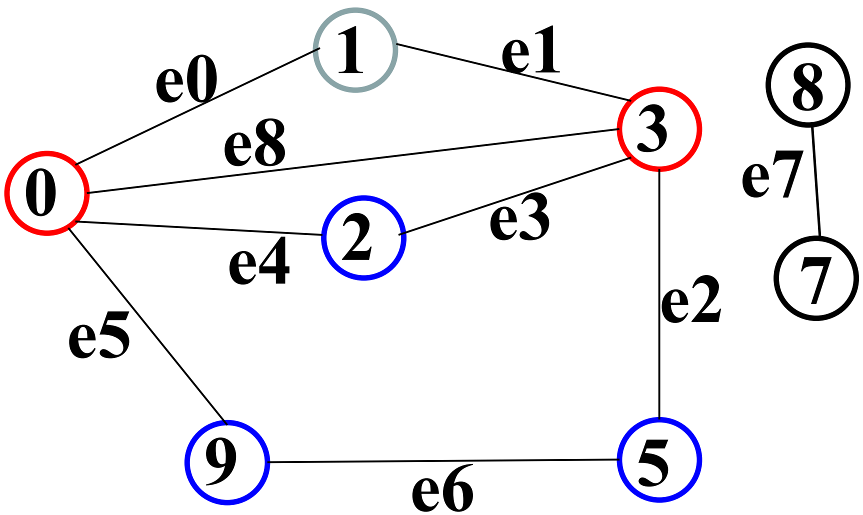
Peter M. Kogge
McCourtney Prof. of CSE
Univ. of Notre Dame
IBM Fellow (retired)

Outline

- Motivation
- Jaccard Coefficients
- A MapReduce Baseline
- Variations
- A Possible Heuristic
- Suggested Benchmarks
- Key Questions

Graph 500

- Start with a root, find reachable vertices
- Simplifications: only 1 kind of edge, no weights
- Performance metric: TEPS: Traversed Edges/sec



Level	Scale	Size	Vertices (Billion)	TB	Bytes /Vertex
10	26	Toy	0.1	0.02	281.8048
11	29	Mini	0.5	0.14	281.3952
12	32	Small	4.3	1.1	281.472
13	36	Medium	68.7	17.6	281.4752
14	39	Large	549.8	141	281.475
15	42	Huge	4398.0	1,126	281.475
				Average	281.5162

Scale = log2(# vertices)

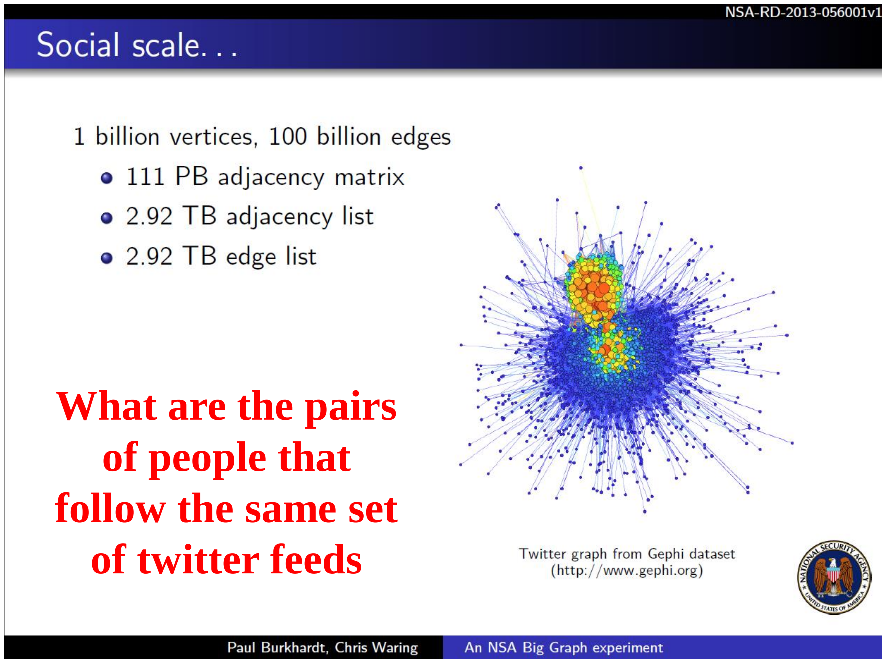
Starting at 1: 1, 0, 3, 2, 9, 5

Limitations of BFS as a Benchmark

Has provided rich set of new algorithms & implementations, BUT:

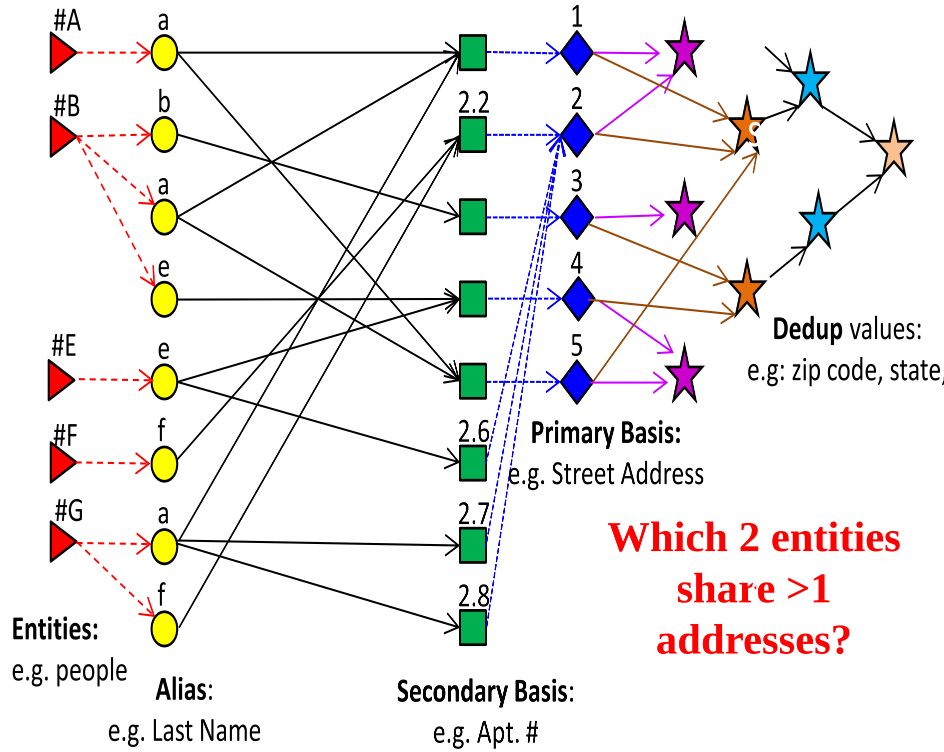
- Complexity only O(E)
  - E = # edges
- Only a batch algorithm
  - Must investigate virtually entire static graph
- No natural incremental variant
- Little non-academic applicability
  - Commercial apps much more focused on limited neighborhoods

Big Graph Relationship Problems



What are the pairs of people that follow the same set of twitter feeds

from "Burkhardt & Waring, An NSA Big Graph Experiment"
http://www.pdl.cmu.edu/SDI/2013/slides/big_graph_nsa_rd_2013_56002v1.pdf



Which 2 entities share >1 addresses?

- Tough Problem: Find vertex ***pairs*** that "share some common property"
- Related graph problem: computing "Jaccard coefficients"
- Commercial version: "Non-obvious Relationship Problems" (NORA)

Sample Real World Problem

Auto Insurance Co: "Tell me about giving auto policy to Jane Doe" in < 0.1sec

- **40+ TB of Raw Data**
- **Periodically clean up & combine to 4-7 TB**
- **Weekly "Boil the Ocean" to precompute answers to all standard queries**
  - **Does W have financial difficulties?**
  - **Does X have legal problems?**
  - **Has Y had significant driving problems?**
  - **Who has shared addresses with Z?**
  - **…**

Relationships

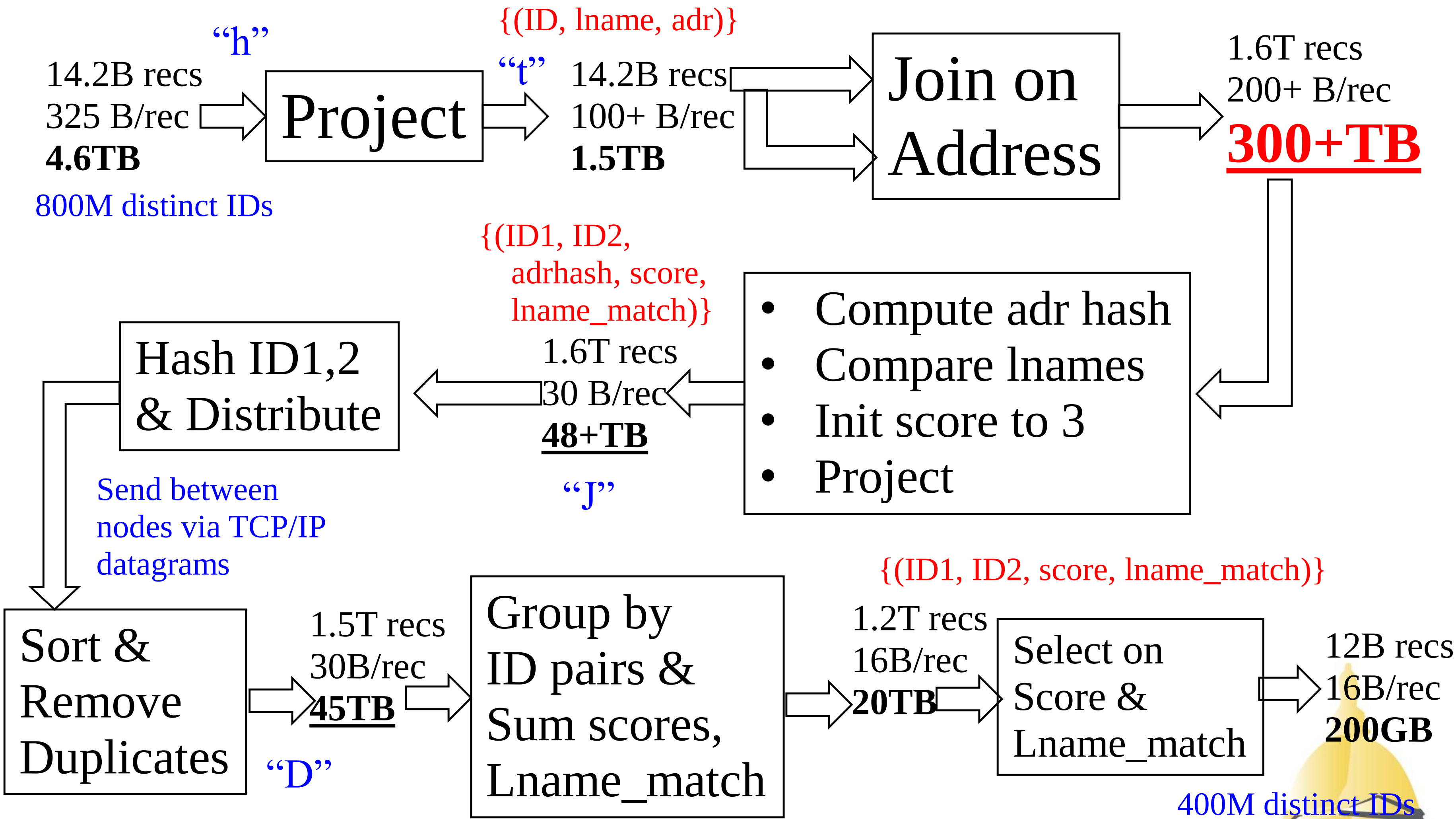
Look up answers to precomputed queries for "Jane Doe" and combine

"Jane Doe has no indicators
But
she has shared multiple addresses with Joe Scofflaw
Who has the following negative indicators …."

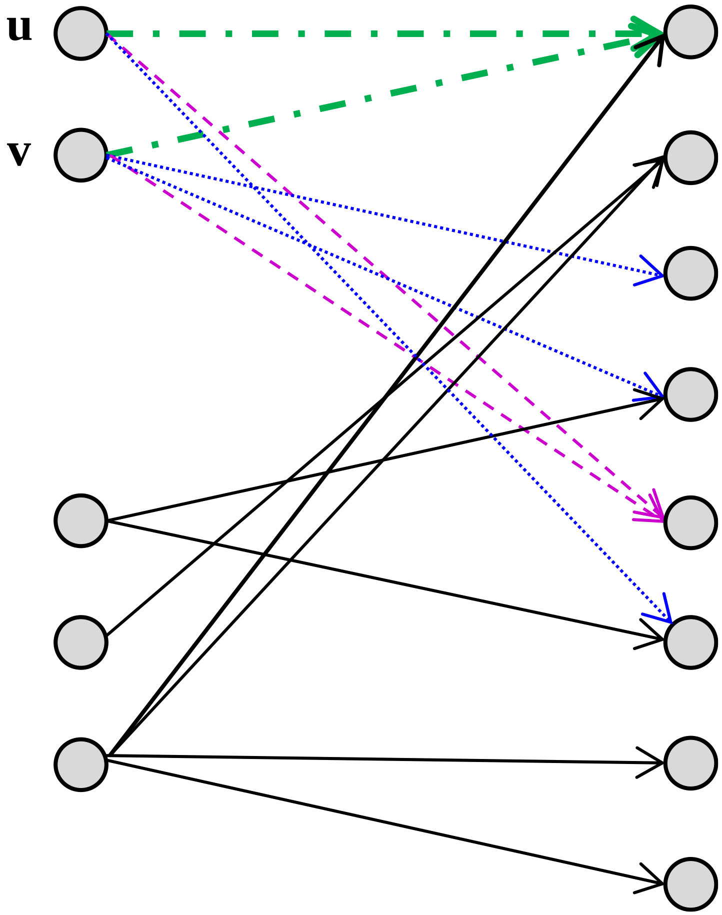
A 2012 Relationship

- Given: 14.2 billion records from
  - 800 million ***entities*** (North American people, businesses)
  - 100 million ***addresses***
- Goal: given entity, find all other entities that
  - Share at least 2 addresses in common
  - Or have one address in common and last name that is "close"
- Matching last names requires processing to check for typos ("Levenshtein distance")
- Above one of dozens or relationships

2012 Processing Flow



Jaccard Coefficient Γ(u , v)



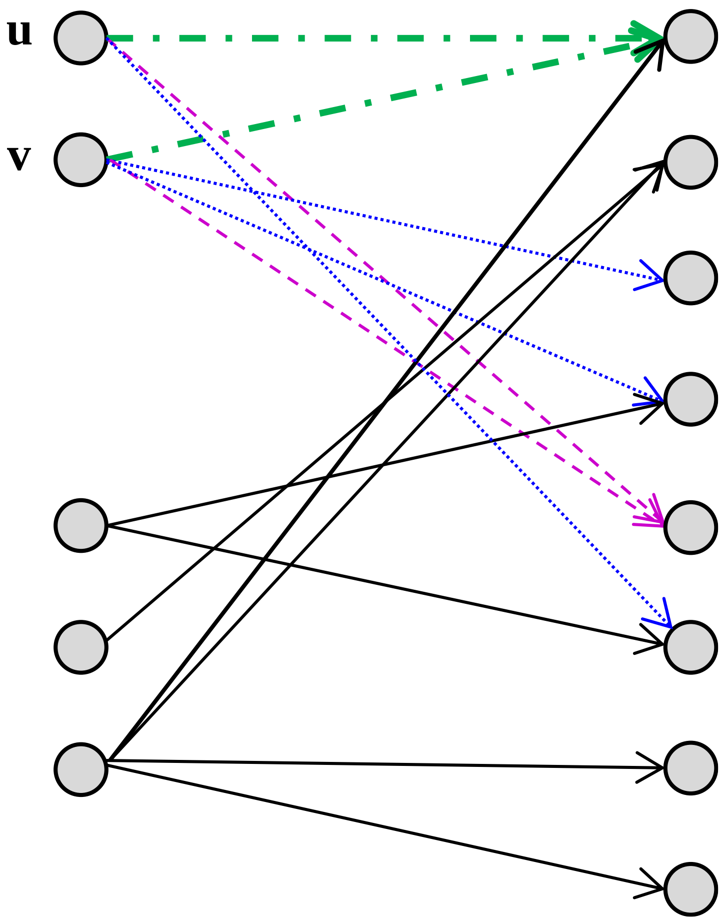
N(u) = set of neighbors of u

Γ(u,v) = fraction of neighbors of u and v that are in common

Γ(u,v) = |N(u) ∩ N(v)|/(N(u) U N(v)|

Green and Purple lead to common neighbors
Blue lead to non-common neighbors

Jaccard Coefficient Γ(u , v)



d(u) = # of neighbors of i

$\gamma(u, v)$ = # of common neighbors

$\Gamma(u,v)$ = fraction of all neighbors that are shared

$\gamma(u, v) = 2$:

$d(u)=3; d(v)=4$:

$\Gamma(u,v) = 2/5 = 0.4$

In General:

$$\Gamma(u,v) = \gamma(u, v) / (d(u)+d(v)- \gamma(u, v))$$

Green and Purple lead to common neighbors

Blue lead to non-common neighbors

Computing a Single γ

- γ(u,v) = # of common neighbors
- Do not need to "visit" neighbors, only enumerate them
- Algorithm requires comparing 2 lists of lengths d(u) & d(v)
  - Let d = max(d(u), d(v))
- If lists are sorted, then O(d)
- If lists not sorted, then O(d*log(d))
  - Clever hash algorithms may reduce to almost O(d)

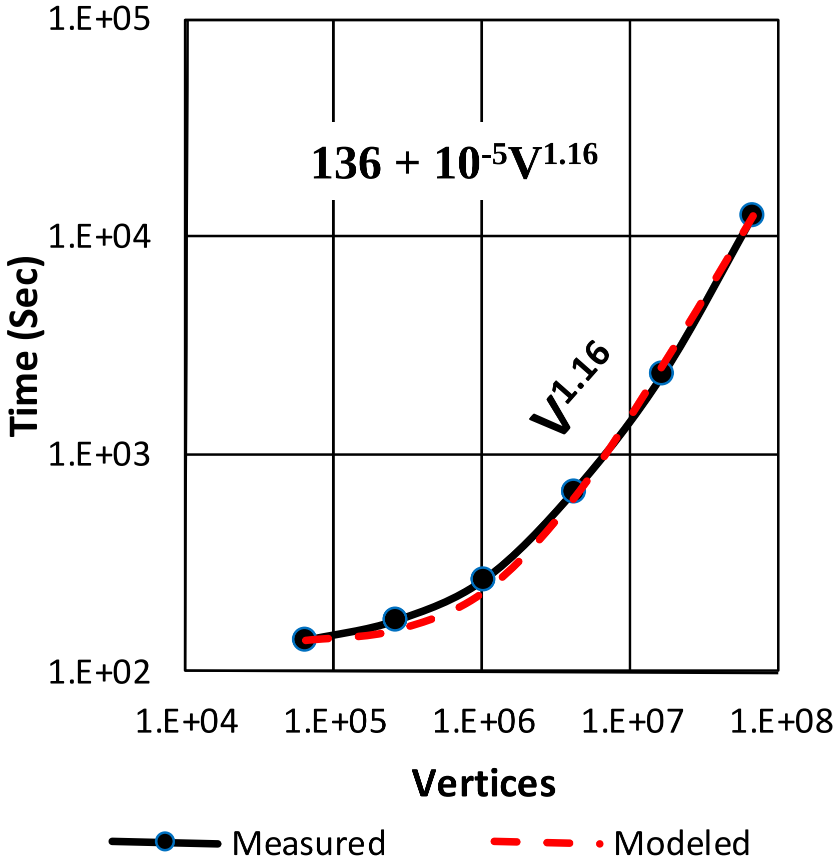
Computing all γs

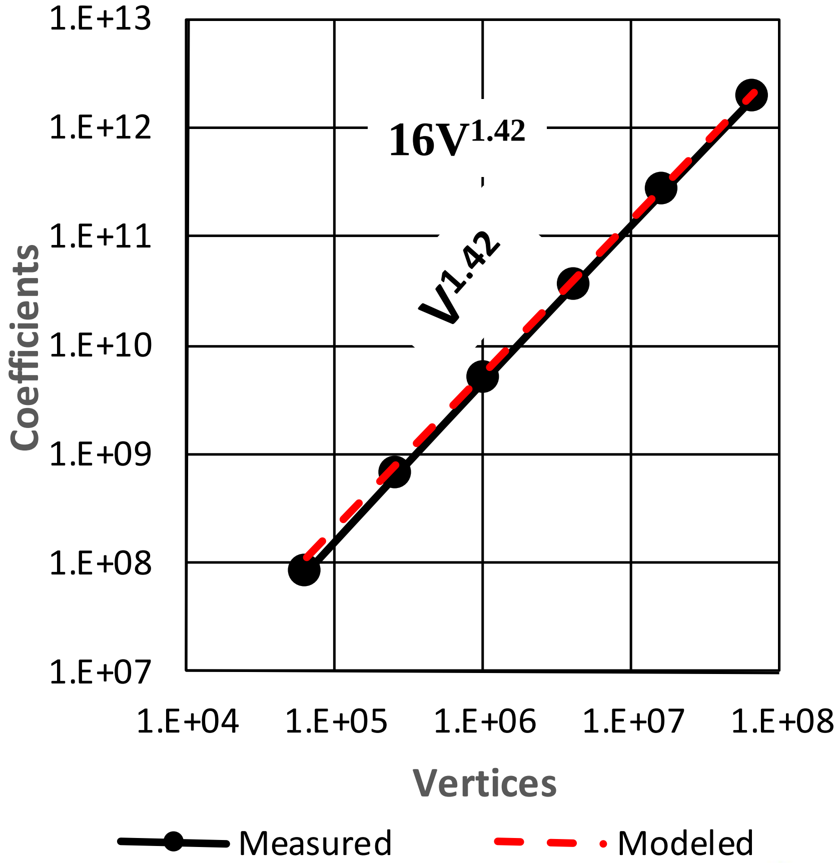
- Apply prior single γ to all pairs
  - $O(V^2d)$ to $O(V^2d*\log(d))$
  - Factor of 2 reduction if only do (u,v) where i<j
- Avoid computing all clearly 0 terms
  - For each u, explore each w, (u,w) an edge
  - Compute γ(u,v) for each v where (v,w) an edge & u<v
  - $O(V*d^3)$ to $O(V*d^4)$
- Backwards: compute γs incrementally
  - Group edges so for each w we have {x|(x,w)}
  - For each x & y in this set, increment γ(x,y)
  - $O(Vd^2)$: Avoid $O(V^2)$ initialization by dynamic creation & initialization
- Sparse Matrix Equivalency: A = adjacency matrix
  - $\gamma = AA^T$; O(nnz(A))

Jaccard via MapReduce

- 1 MapReduce step: 3 phases
  - **Map** some function over all data to (key,value) stream
  - **Group** pairs by key
  - **Reduce** each group
- Two reported Jaccard implementations
  - 3-steps with V = 1 million edges on 50 node cluster
  - 5-steps with V = 64 million vertices & 1 billion edges
    - Burkhardt "Asking Hard Graph Questions," Beyond Watson Workshop, Feb. 2014.
    - RMAT matrices, average d(i) = 16
    - 1000 node system, each with 12 cores & 64GB

Measurements & Trends





JACS (Jaccard Coefficients / Sec) = $1.6E6*V^{0.26}$

Relevant Batch Variations

- Consider graphs with 2 vertex classes
  - A = set of "people", B = set of "addresses"
  - Edges from A to B is person a "resided at" address b
  - $\gamma(a,b)$ = # of common addresses between a and b
- Real world: for |A| = 8E8, |B| = 1E8,
  - $|\gamma| = 1.2E12 = |A|^{1.35}$
- Variations:
  - >2 classes of vertices
  - Exclude paths thru high in-degree neighbors
  - Weight paths on basis of properties (e.g. same last name)
  - Add threshold

Dynamic Variations

- Many applications with dynamic graphs
  - Vertices, edges added/deleted dynamically
- Question: which γs change with edge addition or deletion
  - Perhaps just those that cross threshold, in either direction

Other Variations

- Don't compute/store all possibly $O(V^2)$ γs, But store just statistics of set of γs for each vertex
  - Number of non-zero, largest, average, etc.
  - Reduces storage to $O(V)$
  - Serve as starting point for more complex queries
- Expand "neighbors" beyond "1 hop"
  - Real commercial applications at "1.5" hops

A Possible Heuristic

- Goal: constant time elimination of γs that are zero of less than some threshold
- Build bit vector for each vertex that hashes vertex ids into bits
  - Bit i =1 if one of more neighbors fall into set I
- Estimate γ(u,v) by ANDing u & v's bit vectors
  - If results are 0, then γ = 0
- Can also estimate upper bound on γ(u,v)
  - Simple function of # of 1s in bit vectors
- If benchmark uses a threshold, this estimate can prevent computation when bound<threshold
- Based on ideas from SpGEMM package

Suggested Benchmarks

- Simple batch: compute all non-zeros
  - Perhaps use same RMAT as for BFS
  - Performance Metric JACS
- Multi-class benchmark
  - One metric: time to compute all γs as above
  - Optionally append Jaccard statistics as properties to each vertex of a class
- Real-time event detection
  - Start with precomputed γ statistics
  - Input stream of additions/deletions
  - Check for changes in γs
  - Performance metric: change throughput

Key Questions for More Precise Benchmark Definitions

- Formal definition of batch and incremental
- Where do non-zeros go? File, properties, …
- Tie data sets to real applications
- Explore real-world distribution of γs and Γs to understand growth rates better
- Build in options such as thresholds
- Develop complexity models, esp. parallel
- Verify correct solutions
- Develop reference implementations

Acknowledgements

- Funded in part by the Univ. of Notre Dame, Notre Dame, IN, USA