Georgia Tech School of Computational Science and Engineering

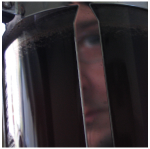
Graph Analysis Trends and Opportunities

Dr. Jason Riedy and Dr. David Bader

Georgia Institute of Technology

CMG Performance & Capacity, November 2014

Dr. Jason Riedy



- Research Scientist II, Computational Science and Engineering
- PhD UC Berkeley, 2010
- Major developer of STING, `community-el`, and other used graph analysis codes
- PI or co-PI on > 5 current funded graph analysis projects
- Primary author of the Graph500 specification
- Program Committees for HPC conferences including IPDPS, HiPC, ICPP
- 20+ referreed publications, dozens of cited technical reports, $\geq$ 350 citations, *etc.*
- Widely used code in packages like LAPACK, BLAS; contributions ranging from git to GNU R and Octave

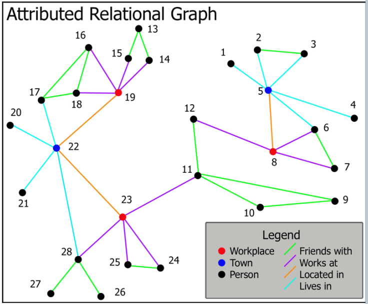
Outline

(insert prefix here)-scale data analysis

Cyber-security	Identify anomalies, malicious actors
Health care	Finding outbreaks, population epidemiology
Social networks	Advertising, searching, grouping
Intelligence	Decisions at scale, regulating algorithms
Systems biology	Understanding interactions, drug design
Power grid	Disruptions, conservation
Simulation	Discrete events, cracking meshes

- Graphs are a unifying motif for data analysis.
  - Changing and *dynamic* graphs are important!

Why Graphs?



- Smaller, more generalized than raw data.
- Taught (roughly) to all CS students...
- Semantic attributions can capture essential *relationships*.
- Traversals can be faster than filtering DB joins.
- Provide clear phrasing for queries about *relationships*.

Often next step after dense and sparse linear algebra.

Graphs: A Fundamental Abstraction

Structure for “unstructured” data

- Traditional uses:
  - Route planning on fixed routes
  - Logistic planning between sources, routes, destinations
- Increasing uses:
  - Computer security: Identify anomalies (e.g. spam, viruses, hacks) as they occur, insider threats, control access, localize malware
  - Data / intelligence integration: Find smaller, relevant subsets of massive, “unstructured” data piles
  - Recommender systems (industry): Given activities, automatically find other interesting data.

Application: Analyzing Twitter for Social Good

Massive Social Network Analysis: Mining Twitter for Social Good

David Ediger Karl Jiang
Jason Riedy David A. Bader
Georgia Institute of Technology
Atlanta, GA, USA

Courtney Corley Rob Farber
Pacific Northwest National Lab.
Richland, WA, USA

William N. Reynolds
Least Squares Software, Ir
Albuquerque, NM, USA

ICPP 2010

Abstract—Social networks produce an enormous quantity of data. Facebook consists of over 400 million active users sharing over 5 *billion* pieces of information each month. Analyzing this vast quantity of unstructured data presents challenges for software and hardware. We present GraphCT, a *Graph Characterization Toolkit* for massive graphs representing social network data. On a 128-processor Cray XMT, GraphCT estimates the betweenness centrality of an artificially generated (R-MAT) 537 million vertex, 8.6 billion edge graph in 55 minutes and a real-world graph (Kwak, *et al.*) with 61.6 million vertices and 1.47 billion edges in 105 minutes. We use GraphCT to analyze public data from Twitter, a microblogging network. Twitter's message connections appear primarily tree-structured as a news dissemination system. Within the

involves over 400 million active users with an av
120 'friendship' connections each and sharing 5
references to items each month [11].

One analysis approach treats the interactions as
and applies tools from graph theory, social r
analysis, and scale-free networks [29]. Howev
volume of data that must be processed to appl
techniques overwhelms current computational cap
Even well-understood analytic methodologies
advances in both hardware and software to proc
growing corpus of social media.

Social media provides staggering amounts

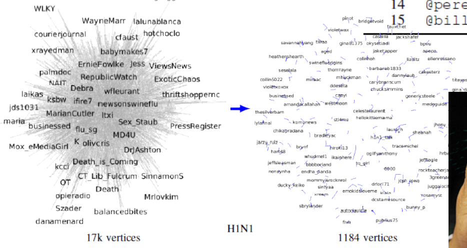


Fig. 3. Subcommunity filtering on Twitter data sets

TOP 15 USERS BY BETWEENNESS CENTRALITY

Rank	Data Set H1N1	Data Set atlflood
1	@CDCFlu	@ajc
2	@addthis	@driveafaste
3	@Official_PAX	@ATLCheap
4	@FluGov	@TWCi
5	@nytimes	@HelloNorthGA
6	@tweetmeme	@11AliveNews
7	@mercola	@WSB_TV
8	@CNN	@shaunking
9	@backstreetboys	@Carl
10	@EllieSmith_x	@SpaceyG
11	@TIME	@ATLINtownPa
12	@CDCemergency	@TJsDJs
13	@CDC_eHealth	@ATLien
14	@perezhilton	@MarshallRamsey
15	@billmaher	@Kanye

twitter

public tweets

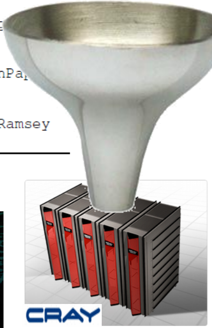


Image credit: bioethicsinstitute.org

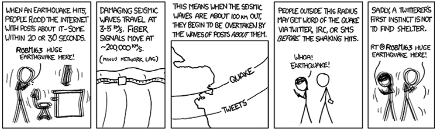
Pacific Northwest NATIONAL LABORATORY

Application: Social Network Analysis

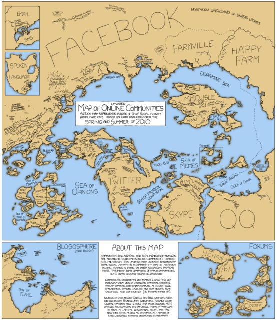
Problems

- Detecting “communities” automatically
- Identifying important individuals
- Given a few members, finding a joint community
- Finding *actual* anomalies

What techniques can scale to massive, noisy, *changing* populations?



`http://xkcd.com/723`



`http://xkcd.com/802`

And more applications...

- Cybersecurity
  - Determine if new packets are allowed or represent new threat in < 5ms...
  - Is the transfer a virus? Illicit?
- Credit fraud forensics ⇒ detection ⇒ monitoring
  - Integrate all the customer's data
  - Becoming closer to real-time, massive scale
- Bioinformatics
  - Construct gene sequences, analyze protein interactions, map brain interactions
  - Amount of *new* data arriving is growing massively
- Power network planning, monitoring, and re-routing
  - Already nation-scale problem
  - As more power sources come online (rooftop solar)...

No shortage of data...

Existing (some out-of-date) data volumes

NYSE	1.5 TB generated daily into a maintained 8 PB archive
Google	"Several dozen" 1PB data sets (CACM, Jan 2010)
LHC	15 PB per year (avg. 21 TB daily)
Wal-Mart	536 TB, 1B entries daily (2006)
EBay	2 PB traditional DB, and 6.5PB streaming, 17 trillion records, 1.5B records/day, web click = 50–150 details. (2009)
Facebook	> 1B monthly users...

- All data is *rich* and *semantic* (**graphs!**) and changing.
- Base data rates include items and not *relationships*.

Data velocities

Data volumes

NYSE	>1.5TB daily
LHC	>41TB daily
NG seq.	150GB per machine daily
Facebook	Who knows?

Data transfer

- 1 Gb Ethernet: 8.7TB daily at 100%, 5-6TB daily realistic
- PB disk rack, parallel 10GE: 1.7PB daily streaming read/write
- CPU ⟷ Memory: QPI,HT: 5+PB/day@100%

Data growth

- Facebook: $> 2\times$/yr
- Twitter: $> 10\times$/yr
- Growing sources: Health, sensors, security

Speed growth

- Ethernet/IB/*etc.*: 4× in next 2 years?
- Memory: Slow growth, possible bump?
- Direct storage: flash, then what?

Streaming graph data

Data Rates

Networks:

- Gigabit ethernet: 81k – 1.5M packets per second
- Over 130 000 flows per second on 10 GigE

Person-level, from `www.statisticsbrain.com`:

- 58M posts per day on Twitter (671 / sec)
- 1M links shared per 20 minutes on Facebook

Opportunities

- Often analyze only changes, not *entire* graph
- Throughput & latency: Different levels of concurrency

Methods

General approaches: Static and Streaming

Different approaches

- High-performance *static graph analysis*
  - Techniques apply to unchanging massive graphs
  - Provides useful after-the-fact information, starting points.
  - Serves many existing applications well: market research, much bio- & health-informatics...
  - Massive-scale algorithms need to be $O(|E|)$ or approximated down to it.
- High-performance streaming graph analysis
  - Focus: smaller dynamic changes within massive graphs
    - Streaming data, not CS-style streaming algorithms
  - Find trends or new information as they appear.
  - Serves upcoming applications: fault or threat detection, trend analysis, online prediction...
  - Can be $O(|\Delta E|)$? $O(\mathrm{Vol}(\Delta V))$?
    - Less data $\Rightarrow$ faster, more efficient, lower latency

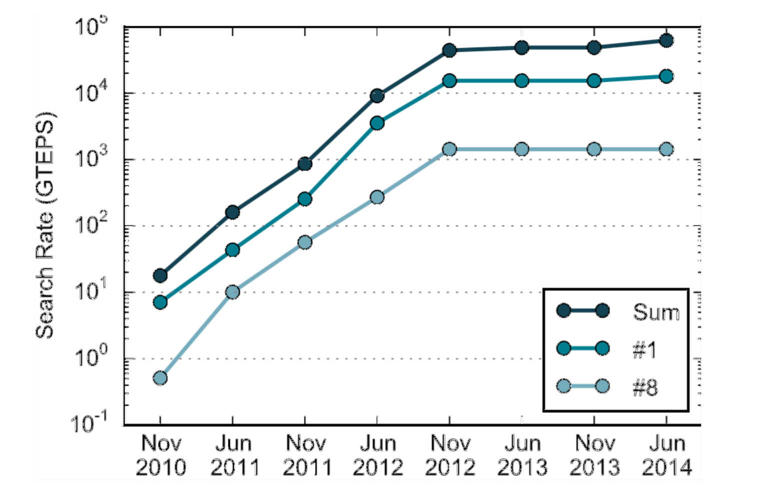
Breadth-First Search

The problem...

Build a tree from a starting vertex by repeatedly visiting all immediate, unvisited neighbors. At each traversal, record a parent. Repeat until there are no unvisited neighbors.

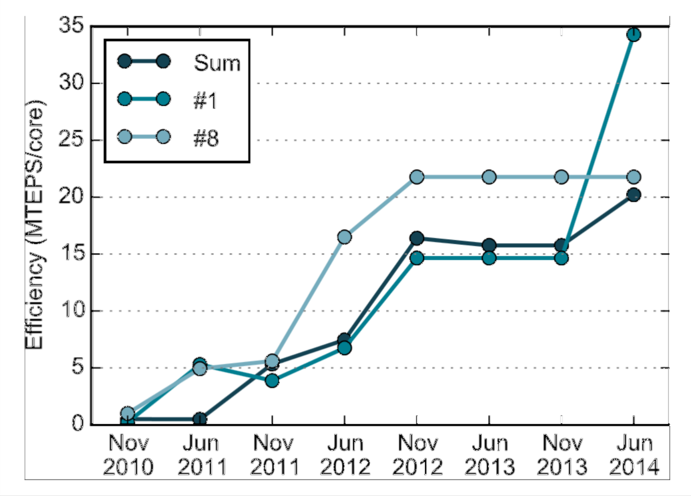
- $O(|V| + |E|)$, but problem-dependent parallel performace
- Core of *many* scalable, parallel graph algorithms
- *Non-deterministic* when any parent works
- Base of the Graph500 benchmark, “fastest traversal”
- Isn’t it done yet? Nope.

Graph500 Performance History



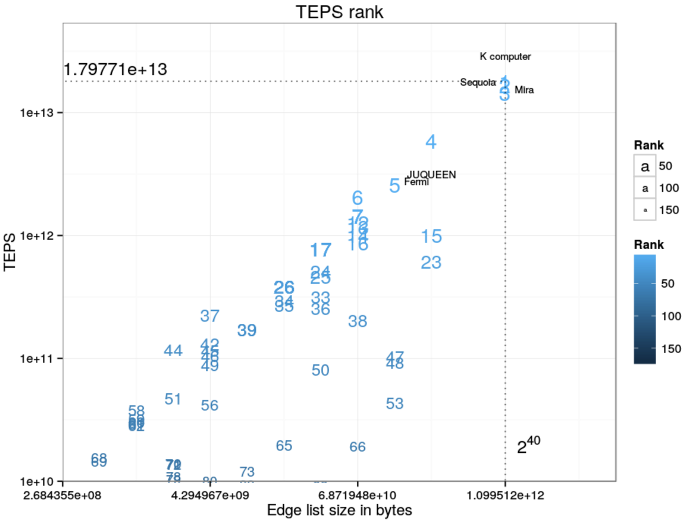
Plot courtesy of Scott Beamer, UC Berkeley

Graph500 Perf/Cores History



Plot courtesy of Scott Beamer, UC Berkeley

Graph500 Perf v. Size, Summer 2014



Streaming Queries

Different kinds of questions

- How are individual graph metrics (*e.g.* clustering coefficients) changing?
- What are the patterns in the changes?
  - Are there seasonal variations?
  - What are responses to events?
- What are *temporal anomalies* in the graph?
  - Do key members in clusters / communities change?
  - Are there indicators of event responses before they are obvious?

New kinds of queries, new challenges...

Performance on Streaming Graphs

Work at Georgia Tech

- Triangle counting / clustering coefficients
  - Up to 130k graph updates per second on X5570 (Nehalem-EP, 2.93GHz)
- Connected components & spanning forest
  - Over 88k graph updates per second on X5570
- Community detection & maintenance
  - Up to 100 million updates per second, 4-socket 40-core Westmere-EX
  - *(Note: Most updates do not change communities...)*
- Incremental PageRank
  - Reduce lower latency by $> 2\times$ over restarting
- Betweenness centrality
  - $O(|V| \cdot (|V| + |E|))$, can be sampled
  - Speed-ups of $40\times$–$150\times$ over static recomputation

Algorithmic Disruptions

- **Current:** Computing on the data *as it arrives*, not recomputing over all data.
  - Faster, lower latency, lower power...
- New algorithms for old problems.
  - Many practical parallel graph algorithms are not "work-efficient."
  - New work is finding work-efficient *and* practical methods: Connected components (CMU: Shun, Dhulipala, Blelloch, SPAA 2014), betweenness centrality (McLaughlin and Bader, SC14)
- Approximations and coping with errors
  - There is very little approximation theory for graph algorithms.
  - Not sure which metrics are sensitive to sampling, errors... (Zakrzewska and Bader, PPAM 2013)

Tools

Tools

Rough Categories

Graph DB	Neo4j, Sparksee (was DEX), AllegroGraph, Sesame, Titan, Flock...
Clusters/Cloud	GraphX, Pregel, giraph, pegasus...
“Capability”	igraph, networkX
HPC	KDT / GraphBLAS, GraphLab, NetworKIT, GraphCT
Streaming	GT STINGER

Graph Databases

Pros

- Incredibly flexible data models
- Large ecosystem:
  - query and viz tools
  - data management tools

Cons

- Standard query languages do not support most algorithms.
- The flexibility costs performance. Analysis algorithms run $10\times - 100\times$ slower than more specific analysis tools, at least. ("A Performance Evaluation of Open Source Graph Databases," R. McColl, *et al.*, 2014)

Cluster/Cloud Tools

Pros

- Growing ecosystem, large buzz
- Simple to write simple analyses.
- Often the only systems that handle hardware failure!

Cons

- Performance can be comparable to graph databases...
- Often incredibly difficult to write more complex algorithms
- Clusters are expensive compared to single-node.
  - Many more power supplies
  - Wasted memory on OSes

“Capability” Tools

Pros

- Stockpiles of algorithms
- Available for many interactive environments (*e.g.* R)
- Good solution for exploring analysis of small data sets

Cons

- Rarely ever parallel
- Often cannot scale to large problems

HPC Tools

Pros

- HPC: Fast. Really fast. Often fastest.
- Scale to large problems
- Exist for traditional HPC boxes, “cloud” allocations, *etc*.
  - Also for large-memory servers!

Cons

- Distributed-memory versions use very focused models for performance
  - GraphBLAS: Sparse matrix - sparse vector product
  - GraphLab: Vertex programs
- If your problem does not fit the model...
- Algorithms still being developed

Streaming Tools

Pros

- Great fit for streaming problems!
- Astounding speed-ups over static re-analysis. Speed-up grows with problem size.
- Can target high throughput or low latency.

Cons

- There really aren't many tools... (STINGER at GT)
- Terminology is very much in flux...
- Algorithms are still being designed...

Software Disruptions

- New algorithms are being developed, tuning can be astronomically hard.
  - "Work-efficient" is not always fastest, need sampling and run-time algorithm selection (McLaughlin and Bader, SC 2014)
- **Combinations:** Let each tool do what it does well.
  - Cloud/cluster: Fantastic for data extraction
  - HPC tools: Fantastic for analysis
  - Combination: Kang and Bader, MTAAP 2010, reduce analysis time by five orders of magnitude.
  - Cloud extraction → streaming processing: Demonstrated with STINGER at Research@Intel 2013, GraphLab Workshop 2013

Hardware

Architecture Requirements for Efficiency

The issues

- Runtime is dominated by latency
  - "Random" accesses to global graph and data storage
  - Can hot-spot: Many accesses to the same place
- Essentially no computation to hide the latency
- Access pattern is problem dependent
  - Prefetching can hinder performance
  - Often only want a small portion of data
- Most parts suffer from abysmal locality in memory
- Cannot require a nuclear reactor.

Architecture Requirements for Efficiency

Some desires

- Large memory capacity
- Low latency, high bandwidth, high injection rate
  - For very small messages!
- Latency tolerance (threading...)
- Light-weight, localized synchronization
- Global address space
  - Partitioning is nigh impossible
  - Ghost nodes everywhere
  - Algorithms are difficult enough to implement

Existing Platforms

- Distributed memory / cluster
  - Cloud-ish: Slow network, massive storage
  - HPC-ish: Fast network, less storage
- Shared memory
  - Single motherboard: Ultra-fast network, little storage
  - Many motherboards: Tricky...
- Accelerators: Tiny memory, incredible bandwidth

Now start combining the platforms...

Mapping Problems to Platforms

- Distributed memory / cluster
  - Cloud-ish: Fantastic for massive storage, extraction
  - HPC-ish: Great for known, forensic analysis on extracted graph
  - All of them eat power.
- Shared memory
  - Highly-threaded, single node: Focused analysis, streaming
  - Highly-threaded, multi-node: Often hard to extract *enough* parallelism (Cray XMT / URiKA)
  - Multi-node virtual shared memory: Re-eval in progress
  - Single node often eats less power, but...
- Accelerators
  - Very, *very* focused analysis
  - Can be very energy-efficient (McLaughlin, Riedy, Bader, HPEC 2014)

Distruptive Platform Changes

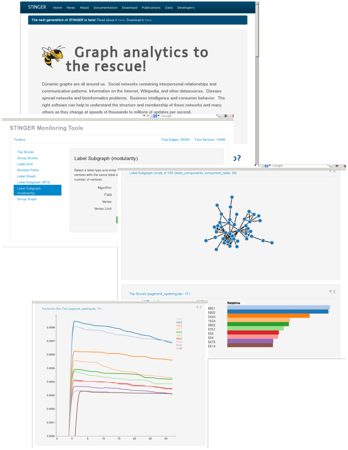
- In next 3–5 years, memory is going to change.
  - 3D stacked memory (IBM, NVIDIA)
  - Hybrid memory cube (HMC Cons., Micron, Intel)
  - Programming logic layer on-chip
  - Possibly non-volatile
  - Order of magnitude higher bandwidth
  - Order of magnitude lower energy cost
- This is happening. You can obtain HMC-FPGA combinations for testing.
- Interconnects are changing.
  - Processor $\Leftrightarrow$ memory $\Leftrightarrow$ accelerator (NVLink, Phi)
  - Data-center networks finally may change, not just *n*GbE

Summary and Opportunities

We live in interesting times.

- Graph analysis tools, platforms are developing rapidly.
  - Only just starting to combine platforms and map problems appropriately.
- Performance is developing rapidly.
  - New algorithms, improved implementations, better platform choices
  - New approaches like streaming and approximation
- Even bigger changes are coming.
  - Can you imagine a PB of non-volatile storage at nearly RAM speed and latency?

STINGER: Where do you get it?



www.cc.gatech.edu/stinger/

Gateway to

- code,
- development,
- documentation,
- presentations...

Remember: Still academic code, but maturing.

Users / contributors / questioners: Georgia Tech, PNNL, CMU, Berkeley, Intel, Cray, NVIDIA, IBM, Federal Government, Ionic Security, Citi

Acknowledgment of support

DEPARTMENT OF DEFENSE UNITED STATES OF AMERICA
DEPARTMENT OF ENERGY UNITED STATES OF AMERICA
NASA
NATIONAL INSTITUTES OF HEALTH
DARPA
OAK RIDGE National Laboratory
Pacific Northwest NATIONAL LABORATORY
Sandia National Laboratories
IBM
intel
NVIDIA
NORTHROP GRUMMAN
LexisNexis
SONY
Microsoft Research
CRAY
Sun microsystems
XILINX
TOSHIBA