Georgia Tech School of Computational Science and Engineering

STING: Spatio-Temporal Interaction Networks and Graphs for Intel Platforms

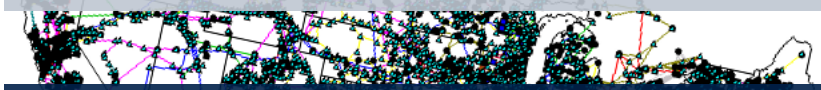
David Bader, Jason Riedy, David Ediger, Rob McColl, Timothy G. Mattson*, …

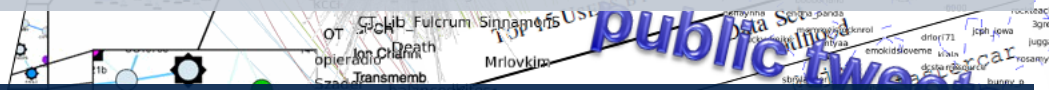
Georgia Institute of Technology

17 January 2014

(insert current prefix here)scale data analysis

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



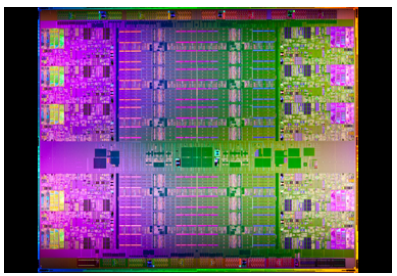


- Data analysis runs throughout.
- Many problems can be phrased through *graphs*.
  - Any many are changing and *dynamic*.

GT's part of Intel's Non-Numeric Computing Program

Goal

Supporting analysis of massive graphs under rapid change across the spectrum of Intel-based platforms.



STING on Intel-based platforms

- Maintain a graph and metrics under large data changes
  - Performance, documentation, distribution
  - Available at `http://www.cc.gatech.edu/stinger/`

Outline

Motivation: Graph Algorithms for Analysis

STING

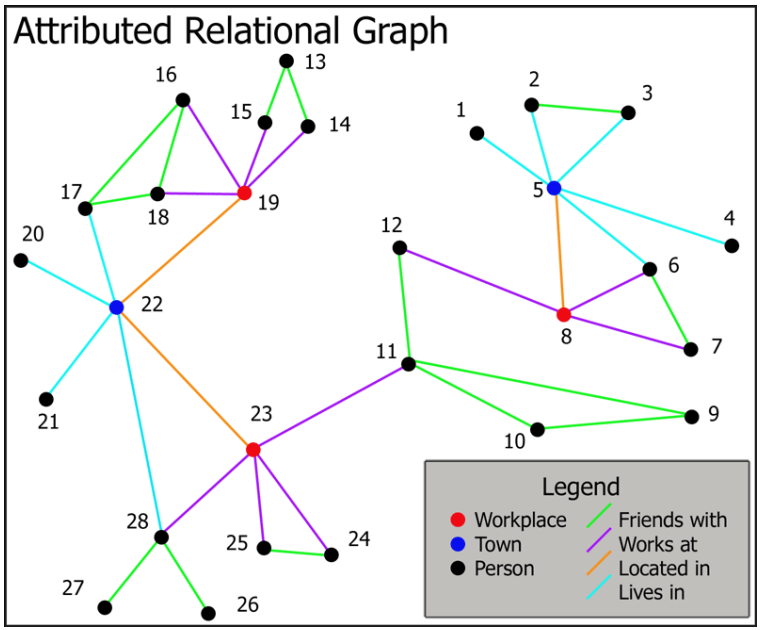
Review of Accomplishments

Selected Details on Recent Work
- Dynamic Community Updating
- Incremental PageRank

Related Projects (Future Plans)

Artifacts of the Intel Non-Numeric Computing Program

Why Graphs?



- Smaller 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.

- *Note: Extracting data into a graph is a separate topic, see Intel's GraphBuilder.*

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.

On to STING...

Motivation: Graph Algorithms for Analysis

STING

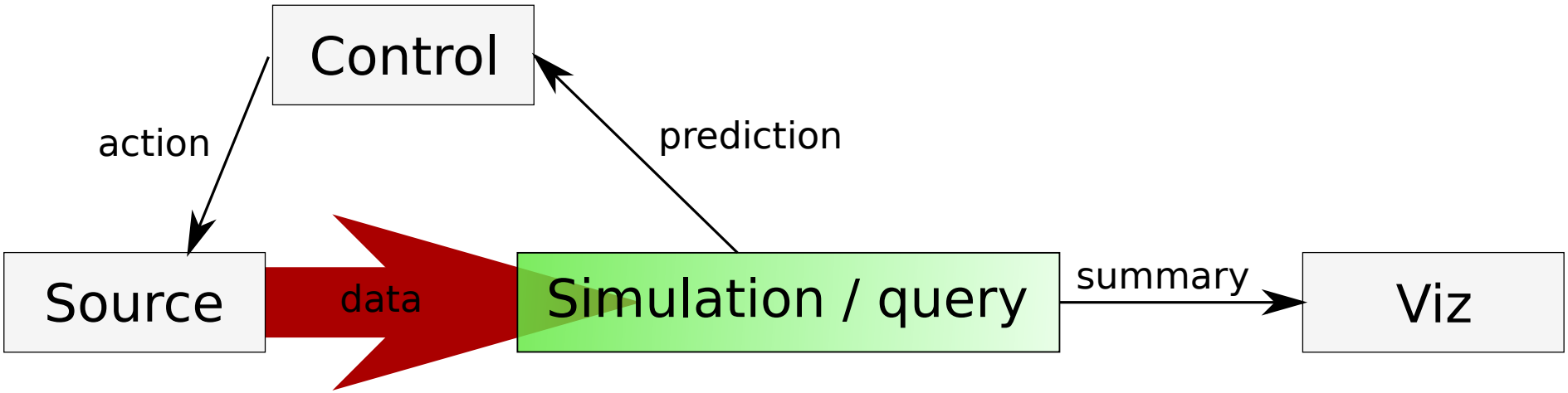
Review of Accomplishments

Selected Details on Recent Work

Related Projects (Future Plans)

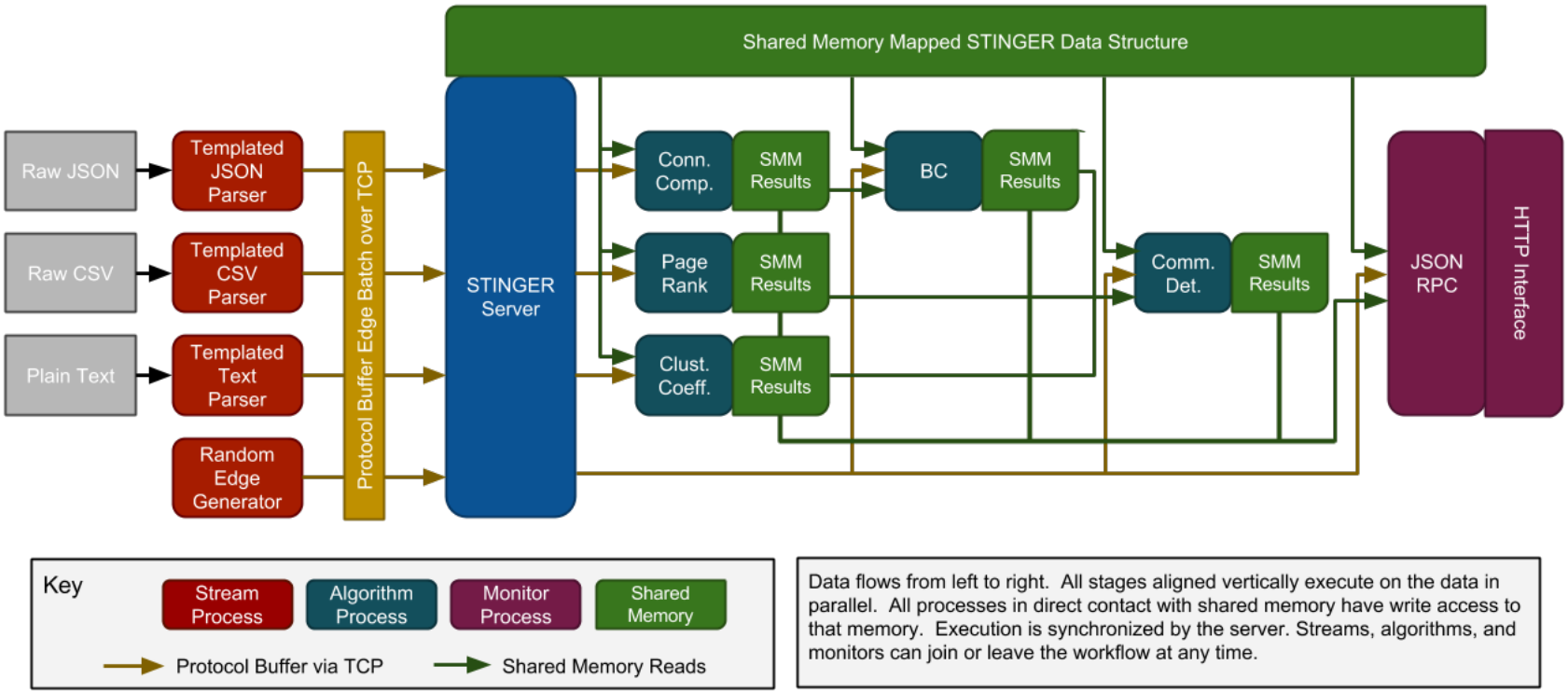
Artifacts of the Intel Non-Numeric Computing Program

STING's focus



- STING manages queries against changing graph data.
  - Visualization and control often are application specific.
- Ideal: Maintain many persistent graph analysis kernels.
  - One current graph snapshot, kernels keep smaller histories.
  - Also (a harder goal), coordinate the kernels' cooperation.

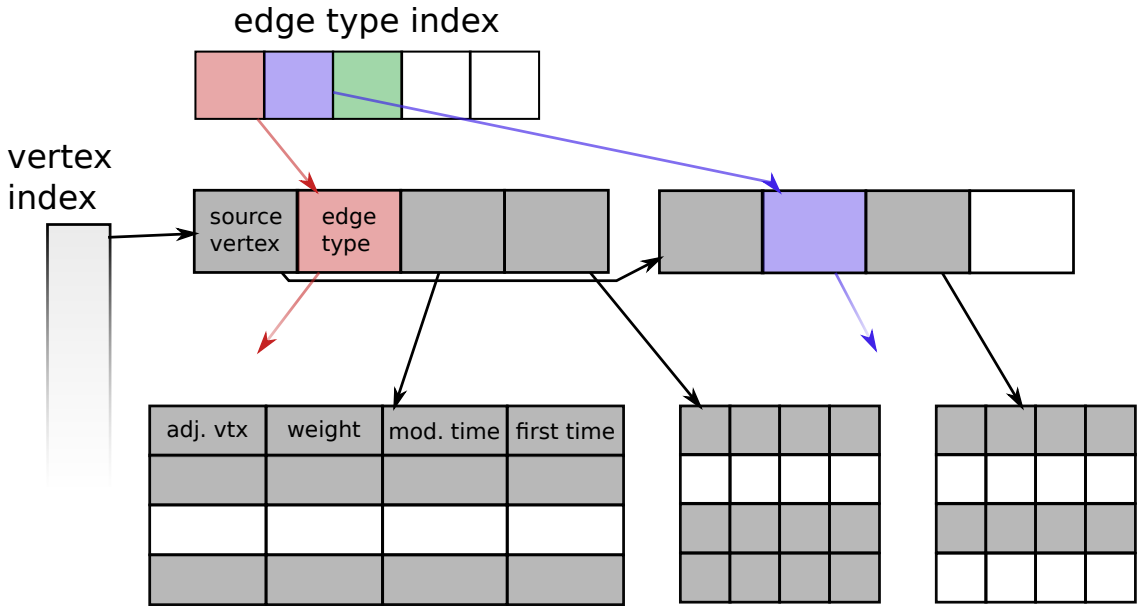
STING: High-level architecture



Slide credit: Rob McColl and David Ediger

- OpenMP + sufficiently POSIX-ish
- Multiple processes for resilience

STING Extensible Representation: Core data structure



Initial considerations [Bader, *et al.*]

- Be useful for the entire “large graph” community
- Portable semantics and high-level optimizations across multiple platforms & frameworks (XMT C, MTGL, etc.)
- Permit good performance: No single structure is optimal for all.
- Assume globally addressable memory access and atomic operations
- Support multiple, parallel readers and a single writer process
  - Large graph ⇒ rare conflicts, may ignore synchronization

STING in use

Who is using it?

- Internally, well, us. Some details later.
- Outside institutions in multiple research projects
  - Center for Adaptive Supercomputing Software – Multithreaded Architectures (CASS-MT) directed by PNNL
  - PRODIGAL team for DARPA ADAMS (Anomaly Detection at Multiple Scales) including GTRI, SAIC, Oregon State U., U. Mass., and CMU
- Several industrial companies in the area of data analytics and security
- Well-attended tutorials given at PPoPP, in MD, and locally

STING: Where do you get it?

www.cc.gatech.edu/stinger/
Gateway to

- code,
- development,
- documentation,
- presentations...
- (working on usage and development screencasts)

Remember: Still academic code, but maturing.

Selected Program Accomplishments I

- In the beginning: Initial streaming algorithms
  - Maintaining clustering coefficients[1] (counting triangles)
    - Up to 130 000 graph updates per second on X5570 (Nehalem-EP, 2.93GHz)
    - 2000× speed-up over static recomputation
  - Connected components[2]
    - Up to 88 700 graph updates per second on X5570, 233× speed-up
- Community detection[3, 4, 5]
  - First parallel algorithm for agglomerative community detection, 3.3B edges clustered in 505s on a 4-socket, 10-core Westmere-EX.
  - Win 10th DIMACS Implementation Challenge's Mix Challenge
  - Enabled moving computation from special server + batch to laptops and servers.

Selected Program Accomplishments II

- Still maintained, used by external groups like Cray, people finally threatening performance.

- Expanded streaming algorithms
  - Improved connected components[6]
    - Does *not* fall back on re-computation for deletions
    - Up to 137× speed-up over static re-computation
  - Community maintenance[7] and monitoring*
    - Up to 100 million updates per second, speed-ups from 4× to 3 700×
    - Demonstrated at Research@Intel and GraphLab Workshop 2013
  - PageRank*
    - Reduce edge traversals from 30% – 90%.
  - Betweenness centrality* [8, 9]
    - Up to 148× speed-up
- Optimization and software structure of STING[10]

Selected Program Accomplishments III

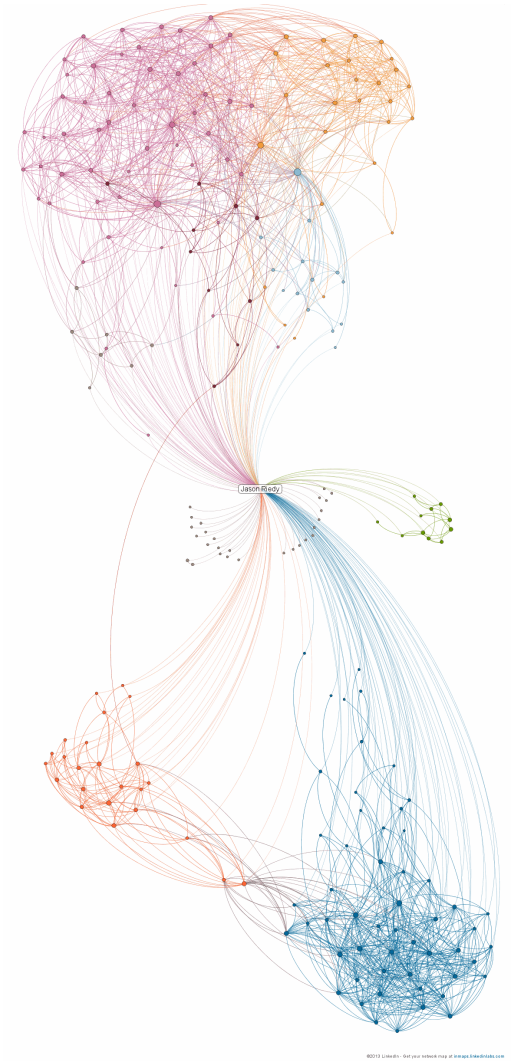
- Up to 14× performance improvement on Intel-based platforms[11], *best paper award* at HPEC12
- Research@Intel 2013 demo collaborative with Intel GraphBuilder: extract and transform data from a Hadoop store, transfer to STING for analysis

- Something different:
  - 10^{th} DIMACS Implementation Challenge on Graph Partitioning and Graph Clustering [12]
    - Very useful workshop, 13-14 February 2012 at Georgia Tech
    - Intel sponsorship: Thank you!
    - Graph archive being used in other projects (although not always cited appropriately)
  - PASQUAL [13]
    - First *de novo* genome assembler that is both scalable and reliable (doesn't crash).
    - `http://www.cc.gatech.edu/pasqual/` (not STING-based)

Community Detection

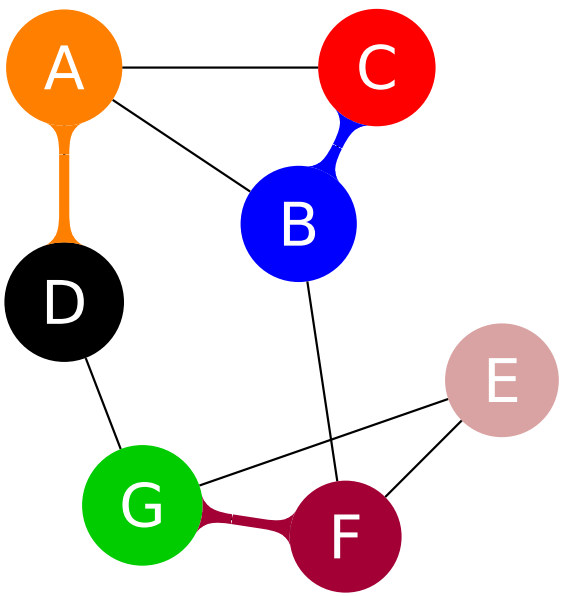
What do *we* mean?

- Partition a graph's vertices into disjoint communities.
- Locally optimize some metric, *e.g.* modularity, conductance, ...
- Try to capture that vertices are *more similar* within one community than between communities.



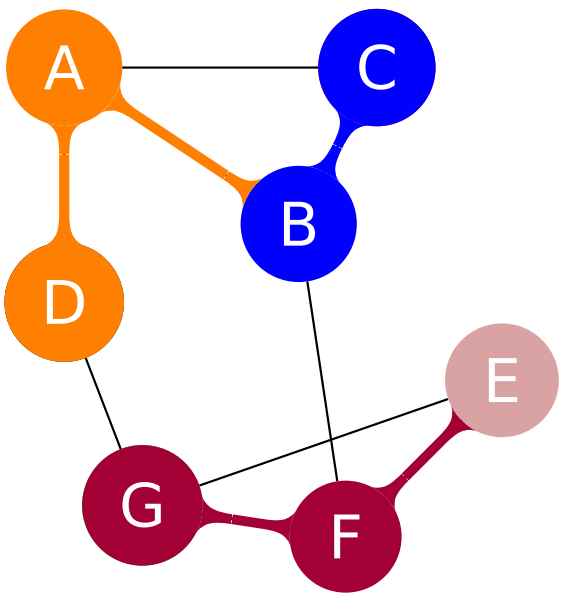
Jason's network via LinkedIn Labs

Parallel agglomerative method



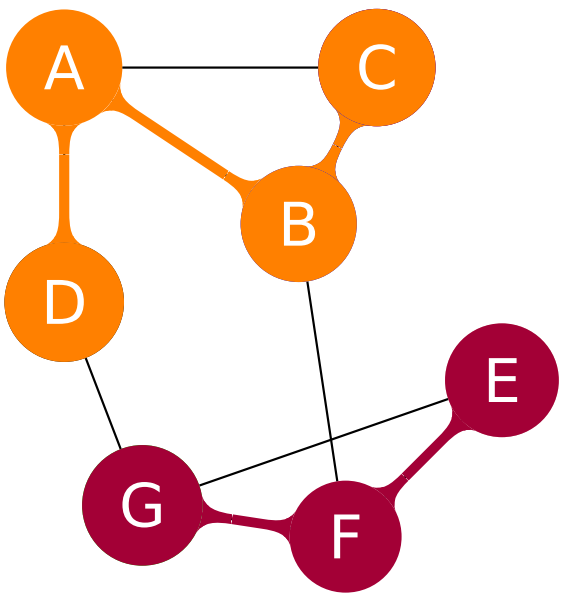
- Use a matching to avoid a serializing queue.
- Compute a heavy weight, large matching.
  - Simple greedy algorithm.
  - Maximal matching, within 2× max weight.
- Merge all matched communities in parallel.
- Highly scalable, $5.8 \times 10^6 - 7 \times 10^6$ edges/sec, 8× – 16× speed-up on 80-thread E7-8870
- Agnostic to weighting, matching…
  - Works with many *weighting* and termination criteria.

Parallel agglomerative method



- Use a matching to avoid a serializing queue.
- Compute a heavy weight, large matching.
  - Simple greedy algorithm.
  - Maximal matching, within 2× max weight.
- Merge all matched communities in parallel.
- Highly scalable, $5.8 \times 10^6 - 7 \times 10^6$ edges/sec, 8× – 16× speed-up on 80-thread E7-8870
- Agnostic to weighting, matching…
  - Works with many *weighting* and termination criteria.

Parallel agglomerative method



- Use a matching to avoid a serializing queue.
- Compute a heavy weight, large matching.
  - Simple greedy algorithm.
  - Maximal matching, within 2× max weight.
- Merge all matched communities in parallel.
- Highly scalable, $5.8 \times 10^6 - 7 \times 10^6$ edges/sec, 8× – 16× speed-up on 80-thread E7-8870
- Agnostic to weighting, matching…
  - Works with many *weighting* and termination criteria.

Community Monitoring Algorithm

Given a batch of edge insertions and removals:

1. Collect the set of possibly moved vertices ΔV.
   - Add the endpoints* of every edge insertion and removal to ΔV.
   - *(STING support routine, scatter/gather and atomic CAS.)*
2. Extract each $v \in \Delta V$ from its community into a singleton.
   - This is the complex piece, see Riedy & Bader, MTAAP13 [7].
3. Re-start agglomeration from the expanded community graph.

*: More recent experiments (by Anita Zakrzewska) add larger neighborhoods. Little effect on modularity, large effect on performance.

Platform: Intel® E7-4820-based server

Tolerates some latency by hyperthreading:

- "Westmere-EX:" 2 threads / core, 8 cores / socket, four sockets.
  - Fast cores (2.0 GHz), fast memory (1 066 MHz).
  - Not so many outstanding memory requests (60/socket), but large caches (18 MiB L3 per socket).
- Good system support
  - Transparent hugepages reduces TLB costs.
  - Fast, user-level locking. (HLE would be better...)
  - OpenMP, although I didn't tune it...
- Four processors (64 threads), 1 TiB memory
- gcc 4.7.2, Linux kernel 3.9 pre-release
- Acknowledgment: Donated by Oracle.

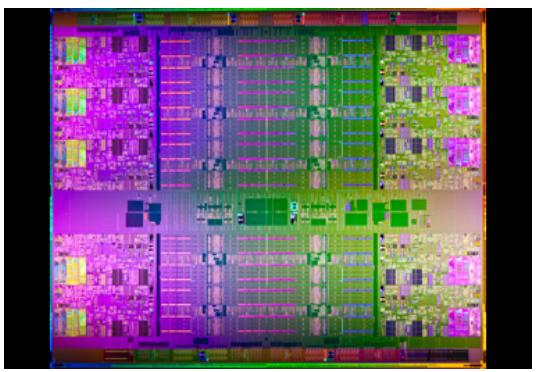


Image: Intel® press kit

Platform: Cray/YarcData XMT2

Tolerates latency by massive multithreading:

- Hardware: 128 threads per processor
  - Context switch on every cycle (500 MHz)
  - Many outstanding memory requests (180/proc)
  - “No” caches...
- Flexibly supports dynamic load balancing
  - Globally hashed address space, no data cache
- Support for fine-grained, word-level synchronization
  - Full/empty bit on with every memory word

- 64 processor XMT2 at CSCS, the Swiss National Supercomputer Centre.
- 500 MHz processors, 8192 threads, 2 TiB of shared memory

Image: cray.com

Test Data

- Combination of real and artificial
- Start with (small) real-world graphs from 10th DIMACS Challenge collection:

Name	$\|V\|$	$\|E\|$	$\|C\|$	$\|E_C\|$
caidaRouterLevel	192 244	1 218 132	18 343	30 776
coPapersDBLP	540 486	30 866 466	1 401	205 856
eu-2005	862 664	16 138 468	55 624	194 971

- Generate artificial edge actions:
  - Compute an initial clustering.
  - Generate random edge insertions between and removals within communities.
  - 15/16: Insertions. Select communities $\propto$ volume, vertices $\propto$ 1/degree.
  - 1/16: Removals. Uniform sampling (w/o replacement) of existing edges, then inserted edges.

Peak updates per second

- There is speed-up per # of threads, *etc*. on large batches, large graphs. Not the interesting story.

Graph	Platform	#T	Batch	Upd/Sec	Speed-up*	Latency(s)
caidaRouterLevel	IA32-64	56	1e5	1.2e+7	4.0	8.3e-3
	XMT2	56	1e6	2.5e+6	4.3	4.0e-1
coPapersDBLP	IA32-64	20	1e6	2.9e+6	11	3.5e-1
	XMT2	48	3e5	2.2e+6	21	1.4e-1
eu-2005	IA32-64	40	1e5	4.8e+6	330	2.1e-2
	XMT2	64	1e6	2.1e+6	43	4.9e-1

Large batches, many threads $\Rightarrow$ $> 10^6$ updates per second, but high latency

$*$: Speed up over static recomputation. Quality doesn't decrease significantly v. static recomputation.

Least latency between updates

Graph	Platform	#T	Batch	Upd./Sec	Speed-up*	Latency (s)
caidaRouterLevel	IA32-64	2	1	4.6e+2	430	2.2e-3
	XMT2	4	30	5.2e+3	230	5.8e-3
coPapersDBLP	IA32-64	4	30	7.0e+3	1900	4.3e-3
	XMT2	40	1	1.4e+2	380	7.1e-3
eu-2005	IA32-64	12	1	4.2e+2	3500	2.4e-3
	XMT2	20	10	1.5e+3	3200	6.5e-3

Small batches, fewer threads $\Rightarrow$ fast response, but $< 10^4$ updates per second

$*$: Speed up over static recomputation. Quality doesn't decrease significantly v. static recomputation.

Incremental PageRank

- PageRank: Well-understood method for ranking vertices based on random walks (related to minimizing conductance).
- Equivalent problem, solve $(I - \alpha A^T D^{-1})x = (1-\alpha)v$ given initial weights v.
- Goal: Use for *seed set expansion*, sparse v.
- State-of-the-art for updating x when the graph represented by A changes? Re-start iteration with the previous x.
- Can do significantly better for low-latency needs.
- Compute the change Δx instead of the entire new x.

Incremental PageRank: Iteration

Iterative solver

Step $k \rightarrow k+1$:

$$\Delta x^{(k+1)} = \alpha(A+\Delta A)^T(D+\Delta D)^{-1}\Delta x^{(k)} + \alpha\left[(A+\Delta A)^T(D+\Delta D)^{-1} - A^T D^{-1}\right]x$$

- Additive part: Non-zero only at changes.
- Operator: Pushes changes outward.

Incremental PageRank: Limiting Expansion

Iterative solver

Step $k \rightarrow k+1$:

$$\Delta\hat{x}^{(k+1)} = \alpha(A+\Delta A)^T(D+\Delta D)^{-1}\Delta\hat{x}^{(k)}_{|\text{ex}} + \alpha\Delta\hat{x}_{|\text{held}}$$
$$\alpha\left[(A+\Delta A)^T(D+\Delta D)^{-1} - A^TD^{-1}\right]\hat{x}$$

- Additive part: Non-zero only at changes.
- Operator: Pushes sufficiently large changes outward.

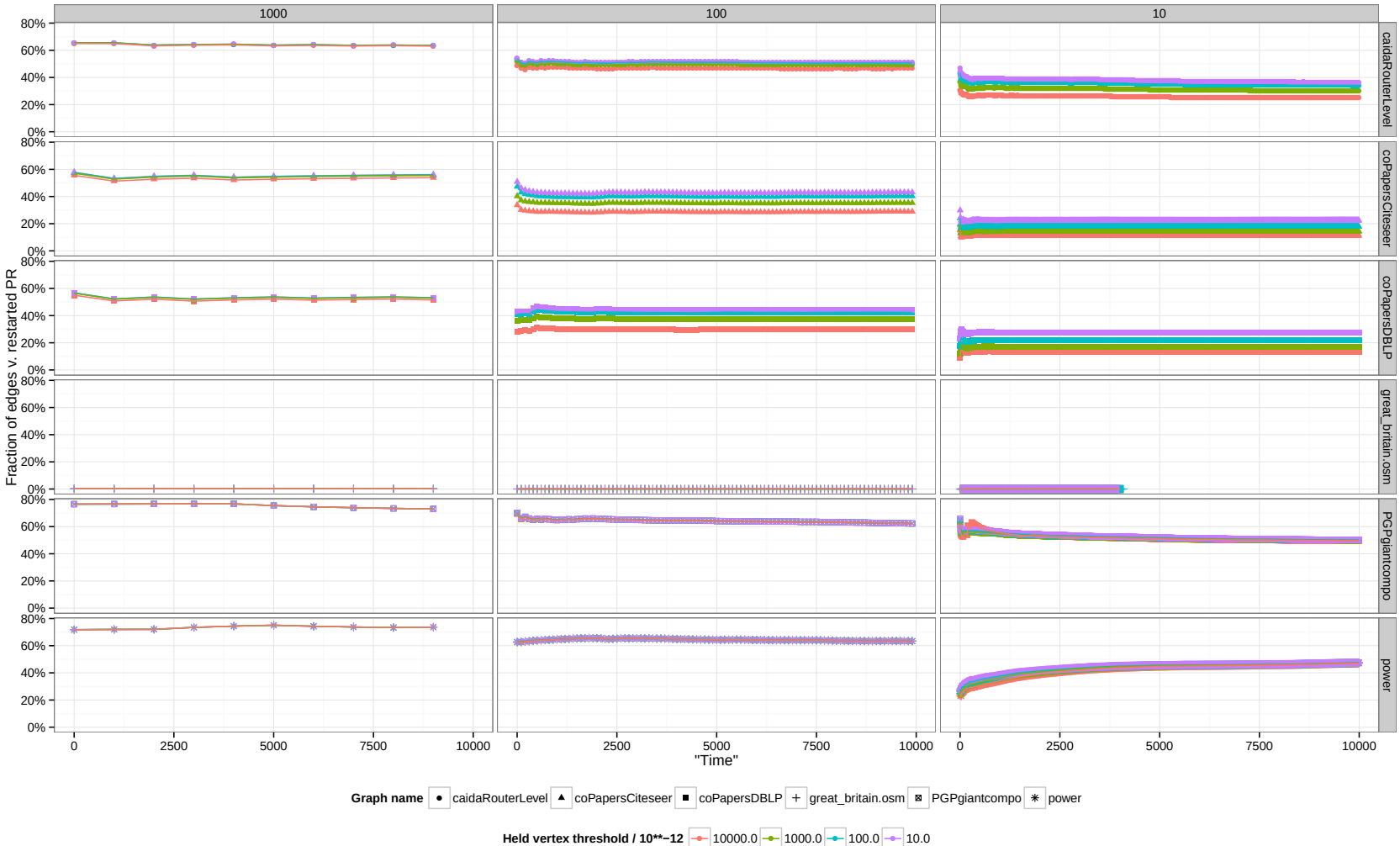
Incremental PageRank: Test Cases

- Initial, high-level implementation via sparse matrices in Julia.
- Test graphs from the 10^{th} DIMACS Implementation Challenge.
- Add uniformly random edges... worst case.
- Up to 100k in different batch sizes.
- One sequence of edge actions per graph shared across experiments.
- Conv. & hold base threshold: 10^{-12}

Graph	$\|V\|$	$\|E\|$	Avg. Deg.	Size (MiB)
caidaRouterLevel	192 244	609 066	3.17	5.38
coPapersCiteseer	434 102	1 603 6720	36.94	124.01
coPapersDBLP	540 486	15 245 729	28.21	118.38
great-britain.osm	7 733 822	8 156 517	1.05	91.73
PGPgiantcompo	10 680	24 316	2.28	0.23
power	4 941	6 594	1.33	0.07

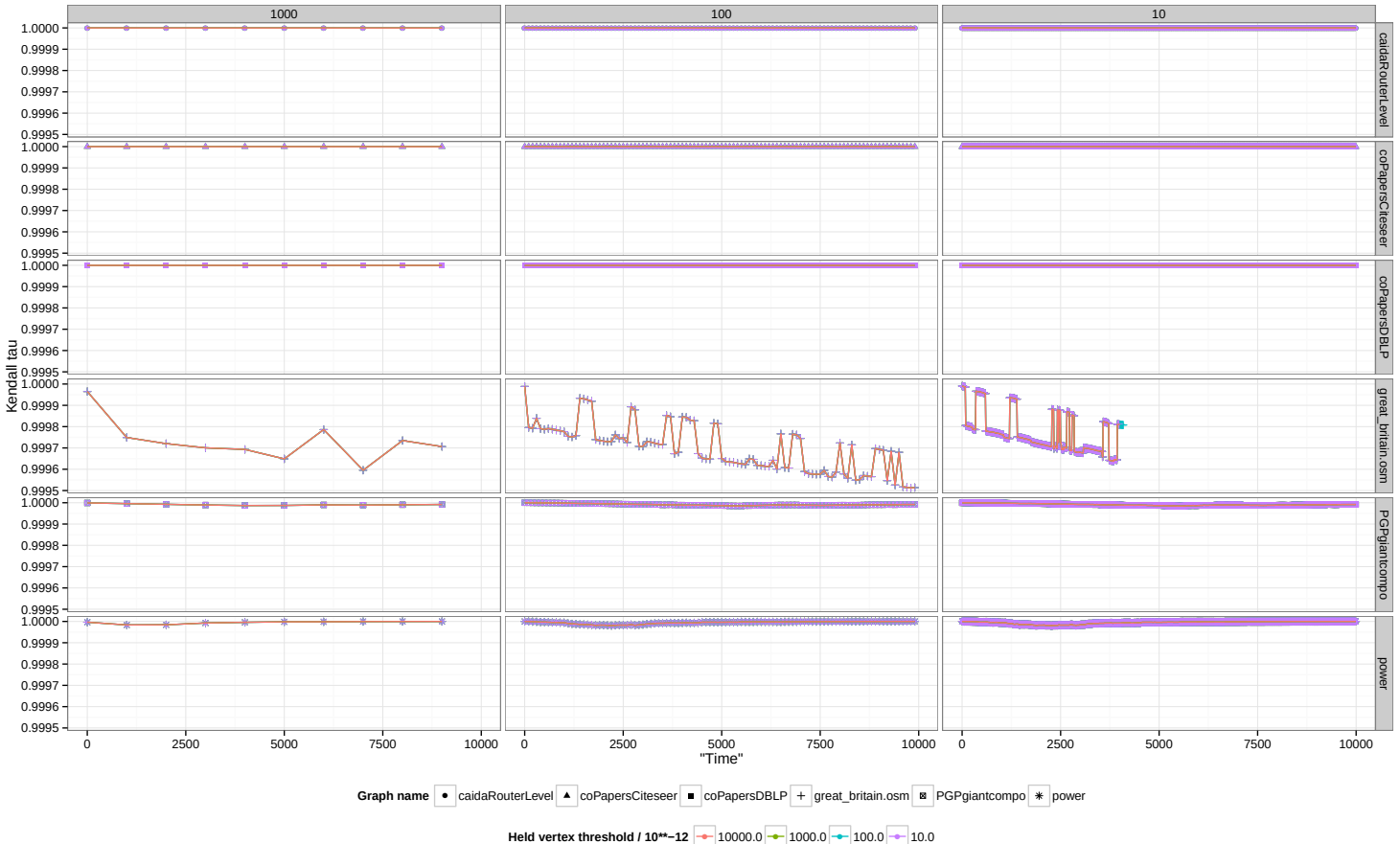
Incremental PageRank: Edge Traversal Savings

Percent of edge traversals relative to re-started iteration:



Incremental PageRank: Quality

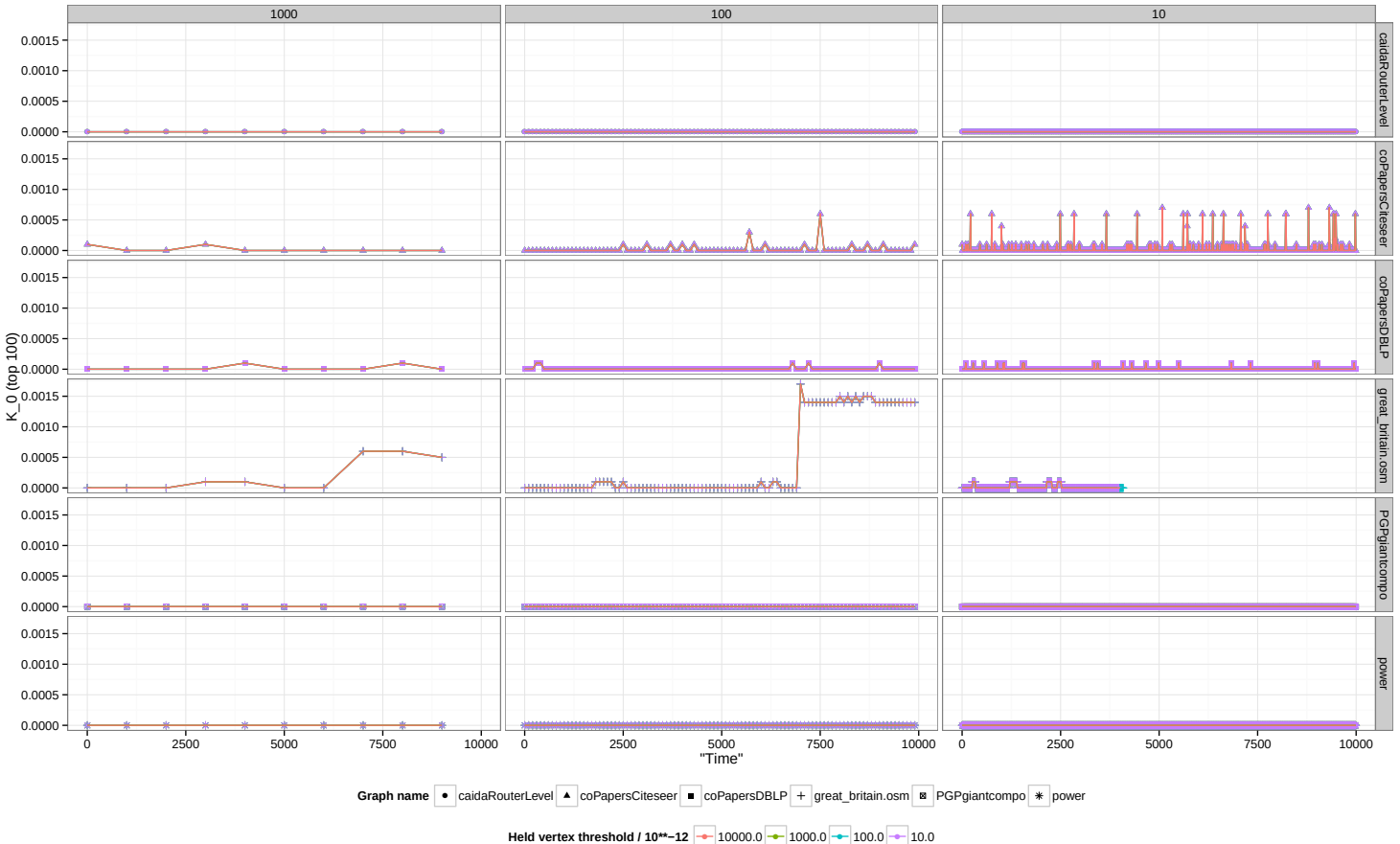
Kendall τ (count of inversions) for ranking result:



Larger is more similar, max. value one. All values here in (0.995, 1].

Incremental PageRank: Quality for Seed-Set Use

Fagin, *et al.*'s extension Kendall τ for the top 100 list:



Smaller is more similar, min. value one. All values here in $[0, 0.0018)$.

GRATEFUL: *Gr*aph *A*nalysis *T*ackling Power *Ef*ficiency, *U*ncertainty, and *L*ocality

David A. Bader (PI), Jason Riedy (co-PI)

Under DARPA PERFECT (Power Efficiency Revolution for Embedded Computing Technologies) #HR0011-13-2-0001

Goal: Improve ops/watt by factor of 50×–75×!

- Providing mission-critical analysis of semantic data where needed:
  - as close to the data and data use as possible
  - using the least power necessary to
  - enable rapid decisions with reliable analysis.
- GRATEFUL is developing
  - new algorithms providing new, fundamental capabilities
  - while coping with real-world uncertainties
  - under run-time power and environmental constraints.
- STING: Static ⇒ dynamic saves power, lowers latency.

The XScala Project: A Community Repository for Model-driven Design and Tuning of Data- Intensive Applications for Extreme-scale Accelerator-based Systems

David A. Bader (PI, GT), Viktor Prasanna (PI, USC),
Richard Vuduc (co-PI, GT), Jason Riedy (co-PI, GT)

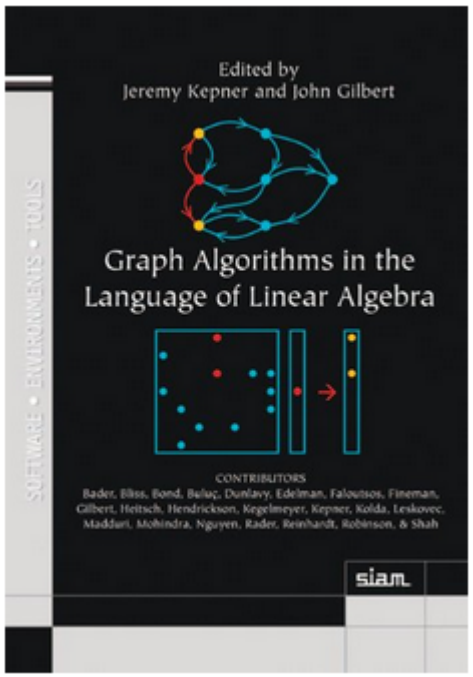
NSF ACI-1339745, an SI2-SSI project

Goal: Accelerate ***science***!

- Map irregular data problems onto "accelerators" **and** make them usable
  - *Science* accelerators: multi-threaded, multi-core; GPGPU; cloud; *etc*.
- Develop models and analysis kernels together.
- **Not just graphs**: also strings, machine learning, …
- **Not just GPUs**: also parallelizing sequential roadblocks, mapping onto FPGAs, finding opportunities for platforms, …

Graph Algorithm Primitives

Community standards effort underway: Timothy Mattson, Jeremy Kepner, John Gilbert, Aydin Buluç, …)



- State of the art for “Graphs in the language of Linear algebra” is mature enough to support a standard set of BLAS.
- Benefits of this standard include:
  - Separation of concerns spares graph algorithm experts the pain of producing optimized primitives (the vendors will produce these … just as with traditional BLAS)
  - Prevents mass scale “reinvention of the wheel” with each graph research group creating their own primitives.

Higher-Level (aka Fuzzier) Plans

- More integration into entire analysis tool chains.
  - Continue partnering with ETL researchers (*e.g.* GraphBuilder), visualization researchers, area experts.
  - Keep STING focused on the graph mapping and algorithms portion.
- Learn how/what to forget. Cannot keep *all* data.
- Make writing high-performance STING kernels easier:
  - Looking into Julia (julialang.org) for dynamically compiled kernels, although large memory footprint.
  - Keeping an eye on other projects (many related through GRATEFUL, *e.g.* SEJITS in UCB's ASPIRE)
- PGAS for distributed memory platforms.
  - Wants a finer-grained interconnect with higher injection rates than Ethernet or IB…
  - And a PGAS mmap… (aka system-level work)

Motivation: Graph Algorithms for Analysis

STING

Review of Accomplishments

Selected Details on Recent Work

Related Projects (Future Plans)

Artifacts of the Intel Non-Numeric Computing Program

Software & Data

Where & What

STING/STINGER `http://www.cc.gatech.edu/stinger/`; framework for dynamic graph analysis.

community-el `http://www.cc.gatech.edu/~jriedy/community-detection/` or `http://gitorious.org/community-el`; separate static community detection code, simple input, no external dependencies.

PASQUAL `http://www.cc.gatech.edu/pasqual/`; shared-memory scalable and dependable *de novo* assembler

DIMACS Challenge `http://www.cc.gatech.edu/dimacs10/`; data, workshop papers and presentations

All software under a modified BSD license.

Publications I

Note: Not including presentations, posters, or publications that simply used `mirasol`...

[1] D. Ediger, K. Jiang, E. J. Riedy, and D. A. Bader, "Massive streaming data analytics: A case study with clustering coefficients," in *Proc. Workshop on Multithreaded Architectures and Applications (MTAAP)*, Atlanta, Georgia, Apr. 2010.

[2] D. Ediger, E. J. Riedy, D. A. Bader, and H. Meyerhenke, "Tracking structure of streaming social networks," in *Proc. Workshop on Multithreaded Architectures and Applications (MTAAP)*, Anchorage, Alaska, May 2011.

Publications II

[3] E. J. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader, "Parallel community detection for massive graphs," in *Proceedings of the 9th International Conference on Parallel Processing and Applied Mathematics*, Torun, Poland, Sep. 2011.

[4] ——, "Parallel community detection for massive graphs," in *10th DIMACS Implementation Challenge Workshop - Graph Partitioning and Graph Clustering*. Atlanta, Georgia: (workshop paper), Feb. 2012, won first place in the Mix Challenge and Mix Pareto Challenge.

[5] E. J. Riedy, D. A. Bader, and H. Meyerhenke, "Scalable multi-threaded community detection in social networks," in *Workshop on Multithreaded Architectures and Applications (MTAAP)*, Shanghai, China, May 2012.

Publications III

[6] R. McColl, O. Green, and D. Bader, "A new parallel algorithm for connected components in dynamic graphs," in *The 20th Annual IEEE International Conference on High Performance Computing (HiPC)*, Hyderabad, India, Dec. 2013.

[7] E. J. Riedy and D. A. Bader, "Scalable multi-threaded community detection in social networks," in *7th Workshop on Multithreaded Architectures and Applications (MTAAP)*, Boston, MA, May 2013.

[8] O. Green, R. McColl, , and D. Bader, "A fast algorithm for streaming betweenness centrality," in *4th ASE/IEEE International Conference on Social Computing (SocialCom)*, Amsterdam, The Netherlands, Sep. 2012.

Publications IV

[9] O. Green and D. A. Bader, “A fast algorithm for streaming betweenness centrality,” in *5th ASE/IEEE International Conference on Social Computing (SocialCom)*, Washington, DC, Sep. 2013.

[10] J. Riedy, H. Meyerhenke, D. A. Bader, D. Ediger, and T. G. Mattson, “Analysis of streaming social networks and graphs on multicore architectures,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Kyoto, Japan, Mar. 2012. [Online]. Available: http://www.slideshare.net/jasonriedy/icassp-2012-analysis-of-streaming-social-networks-and-graph

Publications V

[11] D. Ediger, R. McColl, J. Riedy, and D. A. Bader, “STINGER: High performance data structure for streaming graphs,” in *The IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, Sep. 2012, best paper award.

[12] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, Eds., *Graph Partitioning and Graph Clustering. 10th DIMACS Implementation Challenge Workshop*, ser. Contemporary Mathematics. Georgia Institute of Technology, Atlanta, GA: American Mathematical Society and Center for Discrete Mathematics and Theoretical Computer Science, 2013, no. 588.

Publications VI

[13] X. Liu, P. Pande, H. Meyerhenke, and D. A. Bader, "PASQUAL: Parallel techniques for next generation genome sequence assembly," *IEEE Transactions on Parallel & Distributed Systems*, vol. 24, no. 5, pp. 977–986, 2013.

[14] J. Fairbanks, D. Ediger, R. McColl, D. Bader, and E. Gilbert, "A statistical framework for analyzing streaming graphs," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Aug. 2013.

People!

GT *is* an educational institution

- Related graduate students:
  - David Ediger: graduated, Ph.D., at GTRI
  - Rob McColl: nearing graduation, shared with GTRI
  - Oded Green: nearing graduation
  - James Fairbanks
  - Anita Zakrzewska
  - Xing Liu: PASQUAL
  - Pushkar Pande: PASQUAL, graduated, MS
- Related post-doc:
  - Henning Meyerhenke: Juniorprof. Dr. at Karlsruhe Inst. Tech.

Acknowledgment of support