Massive Streaming Data Analytics: A Case Study with Clustering Coefficients

David Ediger, Karl Jiang, Jason Riedy and David A. Bader

Georgia Tech College of Computing
Computational Science and Engineering

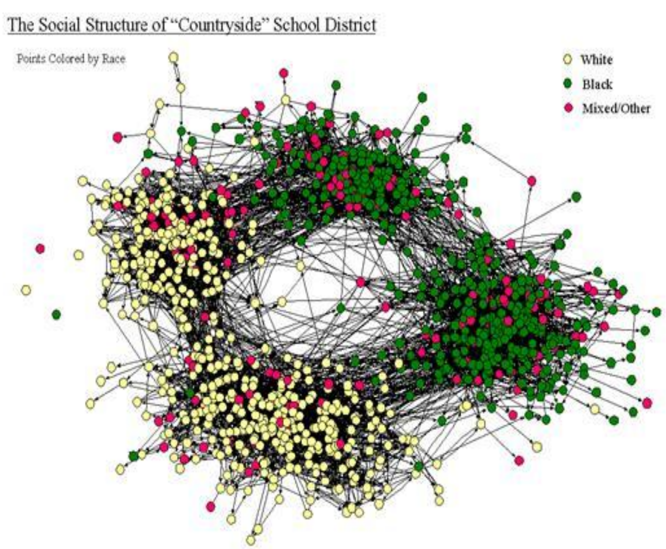
Overview

- Motivation
- A Framework for Massive Streaming Data Analytics
- STINGER
- Clustering Coefficients
- Results on Cray XMT & Intel Nehalem-EP
- Conclusions

Data Deluge

Current data rates:

- NYSE: 1.5TB daily
- LHC: 41TB daily
- LSST: 13TB daily



- 1 Gb Ethernet: 8.7TB daily at 100%, 5-6TB daily realistic
- Multi-TB storage on 10GE: 300TB daily read, 90TB daily write

Emerging Applications

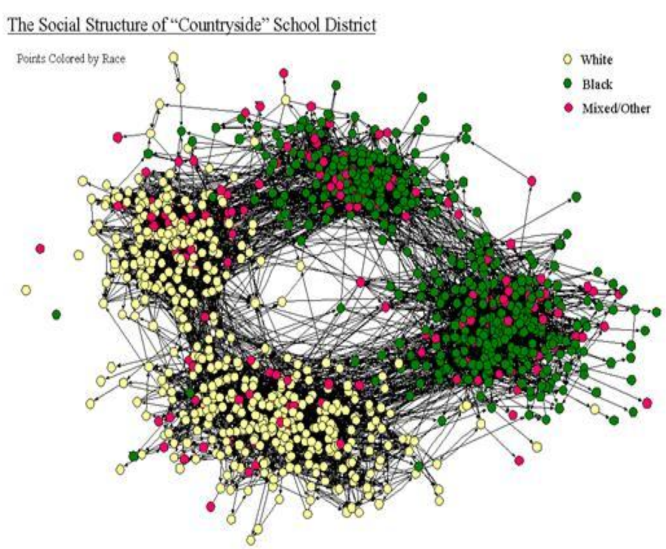
Business Analytics

Social Network Analysis

Data Deluge

Current data sets:

- NYSE: 8PB
- Google: >12PB
- LHC: >15PB



- CPU<->Memory:
  - QPI,HT: 2PB/day@100%
  - Power7: 8.7PB/day
- Mem:
  - NCSA Blue Waters tgt: 2PB

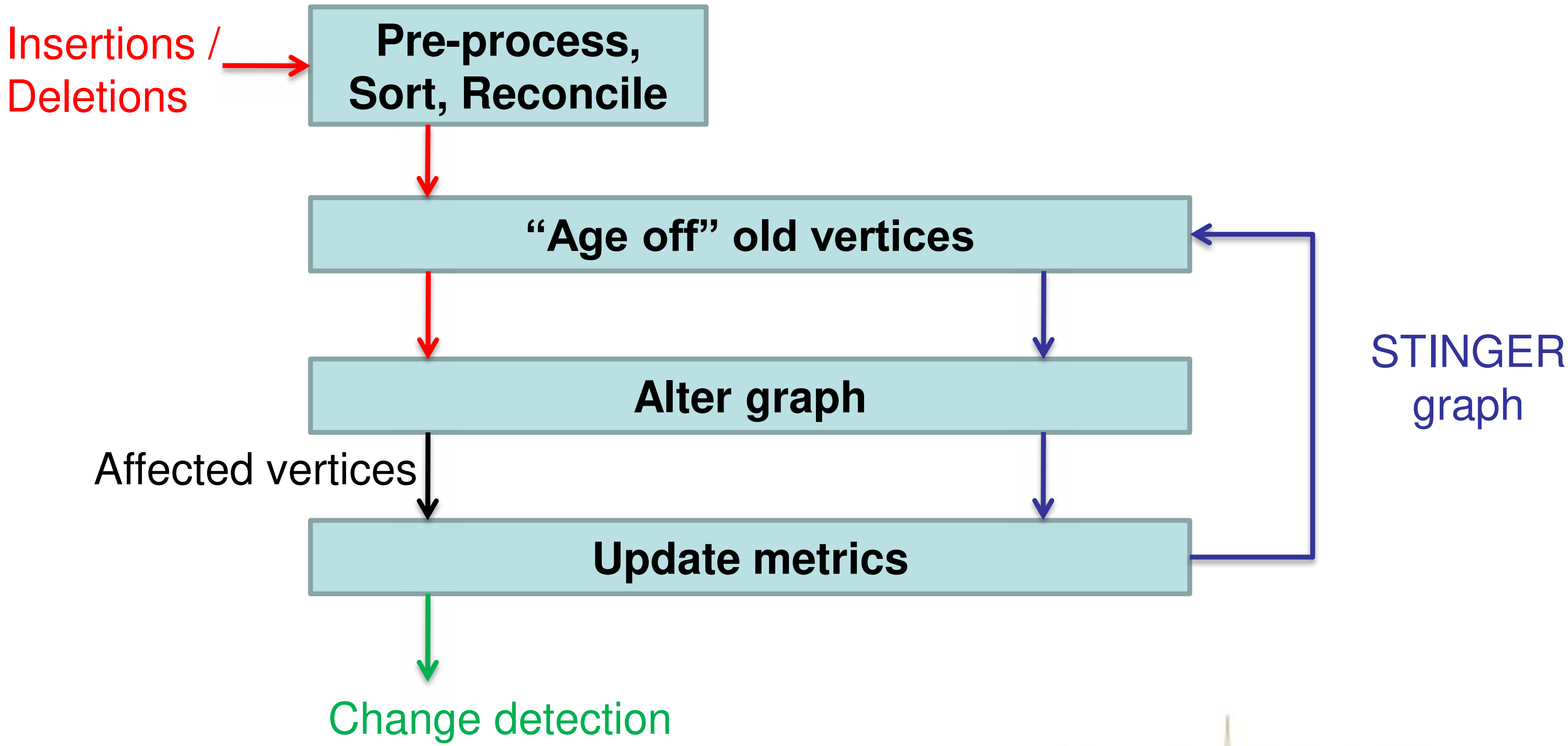
→ Even with parallelism, current systems cannot handle more than a few passes... per day.

Our Contributions

- A new computational approach for the analysis of complex graphs with streaming spatio-temporal data
- STINGER
- Case study: clustering coefficients
  - Bloom filters and batch updates
  - 4 orders of magnitude faster than recomputation

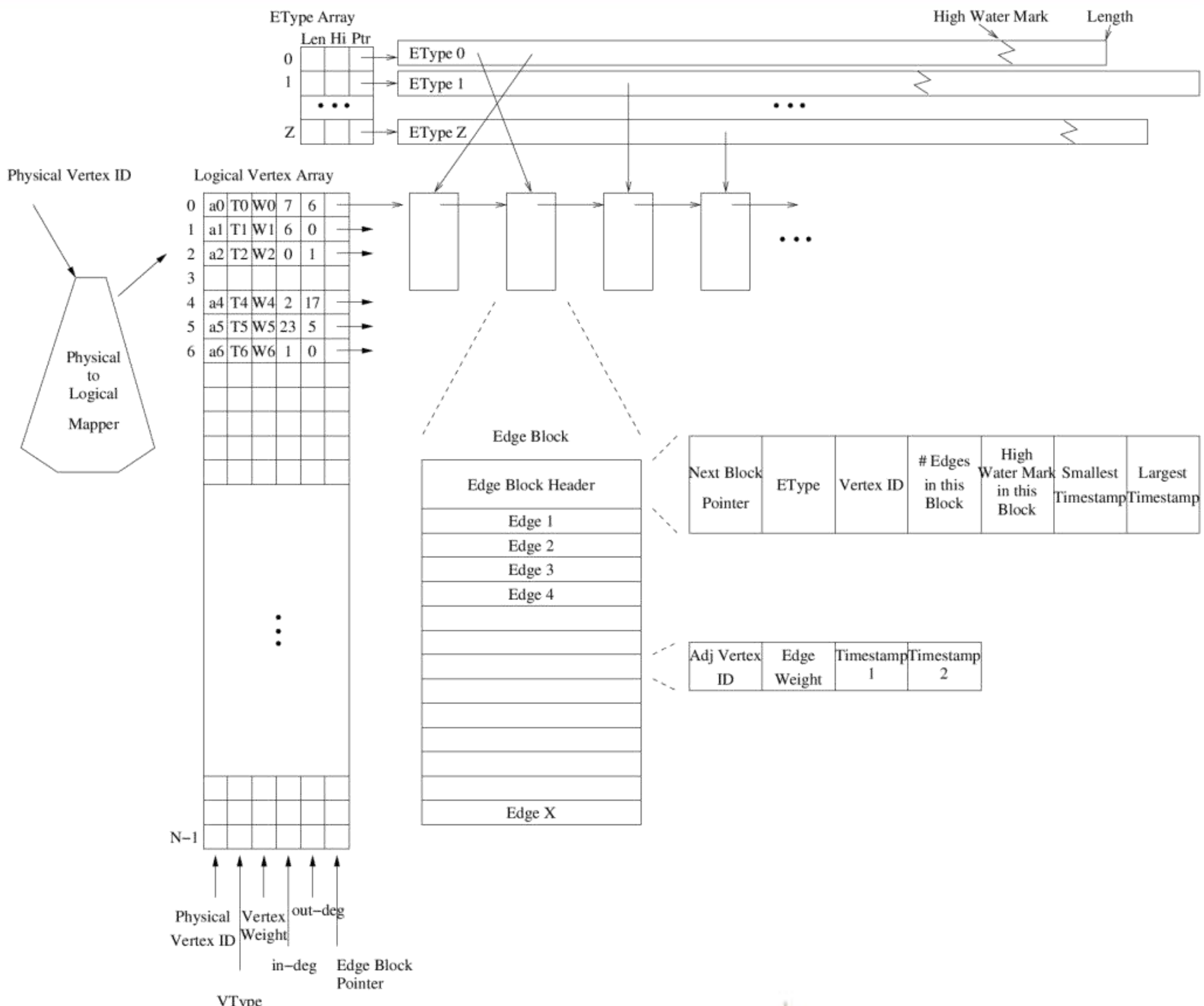
Massive Streaming Data Analytics

- Accumulate as much of the recent graph data as possible in main memory.



STINGER: A temporal graph data structure

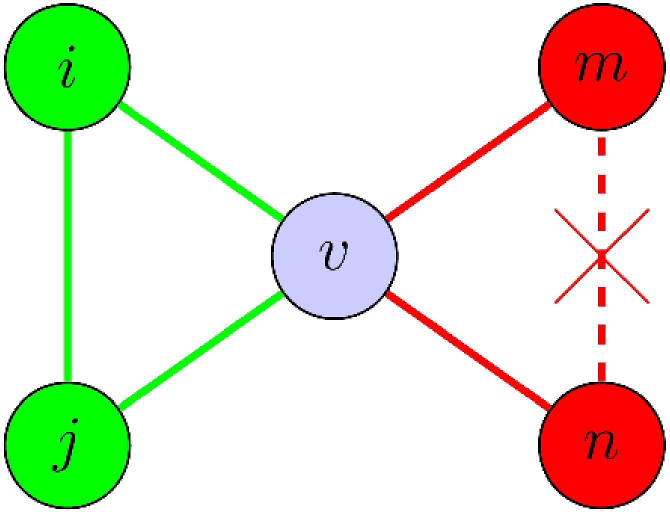
- Semi-dense edge list blocks with free space
- Compactly stores timestamps, types, weights
- Maps from application IDs to storage IDs
- Deletion by negating IDs, separate compaction



Georgia Tech College of Computing

Definition of Clustering Coefficients

- Defined in terms of ***triplets***.
- # closed triplets / # all triplets

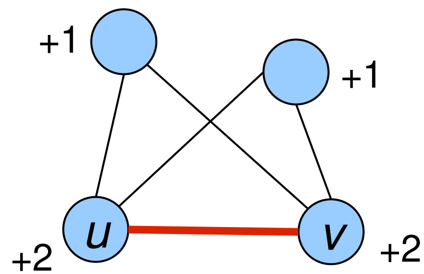


- *i-j-v* is a ***closed triplet*** (triangle).
- *m-v-n* is an ***open triplet***.
- Locally, count those around *v*.
- Globally, count across entire graph.
  - Multiple counting cancels (3/3=1)

- Useful for understanding topology, community structure, and small-worldness (Watts98).

Streaming updates to clustering coefficients

- Monitoring clustering coefficients could identify anomalies, find forming communities, etc.
- Computations stay local. A change to edge <*u*, *v*> affects only vertices *u*, *v*, and their neighbors.



- Need a fast method for updating the triangle counts, degrees when an edge is inserted or deleted.
  - Dynamic data structure for edges & degrees: STINGER
  - Rapid triangle count update algorithms: exact and approximate

The Local Clustering Coefficient

$$C_v = \frac{\text{number of closed triplets centered around } v}{\text{number of triplets centered around } v}.$$

$$C_v = \frac{\sum_{i \in e_v} |e_i \cap (e_v \setminus \{v\})|}{d_v(d_v - 1)} = \frac{T_v}{d_v(d_v - 1)}.$$

Where e_k is the set of neighbors of vertex k and d_k is the degree of vertex k

We will maintain the numerator and denominator separately.

Algorithm for Updates

Algorithm 1 An algorithmic framework for updating local clustering coefficients. All loops can use atomic increment and decrement instructions to decouple iterations.

Input: Edge $\langle u, v \rangle$ to be inserted (+) or deleted (−), local clustering coefficient numerators T, and degrees d

Output: Updated local triangle counts T and degrees d

$d_u \leftarrow d_u \pm 1$

$d_v \leftarrow d_v \pm 1$

$count \leftarrow 0$

for all $x \in e_v$ **do**

    if $x \in e_u$ **then**

        $T_x \leftarrow T_x \pm 1$

        $count \leftarrow count \pm 1$

$T_u \leftarrow T_u \pm count$

$T_v \leftarrow T_v \pm count$

Three Update Mechanisms

- Update local & global clustering coefficients while edges <*u*, *v*> are inserted and deleted.
- Three approaches:

1. Exact: Explicitly count triangle changes by doubly-nested loop.
   - $O(d_u * d_v)$, where d_x is the degree of x after insertion/deletion
2. Exact: Sort one edge list, loop over other and search with bisection.
   - $O((d_u + d_v) \log (d_u))$
3. Approx: Summarize one edge list with a Bloom filter. Loop over other, check using $O(1)$ **approximate** lookup. May count too many, never too few.
   - $O(d_u + d_v)$

Bloom Filters

Bit Array	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bloom Filter	0	0	1	0	0	0	0	0	1	0	1	1

HashA(10) = 2
HashB(10) = 10

HashA(23) = 11
HashB(23) = 8

- **Bit Array:** 1 bit / vertex
- **Bloom Filter:** less than 1 bit / vertex
- Hash functions determine bits to set for each edge
- Probability of false positives is known (prob. of false negatives = 0)
  - Determined by length, # of hash functions, and # of elements
- Must rebuild after a deletion

Experimental Methodology

- RMAT (Chakrabarti04) as a graph & edge generator.
- Generate graph with SCALE and edge factor F, $2^{SCALE}F$ edges.
  - SCALE 24: 17 million vertices
  - Edge factors 8 to 32: 134 to 537 million edges
- Generate 1024 actions.
  - Deletion chance 6.25% = 1/16
  - Same RMAT process, will prefer same vertices.
- Start with an exact triangle count, run individual updates.
- For batches of updates, generate 1M actions.

The Cray XMT

- Tolerates latency by massive multithreading.
  - Hardware support for 128 threads on each processor
  - Globally hashed address space
  - No data cache
  - Single cycle context switch
  - Multiple outstanding memory requests
- Support for fine-grained, word-level synchronization
  - Full/empty bit associated with every memory word

Image Source: cray.com

- Flexibly supports dynamic load balancing.
- Testing on a 128 processor XMT: 16384 threads
  - 1 TB of globally shared memory

The Intel 'Nehalem-EP'

- Dual socket Intel Xeon E5530 @ 2.4 GHz
- 12 GB memory
- 8 Physical Cores, 2x SMT
- 32 GB/s per socket

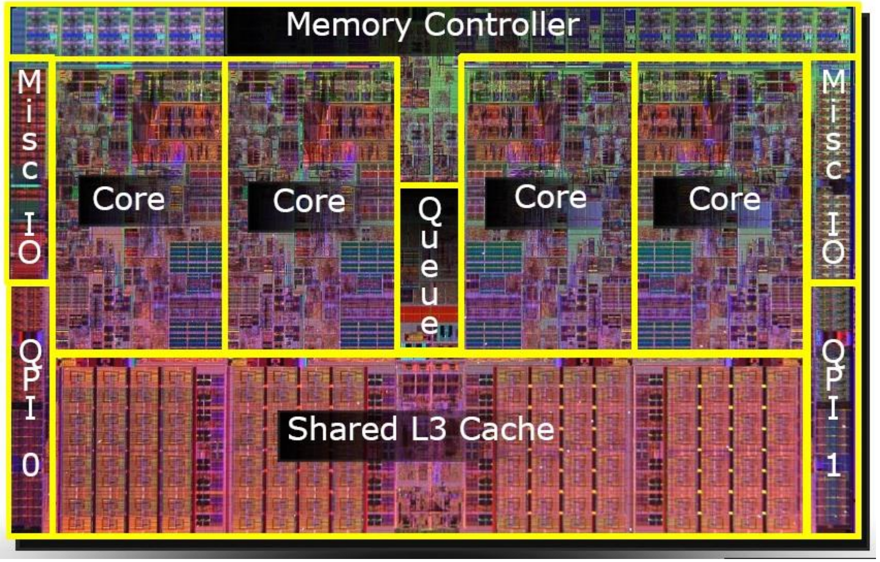
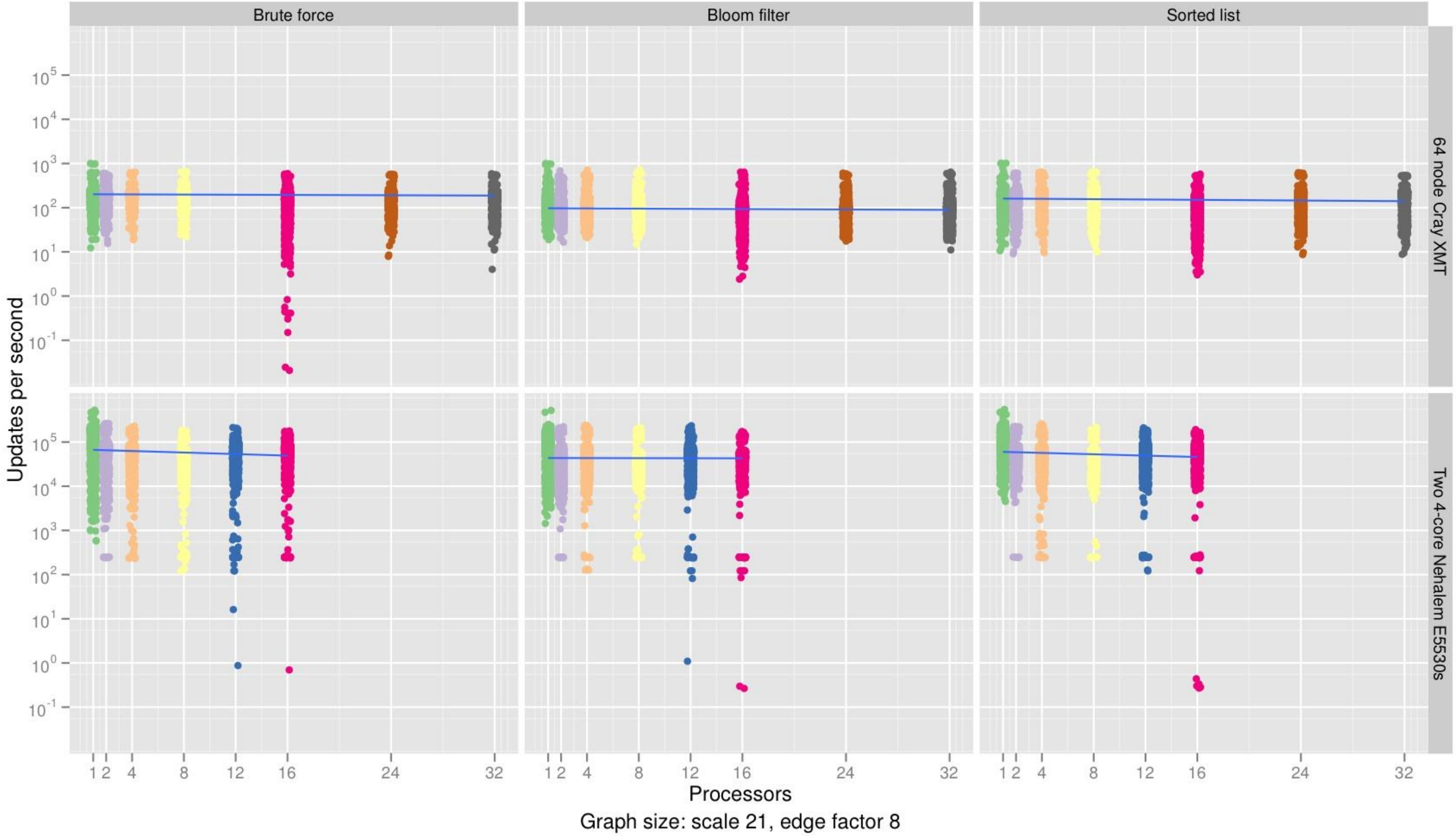
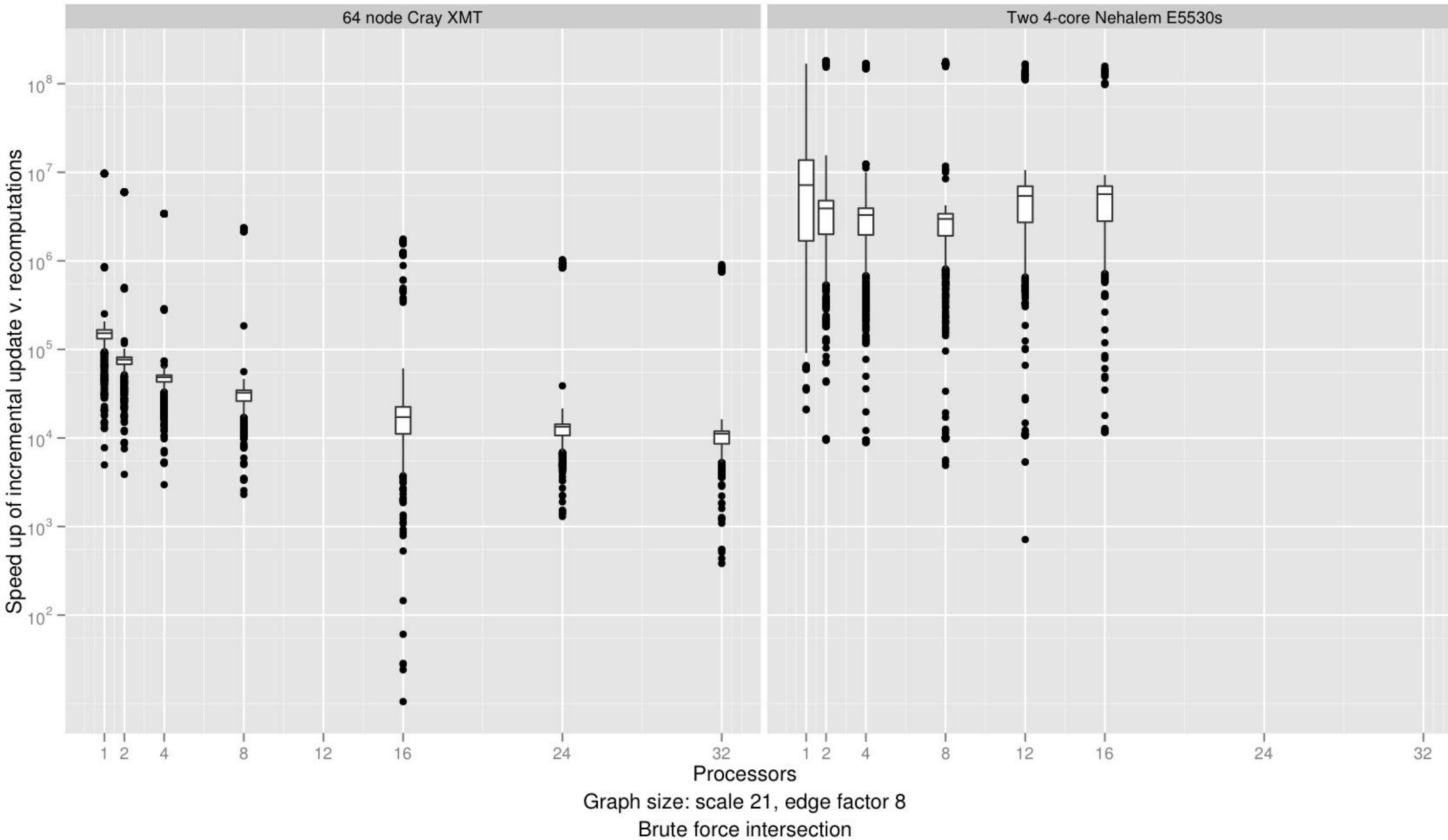


Image Source: intel.com

Updating clustering coefficients one-by-one



Speed-up over recomputation



- Cray XMT: over 10,000x faster
- Intel Nehalem: over 1,000,000x faster

Updating clustering coefficients in a batch

- Start with an exact triangle count, run individual batched updates:
  - Consider B updates at once.
  - Loses some temporal resolution within a batch. Changes to the same edge are collapsed.
- Result summary (updates per second)

Algorithm	B = 1	B = 1000	B = 4000
Exact	90	25,100	50,100
Approx.	60	83,700	193,300

32 of 64P Cray XMT, 16M vertices, 134M edges

Conclusions

- STINGER: efficiently handles graph traversal and edge insertion & deletion.
- A serial stream of edges contains sufficient parallelism for Cray XMT to obtain 550x speed-up over edge-by-edge updates.
- Bloom filters may introduce an approximation, but can achieve an additional 4x speed-up on the Cray XMT.

References

- D. A. Bader, J. Berry, A. Amos-Binks, D. Chavarría-Miranda, C. Hastings, K. Madduri, and S. C. Poulos, “STINGER: Spatio-Temporal Interaction Networks and Graphs (STING) Extensible Representation,” Georgia Institute of Technology, Tech. Rep., 2009.
- D. Chakrabarti, Y. Zhan, and C. Faloutsos, “R-MAT: A recursive model for graph mining,” in *Proc. 4th SIAM Intl. Conf. on Data Mining (SDM)*. Orlando, FL: SIAM, Apr. 2004.
- D. Watts and S. Strogatz, “Collective dynamics of small world networks,” *Nature*, vol. 393, pp. 440–442, 1998.

Acknowledgments