Efficient Dynamic Graph Algorithms on GPU Graph Frameworks

ELECTRICAL [+] COMPUTER

E N G I N E E R I N G

May 14, 2019

Euna Kim

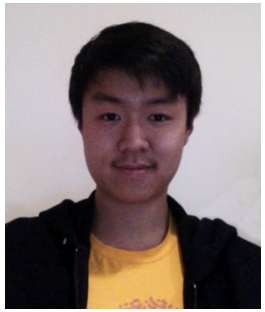
Georgia Tech School of Electrical and Computer Engineering

High Performance Computing group

- Graph data analysis, design and implementation of scalable graphs algorithms for both static and dynamic graphs

- Advisor: Dr. David Bader
- Dr. David Ediger, Ph.D. Georgia Tech, GTRI, STINGER development
- Mrs. Euna Kim, MSc. GT (2011), 11 years at Sony & Samsung
- Mr. James Fox, BSc. UC Berkeley (2016)
- Mr. Kasimir Gabert, MSc. GT (2012), 4 years Sandia
- Mrs. Xiaojing An, MSc. Chinese Academy of Science (2017)

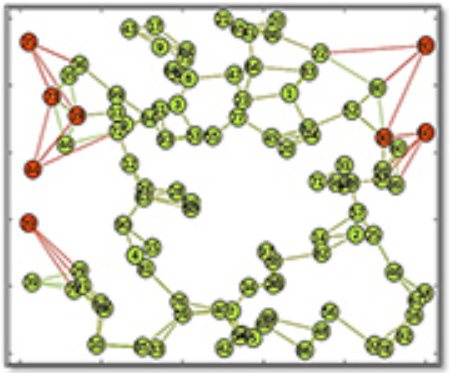


DARPA HIVE

- Graph Processor Development

Cyber Security



Which IP events are probes on the network?

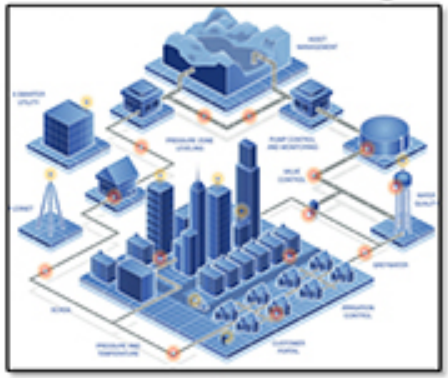
- Who are they probing, who have they infected in the network?
- Only a small number of events are probes – graph is sparse.

Social Media Analysis

Who influences me to buy a product?

- Who has access to my social media pages, what are they saying to me?
- Since only a few people have direct influence on me – graph is sparse.

Infrastructure Monitoring



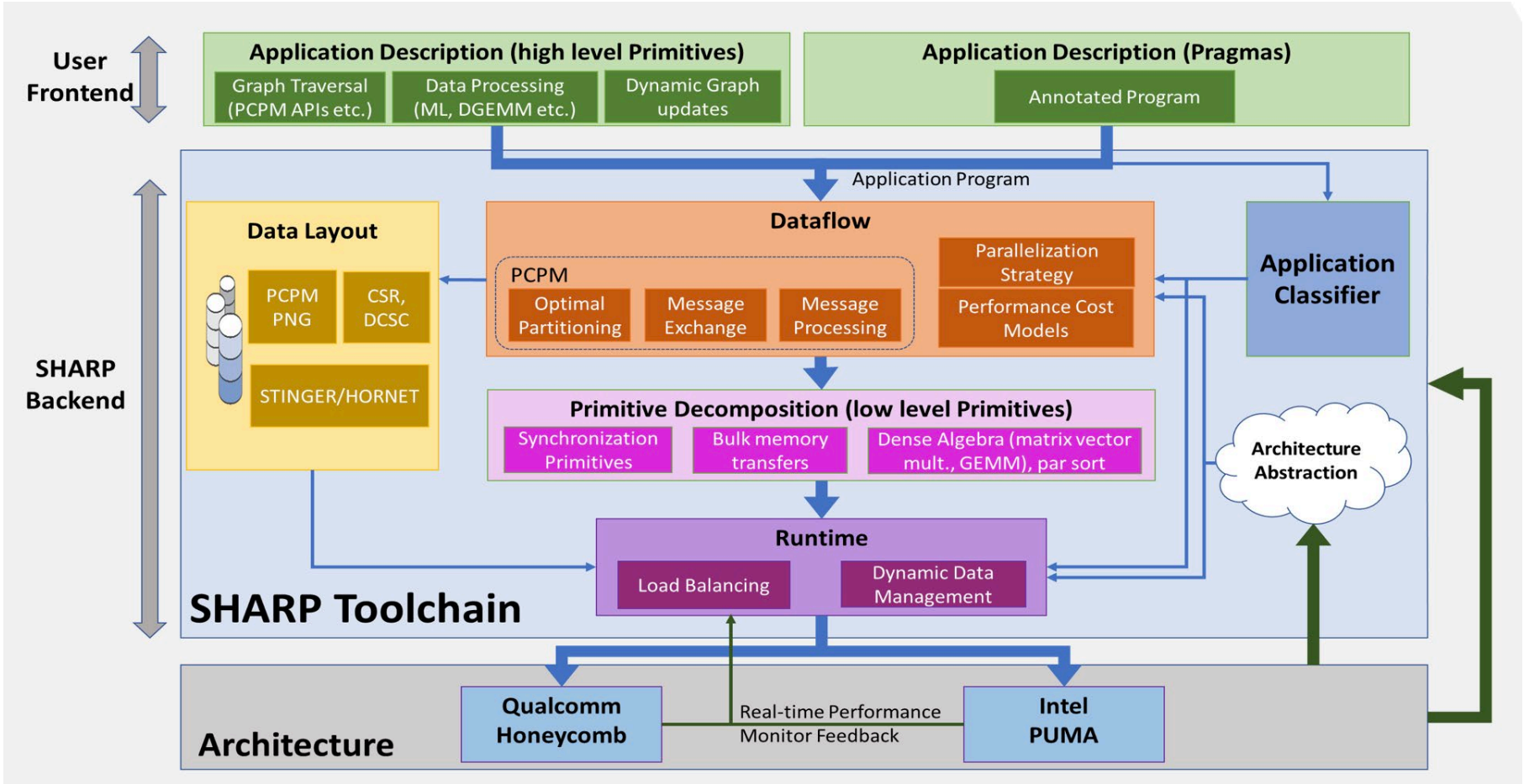
Can I spot failures before they become critical?

- How do I avoid cascading failures and what are the system dependencies?
- Only a small number of critical dependencies – graph is sparse.

https://www.darpa.mil/program/hierarchical-identify-verify-exploit

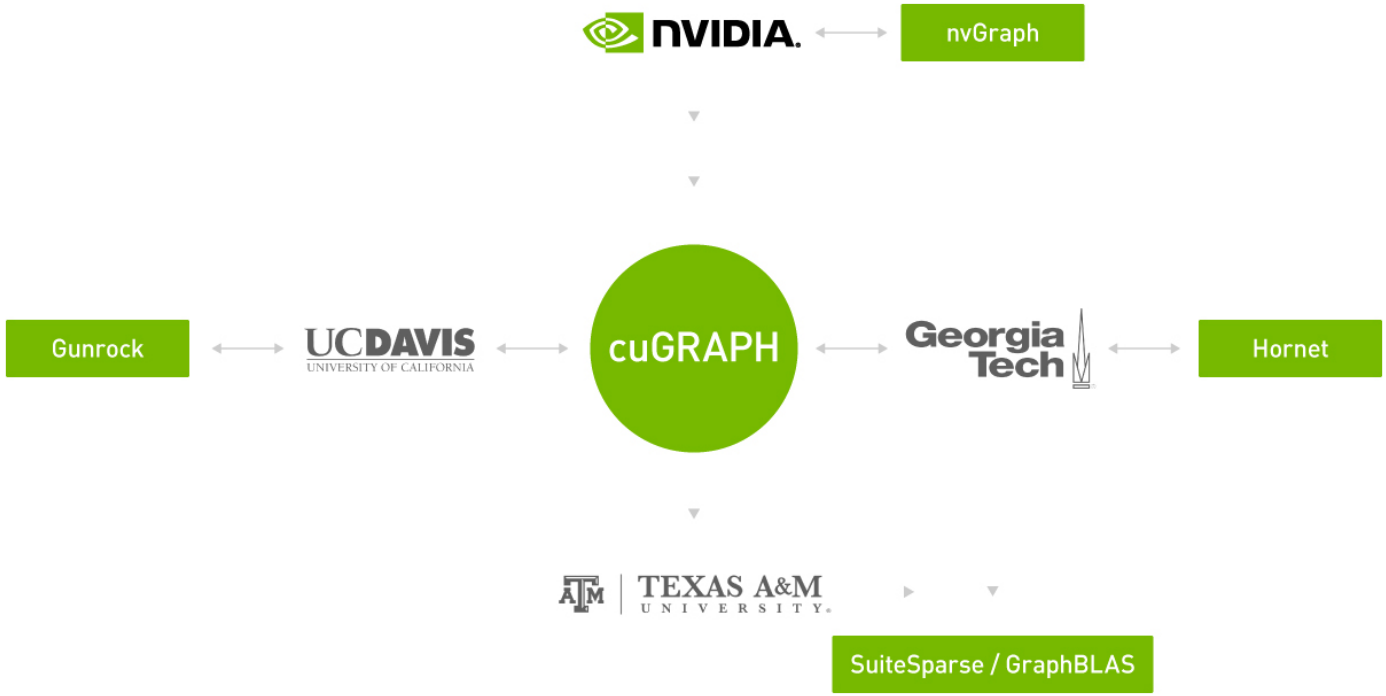
https://www.hpcwire.com/2017/06/05/darpa-picks-intel-qualcomm-pnnl-2-others-tackle-hive-project/

SHARP Team


User Frontend
Application Description (high level Primitives)
Graph Traversal (PCPM APIs etc.)
Data Processing (ML, DGEMM etc.)
Dynamic Graph updates
Application Description (Pragmas)
Annotated Program
Application Program
SHARP Backend
Data Layout
PCPM PNG
CSR, DCSC
STINGER/HORNET
Dataflow
PCPM
Optimal Partitioning
Message Exchange
Message Processing
Parallelization Strategy
Performance Cost Models
Application Classifier
Primitive Decomposition (low level Primitives)
Synchronization Primitives
Bulk memory transfers
Dense Algebra (matrix vector mult., GEMM), par sort
Architecture Abstraction
Runtime
Load Balancing
Dynamic Data Management
SHARP Toolchain
Architecture
Qualcomm Honeycomb
Real-time Performance Monitor Feedback
Intel PUMA

NVIDIA

- NVAIL program partnership



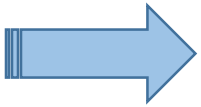
https://news.developer.nvidia.com/graph-technology-leaders-combine-forces-to-advance-graph-analytics/

Graph processing

- Graph Data Structure: Entity and Relationship representation of data
- Graph Format: Dense/Sparse, COO/CSR/… for sparse graphs
- Graph Application: BFS, DFS, PageRank, SSSP, Triangle Counting, etc.
- Graph System: 1) Architecture: CPU(s), GPU(s), Hybrid (CPUs and GPUs)
  2) Memory: Shared/Distributed
- **Graph Framework**: cuStinger, **Hornet, Gunrock**, Ligra, AIM, etc.
- Graph Libraries: GraphBLAS, **cuGraph** (CUSP, SPMV), etc.
- Graph Challenges: Scalability, Irregularity, Distributed/Parallel approaches
- Graph Challenges on GPUs: Load balancing, Memory footprint
- Graph Types: Real-world graphs, Synthetic graphs such as RMAT, Kronecker graphs
- Graph Programming Models: BLAS, Vertex/Edge Centric, Gather-Scatter (GAS) model

Performance

- **Performance factors**
  1) Applications & Graph data characteristics
  2) Hardware choice
  3) Data structure
  4) Programming model
  5) Optimization / Parallelization / Synchronization
  6) Algorithm implementation

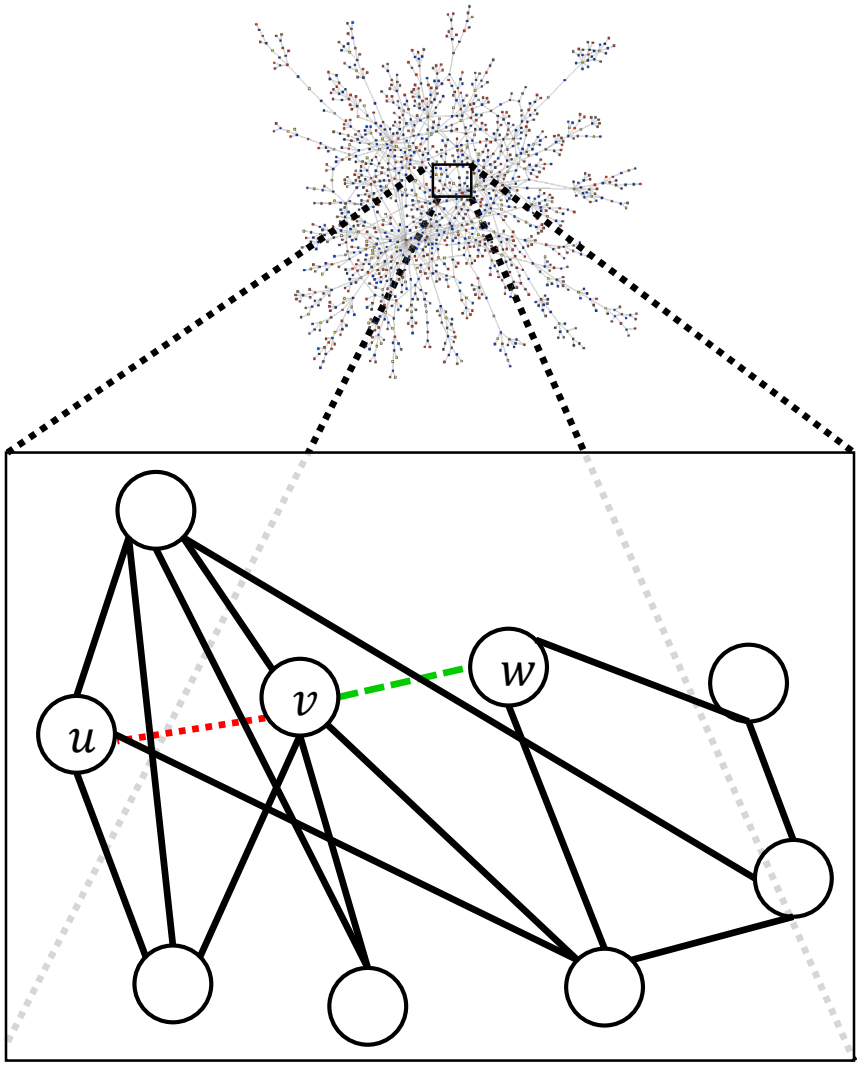


Graph Frameworks

- **Graph Benchmarks**: **Graph500 – BFS**, SSSP, PageRank, Connected Components, Betweenness Centrality, Triangle Counting and extra.
- **Graph Performance Metrics**: FLOPS(LINPACK), **GTEPS**(Graph500), Watt(Energy)

Dynamic Graph Examples

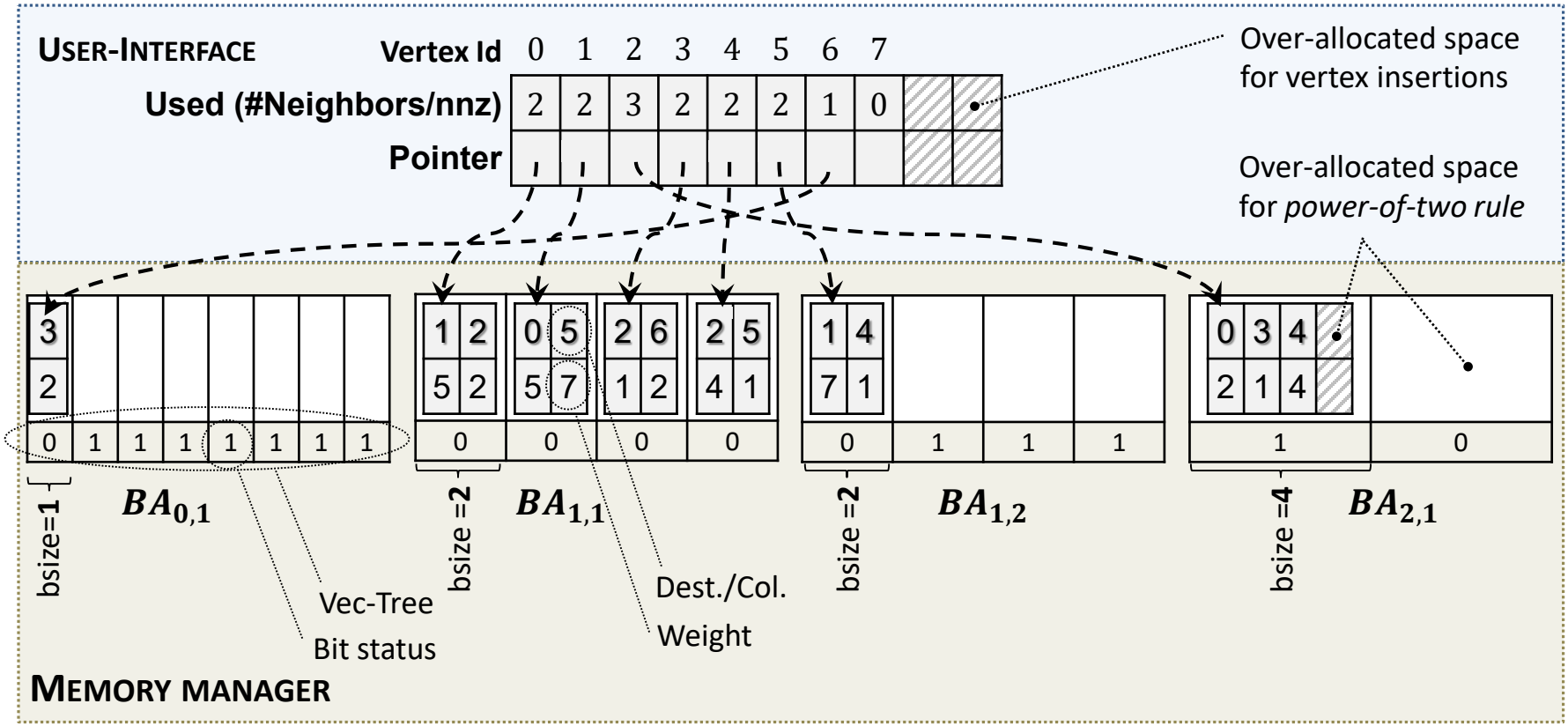
- Only a subset of the entire Dynamic:
  - At time t:
    - v and w become friends.
    - $insert\_edge\ (v, w)$
  - At time $\hat{t}$:
    - $u\ and\ v\ no\ longer\ friends$
    - $delete_{edge(u,v)}$
- Additional operations include vertex insertions & deletions



Oded Green, GTC-DC-17

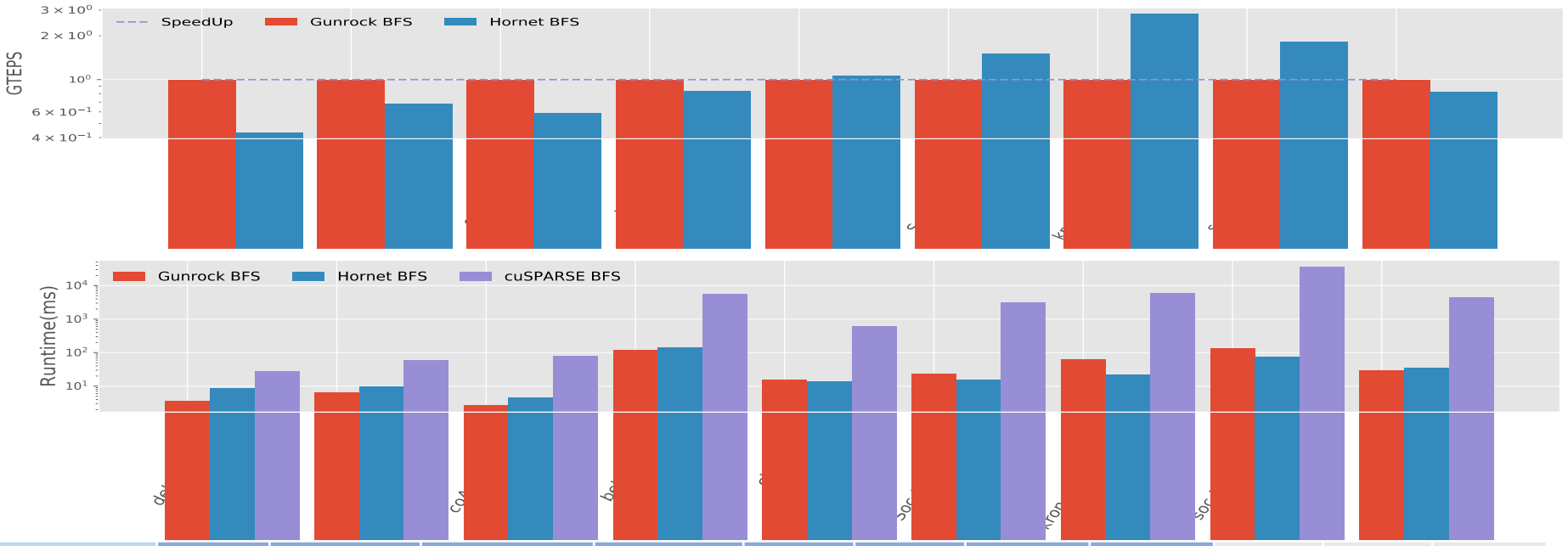
Hornet

- A flexible data structure to support static / dynamic graphs



Oded Green, GTC-DC-17

GPU Graph Frameworks Comparison



Graph_name	vertex	edges(directed)	edges(undirected)	highest_degree	root	iteration	node_visited	edge_visited	Gunrock	Hornet	cuSPARSE
delaunay_n13	8192	49094	98188	12	3976	41	8192	49094	3.619909	8.4	27.8426
ak2010	45,292	217,098	434196	430	35989	46	43281	204182	6.5081	9.5	58.4386
coAuthorsDBLP	299,067	1,955,352	3910704	336	4344	14	299067	1955352	2.6989	4.6	77.2696
belgium_osm	1,441,295	3,099,940	6199880	10	120988	1304	1441295	3099940	119.1669	143.1	5487.33
cit-Patents	3,774,768	16,518,948	33037896	793	2514765	17	3764117	33023480	14.859915	13.9	610.242
Soc-LiveJournal1	4,847,571	68,993,773	137987546	20333	10009	13	4843953	85691368	23.200989	15.5	3007.42
kron_g500-logn21	2,097,152	182,084,020	364168040	213904	1930586	5	1543901	182081678	61.538935	21.8	5782.38
soc-twitter-2010	21297772	265,025,545	530051090	698112	10119000	15	21297772	530051090	135.9649	74.9	35616.9
uk2002	18,520,486	298,113,762	596227524	2449	15353974	40	17994090	284157721	28.498888	34.6	4454.43

BFS implementation & Verification

(1) Hornet Implementation

- Running the same algorithm in CPU and compare the result to results on GPU

(2) Gunrock Implementation

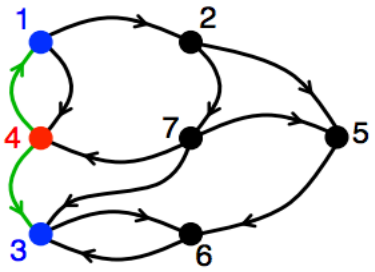
- Removing the cycles in the beginning
- The difference of precedents' label value connected with an edge in the BFS tree should be exactly 1

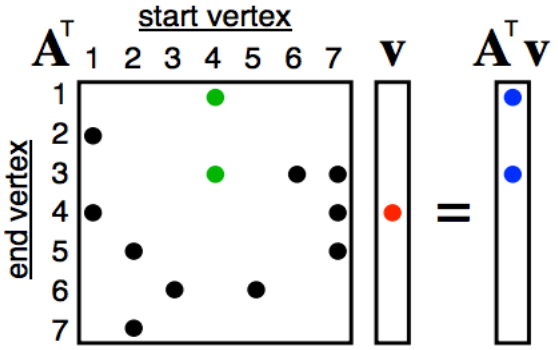
**Graph 500 suggestions

- Remove cycles first and the BFS levels should differ by exactly one

(3) cuSPARSE

- Test with small contrived matrix: all elements in vector after each SpMV computation
- GraphBLAS(a set of mathematical operations for graph algorithm) representation of BFS





$\oplus \equiv \cup \qquad \otimes \equiv \cap \qquad a, b, c \subset \mathbb{Z}$

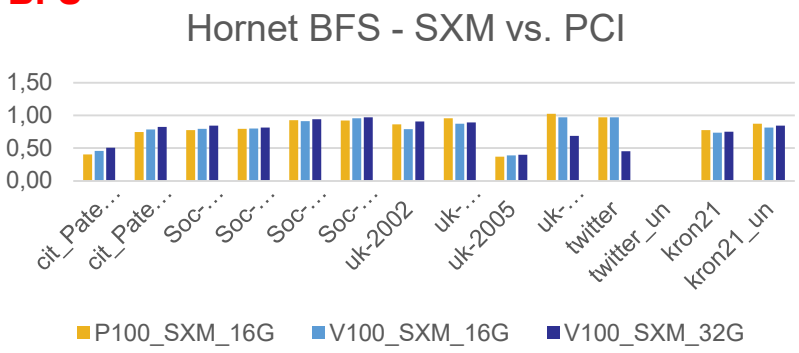
$\mathbf{A} : \mathbb{S}^{m \times l}$, $\mathbf{B} : \mathbb{S}^{l \times m}$, and $\mathbf{C} : \mathbb{S}^{m \times n}$

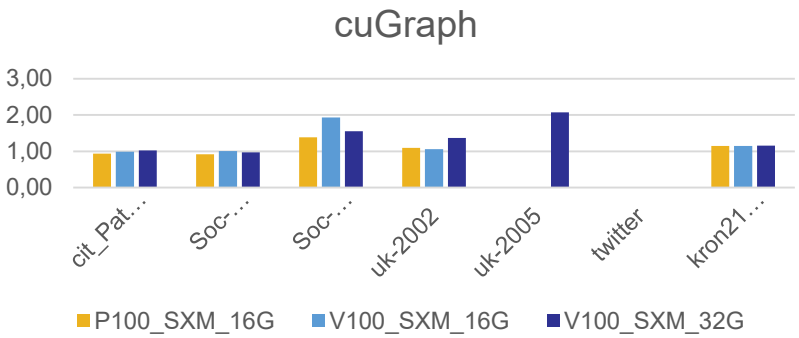
$$\mathbf{C}(i,j) = \sum_{k=1}^{l} \mathbf{A}(i,k)\mathbf{B}(k,j)$$

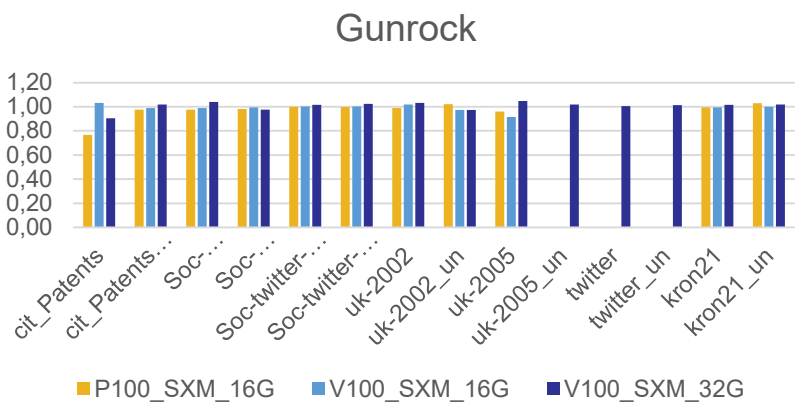
$$\mathbf{C}^{\mathsf{T}} \oplus= \mathbf{A}^{\mathsf{T}} \oplus.\otimes \mathbf{B}^{\mathsf{T}}$$

where $\mathbf{A} : \mathbb{R}^{m \times l}$, $\mathbf{B} : \mathbb{R}^{l \times n}$, and $\mathbf{C} : \mathbb{R}^{m \times n}$.

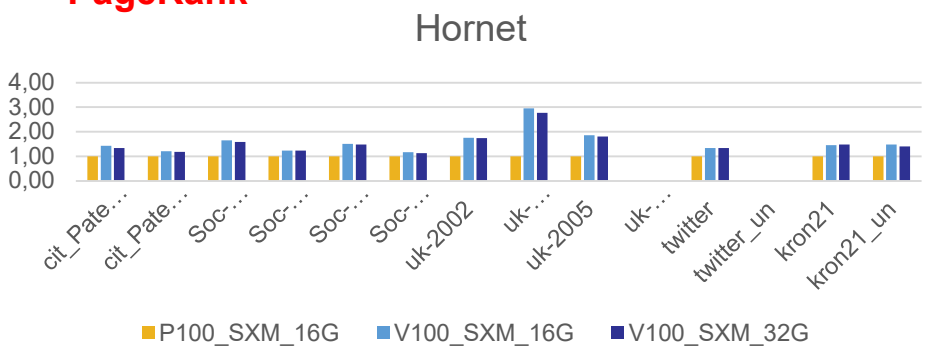
BFS

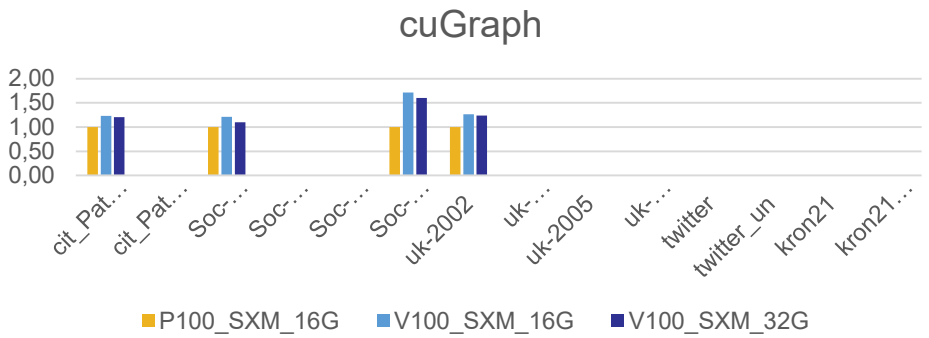


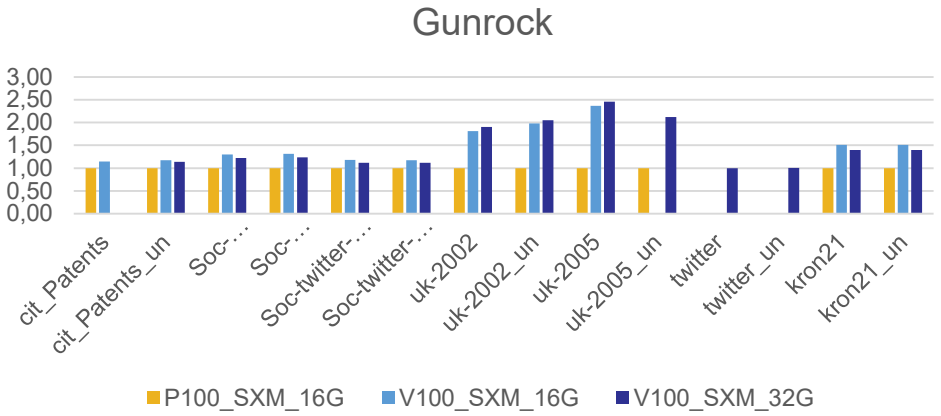




PageRank

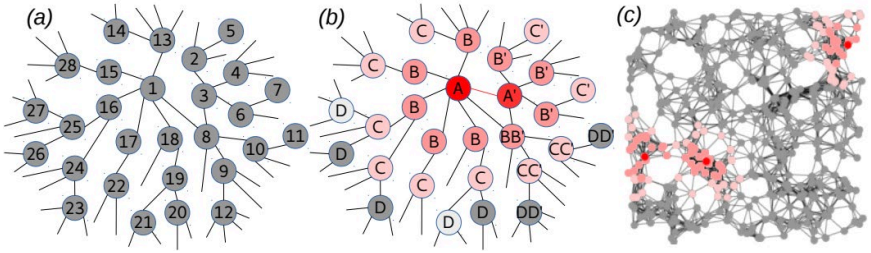


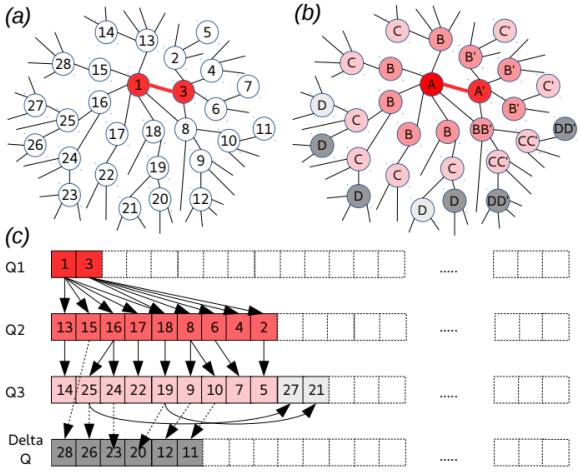


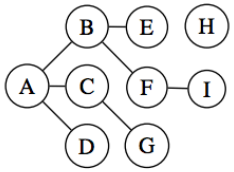
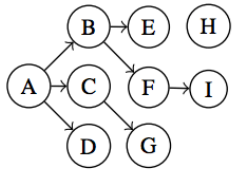
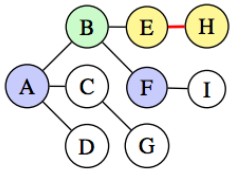
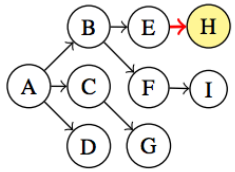
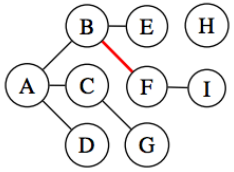
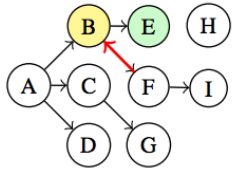
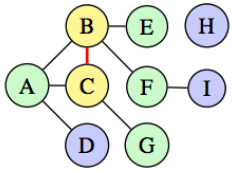
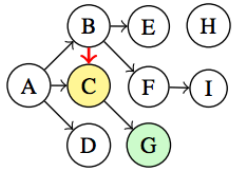
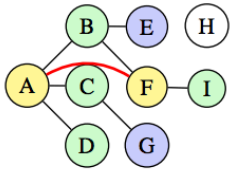
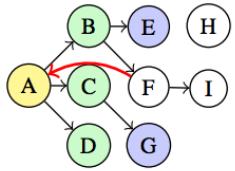


Dynamic PageRank Algorithm

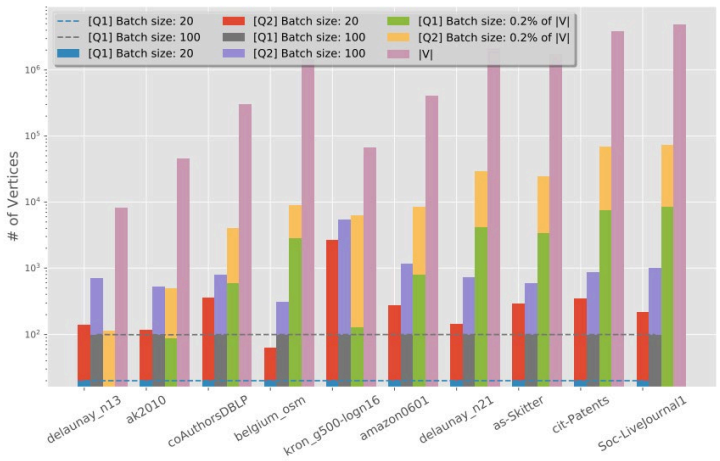
$$PR^{t+1}(v) = \frac{(1-d)}{|V|} + d \sum_{u \in adj(v)} \frac{PR^t(u)}{out\_degree(u)}$$





Case	Undirected	Directed
a) Initial state		
b) $e(E, H)$ inserted		
c) $e(F, B)$ inserted		
d) $e(B, C)$ inserted		
e) $e(F, A)$ inserted		

Dynamic PageRank Performance



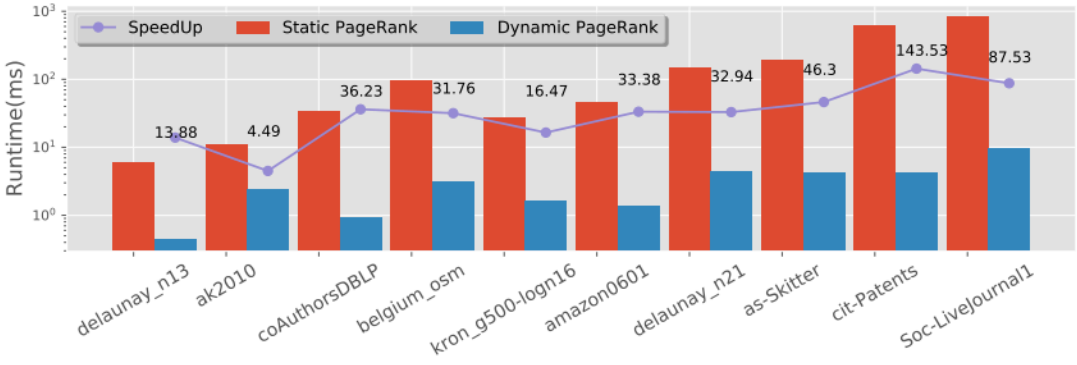
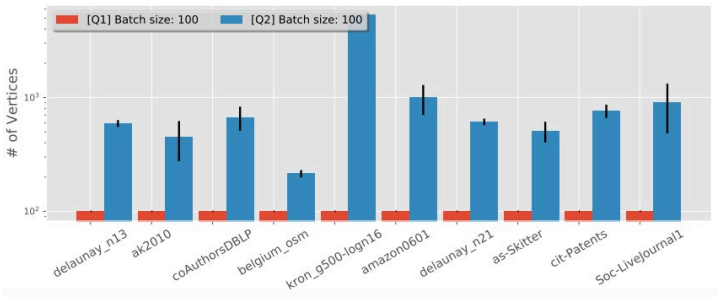


Fig. 5. Execution time (log scale) for both static graph and dynamic graph algorithms for a batch size of 20 edges.



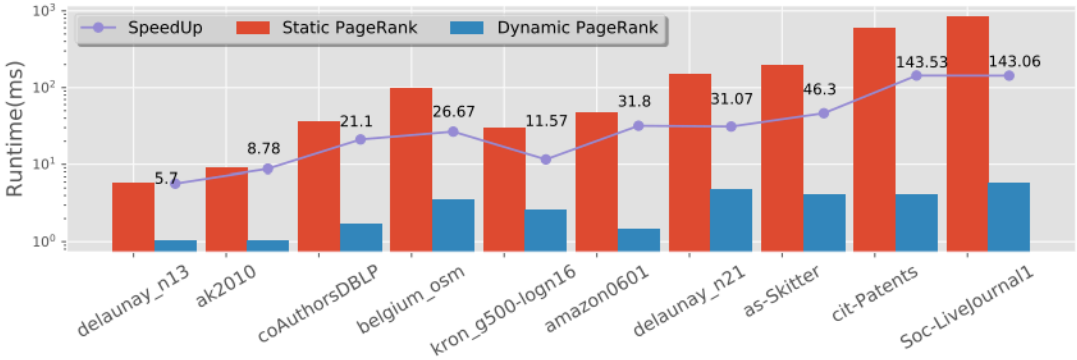


Fig. 6. Execution time (log scale) for both static graph and dynamic graph algorithms for a batch size of 100 edges.

Future

- New applications discovery
- ML + Graph algorithms (GraphSAGE, GraphSAINT…)
- Extreamely large scale graphs (c.f. Graph500)
- New programming models for the new graph processors
- Hardware/Software co-design
- Accelerators (GPU, FPGA, etc)