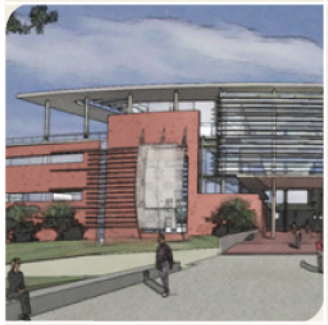
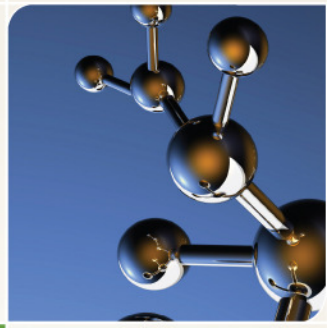
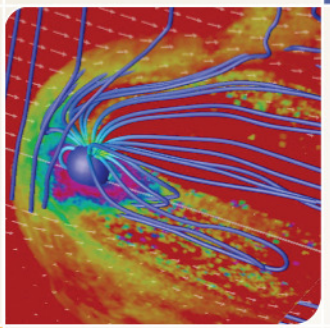
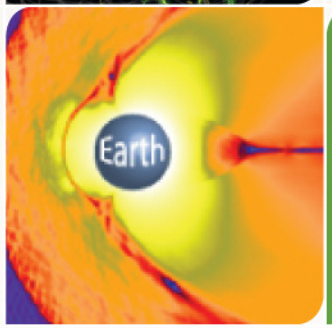
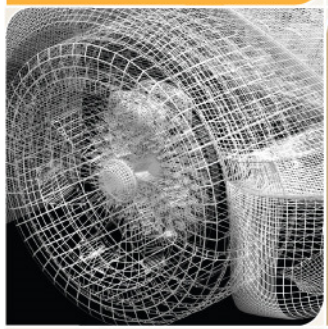


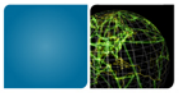






cuSTINGER - Supporting Dynamic Graph Algorithms for GPUs

Oded Green & David Bader

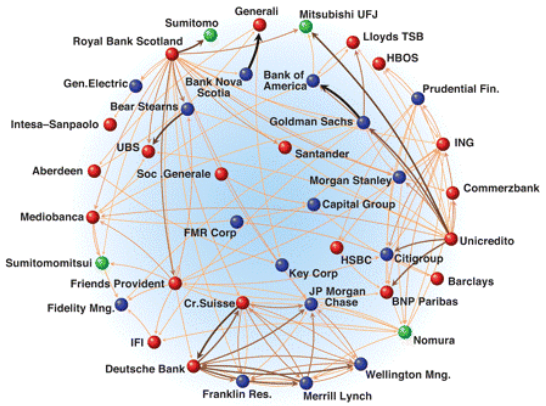
Georgia Tech College of Computing
Computational Science and Engineering


What we will see today

- The first dynamic graph data structure for the GPU.
  - Scalable in size
  - Supports the same functionality is its CPU counterpart
- Supports extremely fast update rates.
- Good performance for static graph algorithms.

Big Data problems need Graph Analysis



Financial networks:

- Transactions between players.
- Different transactions types (property graph)

Communication networks:

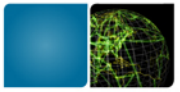
- World-wide connectivity
- High velocity changes
- Different types of extracted data:
  - Physical communication network.
  - Person-to-person communication network

Health-Care networks:

- Various players.
- Pattern matching and epidemic monitoring.
- Problem sizes have doubled in last 5 years.

Graphs are a unifying motif for data analytics.

More importantly are *dynamic and streaming* graphs!

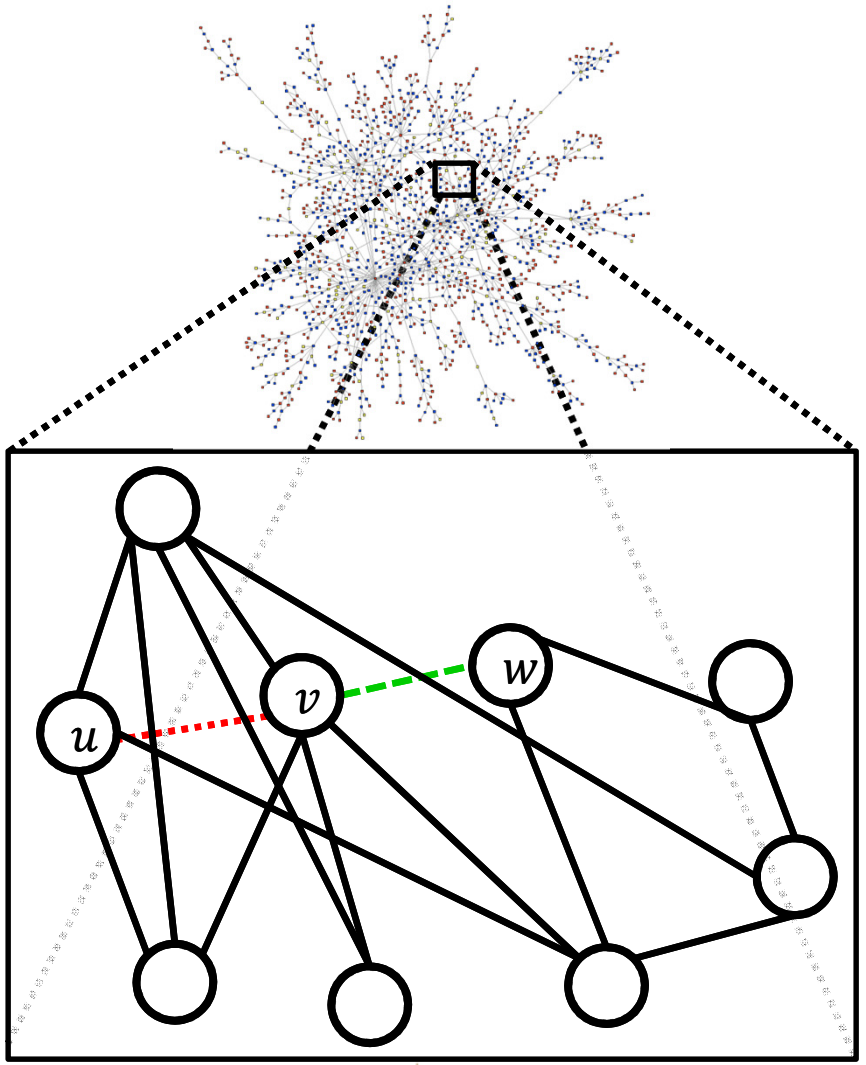




Definitions

- STINGER: Spatio-Temporal Interaction Networks and Graphs (STING) Extensible Representation
- Dynamic graphs
  - Graph can change over time.
  - Changes can be to topology, edges, or vertices.
    - For example new edges between two vertices.
- Streaming graphs:
  - Graphs changing at high rates.
  - 100s of thousands of updates per second.

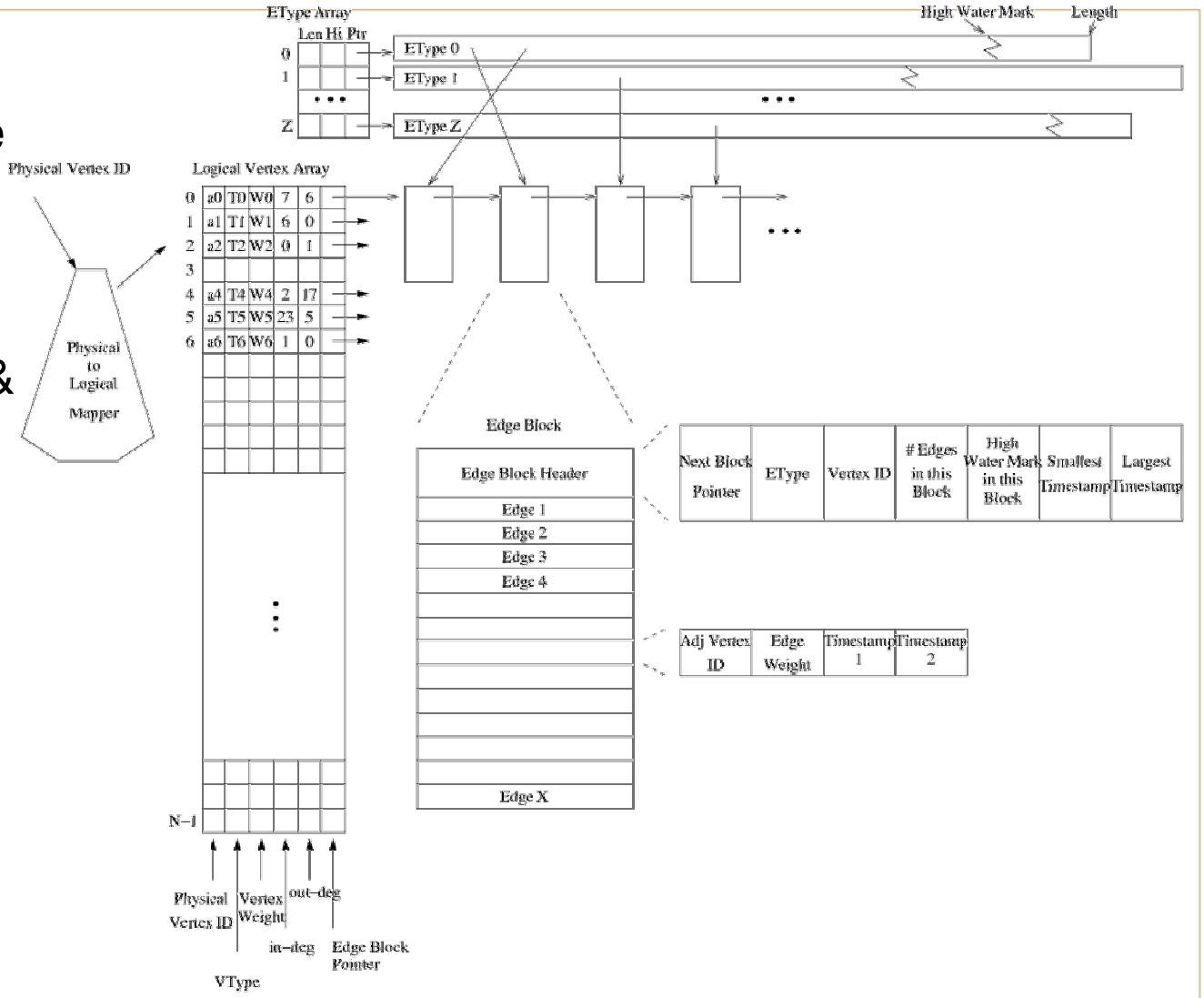
Streaming graph example

- Only a subset of the entire graph…
- Dynamic/Streaming:
  - At time t:
    - v and w become friends.
    - $insert\_edge\ (v, w)$
  - At time $\hat{t}$:
    - $u\ and\ v\ no\ longer\ friends$
    - $delete\_edge(u, v)$

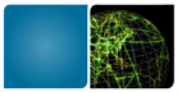


STING Extensible Representation

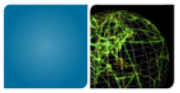
- Semi-dense edge list blocks with free space
- Supports property graphs (vertex & edge type, vertex & edge weights, time-stamps, and more).
- Maps from application IDs to storage IDs



STINGER


- Enable algorithm designers to implement dynamic & streaming graph algorithms with ease.
- Portable semantics for various platforms
  - Linked list of edge blocks not ideal for the GPU
- Good performance for all types of graph problems and algorithms - static and dynamic.
- Assumes globally addressable memory access

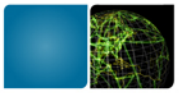




STINGER and cuSTINGER Properties

- ✓ A Simple programming model
- ✓ Millions of updates per second to graph
  - ✓ Updates are not bottlenecks for analytics.
  - ✓ Hundreds of thousands of updates per second for numerous analytics.W
- ✓ Advanced memory manager
  - ✓ Transfers data between host and device automatically
  - ✓ Reduces initialization time
  - ✓ Allows for simple update processes

Main Papers: [Bader *et al.*; 2007; Tech Report]

[Ediger *et al.*; HPEC; 2012], [McColl *et al.*; PPAA; 2014]


Lots of great graph libraries

CPU-based

- Galois
- Ligra
- LLAMA
- STINGER
  - DISTINGER

GPU-based

- Gunrock
- GasCL
- BelRed
- BlazeGraph

Most of these target STATIC graphs and use CSR

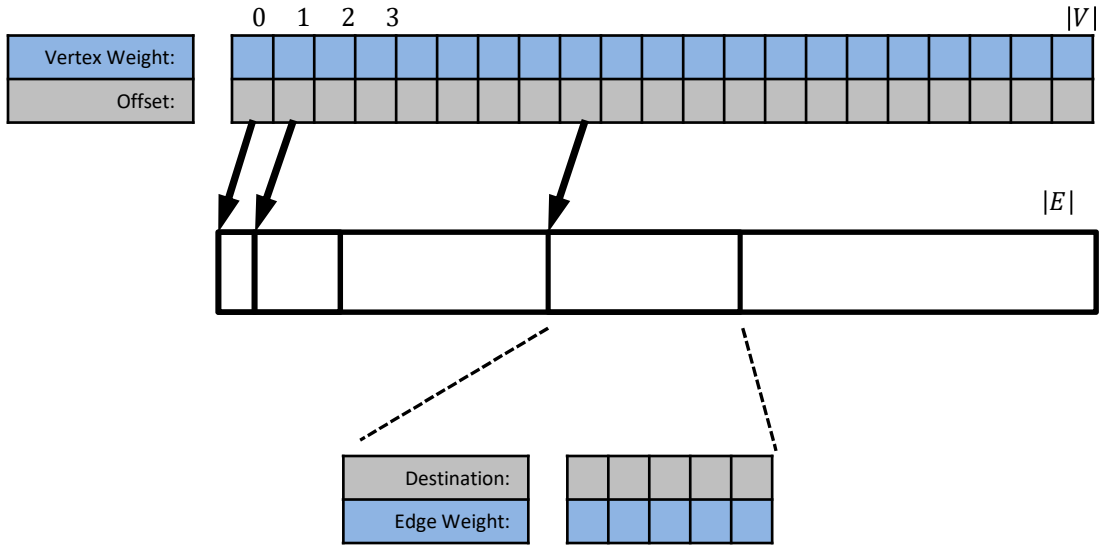
Compressed Sparse Row (CSR)

Pros:

- Uses precise storage requirements
- Great locality
  - Good for GPUs
- Handful of arrays
  - Simple to use and manage

Cons:

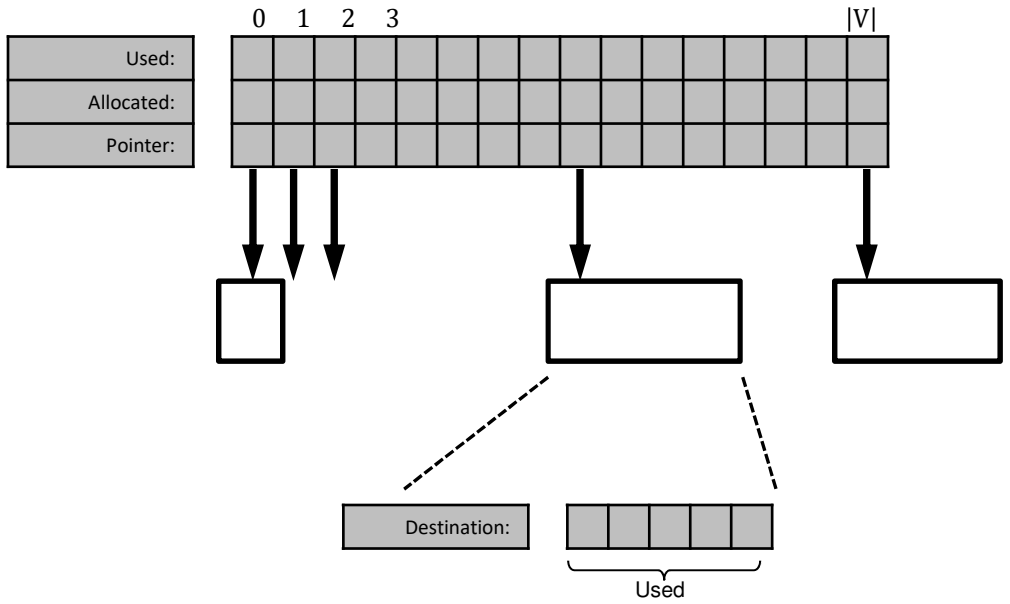
- Inflexible.
- Network growth unsupported
- Topology changes unsupported
- Property graphs not supported



Legend:	Optional Field	Mandatory Field

cuSTINGER – Data Structure

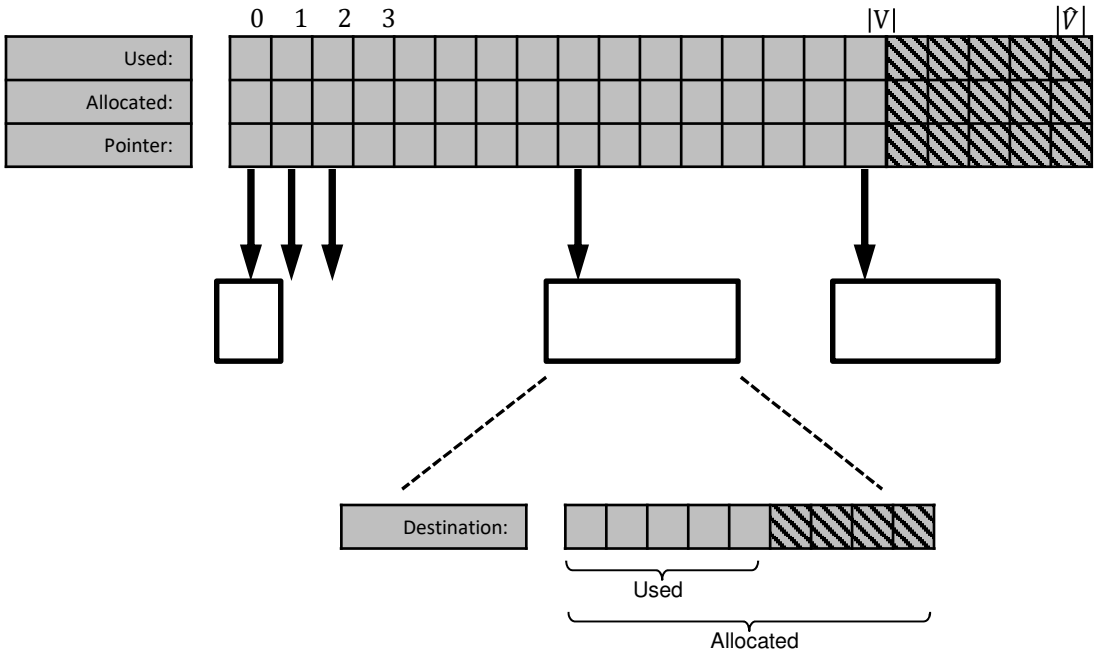
- Great locality
  - STINGER uses an Array of Structures (AOS)
  - cuSTINGER uses a Structure of Arrays (SOA)
- Each vertex has its own adjacency list
- Can compact data similar to CSR.



Legend:	Optional Field	Mandatory Field

cuSTINGER – Supports Growth

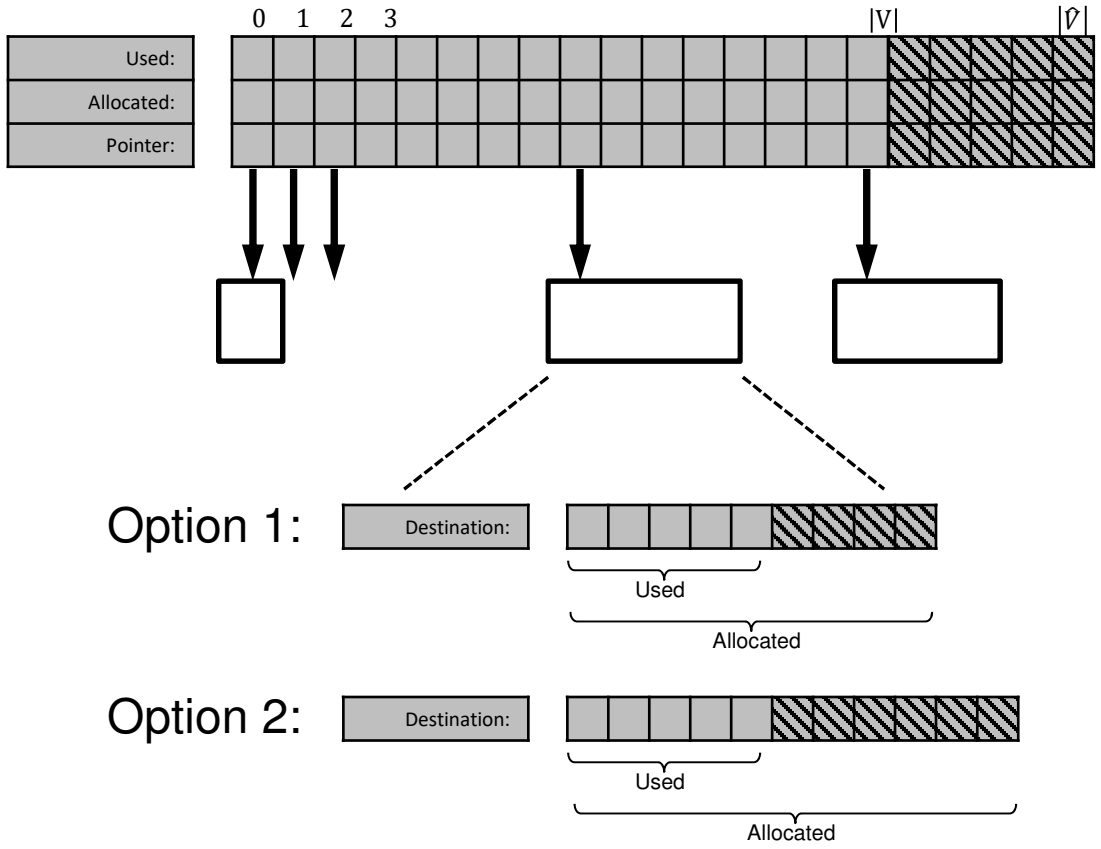
- Great locality
- Supports updates
  - Supports edge insertion and deletion
  - Supports vertex insertion and deletion



Legend:	Optional Field	Mandatory Field

cuSTINGER – Allocation modes

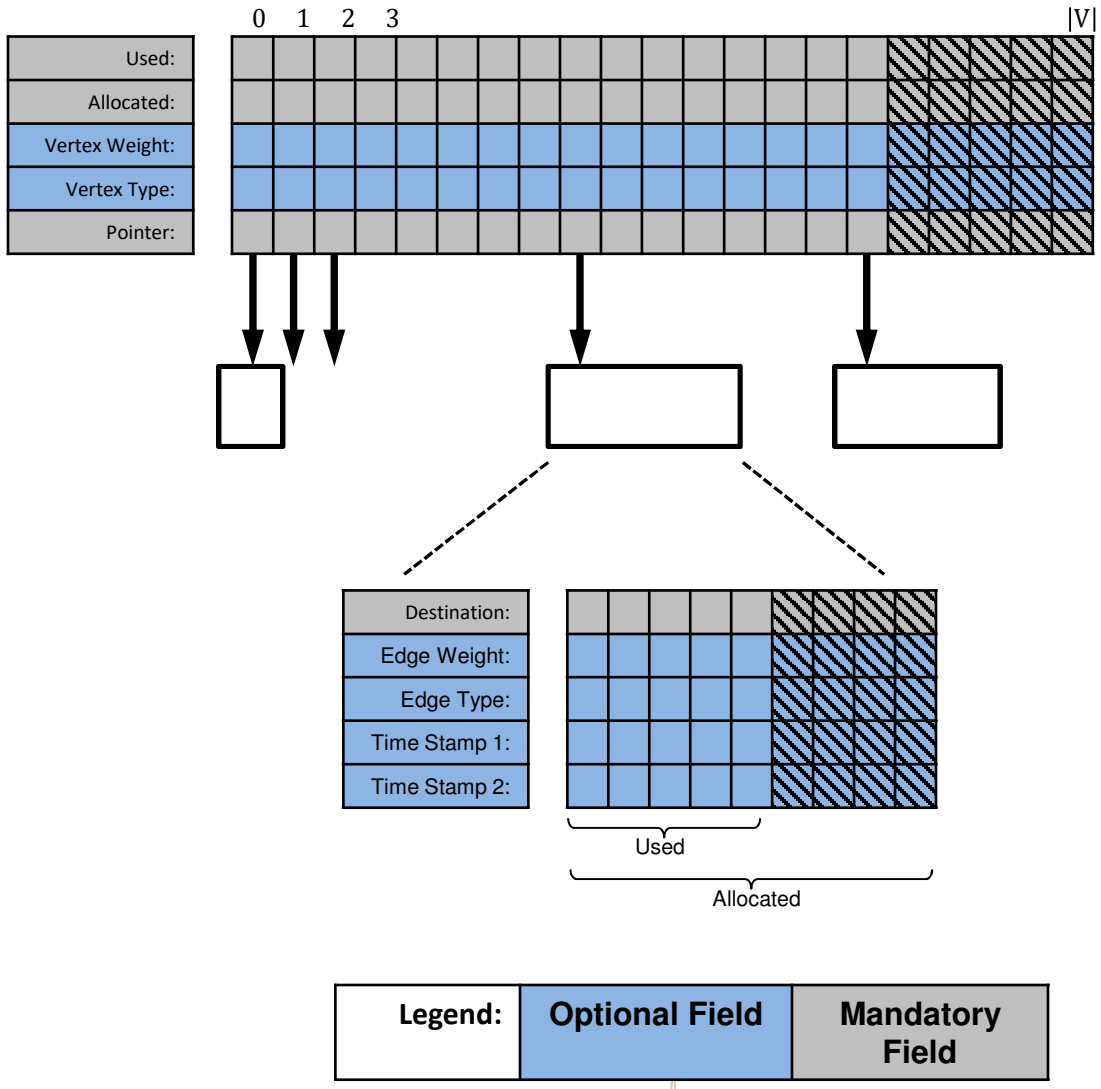
- Great locality
- Supports updates
  - Supports edge insertion and deletion
  - Supports vertex insertion and deletion
- Supports multiple allocation modes
  - Runtime configurable



Legend:	Optional Field	Mandatory Field

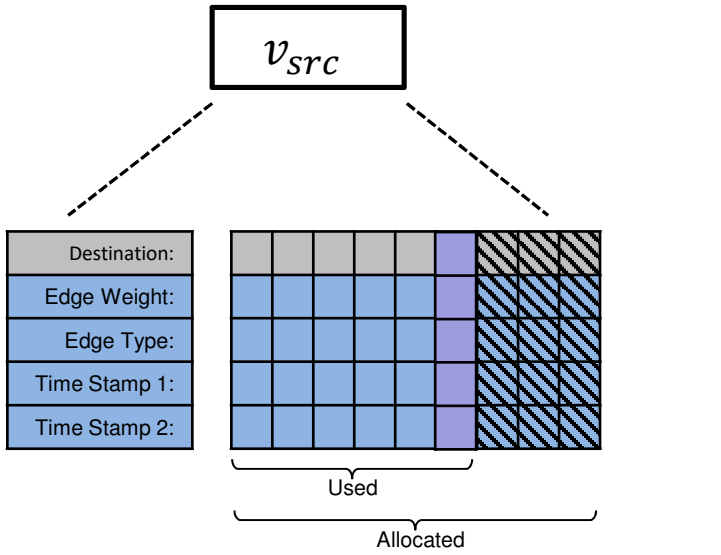
cuSTINGER – Supports Properties

- Great locality
- Supports updates
  - Supports edge insertion and deletion
  - Supports vertex insertion and deletion
- Supports multiple allocation modes
- Supports STINGER properties



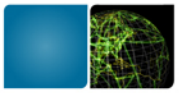
Edge Insertions

- Given an edge update, $e = (v_{src}, v_{dest})$:
  - Check that edge doesn't already exist
  - Check for available space
  - Increment "used" and append to end
  - Adjacency list is not sorted
- Updates are done in batches
  - Better utilization
  - Requires identifying two identical edges in a batch.

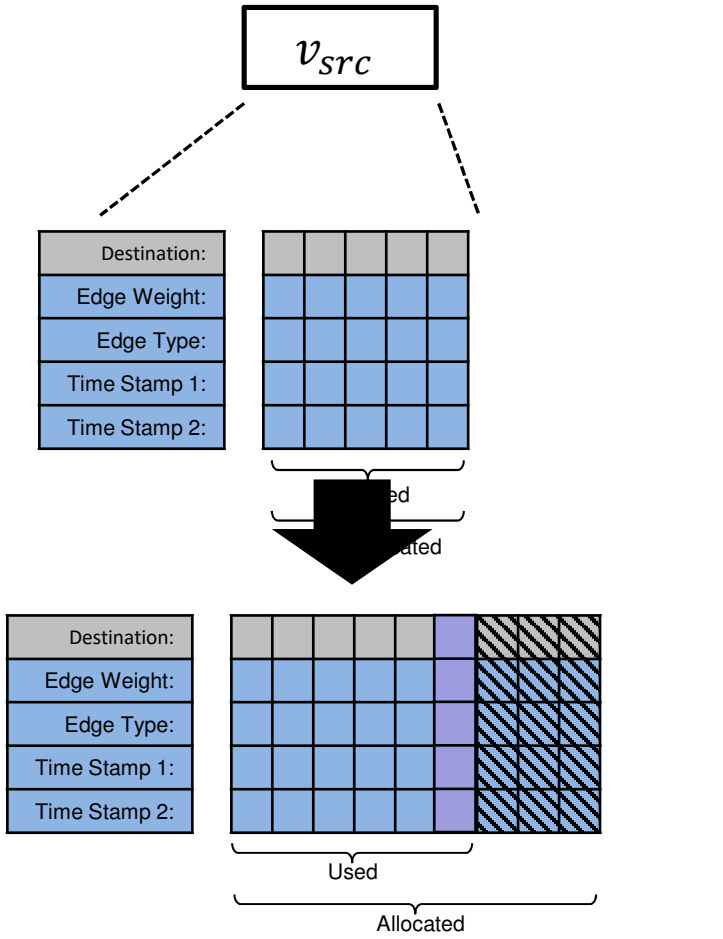


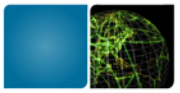
Legend:	Optional Field	Mandatory Field

Edge Insertions – Out of Memory

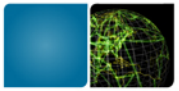

- Given an edge update, $e = (v_{src}, v_{dest})$
- Adjacency list is full
- Allocate new list
- Copy old list into new list
- Append to end




Experiment Setup

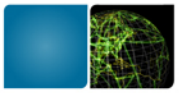
- NVIDIA K40 GPU
  - Kepler micro-architecture
  - 15 SMs, total of 2880 SPs
  - 12GB of RAM
- Intel i7-4770K
  - Haswell micro-architecture
  - Quad core
  - 8MB L3 cache
  - 32GB of RAM


Inputs Graphs

- DIMACS 10 Graph Implementation Challenge
- SNAP – Stanford Network Analysis Project

Name	Type	$\|V\|$	$\|E\|$	Source
$coAuthorsDBLP$	Collaboration	$299k$	$1.95M$	DIMACS
$as-skitter$	Trace route	$1.69M$	$11.1M$	SNAP
$kron\_21$	Random	$2M$	$201M$	DIMACS
$cit-patents$	Citation	$3.77M$	$16.5M$	SNAP
$cage15$	Matrix	$5.15M$	$94M$	DIMACS
$uk-2002$	Webcrawl	$18.52M$	$523M$	DIMACS

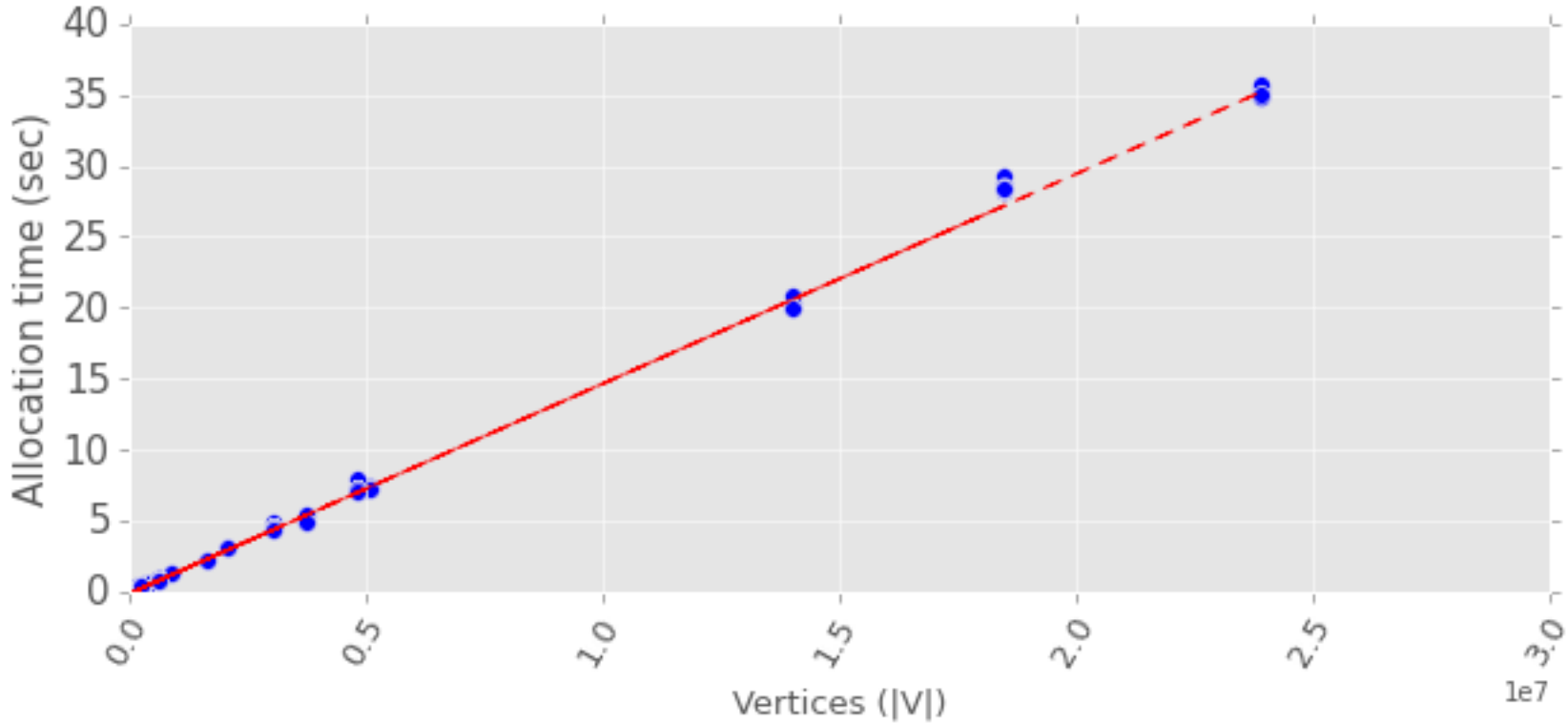




Experiment metrics

- Initialization time
  - Preferably as small as possible
- Update rate
  - Number of updates per second that cuSTINGER can sustain
- Static graph support
  - We compare a clustering-coefficient implementation using CSR with a CUSTINGER implementation

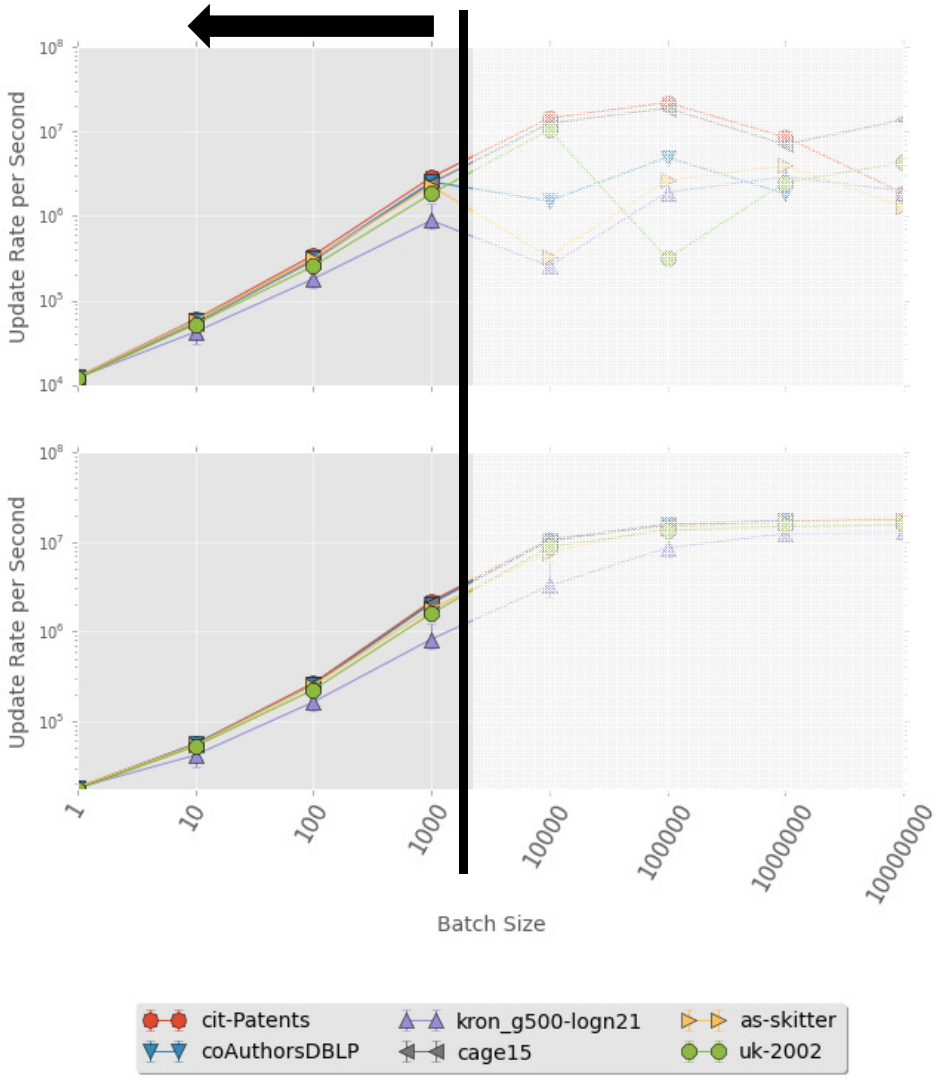
Initialization Time

- Time correlated with number of vertices



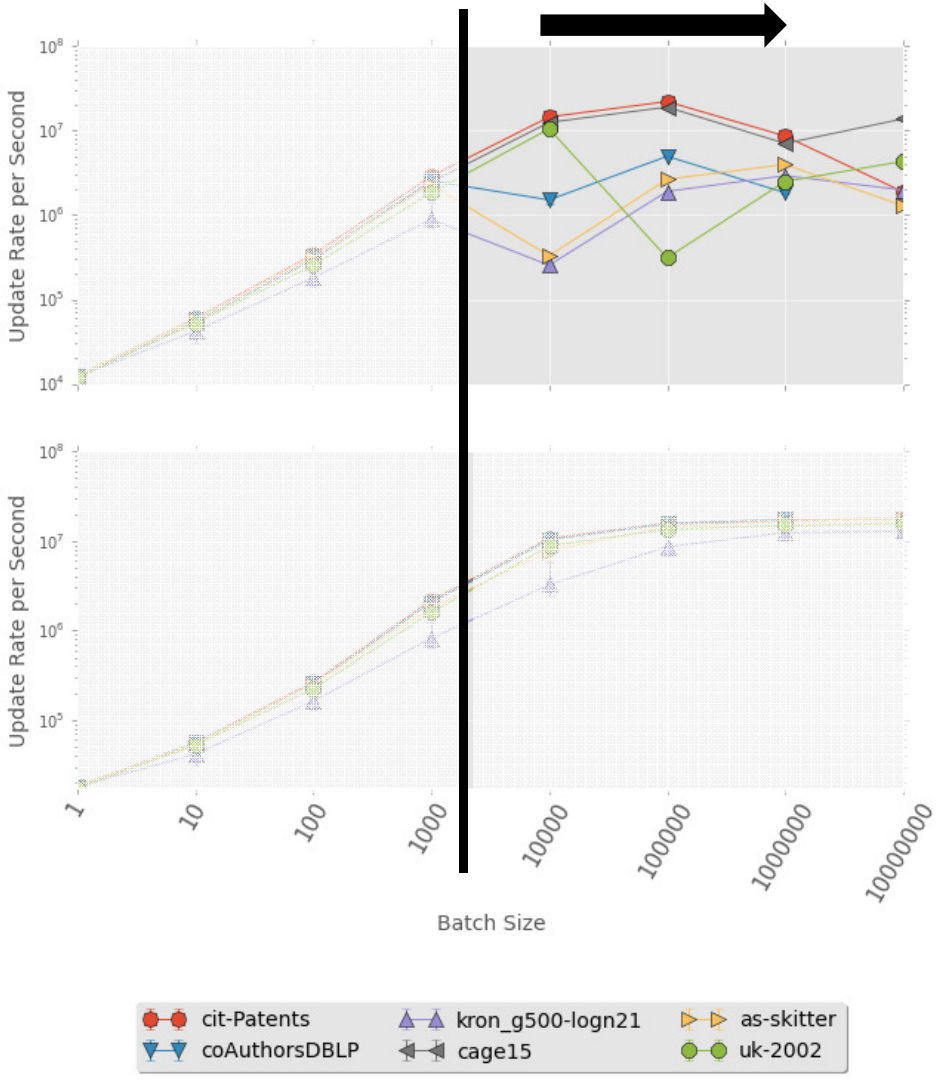
Update rate – Small Batches

- Updating a single edge at a time:
  - $15K$ updates per second
  - Same rate for insertions and deletions
- For small batches
  - Upto 1000 edges per batch
  - Millions of updates per second



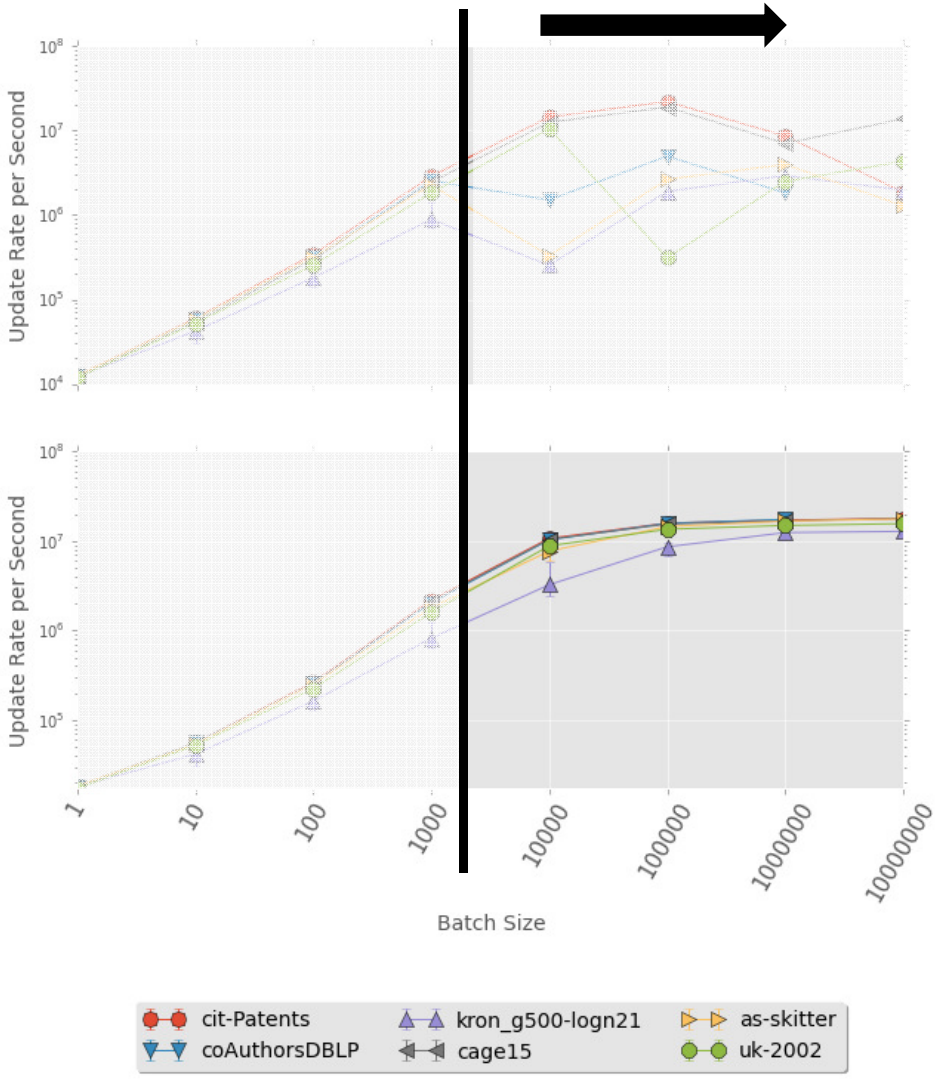
Insertion rate – Large Batches

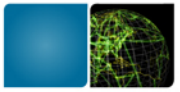
- Increase chance of vertex not having enough storage available.
- Some structures are copied back from device to host
  - Overhead is big for mid-size batches.
  - Overhead "$disappears$" for larger batches.



Deletion rate – Large Batches

- Performance is consistent for all graphs and unique batches.
- No memory allocation or de-allocation are required.
  - Unlike for the insertions case.
- Currently, memory reclamation is not supported.

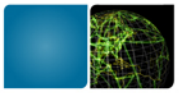



Triangle Counting – Static Graph

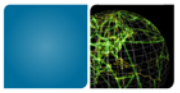
Name	$\|V\|$	$\|E\|$	Time-CSR (sec.)	Time-cuSTINGER (sec.)	Execution Difference
$coAuthorsDBLP$	$299k$	$1.95M$	0.218	0.242	+10%
$as-skitter$	$1.69M$	$11.1M$	57.14	59.37	+3.8%
$kron\_21$	$2M$	$201M$	2992	2996	+0.14%
$cit-patents$	$3.77M$	$16.5M$	0.814	0.830	+2%
$cage15$	$5.15M$	$94M$	6.544	7.204	+10%
$uk-2002$	$18.52M$	$523M$	424.9	431.4	+1.6%

- Algorithm taken from [Green et al; IA^3;2014]
- Simply replace CSR accesses with cuSTINGER
- Execution times are similar


Summary

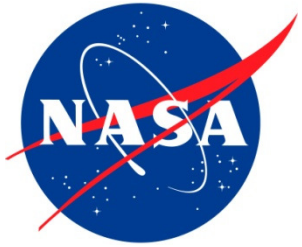
- cuSTINGER supports high update rates
- Memory manager
  - Responsible for allocating and transferring data on/from device
  - Reduces initialization time
  - Programmers can focus on algorithms instead of complex data management
- Great performance for both dynamic and static graph algorithms

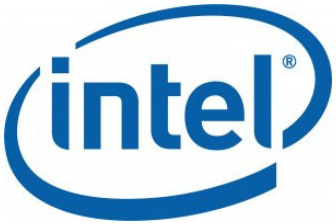



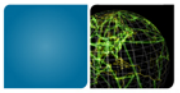

Acknowledgments

- Devavret Makkar, Graduate Student (Georgia Tech)

Acknowledgment of Support

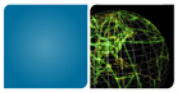





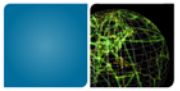

Thank you

- Email: ogreen@gatech.edu
- STINGER:
  - Documentation: http://stingergraph.com/
- cuSTINGER
  - Coming soon…

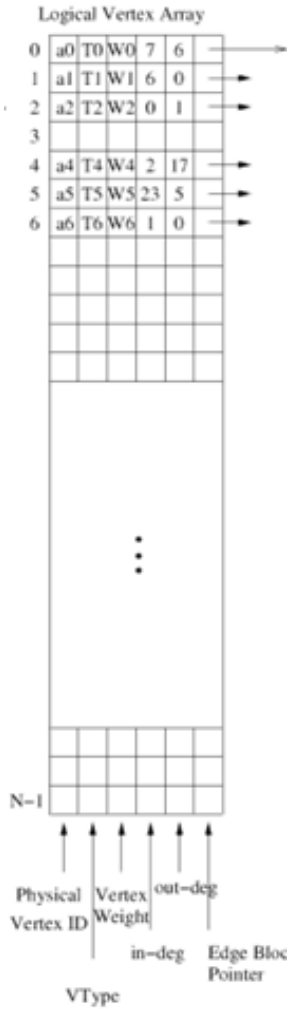



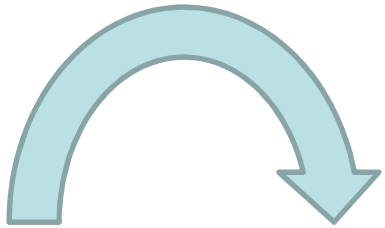

Backup Slides

Array of Structures Vs. Structure of Arrays


STINGER (AOS)





90° degrees

cuSTINGER (SOA)