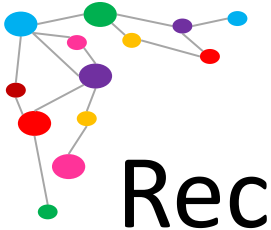


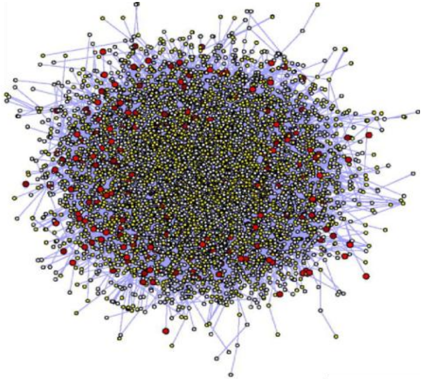
Recent Advances in Multi-GPU Graph Processing

G. Carbone[1], M. Bisson[2], M. Bernaschi[3], E. Mastrostefano[1], F. Vella[1]

[1]Sapienza University Rome - Italy

[2]NVIDIA U.S.

[3]National Research Council – Italy



March 2015

Why Graph Algorithms

- Analyze large networks
  - Evaluate structural properties of networks using common graph algorithms (BFS, BC, ST-CON, ...)
  - Large graphs require parallel computing architectures
- High performance graph algorithm:
  - Most of graph algorithms have low arithmetic intensity and irregular memory access patterns
  - How do GPU perform running such algorithms?
  - GPU main memory is currently limited to 12GB
  - For large datasets, cluster of GPUs are required

Large Graphs

- Large scale networks include hundred million of nodes
- Real-world large scale networks feature a power law degree distribution and/or small diameter

	# Vertices	# Edges	Diameter
wiki-Talk	2.39E+06	5.02E+06	9
com-Orkut	3.07E+06	1.17E+08	9
com-LiveJournal	4.00E+06	3.47E+07	17
soc-LiveJournal1	4.85E+06	6.90E+07	16
com-Friendster	6.56E+07	1.81E+09	32

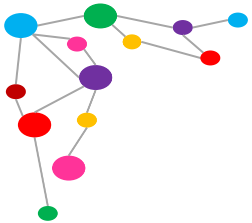
SNAP Source: Stanford Large Network Dataset Collection

Distributed Breadth First Search

- Developed according to the Graph 500 specifications
  - Generate edge list using RMAT generator
  - Support up to SCALE 40 and Edge Factor 16 (where $|V| = 2^{SCALE}$ and $|M| = 16 \times 2^{SCALE}$)
  - Use 64 bits for vertex representation
- Performance metric: Traversed Edges Per Second (TEPS)
- Implementation for GPU clusters
- Hybrid Programming paradigm: CUDA + Message Passing (MPI and APEnet)
- Level Synchronous Parallel BFS
- Data structure divided in subsets and distributed over computational nodes

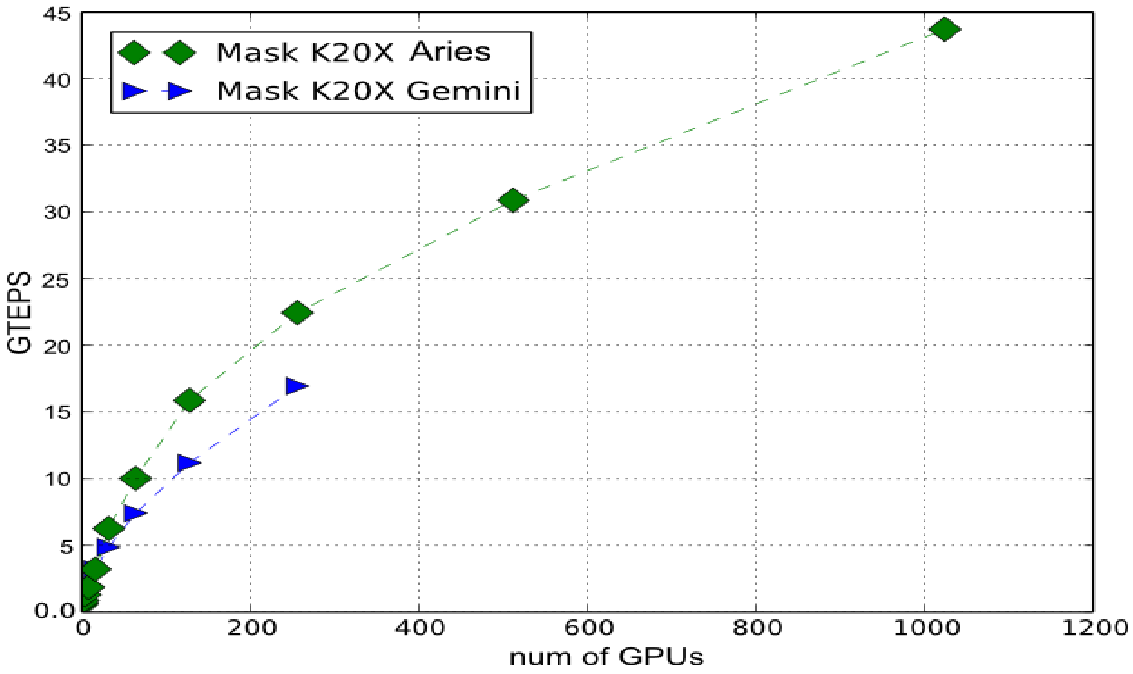
1-D BFS

- 1-D Graph Partitioning
- Balanced thread workload
  - Map threads to data by using scan and search operations
- Enqueue vertices only once (avoiding duplicates)
  - Local mask array to mark both local and connected vertices
- Reduce message size
  - Communication pattern to exchange predecessor vertices only when BFS is completed avoiding sending them at each BFS level
  - Use 32 bits representation to exchange vertices instead of 64 bits



1-D Results

Weak Scaling Plot (RMAT Graph SCALE 21 – 31)



2-D BFS

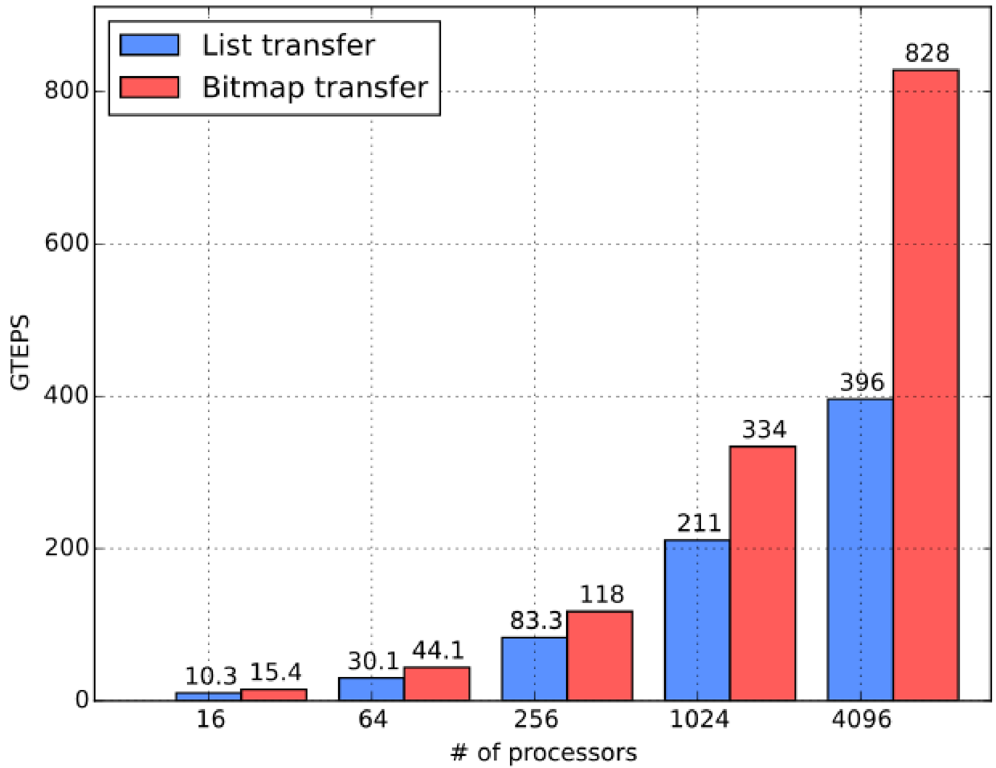
- 2-D Graph partitioning
  - Improved scalability avoiding all-to-all communications

- Atomic Operations
  - Local computation leverages efficient atomic operations on Kepler
  - 2.3x improvement from S2050 (Fermi) to K20X (Kepler) on single GPU

- Further reduction of message size
  - Use a bitmap to exchange vertices among nodes

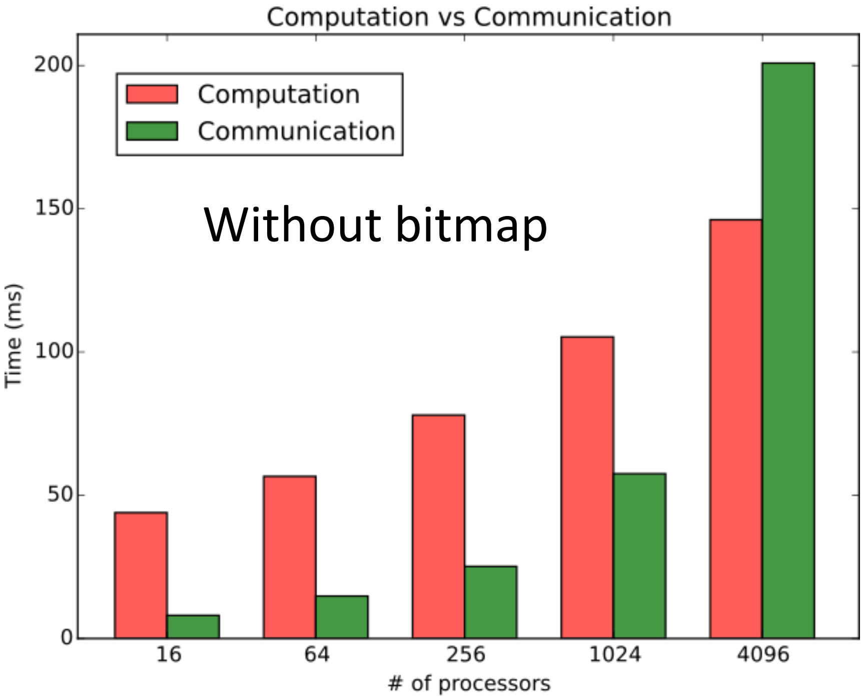
2-D Results

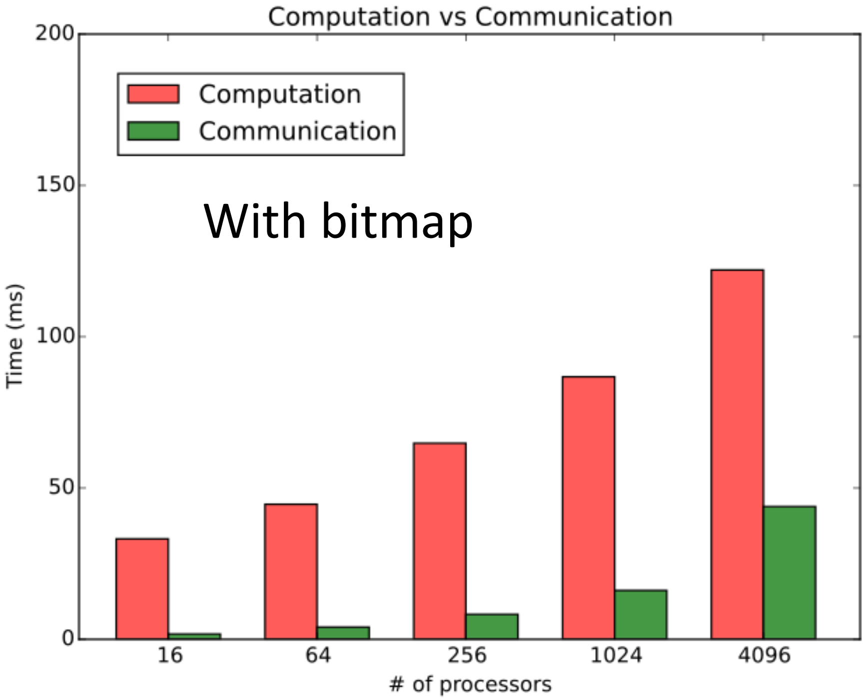
Weak Scaling Plot (RMAT Graph SCALE 21 – 33)



2-D BFS Bitmap based transfer

Use bitmap to exchange vertices information





2D BFS Results on Real Graph*

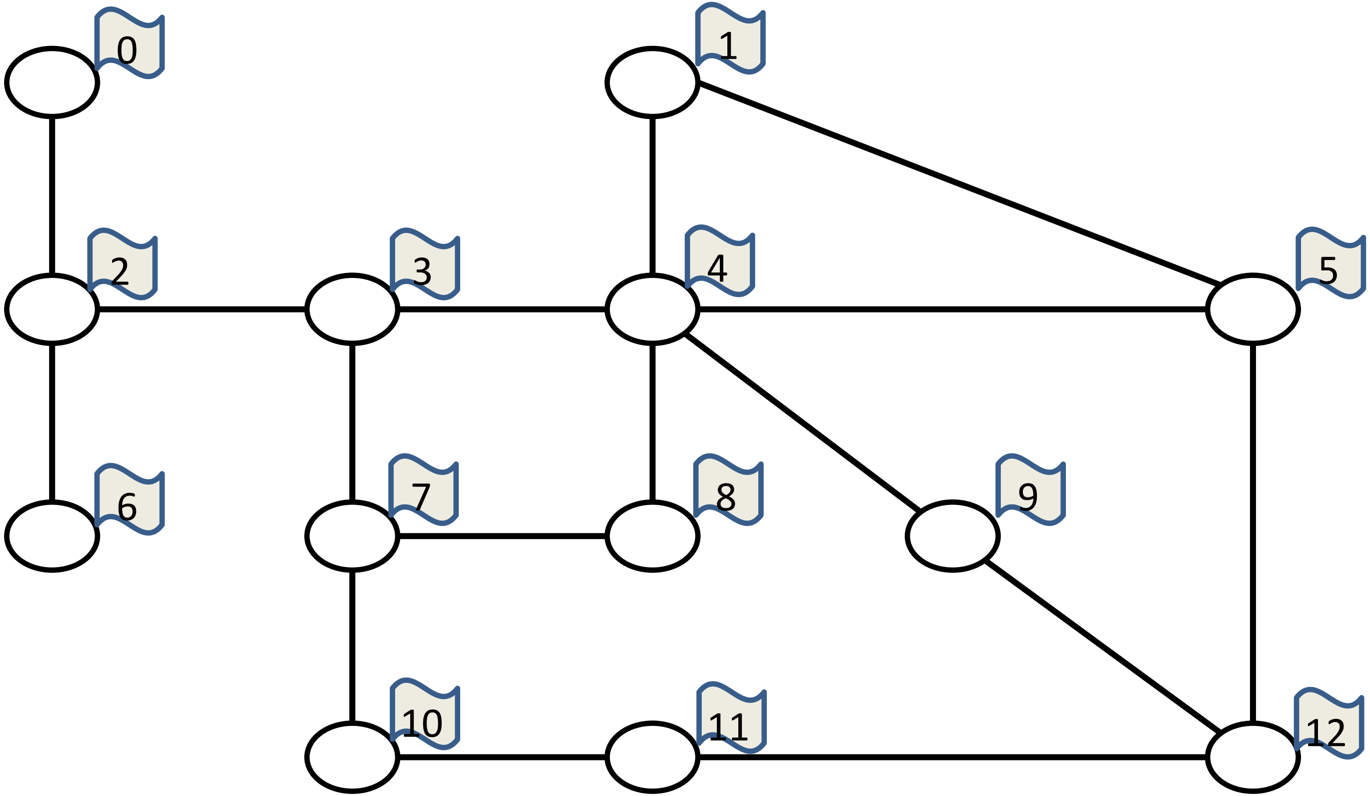
Data Set Name	Vertices	Edges	Scale	EF	# GPUs	GTEPS	BFS Levels
com-LiveJournal	4.00E+06	3.47E+07	22	9	2	0.77	14
soc-LiveJournal1	4.85E+06	6.90E+07	22	14	2	1.25	13
com-Orkut	3.07E+06	1.17E+08	22	38	4	2.67	8
com-Friendster	6.56E+07	1.81E+09	25	27	64	15.68	24

*Source: Stanford Large Network Dataset Collection

ST-CON

- Decision problem
  - Given source vertex ***s*** and destination vertex ***t*** determine if they are connected
  - Output the shortest path if one exists
- Straightforward solution by using BFS
  - Start a BFS from ***s*** and terminate if ***t*** is reached
- Parallel ST-CON
  - Start two BFS in parallel from ***s*** and ***t***
  - Terminate if the two paths meet

Parallel ST-CON

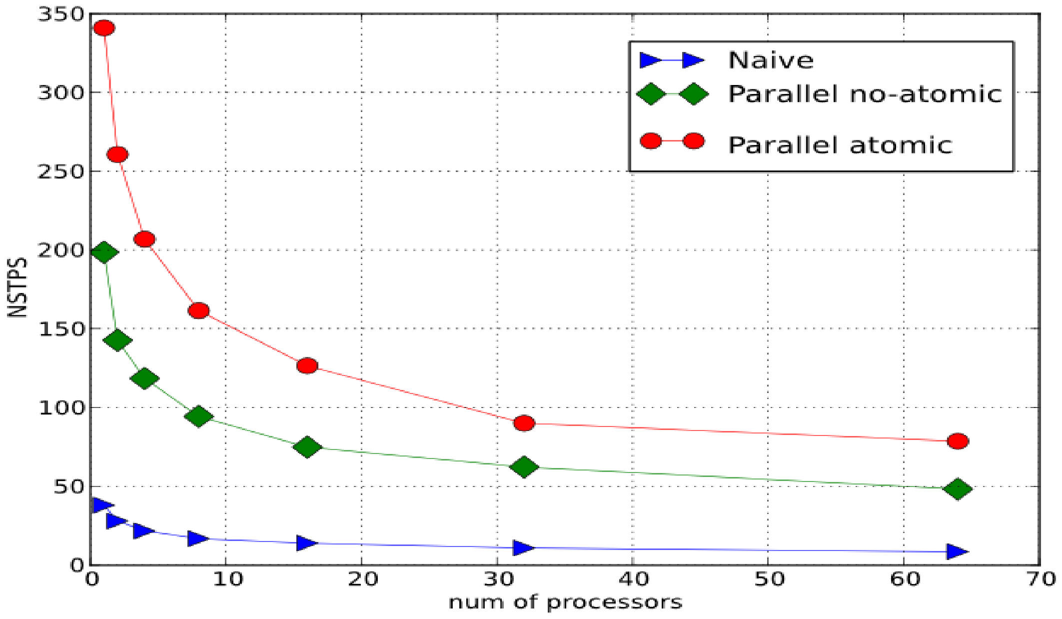


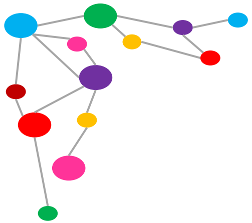
Distributed ST-CON

- Atomic-operations based solution
  - Use atomic operations to update visited vertices
  - Finds only one ***s-t*** path
- Data structure duplication solution
  - Use distinct data structures to track ***s*** and ***t*** paths
  - At each BFS level check if there are vertices visited by both
  - Finds all ***s-t*** paths
- Performance metric
  - Number of ***s-t*** Pairs Per Second (NSTPS)
  - Execute ST-CON algorithm over a set of ***s-t*** pairs randomly selected

ST-CON Results

Weak Scaling Plot (RMAT Graph SCALE 21 – 27)

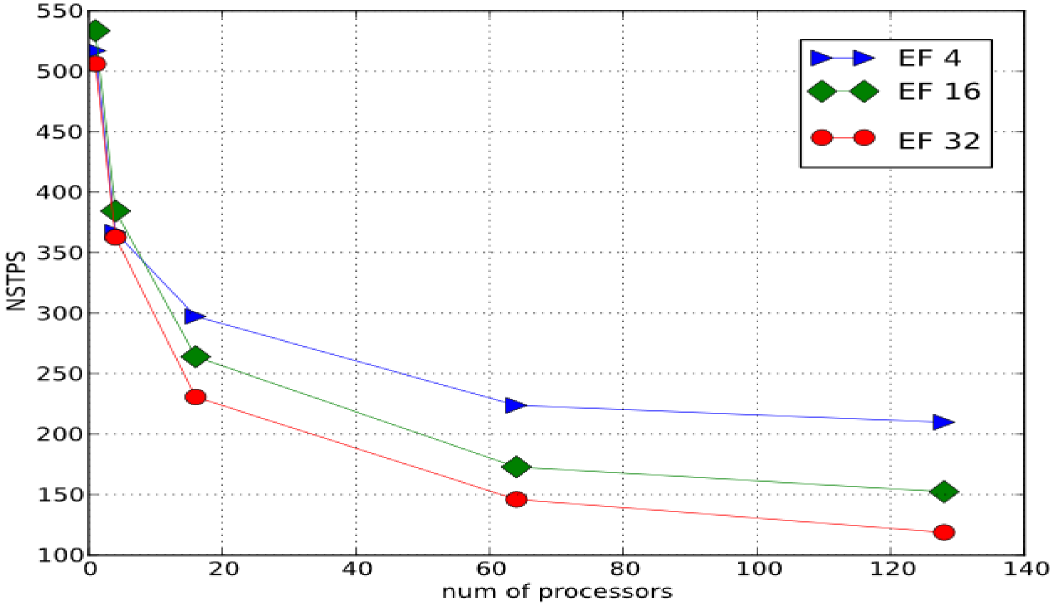




ST-CON Results

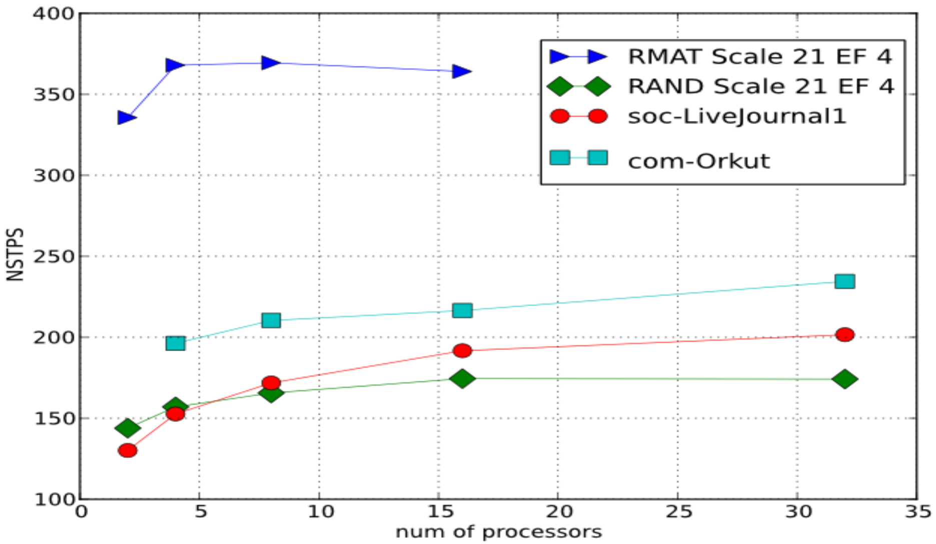
Weak Scaling Plot (RMAT Graph SCALE 19 – 26)

Only Parallel Atomic with different Edge Factor



ST-CON Results

Strong Scaling Plot (Parallel Atomic)



Bernaschi, M., Carbone, G., Mastrostefano, E., & Vella, F.
Solutions to the st-connectivity problem using a GPU-based distributed BFS.
Journal of Parallel and Distributed Computing, Volume 76, *Pages 145-153* February 2015

Betweenness Centrality

Misure of the influence of a node in a given network used in network analysis, transportation networks, clustering, etc.

$$BC(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

- σ_{st} is the number of shortest paths from **s** to **t**
- $\sigma_{st}(v)$ is the number of shortest paths from **s** to **t** passing through **v**

- Best known sequential algorithm requires O(*mn*) time-complexity and O(n+m) space-complexity (Brandes2001)
- No satisfactory performance for large-scale graphs (biology systems and social networks)

Distributed BC

- Parallel distributed based on Brandes algorithm
  - 2D BFS as building block
  - Distributed dependency accumulation

Dependency is:

$$\delta_s(v) = \sum_{w \in Succ(v)} \frac{\sigma_{sv}}{\sigma_{sw}} (1 + \delta_s(w))$$

BC scores become:

$$BC(v) = \sum_{s \neq v} \delta_s(v)$$

A R-MAT graph with 2M nodes and ≈ 32M Edges requires about 20 hours on 4 K40 GPUs !!

Conclusions

- Best algorithm has still O(*mn*) complexity

- Reduce *n*
  - 1-degree reduction (≈ 15% on R-MAT) Saríyüce2013, Baglioni2012
  - 2-degree reduction (≈ 8% on R-MAT)
  - Further heuristics to reduce the size of the graph to be analyzed

- Improve parallelism
  - Multi-source BFS

Thank You!

giancarlo.carbone@uniroma1.it

Please complete the Presenter Evaluation sent to you by email or through the GTC Mobile App. Your feedback is important!