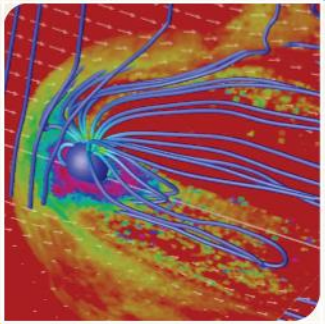
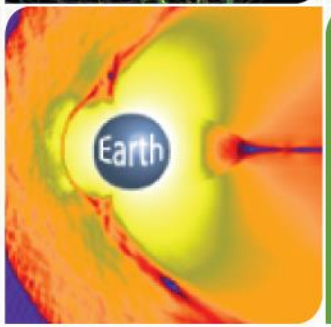
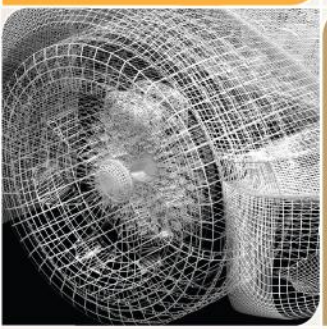




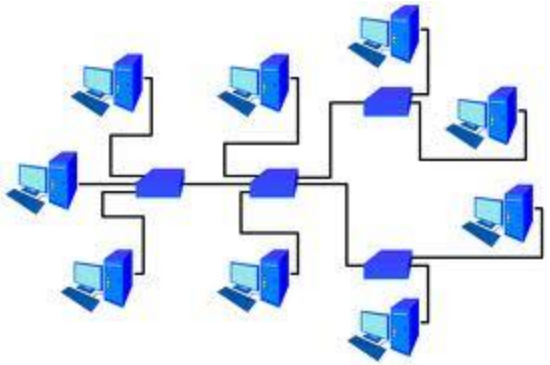
Fast Execution of Simultaneous Breadth-First Searches on Sparse Graphs

Adam McLaughlin and David A. Bader

Georgia Tech College of Computing
Computational Science and Engineering

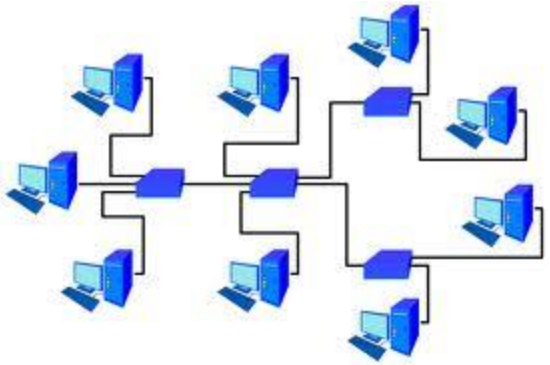
Applications of interest…

- Computational biology
- Social network analysis
- Urban planning
- Epidemiology
- Hardware verification



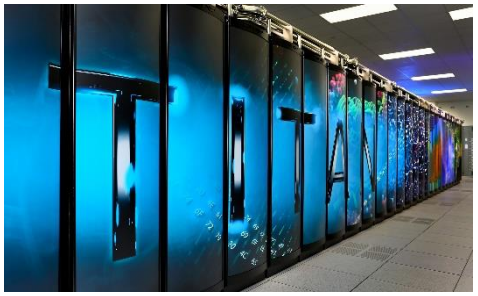
Applications of interest…

- Computational biology
- Social network analysis
- Urban planning
- Epidemiology
- Hardware verification

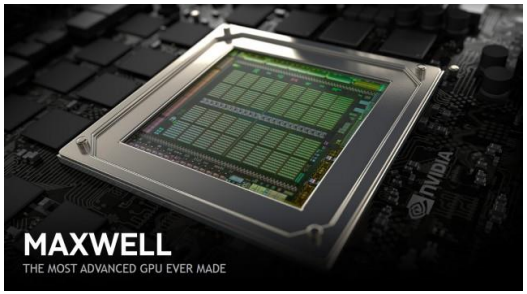


- Common denominator: **Graph Analysis**

“Traditional” HPC is Expensive



- Tianhe-2: 17.8 MW
- Titan: 8.2 MW
- Distributed systems are often overkill
  - Too much time and energy wasted on expensive communication
  - Shared memory is large enough (~1 TB)
- Leverage the high memory bandwidth of NVIDIA GPUs



GPUs are Challenging to Program

- Months of domain expert programmer time required to develop/optimize code
- Efforts are typically limited to a single problem, architecture, or data set
  - Little code reuse
  - Limited number of libraries
  - Opaque, yet drastic, performance consequences



What makes GPU Computing so Difficult?

- Parallel programming challenges
  - Deadlock, synchronization, race conditions
- Architectural/Ecosystem challenges
  - Programmer managed shared memory
  - Deep knowledge of the underlying architecture required
- Challenges unique to graph analysis
  - Data dependent memory access patterns

Solution: Abstraction

- Abstract details of parallel programming from end users
- Let social scientists, analysts, etc. focus on gathering insights
- Let domain experts focus on parallel programming, architectural details
  - Encourage modularity and code reuse

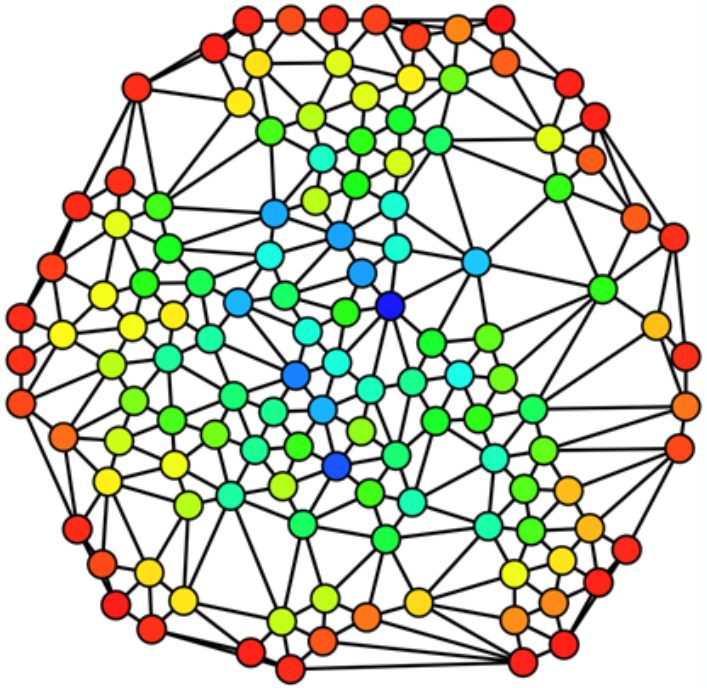
Related Work

- Abstractions for graph analysis
  - User applies code that operates on active vertices and provides the next frontier of vertices
  - **Galois** [Nguyen *et al.* SOSP '13]
  - **Ligra** [Shun *et al.* PPoPP '13]
  - **Gunrock** [Wang *et al.* PPoPP '16]
- "Hard-wired" implementations
  - BFS [Merrill *et al.* PPoPP '12]
  - `hybrid_BC` [McLaughlin and Bader SC '14]
  - SSSP [Davidson *et al.* IPDPS '14]

The Multi-Search Abstraction

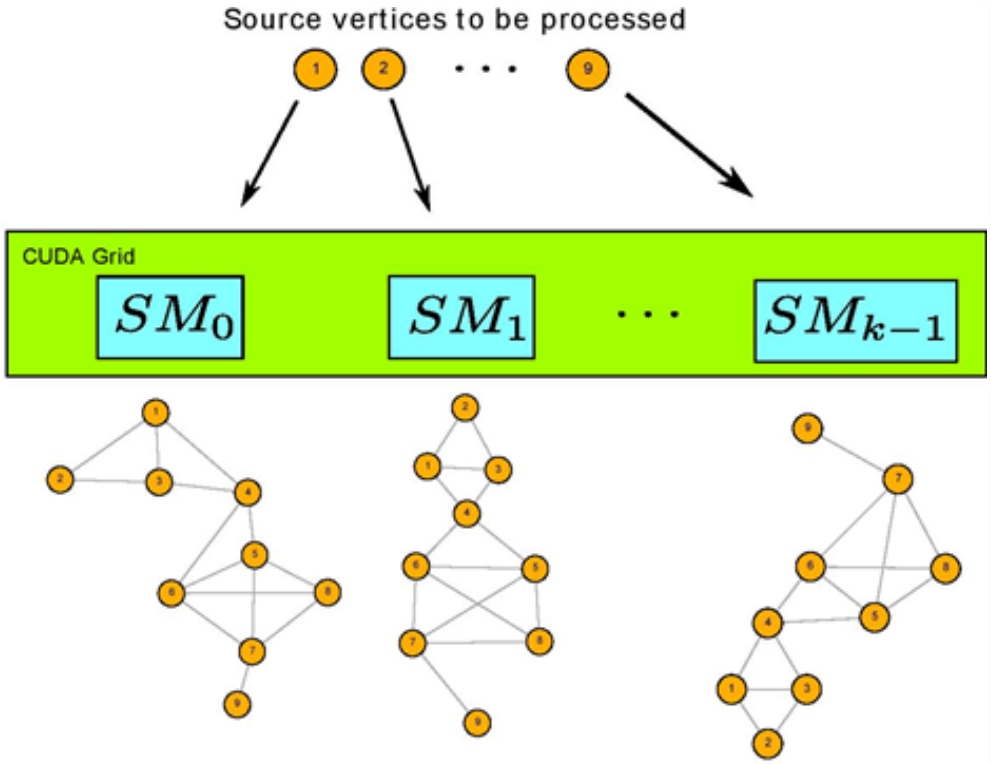
- Fits any problem requiring the simultaneous execution of many breath-first searches

1. All-Pairs Shortest Paths
2. Diameter Computations
3. Transitive Closures
4. **Betweenness Centrality**



What makes this abstraction different?

- Traversal based, but utilizes *coarse-grained parallelism*
  - Prior abstractions parallelize within the context of a single BFS
  - Our abstraction parallelizes *across* BFSs



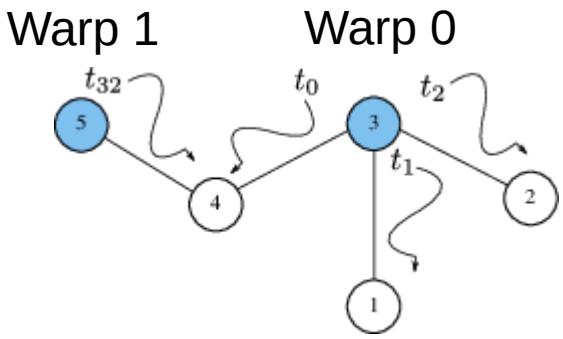
Multi-Search: APSP Example

- Users need to implement a small number of short functions

```
void init(int s)
{
        for(int k=0; k<n; k++) //For each vertex
        {
                if(k == s) d[s][k] = 0;
                else d[s][k] = INT_MAX;
        }
}
void visitVertex(int s, int u, int v, queue Q)
{
        if(d[s][v] == INT_MAX)
        {
                d[s][v] = d[s][u] + 1;
                Q.atomic_enqueue(v);
        }
}
```

Multi-Search: Visiting vertices

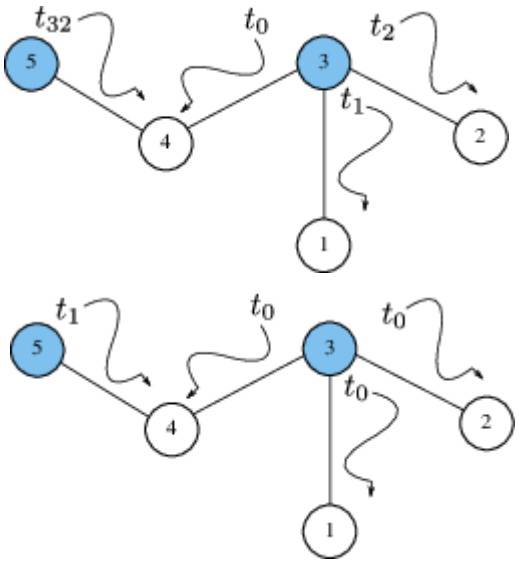
- Use a cooperative, Warp-*based* approach
- Warps concurrently expand adjacency lists of enqueued vertices (1 warp = 32 threads)
- Works great for vertices with high outdegree
  - Coalesced accesses to neighbor lists
- Underutilization for vertices with low outdegree



Multi-Search: Hierarchical Queues

- To resolve this underutilization, we can assign a thread to each enqueued vertex

- Use a thresholding approach
  - Outdegree(v) >= T → Warp processing
  - Outdegree(v) < T → Thread processing



Experimental Setup

- NVIDIA GTX Titan
  - Compute capability 3.5 ("Kepler") GPU
  - Peak theoretical memory bandwidth: 288.4 GB/s
  - 14 SMs, 6GB memory, 837MHz
- Galois/Ligra run on a quad-core CPU
  - Intel Core i7-2600K, 3.4 GHz, 8MB LLC

Benchmark Data Sets

Graph	Nodes	Edges	Notes/Sparsity
333SP	3.71m	22.22m	Ferrari 
adapative	6.82m	27.25m	Urban Sim. 
as-Skitter	1.70m	22.19m	Internet 
auto	0.45m	6.63m	Partitioning 
delaunay_n21	2.10m	12.58m	Triangulation 
ecology1	1.00m	4.00m	Gene Flow 
hollywood-2009	1.14m	115.03m	Movie Actors 
kron_g500-logn19	0.52m	43.56m	Kronecker 
ldoor	0.95m	45.57m	Large Door 
roadNet-CA	1.96m	5.53m	Intersections 
rgg_n_2_21_s0	2.10m	28.98m	Geometric 
thermal2	1.23m	7.35m	Diffusion 



Timing Results: Betweenness Centrality

Framework	*333SP*	*adaptive*	*as-Skitter*	*auto*	*delaunay_n21*	*ecology1*
Galois	4651	7086	1167	637	2004	906
Ligra	3005	3442	1241	665	992	635
Gunrock	1999	4851	N/A	161	712	1458
hybrid_BC	781	993	518	407	373	176
Cooperative	**352**	**601**	**275**	**74**	**174**	**104**

Framework	*hollywood-2009*	*kron_g500-logn19*	*ldoor*	*roadNet-CA*	*rgg_n_2_21_s0*	*thermal2*
Galois	2058	1868	1240	1498	3518	1088
Ligra	4318	623	1751	700	2808	899
Gunrock	630	**406**	395	N/A	N/A	277
hybrid_BC	1591	522	621	403	1066	204
Cooperative	**602**	523	**183**	**145**	**399**	**115**

- Using $k = 8192$ source vertices
- Cooperative is best on 11/12 graphs
- Cooperative is faster & more general than `hybrid_BC`

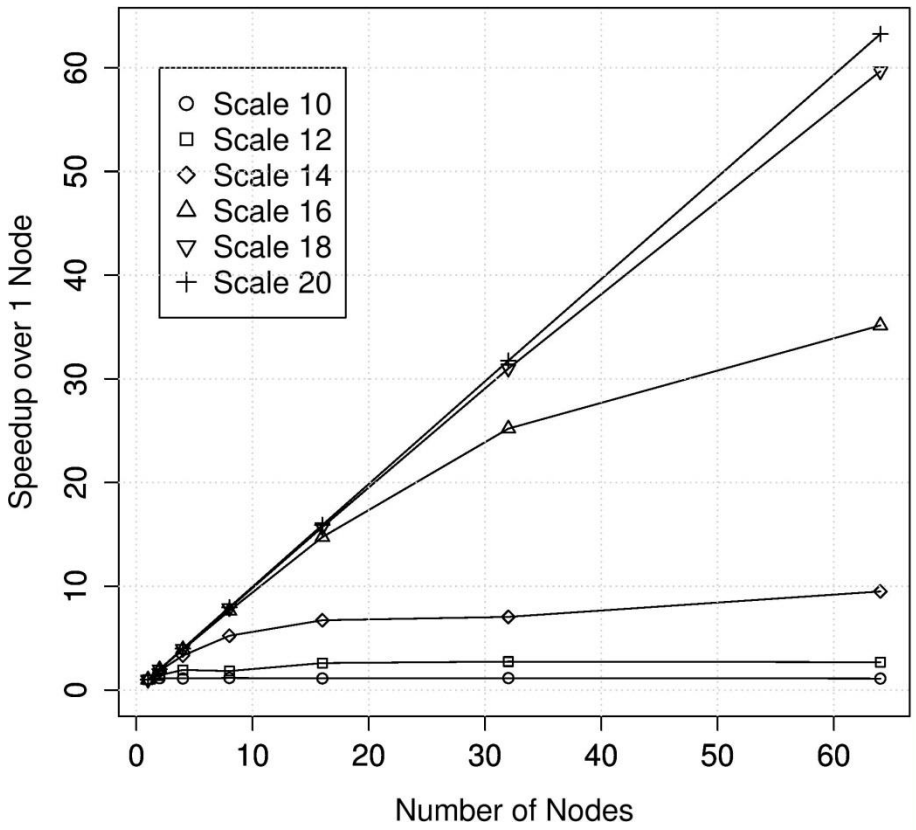
Summary: Betweenness Centrality

	Galois	Ligra	Gunrock	hybrid_BC
Speedup of Cooperative	**7.66x**	**5.82x**	**3.07x**	**2.24x**

- Average speedup over entire graph suite

Multi-GPU Results (BC)

- Linear speedups when graphs are sufficiently large
- 10+ GTEPS for 192 GPUs
- Scaling isn't unique to graph structure
  - Abundant coarse-grained parallelism



Keeneland Initial Delivery System (KIDS)
3 Tesla M2090 GPUs (Fermi) per node
Infiniband QDR Network



Conclusions

- There is **no "one size fits all" solution** for parallel graph algorithms
  - Graph structure is pivotal to performance
- **Abstraction is paramount** for high-performance, reusable applications
  - Prior methods of abstraction miss out on coarse-grained parallelism
  - Easily scales to many GPUs
- If the **distribution of parallelism** changes over time, the **method of parallelism** should change too

Acknowledgment of Support

- Thanks to DARPA and NVIDIA for their support of this work!

Questions

"To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science."– Albert Einstein

https://github.com/Adam27X/graph-utils

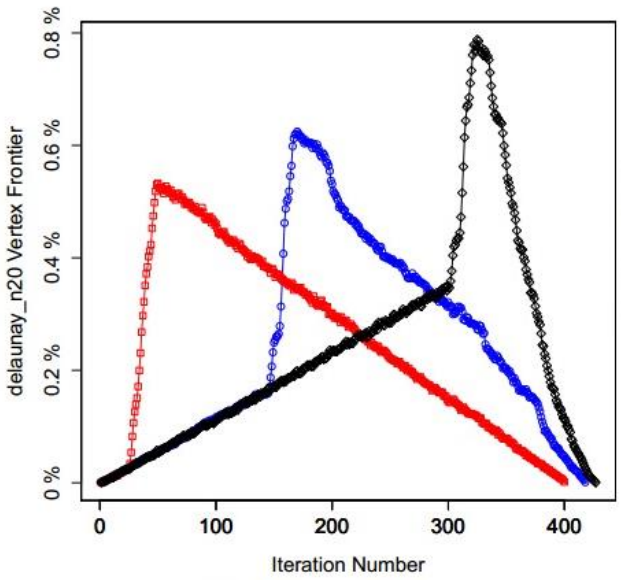
http://users.ece.gatech.edu/~amclaughlin7/research.html



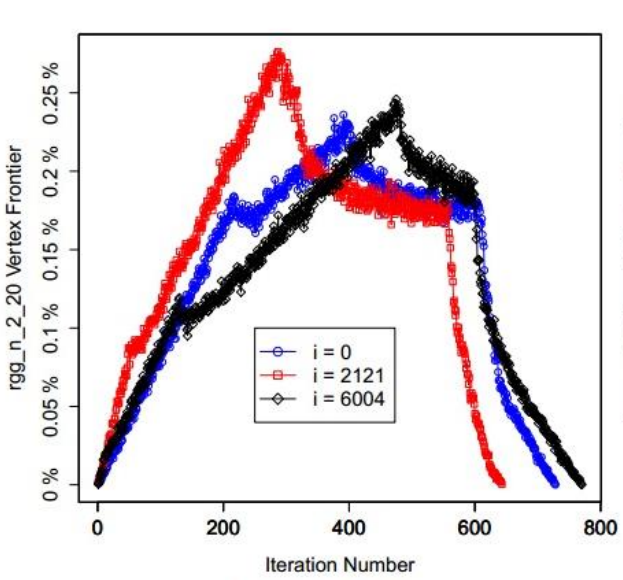
Backup

Motivation for Hybrid Methods

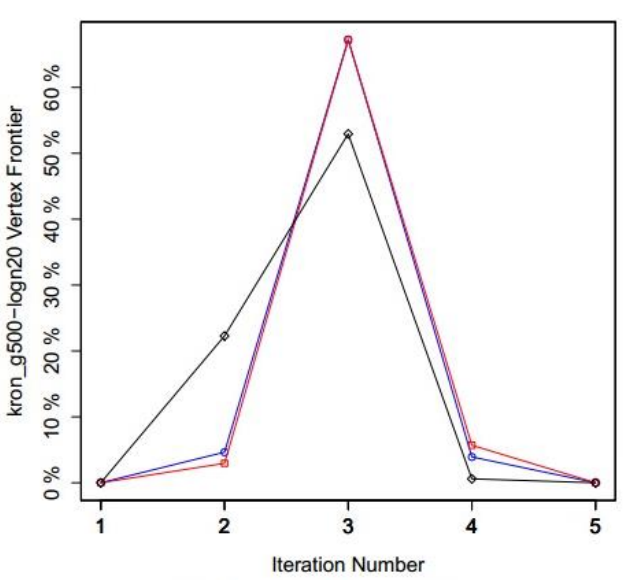
- No one method of parallelization works best



(a) delaunay_n20



(b) rgg_n_2_20



(c) kron_g500-logn20

- High diameter: Only do useful work
- Low diameter: Leverage memory bandwidth

Effect of Thresholding

- T = 0: Warp
- T = ∞: Thread
- Too small: Warp occupancy suffers
- Too large: severe workload imbalances among threads
- T = 16 (Half-warp)