GPU-Accelerated Network Centrality

Erik Saule
collaborative work with:
Ahmet Erdem Sarıyüce (OSU), Kamer Kaya (Sabancı), Ümit V. Çatalyürek (OSU)

University of North Carolina at Charlotte (CS)

GTC 2015

Outline

Centralities - Concept

Answer questions such as

- Who controls the flow in a network?
- Who is more important?
- Who has more influence?
- Whose contribution is significant for connections?

Different kinds of graph

- road networks
- social networks
- power grids
- mechanical mesh

Applications

- Covert network (e.g., terrorist identification)
- Contingency analysis (e.g., weakness/robustness of networks)
- Viral marketing (e.g., who will spread the word best)
- Traffic analysis
- Store locations

Centrality Formally

Closeness Centrality

Let $G = (V, E)$ be an unweighted graph with the vertex set V and edge set E.
$cc[v] = \sum_{u \in V} \frac{1}{d(v,u)}$ where $d(u, v)$ is the shortest path length between u and v.

Betweenness Centrality

Let $G = (V, E)$ be an unweighted graph. Let σ_{st} be the number of shortest paths connecting s amd t. Let $\sigma_{st}(v)$ be the number of such s-t paths passing through v.
$\mathrm{bc}[v] = \sum_{s \neq v \neq t \in V} \delta_{st}(v)$ where
$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$.

Algorithm

In each case, the best algorithm computes the shortest path graph rooted in each vertex of the graph and extract the relevant information. The complexity is $O(E)$ per source, $O(VE)$ in total, which makes its computationally expensive.

Computing Breadth First Traversal (Centrality)

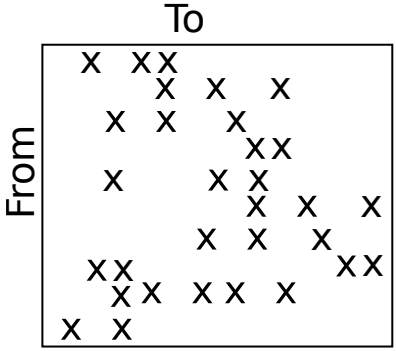
Top-down (scatter writes)

For each element of the frontier, touch the neighbors.
Complexity: $O(E)$
Writes are scattered in memory

Bottom-up (gather reads)

For each vertex, are the neighbors in the frontier?
Complexity $O(ED)$, where D is the diameter of the graph.
Writes are performed once linearly.

Direction Optimizing.
Level synchronous bfs.

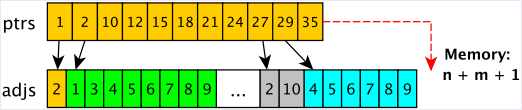


Outline

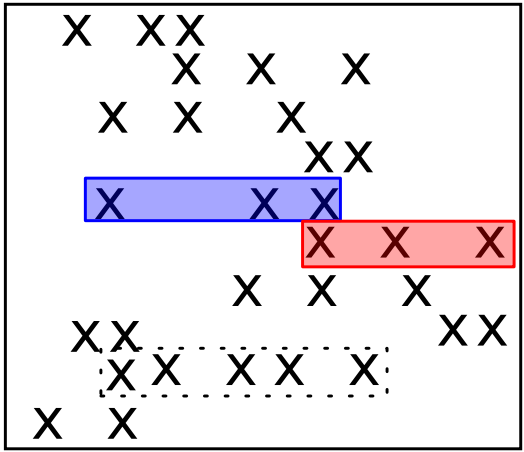
Traditionally ...

Vertex Centric

1 thread: 1 vertex



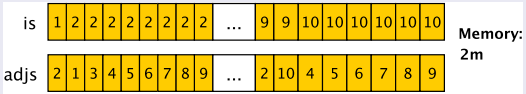
- No graph coalescing
- Vector read is not coalesced
- No atomics
- High divergence (high degree)



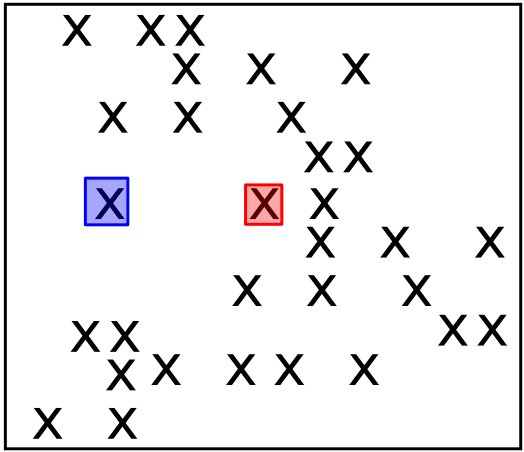
Traditionally ...

Edge Centric

1 thread: 1 edge



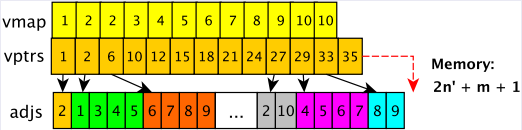
- Graph read is coalesced
- Vector read is not coalesced
- Many atomics
- Little divergence (likely to have adjacent thread doing the same vertex)



Virtual vertex decomposition

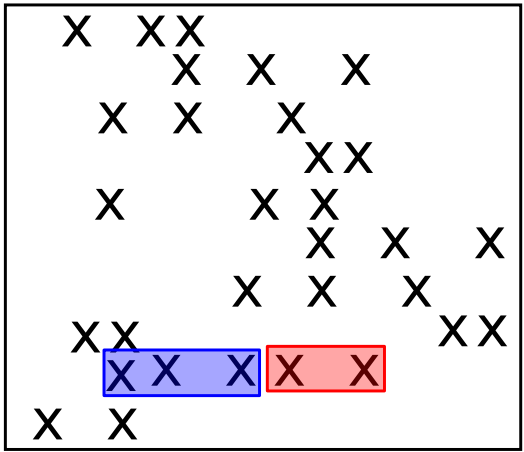
Virtual Vertex

1 thread: 1 virtual vertex



High degree vertices are split in multiple "virtual vertices"

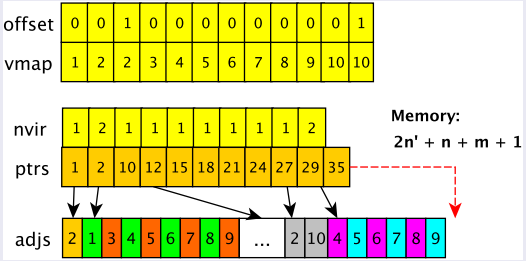
- No graph coalescing
- Vector read is not coalesced
- Some atomics
- Bounded divergence



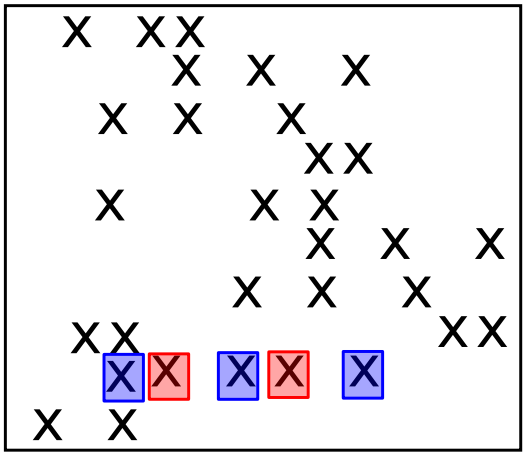
Strided virtual vertex decomposition

Virtual Vertex

1 thread: 1 virtual vertex



- Some graph coalescing
- Vector read is not coalesced
- Some atomics
- Bounded divergence



Experimental Setting

Instances

Graph	$\|V\|$	$\|E\|$	$Avg\|\Gamma(v)\|$	$Max\|\Gamma(v)\|$	Diam.
Amazon	403K	4,886K	12.1	2,752	19
Gowalla	196K	1,900K	9.6	14,730	12
Google	855K	8,582K	10.0	6,332	18
NotreDame	325K	2,180K	6.6	10,721	27
WikiTalk	2,388K	9,313K	3.8	100,029	10
Orkut	3,072K	234,370K	76.2	33,313	9
LiveJournal	4,843K	85,691K	17.6	20,333	15

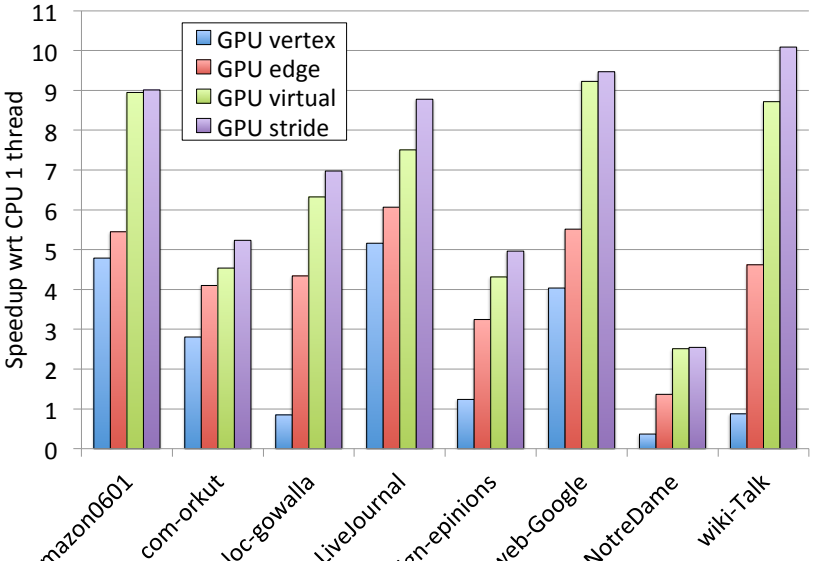
Machines

- 2 Intel Sandybridge EP
- NVIDIA K20

Metric

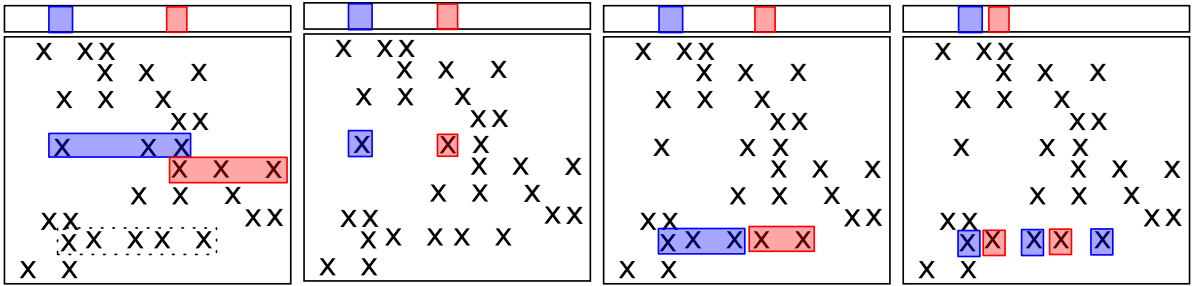
Traversed Edge Per Second: $\frac{VE}{time}$.

First results



Outline

No vector coalescing



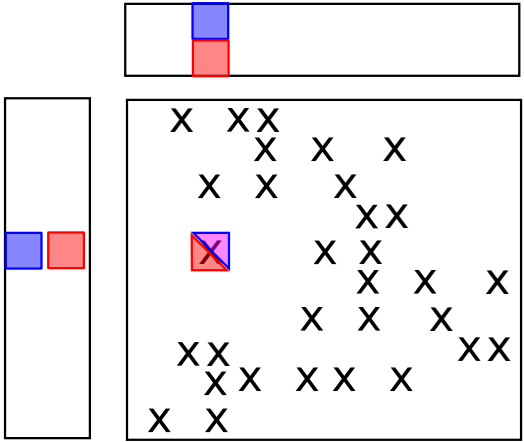
All the representations give vector coalescing only “if you are lucky”

Simultaneous sources traversal

The problem with previous methods is that BFS leaves the coalescing of the access to the vector up to the structure of the graph.

Multiple sources

- All threads of a warp should make similar access pattern.
- Since there are multiple traversals to perform in a Centrality computation, process B traversals at once.
- If a vertex is in the same level of the BFS in multiple traversal, they will be processed at the same time.
- Social networks have most vertices in a few levels.



An SpMV-based approach of BFS for Closeness Centrality

A simpler definition of level synchronous BFS

Vertex v is at level ℓ if and only if one of the neighbors of v is at level $\ell - 1$ and v is not at any level $\ell' < \ell$.

Let $x_i^\ell =$ **true** if vertex i is a part of the frontier at level ℓ.

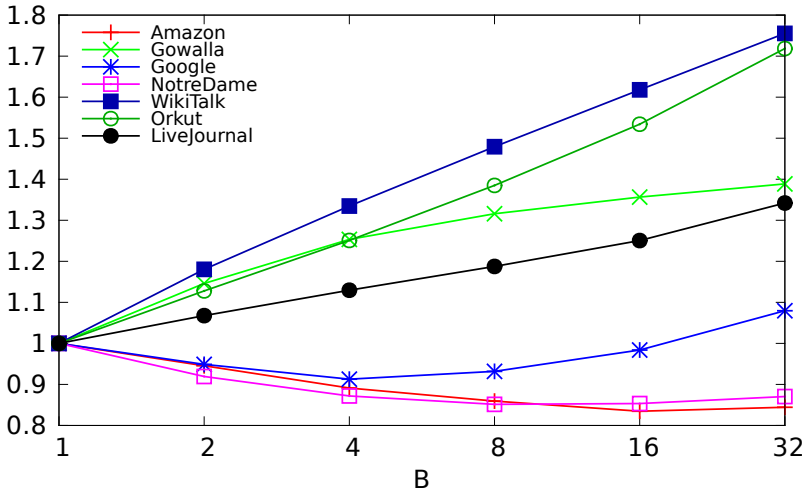
$y^{\ell+1}$ is the neighbors of level ℓ. $y_k^{\ell+1} = OR_{j \in \Gamma(k)} x_j^\ell$. ((OR, AND)-SpMV)

Compute the next level frontier $x_i^{\ell+1} = y_i^{\ell+1} \& \neg (OR_{\ell' \leq \ell} x_i^{\ell'})$.

Contribution of the source to $cc[i]$ is $\frac{x_i^\ell}{\ell}$.

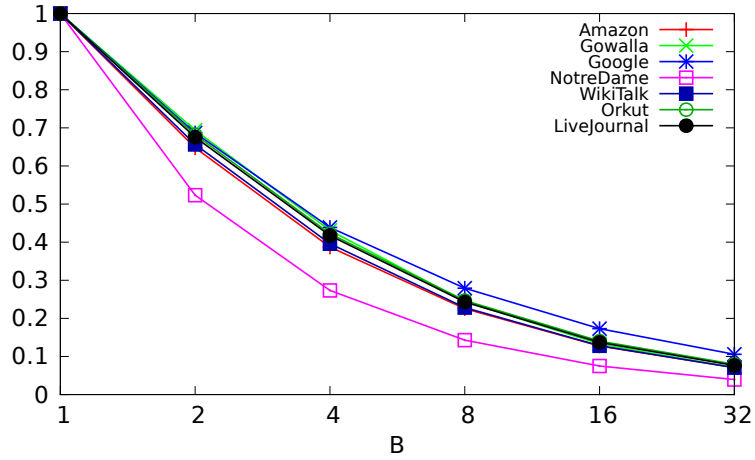
It allows to compute Closeness Centrality by encoding the state of 32 traversal with an *int*.

Impact on working warps



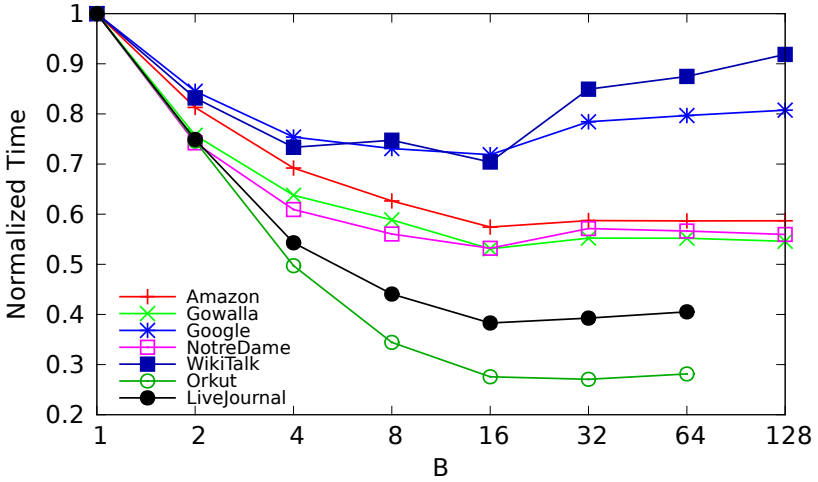
Number of active warps necessary for 32 traversals. Small increase in the number of warps.

Impact on non simultaneous traversal



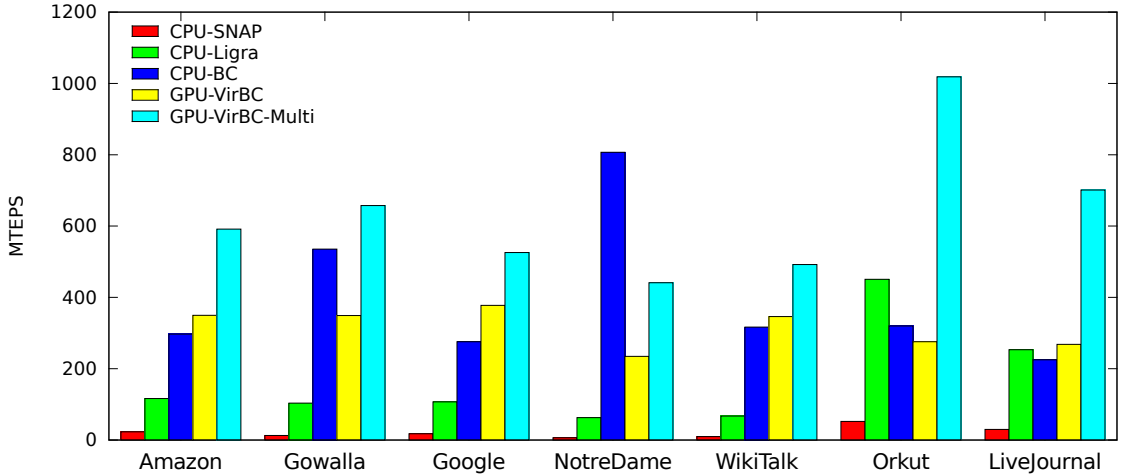
With $B = 4$, 32 traversals of one vertex are distributed in about 40% of 32 warps.
Good coalescing.

Impact on Runtime



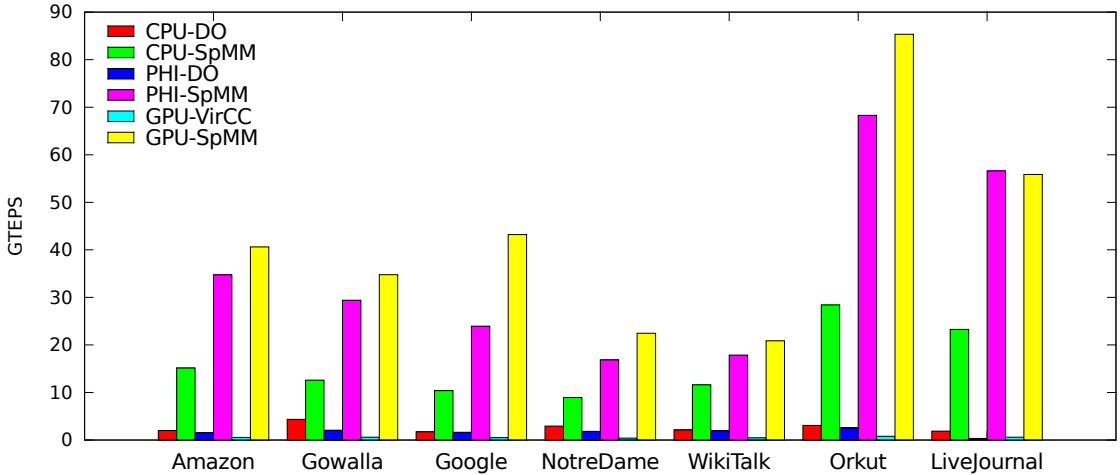
Outline

On other architecture? Betweenness Centrality



$O(DE)$ algorithms (GPU-) is unsuitable for NotreDame because of its high diameter.
CPU: 2 Intel Sandybridge EP (2x8 cores)

On other architecture? Closeness Centrality



CPU: 2 Intel Sandybridge EP (2x8 cores)

PHI: Intel Xeon Phi 5120

Conclusion

Centrality

Betweenness and Closeness Centrality are computed using multiple Breadth First Search traversal.

Graph representation for GPU

- Vertex Centric
- Edge Centric
- Virtual Vertex
- Coalesced Virtual Vertex

Determine parallelism but also memory access patterns and thread divergence.

Multiple traversals

- Centrality requires graph traversal from many different sources.
- Threads of a warp can be set to process different traversal for the same decomposition.
- Provided a vertex is used in the same level in multiple traversals, all the memory accesses can be coalesced.

Improves performance by a factor of 70x. Adapts to CPU architecture for similar effects.

Thank you

Other centrality works (with Sarıyüce, Kaya and Çatalyürek)

- Compression using graph properties (SDM 2013)
- GPU optimization (GPGPU 2013)
- Incremental algorithm (BigData 2013)
- Distributed memory incremental framework (Cluster 2013, ParCo 2015)
- Regularized memory accesses for CPU, GPU, Xeon Phi (MTAAP 2014, JPDC 2015)

More information

Contact : esaule@uncc.edu

Visit: `http://webpages.uncc.edu/~esaule`