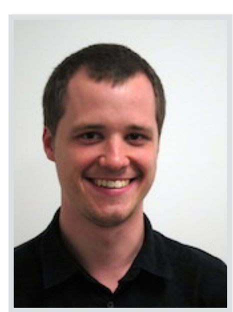
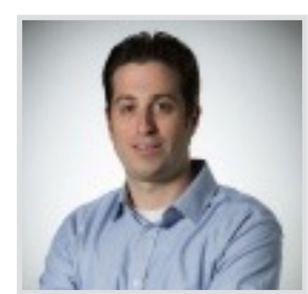
Georgia Tech School of Computational Science and Engineering

An Adaptive Parallel Algorithm for Computing Connectivity

Chirag Jain, Patrick Flick, Tony Pan, Oded Green, Srinivas Aluru

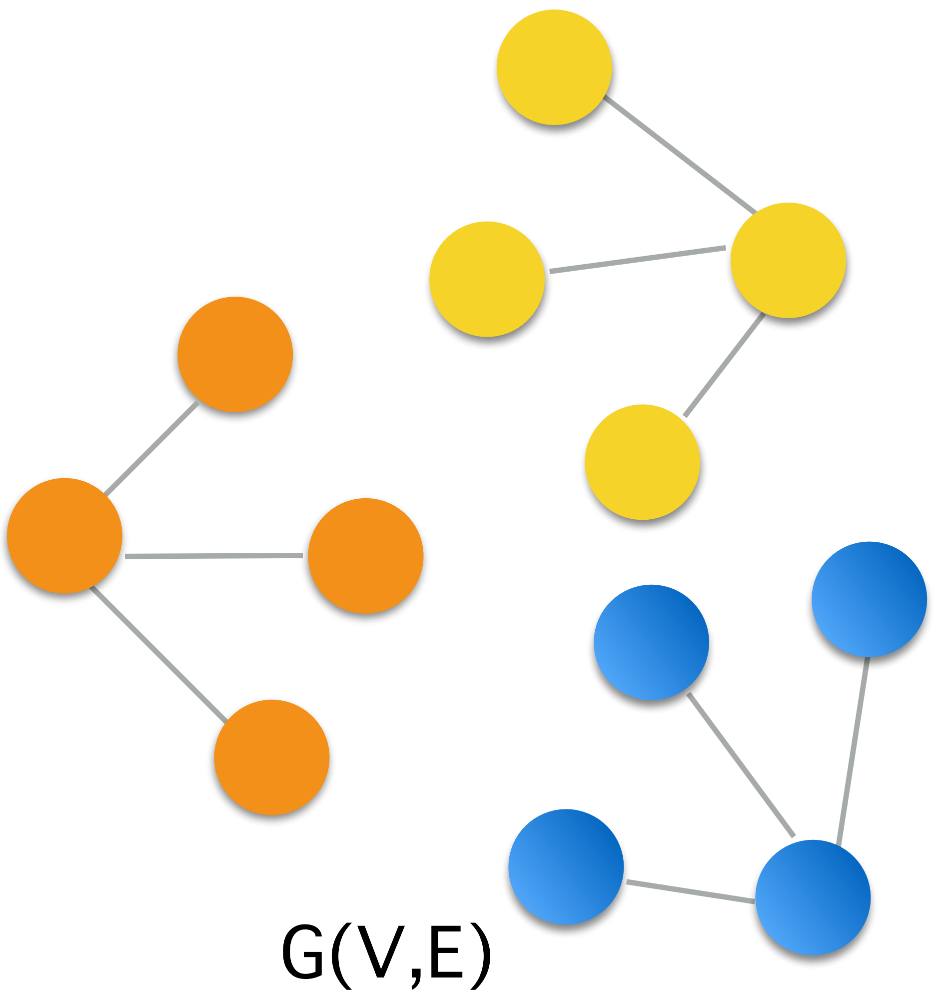






SIAM Workshop on Combinatorial Scientific Computing (CSC16)
October 10, 2016

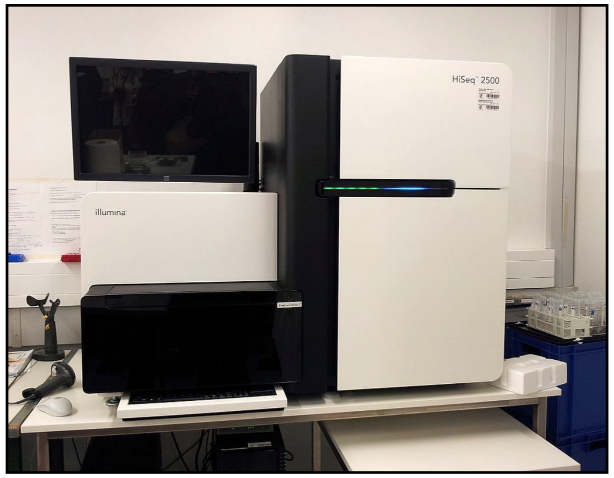
Connected Components

- Finding connected components is at the heart of many graph applications.
- Sequentially, we have linear time $O(|E|)$ solutions.
    - Union-find
    - BFS / DFS



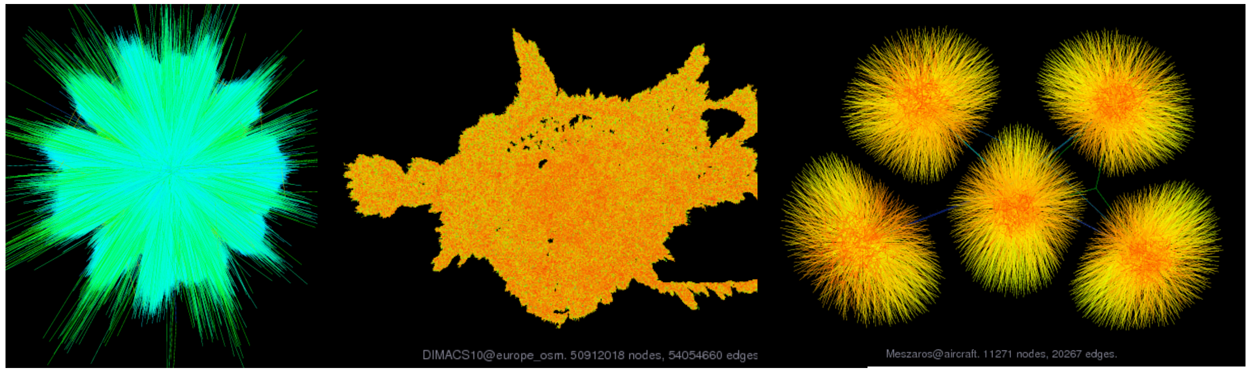
Scaling to Large Graphs

- Sizes of graph datasets continue to grow in multiple scientific domains
  - Bioinformatics : Metagenomics de-Bruijn graphs
    - Iowa Prairie (3.3B reads) - JGI
  - Social networks, WWW
- We need method that scales to graphs with billions/trillion of edges
  - irrespective of graph topology



Sequencing machines generate ~**10^9 DNA reads in 1 day**

> 10^9 content uploads in 1 day



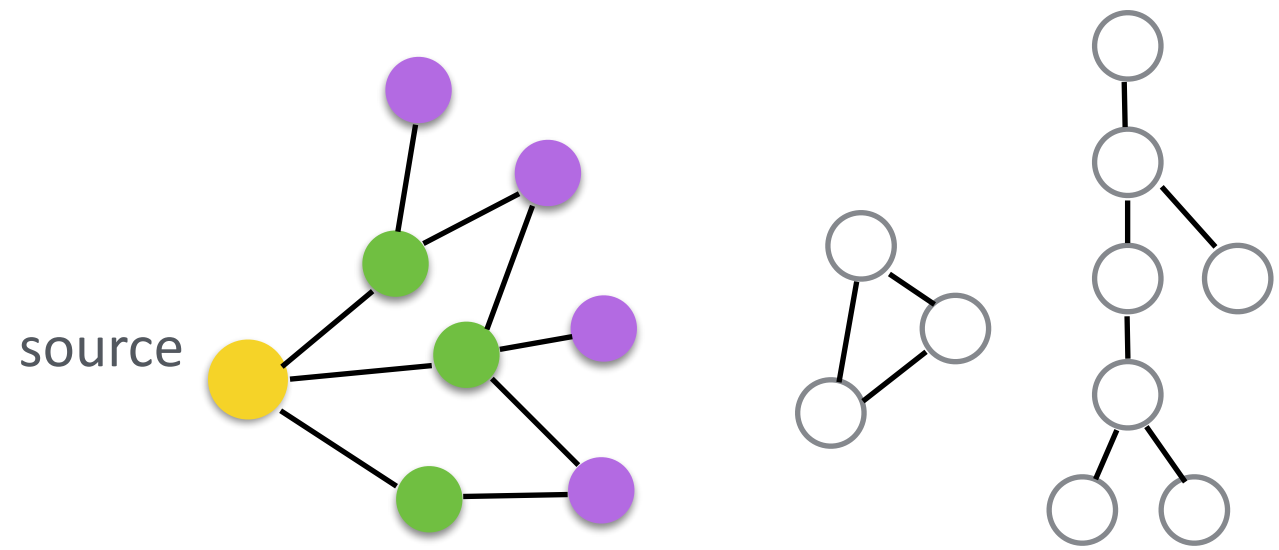
Background

A. Parallel connectivity algorithms

   1. **Parallel BFS**

   2. Shiloach-Vishkin PRAM algorithm (SV)

B. Recent prior work



Buluç and Madduri "Parallel breadth-first search …" SC 11
Beamer *et. al.* "Distributed memory breadth-first search revisited …" IPDPSW 13

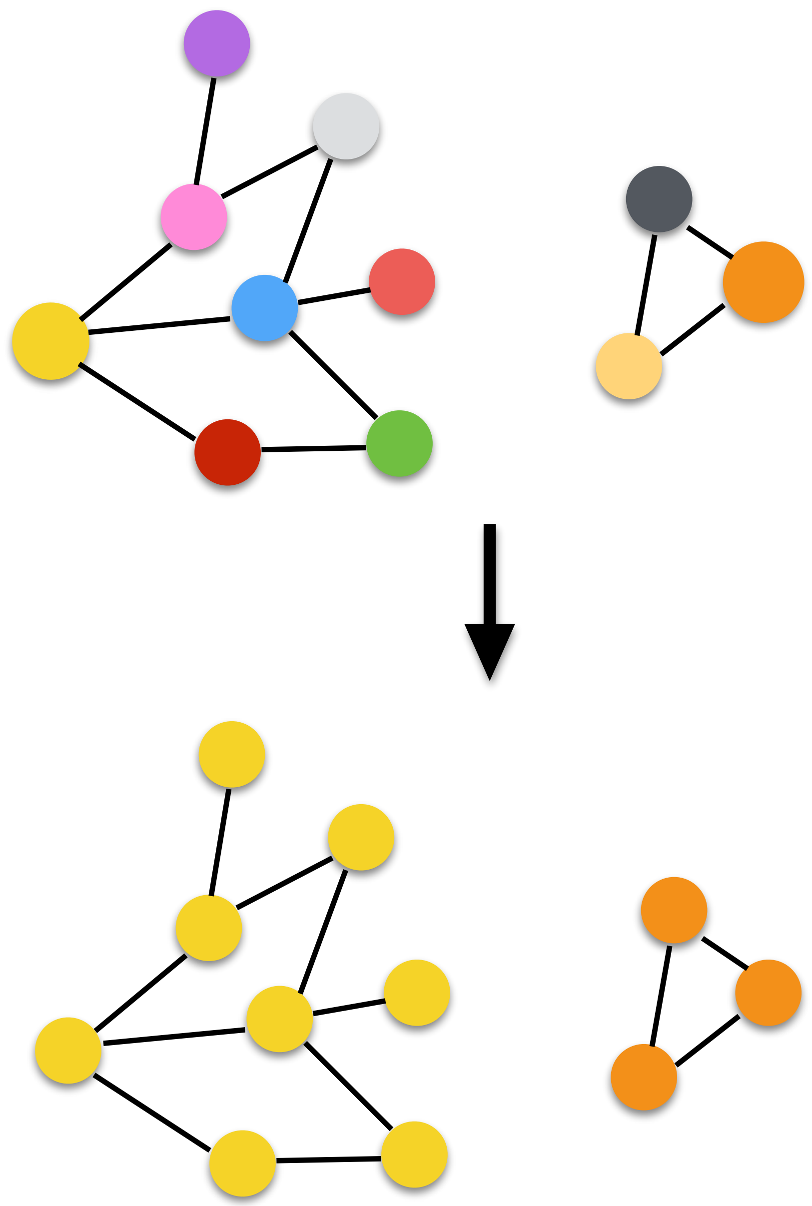
Background

A. Parallel connectivity algorithms

   1. Parallel BFS

   2. **Shiloach-Vishkin PRAM algorithm (SV)**

B. Recent prior work



Shiloach and Vishkin “An O(log n) parallel connectivity algorithm” 1982

Background

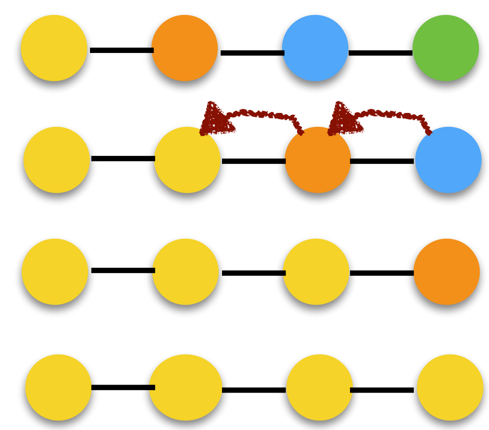
A. Parallel connectivity algorithms

  1. Parallel BFS

  2. **Shiloach-Vishkin PRAM algorithm (SV)**

B. Recent prior work

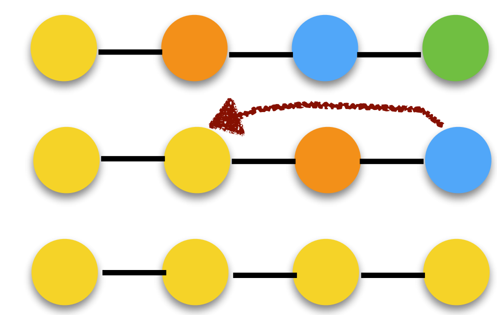
Label Propagation



$O(|V|)$ iterations

→ $O(|E|.|V|)$ work

Shiloach-Vishkin



Pointer jumping for faster convergence

$O(\log |V|)$ iterations

→ $O(|E| \log |V|)$ work

Shiloach and Vishkin “An O(log n) parallel connectivity algorithm” 1982

Background

A. Parallel connectivity algorithms

 1. Parallel BFS

 2. Shiloach-Vishkin PRAM algorithm (SV)

B. **Recent prior work**

Part of popular graph analysis frameworks : GraphX, PowerLyra, PowerGraph

G(V,E) → 1 Parallel BFS iteration → Parallel Label Propagation

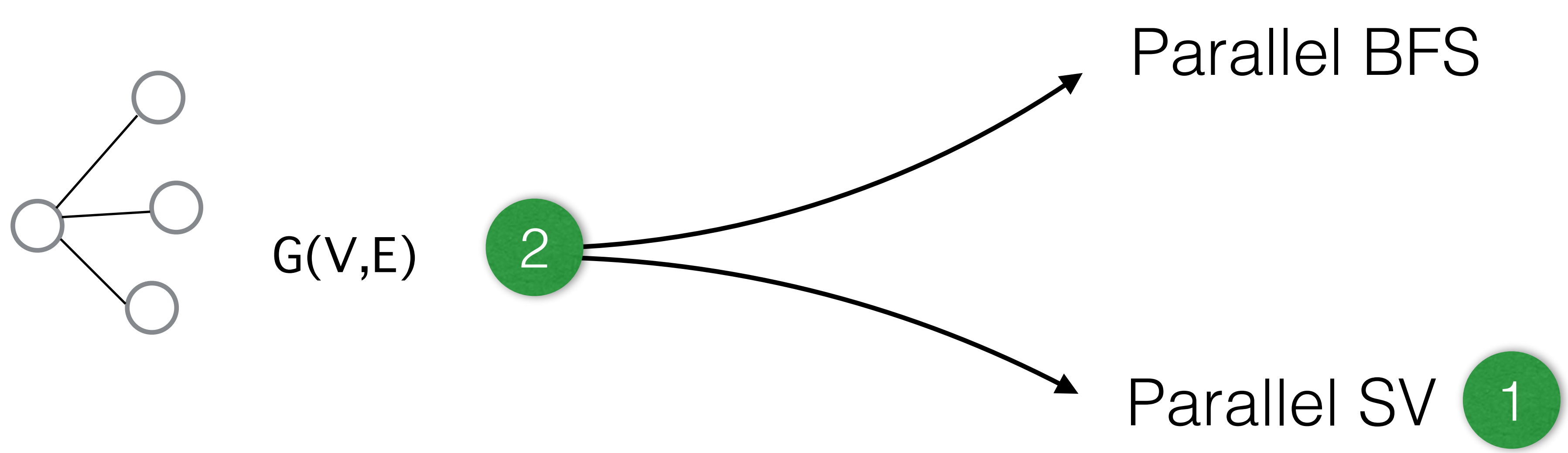
Multistep algorithm

Slota et. al. “A Case Study of Complex Graph Analysis …” IPDPS 2016
Slota et al. “BFS and coloring-based parallel … IPDPS 2014

Contributions

1. Novel **edge-based adaptation of Shiloach-Vishkin algorithm** for distributed memory parallel systems.

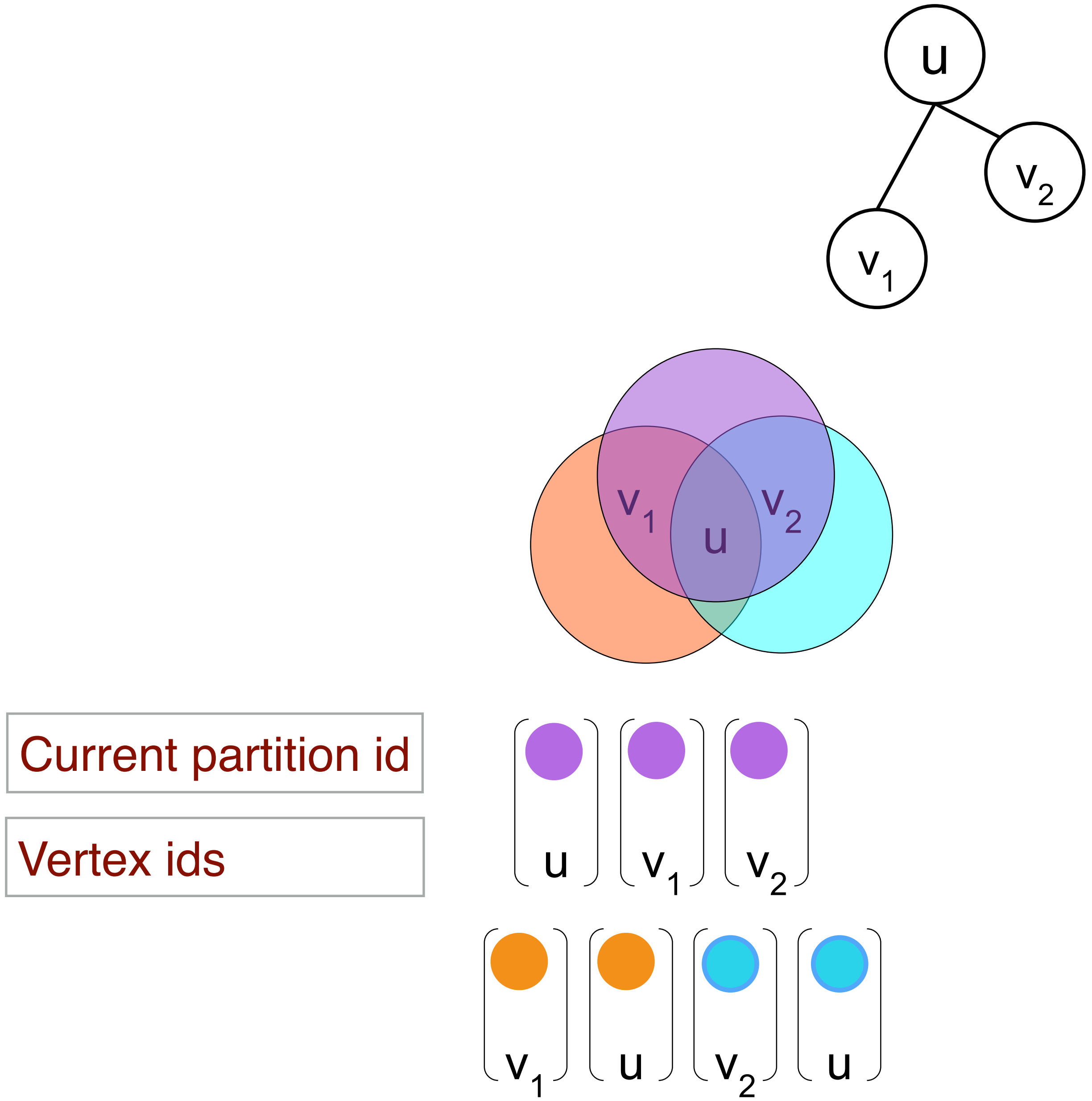
2. Fast heuristic to guide **algorithm selection at run-time**.



Flick et. al. "A parallel connectivity algorithm …" SC 15

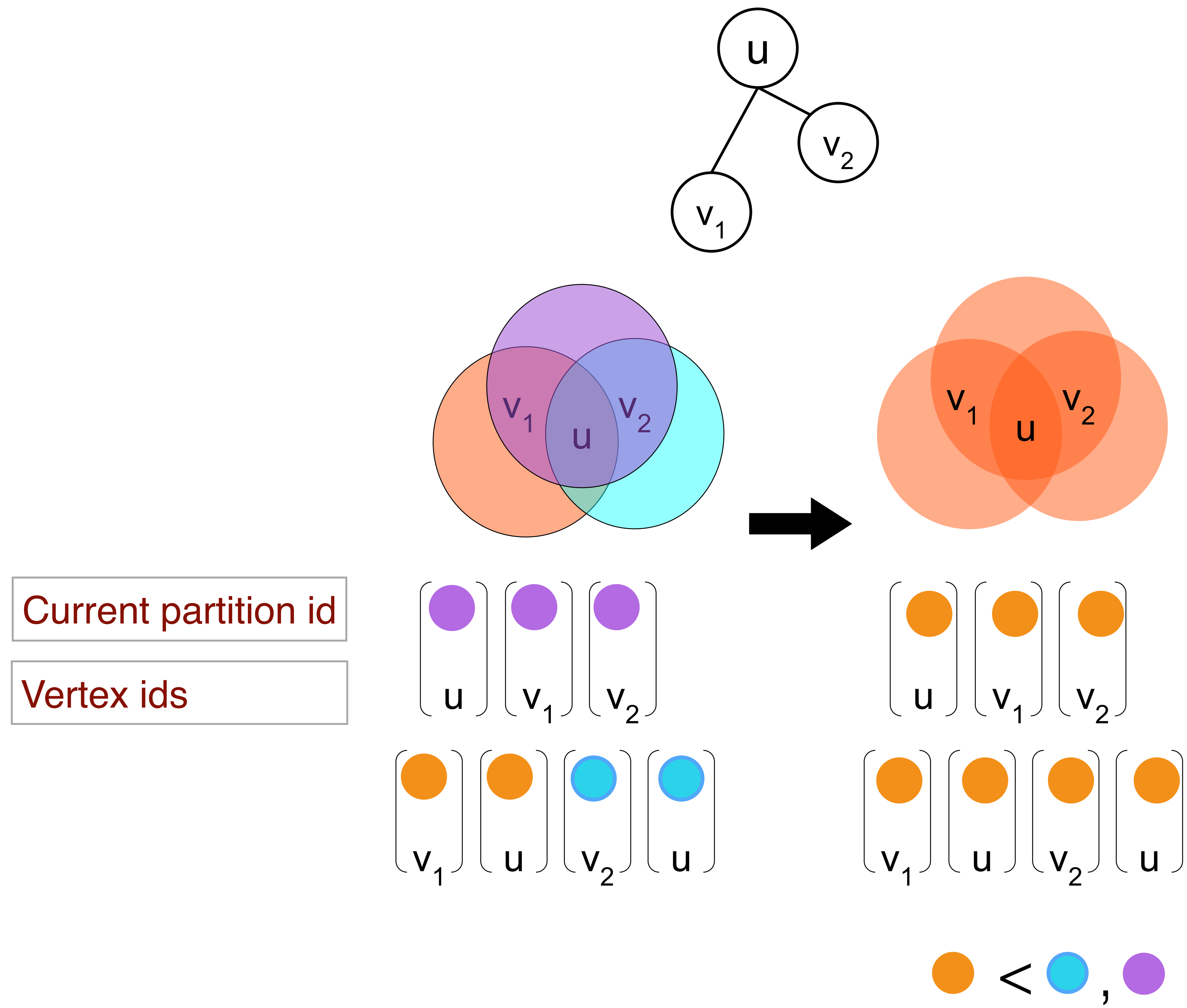
Parallel SV algorithm

- Initialization
  - We work with an array of tuples (call it A) to keep partition id of each vertex.
  - O(|V|) partitions at beginning
  - Size of A : O(|V| + |E|)

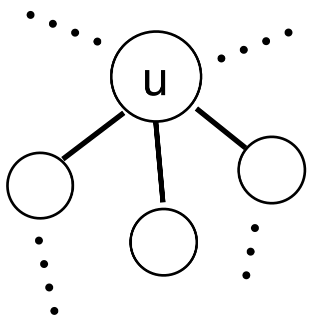


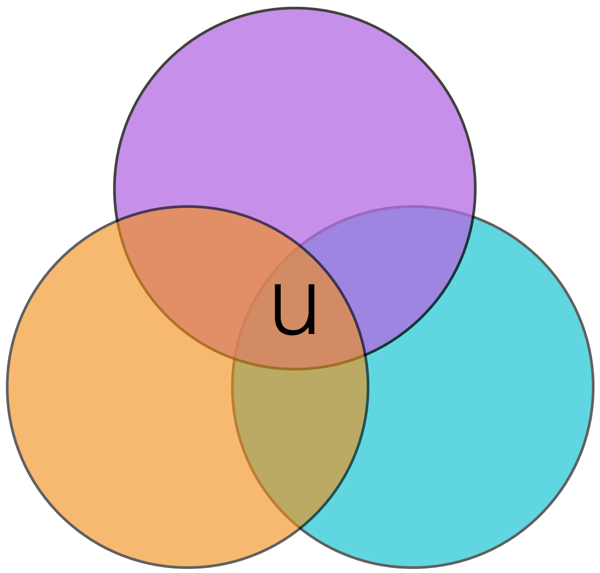
Parallel SV algorithm

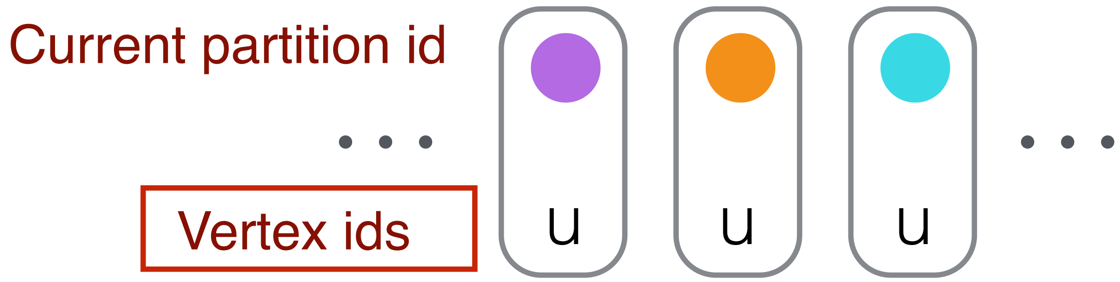
- Initialization
  - We work with an array of tuples (call it A) to keep partition id of each vertex.
  - O(|V|) partitions at beginning
  - Size of A : O(|V| + |E|)



Parallel SV algorithm

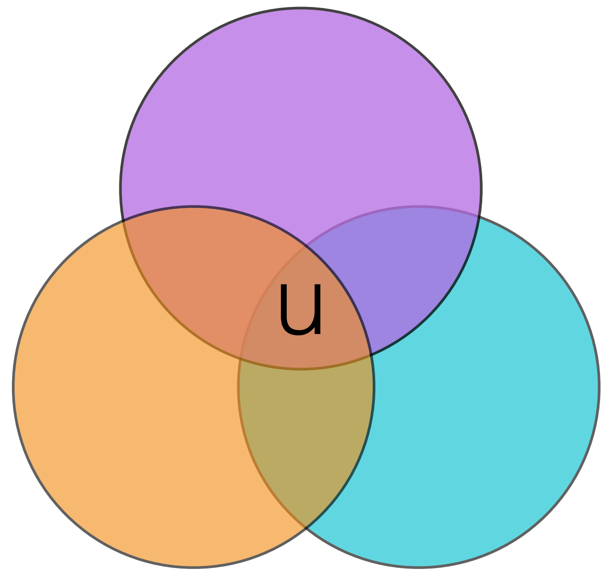


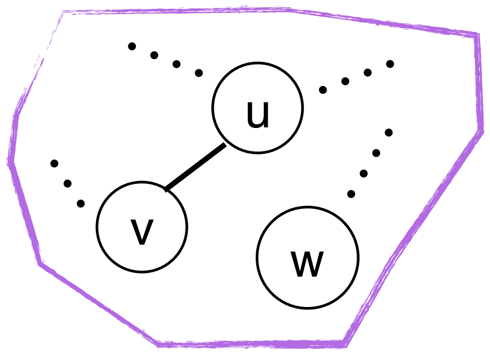


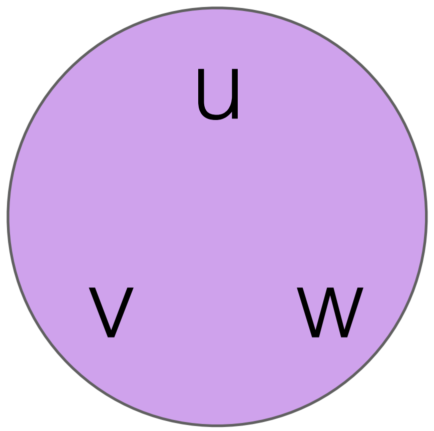


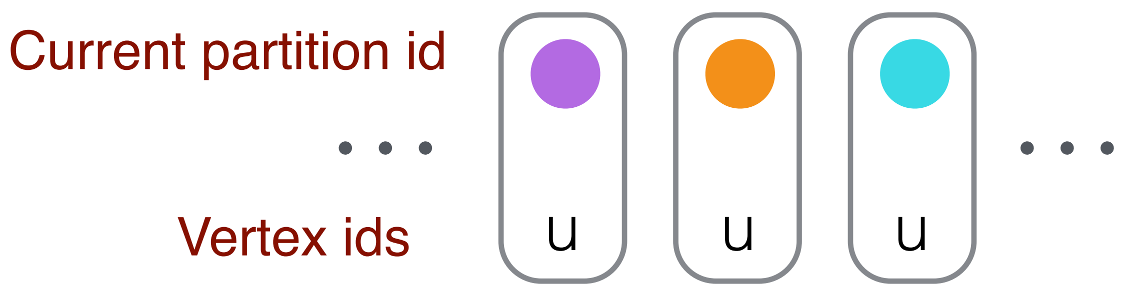
- vertex ‘u’ is member of which all partition ids?
  - Sort **A** by ‘vertex id’ layer

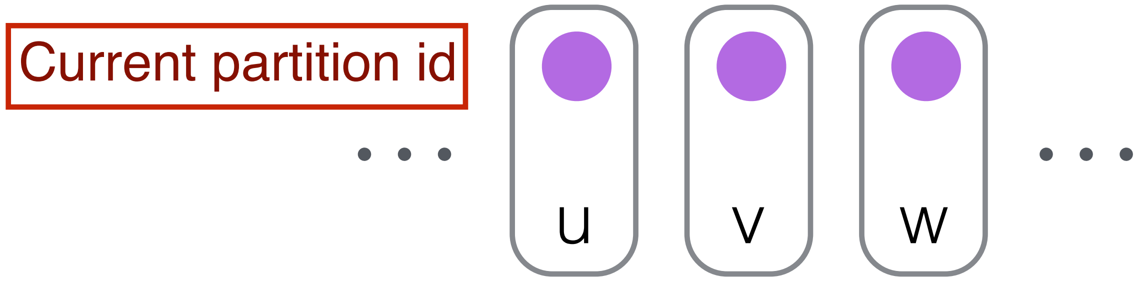
Parallel SV algorithm







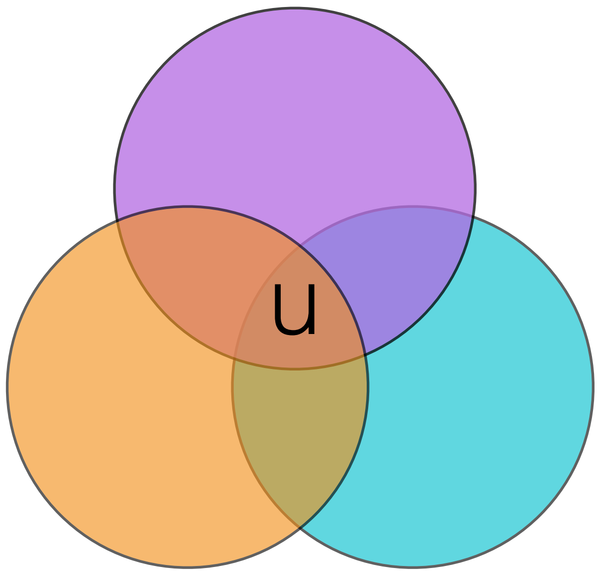


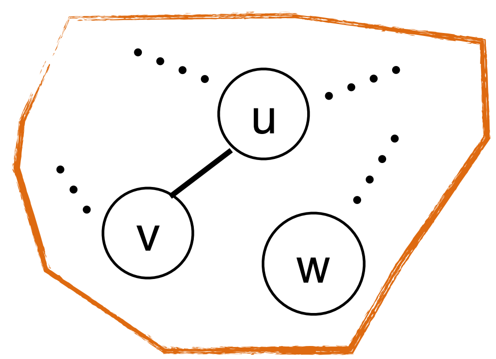


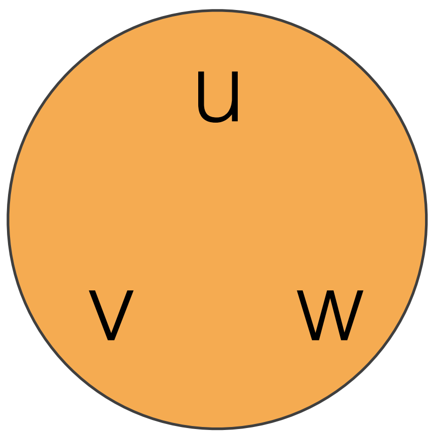
- vertex ‘u’ is member of which all partition ids?
  - Sort **A** by ‘vertex id’ layer

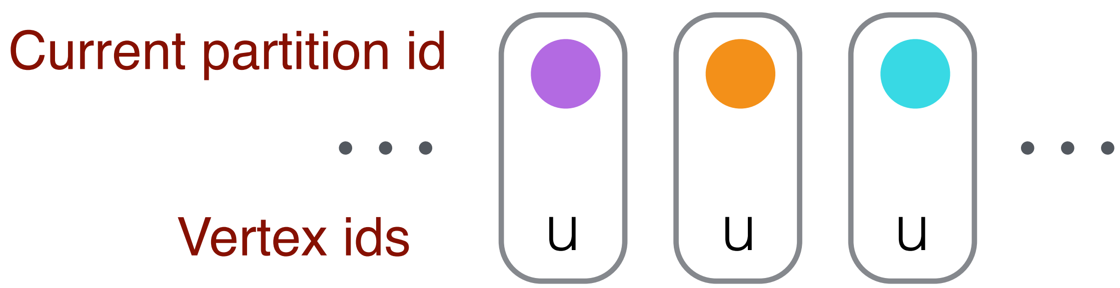
- Which all vertices are member of partition ● ?
  - Sort **A** by ‘partition id’ layer

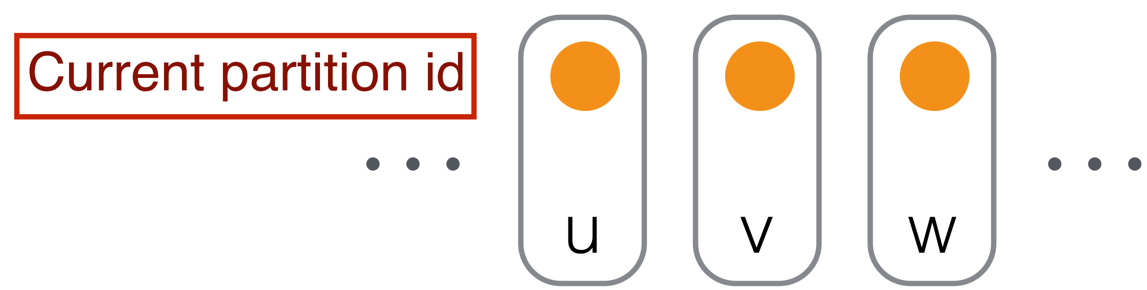
Parallel SV algorithm











- vertex ‘u’ is member of which all partition ids?
  - Sort A by ‘vertex id’ layer

- Which all vertices are member of partition ?
  - Sort A by ‘partition id’ layer

Parallel SV algorithm

- In our implementation, we use parallel sample sort.
- Custom reduction operations to efficiently compute minimums.
- Additional details:

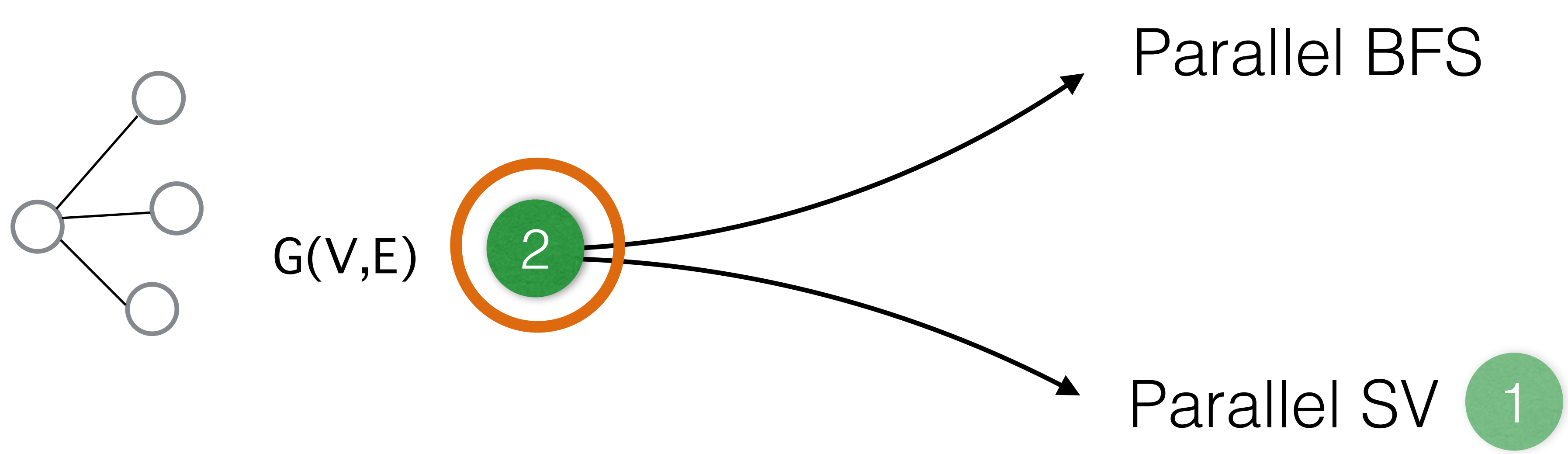
  Check our preprint

  - pointer jumping
  - detect convergence of small components early, load balance
- Runtime : $O(\log |V| \cdot T_{sort}(|V| + |E|, p))$

Contributions

1. Novel **edge-based adaptation of Shiloach-Vishkin algorithm** for distributed memory parallel systems.

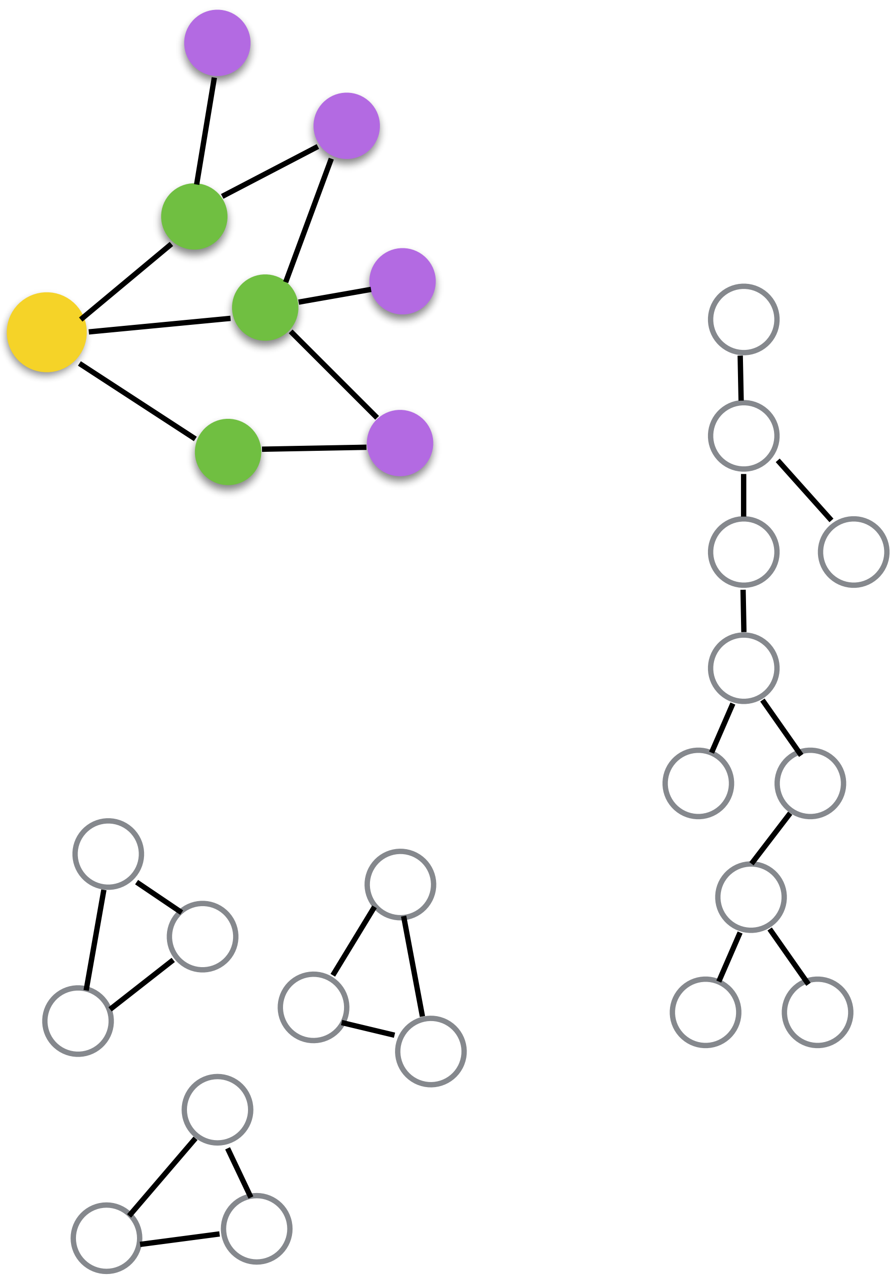
2. Fast heuristic to guide **algorithm selection at run-time**.



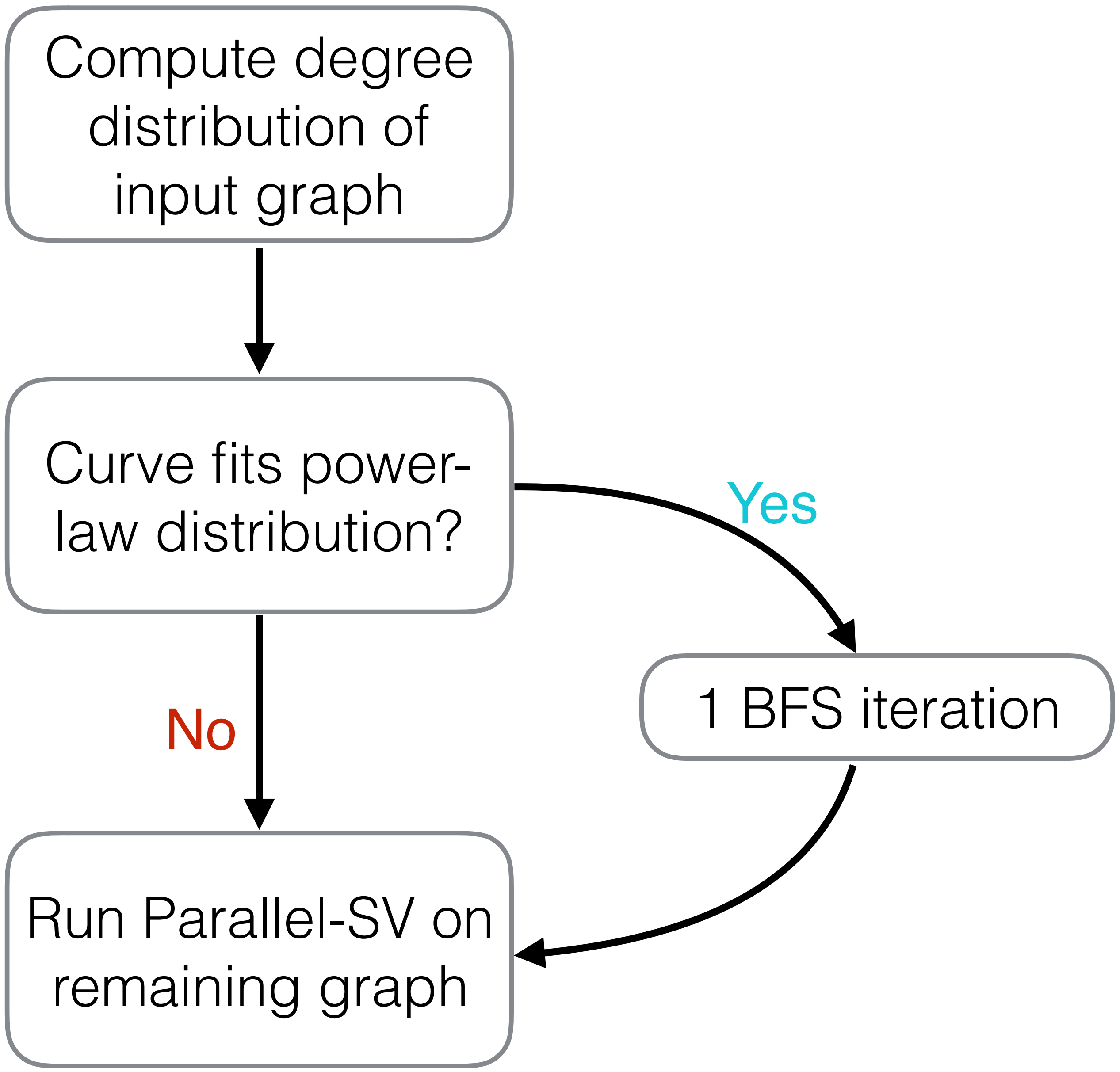
Flick et. al. "A parallel connectivity algorithm …" SC 15

Dynamic hybrid method

- Parallel BFS is close to work efficient for a giant small world graph component.
- Efficiency is lost when :
  - Large number of small components
  - Large diameter of a graph component
- How to decide which algorithm to choose at runtime?



Dynamic hybrid method



Experimental Setup

- **Software** : C++14, MPI, CombBLAS library for parallel BFS
- **Hardware** : Cray XC30 (Edison) at Lawrence Berkeley National Laboratory
  - 5,576 nodes, each with 2 x 12-core Intel Ivy processors and 64 GB RAM
  - 1 MPI process per physical core
- **Timing** :
  - Exclude graph construction and I/O time
  - Profiling starts after having block-distributed list of edges in memory

Buluç and Gilbert "The Combinatorial BLAS: Design …" IJHPCA 2011

Datasets

Id	Dataset	Type	Vertices	Edges	Components	Approx. diameter	Largest component
M1	Lake Lanier	Metagenomic	1.1 B	1.1 B	2.6 M	3,763	53%
M2	Human Metagenome	Metagenomic	2.0 B	2.0 B	1.0 M	3,989	91.1%
M3	Soil (Peru)	Metagenomic	531.2 M	523.6 M	7.6 M	2,463	0.3%
M4	Soil (Iowa)	Metagenomic	53.7 B	53.6 B	319.2 M	-	44.2%
G1	Twitter	Social	52.6 M	2.0 B	29,533	16	99.99%
G2	sk-2005	Web Crawl	50.6 M	1.9 B	45	27	99.99%
G3	eu-usa-osm	Road Networks	74.9 M	82.9 M	2	25,105	65.2%
K1	Kronecker (scale = 27)	Kronecker	63.7 M	2.1 B	19,753	9	99.99%
K2	Kronecker (scale = 29)	Kronecker	235.4 M	8.6 B	73,182	9	99.99%

Datasets

Id	Dataset	Type	Vertices	Edges	Components	Approx. diameter	Largest component
M1	Lake Lanier	Metagenomic	1.1 B	1.1 B	2.6 M	3,763	53%
M2	Human Metagenome	Metagenomic	2.0 B	2.0 B	1.0 M	3,989	91.1%
M3	Soil (Peru)	Metagenomic	531.2 M	523.6 M	7.6 M	2,463	0.3%
M4	Soil (Iowa)	Metagenomic	53.7 B	53.6 B	319.2 M	-	44.2%
G1	Twitter	Social	52.6 M	2.0 B	29,533	16	99.99%
G2	sk-2005	Web Crawl	50.6 M	1.9 B	45	27	99.99%
G3	eu-usa-osm	Road Networks	74.9 M	82.9 M	2	25,105	65.2%
K1	Kronecker (scale = 27)	Kronecker	63.7 M	2.1 B	19,753	9	99.99%
K2	Kronecker (scale = 29)	Kronecker	235.4 M	8.6 B	73,182	9	99.99%

Small world graphs

Datasets

Id	Dataset	Type	Vertices	Edges	Components	Approx. diameter	Largest component
M1	Lake Lanier	Metagenomic	1.1 B	1.1 B	2.6 M	3,763	53%
M2	Human Metagenome	Metagenomic	2.0 B	2.0 B	1.0 M	3,989	91.1%
M3	Soil (Peru)	Metagenomic	531.2 M	523.6 M	7.6 M	2,463	0.3%
M4	Soil (Iowa)	Metagenomic	53.7 B	53.6 B	319.2 M	-	44.2%
G1	Twitter	Social	52.6 M	2.0 B	29,533	16	99.99%
G2	sk-2005	Web Crawl	50.6 M	1.9 B	45	27	99.99%
G3	eu-usa-osm	Road Networks	74.9 M	82.9 M	2	25,105	65.2%
K1	Kronecker (scale = 27)	Kronecker	63.7 M	2.1 B	19,753	9	99.99%
K2	Kronecker (scale = 29)	Kronecker	235.4 M	8.6 B	73,182	9	99.99%

Large diameter graph

Small world graphs

Datasets

Large number of components

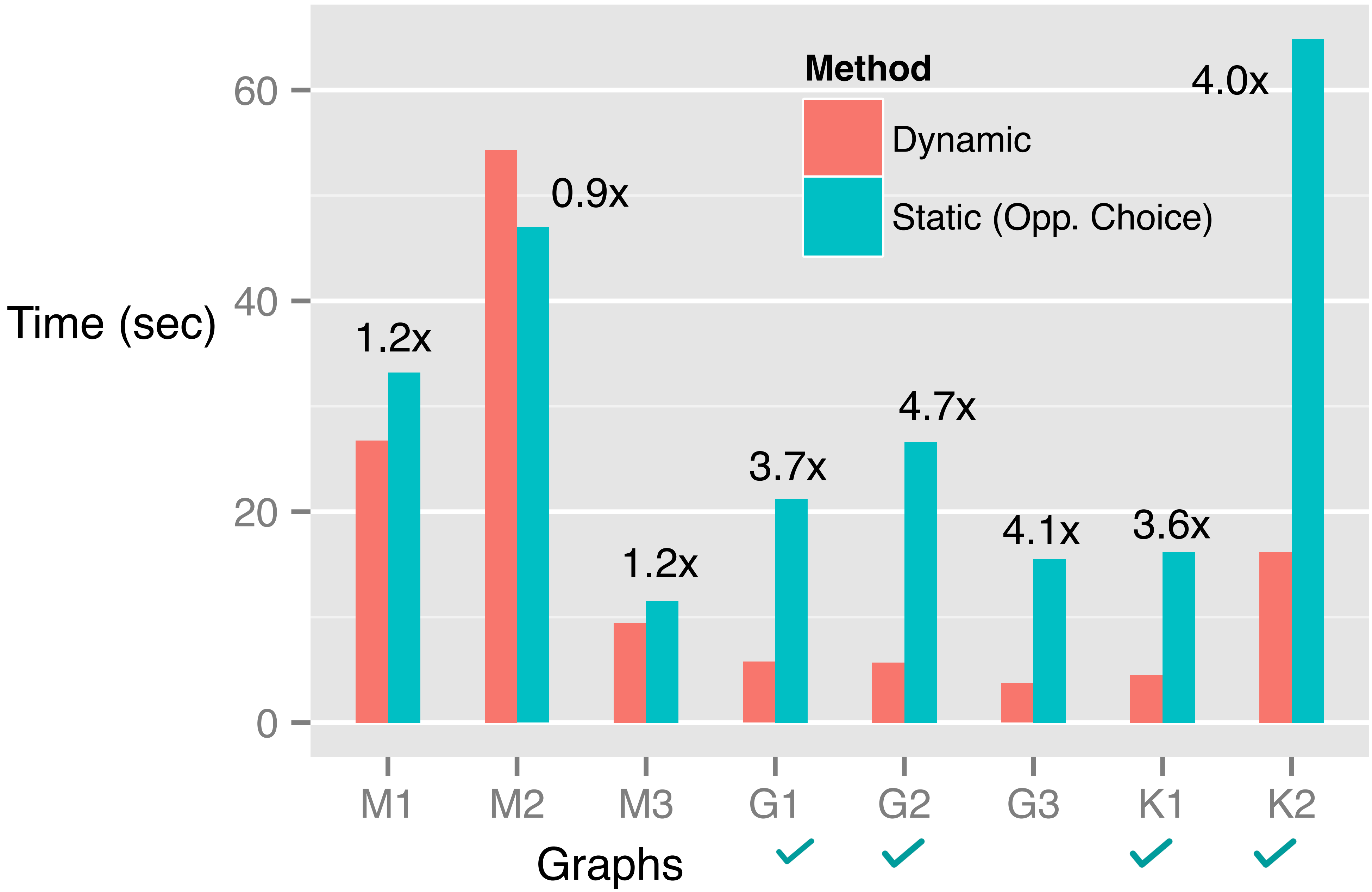
Id	Dataset	Type	Vertices	Edges	Components	Approx. diameter	Largest component
M1	Lake Lanier	Metagenomic	1.1 B	1.1 B	2.6 M	3,763	53%
M2	Human Metagenome	Metagenomic	2.0 B	2.0 B	1.0 M	3,989	91.1%
M3	Soil (Peru)	Metagenomic	531.2 M	523.6 M	7.6 M	2,463	0.3%
M4	Soil (Iowa)	Metagenomic	53.7 B	53.6 B	319.2 M	-	44.2%
G1	Twitter	Social	52.6 M	2.0 B	29,533	16	99.99%
G2	sk-2005	Web Crawl	50.6 M	1.9 B	45	27	99.99%
G3	eu-usa-osm	Road Networks	74.9 M	82.9 M	2	25,105	65.2%
K1	Kronecker (scale = 27)	Kronecker	63.7 M	2.1 B	19,753	9	99.99%
K2	Kronecker (scale = 29)	Kronecker	235.4 M	8.6 B	73,182	9	99.99%

Large diameter graph

Small world graphs

Dynamic Approach

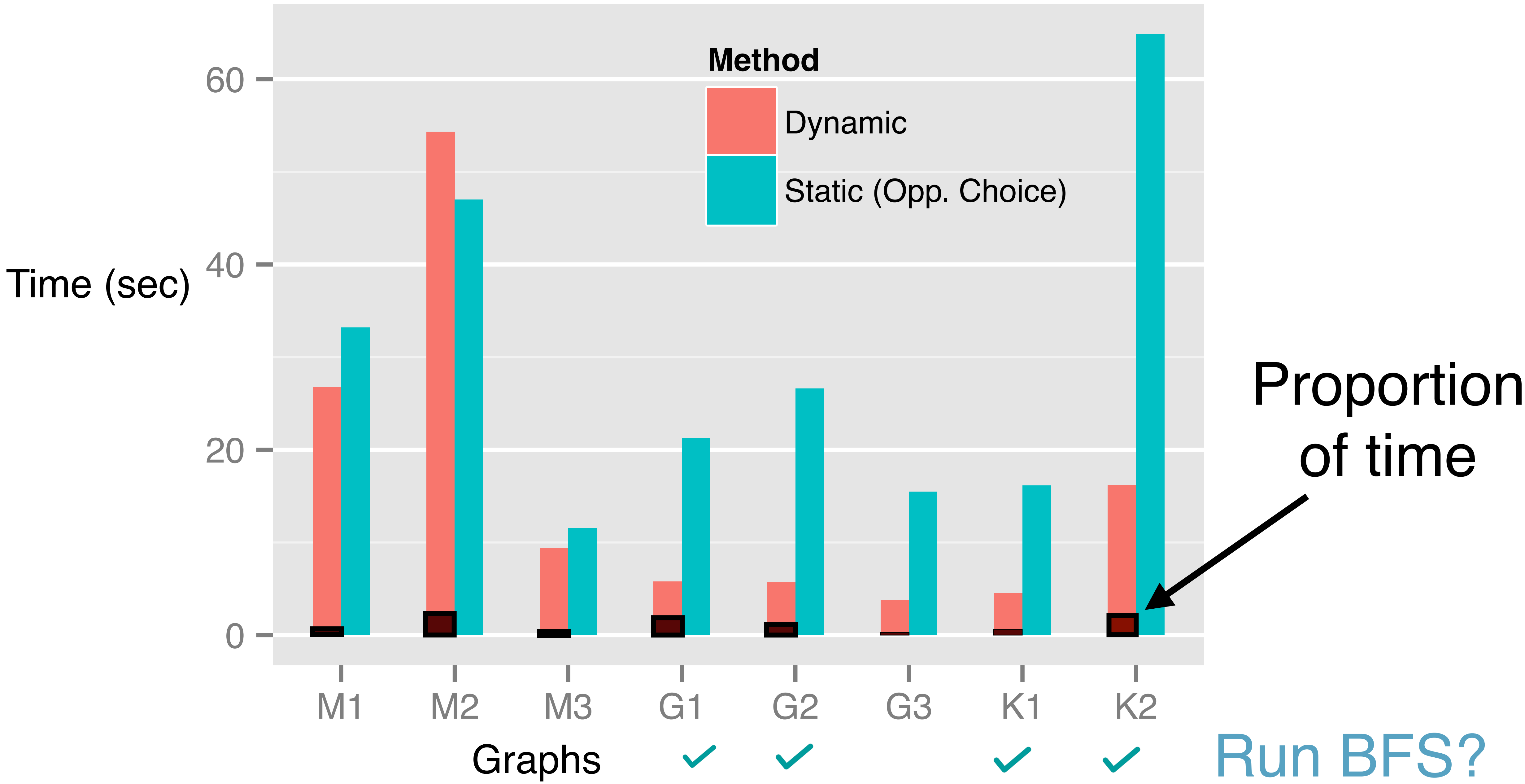
Timings against opposite choice, using 2K cores



Run BFS?

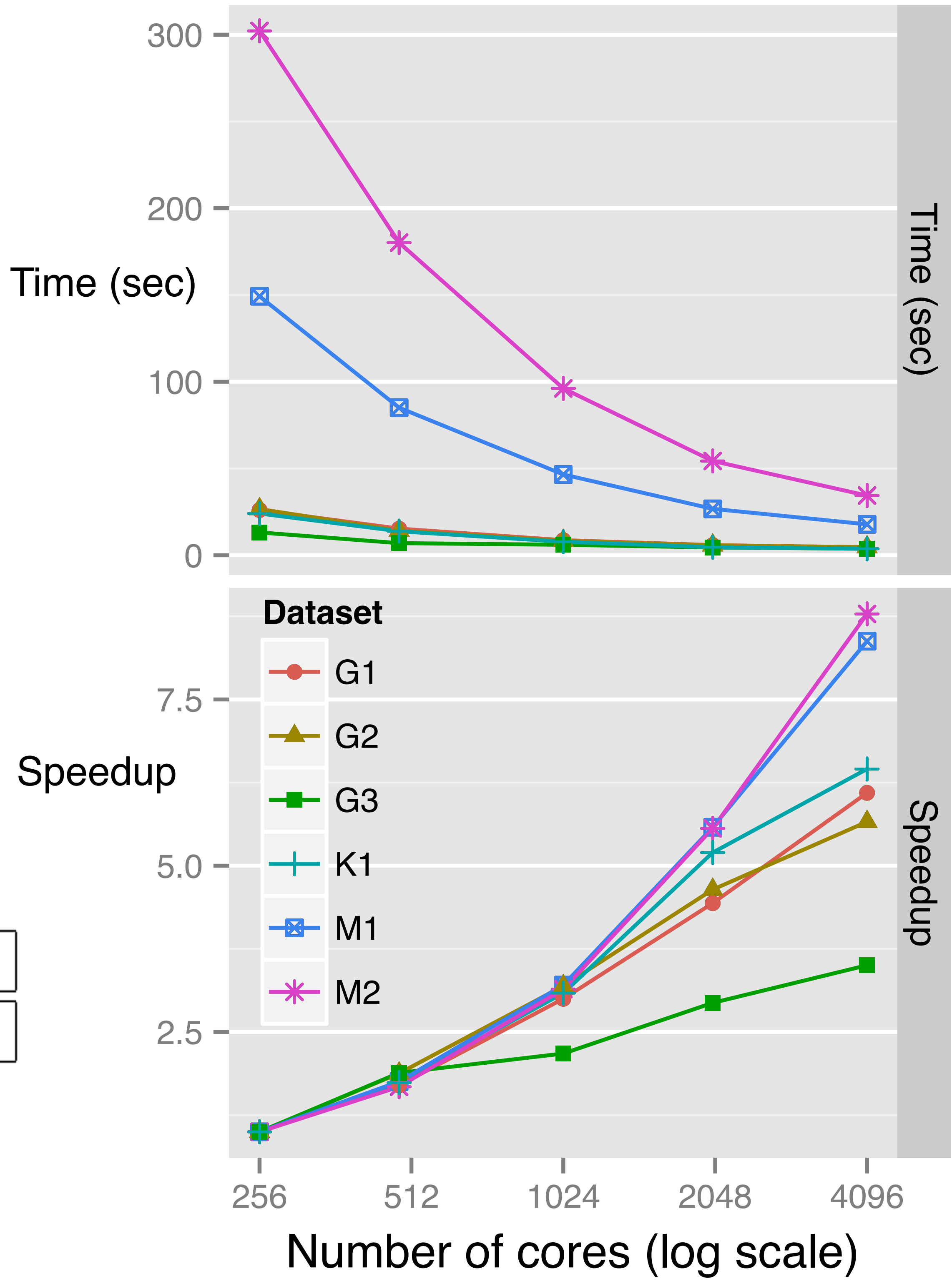
Dynamic Approach

Proportion of time spent in prediction (using 2K cores)



Strong Scalability

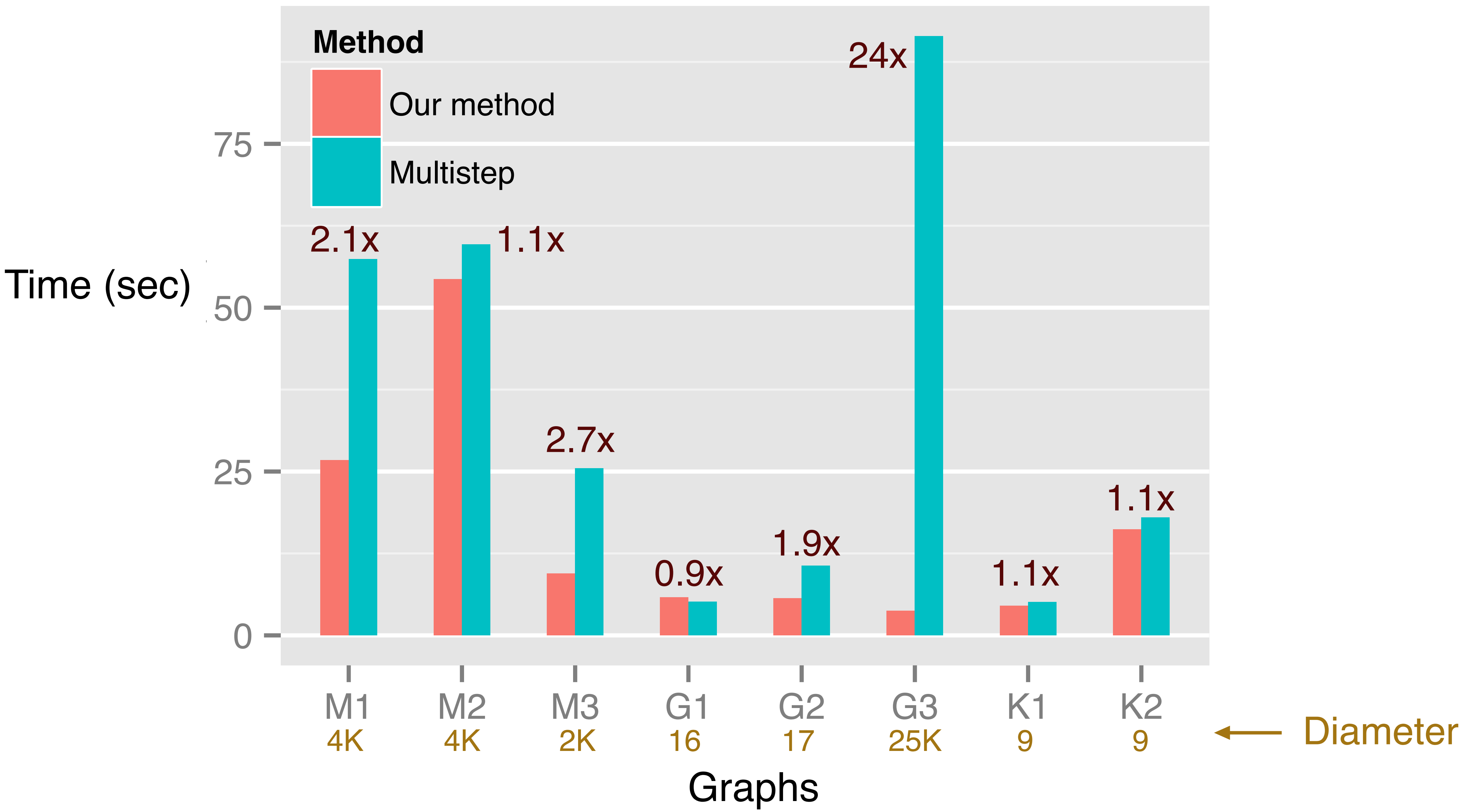
- Maximum speedup of ~8x using 4096 cores (Ideal :16x)
- Sorting benchmark with 2B integers achieves 8.06x speedup as well.



Timings for the largest graph M4

Cores	8281	16384	32761
Time for M4 (sec)	429.89	291.19	214.56

v/s Multistep method



v/s Best sequential method

- Performance comparison against Rem's algorithm (based on union-find)
- Using small graphs that fit in single node (64 GB RAM)

Dataset	**Seq. Time (sec)**	**Speedup** p = 64	256	1024
Kronecker (25)	228.8	10.1	34.3	100.6
M3	406.2	2.5	9.3	27.0
G3	45.9	0.9	3.5	7.6

E. W. Dijkstra, A discipline of programming. 1976

Conclusions

1. Efficient distributed memory parallel connectivity algorithm based on Shiloach-Vishkin approach.

2. Propose heuristic to guide algorithm selection at runtime.

3. Efficient as well as generic, scales on a variety of large graphs.

4. Significant performance gains against previous state-of-the-art, particularly in case of large diameter graphs.

Thank you!

arXiv.org arxiv.org/abs/1607.06156

cjain @ gatech.edu

GitHub github.com/ParBLiSS/parconnect

SC16 Salt Lake City, Utah | hpc matters.

Reproducibility Initiative Award