# Gunrock: A High Performance Graph Processing Library on the GPU

June 4, 2015

Yangzihao Wang
University of California, Davis

# PROBLEM

## High-performance large-scale graph analytic

# PROBLEM

# High-performance large-scale graph analytic on GPUs

# Why Using GPUs for Graph Analytic?

- Graphs are ubiquitous
- Data size is becoming very large
- Graph analytic systems demand more performance

# Large-scale Graph Analytic Is Difficult

- Irregularity of data access and control flow limits performance and scalability
- GPU programming is complex

# Related Work

- Single-node CPU-based systems: Boost Graph Library
- Distributed CPU-based systems: PowerGraph, Pregel
- Specialized GPU algorithms and GPU-based systems

# Related Work

- Single-node CPU-based systems: Boost Graph Library
- Distributed CPU-based systems: PowerGraph, Pregel
- Specialized GPU algorithms and GPU-based systems

Can we do better? (in terms of both performance and expressiveness)

# IDEA: Performance AND expressiveness

- Performance: Integrating high performance GPU computing primitives and optimizations into the core.

- Expressiveness: A data-centric abstraction designed specifically for the GPU

# IDEA: Bulk-Synchronous Programming and Data-Centric

- Graph algorithms as iterative convergent processes
  - A series of bulk synchronous steps
  - Large amount of parallelism within each step
- Manipulate frontiers
  - Generating/reorganizing frontier in parallel
  - Computing on frontier in parallel

# Expressiveness: Gunrock's Data-Centric Abstraction

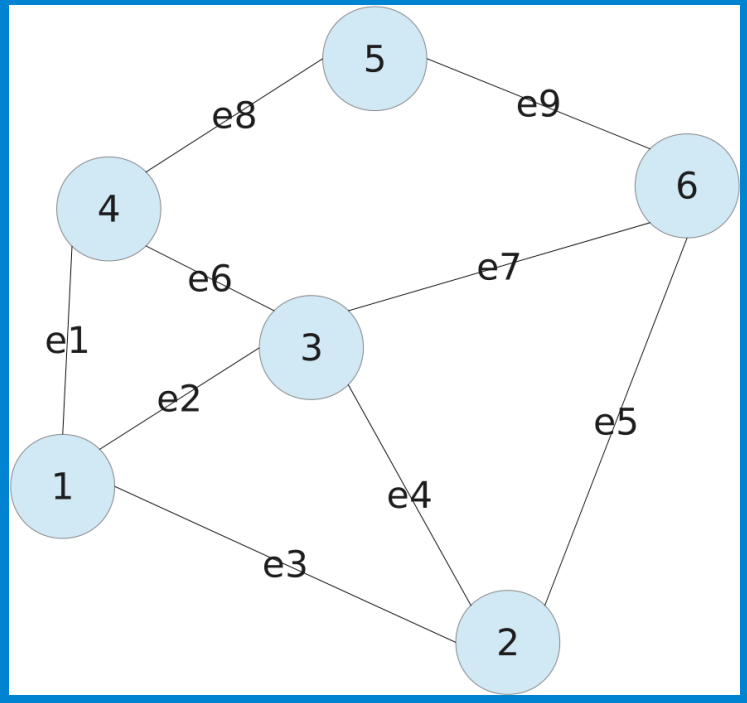
# Gunrock's Key Abstraction Is FRONTIER

Most graph algorithms have two major operations:

- Traverse: moving in the graph and generating new frontier
- Compute: doing computation on frontier

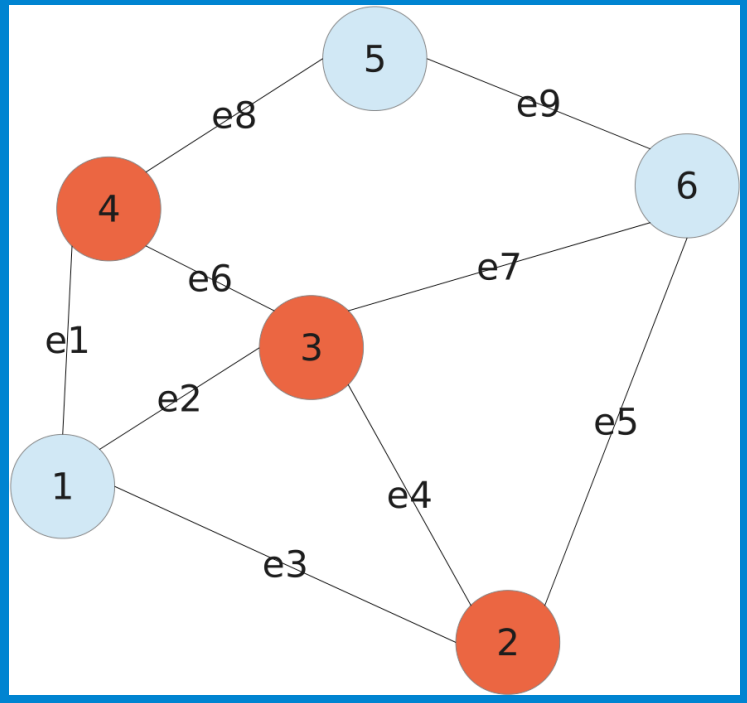
# Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier



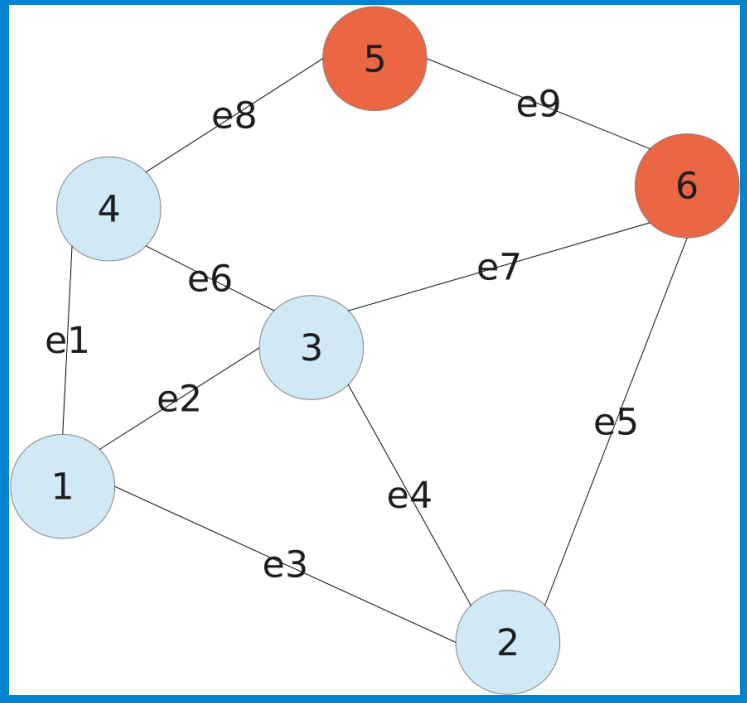
# Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier



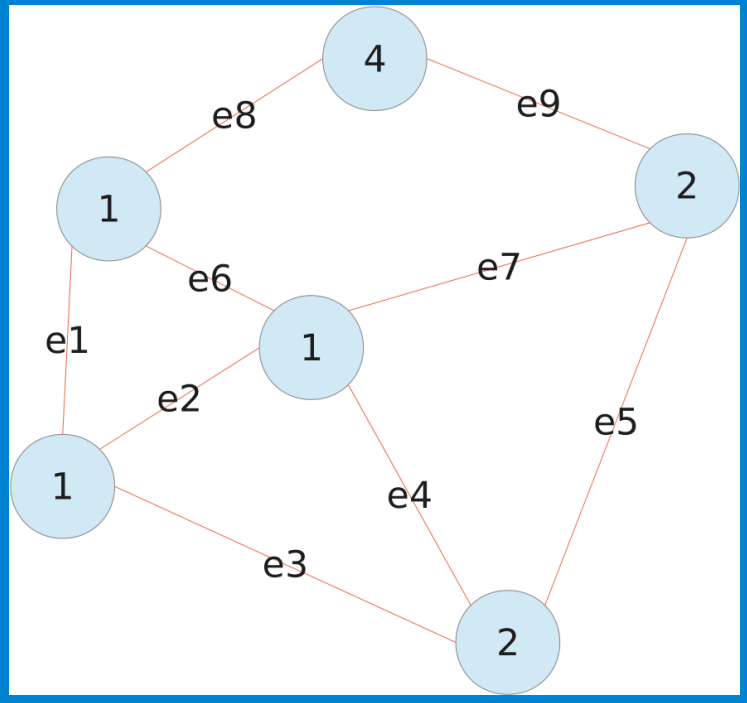
# Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier



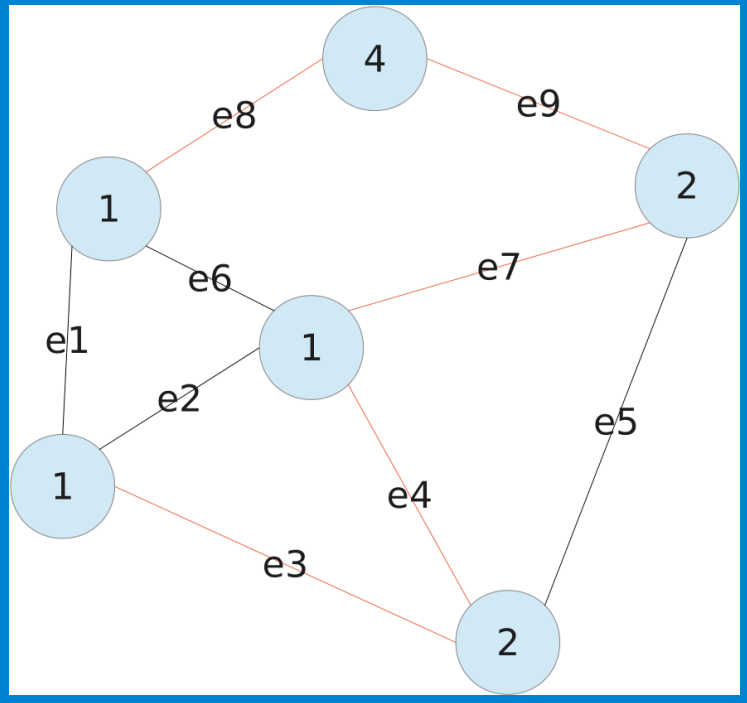
# Gunrock's Traversal Step

- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier



# Gunrock's Traversal Step

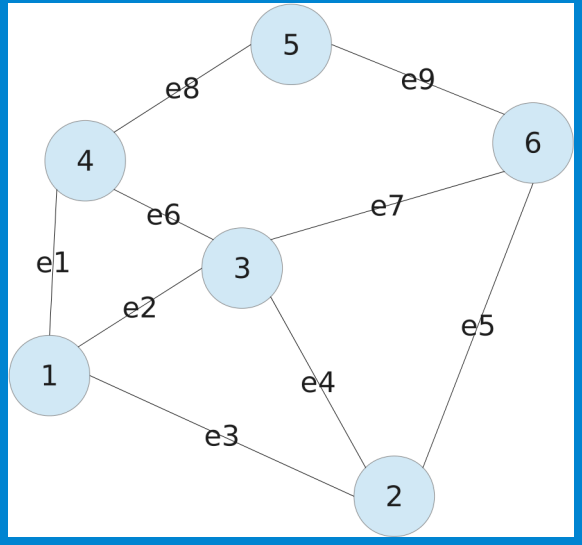
- Advance: visiting the neighbors of the current frontier
- Filter: choosing a subset of the current frontier



# Gunrock's Compute Step

Functors that apply to {edges, vertices}

- “cond” functor: returns a boolean value
- “apply” functor: performs a computation

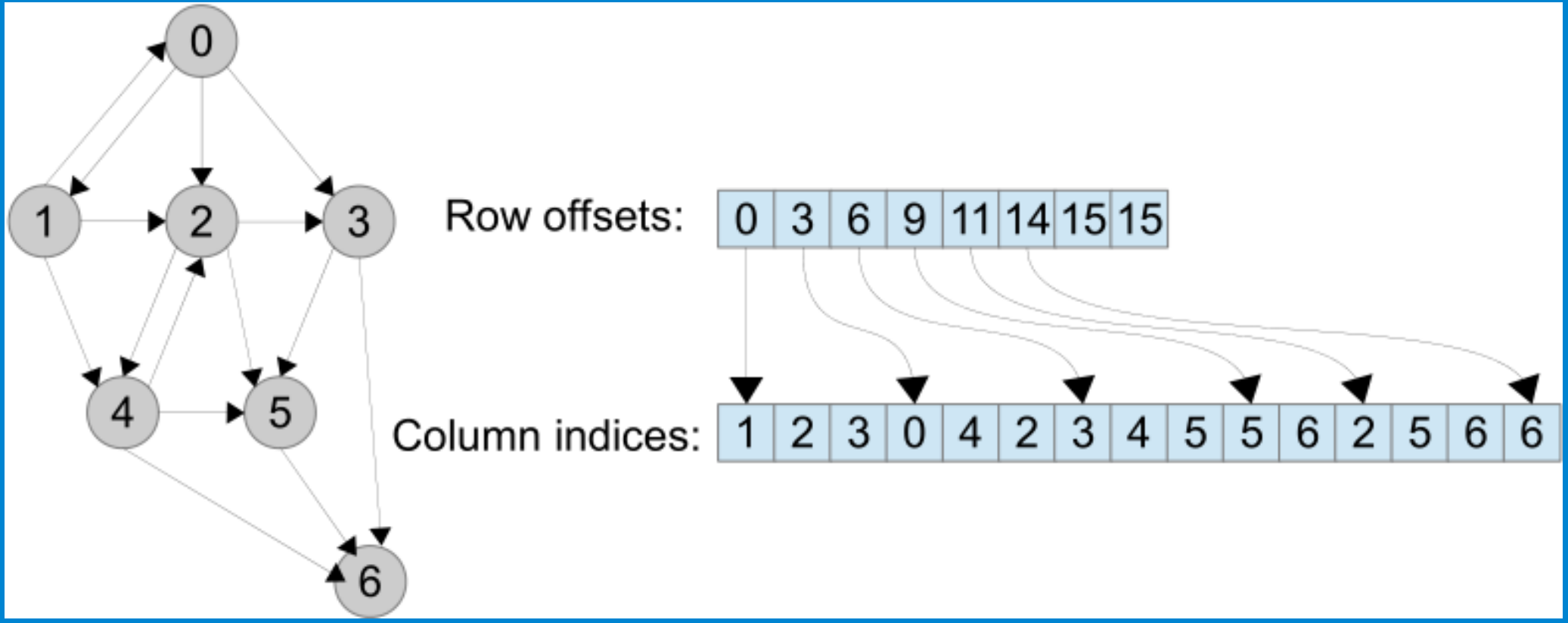


# Graph Primitives in Gunrock (In only Three Files)

- Problem: Initialize the graph data and frontier
- Enactor: GPU kernel entry function which defines a series of operations on frontier
- Functor: User-specified per-node/per-edge computation on frontier

# Performance: Generalized Optimization Strategies

# Graph Data Representation: CSR

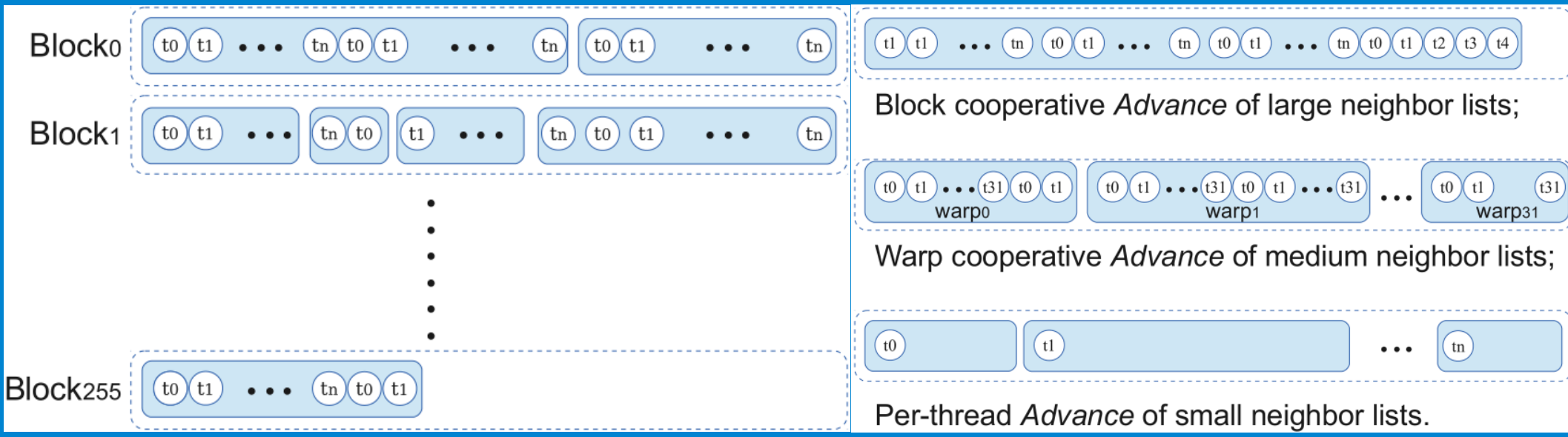


# Workload Mapping and Load-balancing

- Naive method: Let one thread handle the neighbor list of one vertex
- Problem: Highly uneven distribution of node degrees in scale-free graphs

Need load balancing strategy!

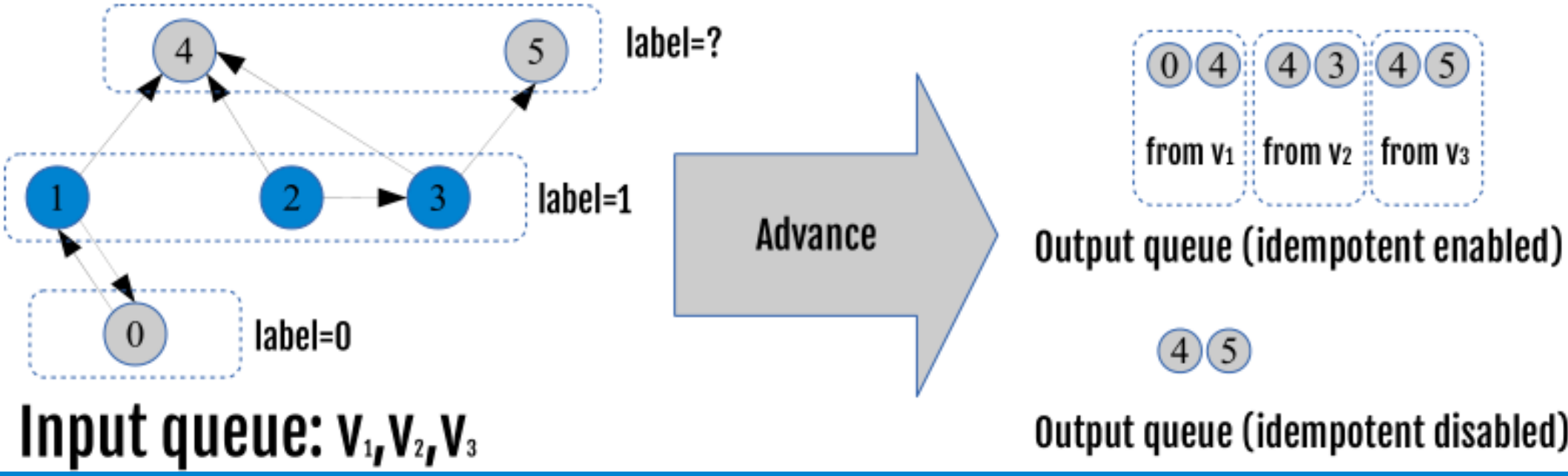
# Workload Mapping and Load-balancing




- Tradeoff between extra processing and load balancing
- A worthwhile extra effort: 2x–20x speedup over non-load balancing library (Medusa v1)

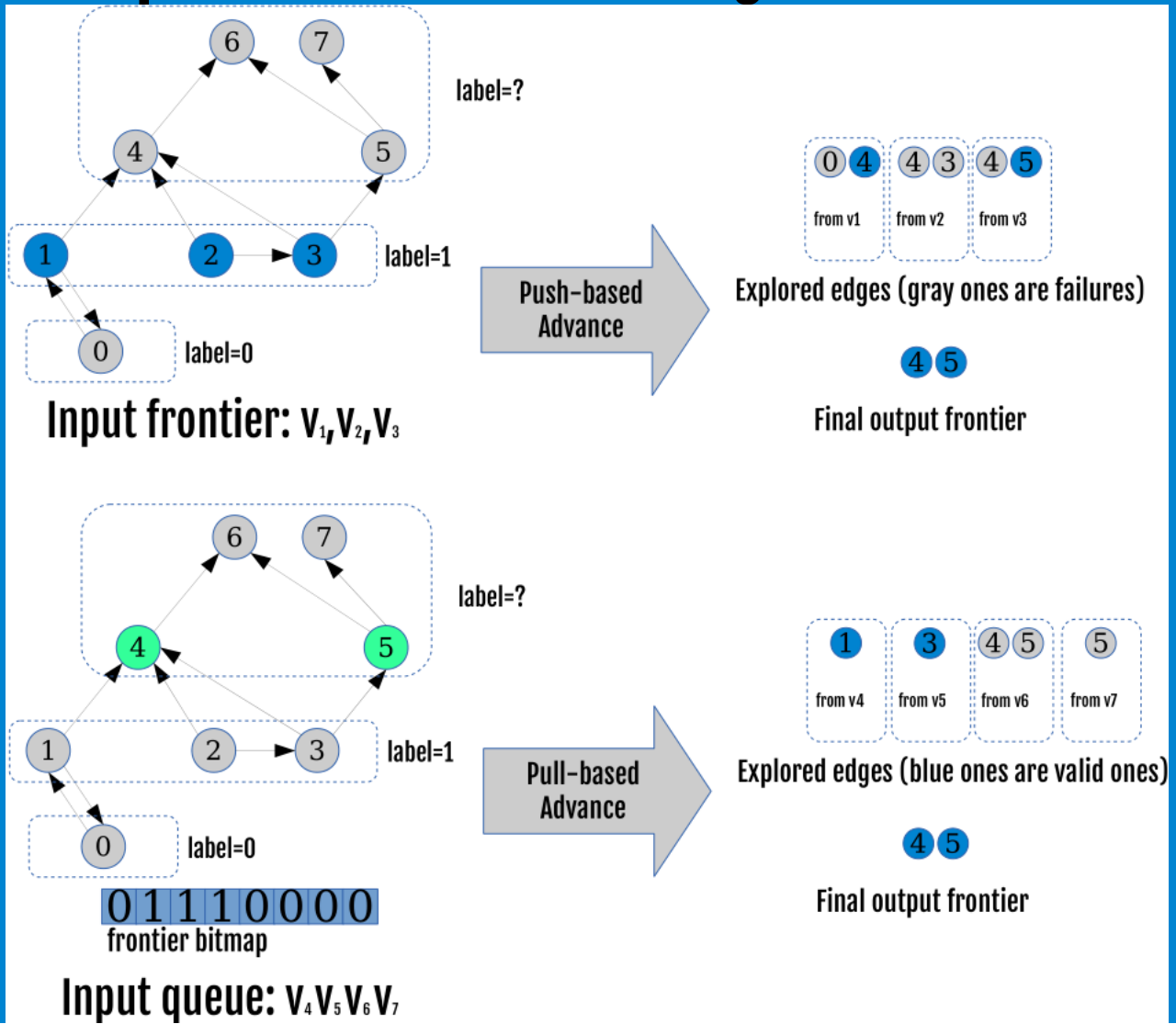
# Data-Centric Abstraction Enables Optimizations

## Idempotent operations (frontier reorganization)



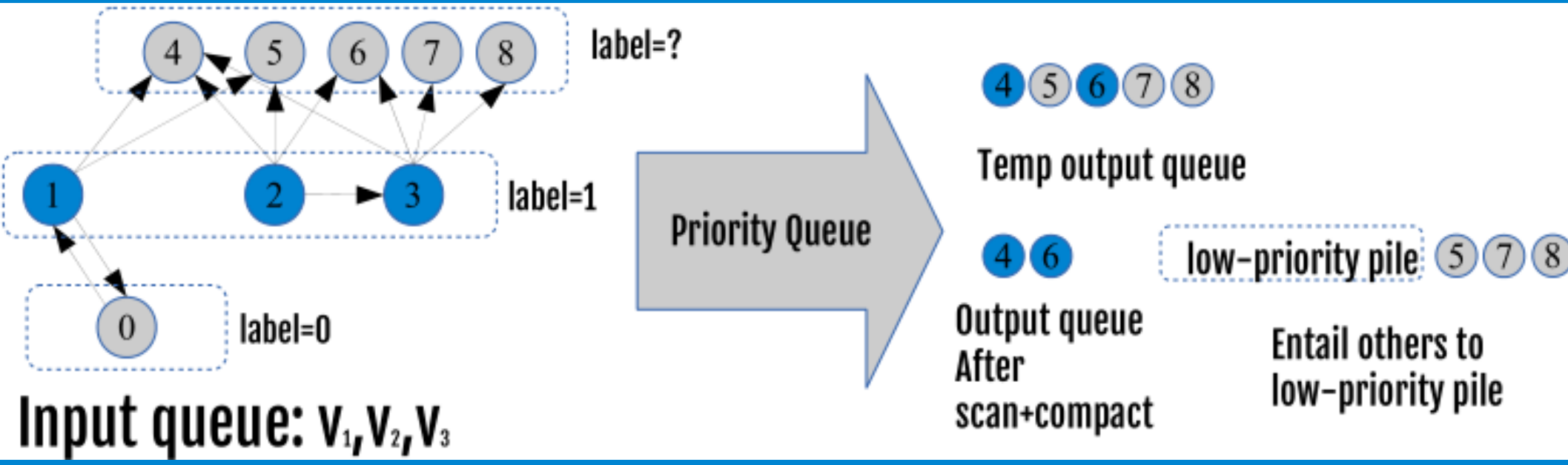
# Data-Centric Abstraction Enables Optimizations

## Pull vs. push operations (frontier generation)



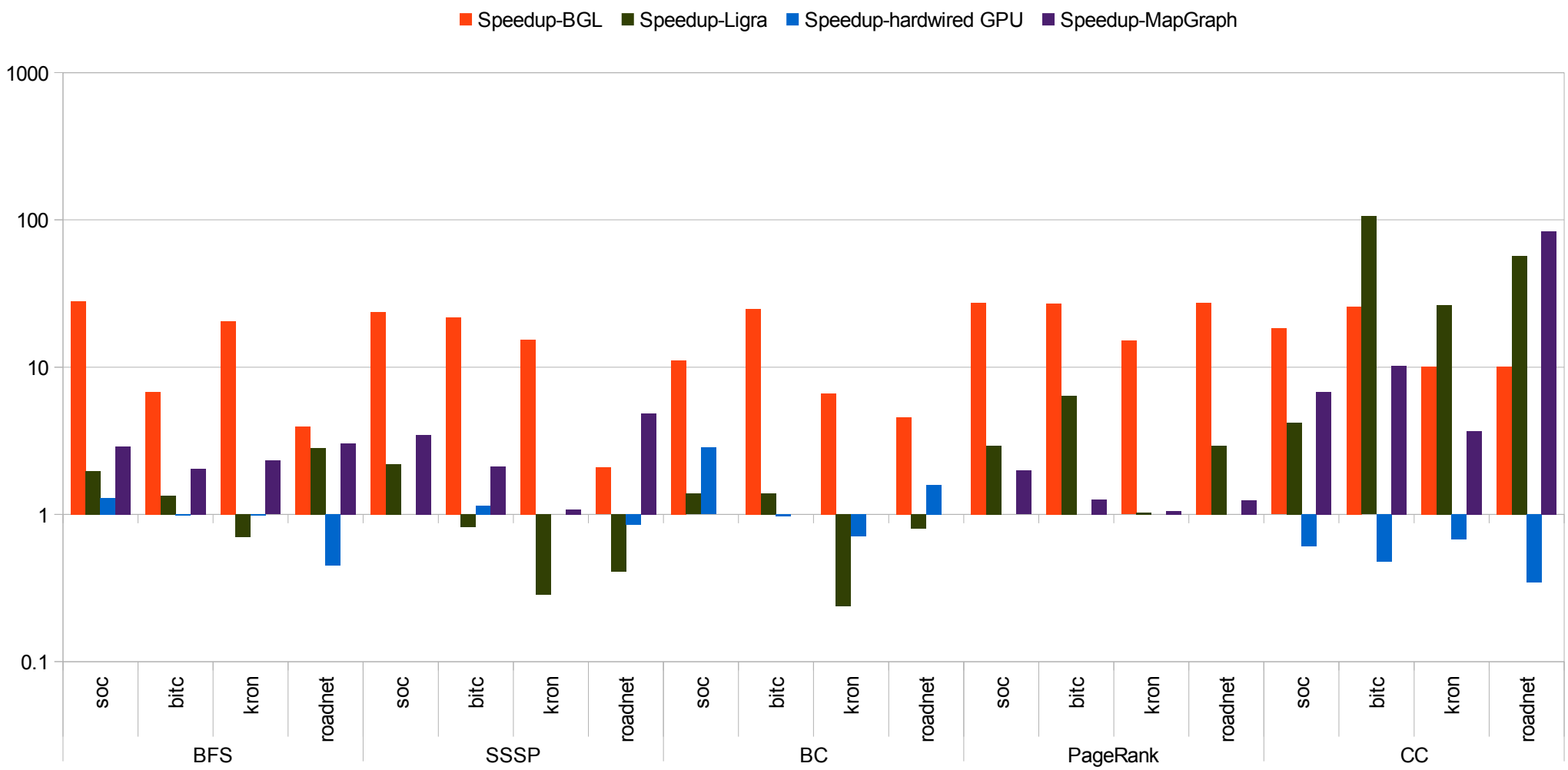
# Data-Centric Abstraction Enables Optimizations

## Priority Queue (frontier reorganization)



# Results, Conclusion, and Future Work

# Performance Against Other Graph Processing Systems



# Expressiveness and Usability

Currently have over 10 graph primitives

- Traversal-based, Node-ranking, Global (connected component, MST)
- LOC under 300 for each primitive

Working on more graph primitives

- Graph coloring, Maximal Independent Set
- Community Detection
- Subgraph Matching

# Future Works

Operations

- Gather reduce
- Global indicator
- Operations on two frontier sets (intersection, join)
- Sampling frontier with different distributions
- Form supervertex

# Future Works

Primitives

- Stochastic Gradient Descent
- Markov-Chain Monte Carlo

Compiler

- Integration with TinkerPop (Popgun!)
- Explorer kernel fusion

# Conclusions

- High-level Abstraction is essential for GPUs to make an impact in graph analytic.
- Gunrock's data-centric, frontier-focused abstraction has good balance between expressiveness and performance

# Questions?

Gunrock Team == Yangzihao Wang, Yuechao Pan, Yuduo Wu, Carl Yang, Leyuan Wang, and our advisor: John D. Owens!