

A Lightweight Infrastructure for Graph Analytics

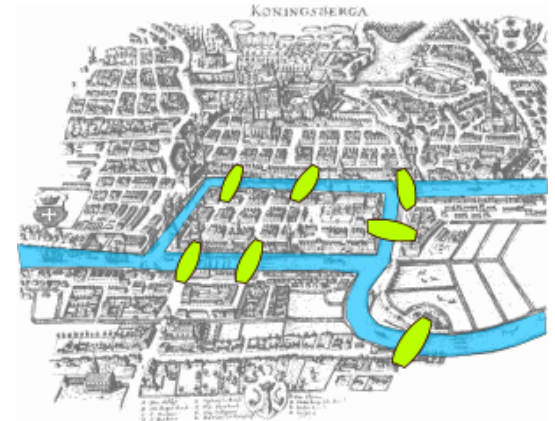
Donald Nguyen

Andrew Lenharth and Keshav Pingali

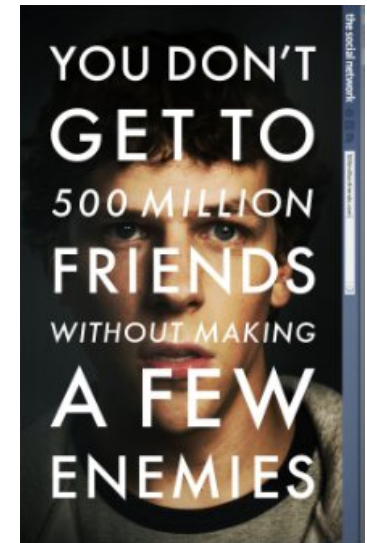
The University of Texas at Austin

What is Graph Analytics?

- Algorithms to compute properties of graphs
 - Connected components, shortest paths, centrality measures, diameter, PageRank, ...
- Many applications
 - Google, path routing, friend recommendations, network analysis
- Difficult to implement on a large scale
 - Data sets are **large**, data accesses are **irregular**
 - Need parallelism and efficient runtimes



Bridges of Königsberg



The Social Network

Contributions

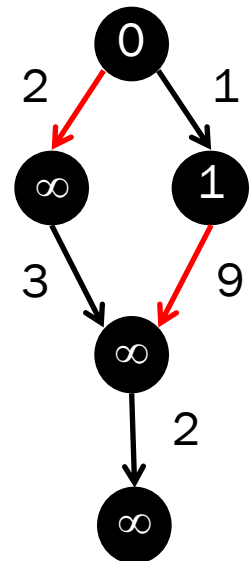
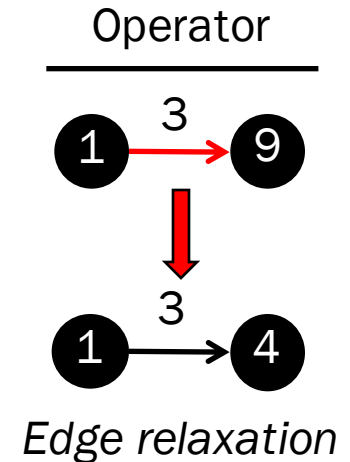
- A lightweight infrastructure for graph analytics based on improvements to the Galois system
 - Example: scalable priority scheduling
- Orders of magnitude faster than third-party graph DSLs for many applications and inputs
 - Galois permits better algorithms to be written
 - Galois infrastructure more scalable

Outline

1. Abstraction for graph analytics applications and implementation in Galois system
2. One piece of infrastructure: priority scheduling
3. Evaluation of Galois system versus graph analytics DSLs

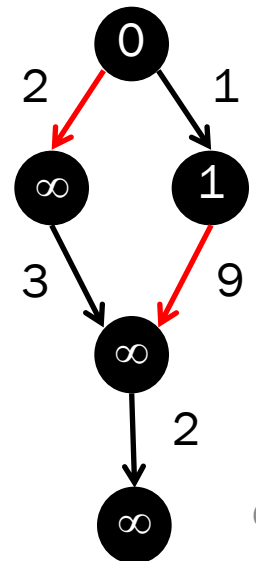
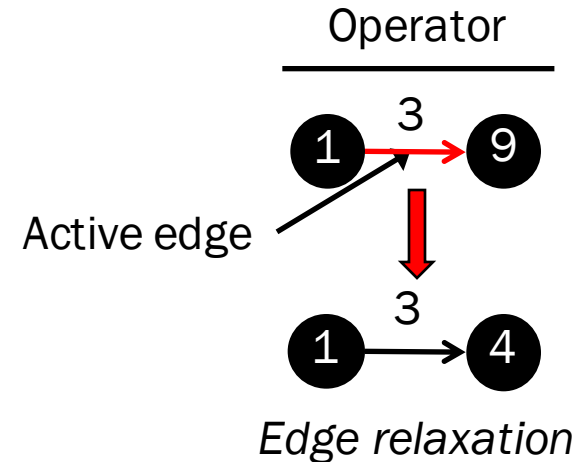
Example: SSSP

- Find the shortest distance from source node to all other nodes in a graph
 - Label nodes with tentative distance
 - Assume non-negative edge weights
- Algorithms
 - Chaotic relaxation $O(2^V)$
 - Bellman-Ford $O(VE)$
 - Dijkstra's algorithm $O(E \log V)$
 - Uses priority queue
 - Δ -stepping
 - Uses sequence of bags to prioritize work
 - $\Delta=1, O(E \log V)$
 - $\Delta=\infty, O(VE)$
- Different algorithms are different schedules for applying relaxations
 - SSSP needs **priority scheduling** for work efficiency



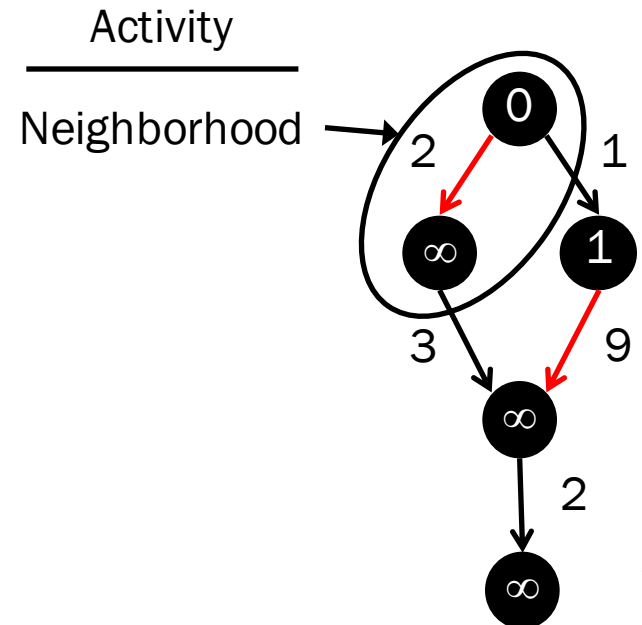
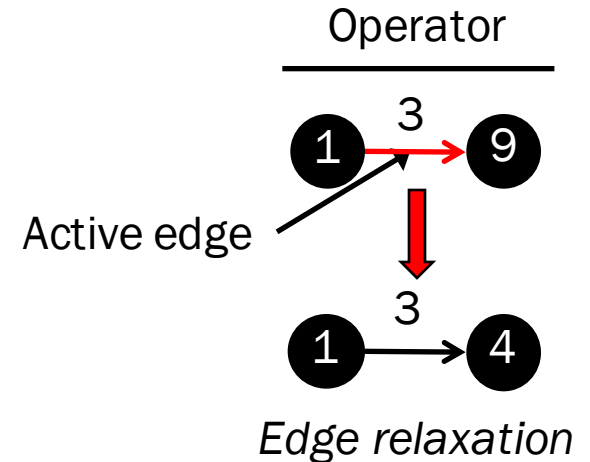
Example: SSSP

- Find the shortest distance from source node to all other nodes in a graph
 - Label nodes with tentative distance
 - Assume non-negative edge weights
- Algorithms
 - Chaotic relaxation $O(2^V)$
 - Bellman-Ford $O(VE)$
 - Dijkstra's algorithm $O(E \log V)$
 - Uses priority queue
 - Δ -stepping
 - Uses sequence of bags to prioritize work
 - $\Delta=1, O(E \log V)$
 - $\Delta=\infty, O(VE)$
- Different algorithms are different schedules for applying relaxations
 - SSSP needs **priority scheduling** for work efficiency




Example: SSSP

- Find the shortest distance from source node to all other nodes in a graph
 - Label nodes with tentative distance
 - Assume non-negative edge weights
- Algorithms
 - Chaotic relaxation $O(2^V)$
 - Bellman-Ford $O(VE)$
 - Dijkstra's algorithm $O(E \log V)$
 - Uses priority queue
 - Δ -stepping
 - Uses sequence of bags to prioritize work
 - $\Delta=1$, $O(E \log V)$
 - $\Delta=\infty$, $O(VE)$
- Different algorithms are different schedules for applying relaxations
 - SSSP needs **priority scheduling** for work efficiency



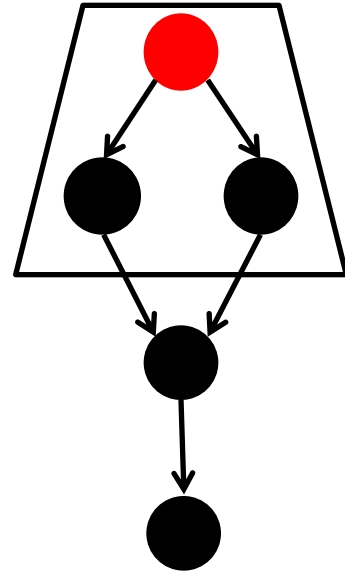
Parallel Program = Operator + Schedule + Parallel Data Structure



Algorithm

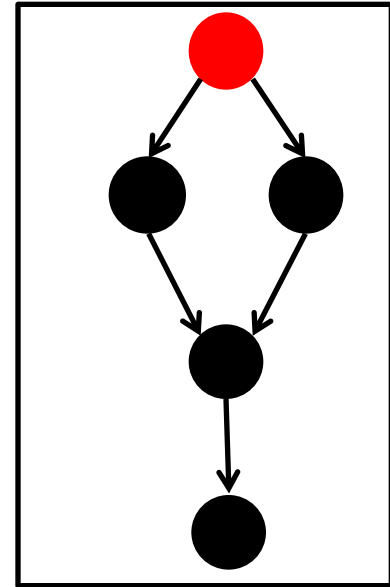
Parallel Program = $\underbrace{\text{Operator} + \text{Schedule}}_{\text{Algorithm}} + \text{Parallel Data Structure}$

- What is the operator?
 - Ligra, PowerGraph: only vertex programs
 - Galois: Unrestricted, may even **morph** graph by adding/removing nodes and edges
- Where/When does it execute?
 - Autonomous scheduling: activities execute **transactionally**
 - Coordinated scheduling: activities execute **in rounds**
 - Read values refer to previous rounds
 - Multiple updates to the same location are resolved with reduction, etc.



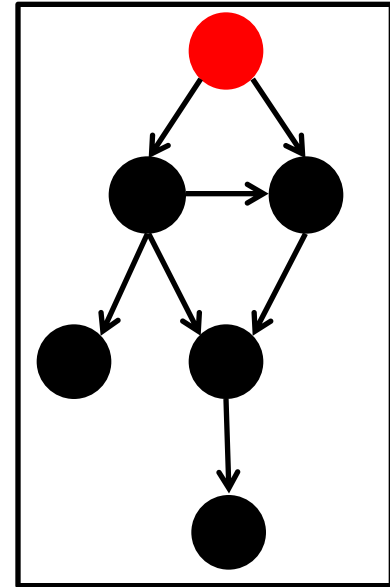
Parallel Program = $\underbrace{\text{Operator} + \text{Schedule}}_{\text{Algorithm}} + \text{Parallel Data Structure}$

- What is the operator?
 - Ligra, PowerGraph: only vertex programs
 - Galois: Unrestricted, may even **morph** graph by adding/removing nodes and edges
- Where/When does it execute?
 - Autonomous scheduling: activities execute **transactionally**
 - Coordinated scheduling: activities execute **in rounds**
 - Read values refer to previous rounds
 - Multiple updates to the same location are resolved with reduction, etc.



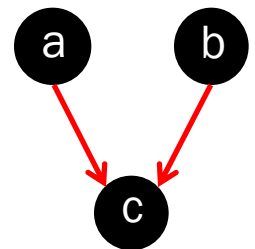
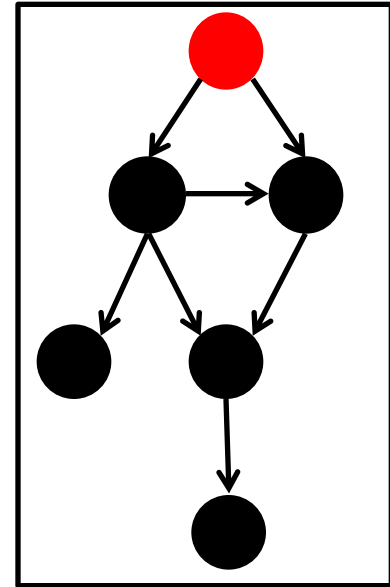
Parallel Program = $\underbrace{\text{Operator} + \text{Schedule}}_{\text{Algorithm}} + \text{Parallel Data Structure}$

- What is the operator?
 - Ligra, PowerGraph: only vertex programs
 - Galois: Unrestricted, may even **morph** graph by adding/removing nodes and edges
- Where/When does it execute?
 - Autonomous scheduling: activities execute **transactionally**
 - Coordinated scheduling: activities execute **in rounds**
 - Read values refer to previous rounds
 - Multiple updates to the same location are resolved with reduction, etc.



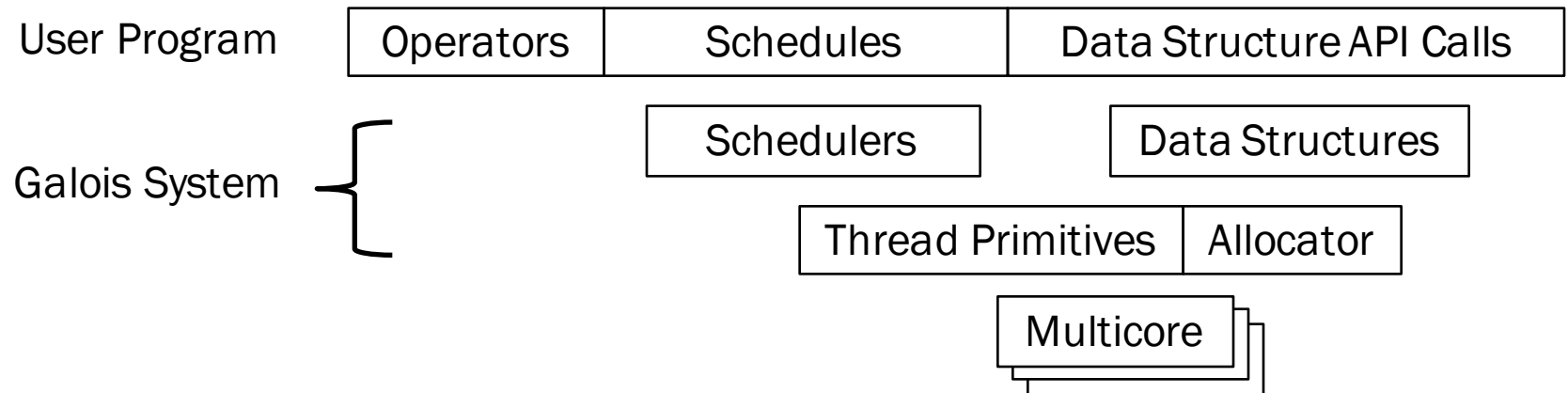
Parallel Program = $\underbrace{\text{Operator} + \text{Schedule}}_{\text{Algorithm}} + \text{Parallel Data Structure}$

- What is the operator?
 - Ligra, PowerGraph: only vertex programs
 - Galois: Unrestricted, may even **morph** graph by adding/removing nodes and edges
- Where/When does it execute?
 - Autonomous scheduling: activities execute **transactionally**
 - Coordinated scheduling: activities execute **in rounds**
 - Read values refer to previous rounds
 - Multiple updates to the same location are resolved with reduction, etc.



Galois System

Parallel Program = Operator + Schedule + Parallel Data Structure



Outline

1. Abstraction for graph analytics applications and implementation in Galois system
2. One piece of infrastructure: priority scheduling
3. Evaluation of Galois system versus DSLs

Galois Infrastructure for Scheduling

- Library of different schedulers
 - FIFO, LIFO, Priority, ...
 - DSL for specifying scheduling policies
- Nguyen and Pingali (ASPLOS '11)

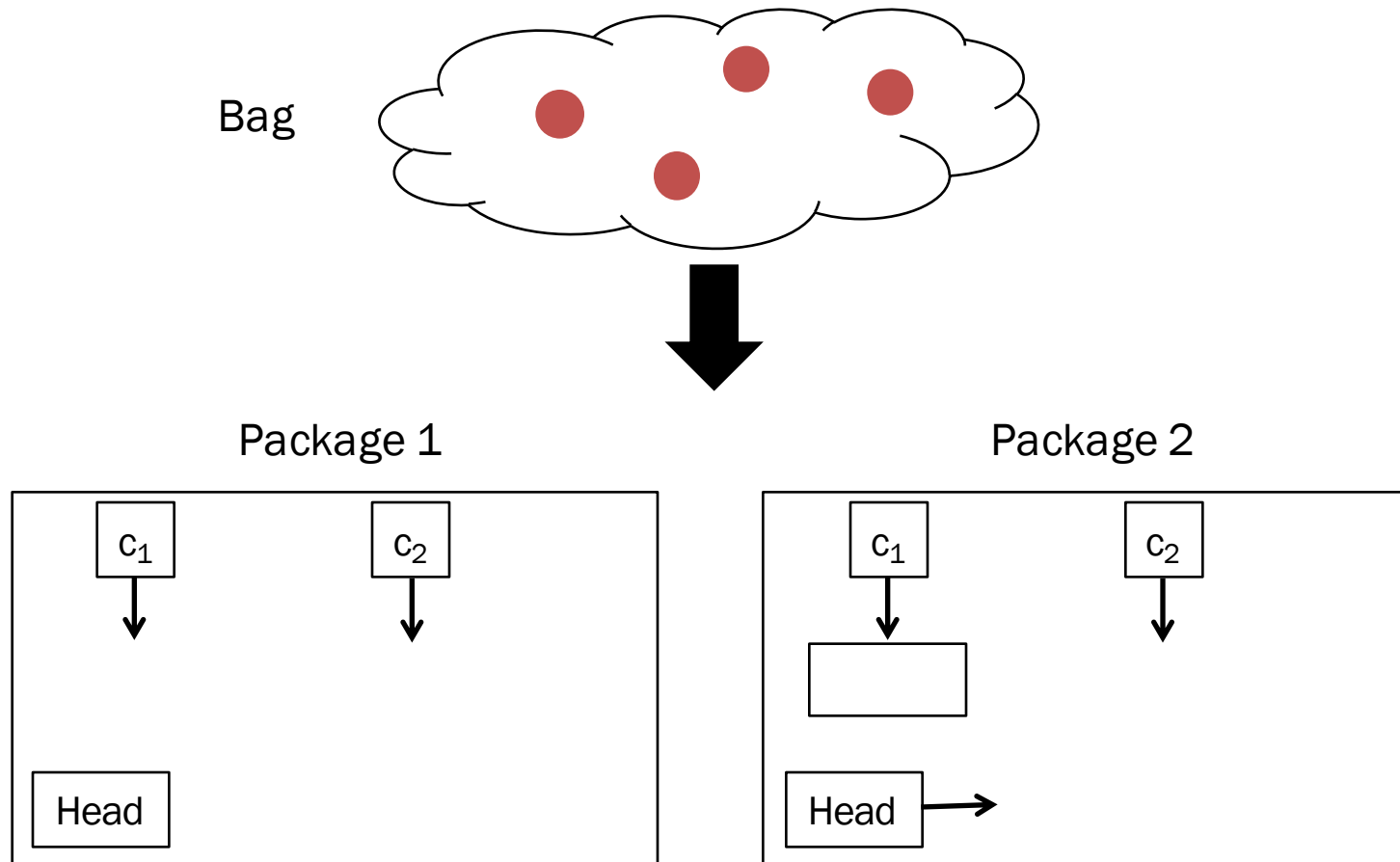
Priority Scheduling

- Concurrent priority queue
 - Implemented in TBB, `java.concurrent`
 - Does not scale: our tasks are very fine-grain
- Sequence of bags
 - One bag per priority level
 - High overhead for finding non-empty, urgent-priority bags

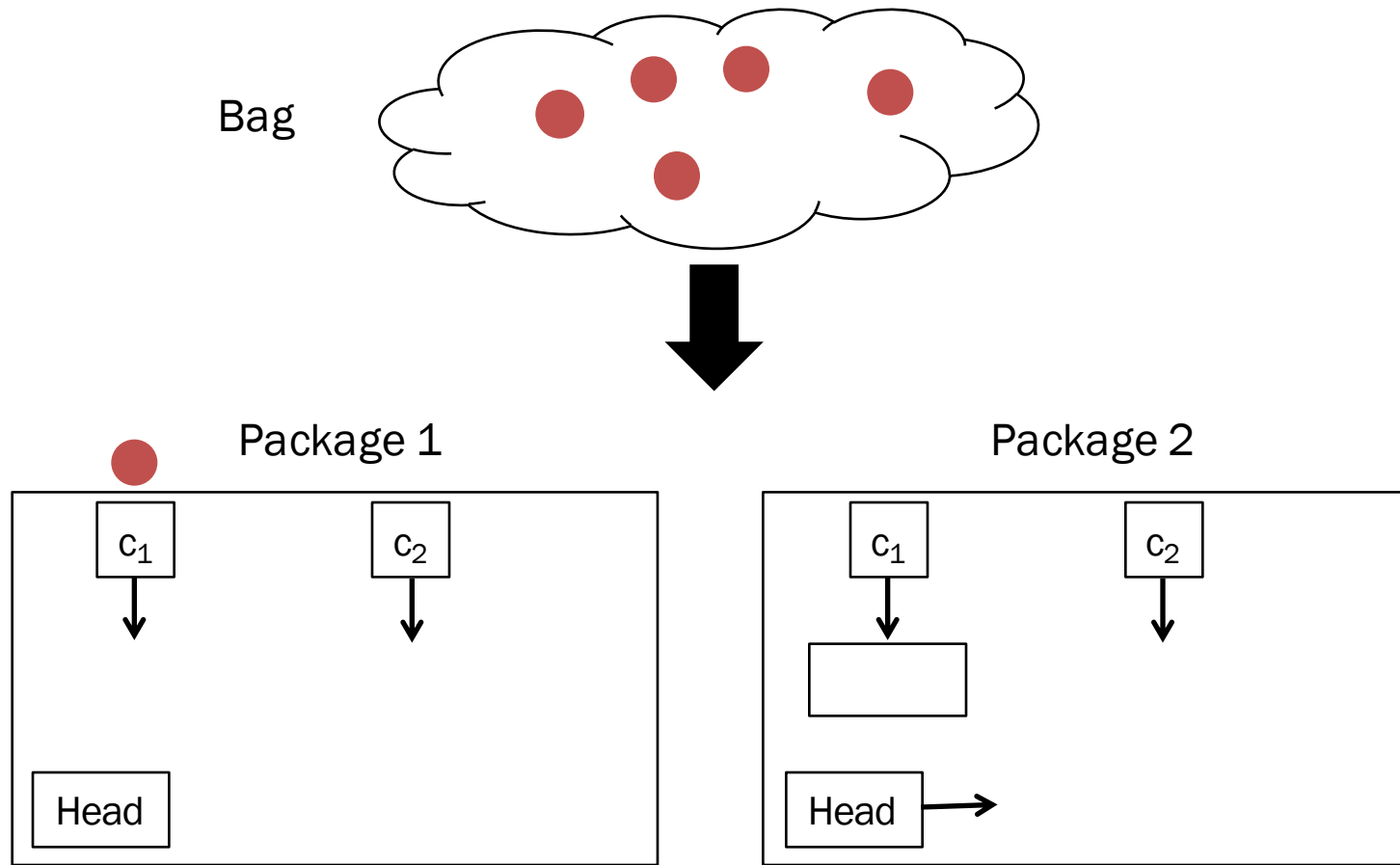
Ordered by Metric (OBIM)

- Sparse sequence of Bags
 - One Bag per priority level
 - To reduce synchronization
 - Sequence is replicated among cores
 - Bag is distributed between cores
 - To reduce overhead of finding work
 - A reduction tree over machine topology to get estimate of current urgent priority

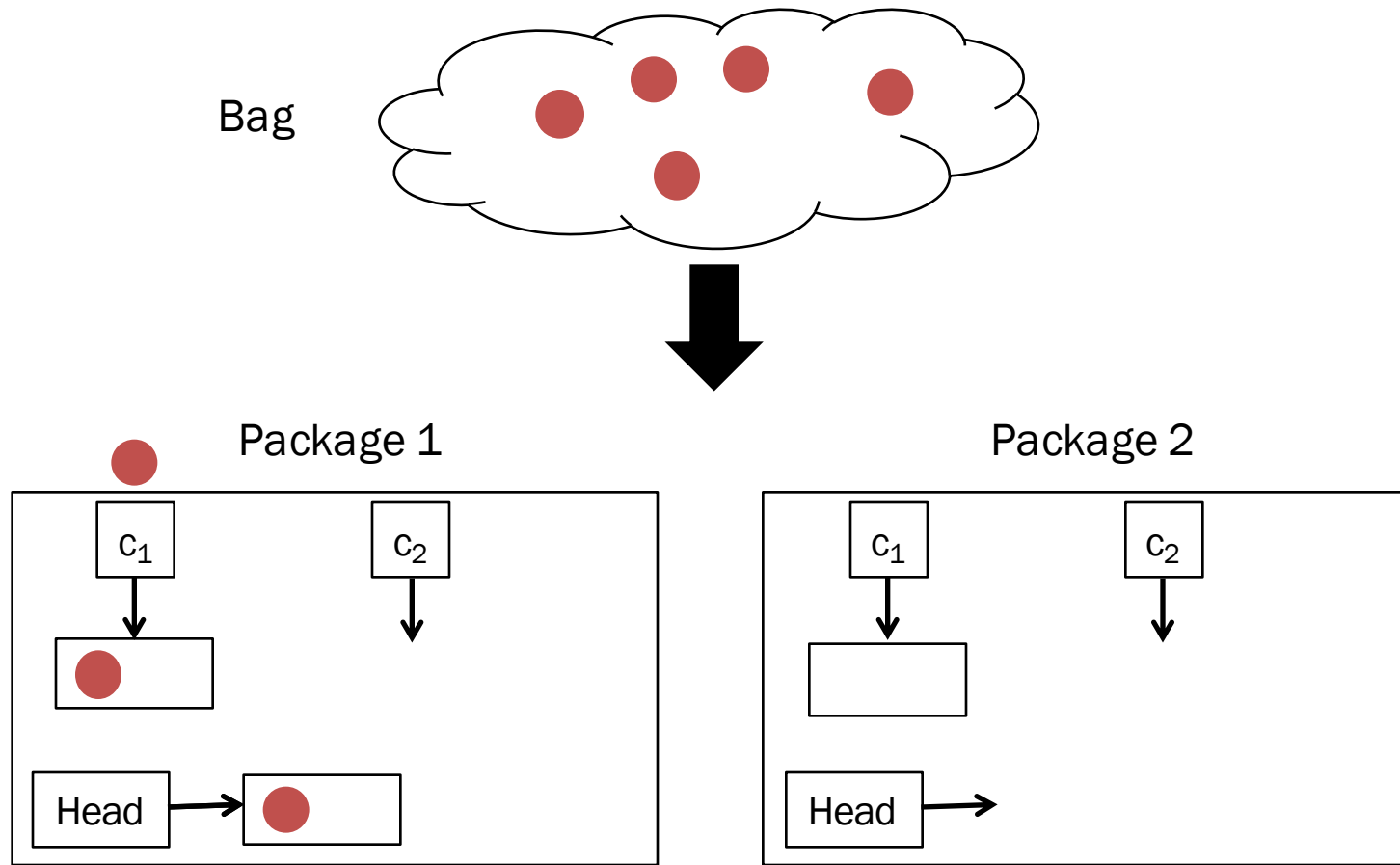
Scheduling Bag



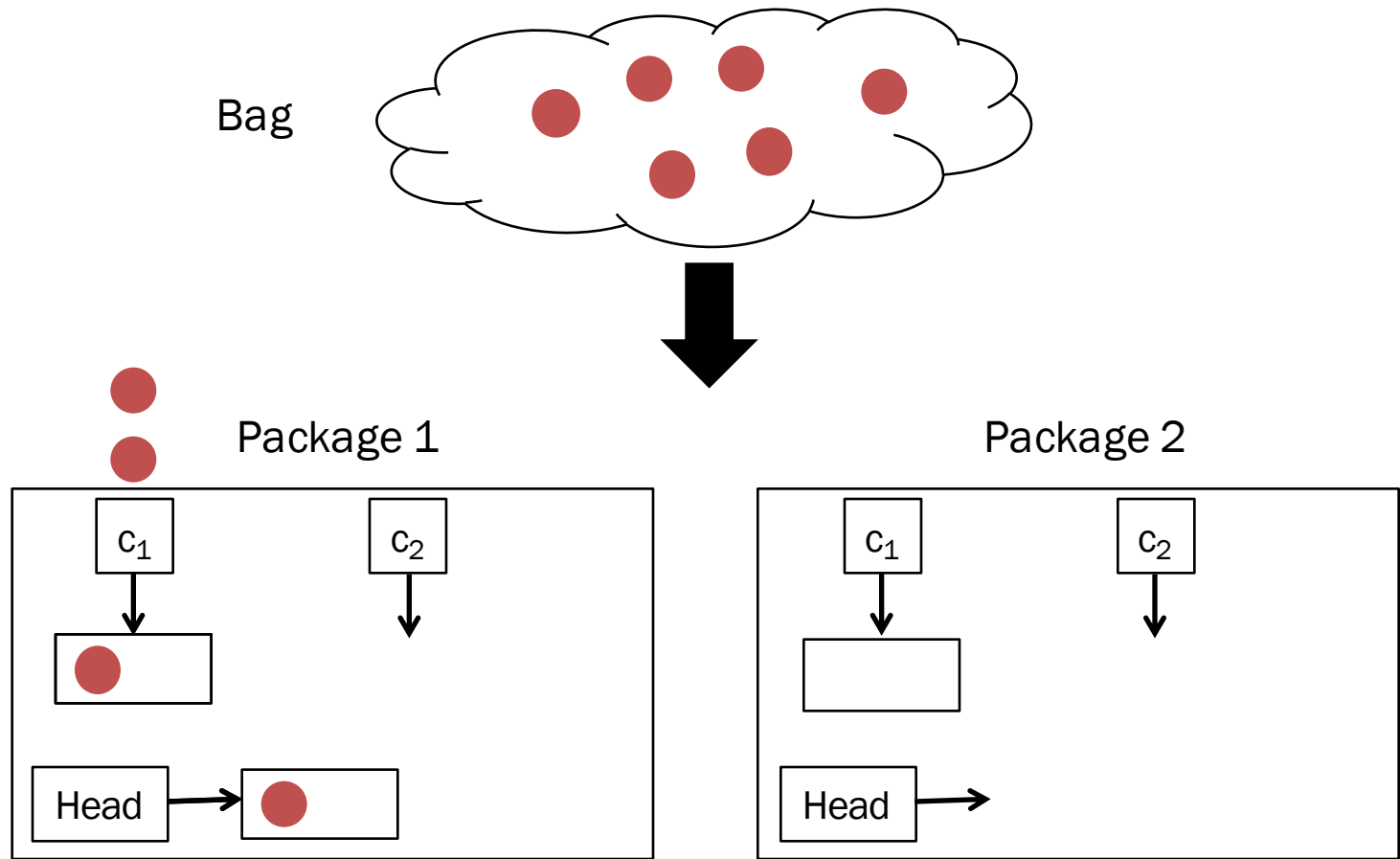
Scheduling Bag



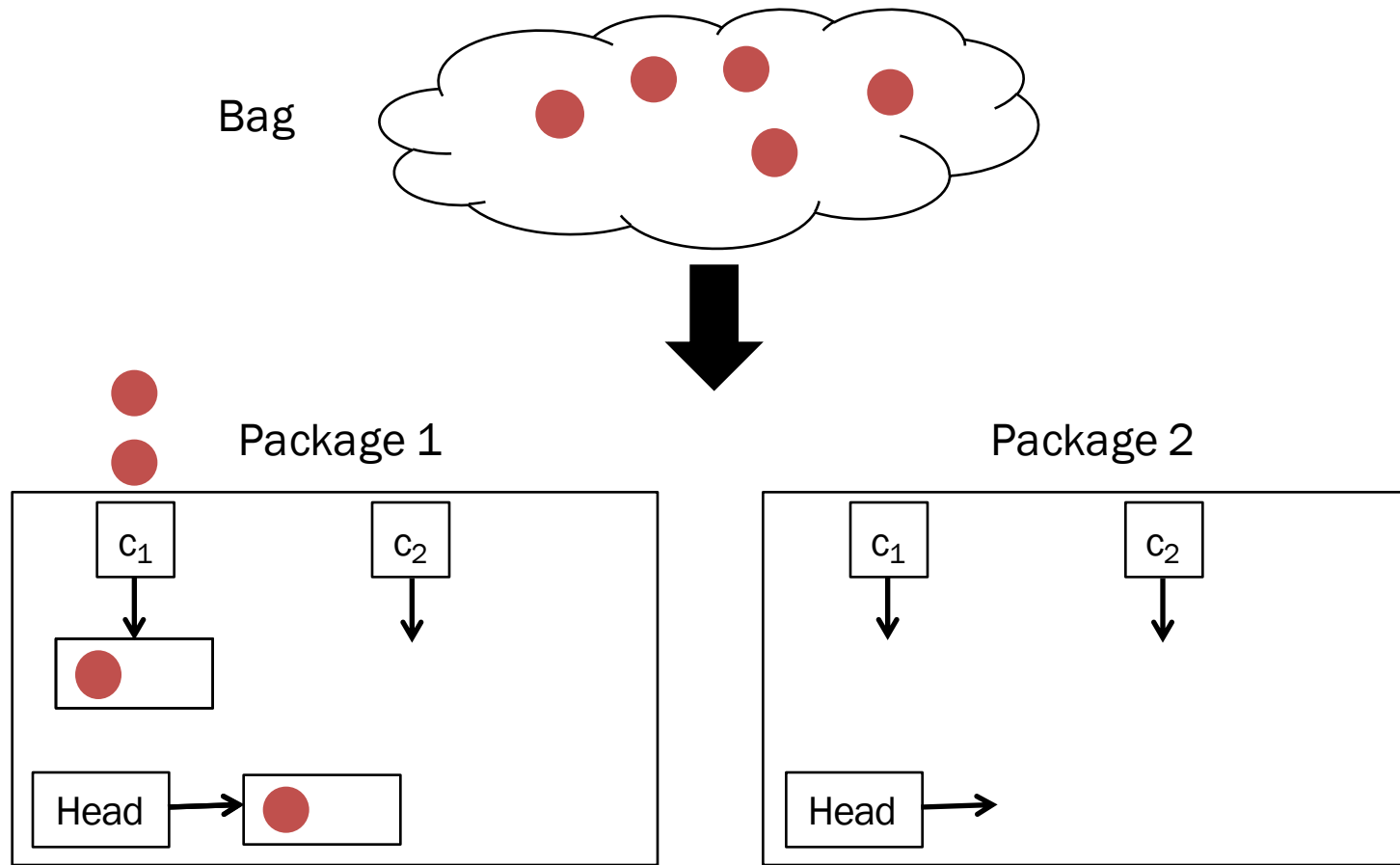
Scheduling Bag



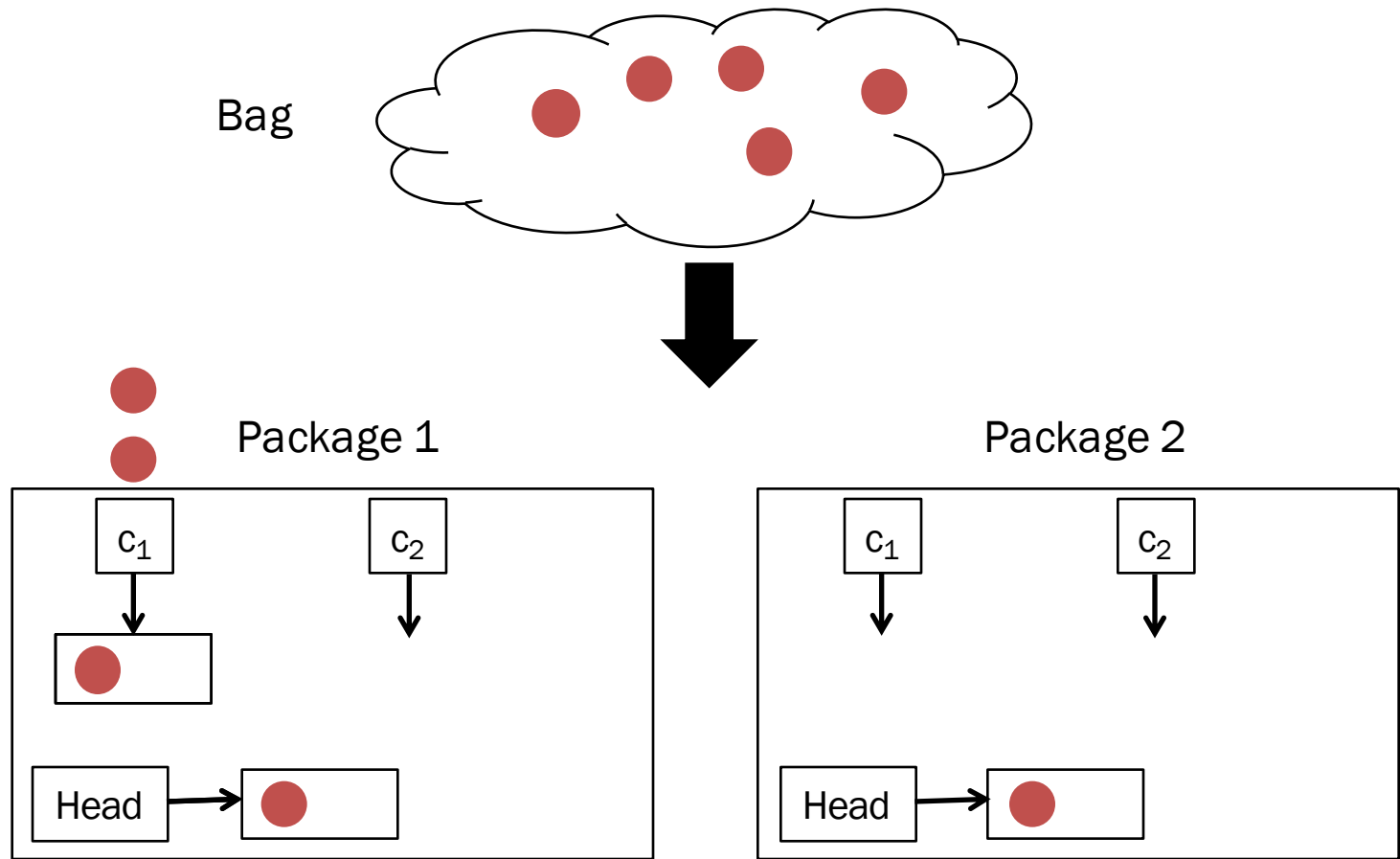
Scheduling Bag



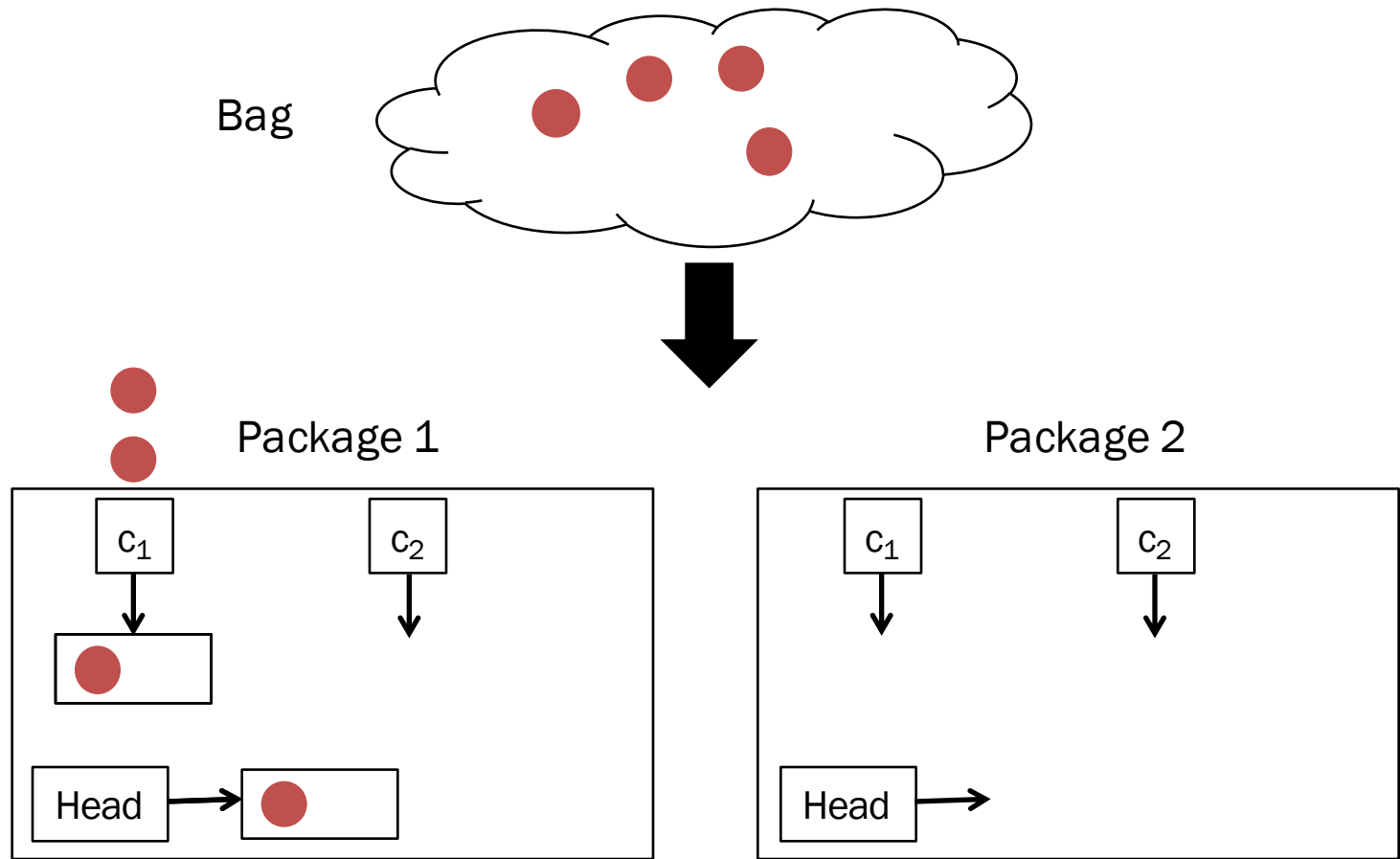
Scheduling Bag



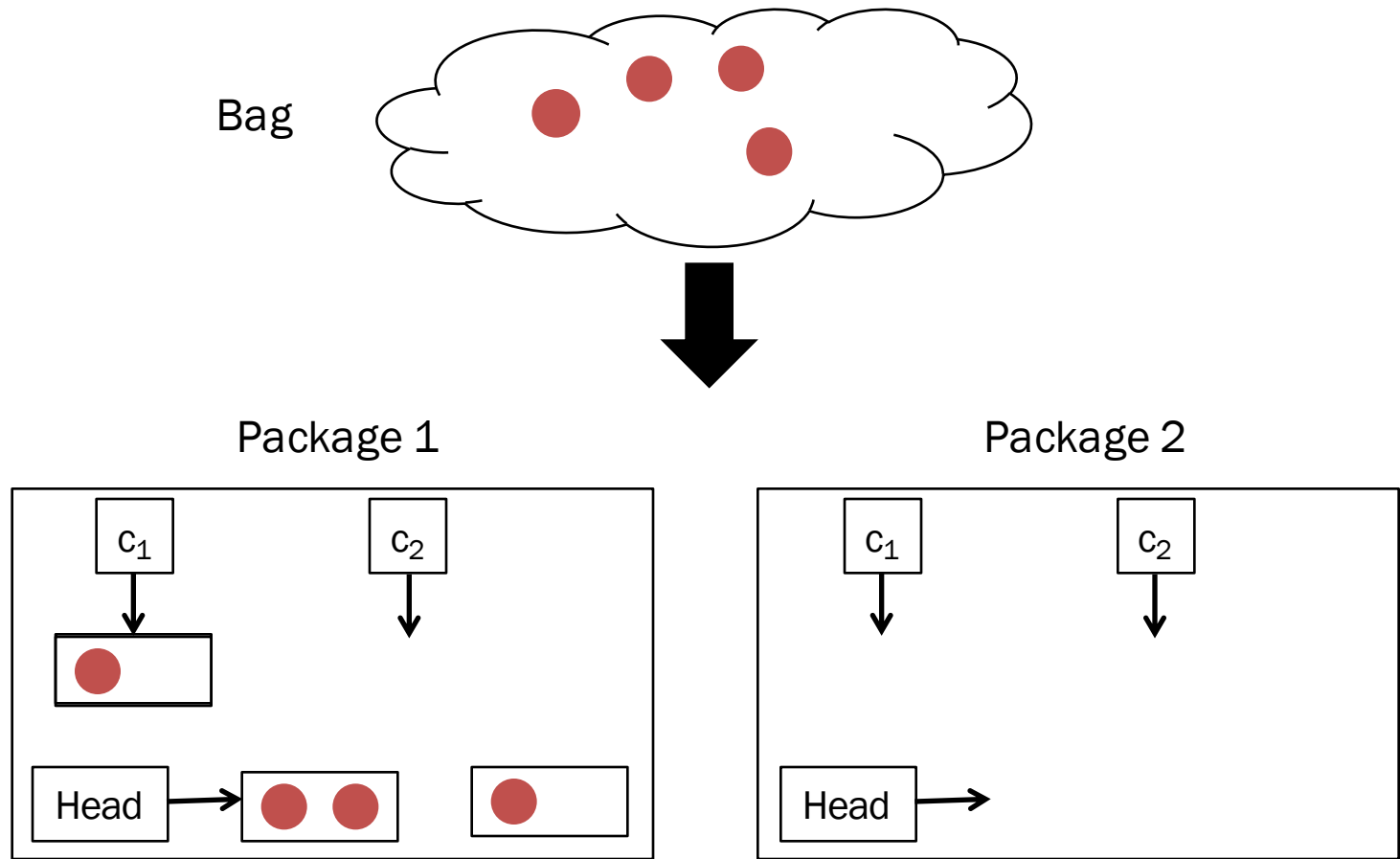
Scheduling Bag



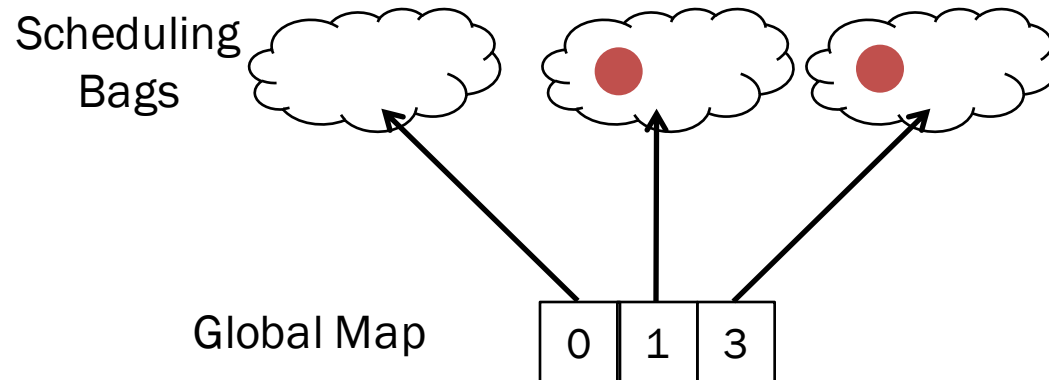
Scheduling Bag



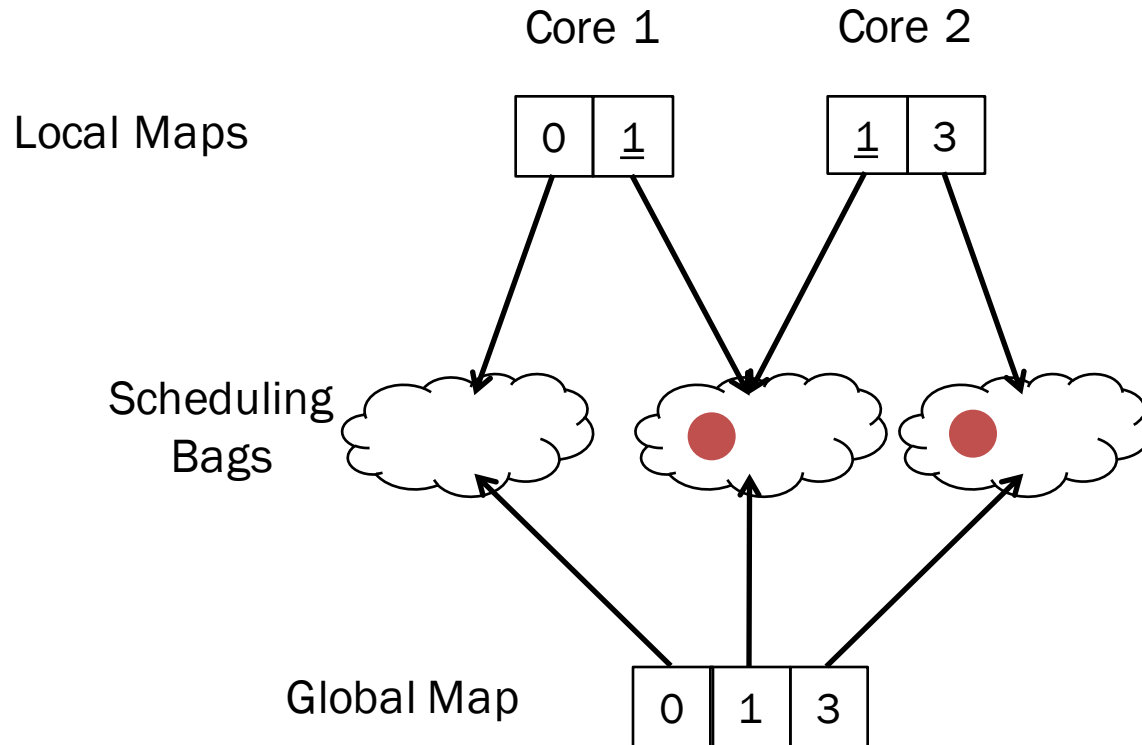
Scheduling Bag



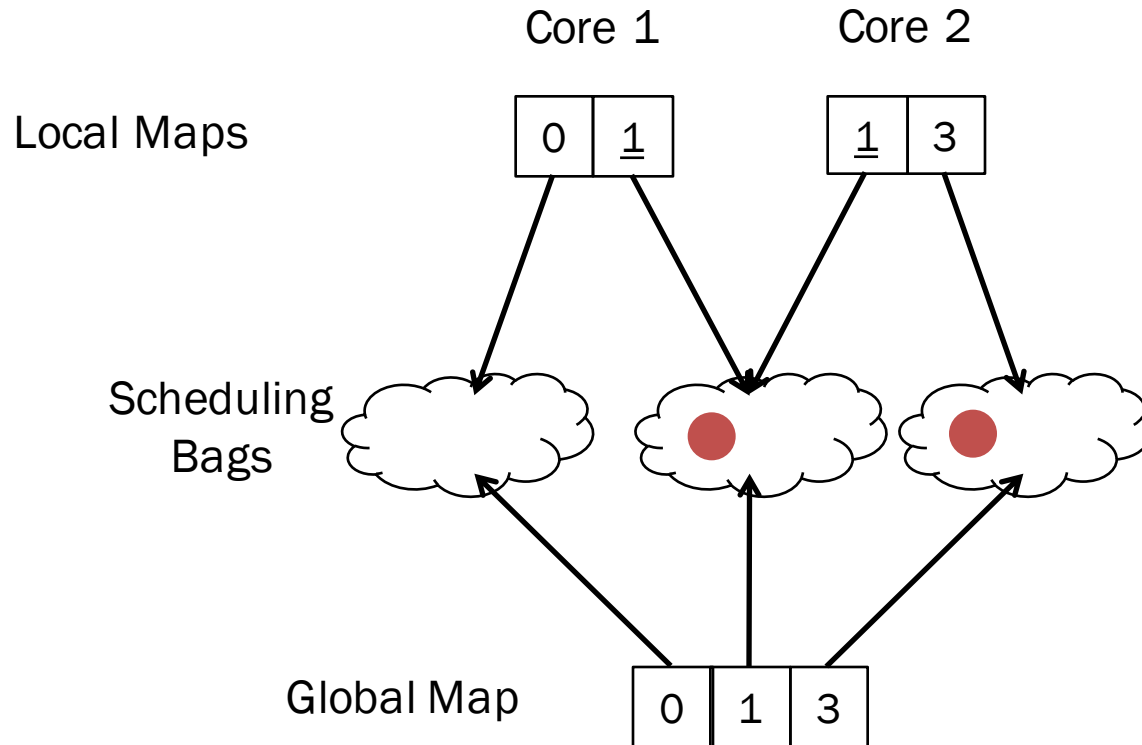
Priority Map



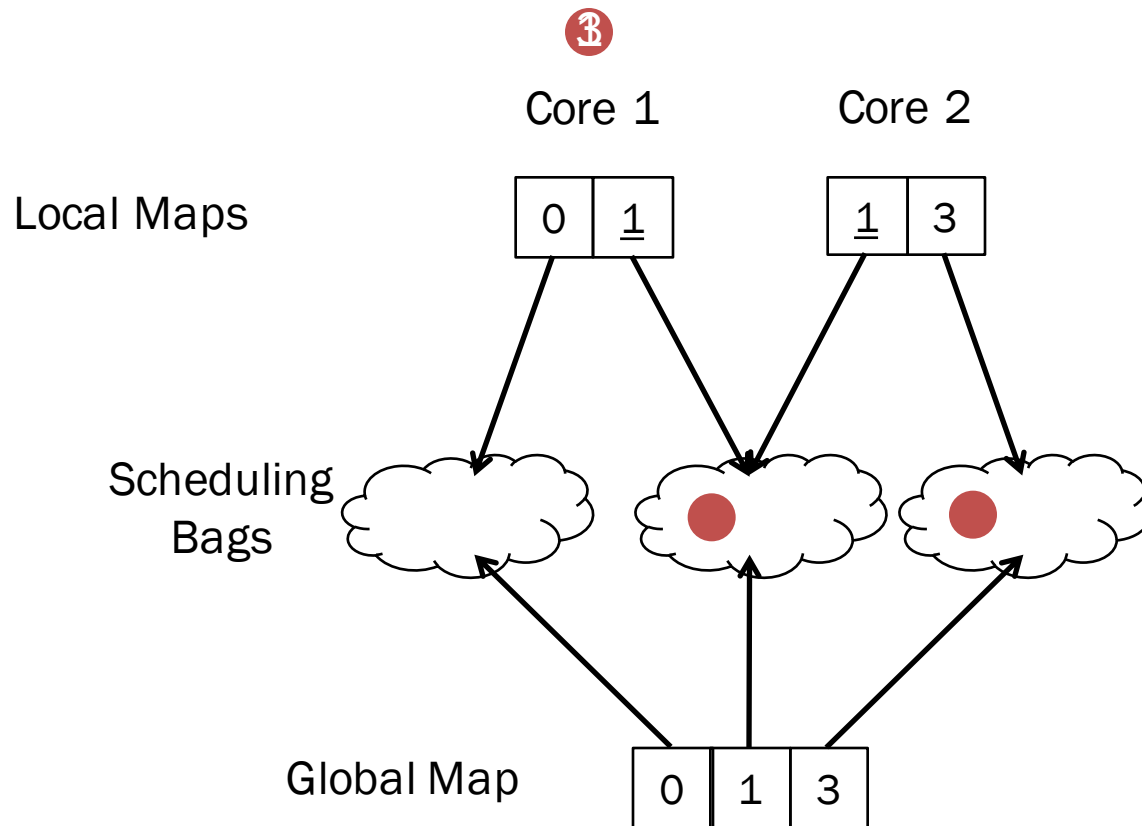
Priority Map



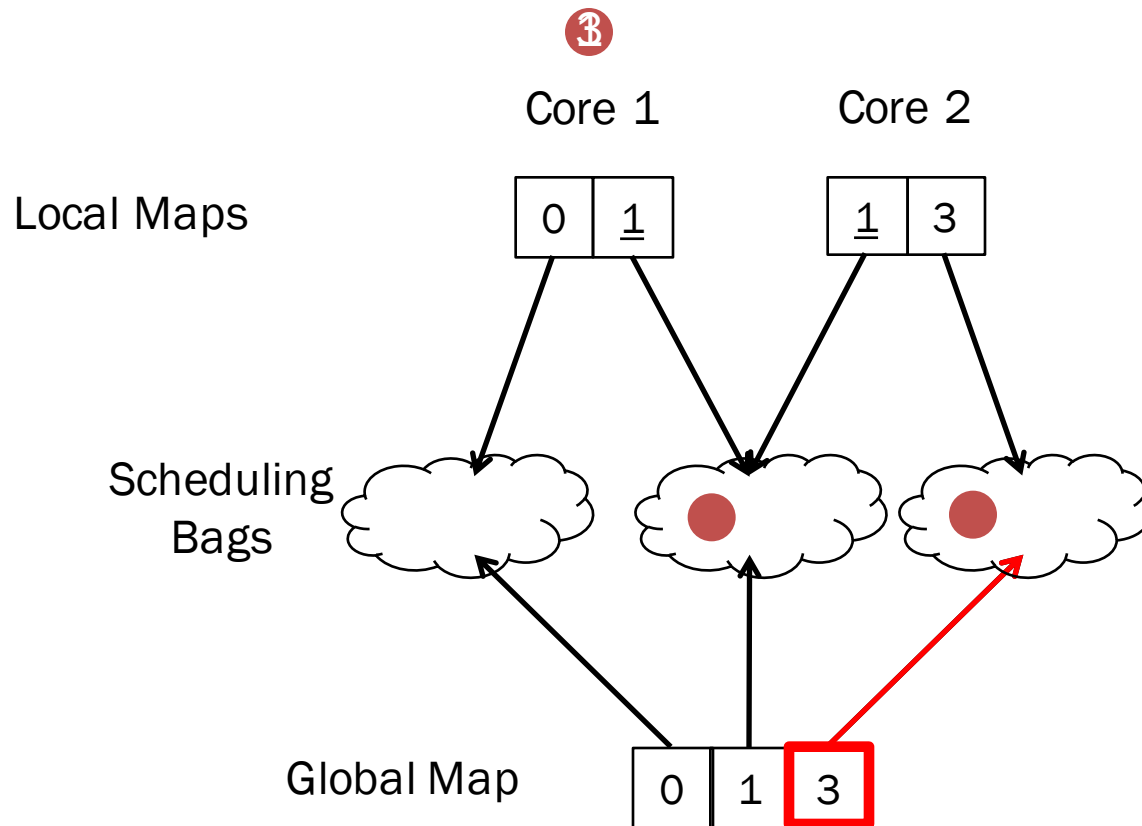
Priority Map



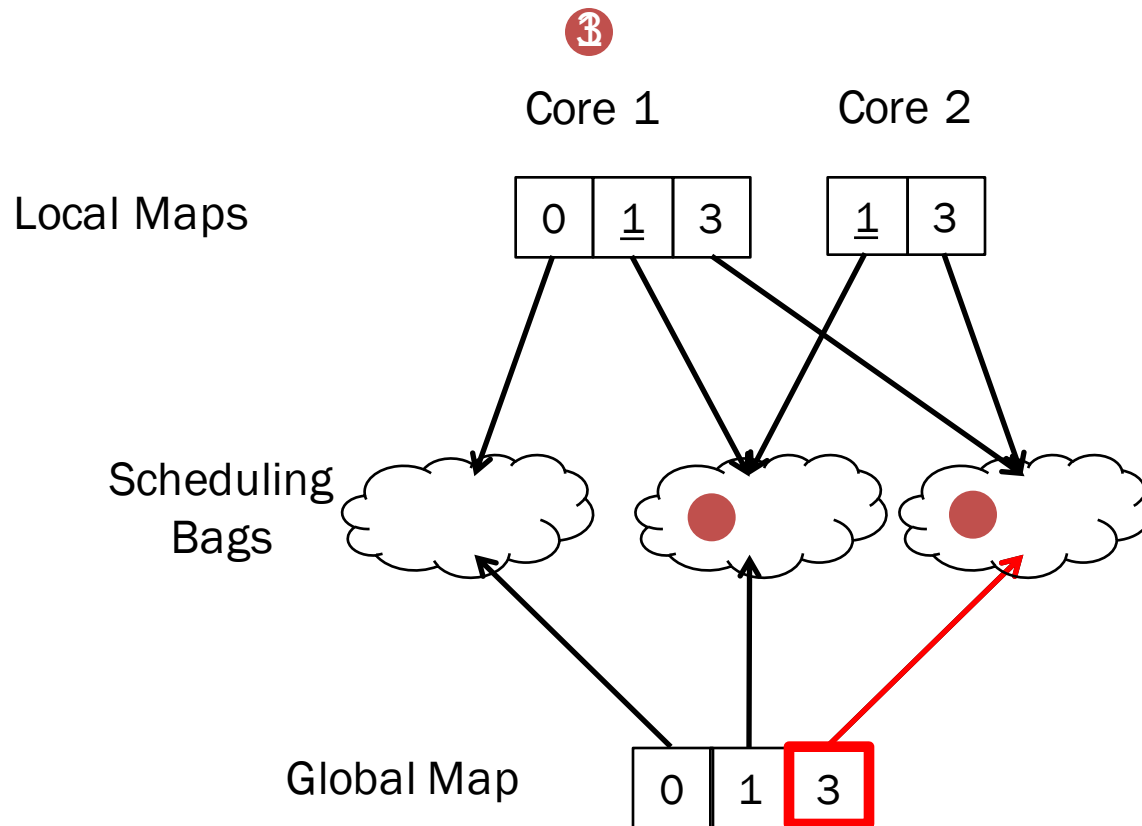
Priority Map



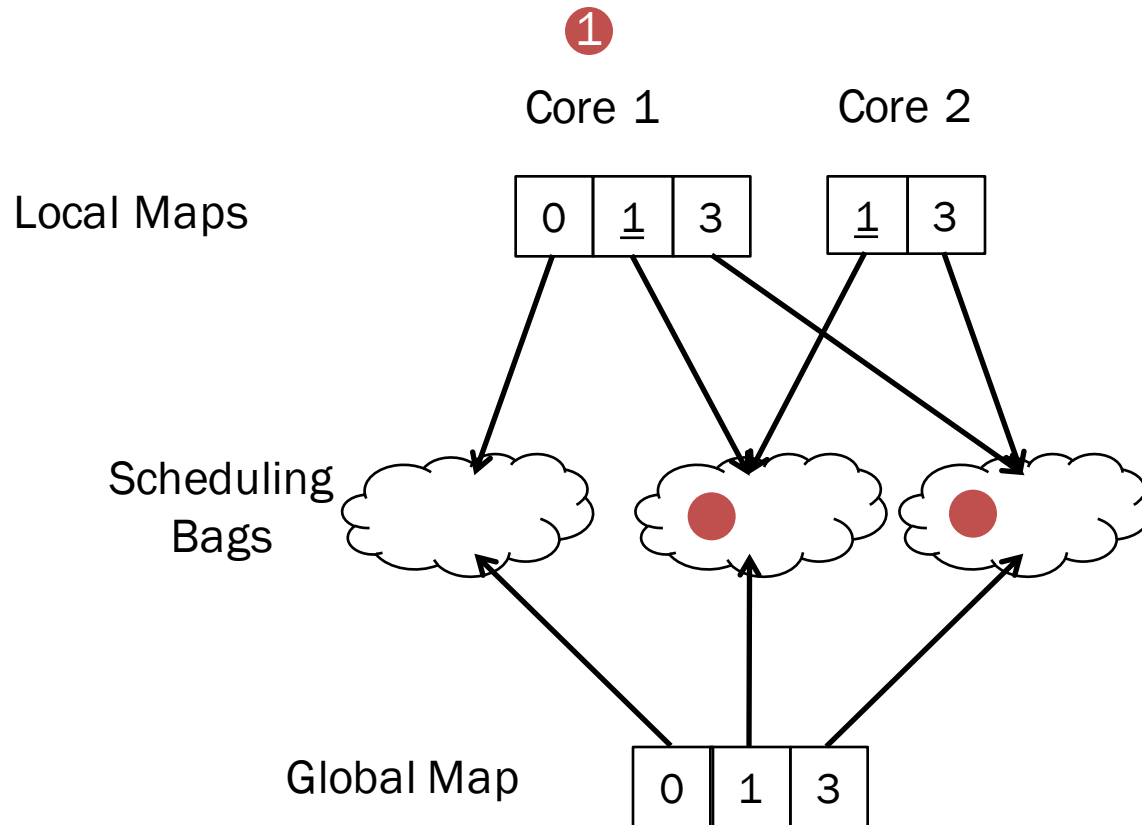
Priority Map



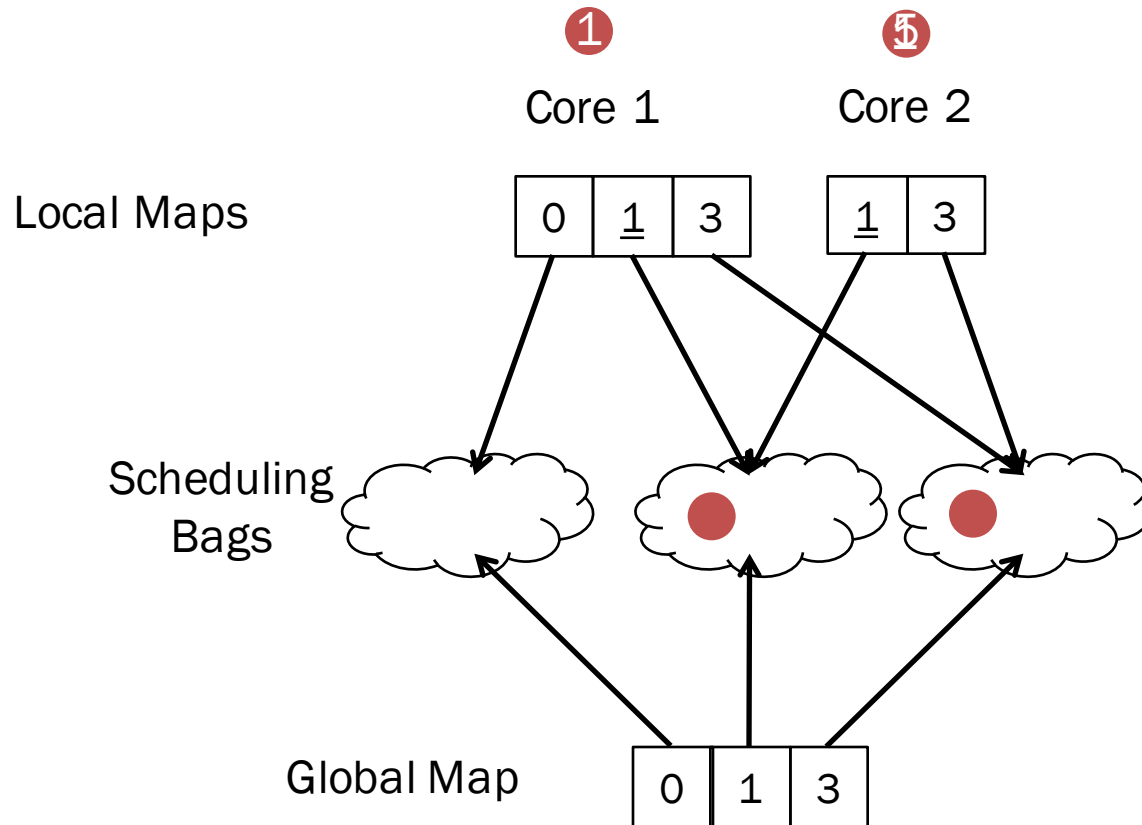
Priority Map



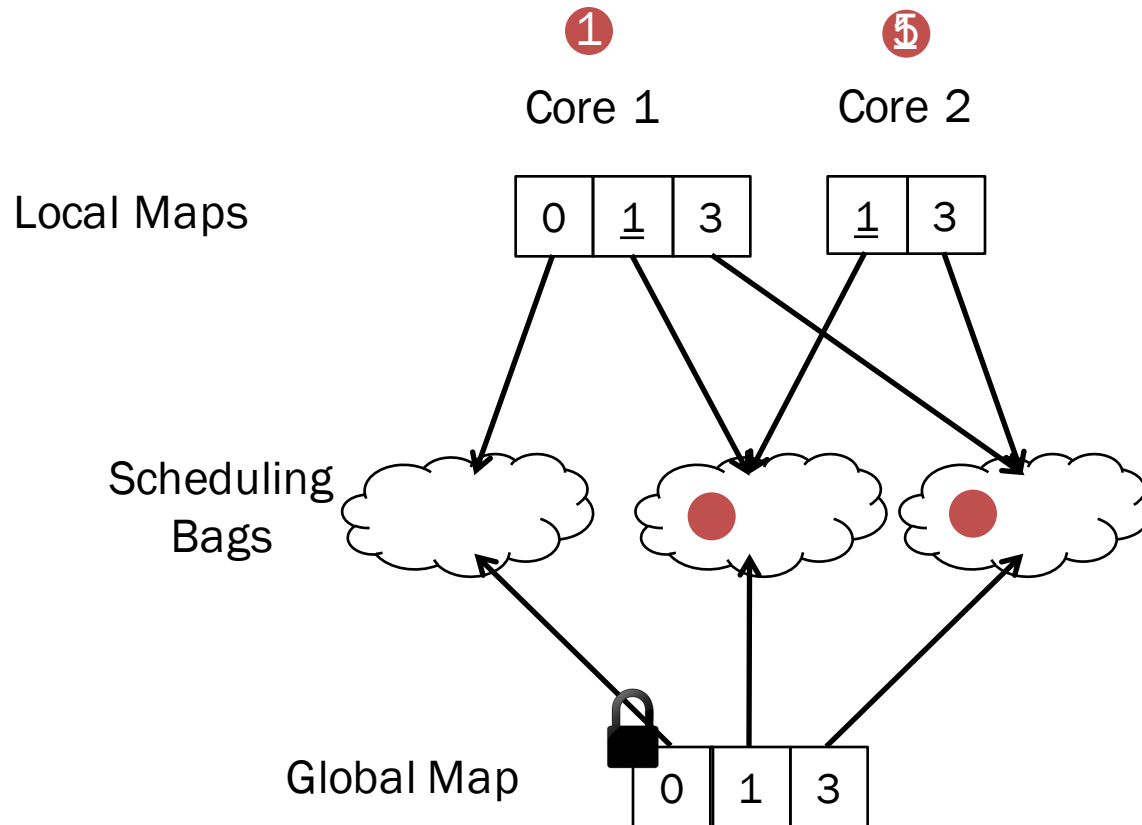
Priority Map



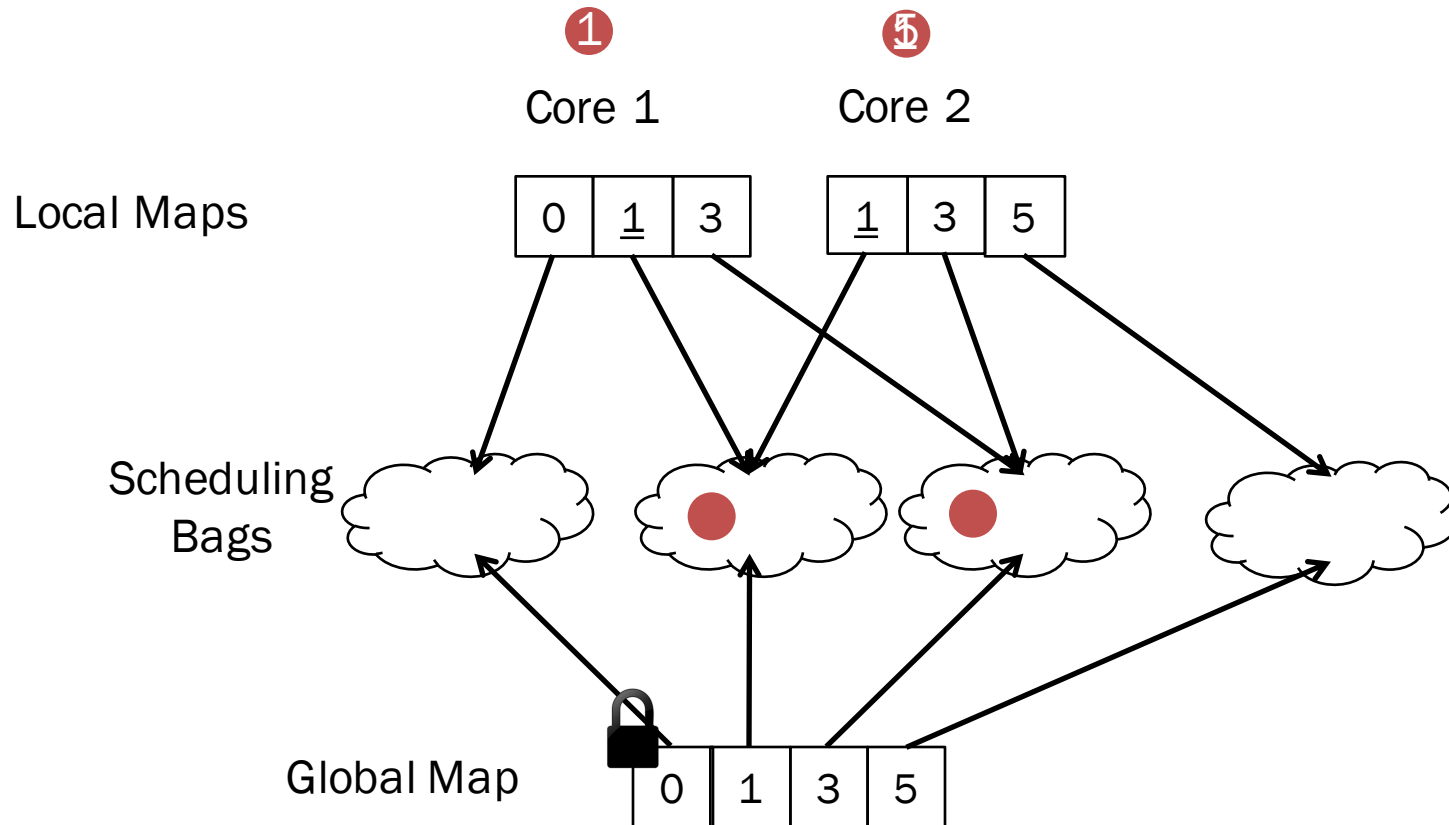
Priority Map



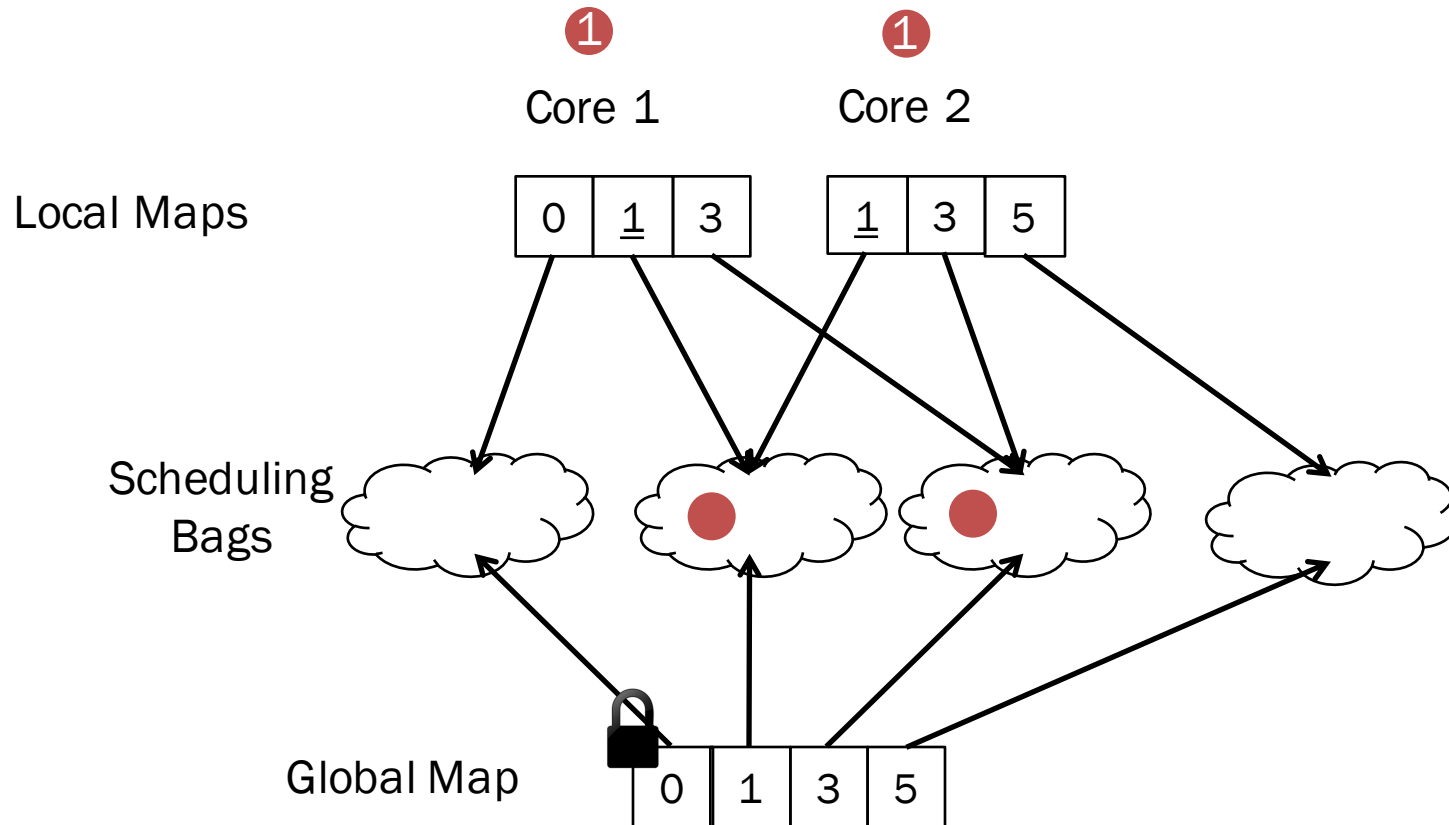
Priority Map



Priority Map



Priority Map



Outline

1. Abstraction for graph analytics applications and implementation in Galois system
2. One piece of infrastructure: priority scheduling
3. Evaluation of Galois system versus graph analytics DSLs

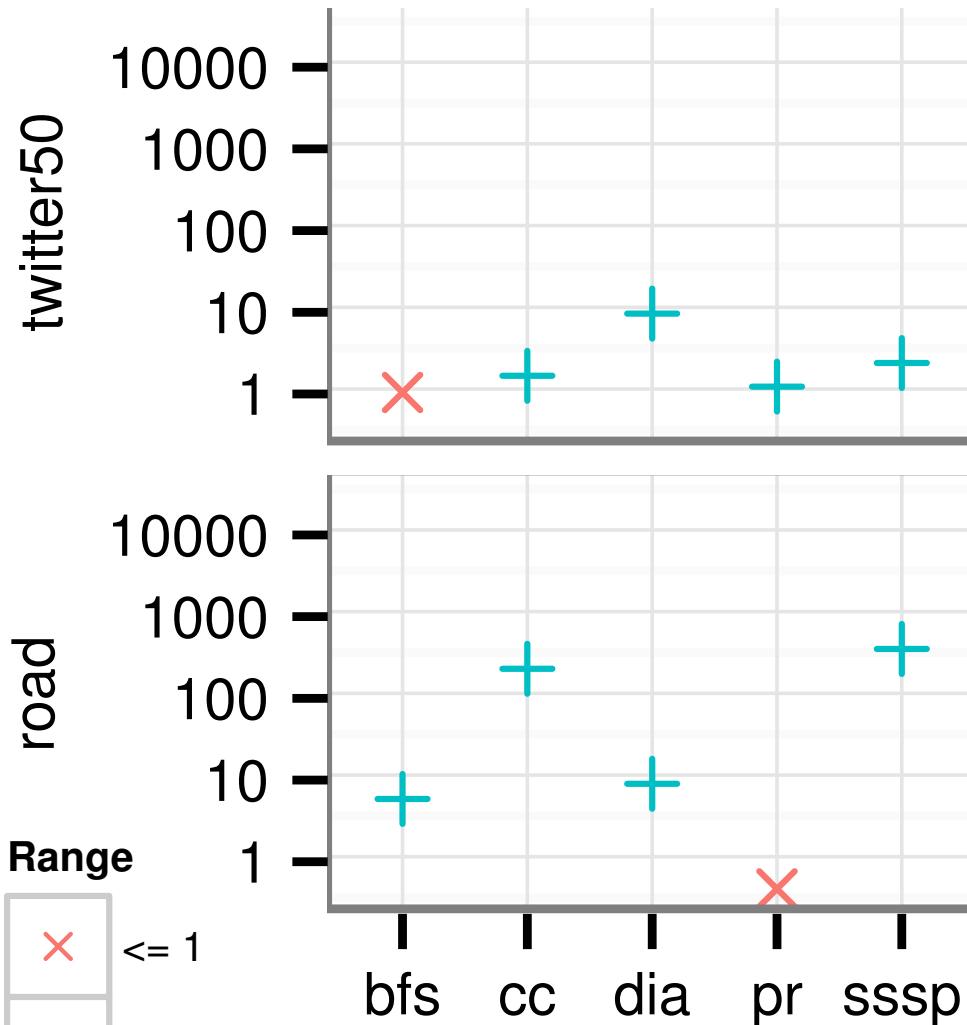
Graph Analytics DSLs

- GraphLab Low et al. (UAI '10)
- PowerGraph Gonzalez et al. (OSDI '12)
- GraphChi Kyrola et al. (OSDI '12)
- Ligra Shun and Blelloch (PPoPP '13)
- Pregel Malewicz et al. (SIGMOD '10)
- ...
- Easy to implement their APIs on top of Galois system
 - Galois implementations called PowerGraph-g, Ligra-g, etc.
 - About 200-300 lines of code each

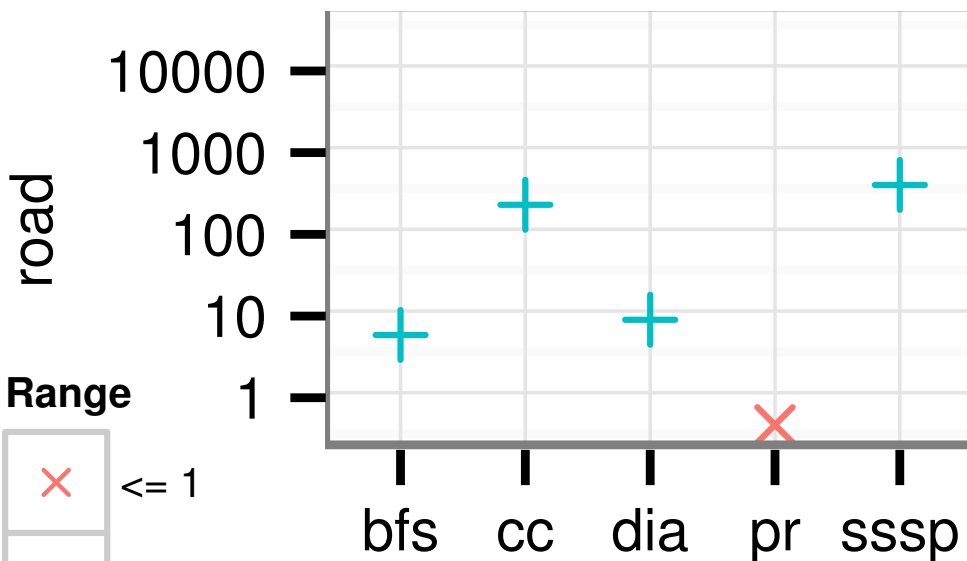
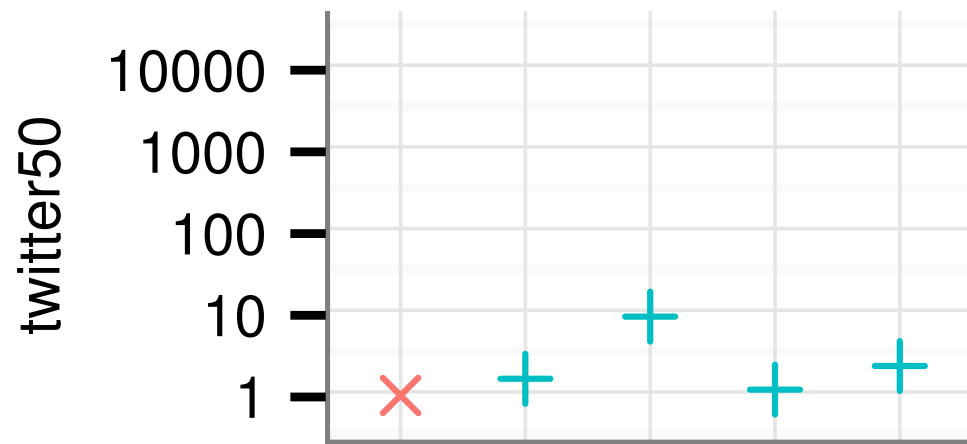
Evaluation

- Platform
 - 40-core system
 - 4 socket, Xeon E7-4860 (Westmere)
 - 128 GB RAM
- Applications
 - Breadth-first search (bfs)
 - Connected components (cc)
 - Approximate diameter (dia)
 - PageRank (pr)
 - Single-source shortest paths (sssp)
- Inputs
 - twitter50 (50 M nodes, 2 B edges, low-diameter)
 - road (20 M nodes, 60 M edges, high-diameter)
- Comparison with
 - Ligra (shared memory)
 - PowerGraph (distributed)
 - Runtimes with 64 16-core machines (1024 cores) does not beat one 40-core machine using Galois

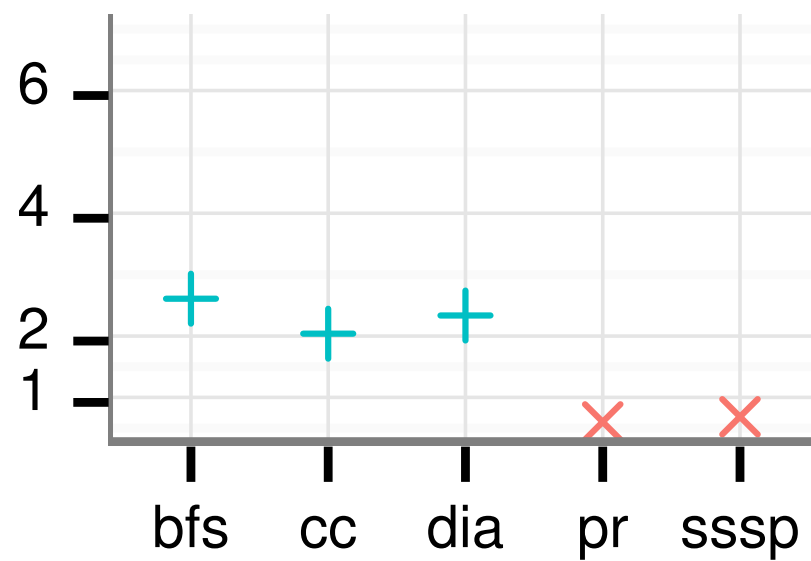
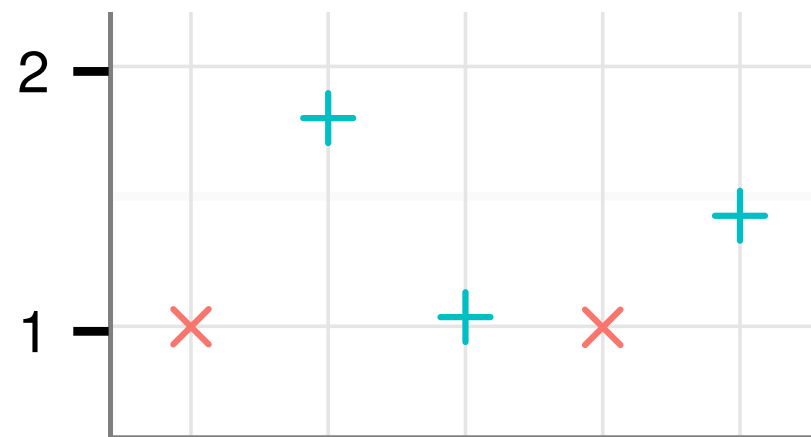
Ligra runtime
Galois runtime



Ligra runtime
Galois runtime

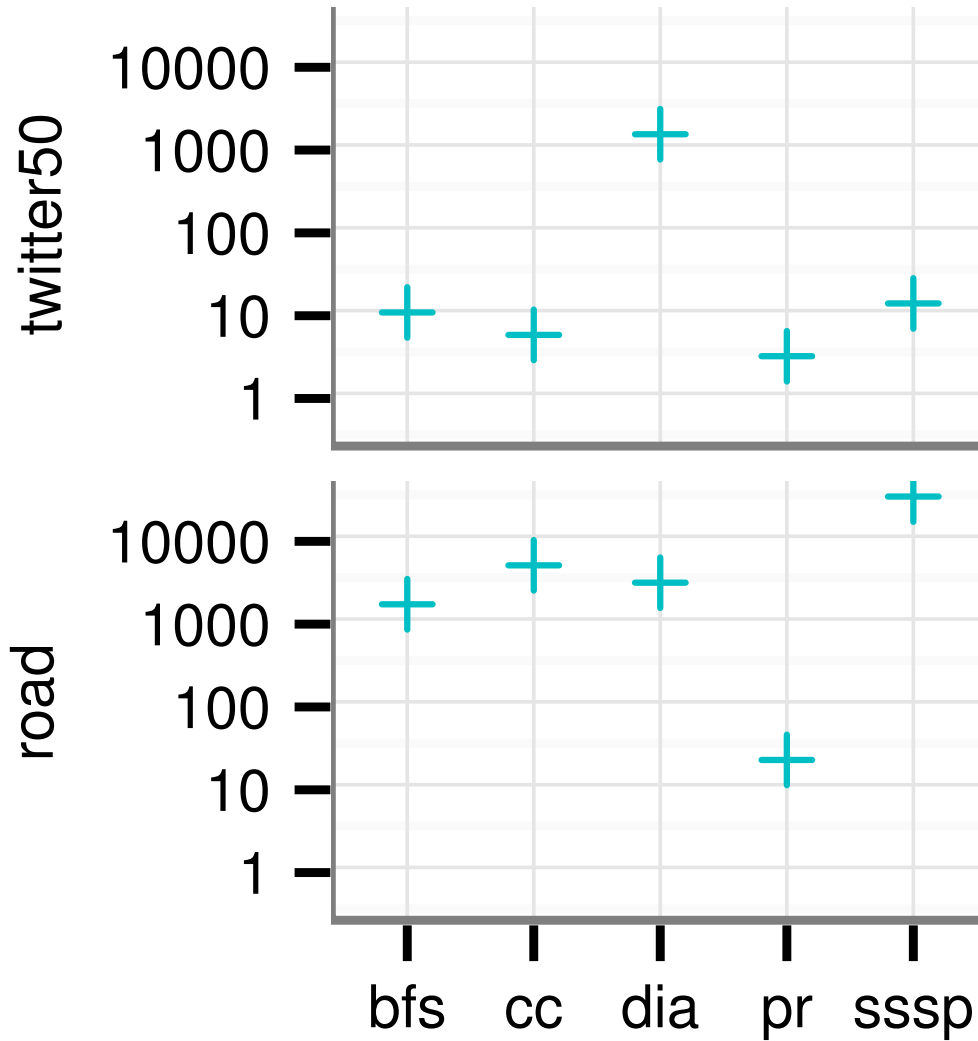


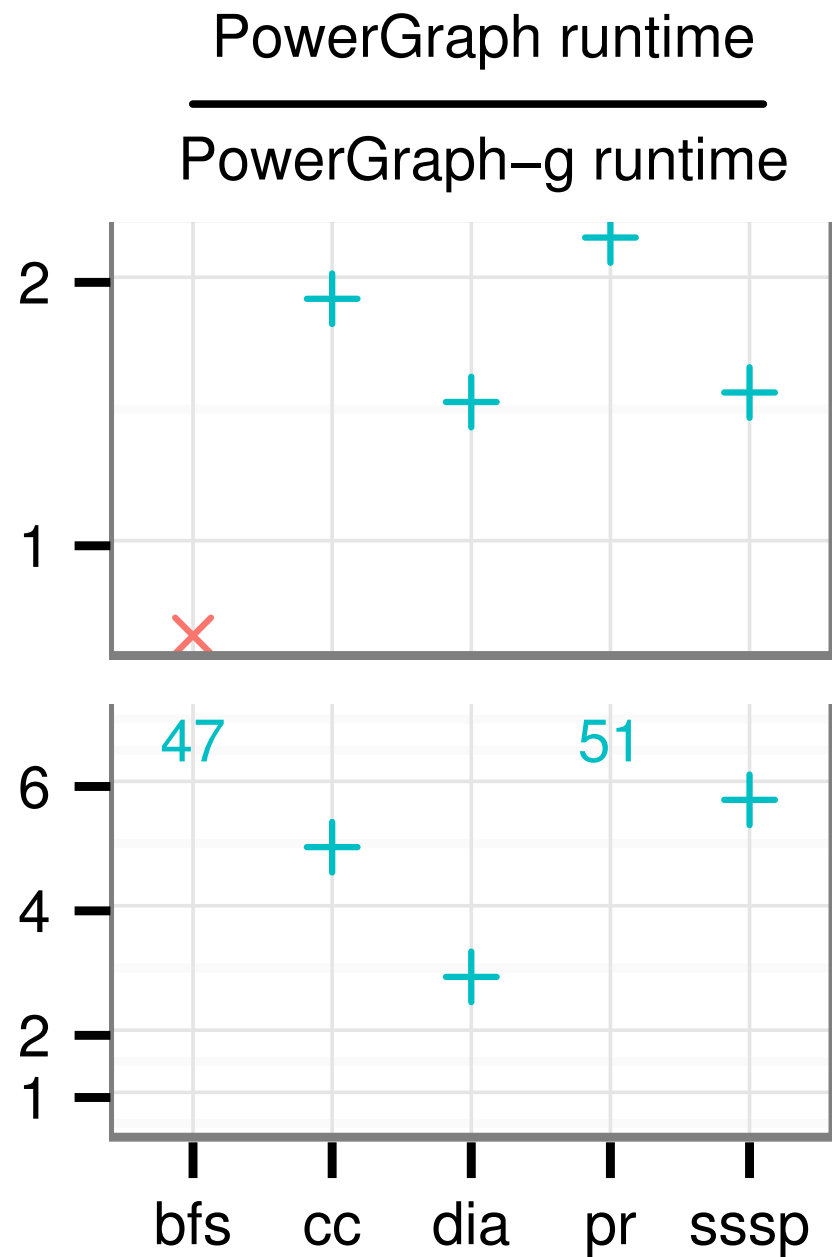
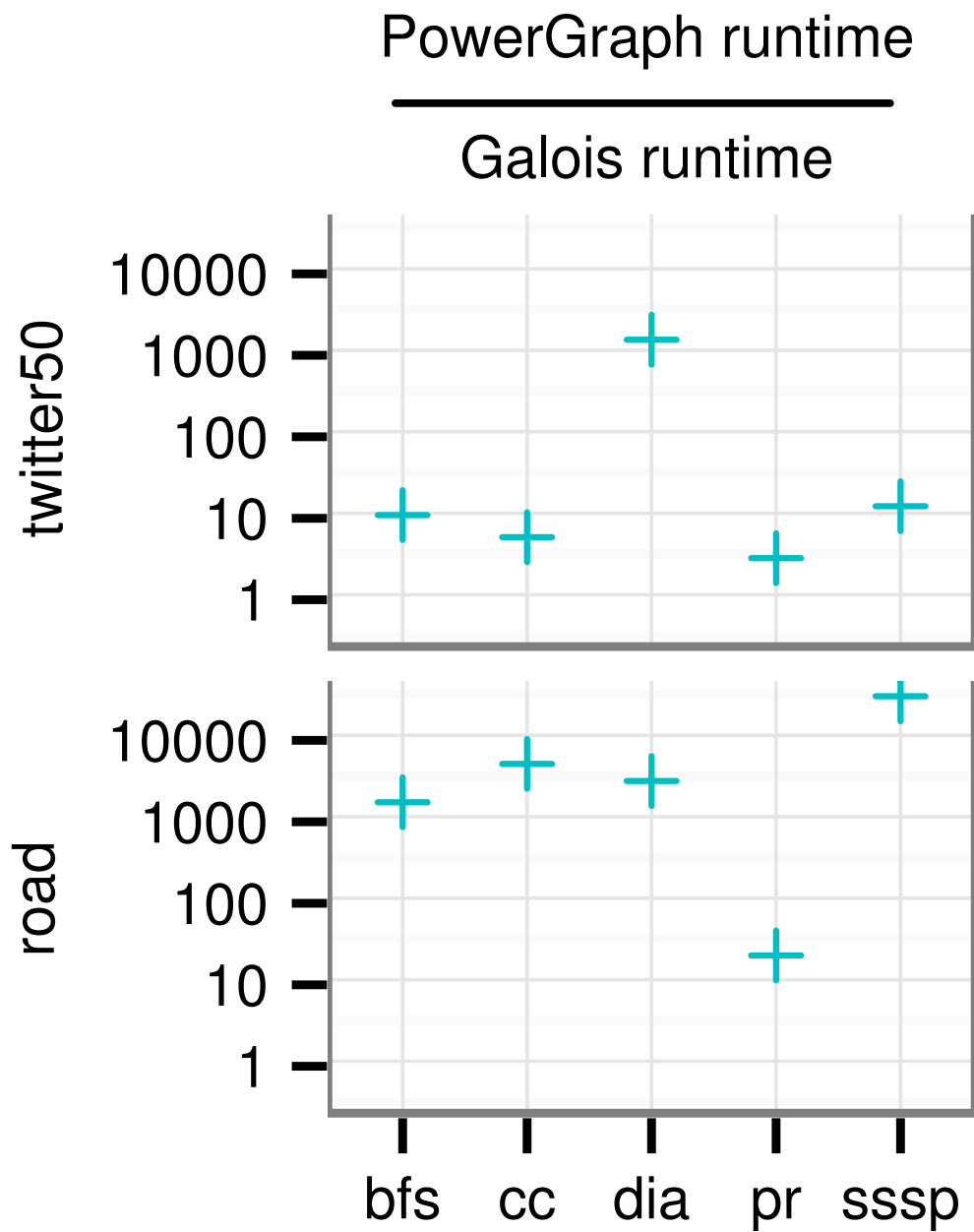
Ligra runtime
Ligra-g runtime

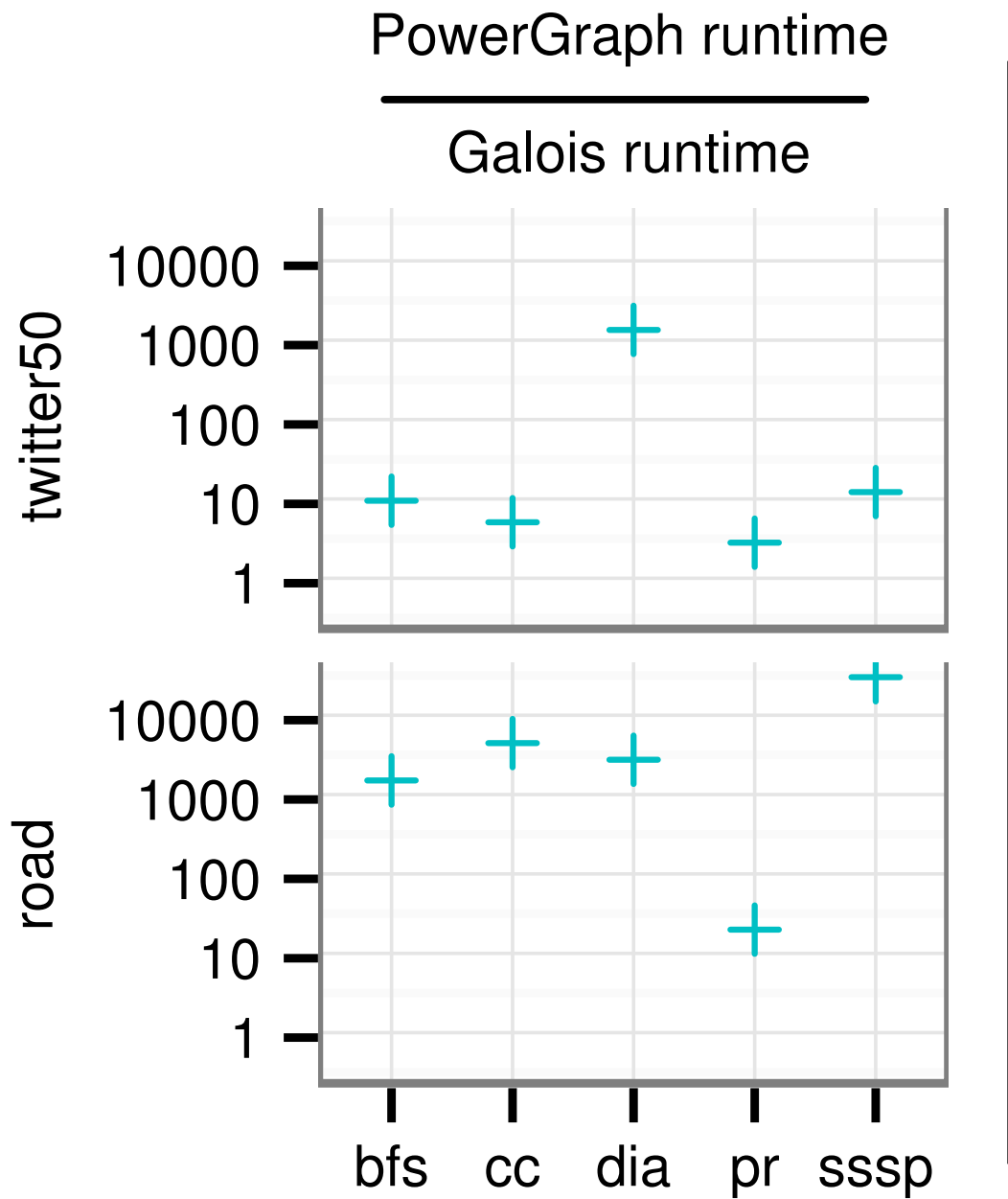


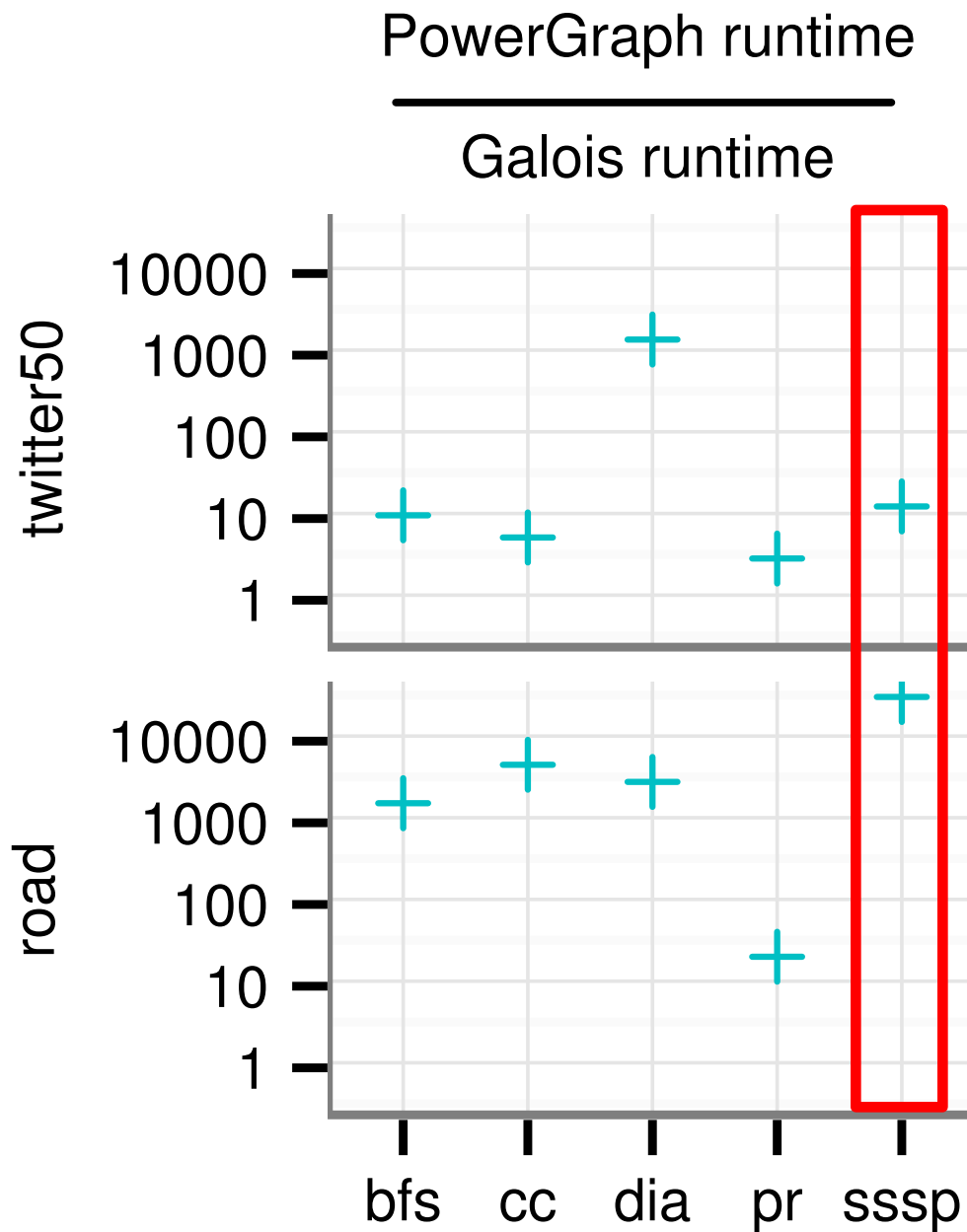
PowerGraph runtime

Galois runtime

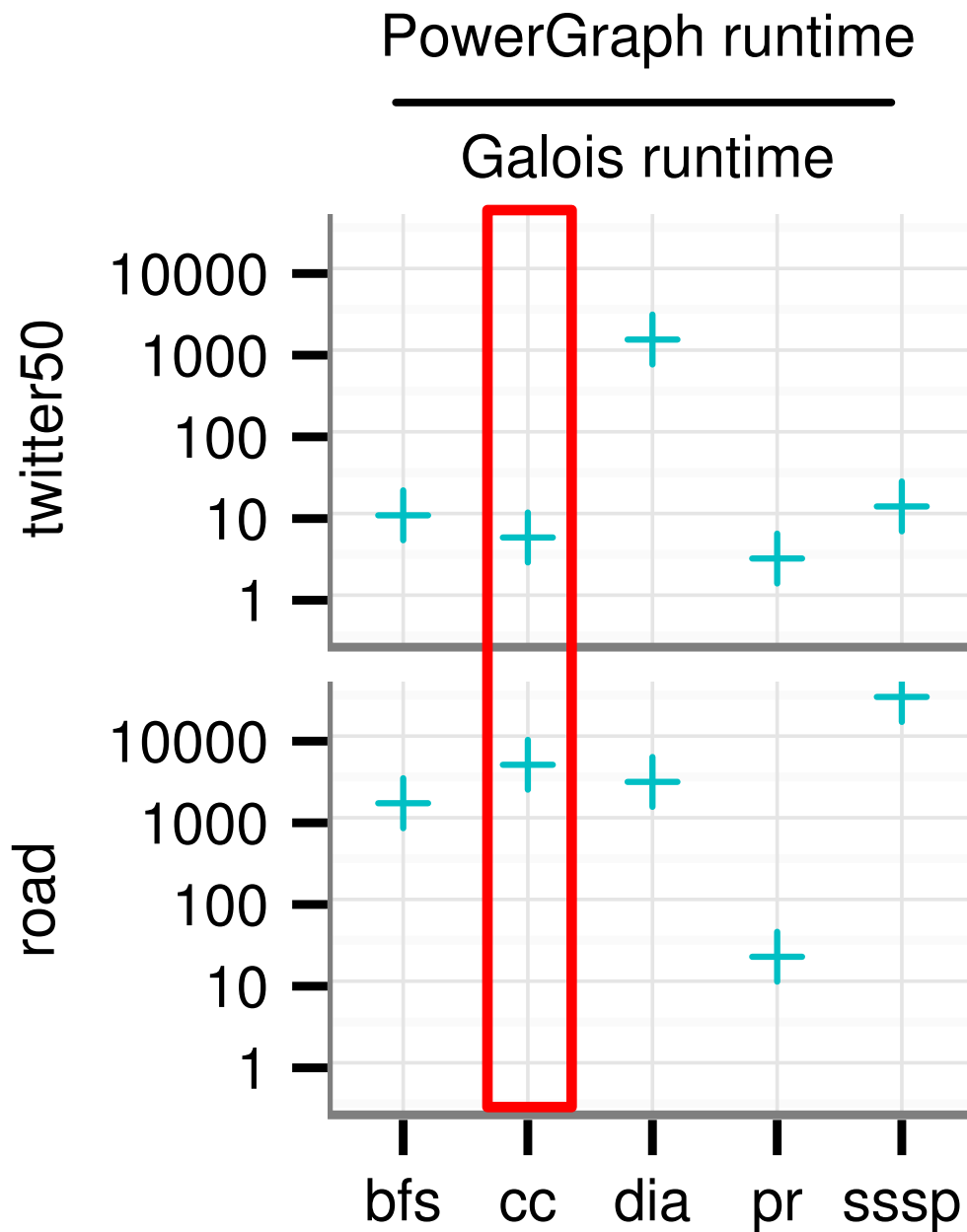




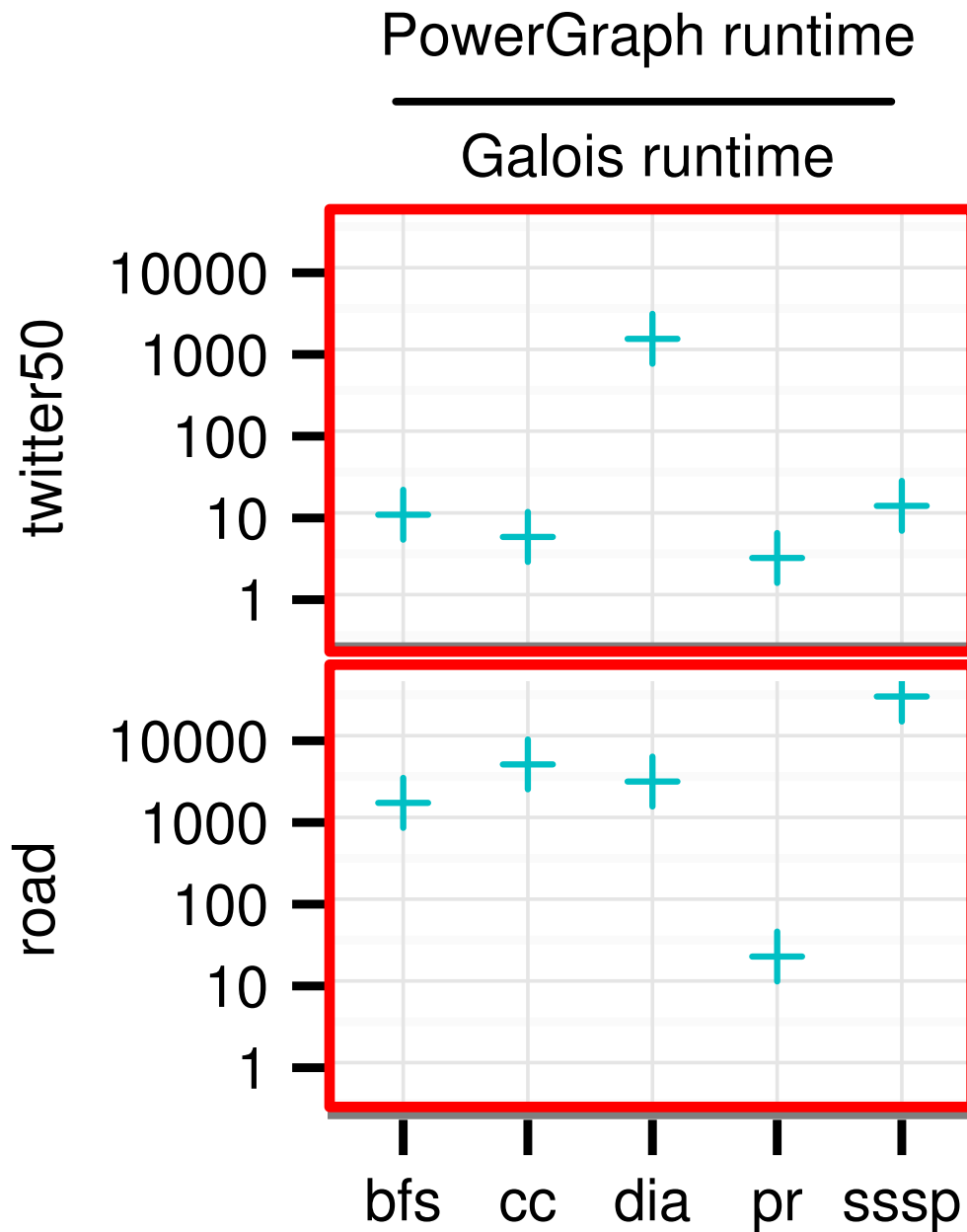




- The best algorithm may require application-specific scheduling
 - Priority scheduling for SSSP
- The best algorithm may not be expressible as a vertex program
 - Connected components with union-find
- Autonomous scheduling required for high-diameter graphs
 - Coordinated scheduling uses many rounds and has too much overhead



- The best algorithm may require application-specific scheduling
 - Priority scheduling for SSSP
- The best algorithm may not be expressible as a vertex program
 - Connected components with union-find
- Autonomous scheduling required for high-diameter graphs
 - Coordinated scheduling uses many rounds and has too much overhead



- The best algorithm may require application-specific scheduling
 - Priority scheduling for SSSP
- The best algorithm may not be expressible as a vertex program
 - Connected components with union-find
- Autonomous scheduling required for high-diameter graphs
 - Coordinated scheduling uses many rounds and has too much overhead

See Paper For

- More inputs, applications
- Detailed discussion of performance results
- More infrastructure: non-priority scheduling, memory management
- Results for out-of-core DSLs

Summary

- Galois programming model is general
 - Permits efficient algorithms to be written
- Galois infrastructure is lightweight
- Simpler graph DSLs can be layered on top
 - Can perform better than existing systems
- Download at <http://iss.ices.utexas.edu/galois>