Quiltdb:
Exploring Network Topology

15-799 course project

Jinliang & Mu

Taken From Seattle Glass Museum

Motivation

- Hierarchical Bandwidth
  - Socket, Memory, Rack, Datacenter, Global
  - Too expensive to improve all-to-all bandwidth
  - Possible to enhance certain connections
    - Rack ring
- Communication Intensive Machine Learning Algos
  - Huge parameter space: 10^8 — 10^10
  - Iterative

Secret Log From Secret Company

- [fig removed]

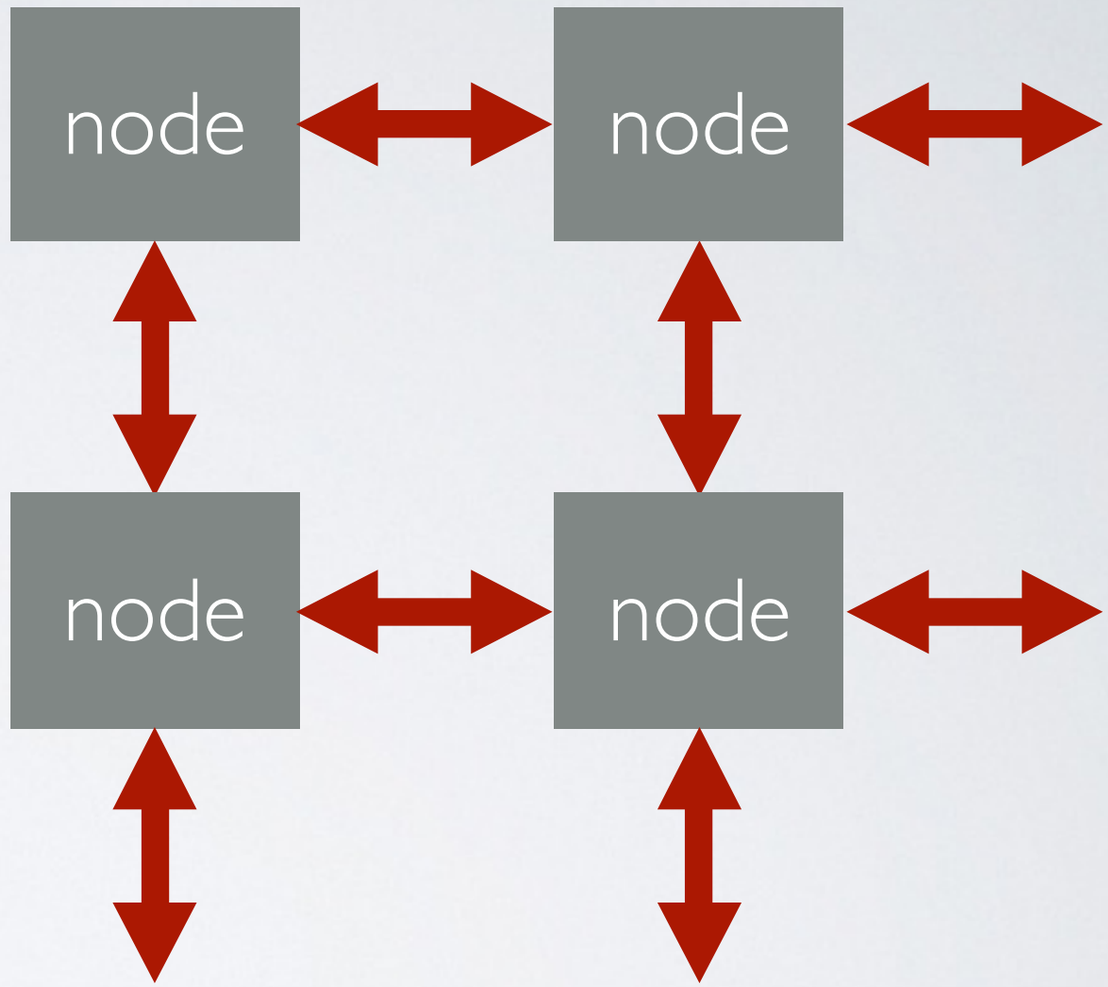
- Async parameter server, 500 machines, 10G network, a bunch of optimizations
- Better hardware or system?

A Bit Of History

- Years ago: mapped algorithms to topologies
  - Grid, Hyper-cube, Butterfly
- Now: avoid topology-specific optimization
  - World is flat
  - MPI, Graphlab, Parameter Server
- Future? History is Tic-Toc?
  - Both data and workloads are beyond hardware
  - People care cost on the big-data age

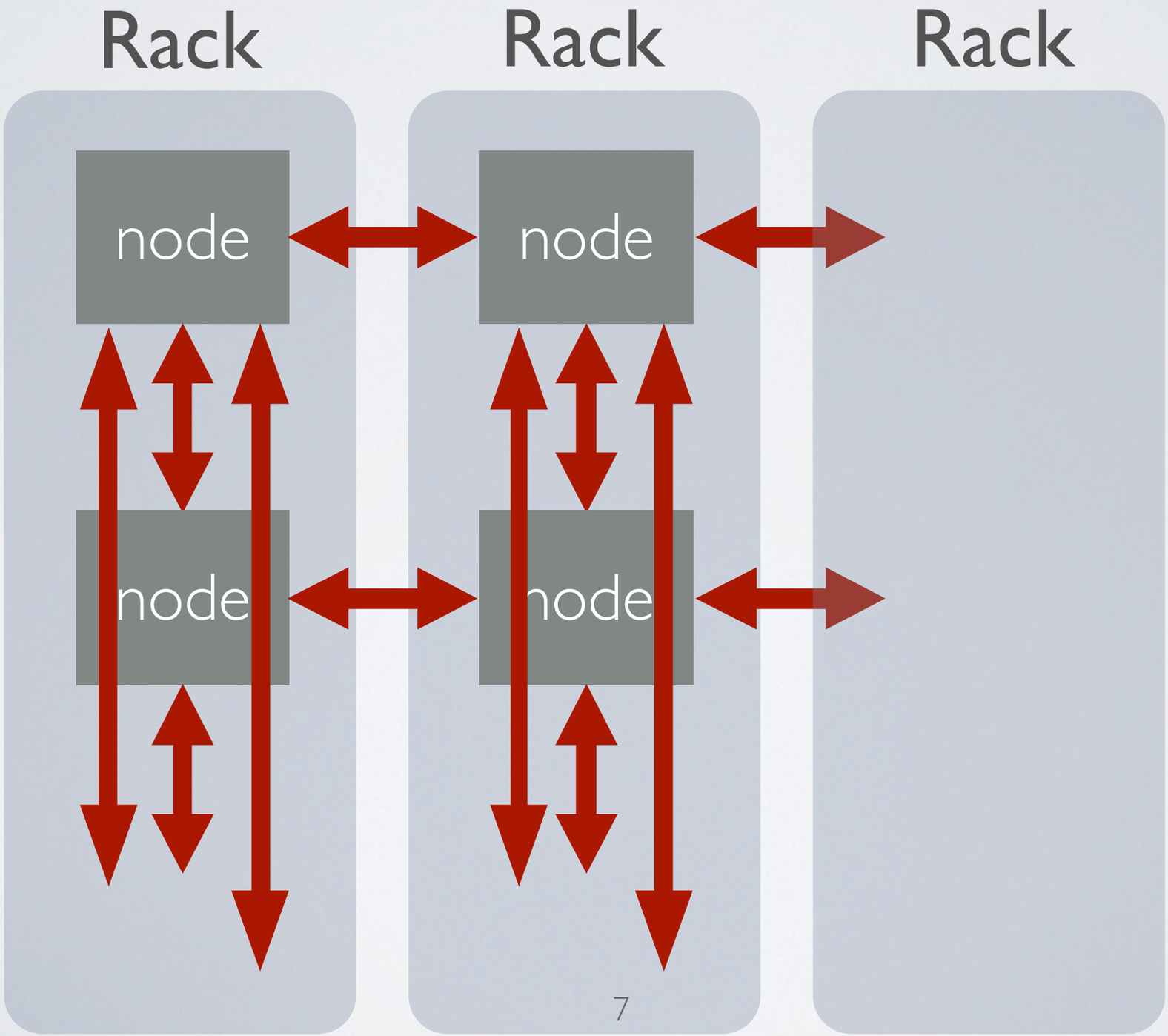
Grid Communication

- Pros:
  - Larger bandwidth
  - Cheaper network hardware
  - Less network congestion
- Cons:
  - Larger delay for far away nodes



Grouping Nodes Into Racks

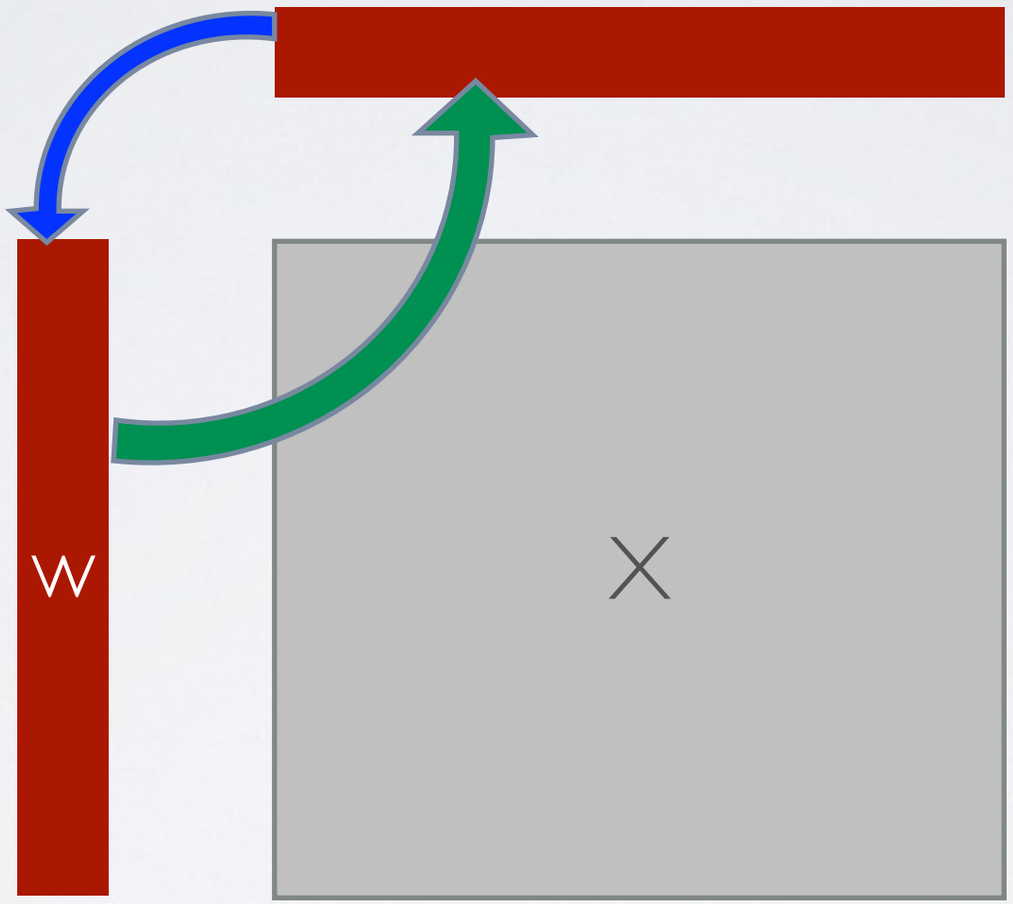
- High bandwidth within a rack



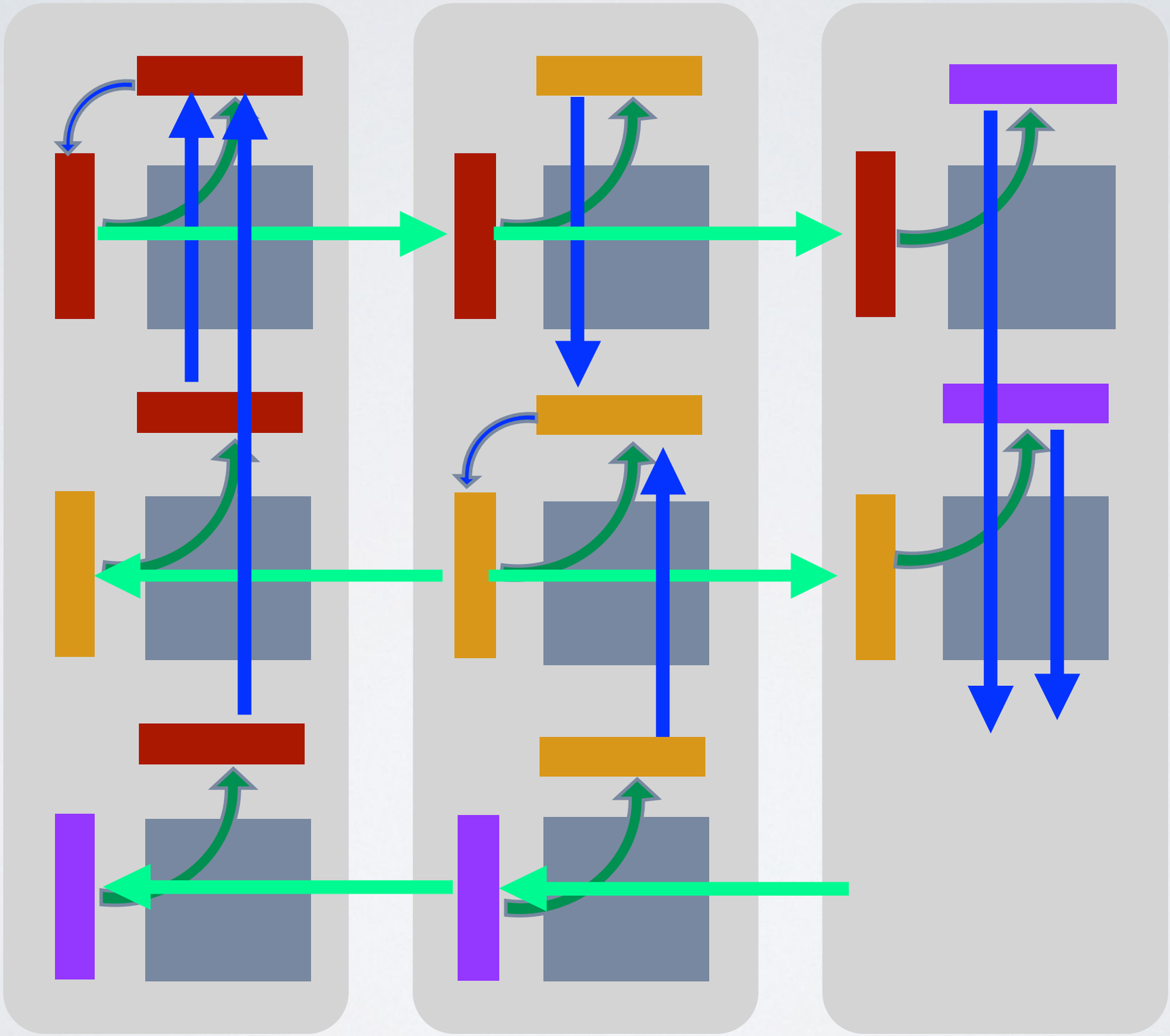
Machine Learning Applications

Pagerank

$$w \leftarrow \alpha X^T D^{-1} w + (1-\alpha)\frac{1}{n}\mathbf{1}$$

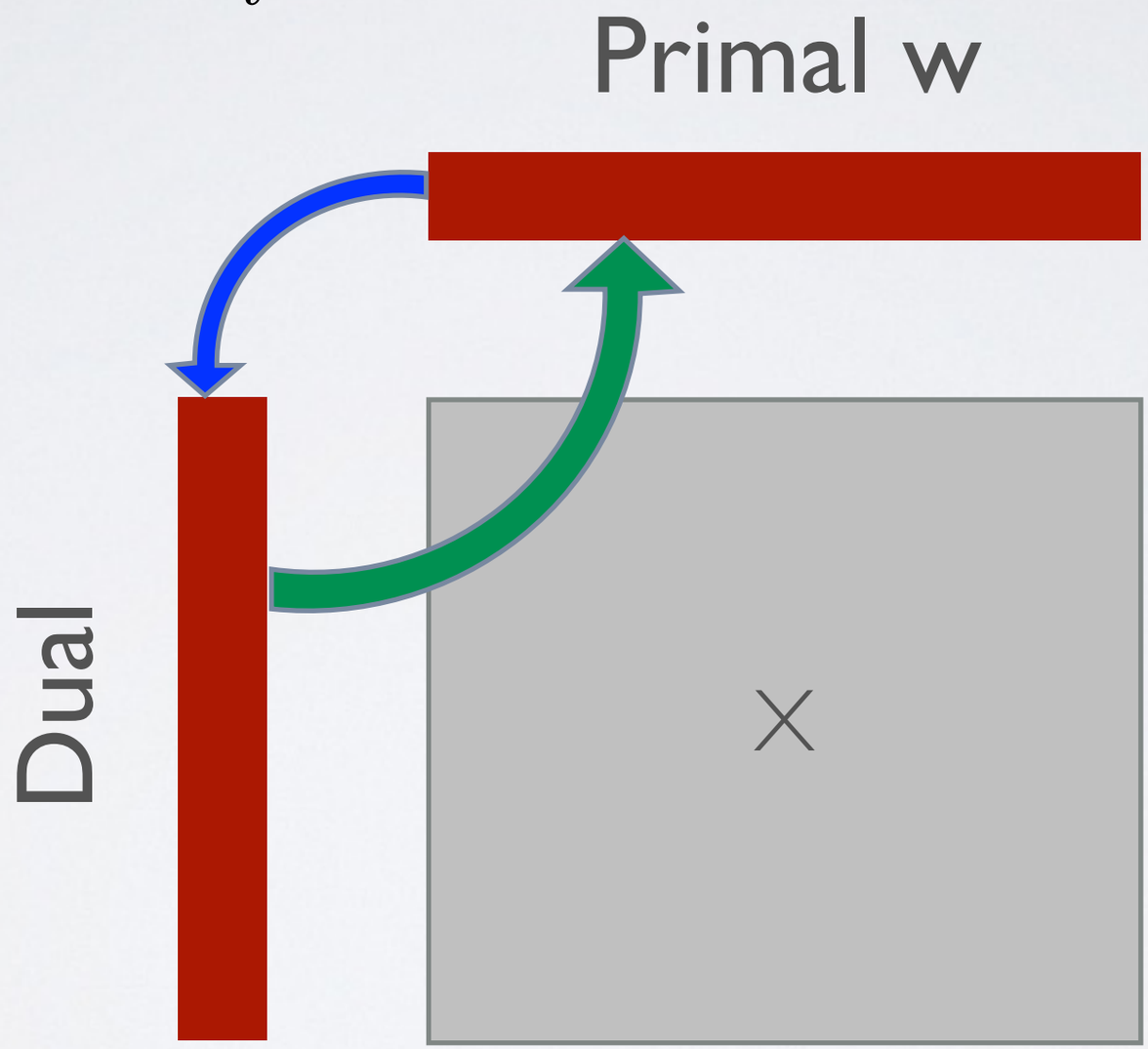


Quilting



Extend To Loss Minimization

$$\min_w \sum_i f(\langle x_i, w \rangle, y_i) + \lambda r(w)$$

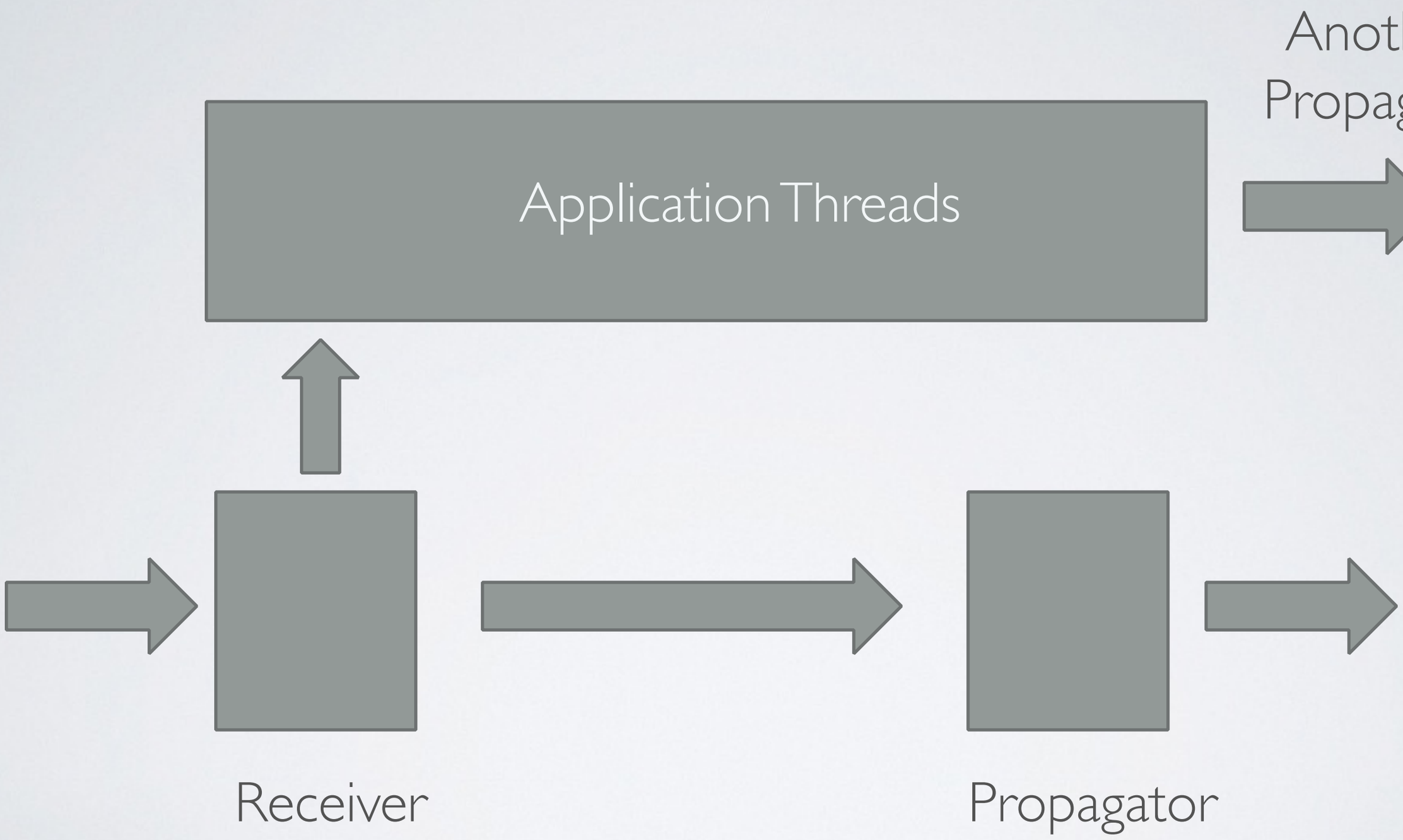


Implementation

What Is Quiltdb?

- Distributed key-value store
- Restricted write: must be commutative and associative - Only inc
- Eventual consistency
- User-defined topology for writes to flow
  - application developers may apply knowledge about the physical network
  - allows cyclic flow

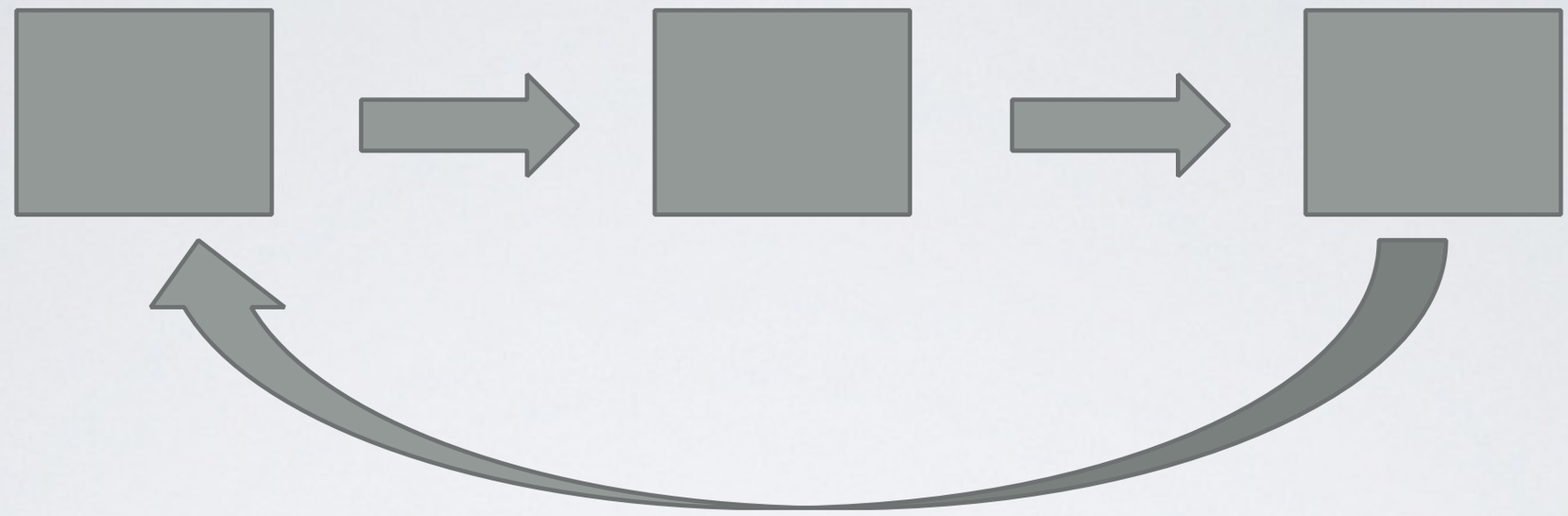
Receiver & Propagator



More Details

- Multiple receiver-propagator pairs (currently 2) per node
- 1 or 0 downstream receiver per propagator
  - At most 2 flows per node.
  - Exactly 1 flow per table.
- Tables are shared within flow path.
- Upon receiving a set of OpLogs (all by receiver):
  - Apply them to local copy
  - Forward them to propagator
  - Execute a user-defined callback

Cycle Flow Of Oplogs

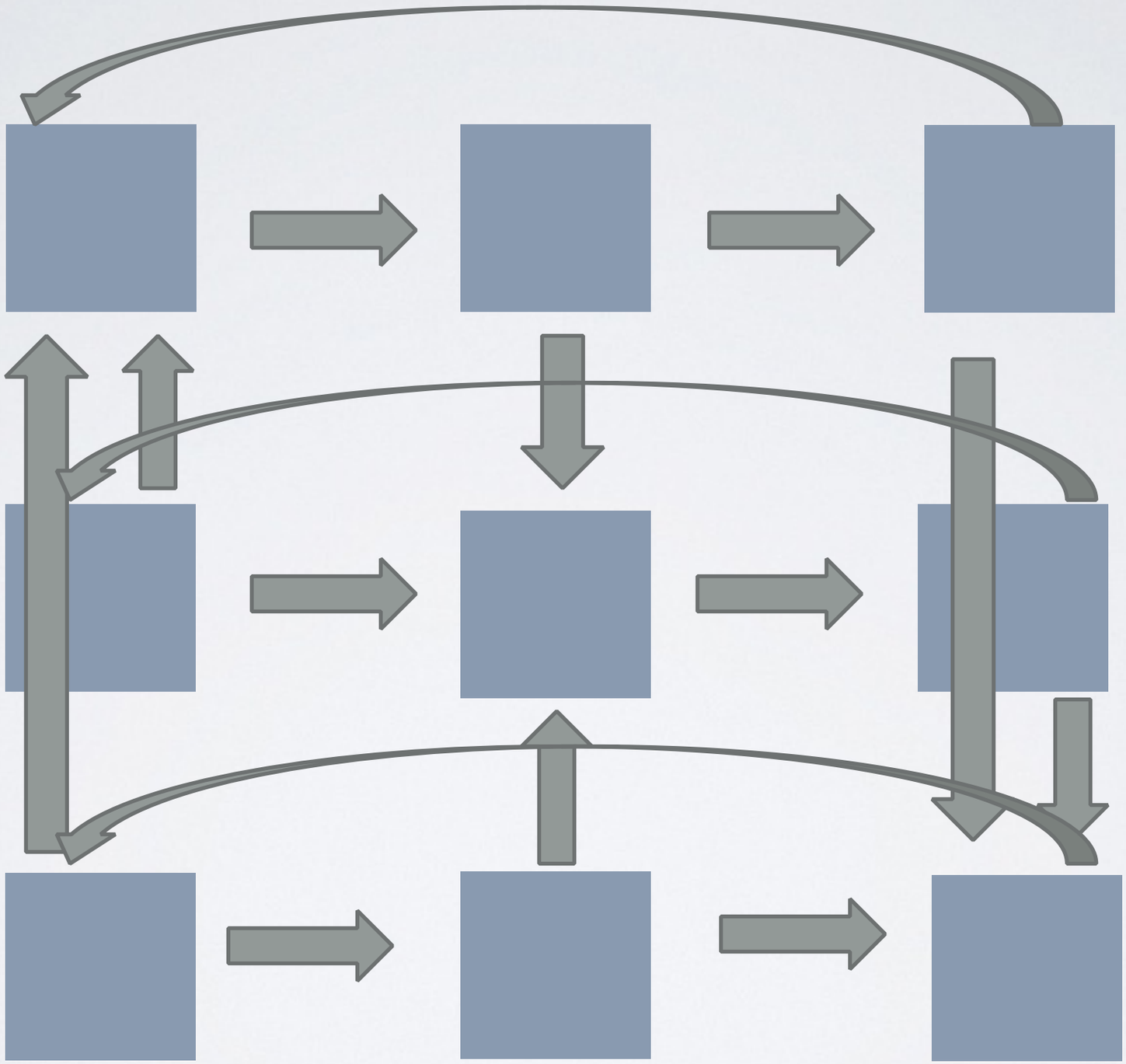


- Why?
  - Reduced bandwidth usage.
- Problem?
  - A node may see an OpLog many times, but it only wants to apply it once.
- Solution?
  - OpLogs subtraction.

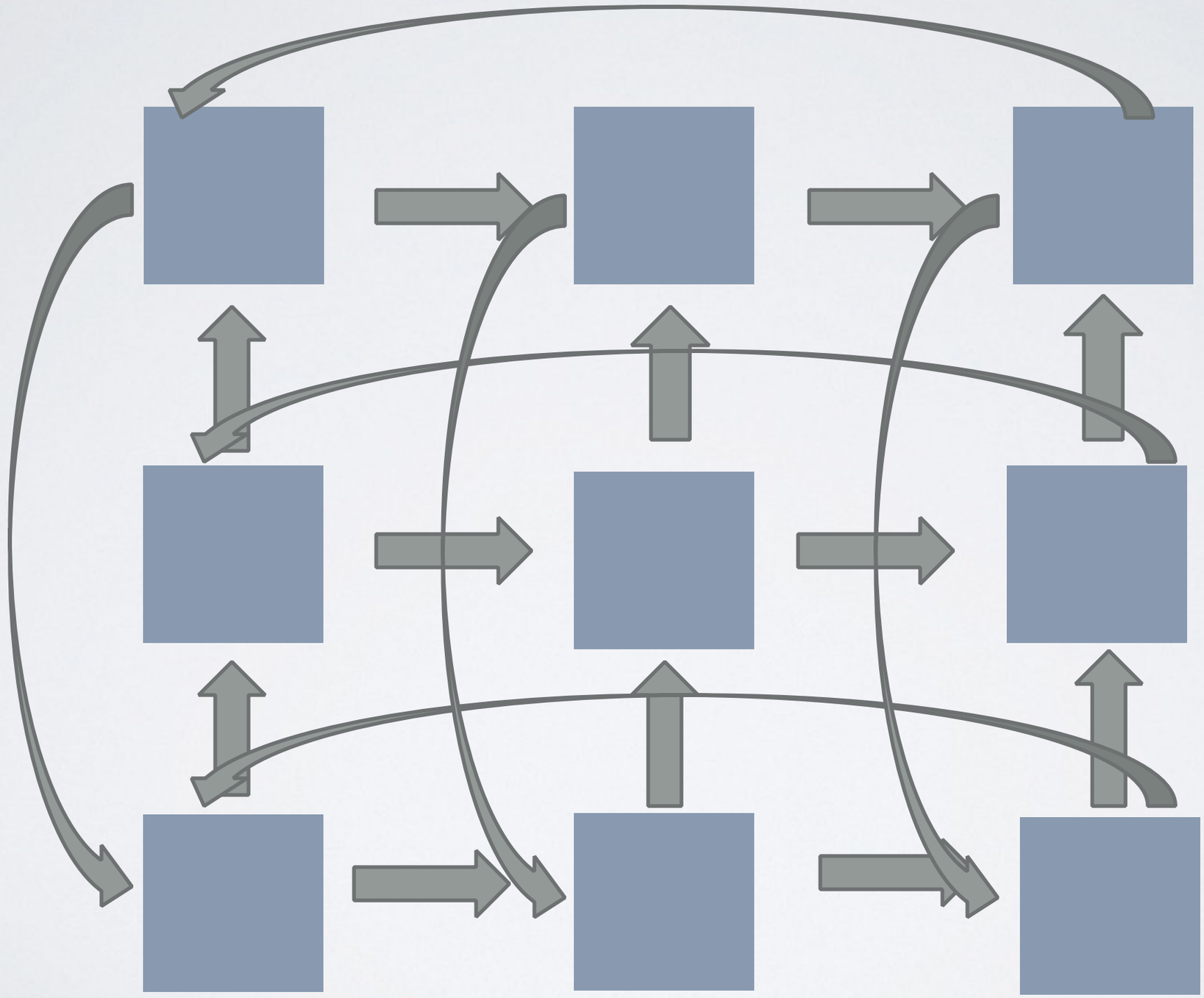
Aggregated Oplogs

- Inc(x, 5) + Inc(x, 2) => Inc(x, 7)
- Pros: reduced OpLog size.
- How?
  - Local OpLogs are batched.
  - Aggregated with received OpLogs.

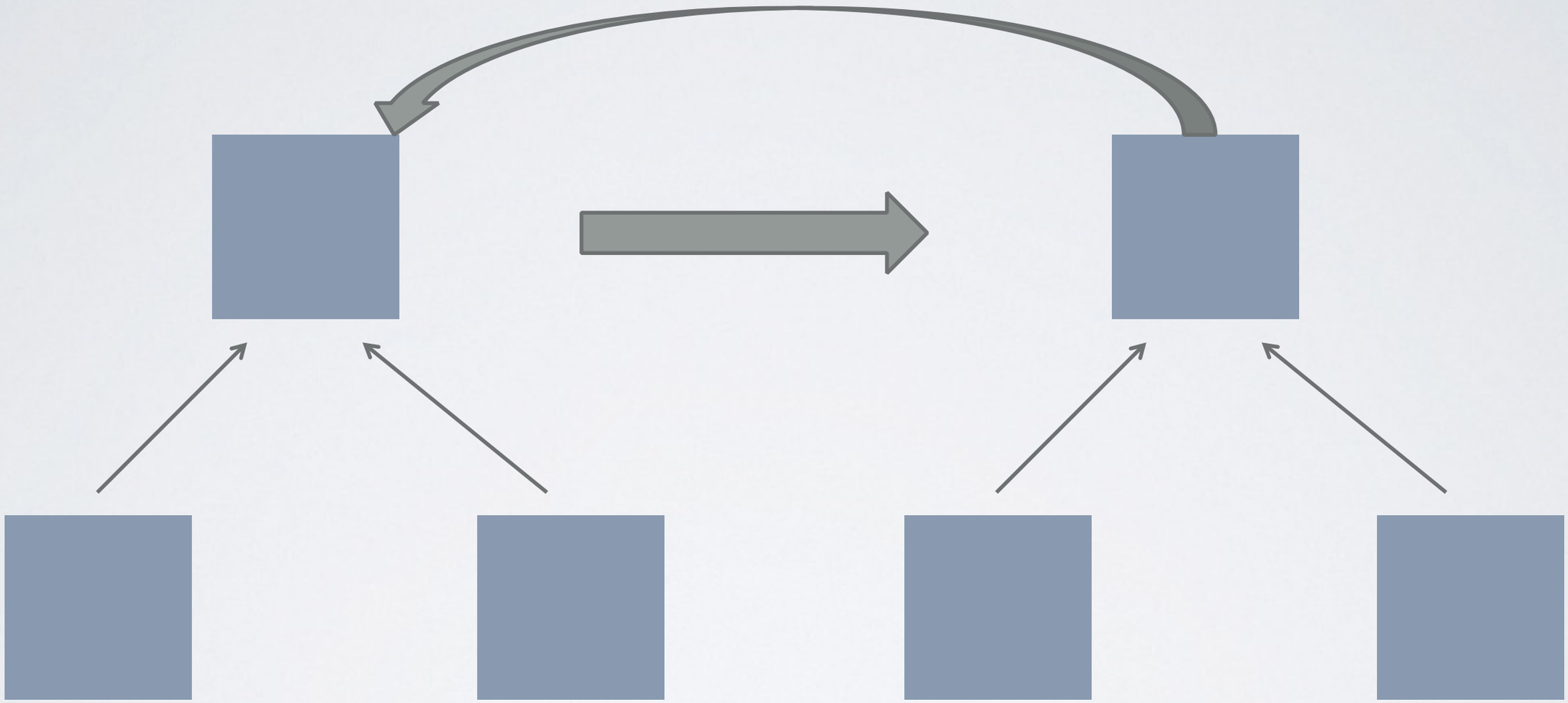
Topology Example: Star+Ring



Topology Example: Ring+Ring



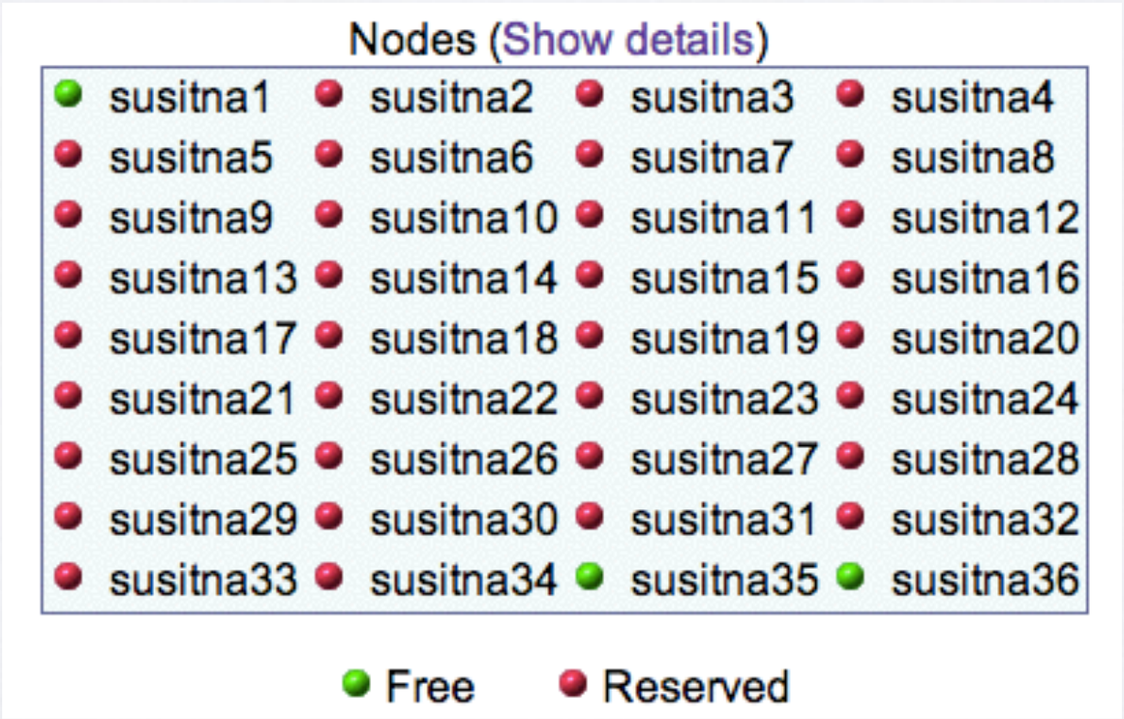
Topology Example: Tree+Ring



Evaluation

Experimental Setup

- Web data
  - 3.6B nodes, 128B edges, 1.3T binary data
- Even Large Labeled Real Data: >10T
- Machine
  - PDL susitna: 64 core, 128G mem, 40G network



Future Work

- Experimental Results
- A VLDB submission