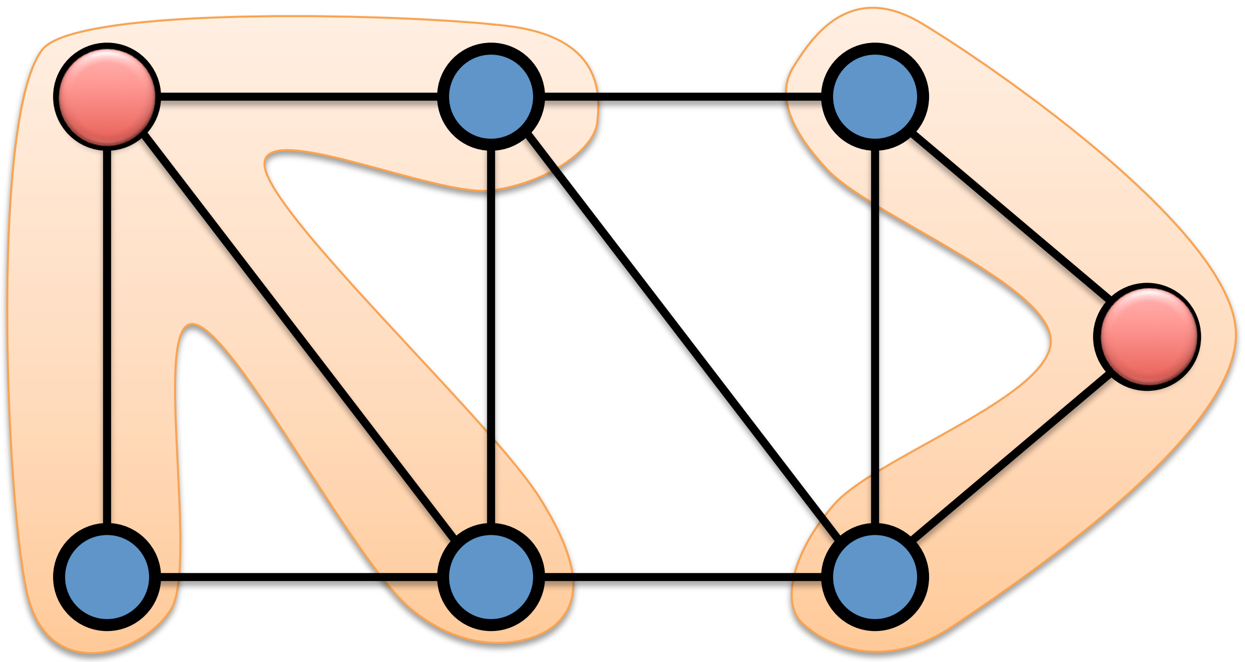
GraphLab and its distributed versions

Mu Li

Adopted slides from Joseph and Yuchen

The **GraphLab** Abstraction

- A user-defined **Vertex Program** runs on each vertex
- **Graph** constrains **interaction** along edges
  - Directly **read** and **modify** the state of adjacent vertices and edges
- **Parallelism**: run multiple vertex programs simultaneously

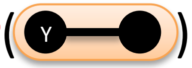


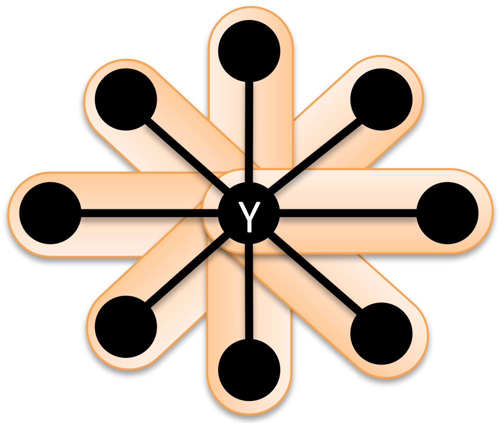
GAS Decomposition

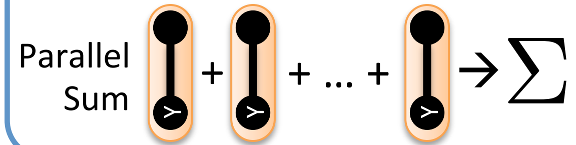
Gather (Reduce)

Accumulate information about neighborhood

User Defined:

- **Gather**() → Σ
- $\Sigma_1 \oplus \Sigma_2 \rightarrow \Sigma_3$



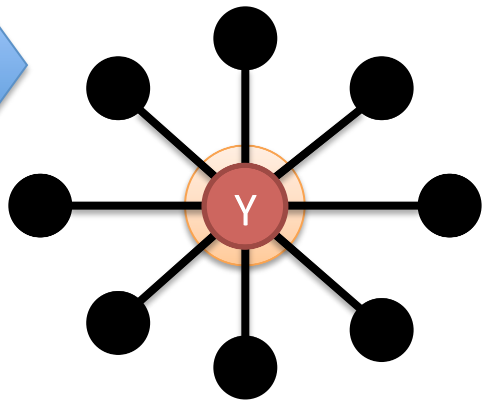


Apply

Apply the accumulated value to center vertex

User Defined:

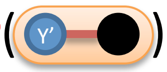
- **Apply**(, Σ) →

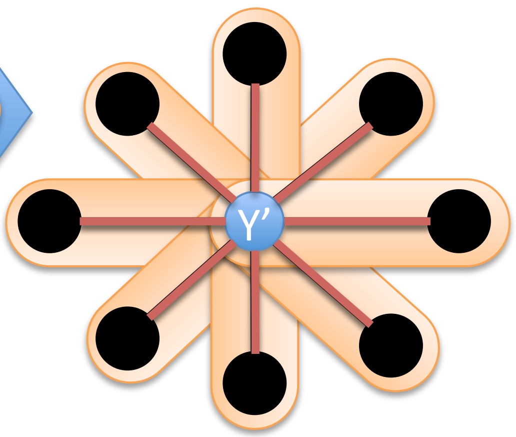


Scatter

Update adjacent edges and vertices.

User Defined:

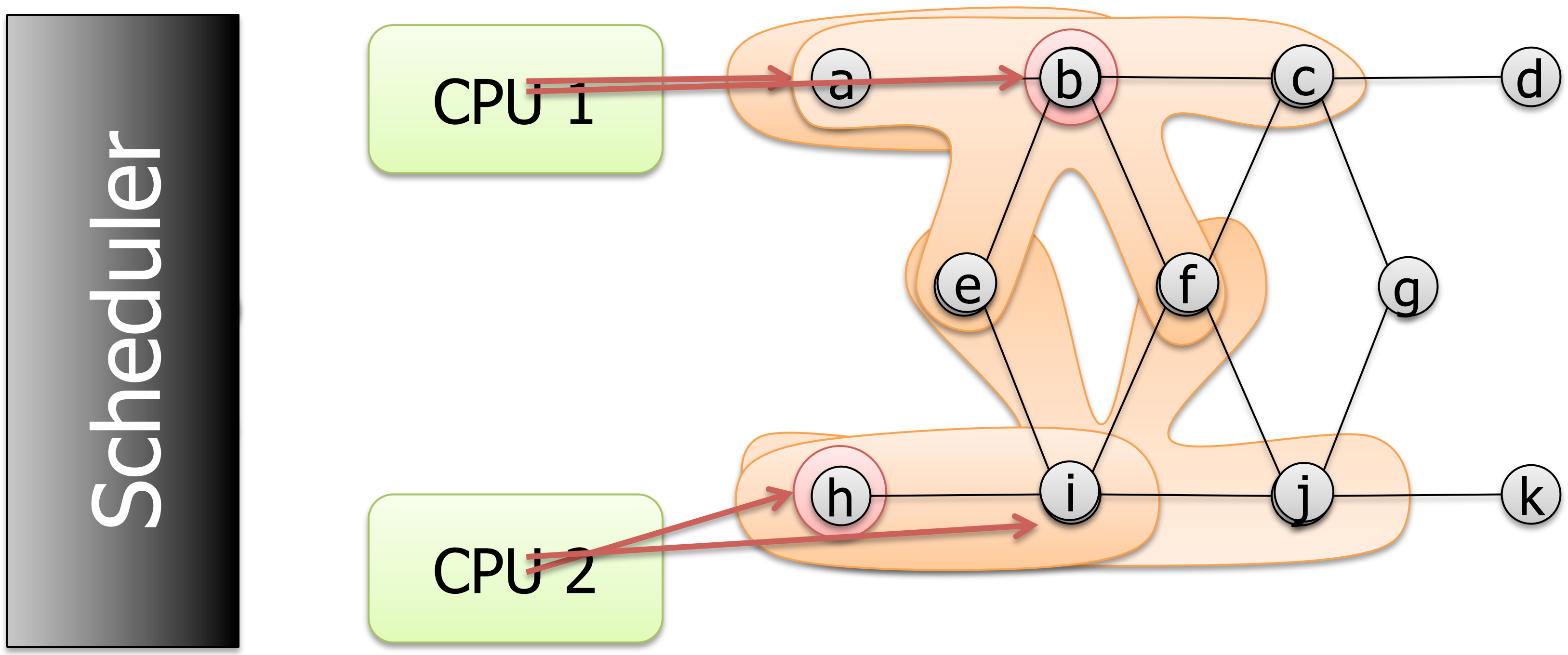
- **Scatter**() →



Update Edge Data & Activate Neighbors

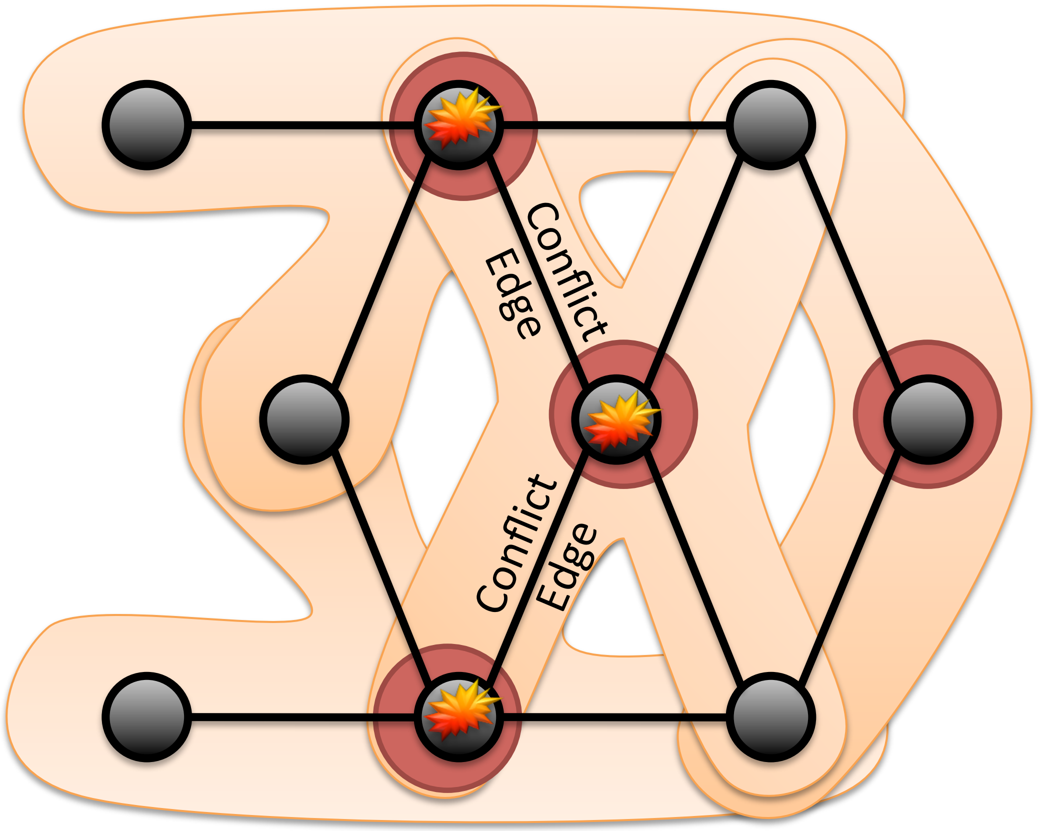
GraphLab is **Asynchronous**

The **scheduler** determines the order that vertices are executed



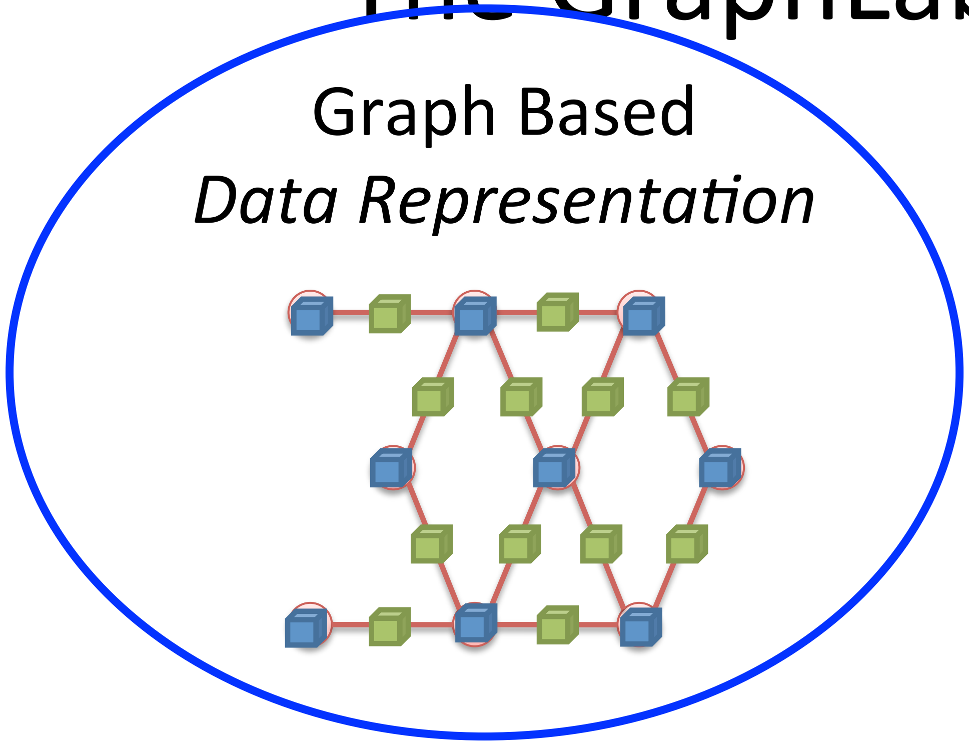
Scheduler can **prioritize** vertices.

GraphLab is **Serializable**



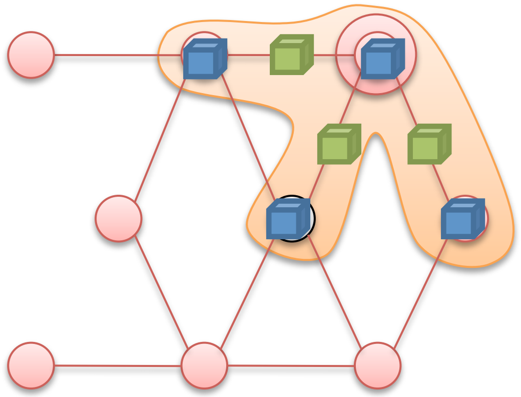
- Automatically ensures **serializable** executions

The GraphLab Framework

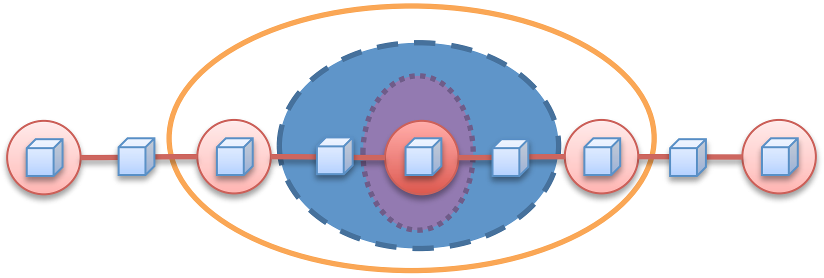


Graph Based
Data Representation

Update Functions
User Computation

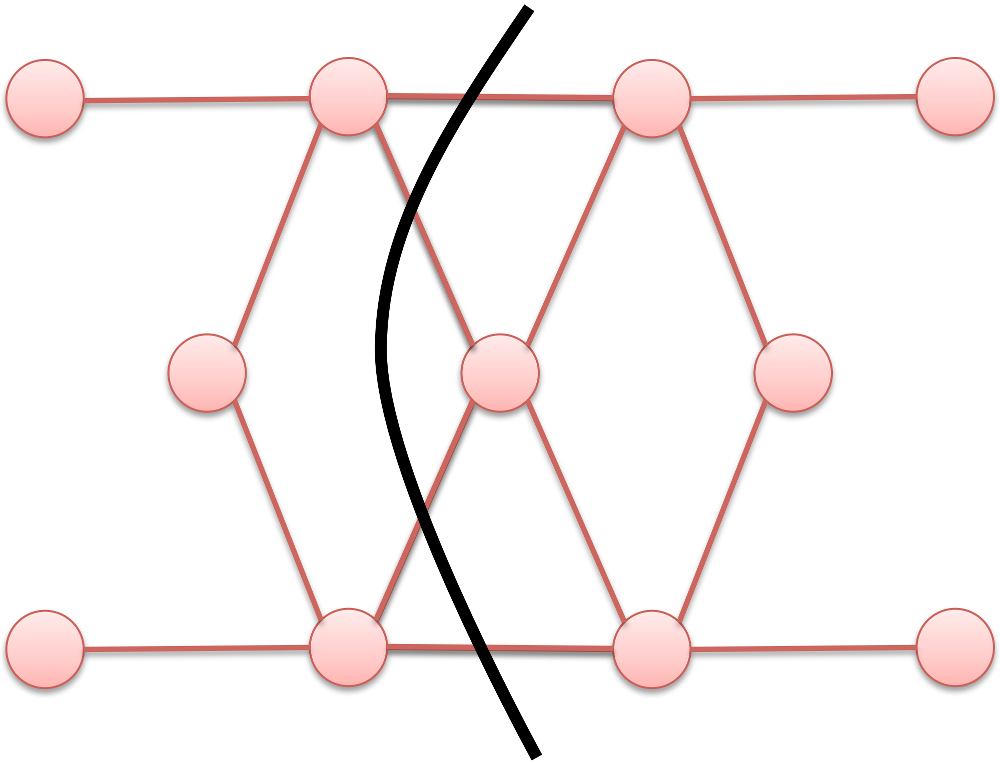


Consistency Model



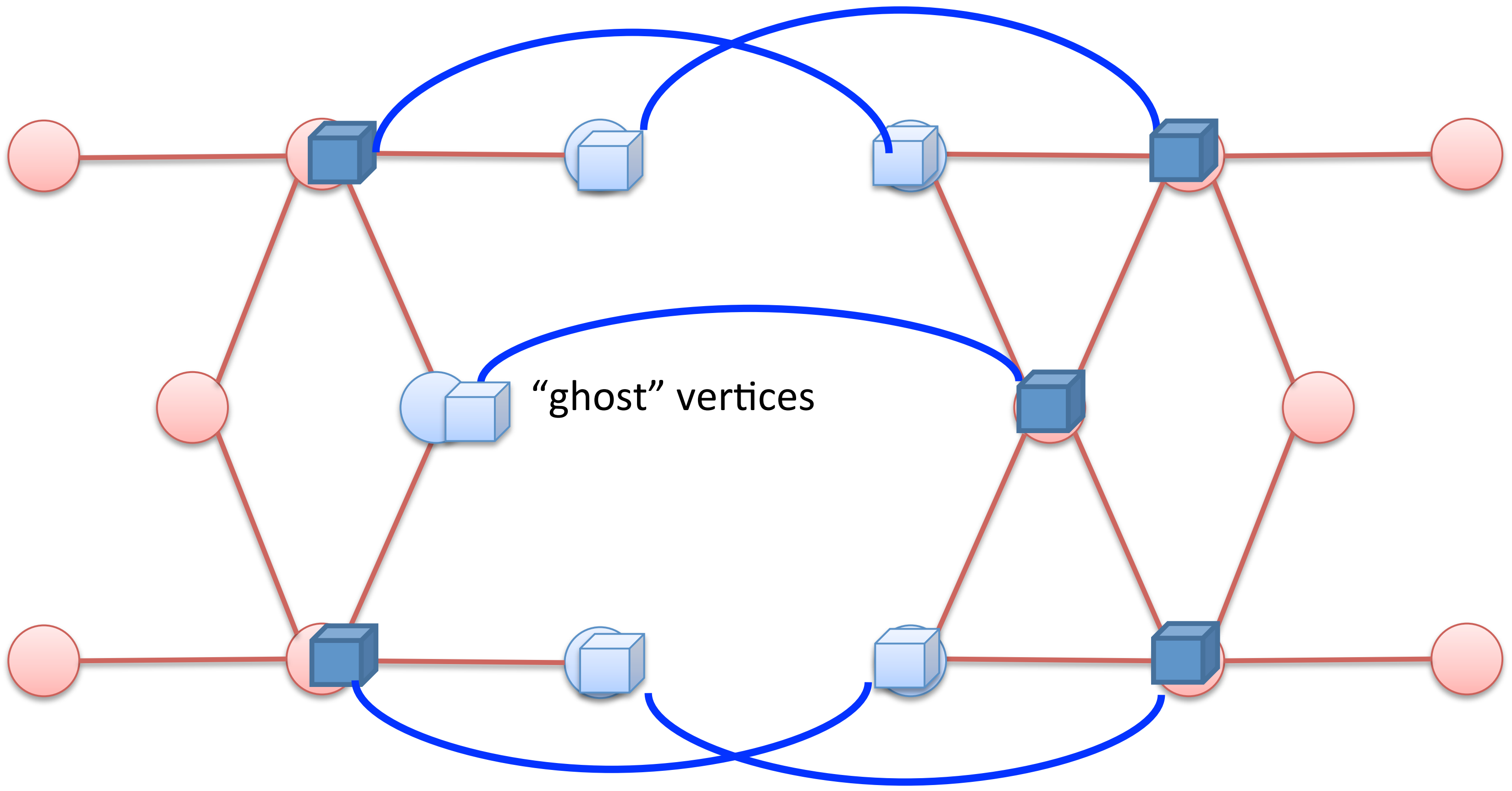
Distributed Graph

Partition the graph across multiple machines.



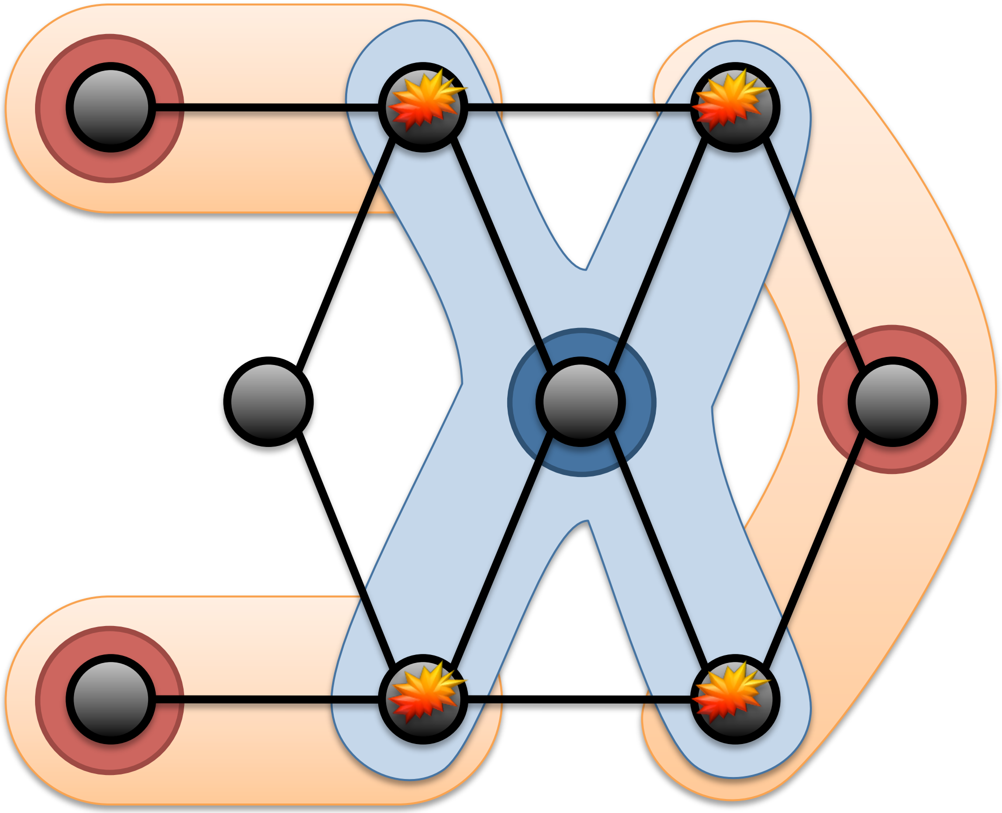
Distributed Graph

- Ghost vertices maintain adjacency structure and replicate remote data.

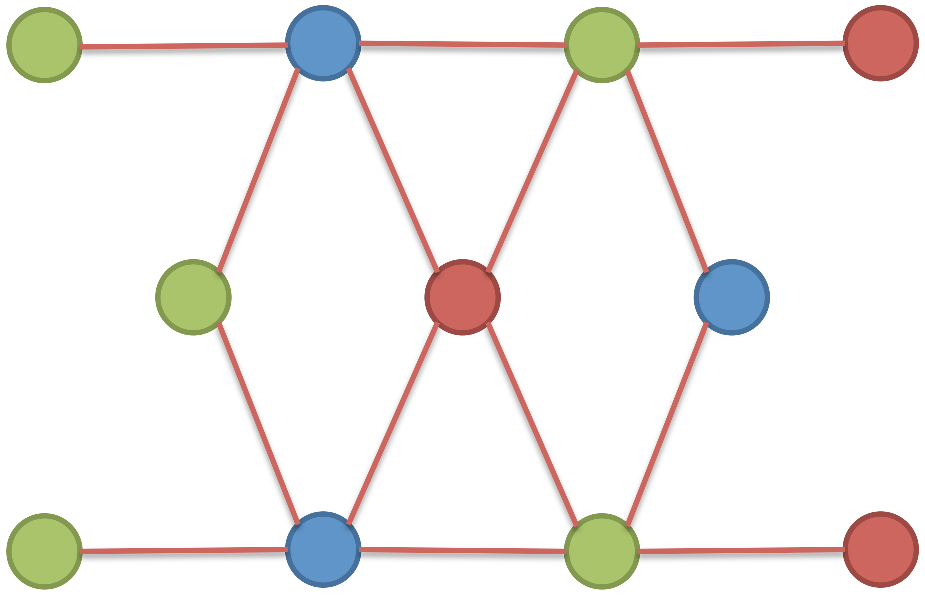


Ensuring Race-Free Code

How much can computation **overlap**?



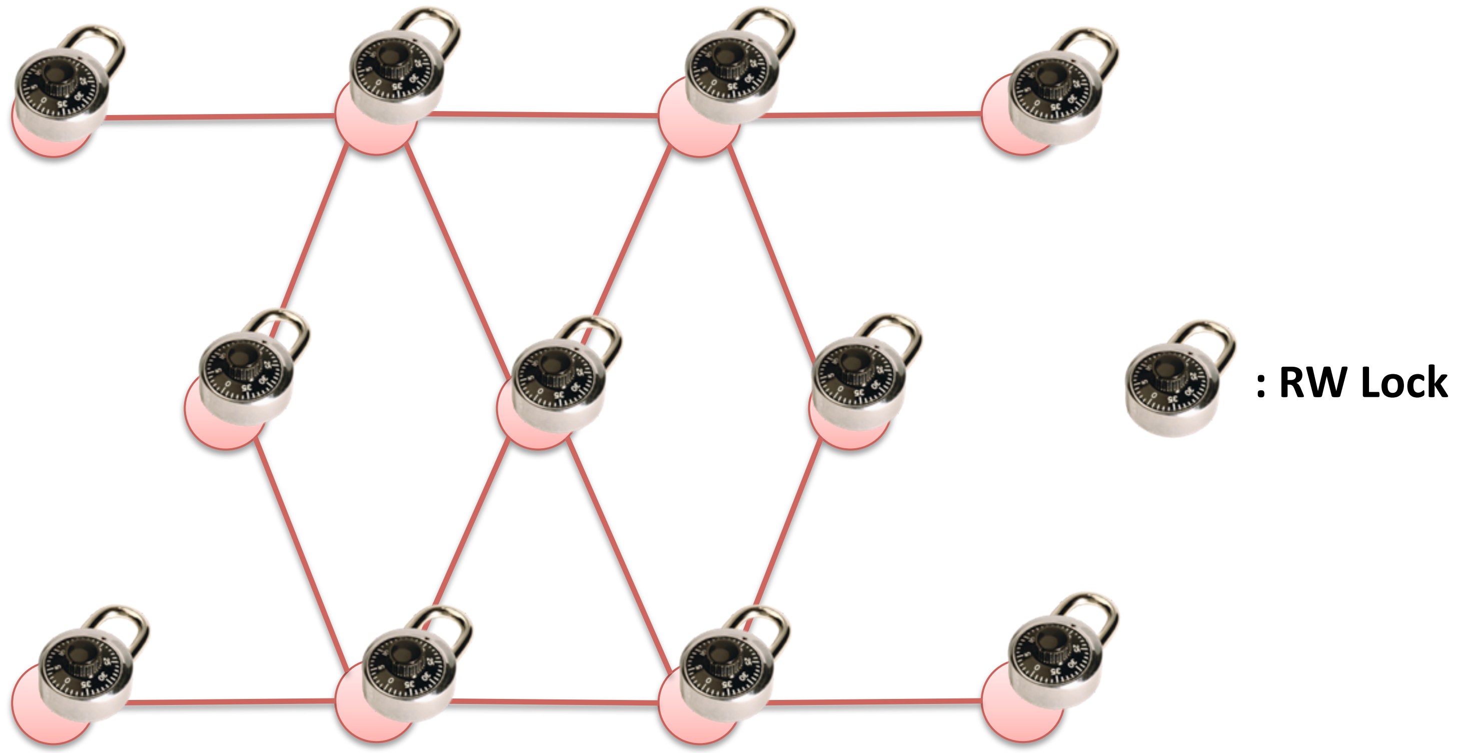
Edge Consistency via Graph Coloring



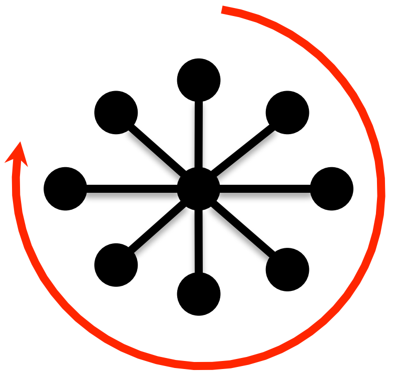
Vertices of the same color are all at least one vertex apart.
Therefore, All vertices of the same color can be run in parallel!

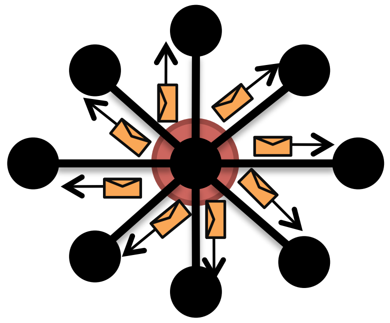
Distributed Locking

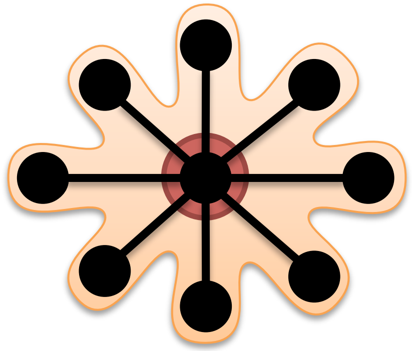
Edge Consistency can be guaranteed through locking.

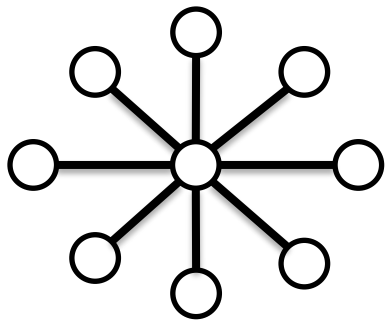


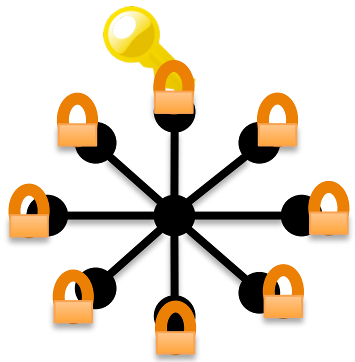
Challenges of **High-Degree** Vertices

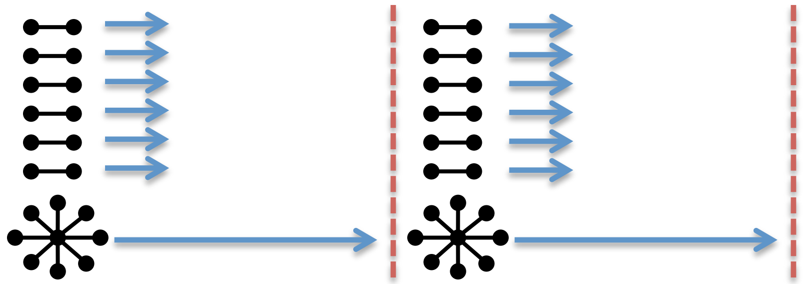

Sequentially process edges


Sends many messages (Pregel)

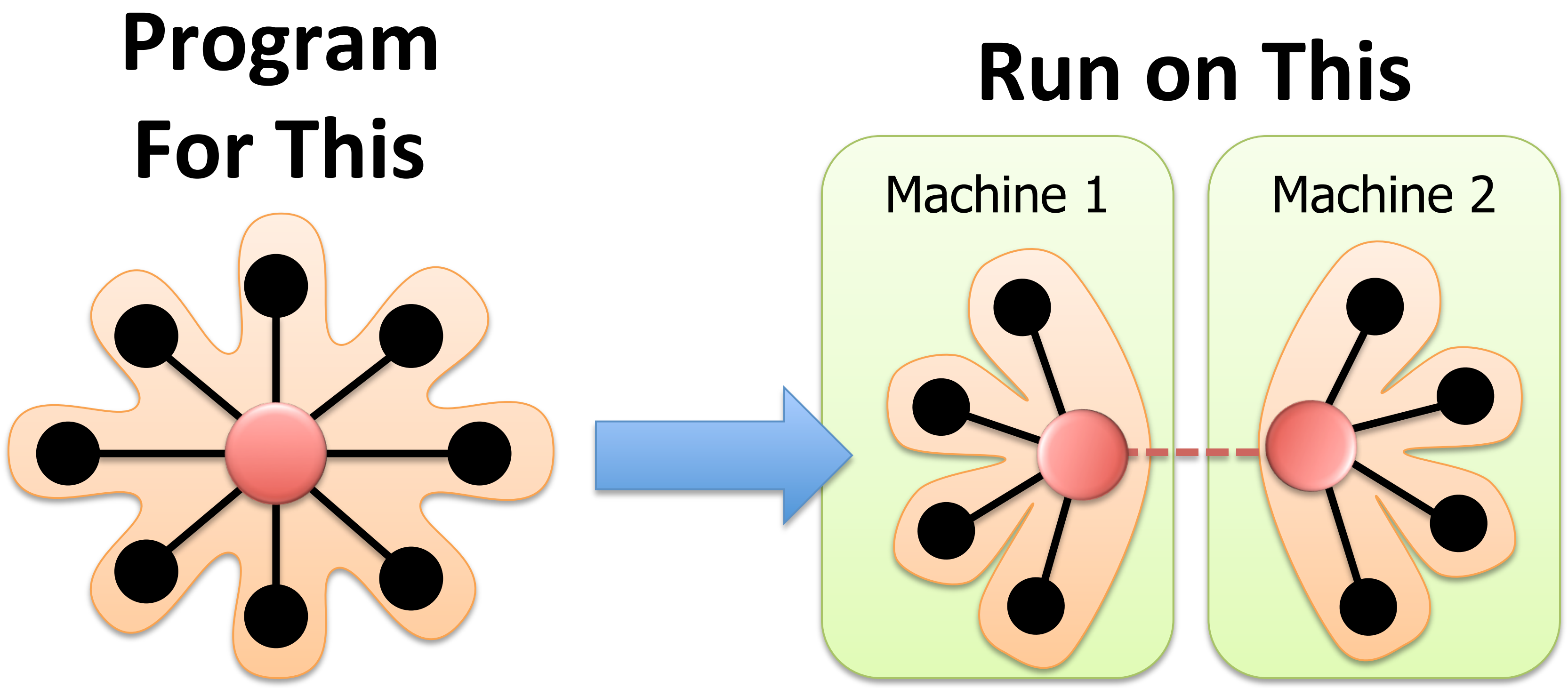

Touches a large fraction of graph (GraphLab)


Edge meta-data too large for single machine


Asynchronous Execution requires heavy locking (GraphLab)


Synchronous Execution prone to stragglers (Pregel)

PowerGraph



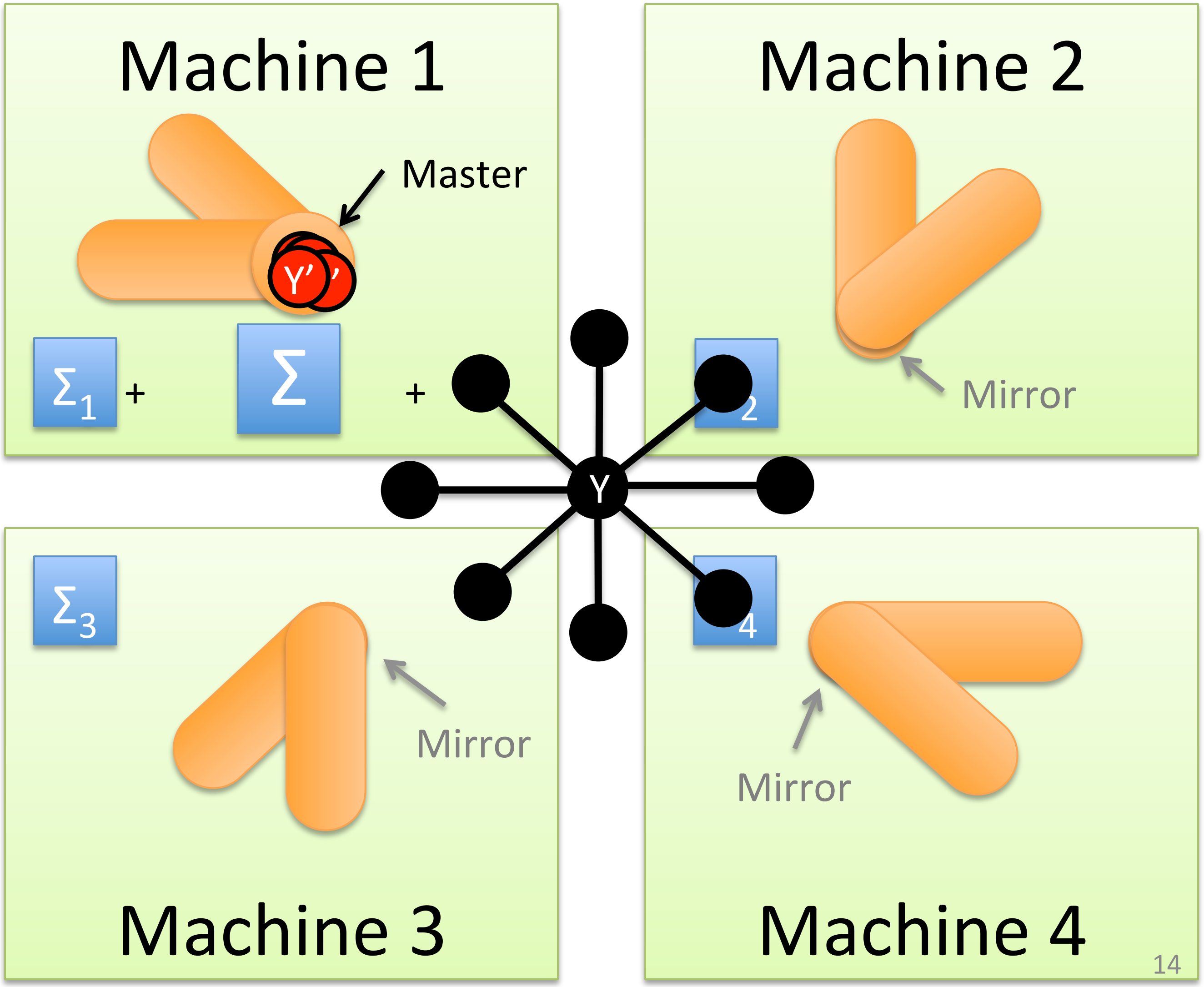
- Split **High-Degree** vertices
- **New Abstraction →** ***<u>Equivalence</u>*** *on Split Vertices*

Distributed Execution of a PowerGraph Vertex-Program

Gather

Apply

Scatter



Things didn’t covered

- Checkpoint
- Graph partition
- Experiments
  - There is no standard benchmark
  - Do not take serious about
    - It is highly related to data, workload, and system
    - comparison with map/reduce
    - Scalability