Big Data Infrastructure

Session 10: Beyond MapReduce — Graph Processing

Jimmy Lin
University of Maryland
Monday, April 13, 2015

Today's Agenda

- What makes graph processing hard?
- Graph processing frameworks
- Twitter case study

What makes graph processing hard?

- Lessons learned so far:
  - Partition
  - Replicate
  - Reduce cross-partition communication
- What makes MapReduce "work"?

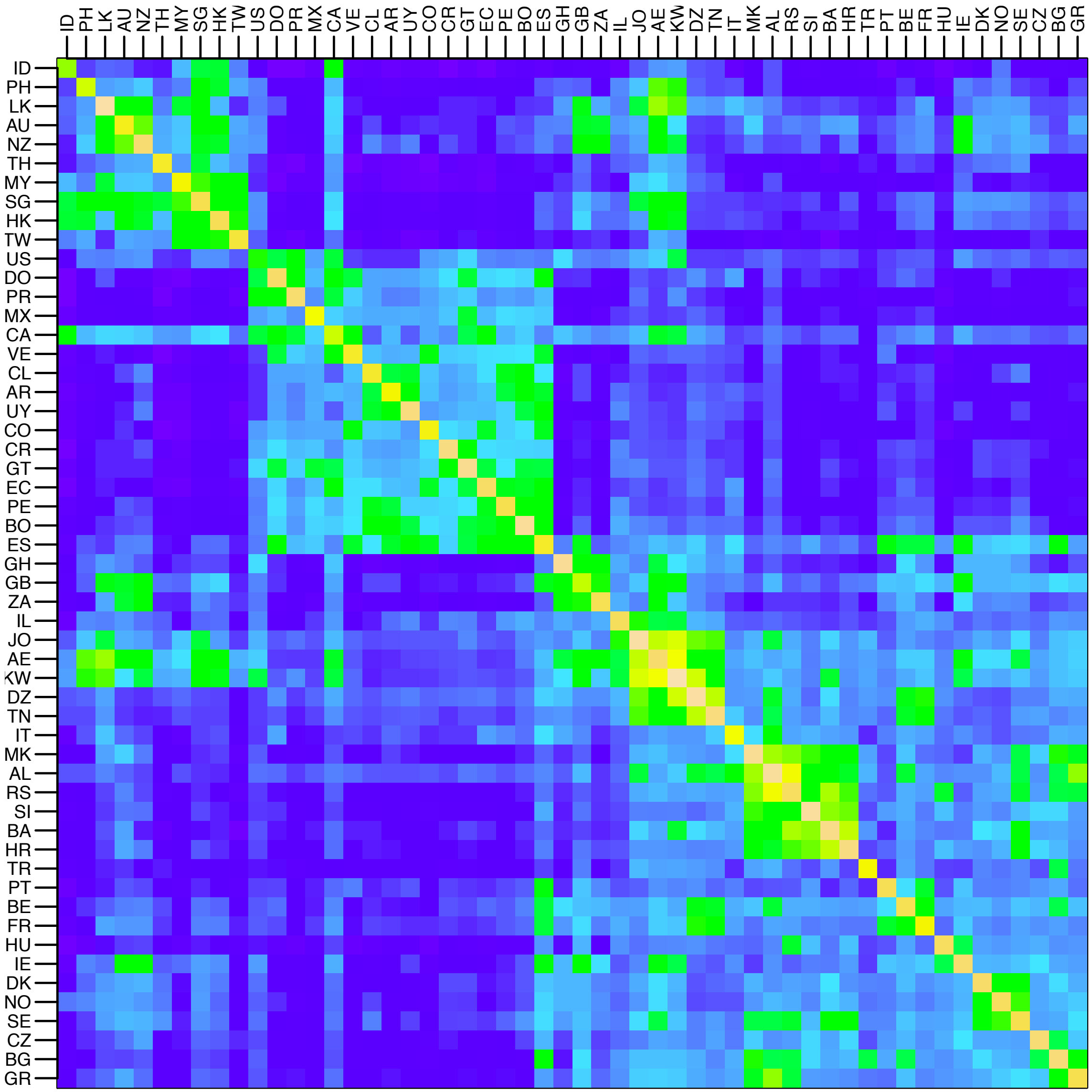
Characteristics of Graph Algorithms

- What are some common features of graph algorithms?
  - Graph traversals
  - Computations involving vertices and their neighbors
  - Passing information along graph edges
- What's the obvious idea?
  - Keep "neighborhoods" together!

Simple Partitioning Techniques

- Hash partitioning
- Range partitioning on some underlying linearization
  - Web pages: lexicographic sort of domain-reversed URLs
  - Social networks: sort by demographic characteristics

Country Structure in Facebook

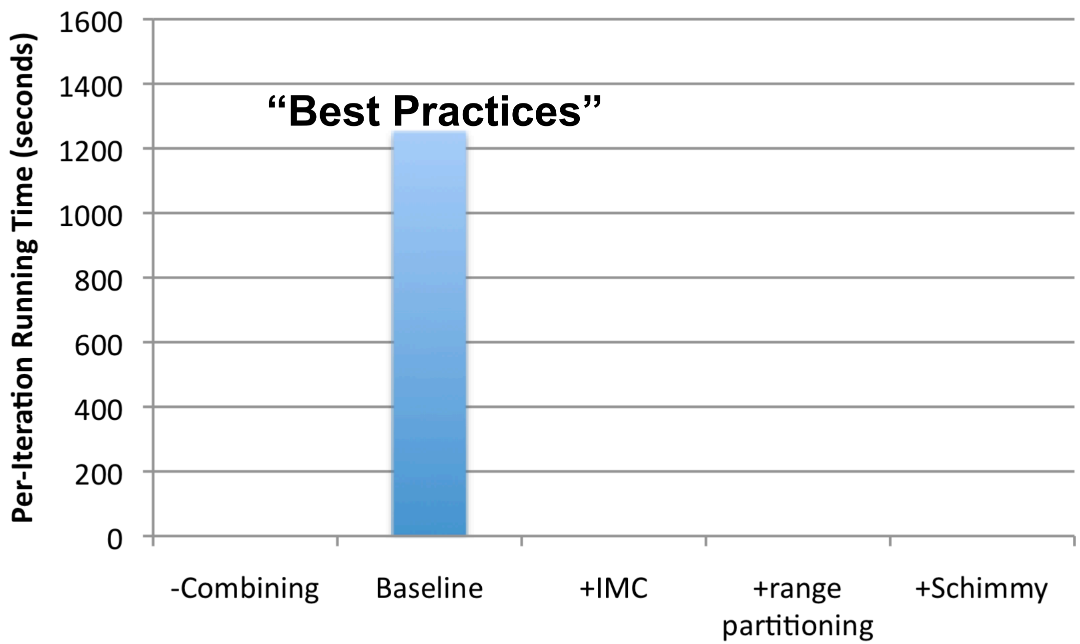


Analysis of 721 million active users (May 2011)

54 countries w/ >1m active users, >50% penetration

Ugander et al. (2011) The Anatomy of the Facebook Social Graph.

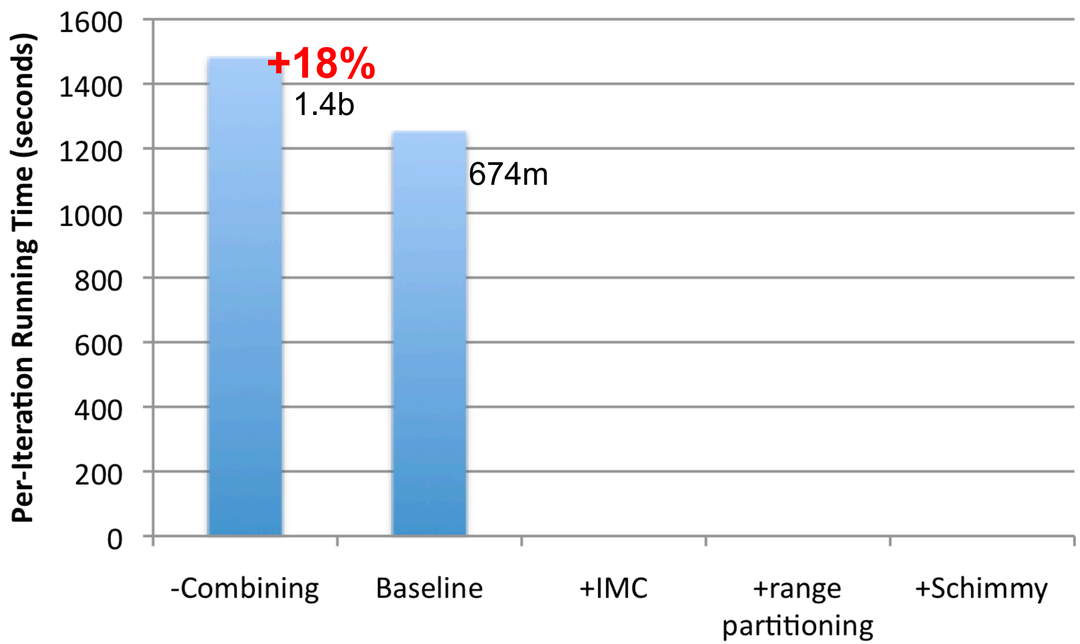
How much difference does it make?



PageRank over webgraph
(40m vertices, 1.4b edges)

Lin and Schatz. (2010) Design Patterns for Efficient Graph Algorithms in MapReduce.

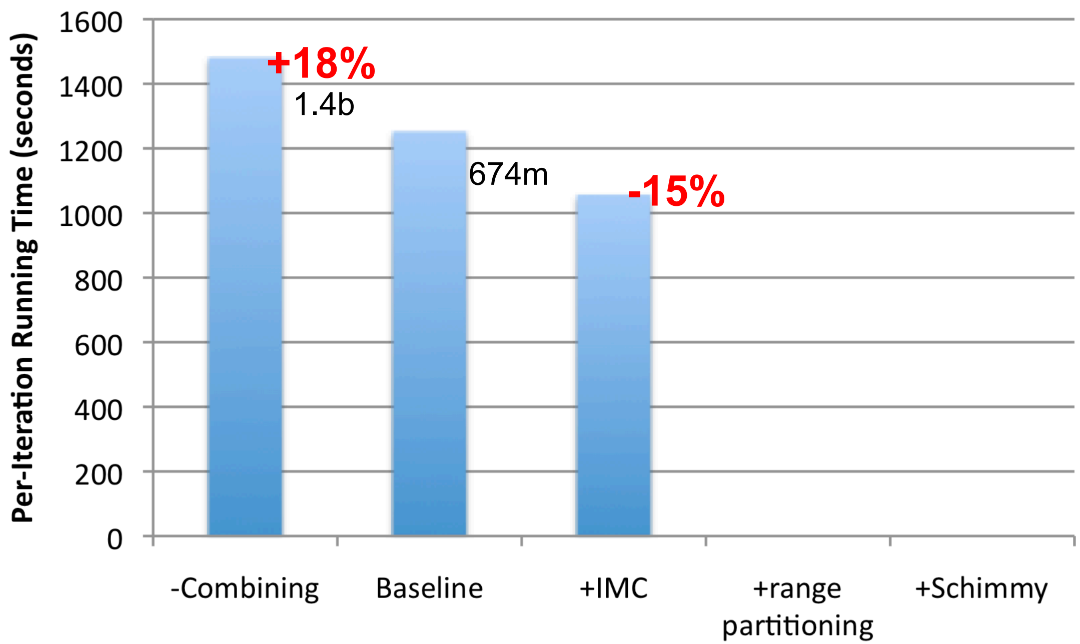
How much difference does it make?



PageRank over webgraph
(40m vertices, 1.4b edges)

Lin and Schatz. (2010) Design Patterns for Efficient Graph Algorithms in MapReduce.

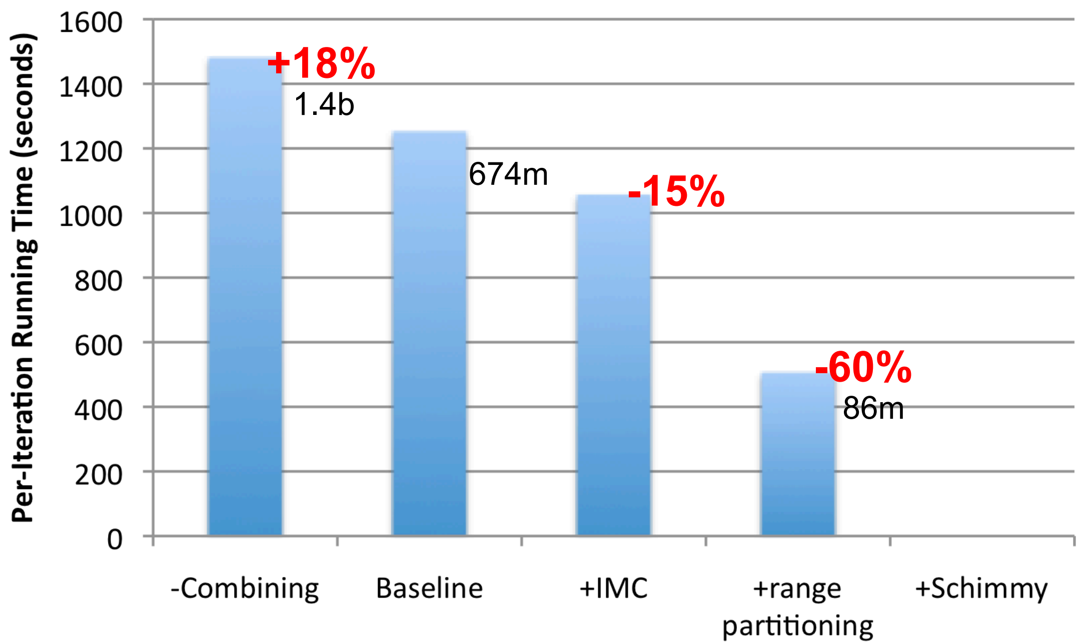
How much difference does it make?



PageRank over webgraph
(40m vertices, 1.4b edges)

Lin and Schatz. (2010) Design Patterns for Efficient Graph Algorithms in MapReduce.

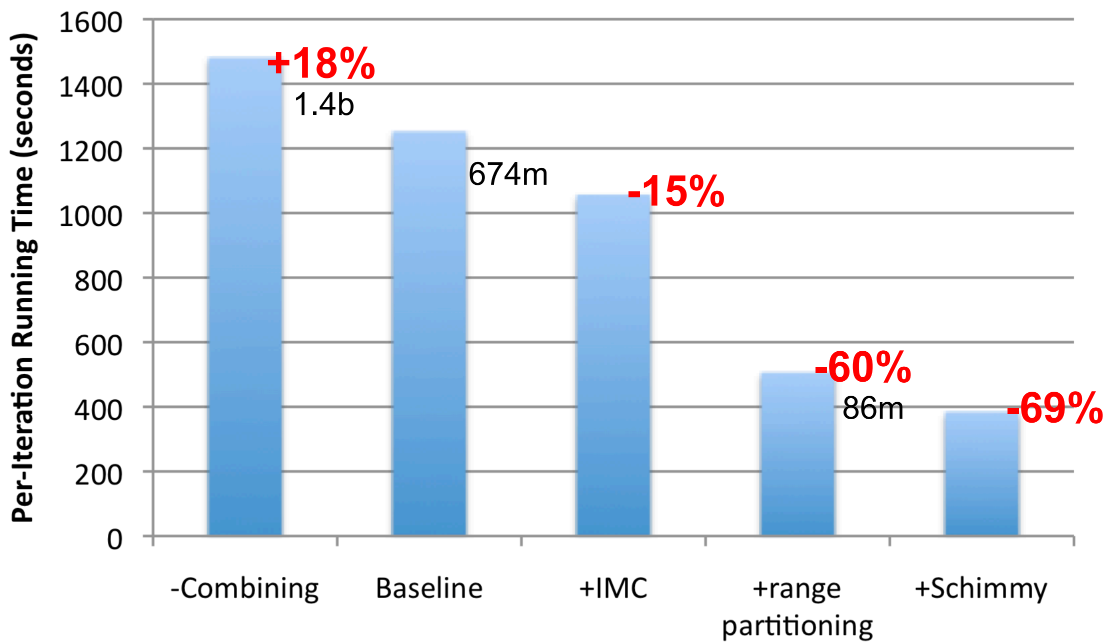
How much difference does it make?



PageRank over webgraph
(40m vertices, 1.4b edges)

Lin and Schatz. (2010) Design Patterns for Efficient Graph Algorithms in MapReduce.

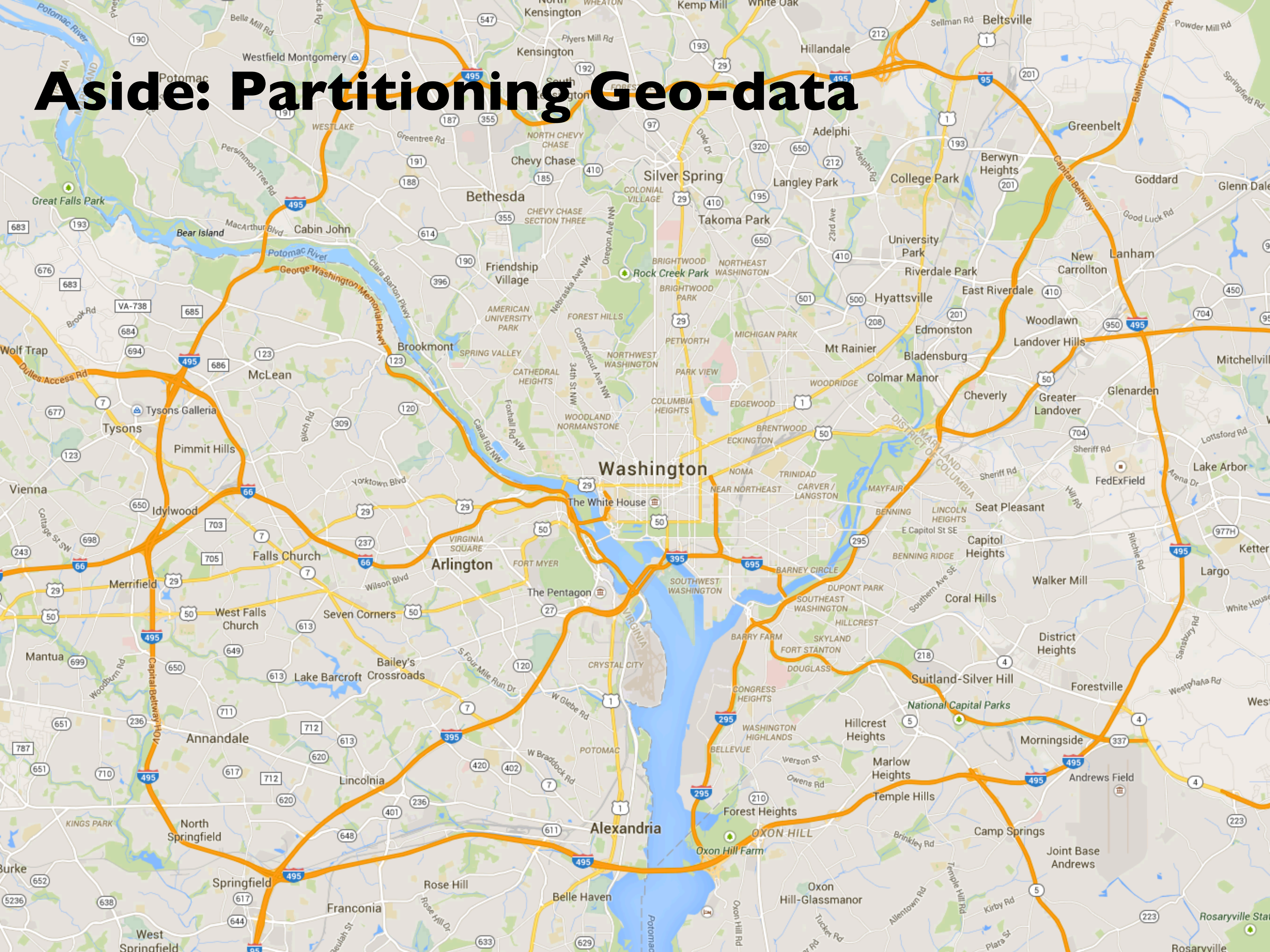
How much difference does it make?



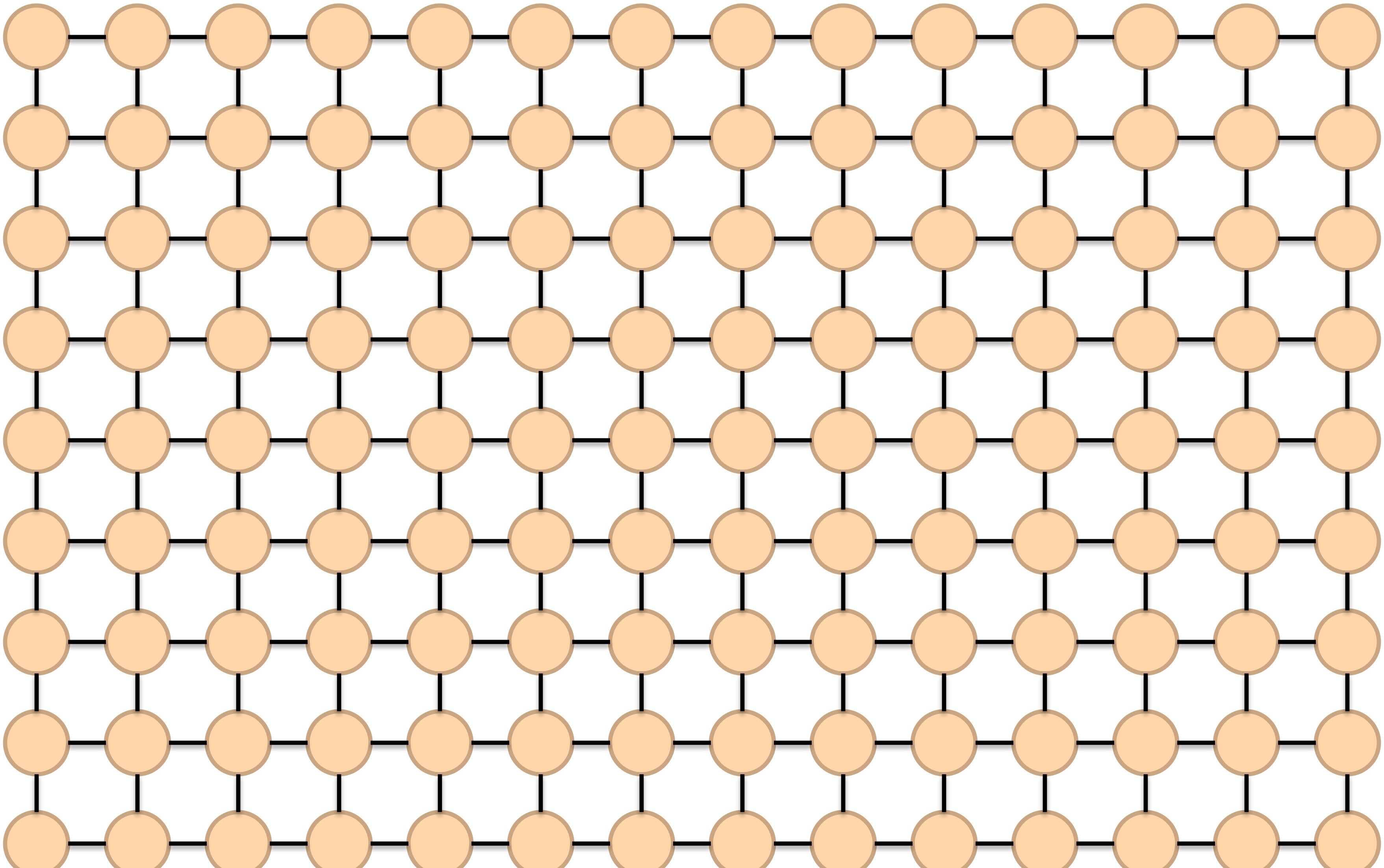
PageRank over webgraph
(40m vertices, 1.4b edges)

Lin and Schatz. (2010) Design Patterns for Efficient Graph Algorithms in MapReduce.

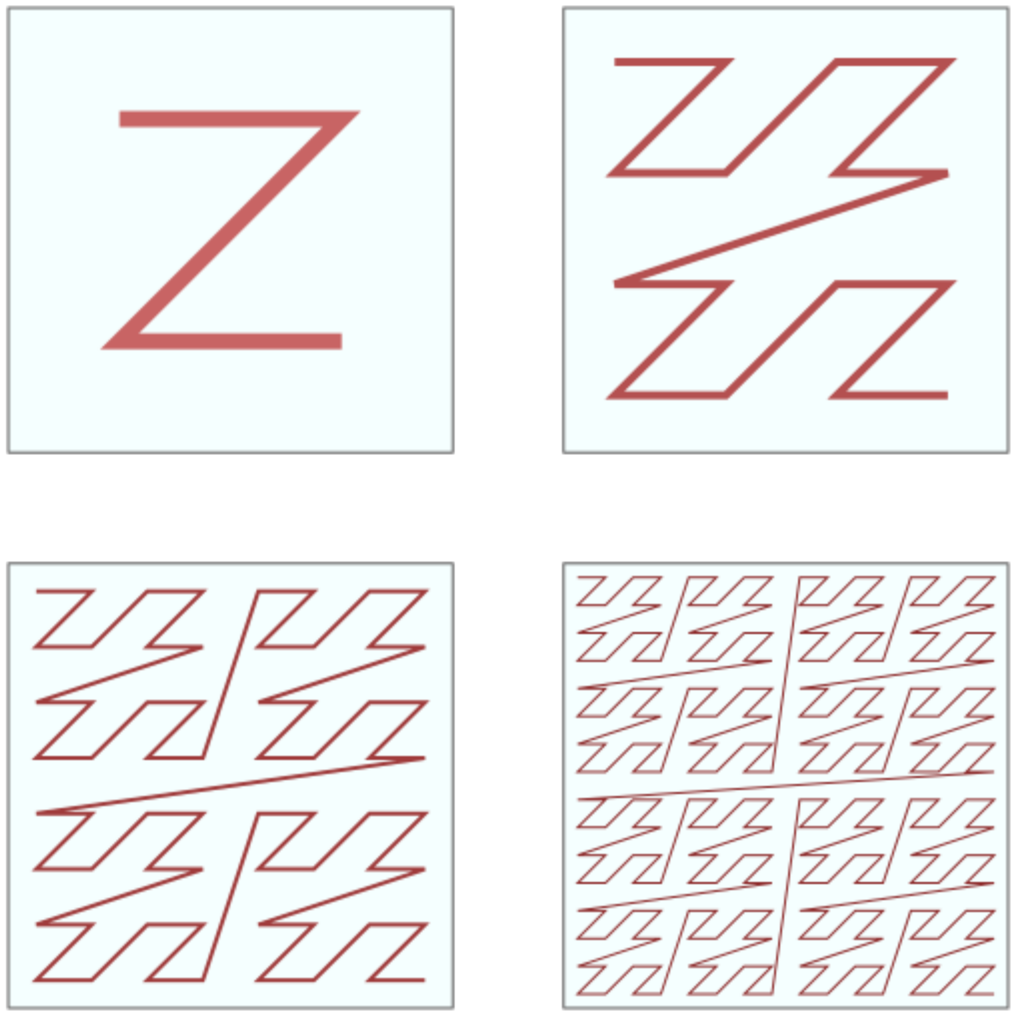
Aside: Partitioning Geo-data



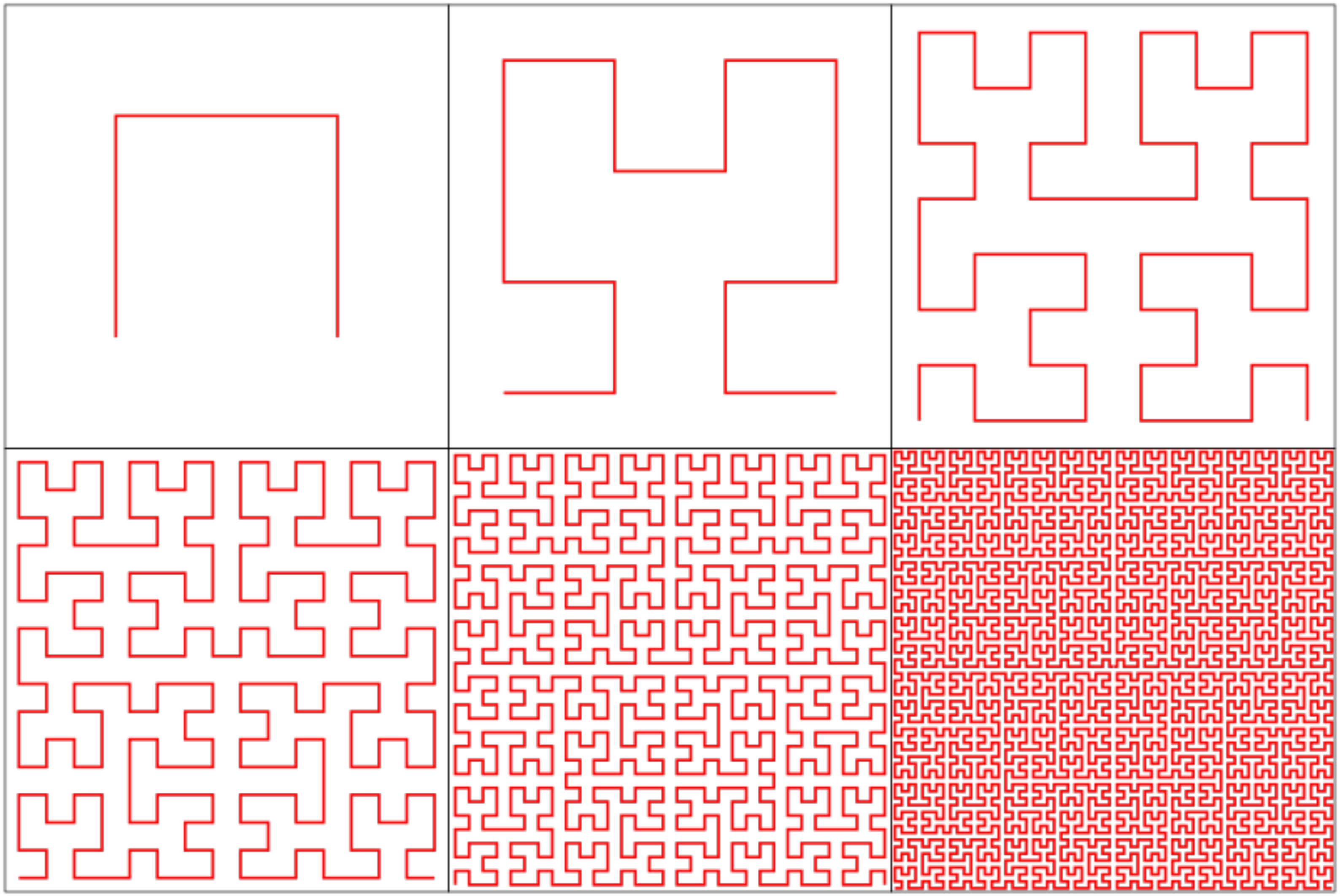
Geo-data = regular graph



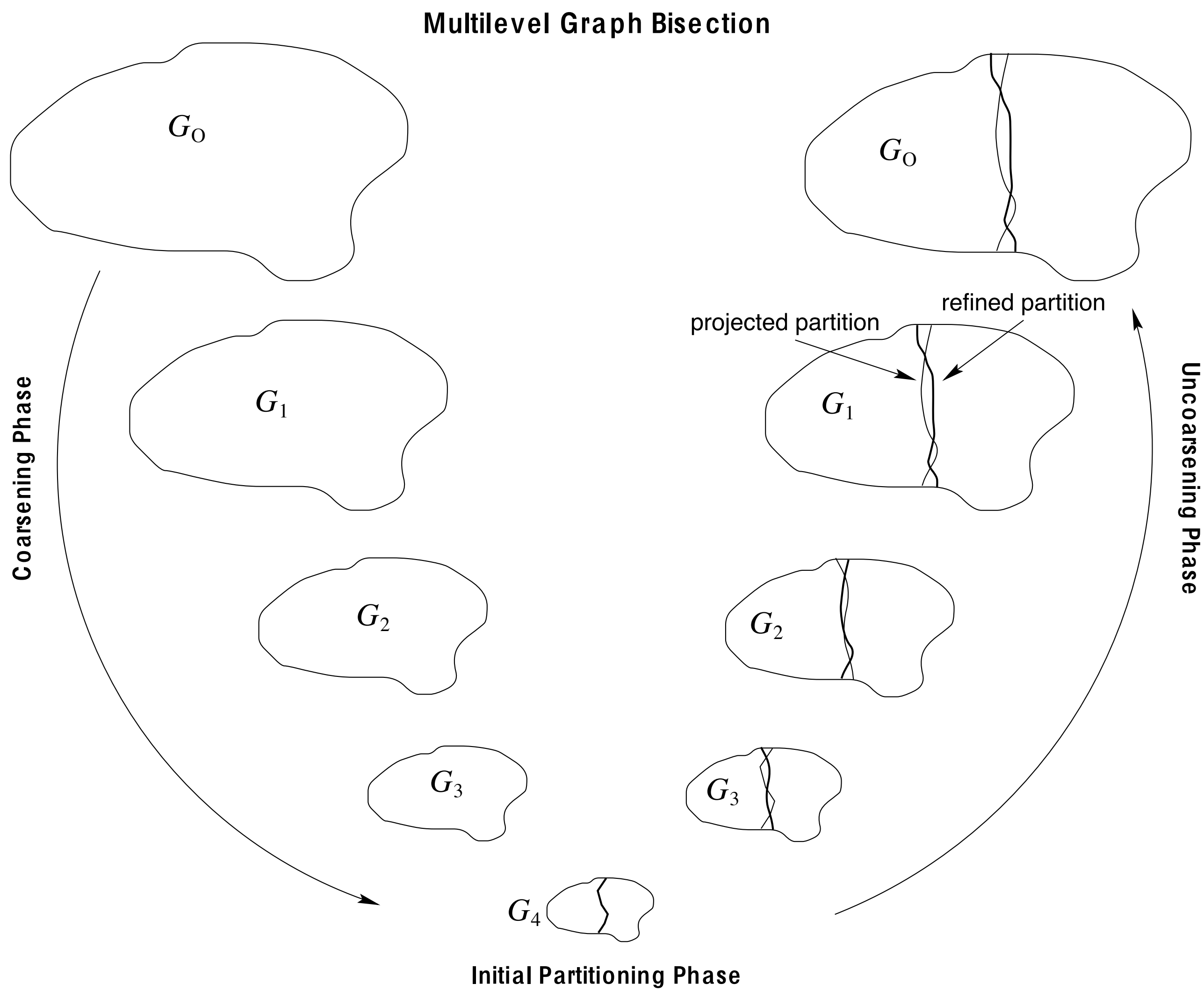
Space-filling curves: Z-Order Curves



Space-filling curves: Hilbert Curves

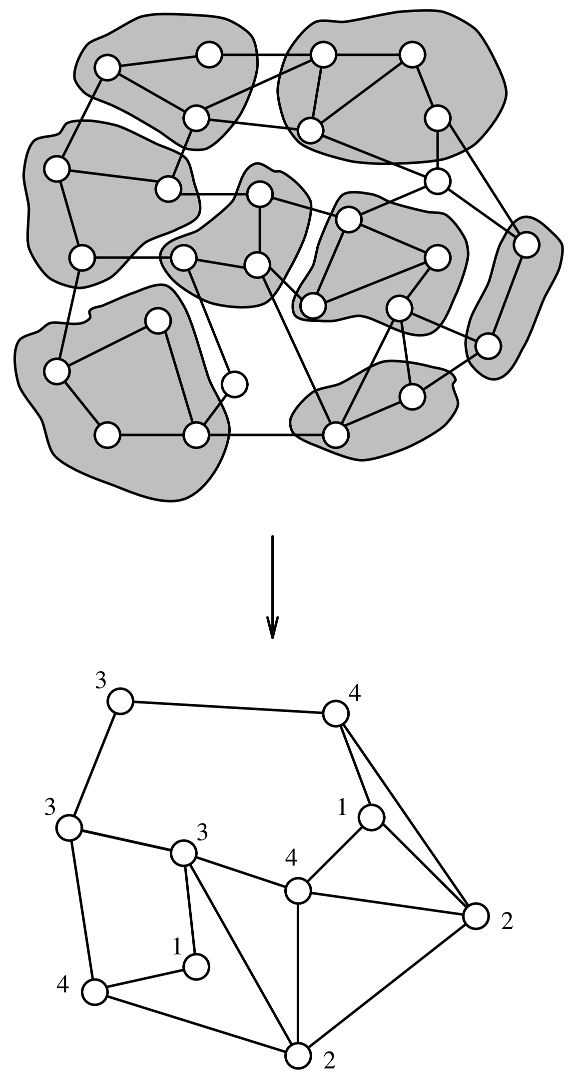


General-Purpose Graph Partitioning



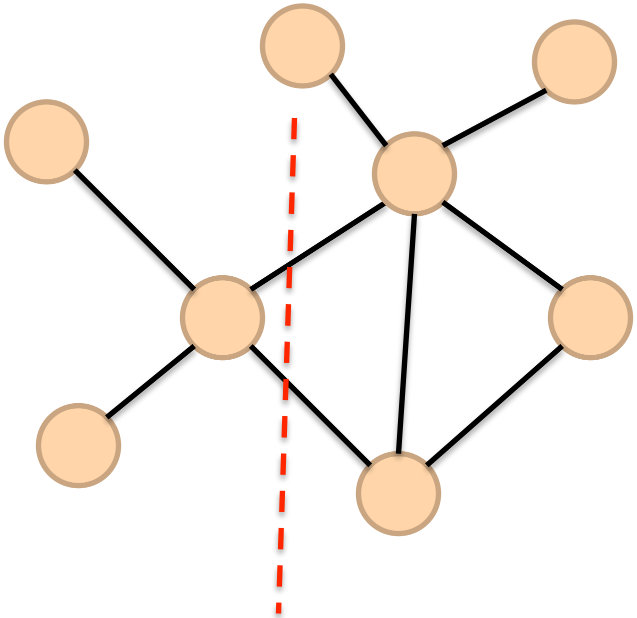
Karypis and Kumar. (1998) A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs.

Graph Coarsening

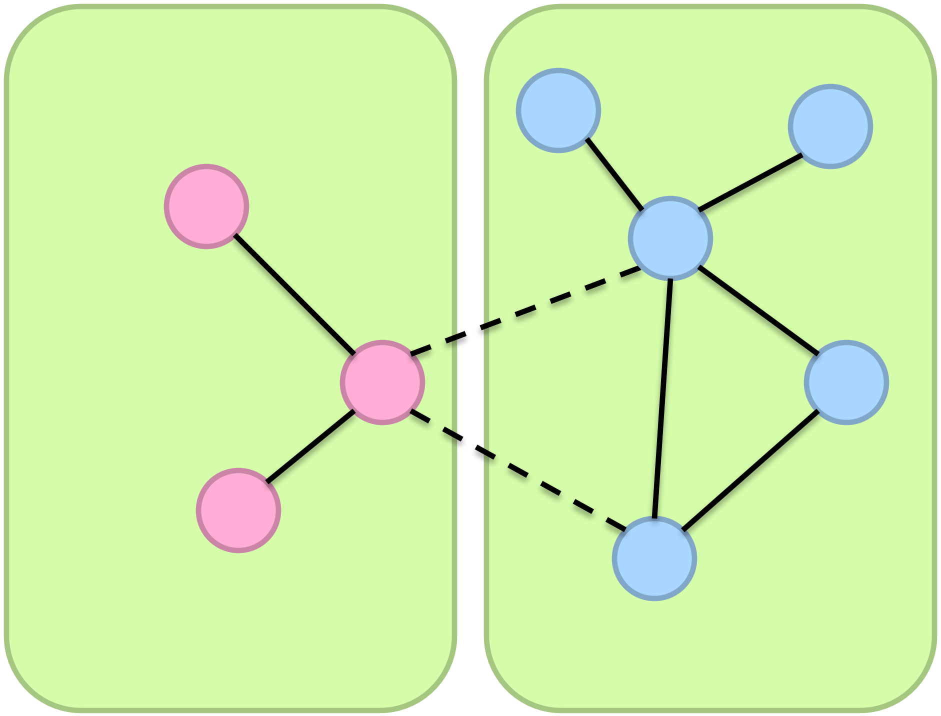


Karypis and Kumar. (1998) A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs.

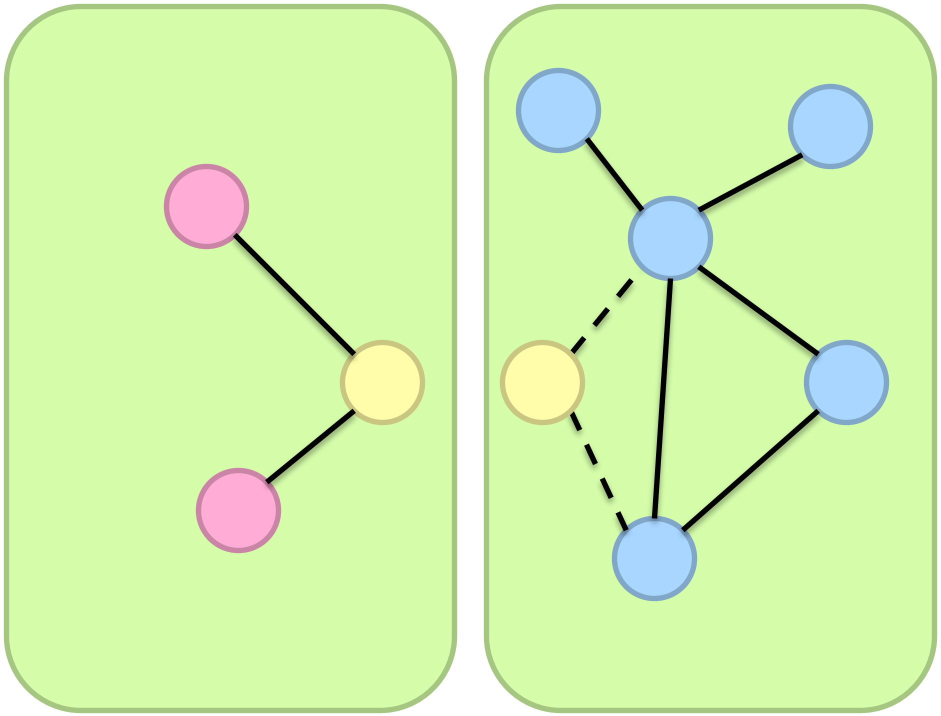
Partition



Partition



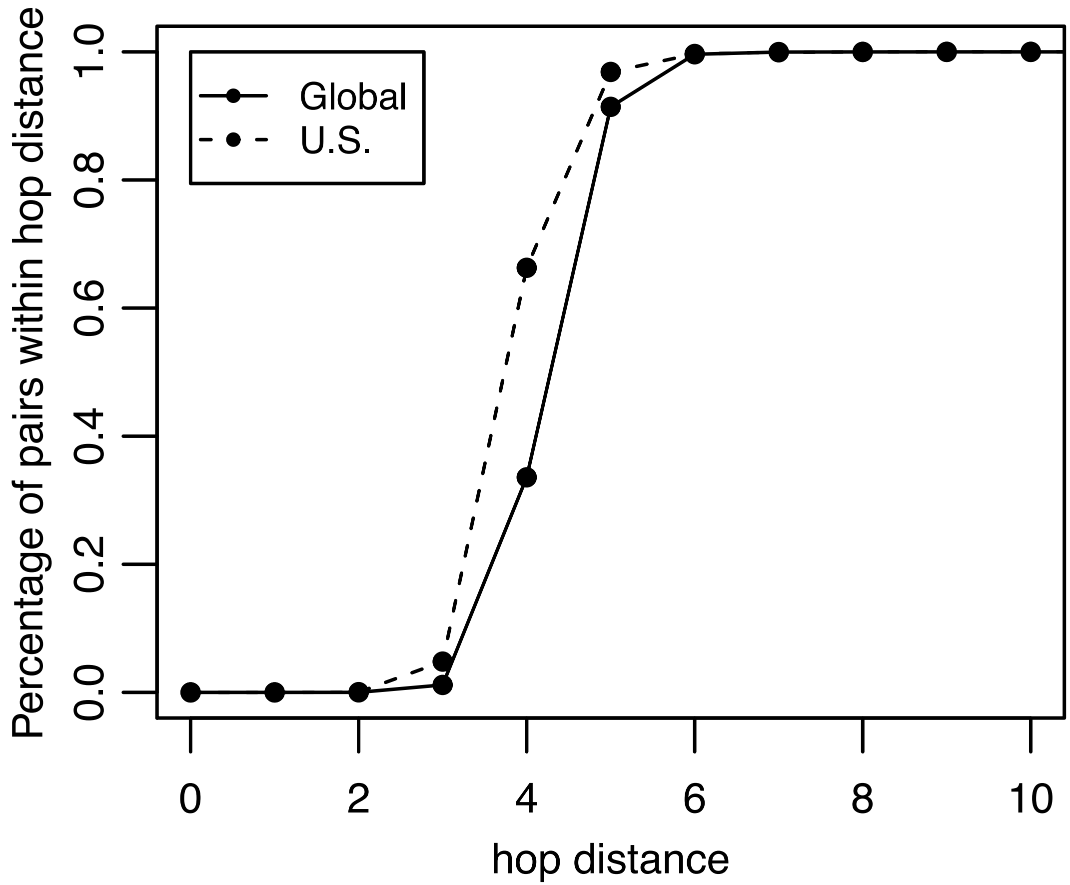
Partition + Replicate



What's the issue?
Solutions?

Neighborhood Replication

What's the cost of replicating *n*-hop neighborhoods?



What's the more general challenge?

Ugander et al. (2011) The Anatomy of the Facebook Social Graph.

What makes graph processing hard?

- It's tough to apply our "usual tricks" :
  - Partition
  - Replicate
  - Reduce cross-partition communication

Graph Processing Frameworks

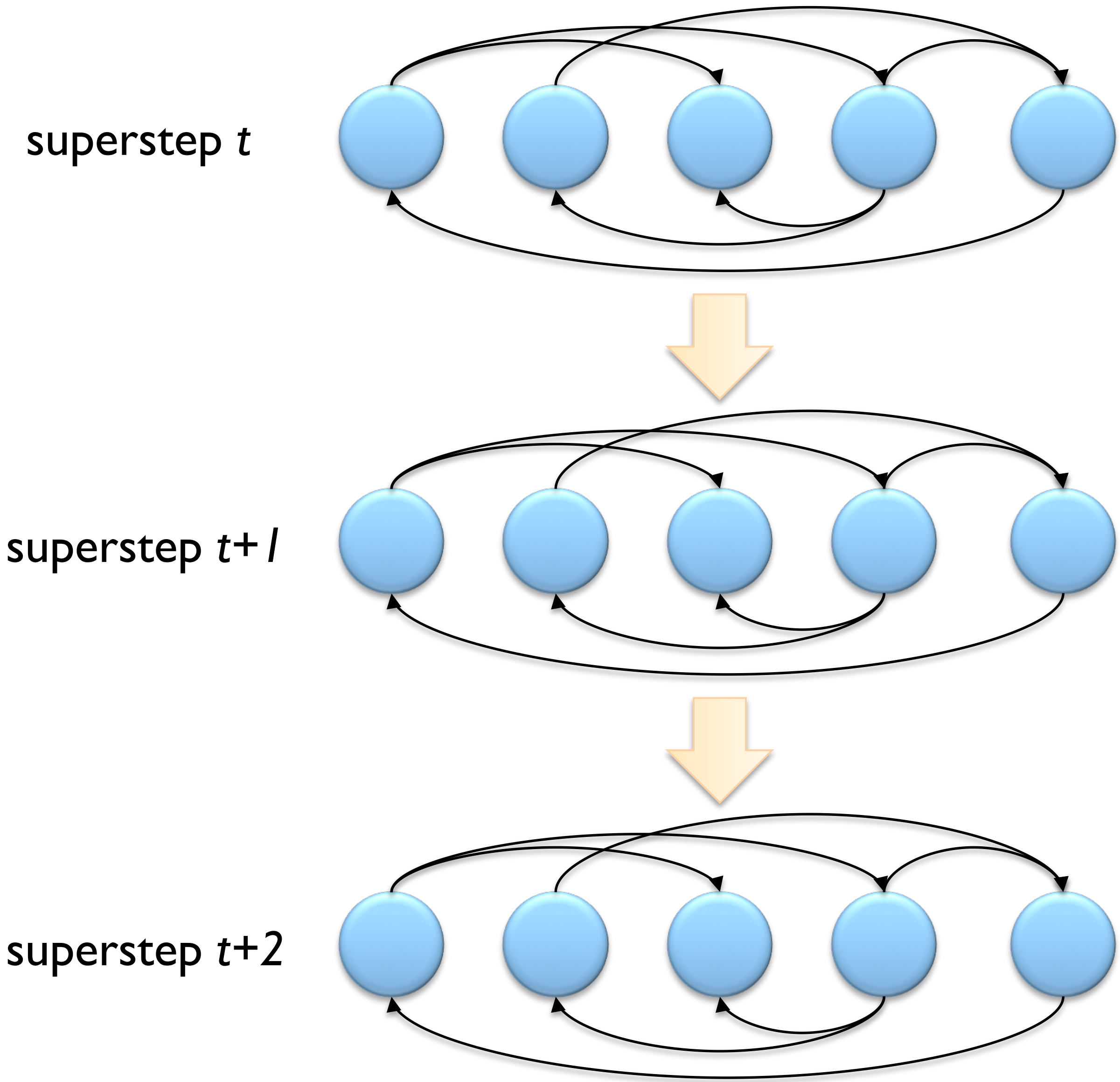
Source: Wikipedia (Waste container)

Pregel: Computational Model

- Based on Bulk Synchronous Parallel (BSP)
  - Computational units encoded in a directed graph
  - Computation proceeds in a series of supersteps
  - Message passing architecture
- Each vertex, at each superstep:
  - Receives messages directed at it from previous superstep
  - Executes a user-defined function (modifying state)
  - Emits messages to other vertices (for the next superstep)
- Termination:
  - A vertex can choose to deactivate itself
  - Is "woken up" if new messages received
  - Computation halts when all vertices are inactive

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel



Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel: Implementation

- Master-Slave architecture
  - Vertices are hash partitioned (by default) and assigned to workers
  - Everything happens in memory
- Processing cycle:
  - Master tells all workers to advance a single superstep
  - Worker delivers messages from previous superstep, executing vertex computation
  - Messages sent asynchronously (in batches)
  - Worker notifies master of number of active vertices
- Fault tolerance
  - Checkpointing
  - Heartbeat/revert

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel: PageRank

```
class PageRankVertex : public Vertex<double, void, double> {
public:
  virtual void Compute(MessageIterator* msgs) {
    if (superstep() >= 1) {
      double sum = 0;
      for (; !msgs->Done(); msgs->Next())
        sum += msgs->Value();
      *MutableValue() = 0.15 / NumVertices() + 0.85 * sum;
    }

    if (superstep() < 30) {
      const int64 n = GetOutEdgeIterator().size();
      SendMessageToAllNeighbors(GetValue() / n);
    } else {
      VoteToHalt();
    }
  }
};
```

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel: SSSP

```
class ShortestPathVertex : public Vertex<int, int, int> {
  void Compute(MessageIterator* msgs) {
    int mindist = IsSource(vertex_id()) ? 0 : INF;
    for (; !msgs->Done(); msgs->Next())
      mindist = min(mindist, msgs->Value());
      if (mindist < GetValue()) {
        *MutableValue() = mindist;
        OutEdgeIterator iter = GetOutEdgeIterator();
      for (; !iter.Done(); iter.Next())
        SendMessageTo(iter.Target(),
        mindist + iter.GetValue());
      }
    VoteToHalt();
  }
};
```

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

Pregel: Combiners

```
class MinIntCombiner : public Combiner<int> {
  virtual void Combine(MessageIterator* msgs) {

  int mindist = INF;
  for (; !msgs->Done(); msgs->Next())
    mindist = min(mindist, msgs->Value());
    Output("combined_source", mindist);
  }

};
```

Source: Malewicz et al. (2010) Pregel: A System for Large-Scale Graph Processing. SIGMOD.

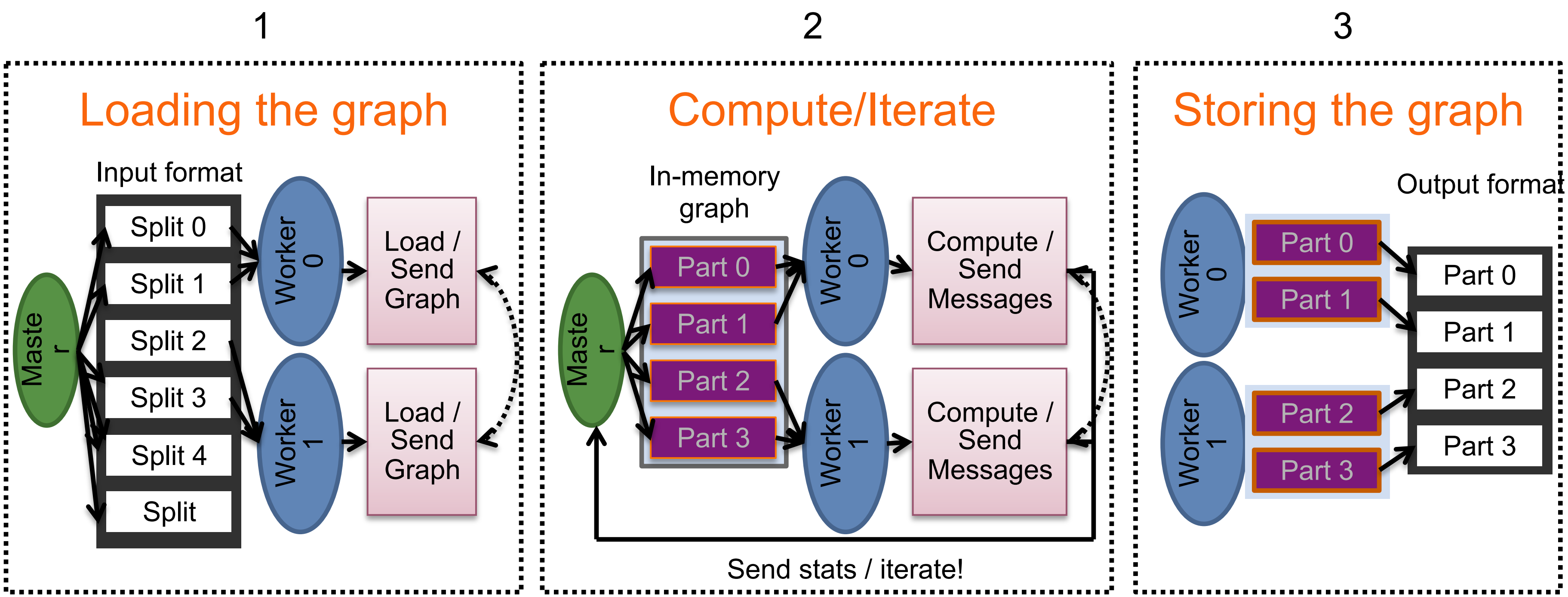
APACHE

GIRAPH

Giraph Architecture

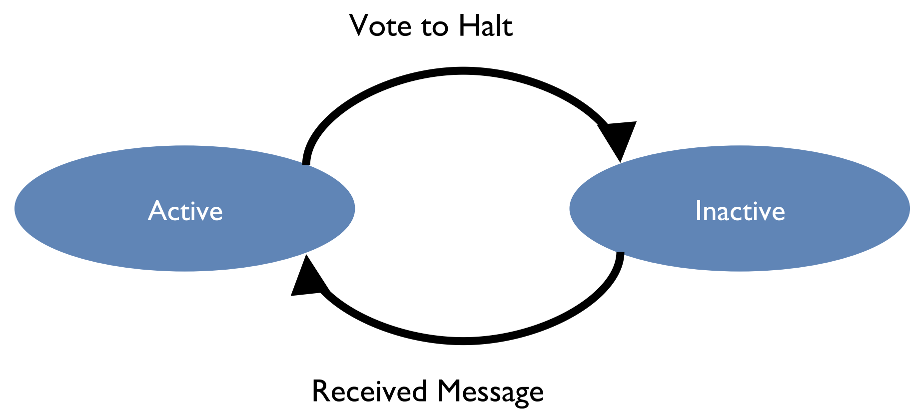
- Master – Application coordinator
  - Synchronizes supersteps
  - Assigns partitions to workers before superstep begins
- Workers – Computation & messaging
  - Handle I/O – reading and writing the graph
  - Computation/messaging of assigned partitions
- ZooKeeper
  - Maintains global application state

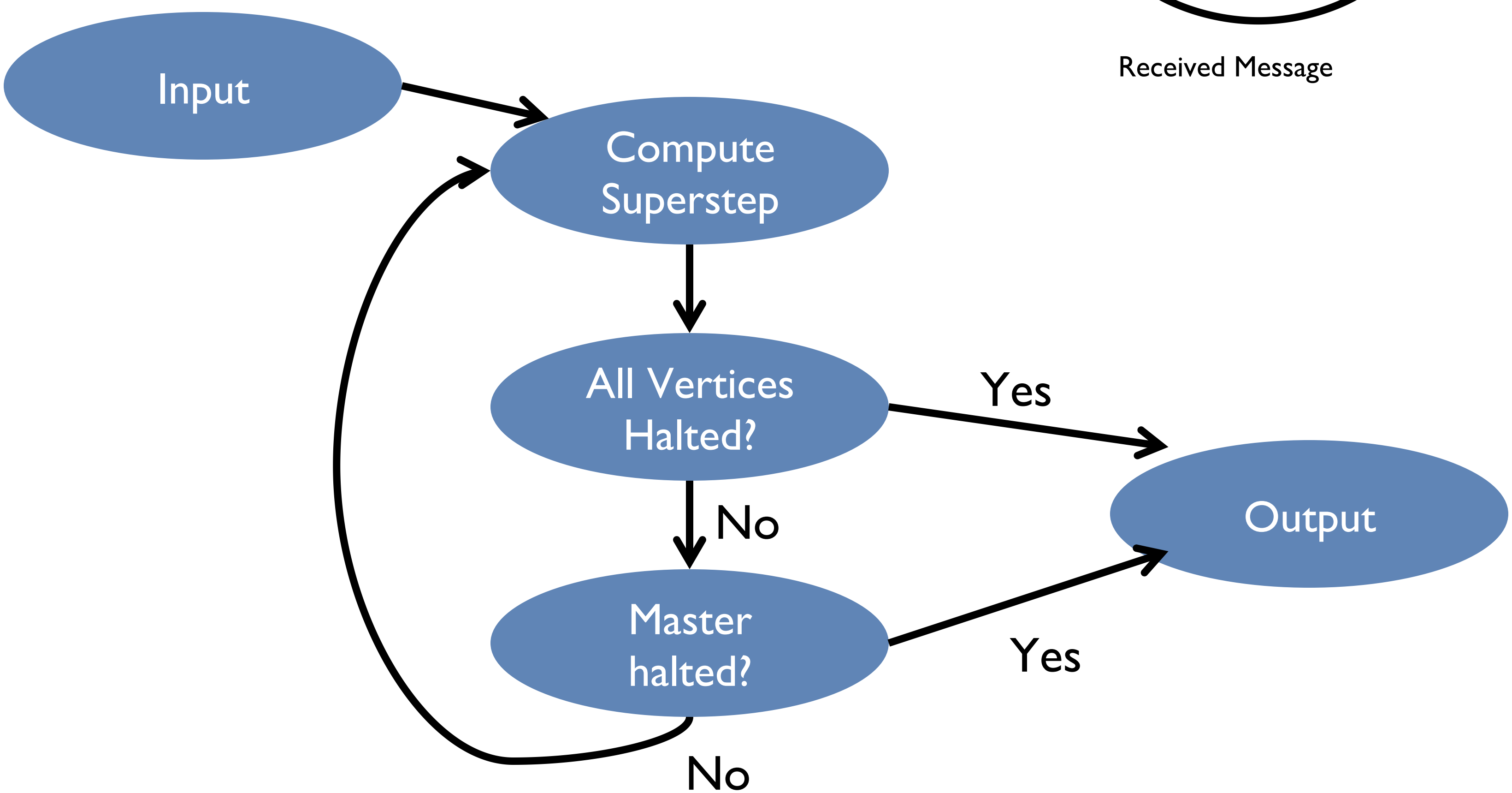
Giraph Dataflow



Giraph Lifecycle

Vertex Lifecycle

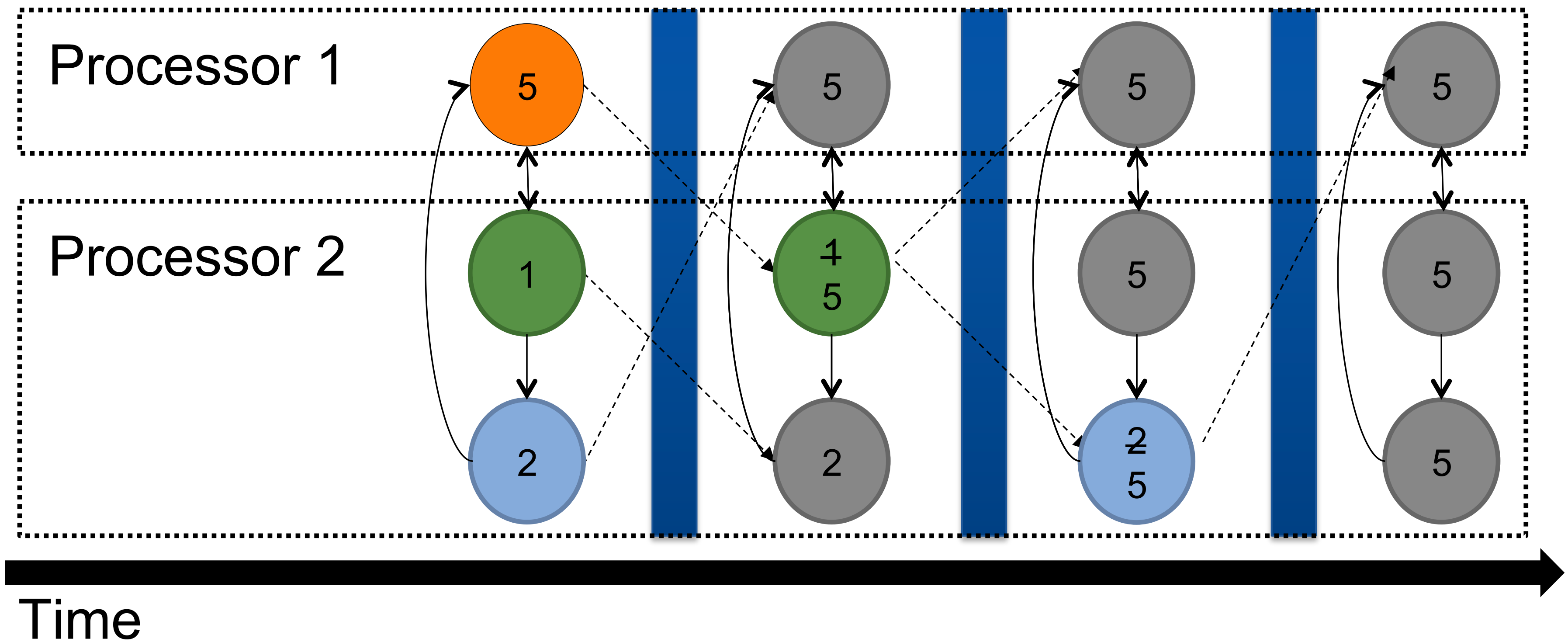




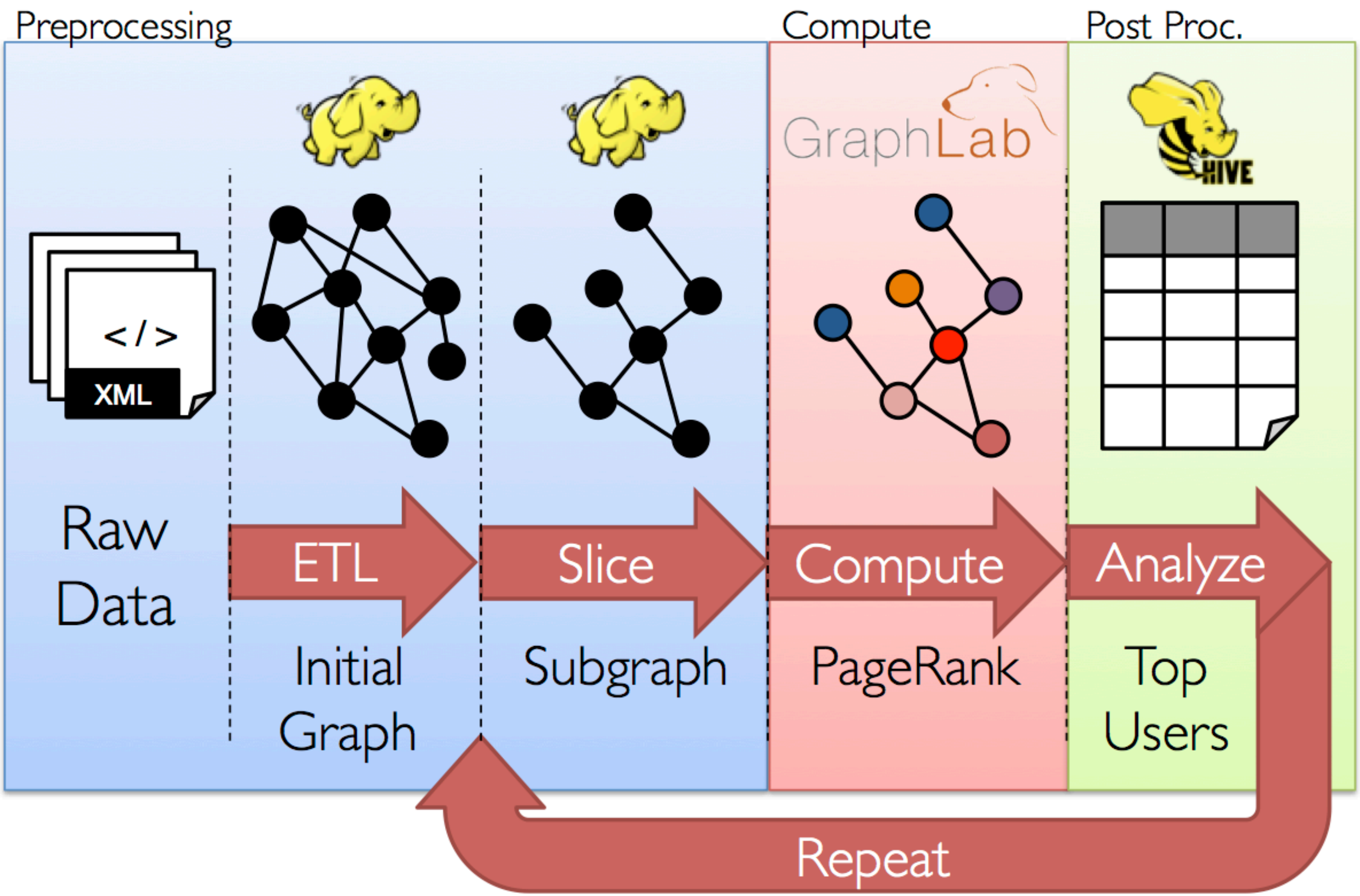
Giraph Example

```java
public class MaxComputation extends BasicComputation<IntWritable, IntWritable,
    NullWritable, IntWritable> {
  @Override
  public void compute(Vertex<IntWritable, IntWritable, NullWritable> vertex,
      Iterable<IntWritable> messages) throws IOException
  {
    boolean changed = false;
    for (IntWritable message : messages) {
      if (vertex.getValue().get() < message.get()) {
        vertex.setValue(message);
        changed = true;
      }
    }
    if (getSuperstep() == 0 || changed) {
      sendMessageToAllEdges(vertex, vertex.getValue());
    }
    vertex.voteToHalt();
  }
}
```

Execution Trace



GraphX: Motivation

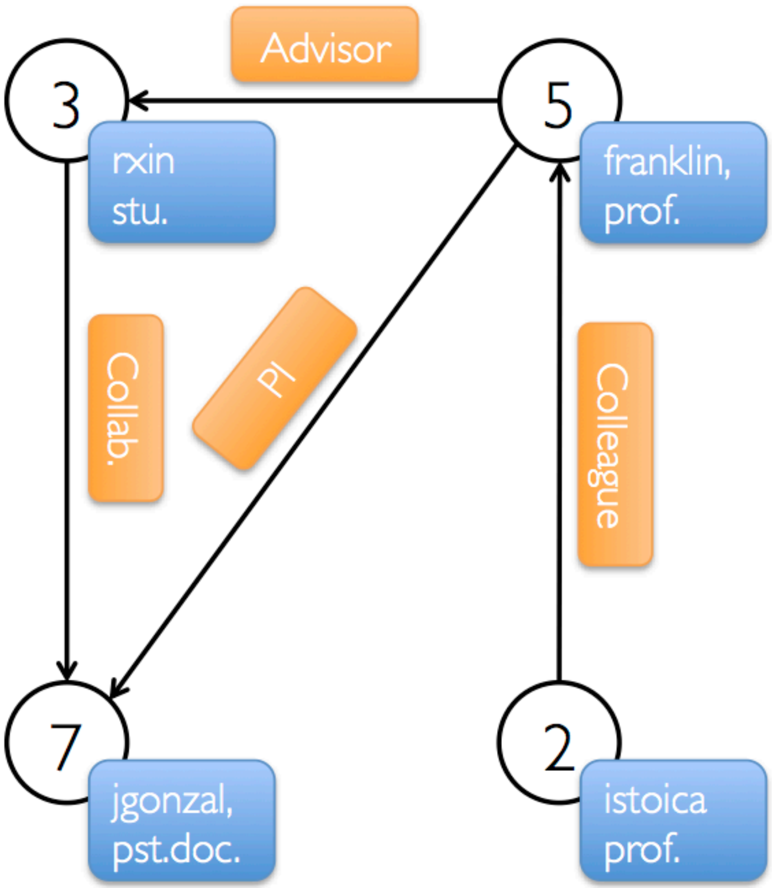


GraphX = Spark for Graphs

- Integration of record-oriented and graph-oriented processing
- Extends RDDs to Resilient Distributed Property Graphs
- Property graphs:
  - Present different views of the graph (vertices, edges, triplets)
  - Support map-like operations
  - Support distributed Pregel-like aggregations

Property Graph: Example

Property Graph



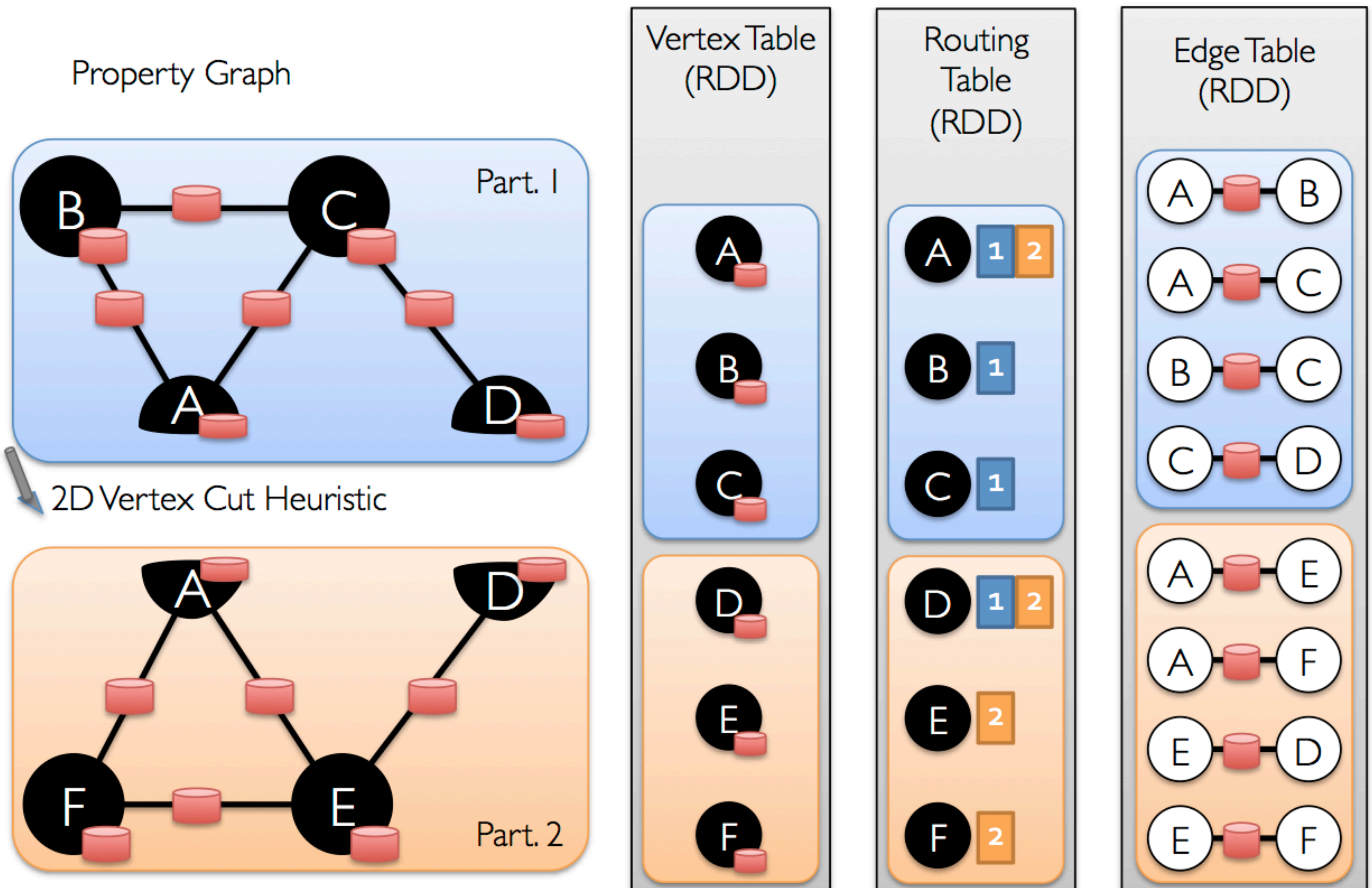
Vertex Table

Id	Property (V)
3	(rxin, student)
7	(jgonzal, postdoc)
5	(franklin, professor)
2	(istoica, professor)

Edge Table

SrcId	DstId	Property (E)
3	7	Collaborator
5	3	Advisor
2	5	Colleague
5	7	PI

Underneath the Covers



Today's Agenda

- What makes graph processing hard?
- Graph processing frameworks
- Twitter case study

Questions?

Source: Wikipedia (Japanese rock garden)