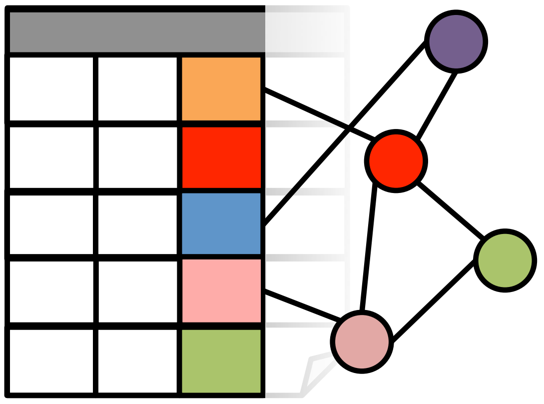
GraphX: Graph Processing in a Distributed Dataflow Framework



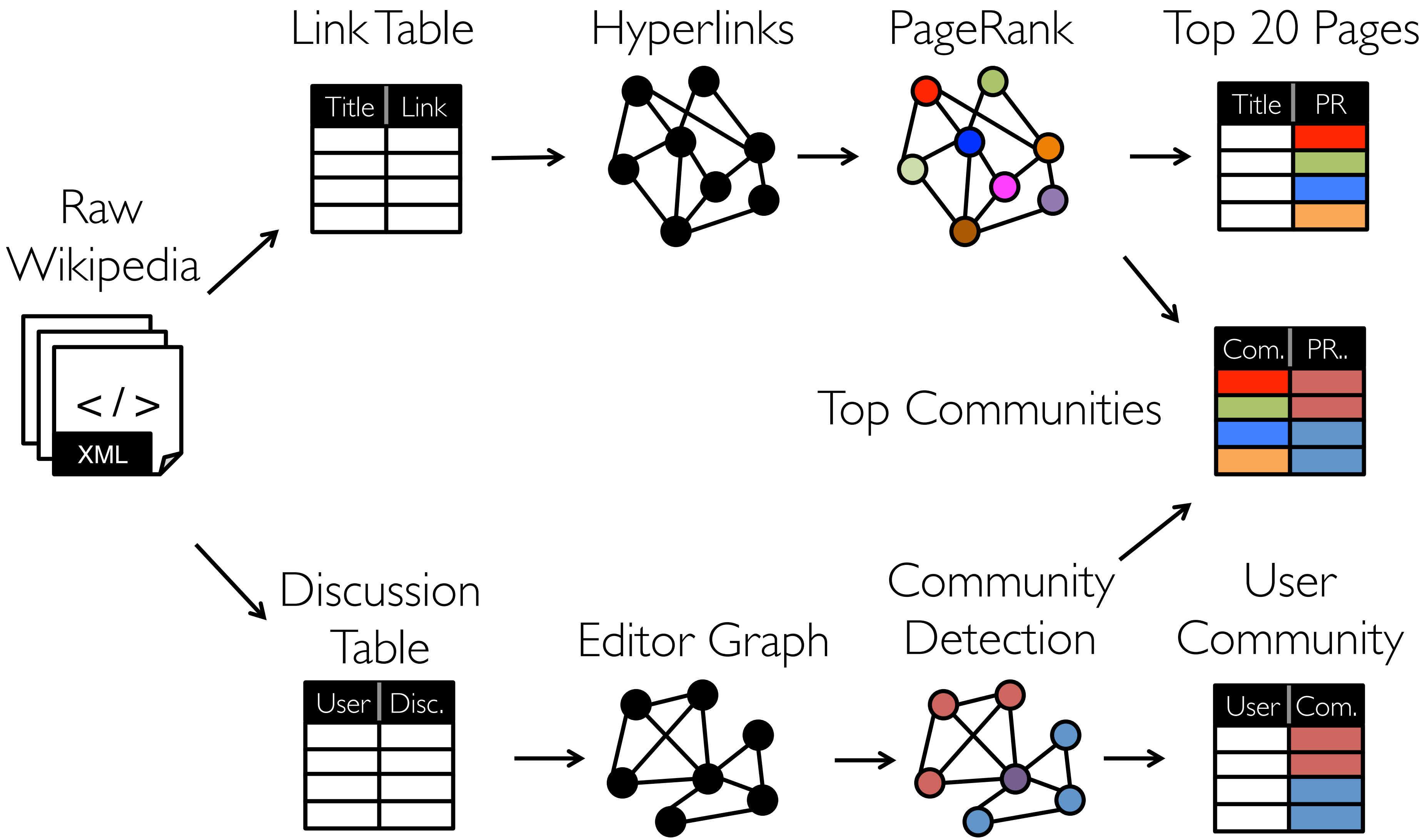
Joseph Gonzalez
Postdoc, UC-Berkeley AMPLab
Co-founder, GraphLab Inc.

Joint work with Reynold Xin, Ankur Dave, Daniel Crankshaw, Michael Franklin, and Ion Stoica

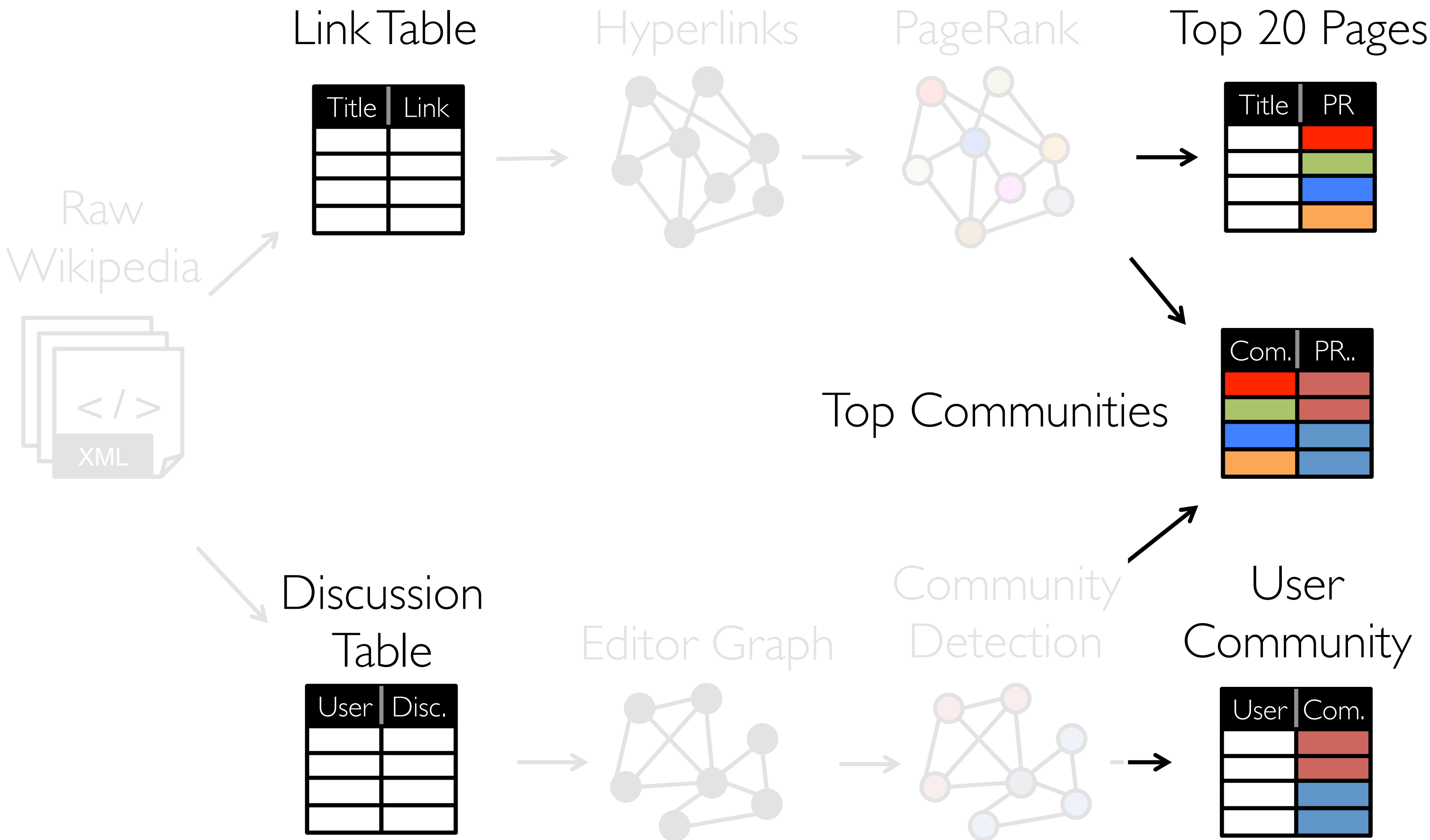
amplab
UC BERKELEY

OSDI 2014

Modern Analytics

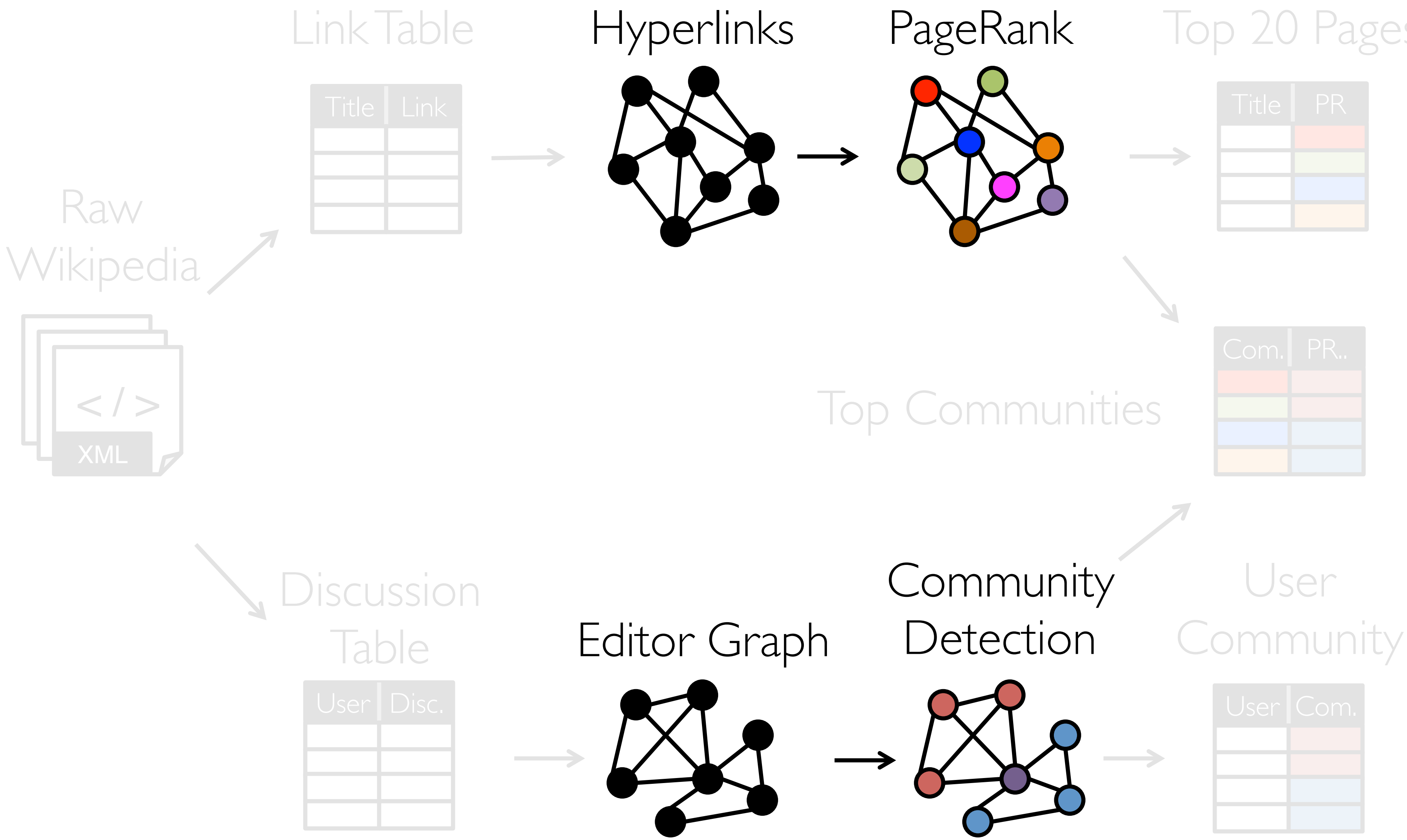


Tables



Link Table
Title
Link
Hyperlinks
PageRank
Top 20 Pages
Title
PR
Raw Wikipedia
< / >
XML
Com.
PR..
Top Communities
Discussion Table
User
Disc.
Editor Graph
Community Detection
User Community
User
Com.

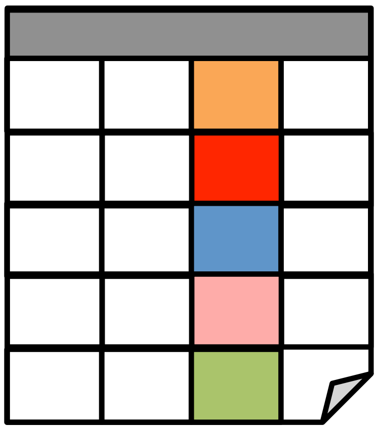
Graphs



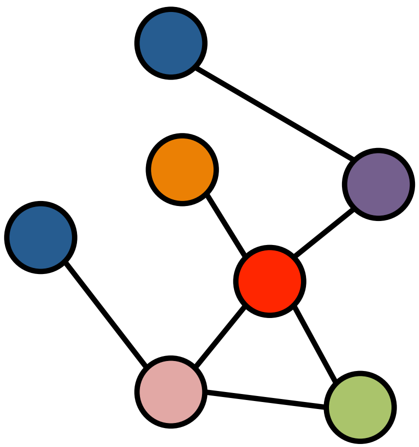
Link Table
Hyperlinks
PageRank
Top 20 Pages
Title
Link
Title
PR
Raw
Wikipedia
< / >
XML
Com.
PR..
Top Communities
Discussion
Table
Editor Graph
Community
Detection
User
Community
User
Disc.
User
Com.

Separate Systems

Tables



Graphs

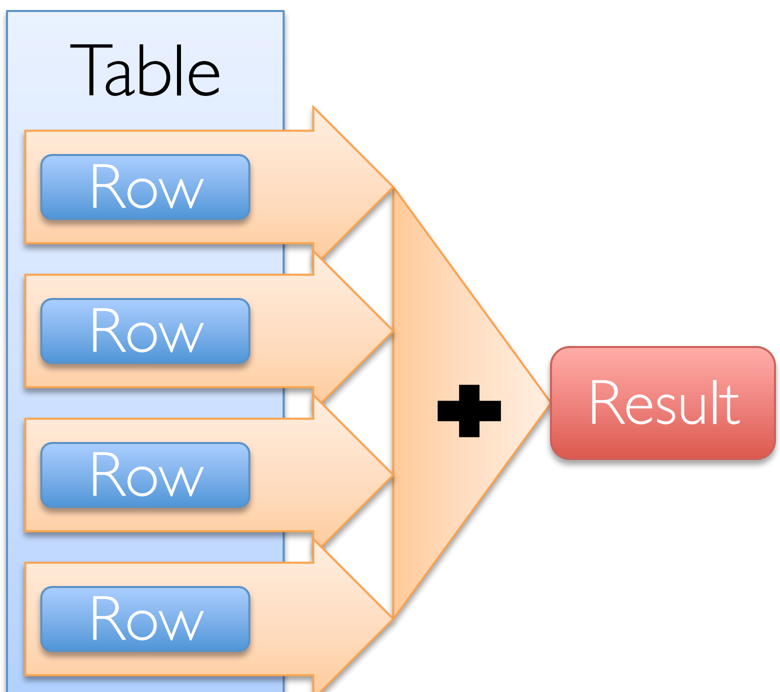


Separate Systems

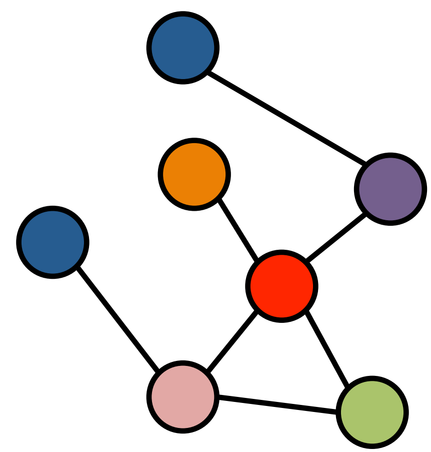
Dataflow Systems

hadoop

Spark



Graphs

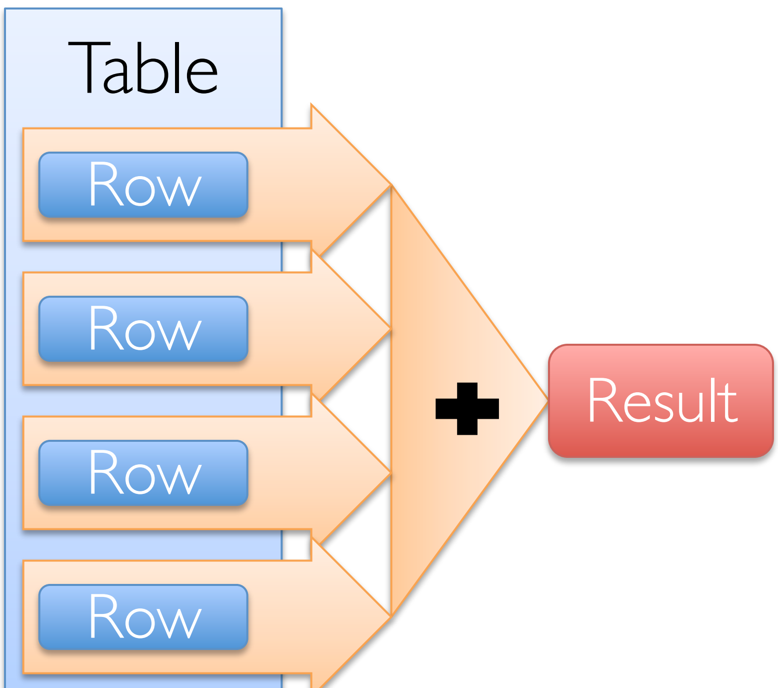


Separate Systems

Dataflow Systems

hadoop

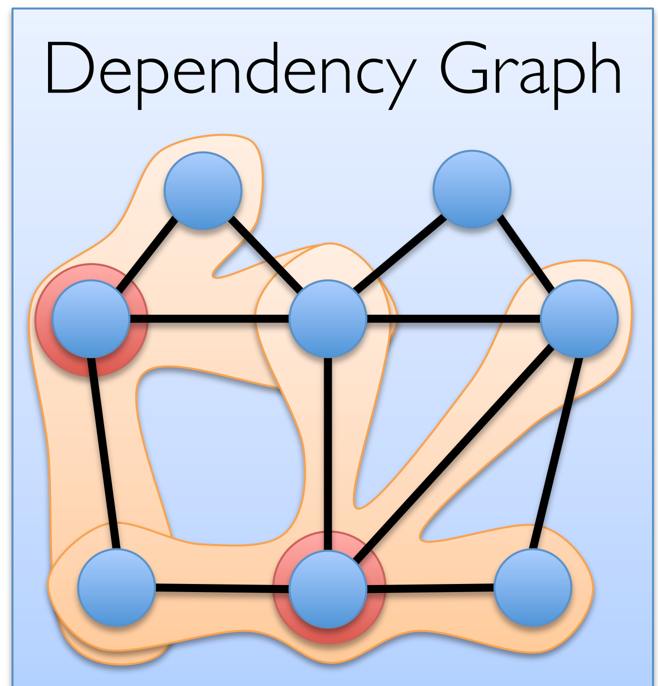
Spark



Graph Systems

Pregel GraphLab

APACHE GIRAPH



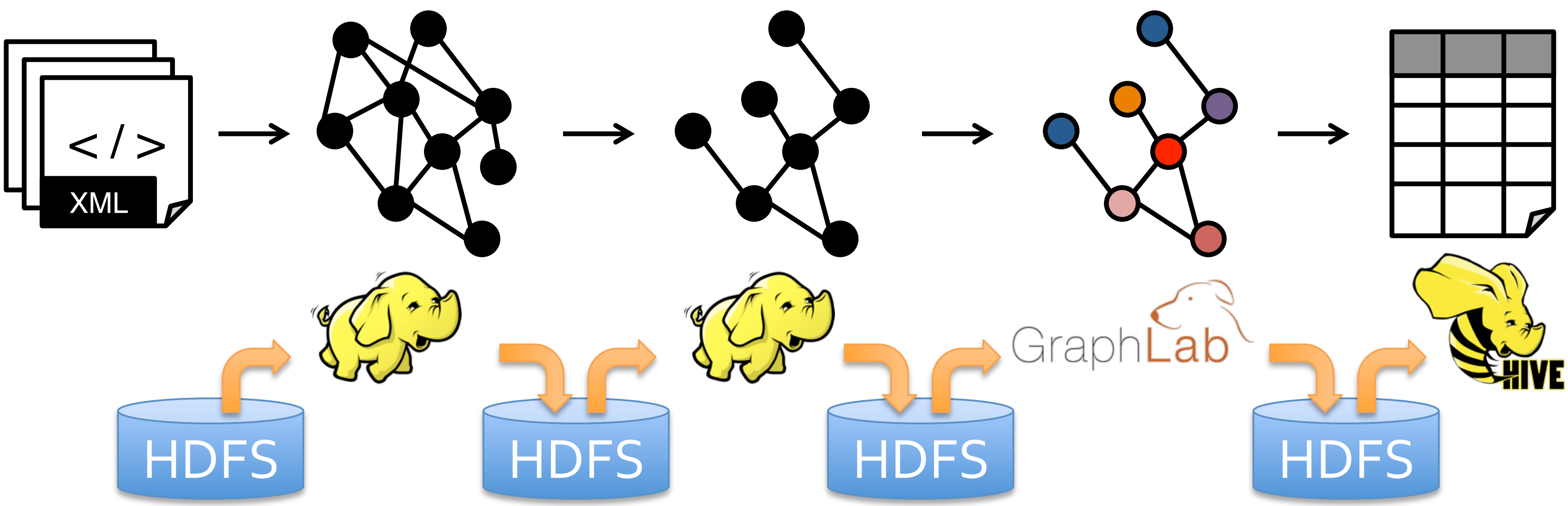
Difficult to Use

Users must *Learn*, *Deploy*, and *Manage* multiple systems

Leads to brittle and often complex interfaces

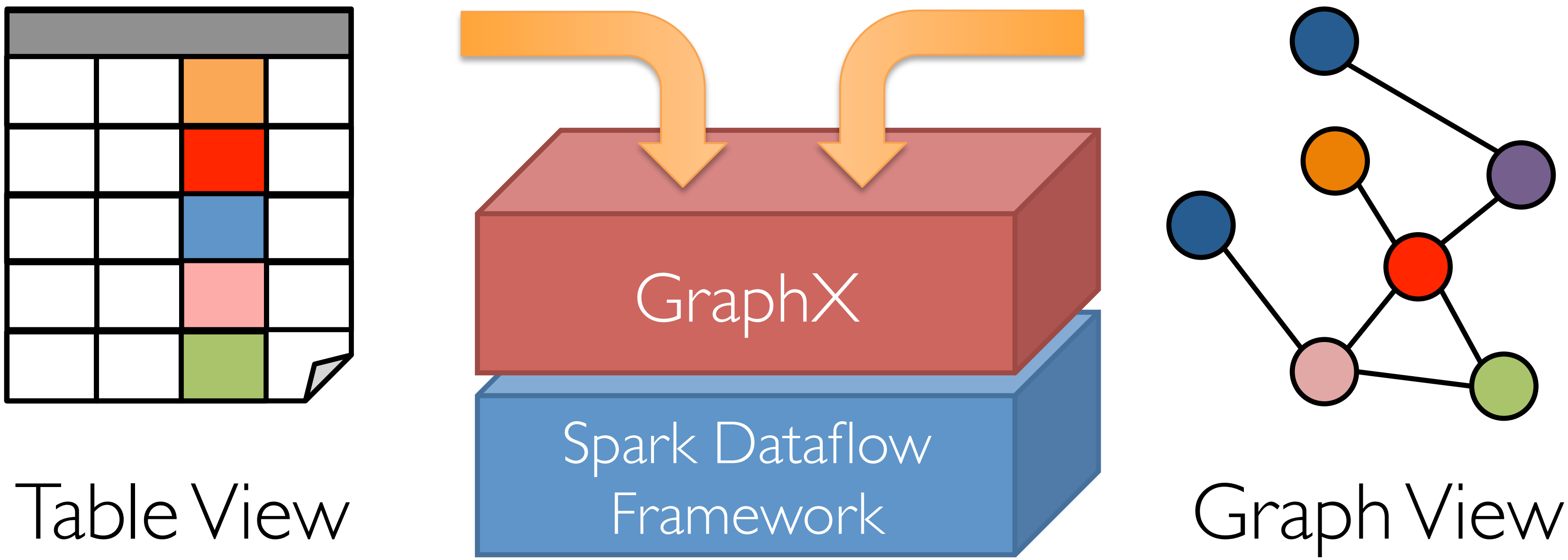
Inefficient

Extensive data movement and duplication across the network and file system



Limited reuse internal data-structures across stages

GraphX Unifies Computation on Tables and Graphs



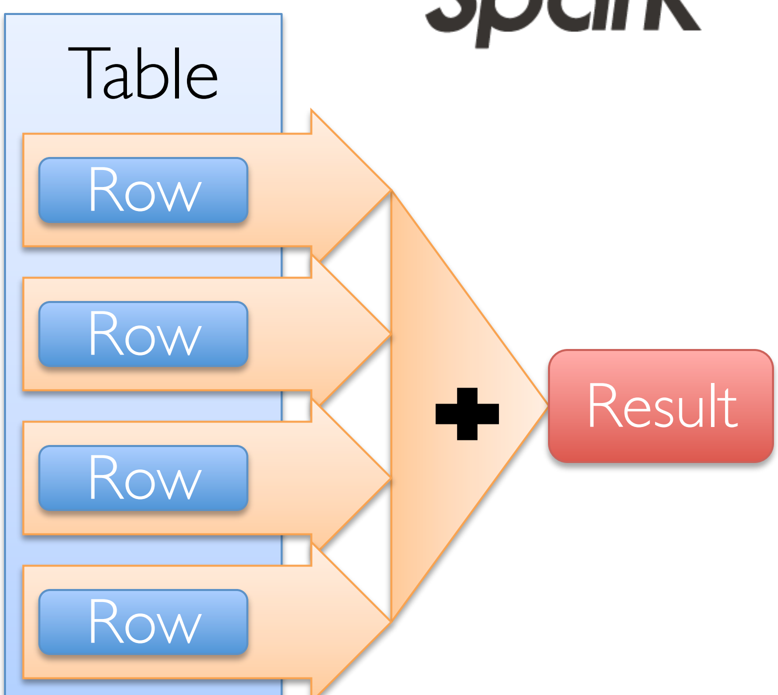
Enabling a single system to *easily* and *efficiently* support the entire pipeline

Separate Systems

Dataflow Systems

hadoop

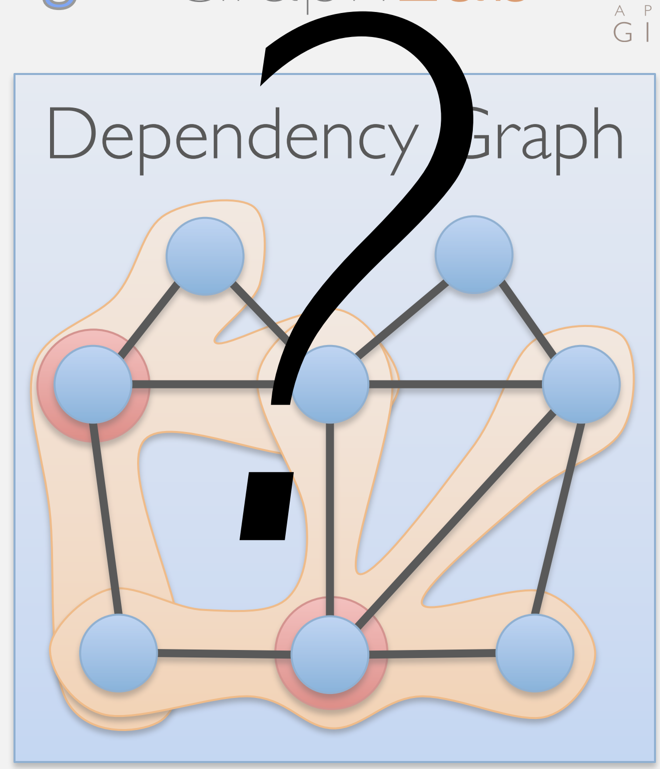
Spark



Graph Systems

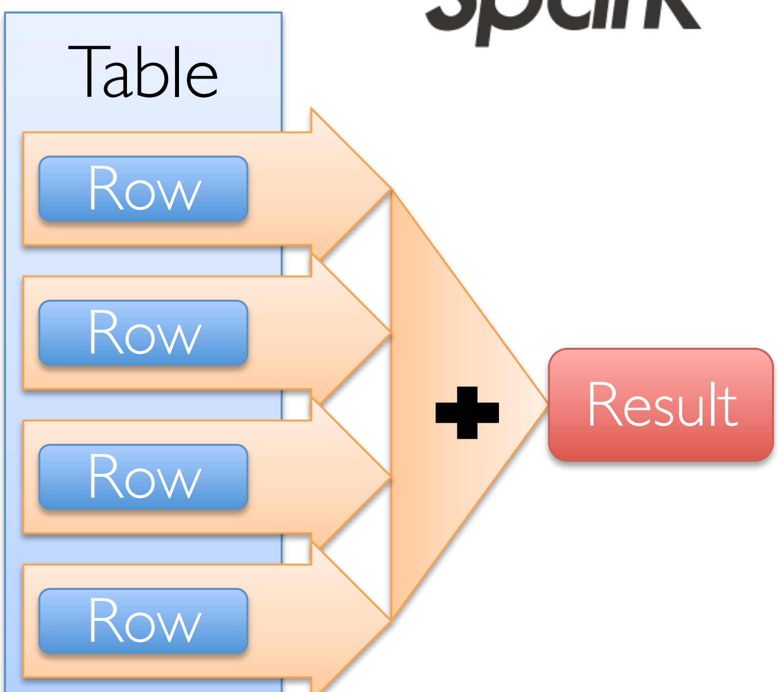
Pregel GraphLab

APACHE GIRAPH

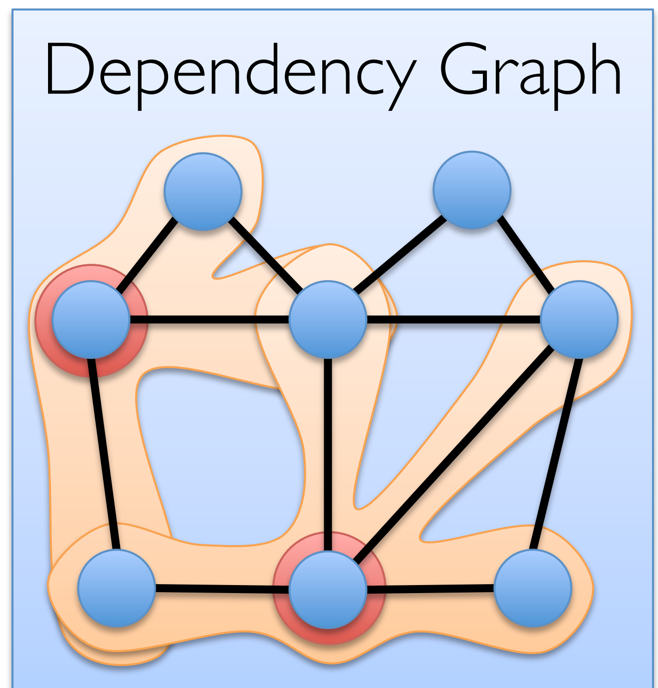


Separate Systems

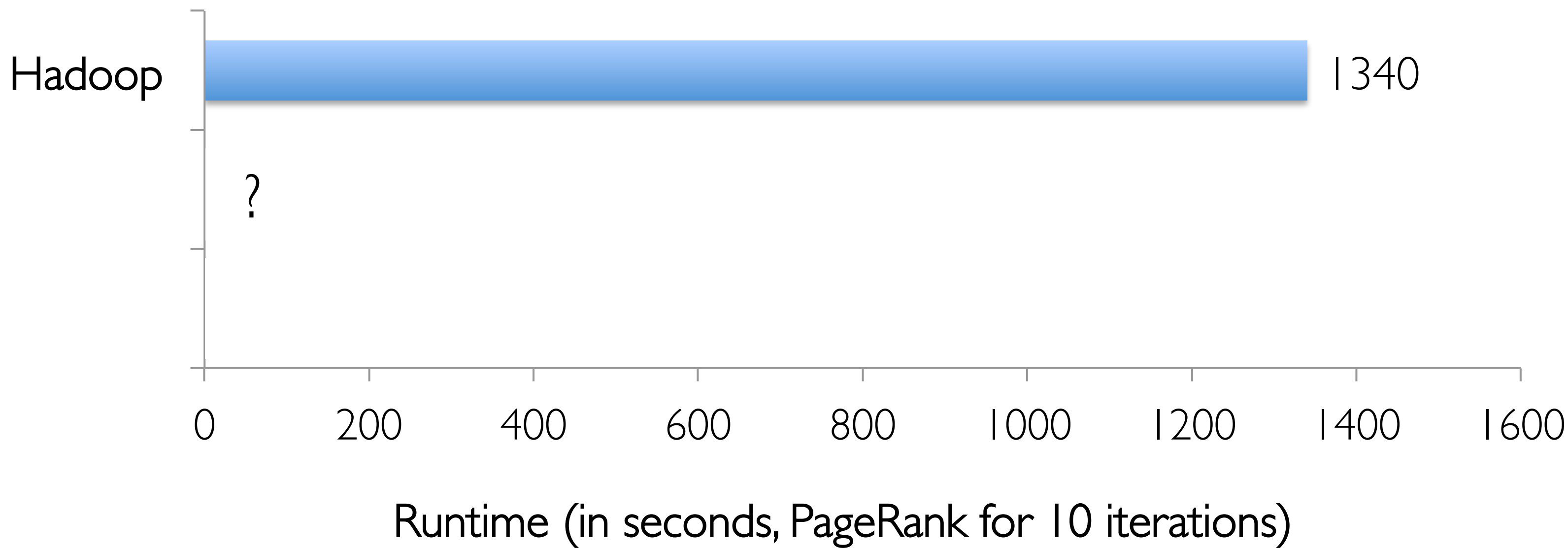
Dataflow Systems



Graph Systems



PageRank on the Live-Journal Graph



Hadoop is *60x slower* than GraphLab

Spark is *16x slower* than GraphLab

Key Question

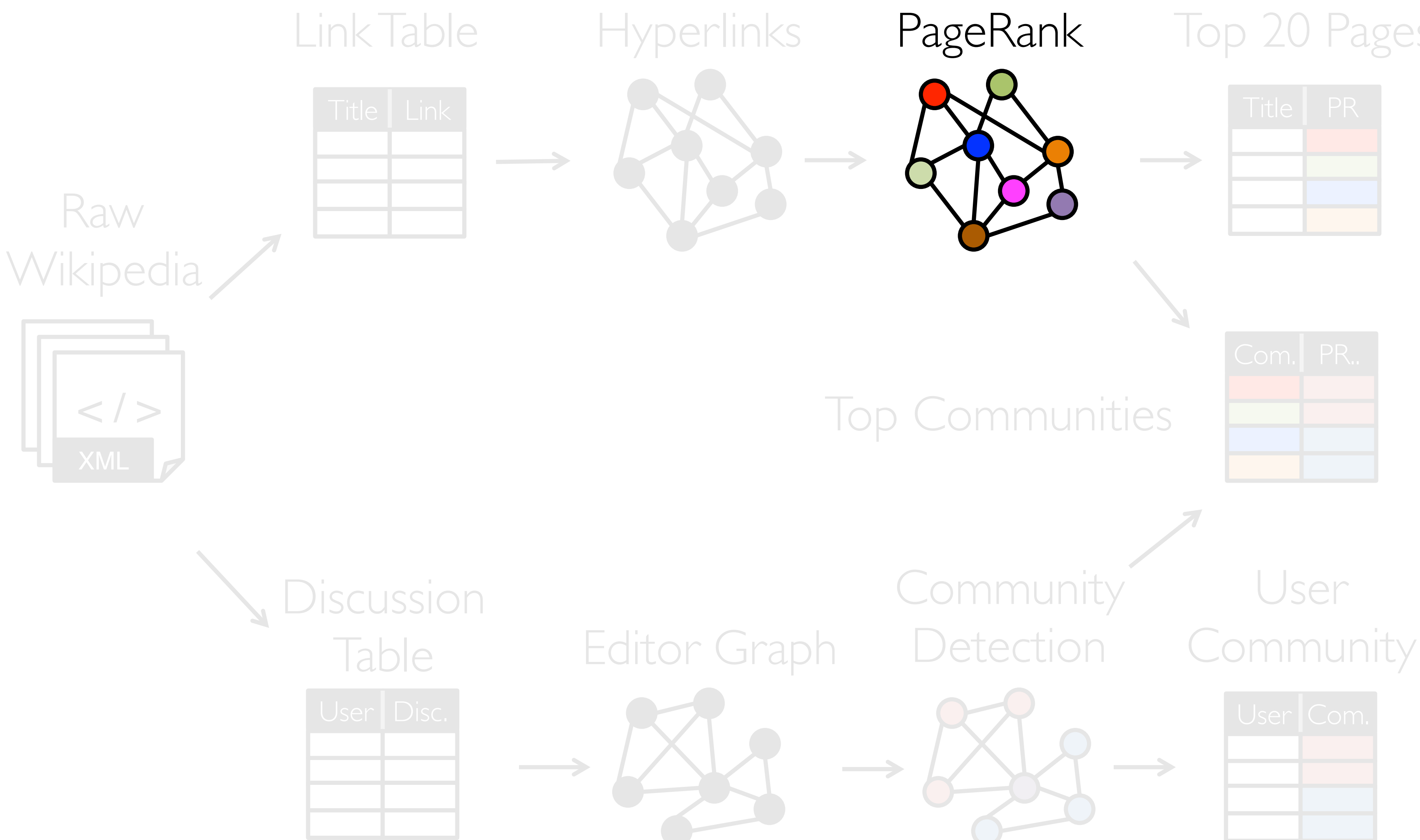
How can we *naturally express* and *efficiently execute* graph computation in a general purpose dataflow framework?

Distill the lessons learned from specialized graph systems

Key Question

How can we *naturally express* and *efficiently execute* graph computation in a general purpose dataflow framework?

Representation   Optimizations


Raw Wikipedia
< / >
XML
Link Table
Title
Link
Hyperlinks
PageRank
Top 20 Pages
Title
PR
Com.
PR..
Top Communities
Discussion Table
User
Disc.
Editor Graph
Community Detection
User Community
User
Com.

Example Computation: PageRank

Express computation *locally*:

$$R[i] = 0.15 + \sum_{j \in \mathrm{InLinks}(i)} \frac{R[j]}{\mathrm{OutLinks}(j)}$$

Rank of Page *i*

Random *Reset Prob.*

Weighted sum of neighbors' ranks

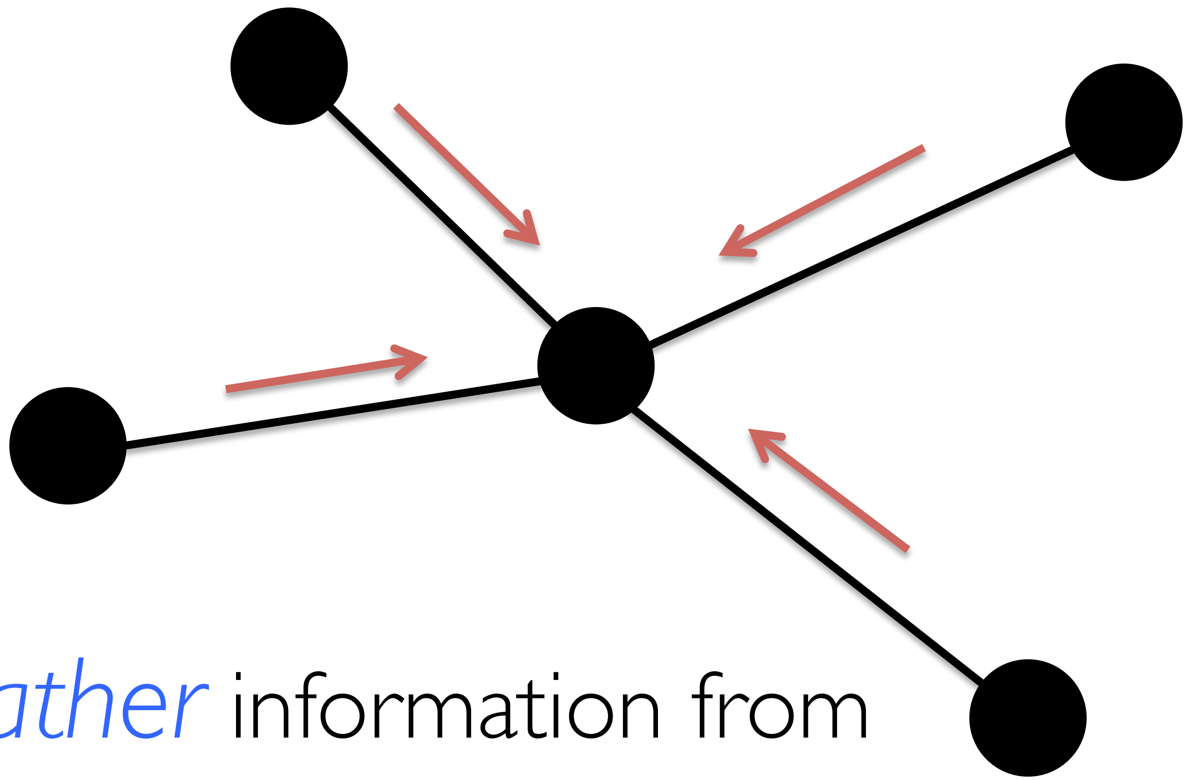
Iterate until convergence

“Think like a Vertex.”

- Malewicz et al., SIGMOD’10

Graph-Parallel Pattern

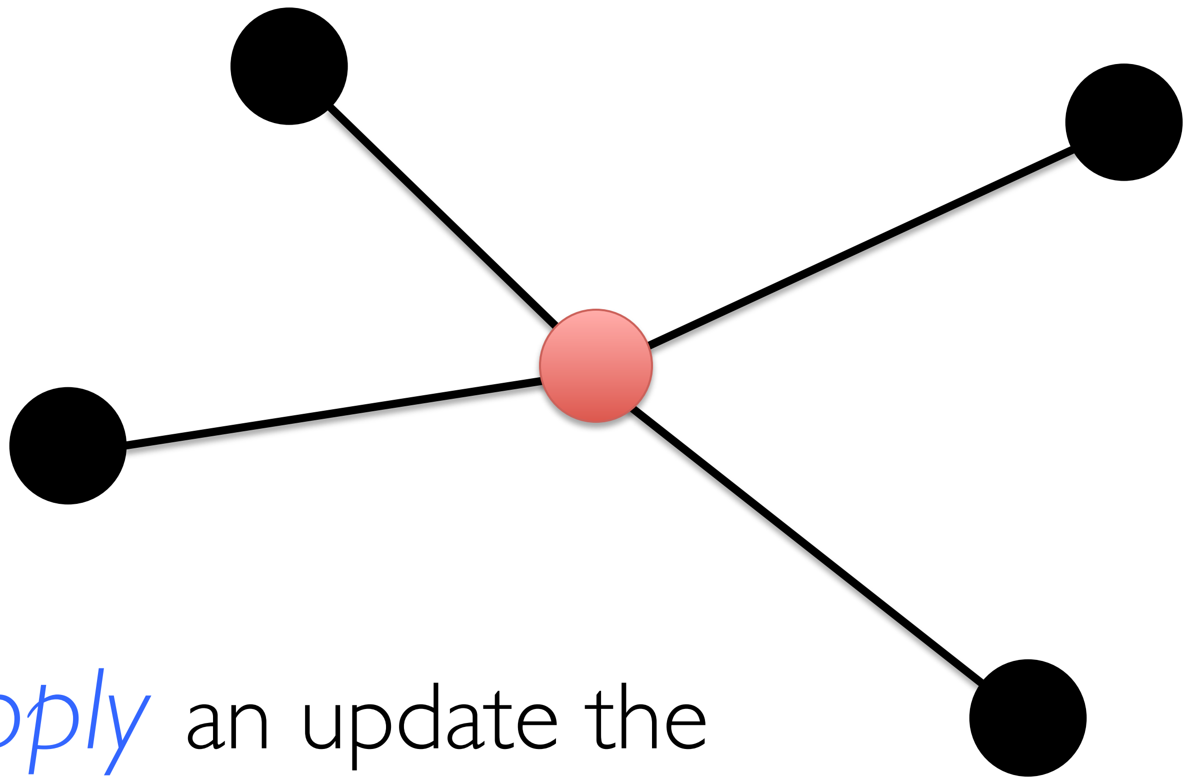
Gonzalez et al. [OSDI'12]



Gather information from neighboring vertices

Graph-Parallel Pattern

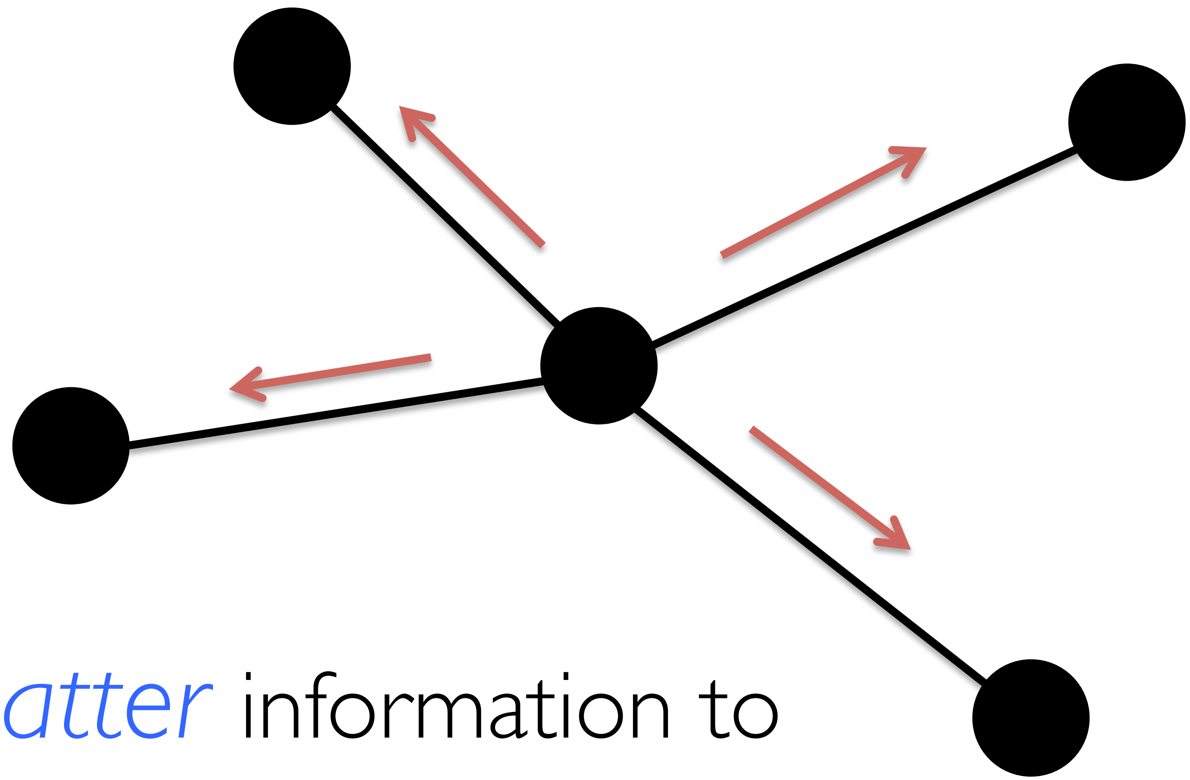
Gonzalez et al. [OSDI'12]



Apply an update the vertex property

Graph-Parallel Pattern

Gonzalez et al. [OSDI'12]



Scatter information to neighboring vertices

Many Graph-Parallel Algorithms

Collaborative Filtering
- » Alternating Least Squares
- » Stochastic Gradient Descent
- » Tensor Factorization

Structured Prediction
- » Loopy Belief Propagation
- » Max-Product Linear Programs
- » Gibbs Sampling

Semi-supervised ML
- » Graph SSL
- » CoEM

MACHINE LEARNING

Community Detection
- » Triangle-Counting
- » K-core Decomposition
- » K-Truss

Graph Analytics
- » PageRank
- » Personalized PageRank
- » Shortest Path
- » Graph Coloring

NETWORK ANALYSIS

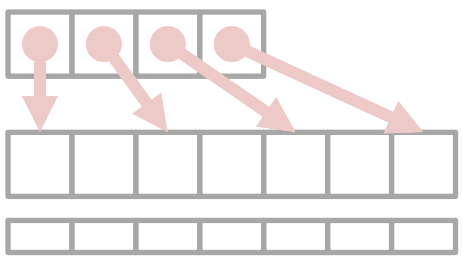
Specialized
Computational
Pattern

→

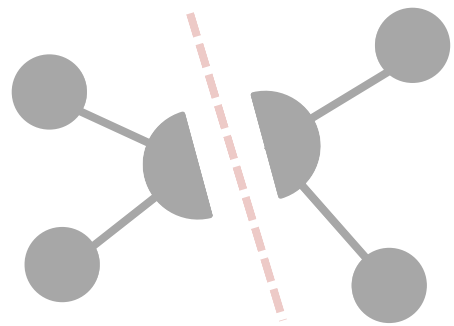
Specialized
Graph
Optimizations

Graph System Optimizations

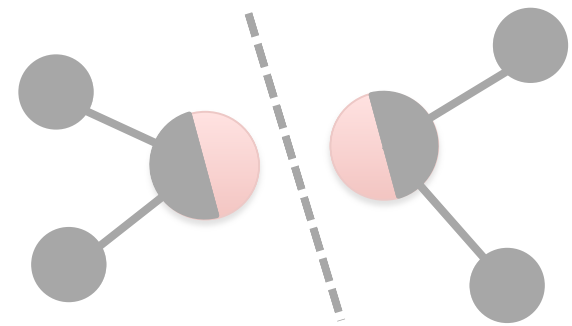
Specialized Data-Structures



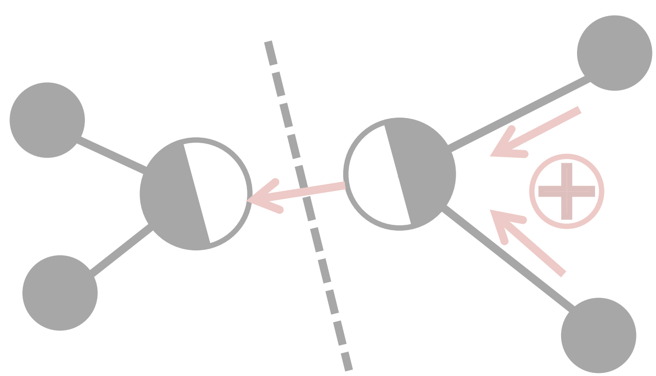
Vertex-Cuts Partitioning



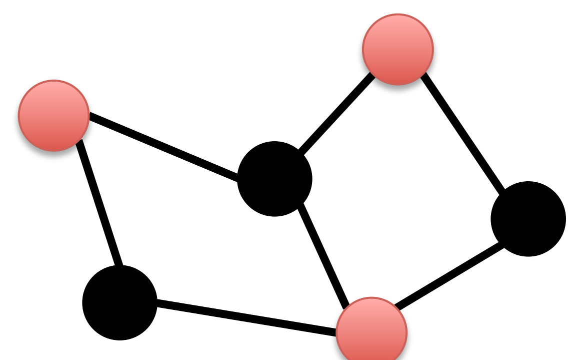
Remote Caching / Mirroring



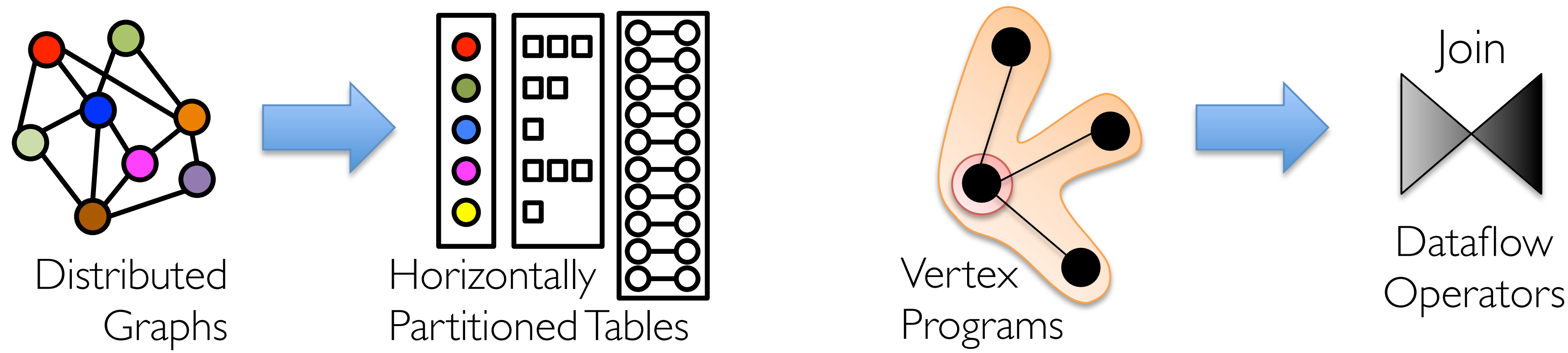
Message Combiners



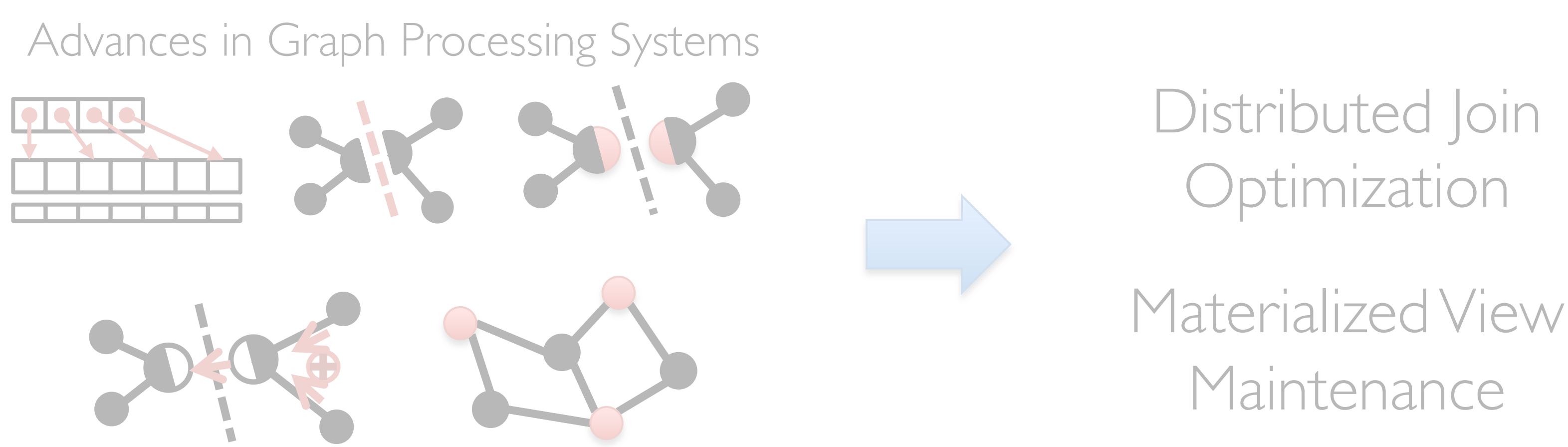
Active Set Tracking



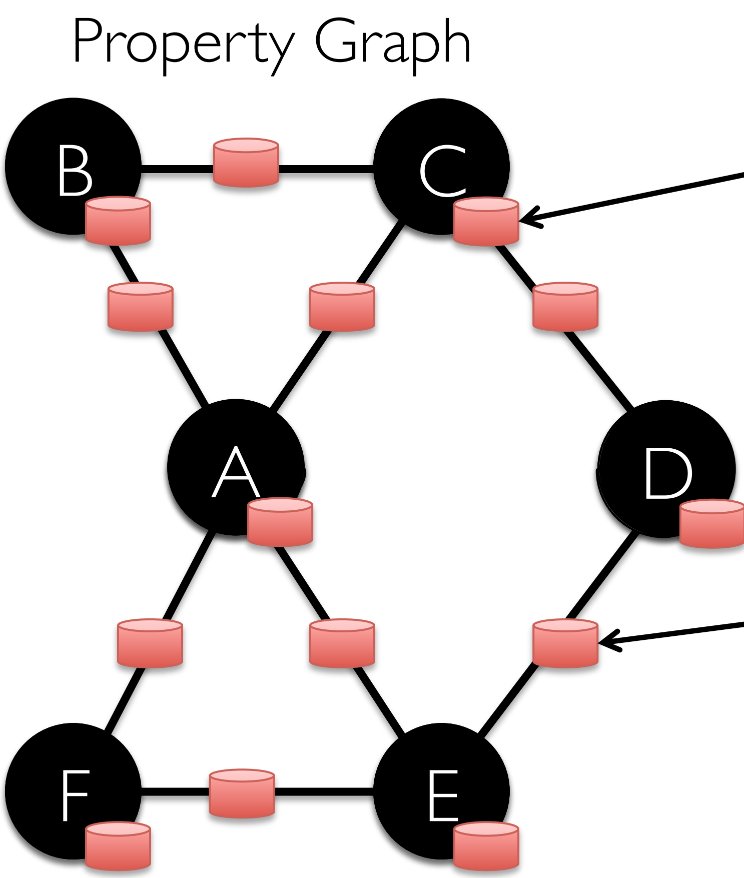
Representation



Optimizations



Property Graph Data Model



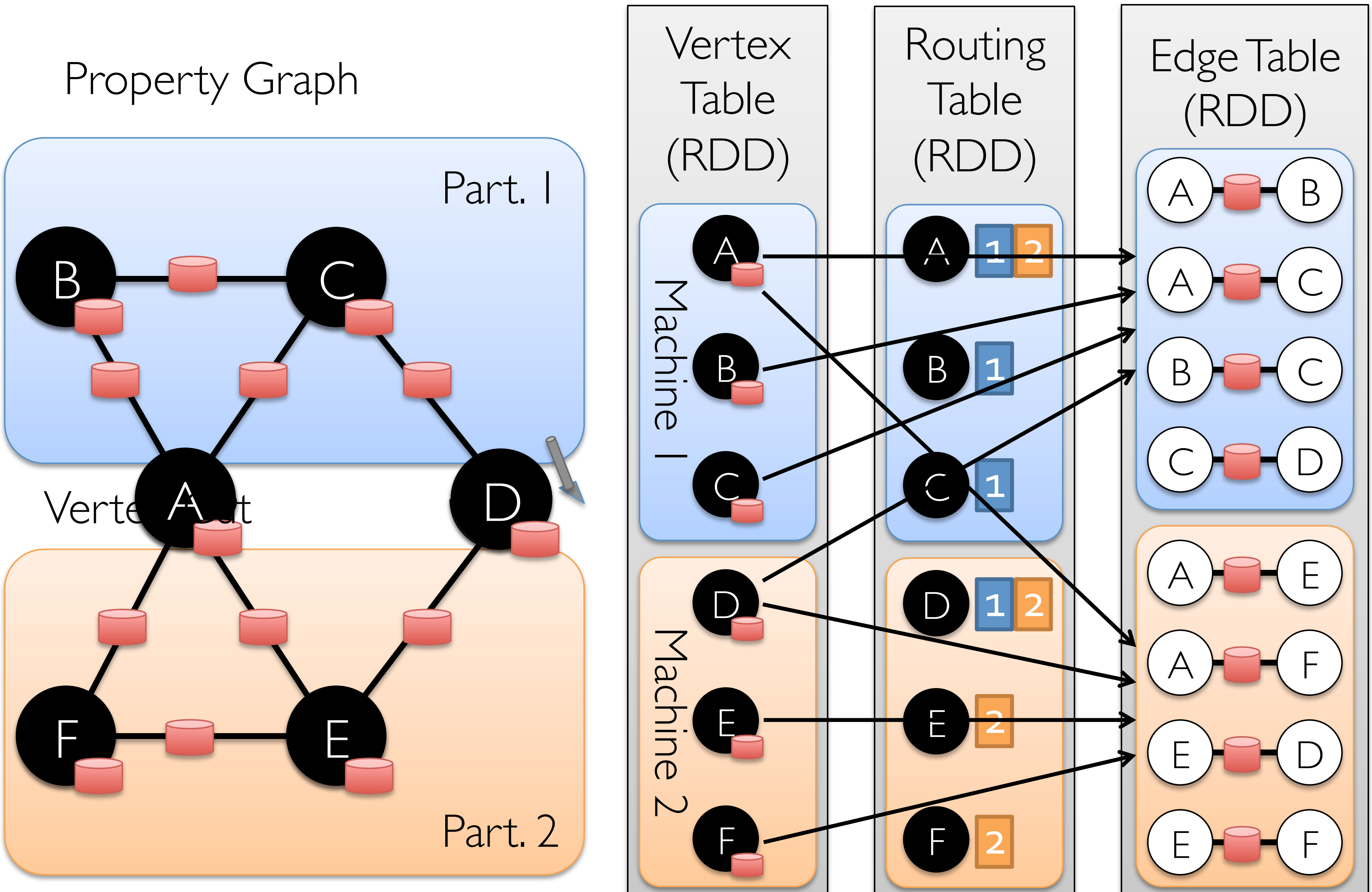
Vertex Property:

- User Profile
- Current PageRank Value

Edge Property:

- Weights
- Timestamps

Encoding Property Graphs as Tables



Separate Properties and Structure

Reuse structural information across multiple graphs

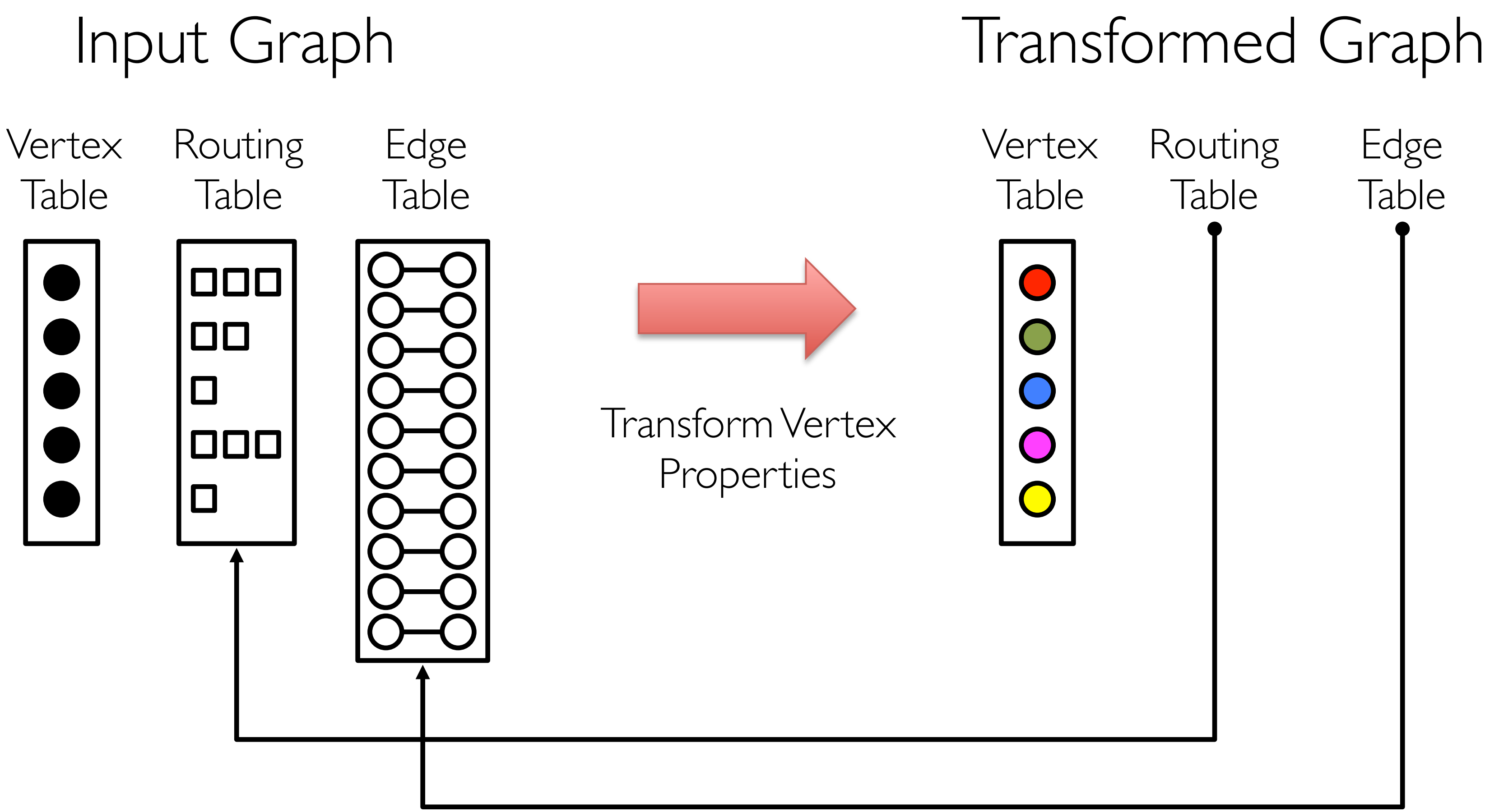


Table Operators

Table operators are inherited from Spark:

`map`	`reduce`	`sample`
`filter`	`count`	`take`
`groupBy`	`fold`	`first`
`sort`	`reduceByKey`	`partitionBy`
`union`	`groupByKey`	`mapWith`
`join`	`cogroup`	`pipe`
`leftOuterJoin`	`cross`	`save`
`rightOuterJoin`	`zip`	`...`

Graph Operators (Scala)

```
class Graph [ V, E ] {
  def Graph(vertices: Table[ (Id, V) ],
            edges: Table[ (Id, Id, E) ])
  // Table Views -----------------
  def vertices: Table[ (Id, V) ]
  def edges: Table[ (Id, Id, E) ]
  def triplets: Table [ ((Id, V), (Id, V), E) ]
  // Transformations ------------------------------
  def reverse: Graph[V, E]
  def subgraph(pV: (Id, V) => Boolean,
               pE: Edge[V,E] => Boolean): Graph[V,E]
  def mapV(m: (Id, V) => T ): Graph[T,E]
  def mapE(m: Edge[V,E] => T ): Graph[V,T]
  // Joins ----------------------------------------
  def joinV(tbl: Table [(Id, T)]): Graph[(V, T), E ]
  def joinE(tbl: Table [(Id, Id, T)]): Graph[V, (E, T)]
  // Computation ----------------------------------
  def mrTriplets(mapF: (Edge[V,E]) => List[(Id, T)],
                 reduceF: (T, T) => T): Graph[T, E]
}
```

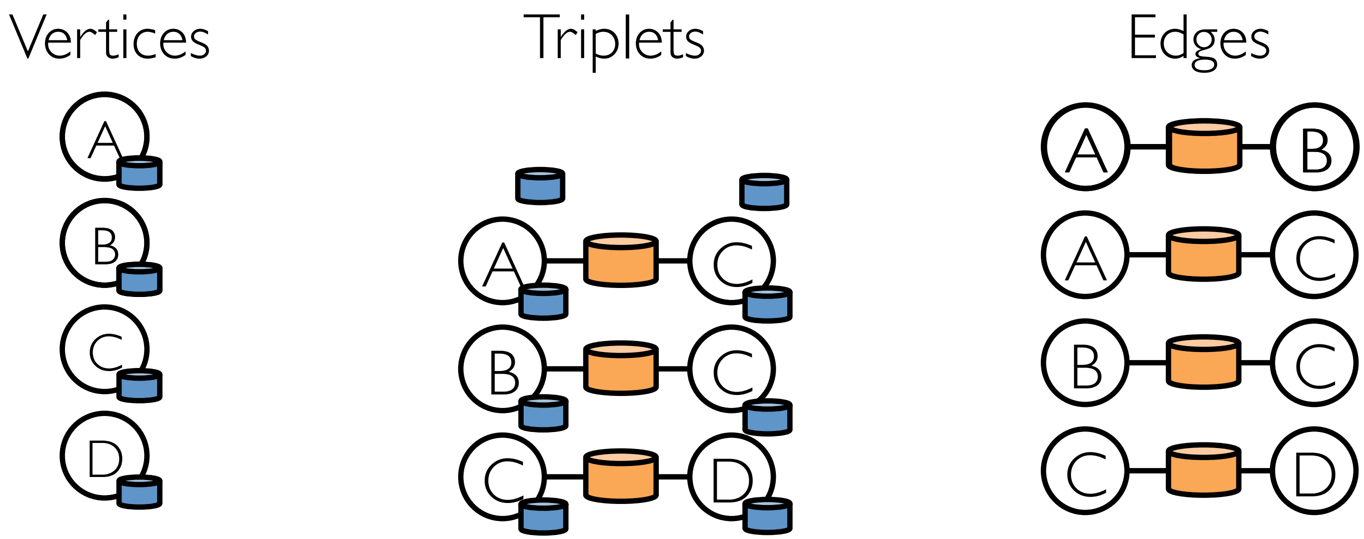
Graph Operators (Scala)

```
class Graph [ V, E ] {
  def Graph(vertices: Table[ (Id, V) ],
            edges: Table[ (Id, Id, E) ])
  // Table Views -----------------
  def vertices: Table[ (Id, V) ]
  def edges: Table[ (Id, Id, E) ]
  def triplets: Table [ ((Id, V), (Id, V), E) ]
  // Transformations ------------------------------
  def reverse: Graph[V, E]
  def subgraph(pV: (Id, V) => Boolean,
               pE: Edge[V,E] => Boolean): Graph[V,E]
  def mapV(m: (Id, V) => T ): Graph[T,E]
  def mapE(m: Edge[V,E] => T ): Graph[V,T]
  // Joins ----------------------------------------
  def joinV(tbl: Table [(Id, T)]): Graph[(V, T), E]
  def joinE(tbl: Table [(Id, Id, T)]): Graph[V, (E, T)]
  // Computation ----------------------------------
  def mrTriplets(mapF: (Edge[V,E]) => List[(Id, T)],
                 reduceF: (T, T) => T): Graph[T, E]
}
```

capture the *Gather-Scatter pattern* from specialized graph-processing systems

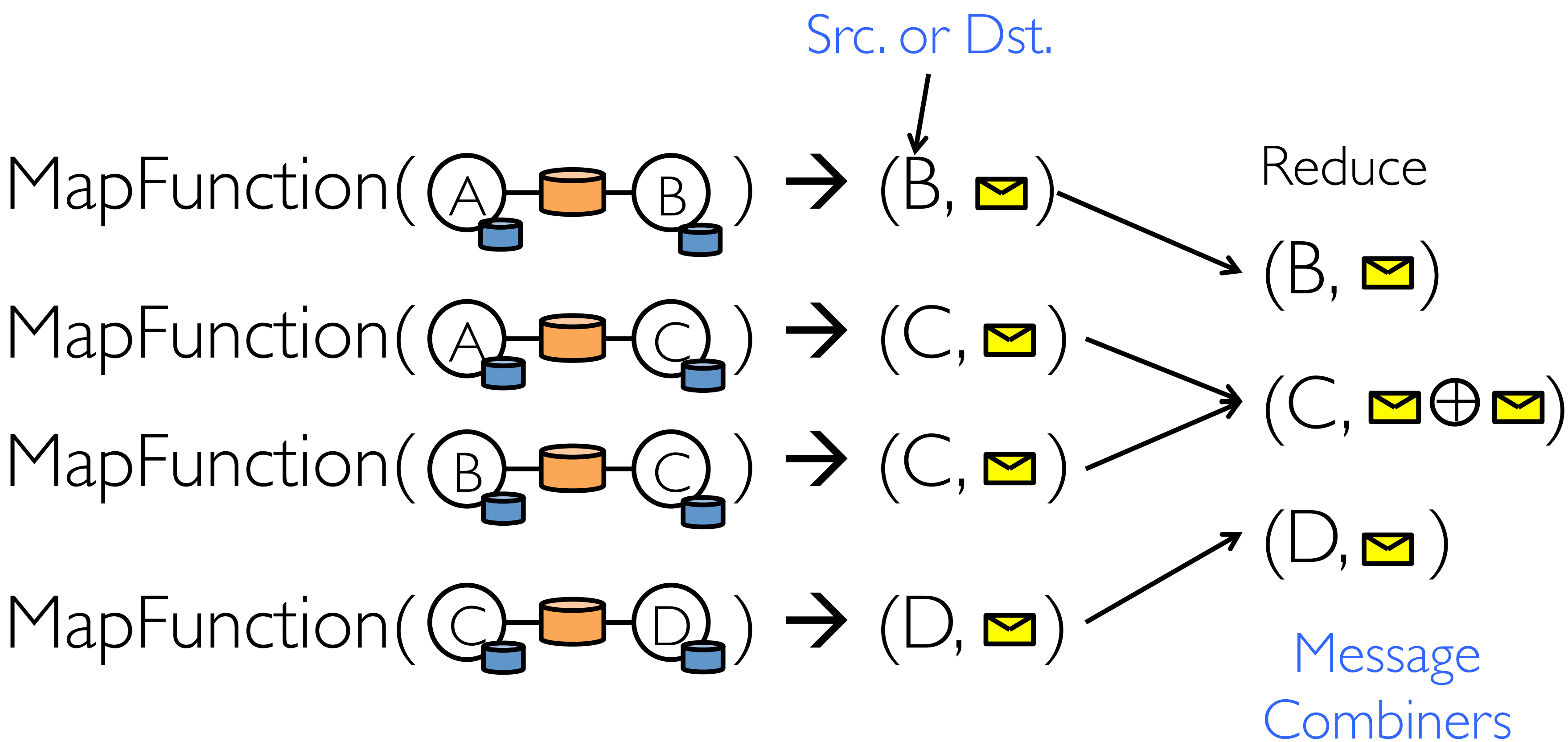
Triplets Join Vertices and Edges

The *triplets* operator joins vertices and edges:



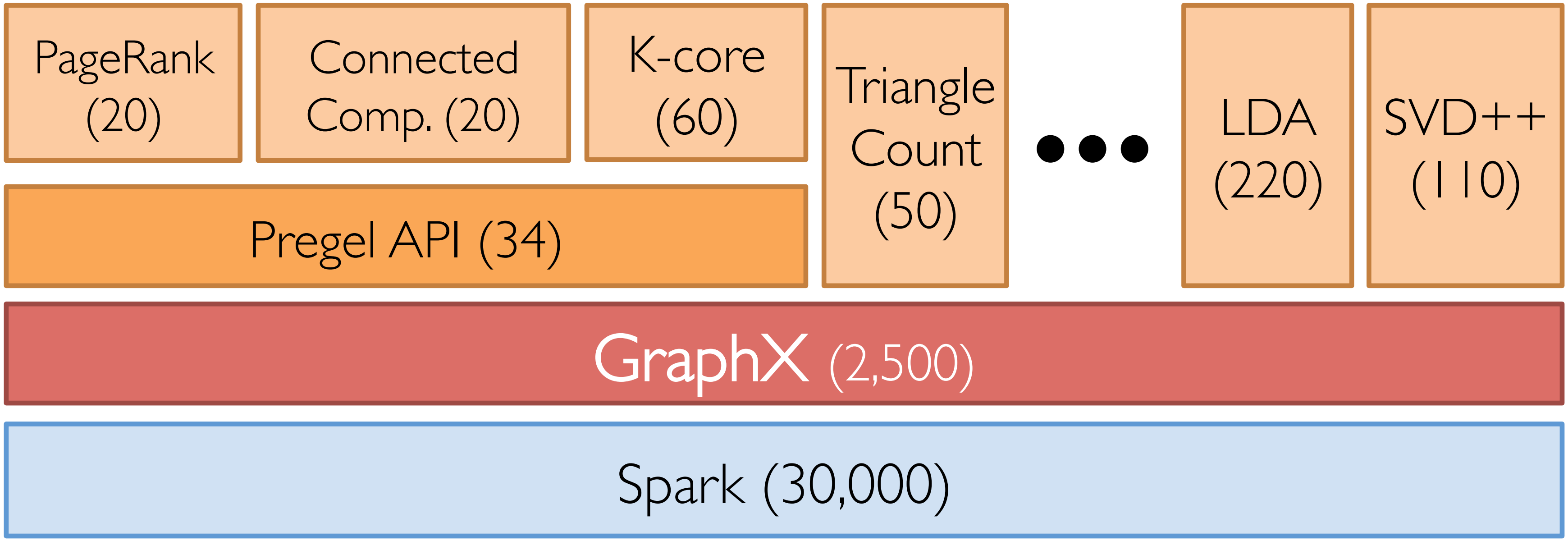
Map-Reduce Triplets

Map-Reduce triplets collects information about the neighborhood of each vertex:



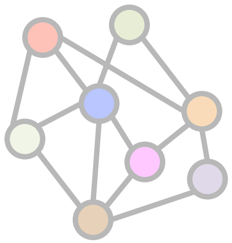
Using these basic GraphX operators we implemented Pregel and GraphLab in under 50 lines of code!

The GraphX Stack (Lines of Code)

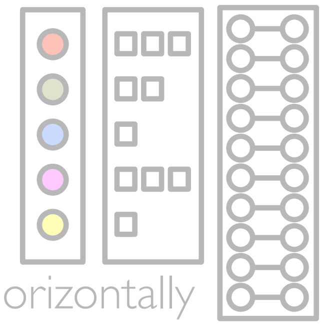


Some algorithms are more naturally expressed using the GraphX primitive operators

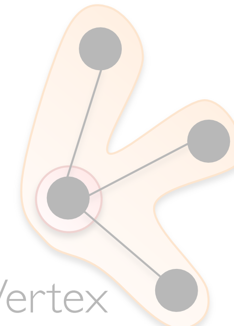
Representation



Distributed
Graphs

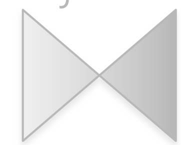


Horizontally
Partitioned Tables



Vertex
Programs

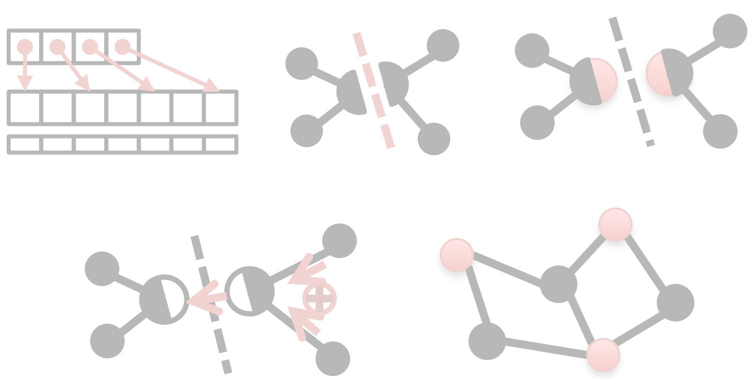
Join



Dataflow
Operators

Optimizations

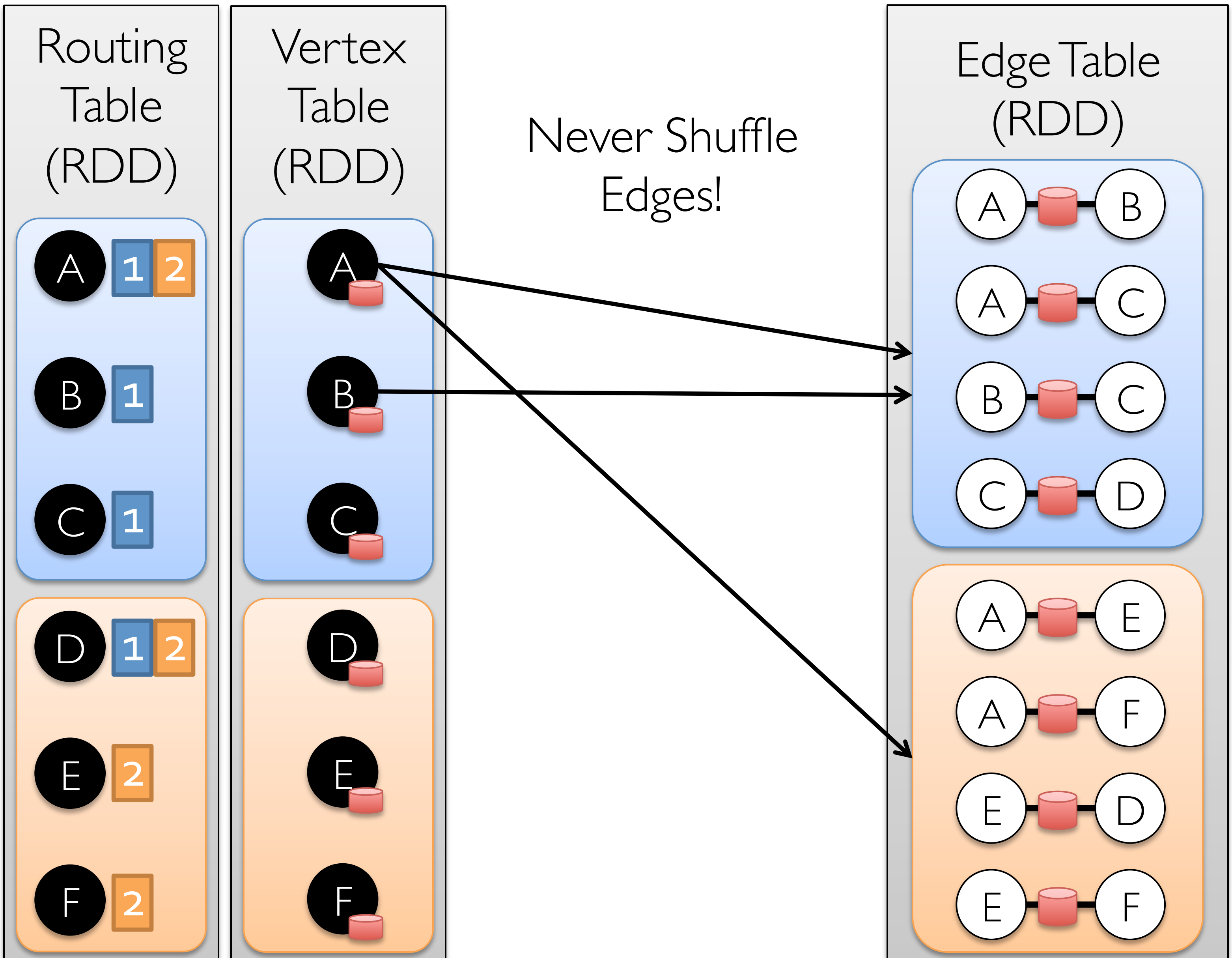
Advances in Graph Processing Systems



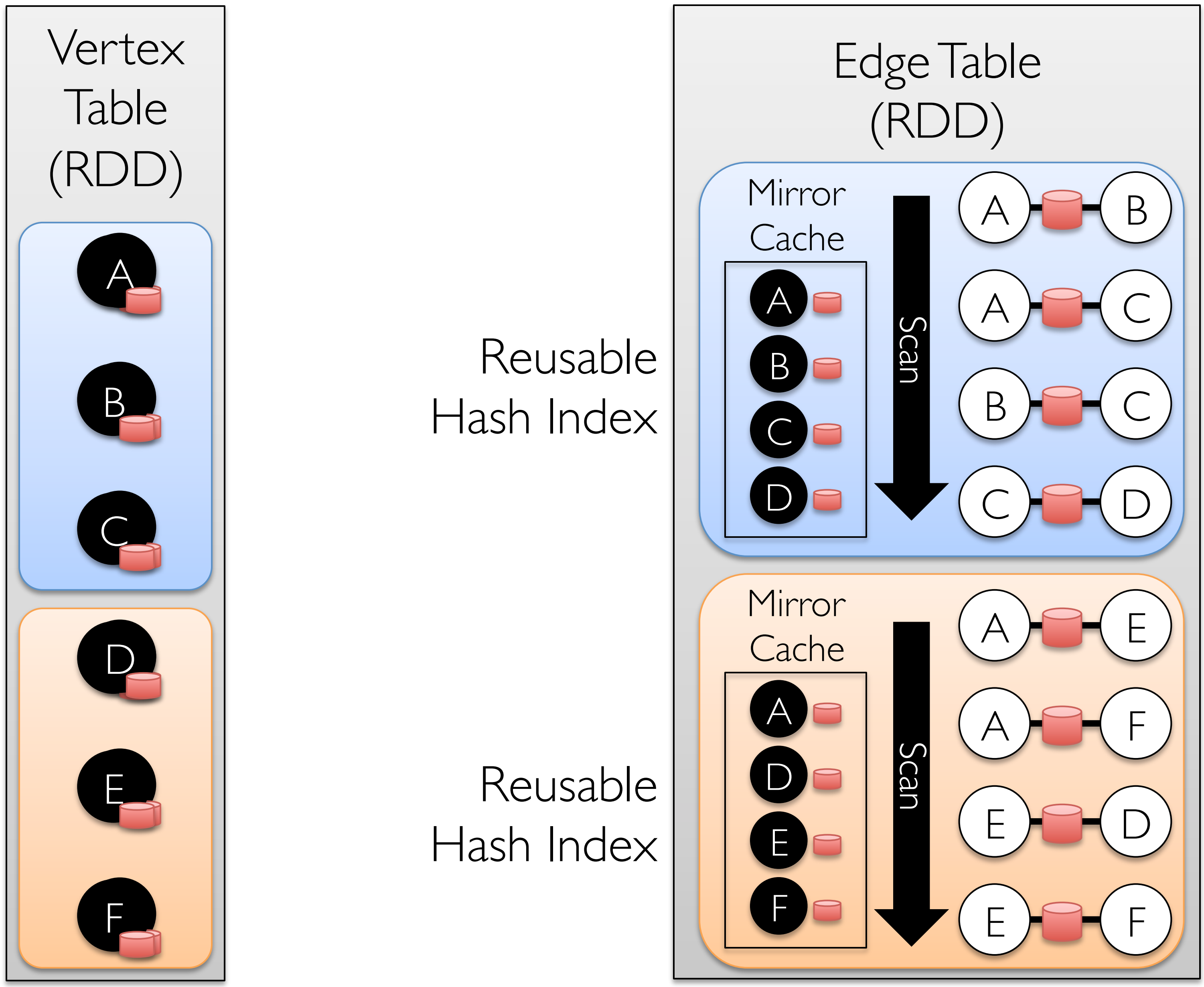
Distributed Join
Optimization

Materialized View
Maintenance

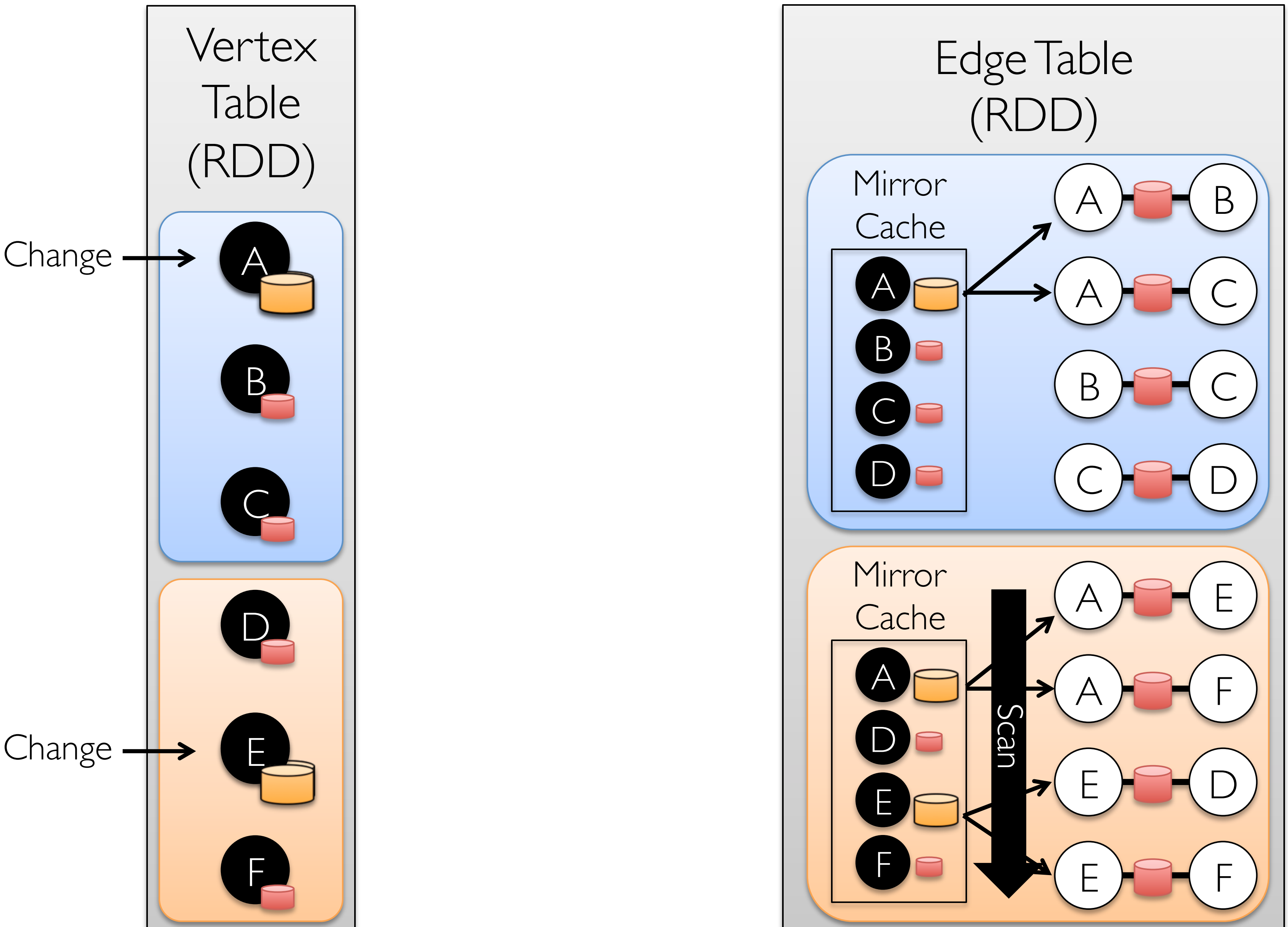
Join Site Selection using Routing Tables



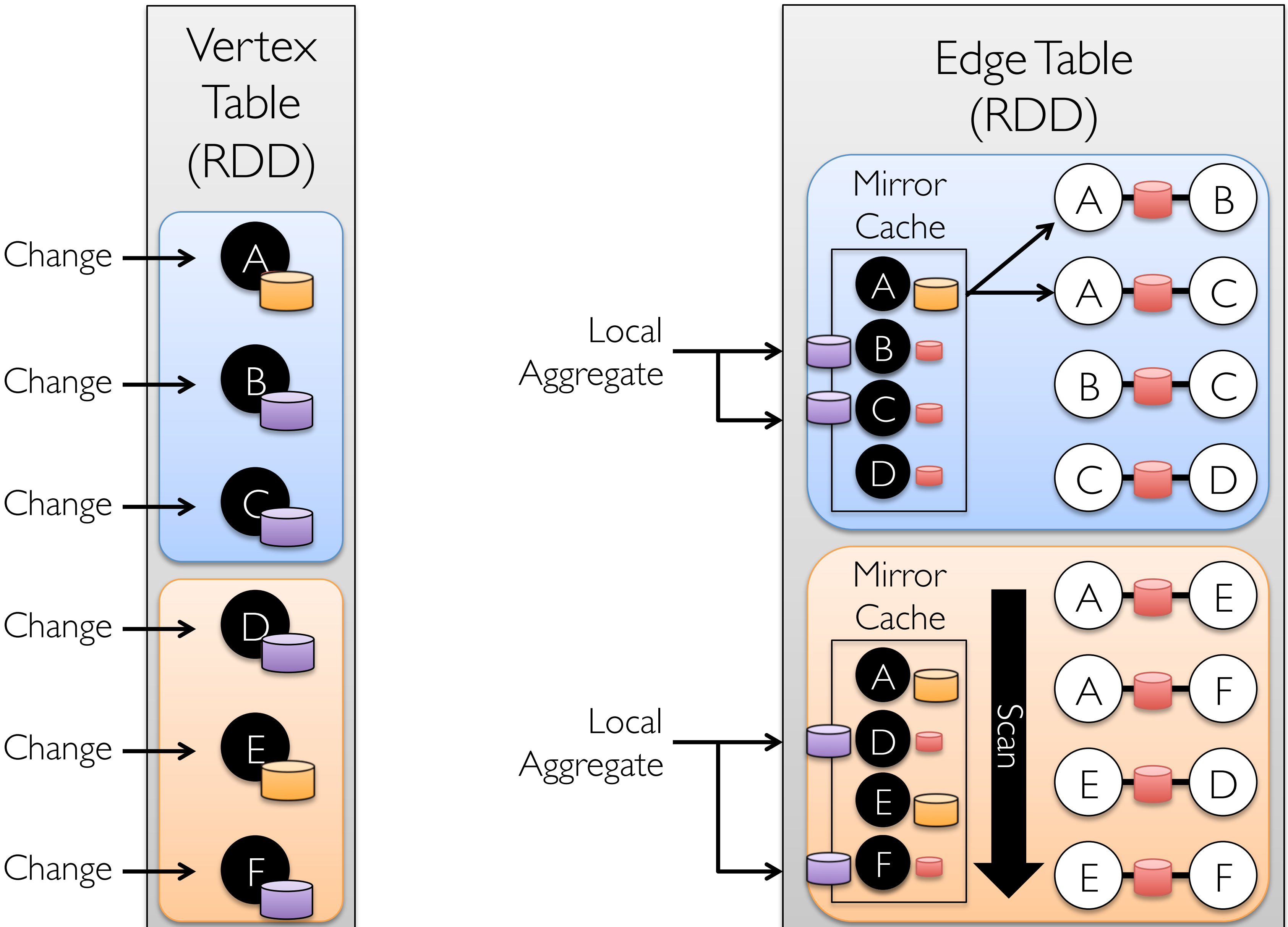
Caching for Iterative mrTriplets



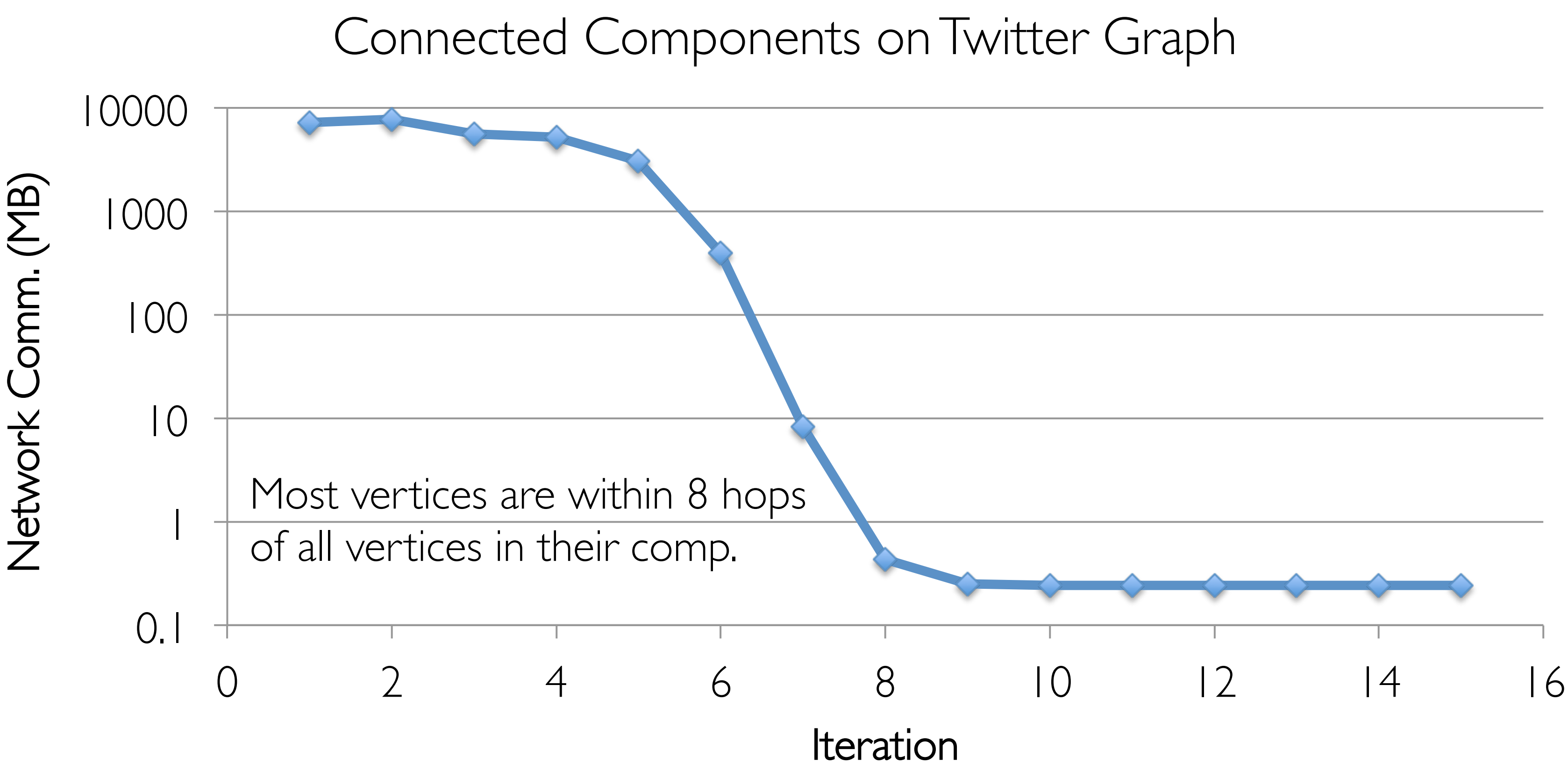
Incremental Updates for Triplets View



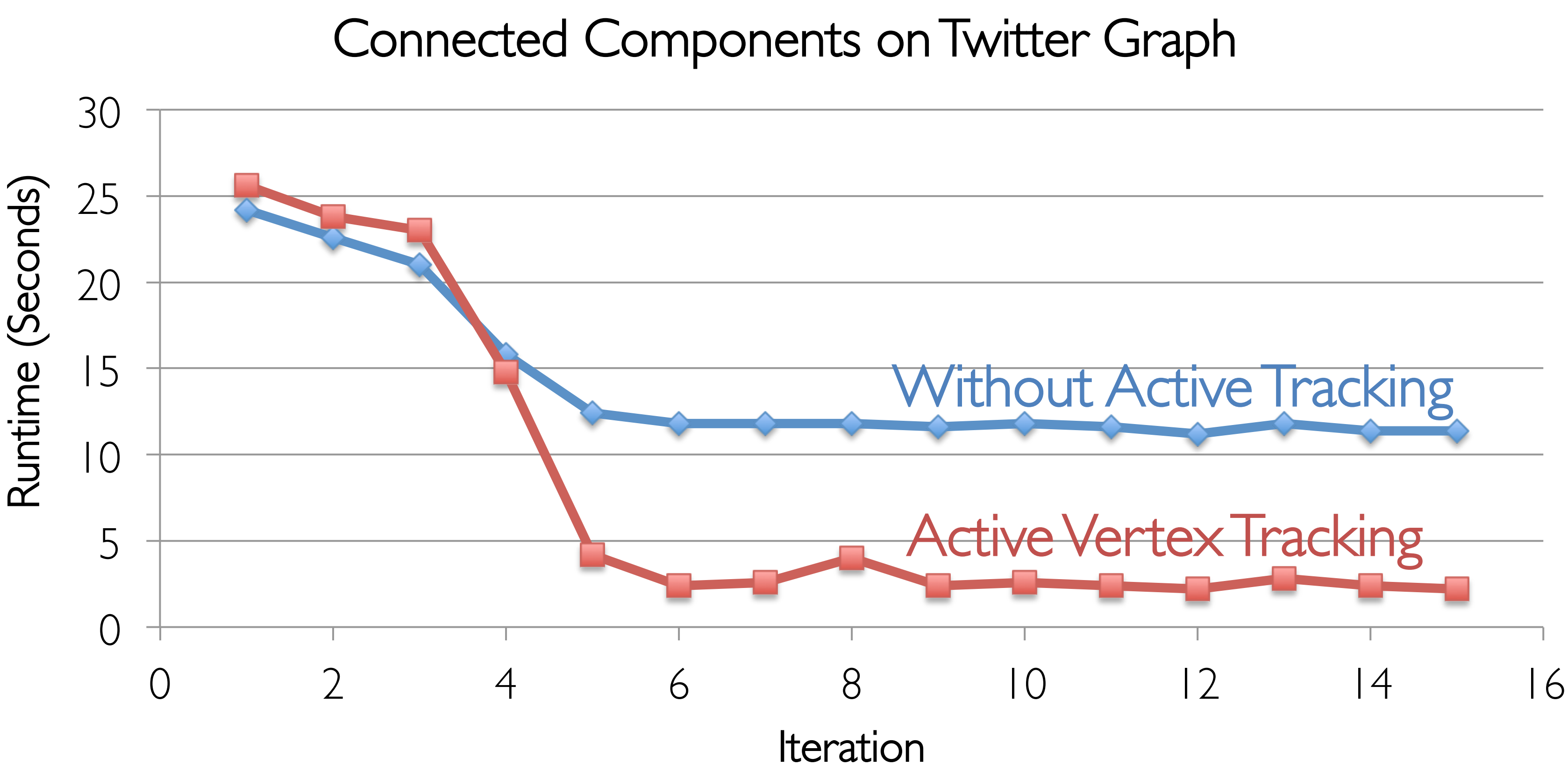
Aggregation for Iterative mrTriplets



Reduction in Communication Due to Cached Updates



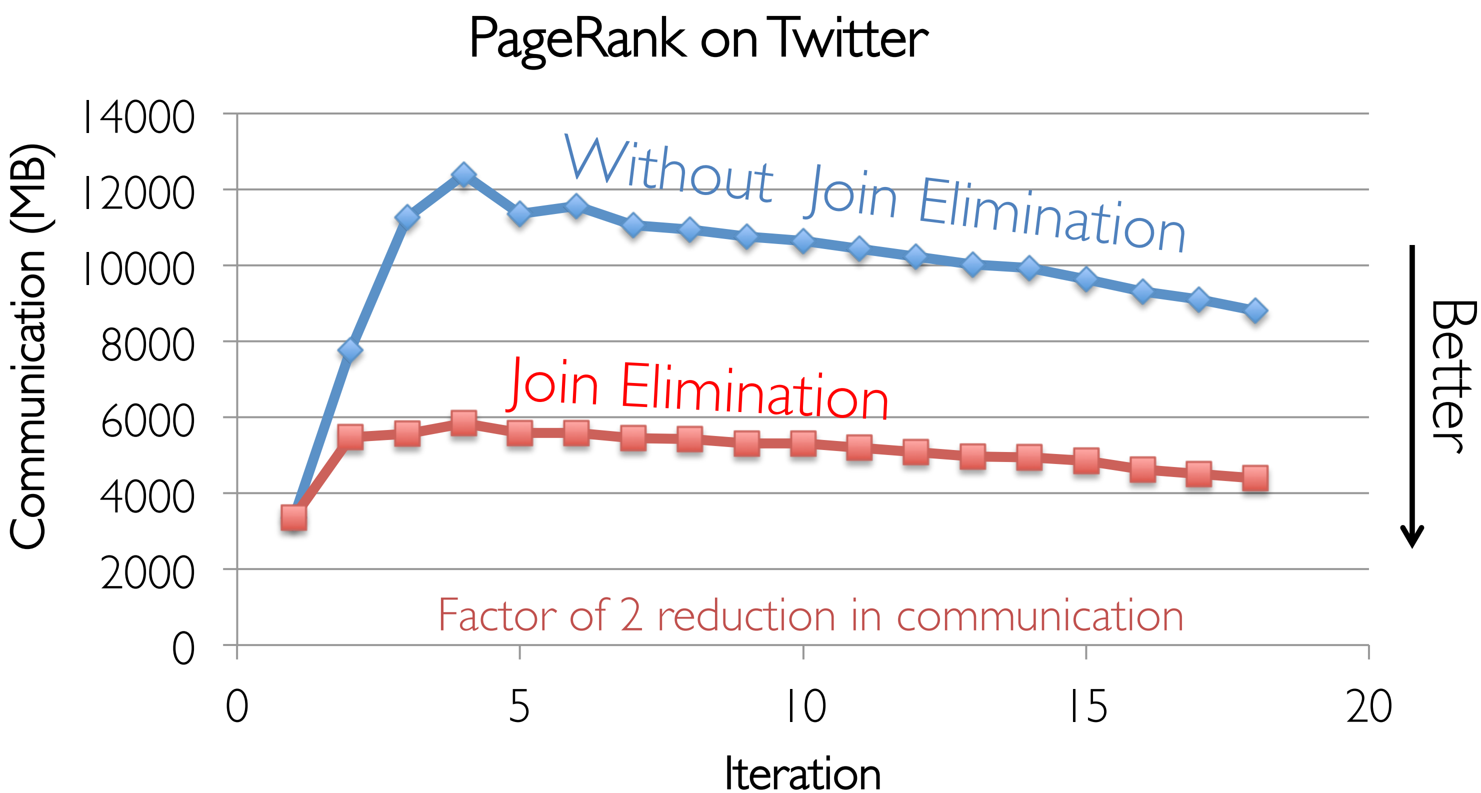
Benefit of Indexing *Active* Vertices



Join Elimination

Identify and bypass joins for unused triplet fields

» Java bytecode inspection



Additional Optimizations

Indexing and Bitmaps:

» To accelerate joins across graphs

» To efficiently construct sub-graphs

Lineage based fault-tolerance

» Exploits Spark lineage to recover in parallel

» Eliminates need for costly check-points

Substantial Index and Data Reuse:

» Reuse routing tables across graphs and sub-graphs

» Reuse edge adjacency information and indices

System Comparison

Goal:

Demonstrate that GraphX achieves performance parity with specialized graph-processing systems.

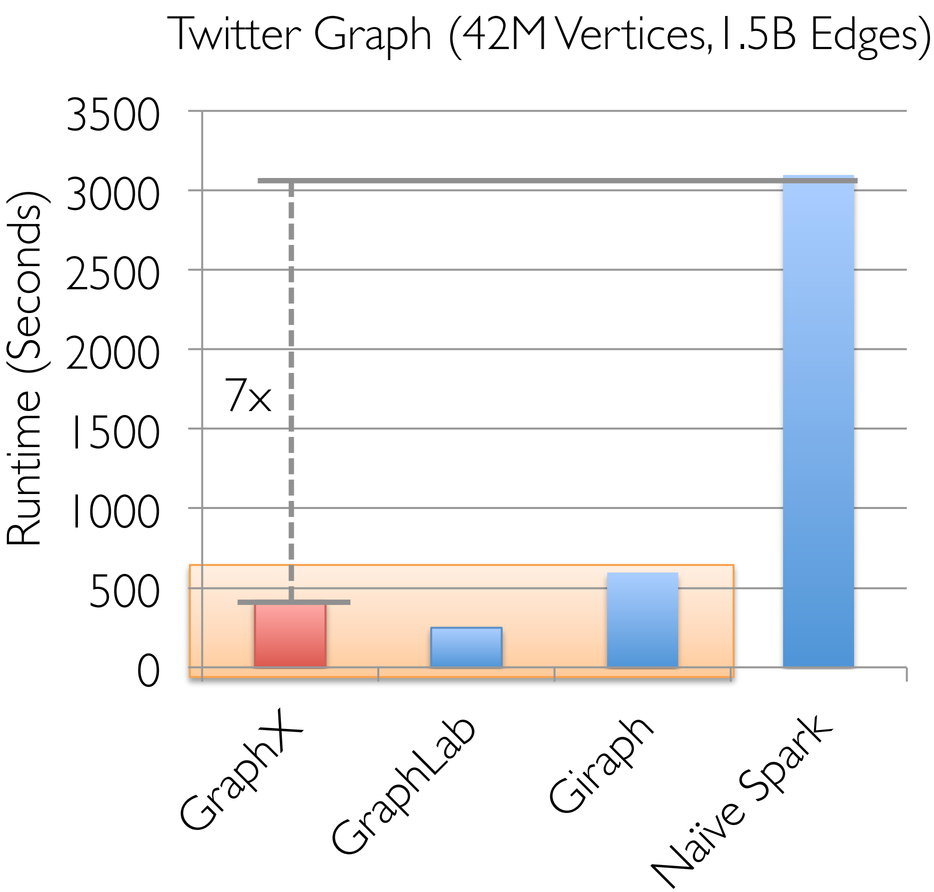
Setup:

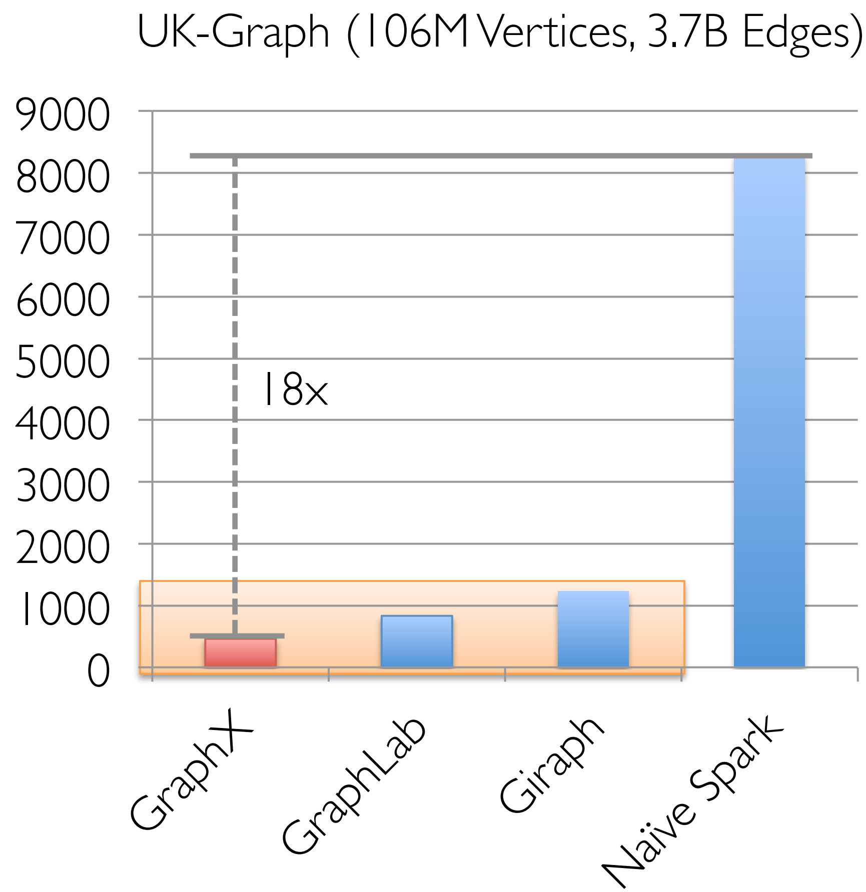
16 node EC2 Cluster (m2.4xLarge) + 1GigE

Compare against GraphLab/PowerGraph (C++), Giraph (Java), & Spark (Java/Scala)

PageRank Benchmark

EC2 Cluster of 16 x m2.4xLarge (8 cores) + 1GigE

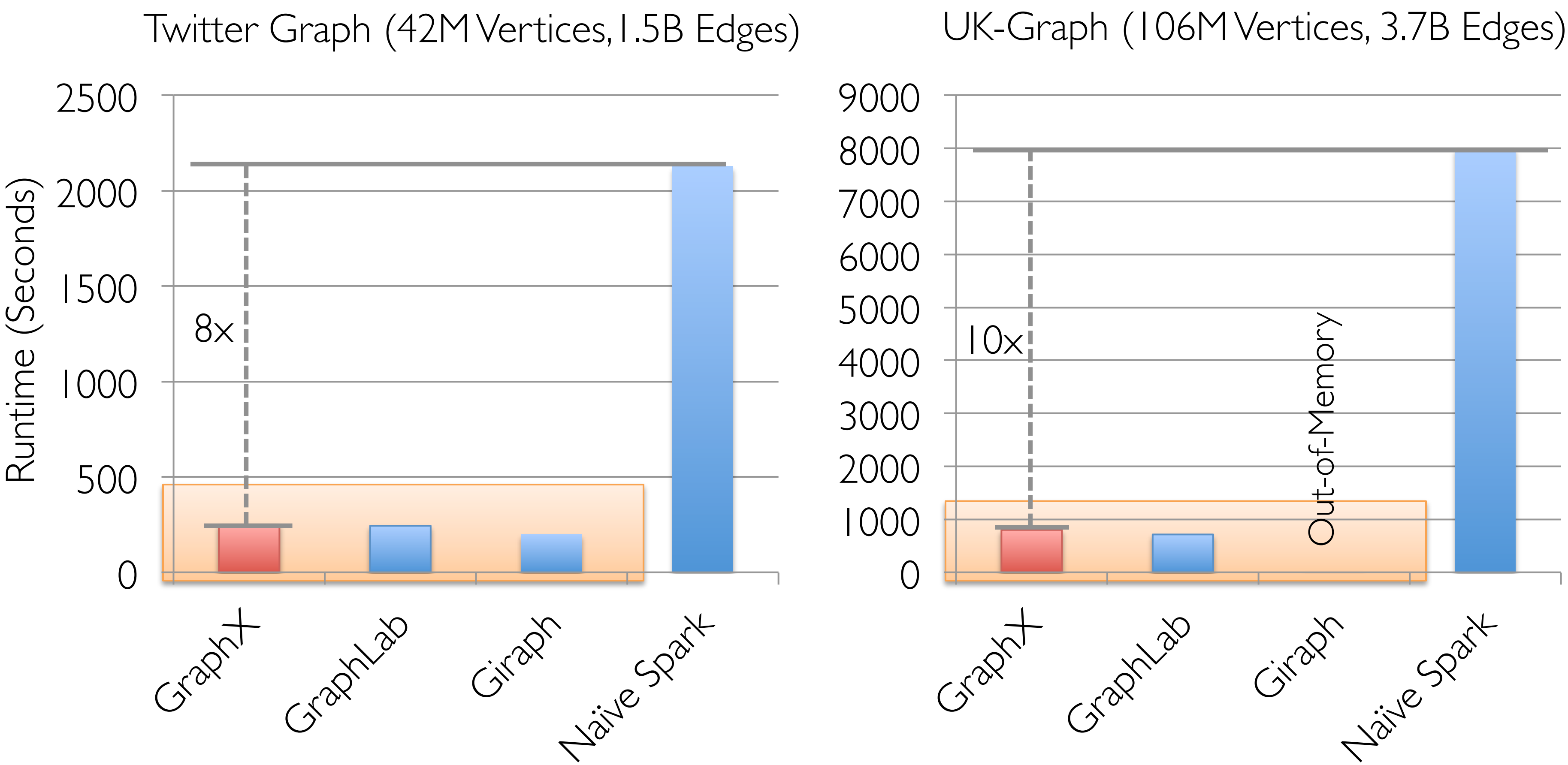




GraphX performs comparably to state-of-the-art graph processing systems.

Connected Comp. Benchmark

EC2 Cluster of 16 x m2.4xLarge (8 cores) + 1GigE

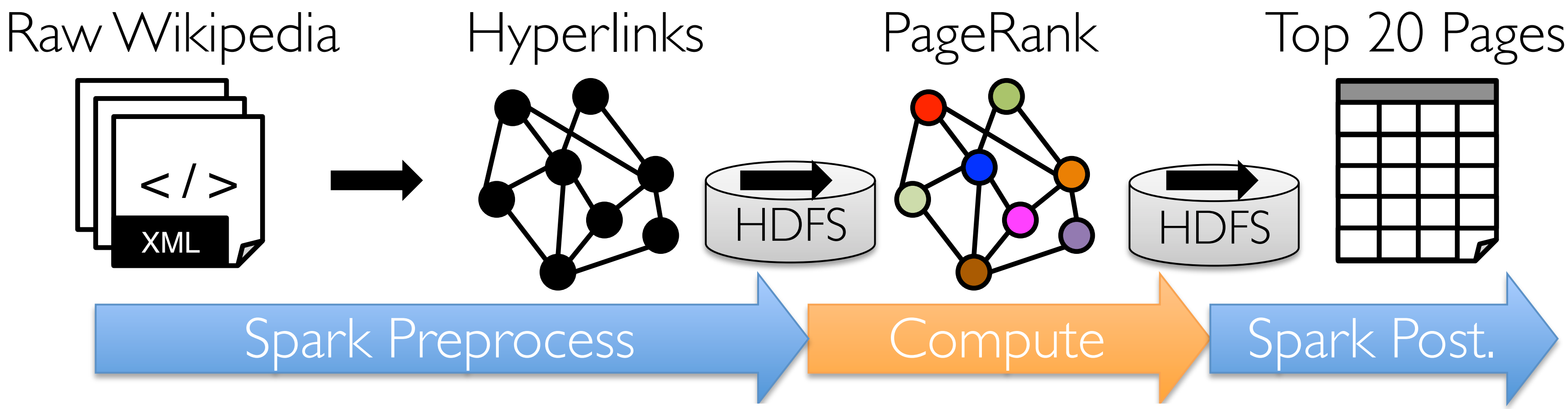


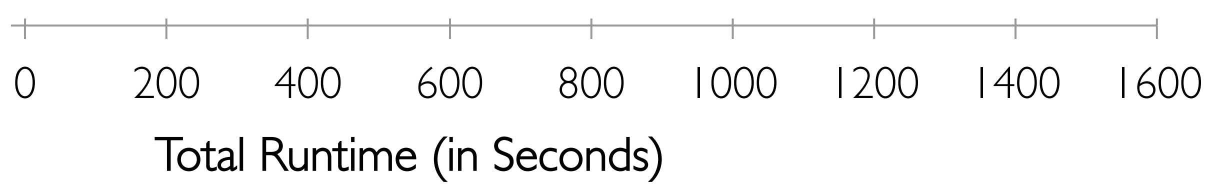
GraphX performs comparably to state-of-the-art graph processing systems.

Graphs are just one stage….

What about a pipeline?

A Small Pipeline in GraphX





Timed end-to-end GraphX is the *fastest*

Adoption and Impact

GraphX is now part of Apache Spark

- Part of Cloudera Hadoop Distribution

In production at Alibaba Taobao

- Order of magnitude gains over Spark

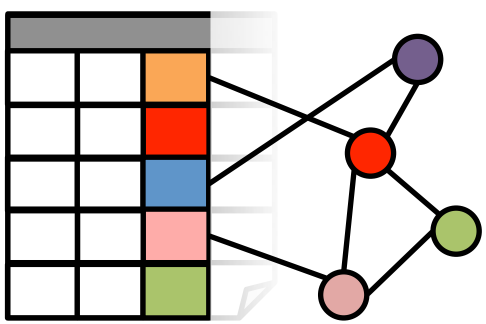
Inspired GraphLab Inc. SFrame technology

- Unifies Tables & Graphs on Disk

GraphX → Unified Tables and Graphs

New API

Blurs the distinction between Tables and Graphs



New System

Unifies Data-Parallel Graph-Parallel Systems

APACHE GIRAPH

Spark

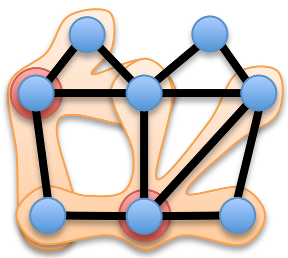
GraphLab

Enabling users to easily and efficiently express the entire analytics pipeline

What did we Learn?

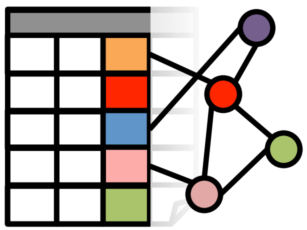
Specialized Systems

Graph Systems



Integrated Frameworks

GraphX

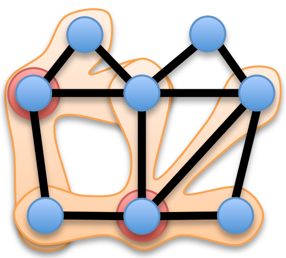


Future Work

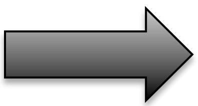
Specialized Systems

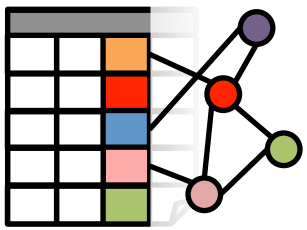
Integrated Frameworks

Graph Systems

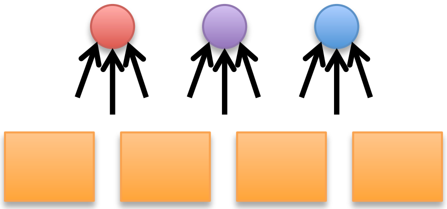


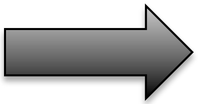
GraphX





Parameter Server





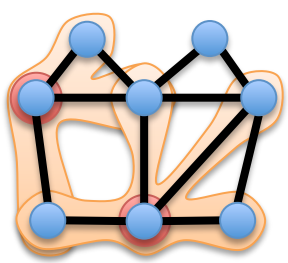
?

Future Work

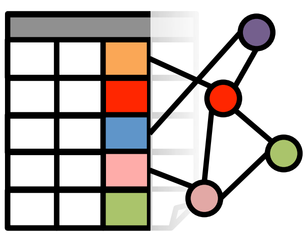
Specialized Systems

Integrated Frameworks

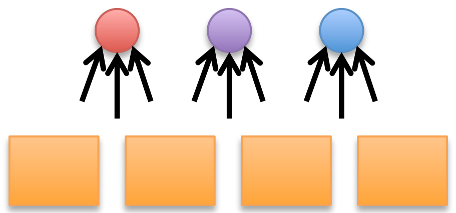
Graph Systems



GraphX



Parameter Server



Asynchrony

Non-deterministic Shared-State

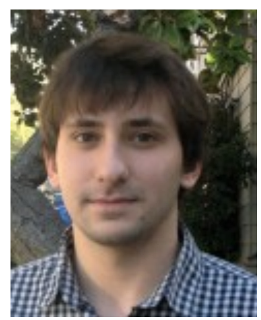
Thank You

http://amplab.cs.berkeley.edu/projects/graphx/

jegonzal@eecs.berkeley.edu

Reynold
Xin

Ankur
Dave



Daniel
Crankshaw

Michael
Franklin

Ion
Stoica

Related Work

Specialized Graph-Processing Systems:

GraphLab [UAI'10], Pregel [SIGMOD'10], Signal-Collect [ISWC'10], Combinatorial BLAS [IJHPCA'11], GraphChi [OSDI'12], PowerGraph [OSDI'12], Ligra [PPoPP'13], X-Stream [SOSP'13]

Alternative to Dataflow framework:

Naiad [SOSP'13]: GraphLINQ
Hyracks: Pregelix [VLDB'15]

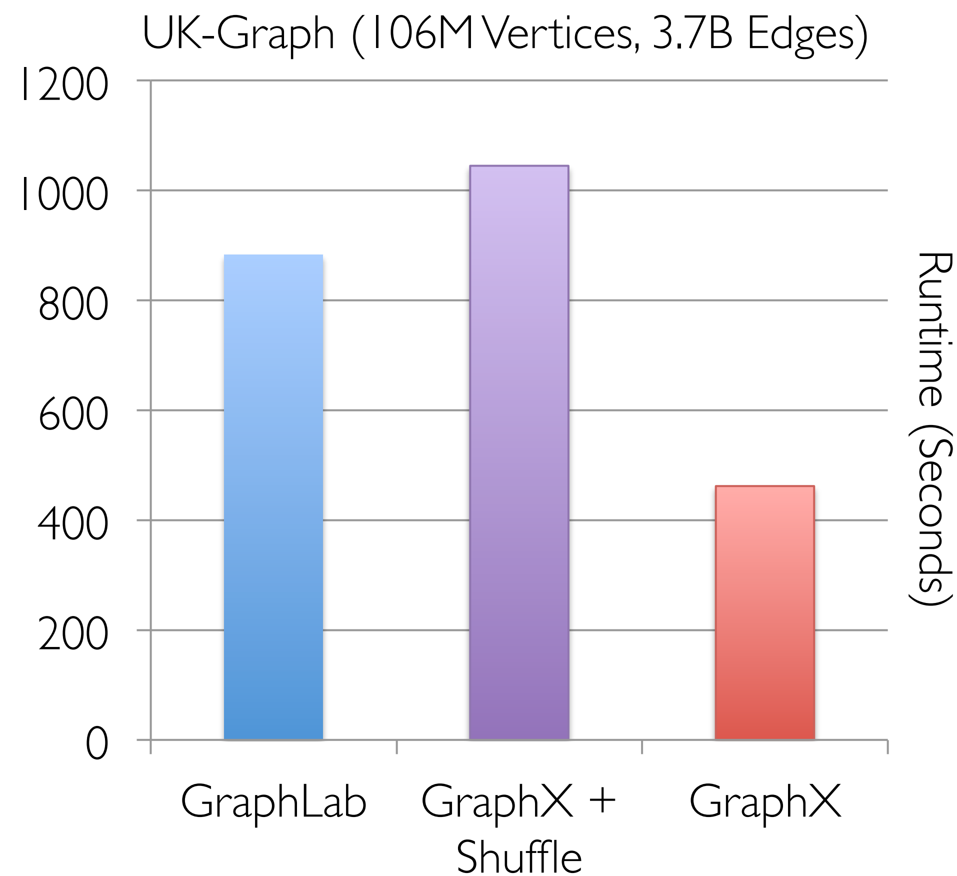
Distributed Join Optimization:

Multicast Join [Afrati et al., EDBT'10]
Semi-Join in MapReduce [Blanas et al., SIGMOD'10]

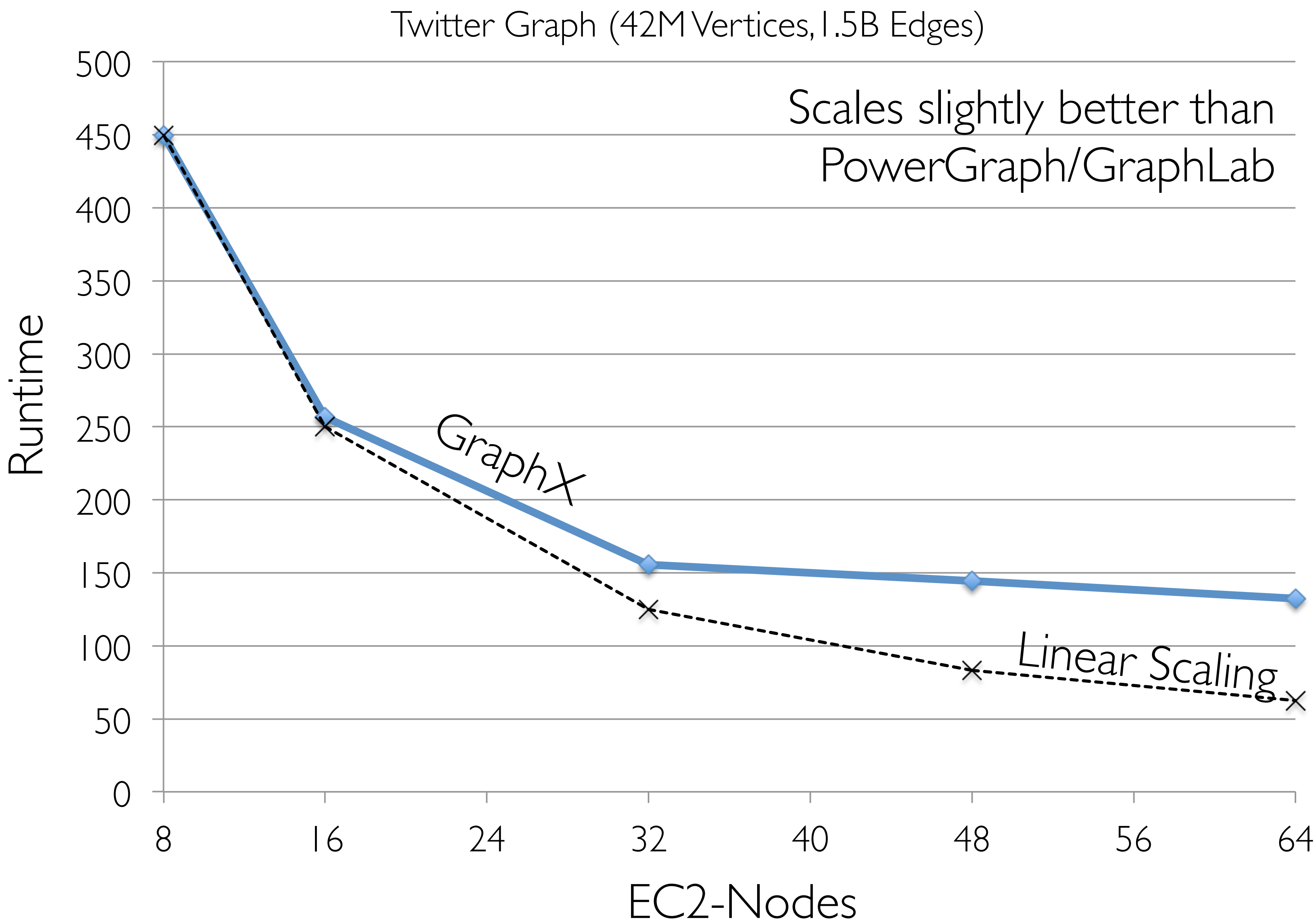
Edge Files Have Locality

GraphLab rebalances the edge-files on-load.

GraphX preserves the on-disk layout through Spark.
→ Better Vertex-Cut



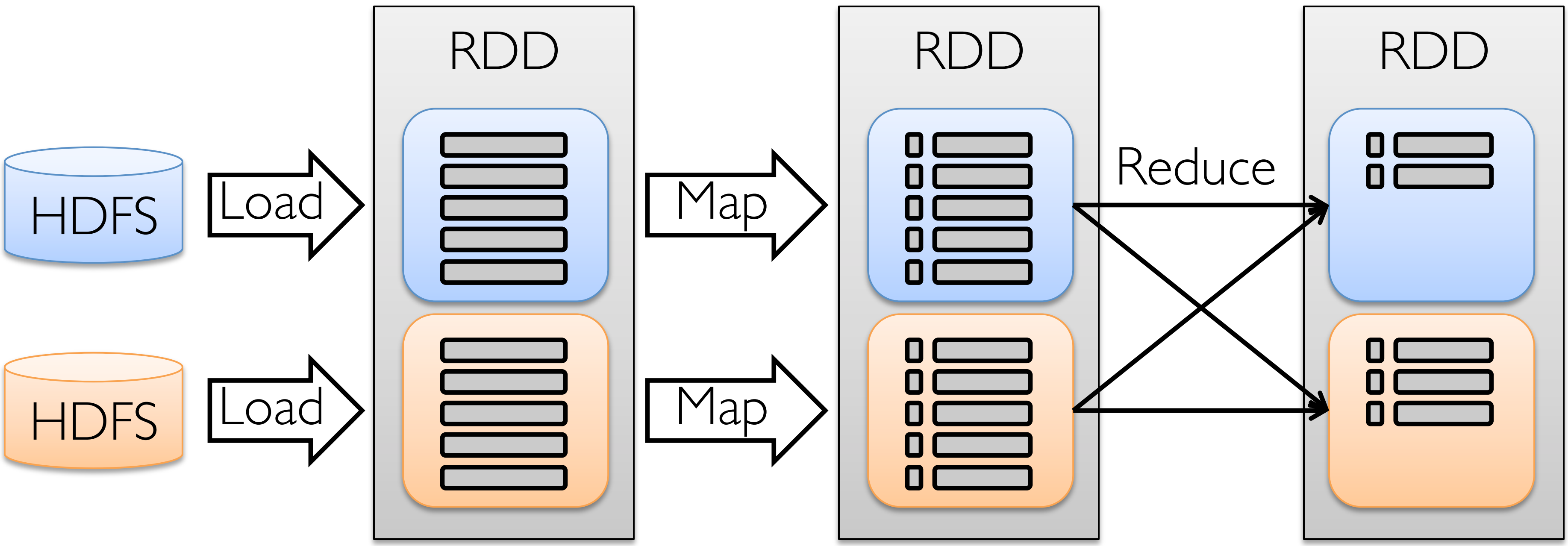
Scalability



Apache Spark Dataflow Platform

Zaharia et al., NSDI'12

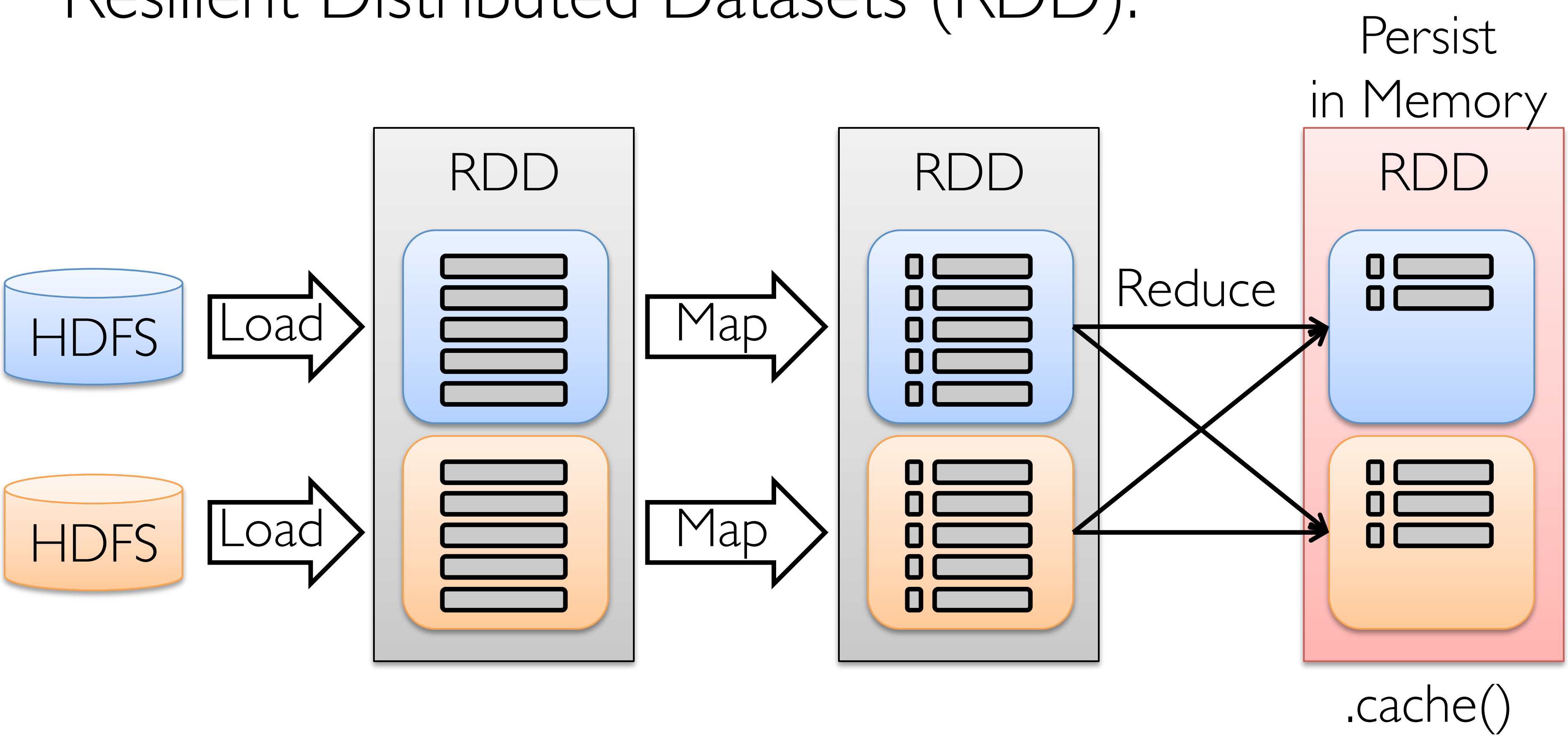
Resilient Distributed Datasets (RDD):



Apache Spark Dataflow Platform

Zaharia et al., NSDI'12

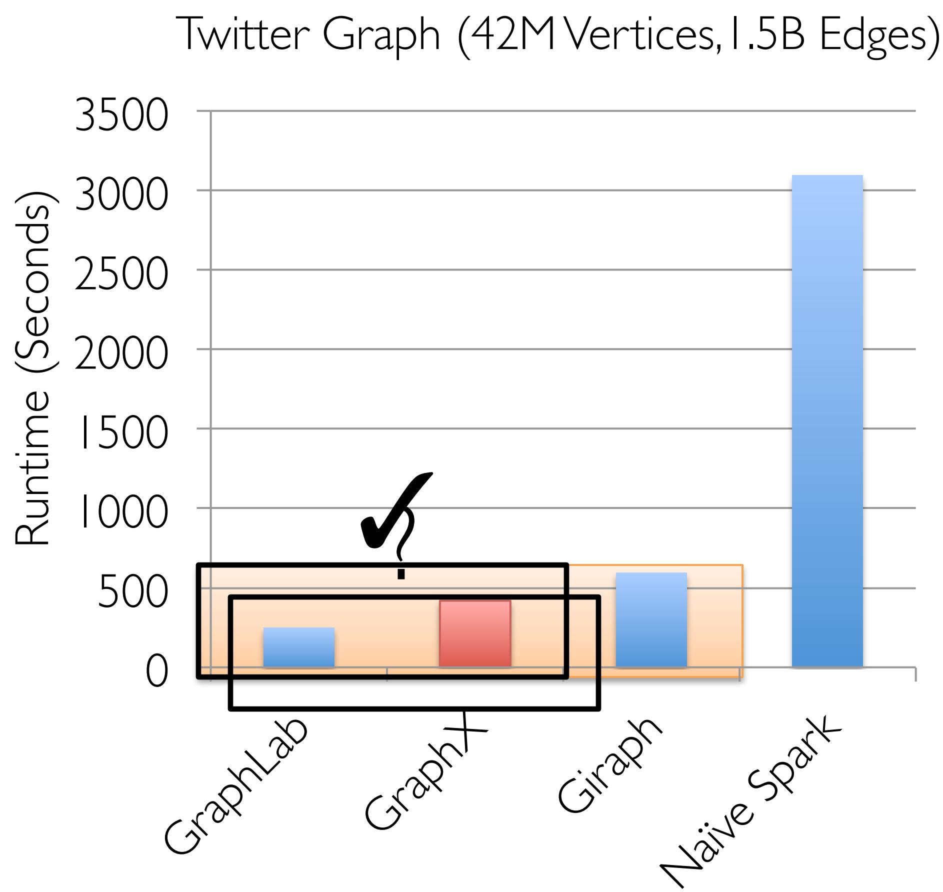
Resilient Distributed Datasets (RDD):

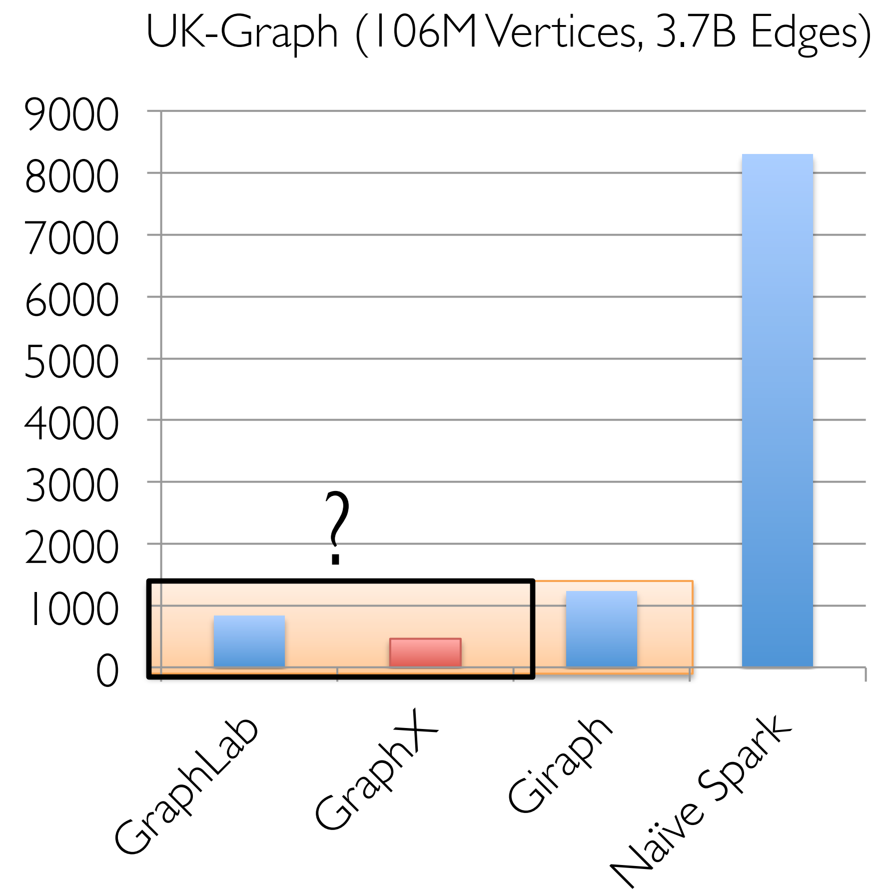


Optimized for iterative access to data.

PageRank Benchmark

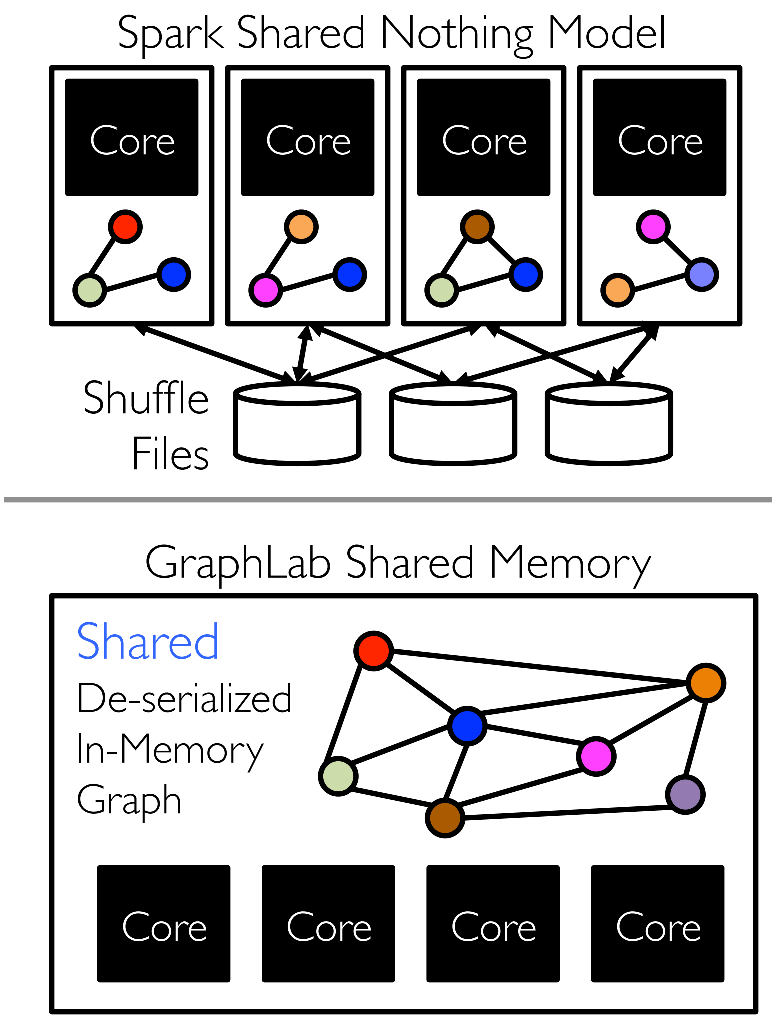
EC2 Cluster of 16 x m2.4xLarge Nodes + 1GigE



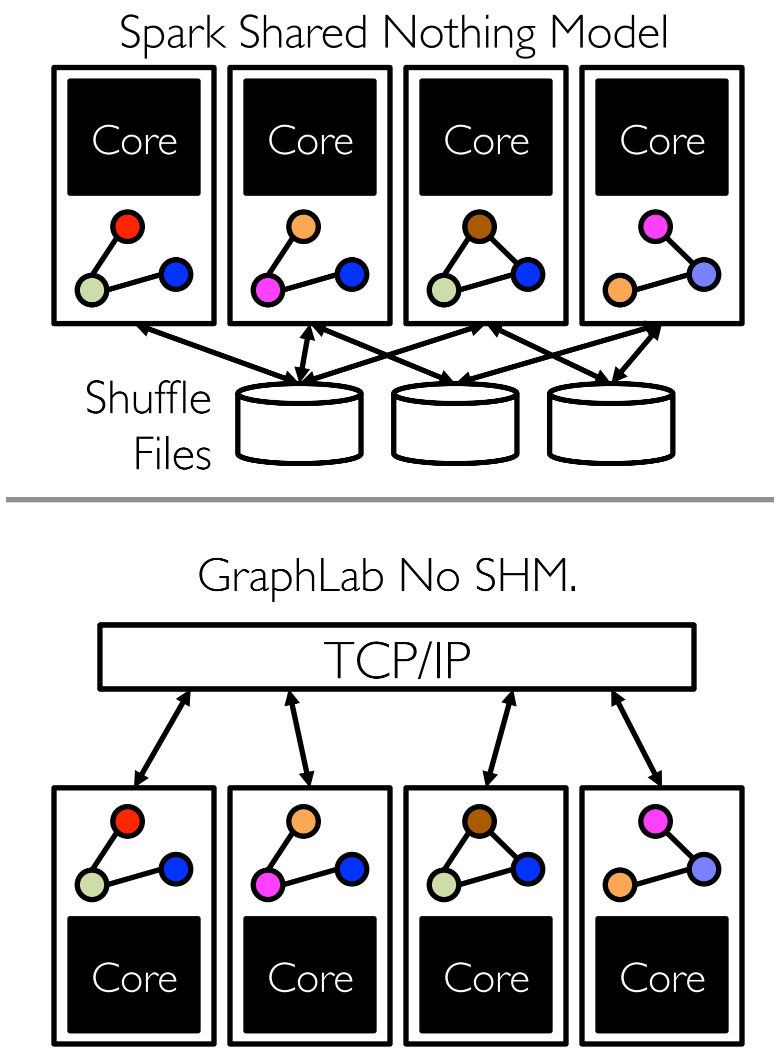


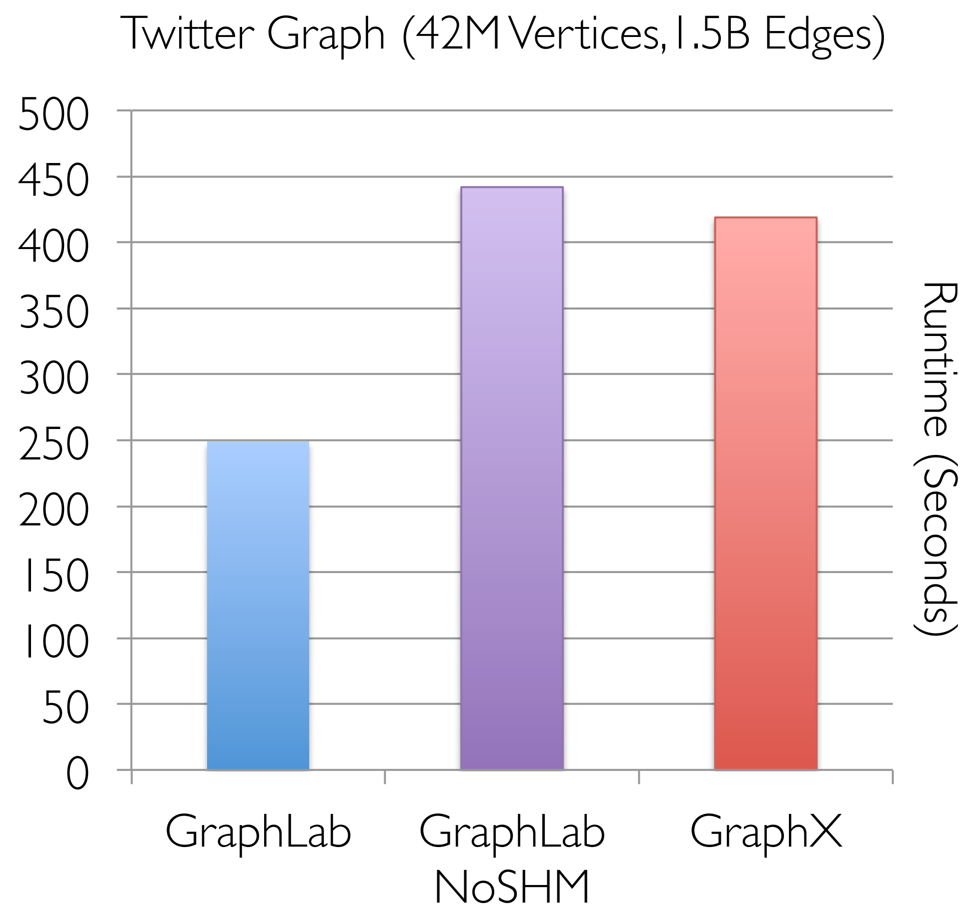
GraphX performs comparably to state-of-the-art graph processing systems.

Shared Memory Advantage

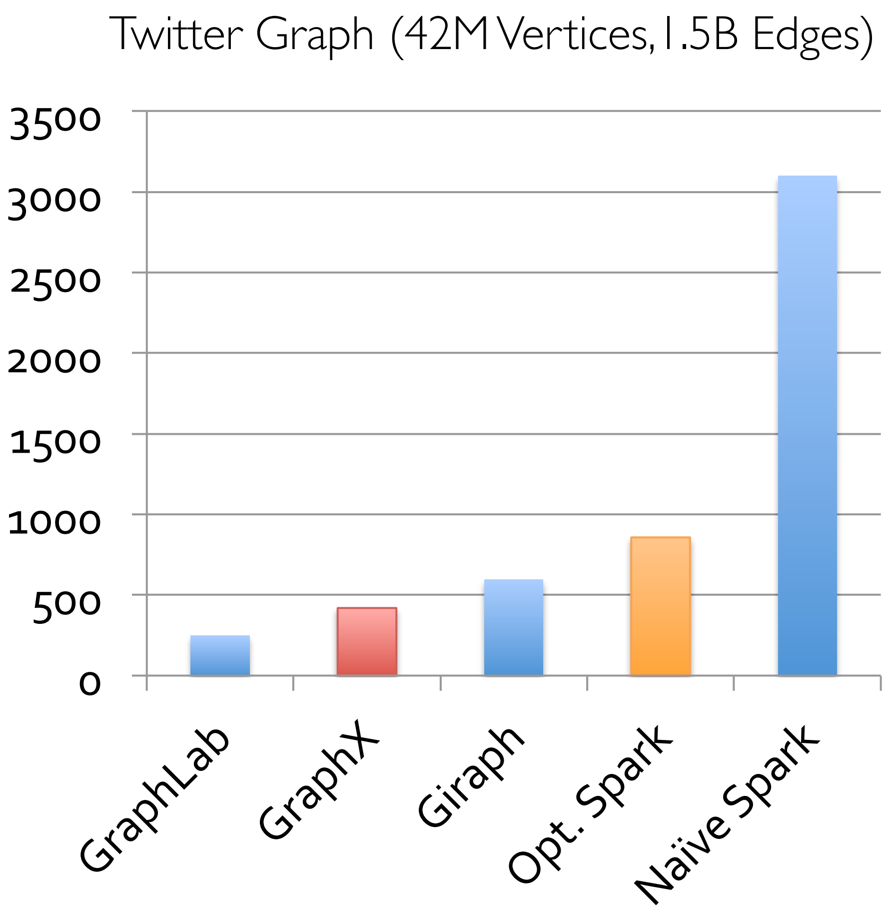


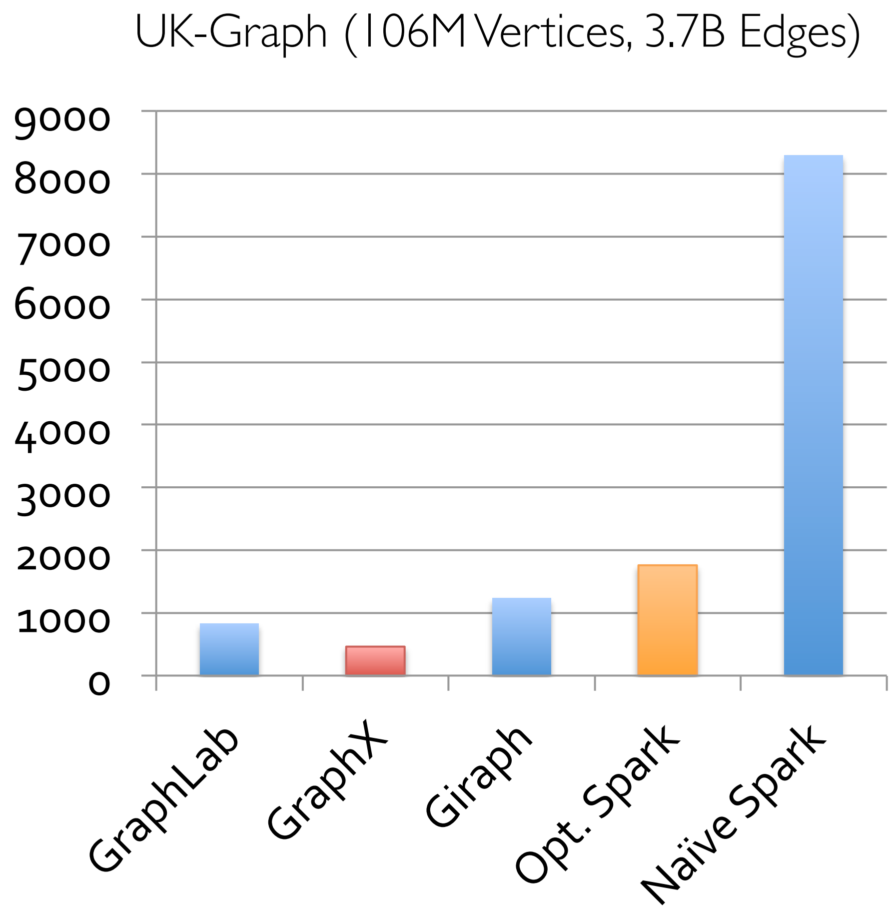
Shared Memory Advantage





PageRank Benchmark

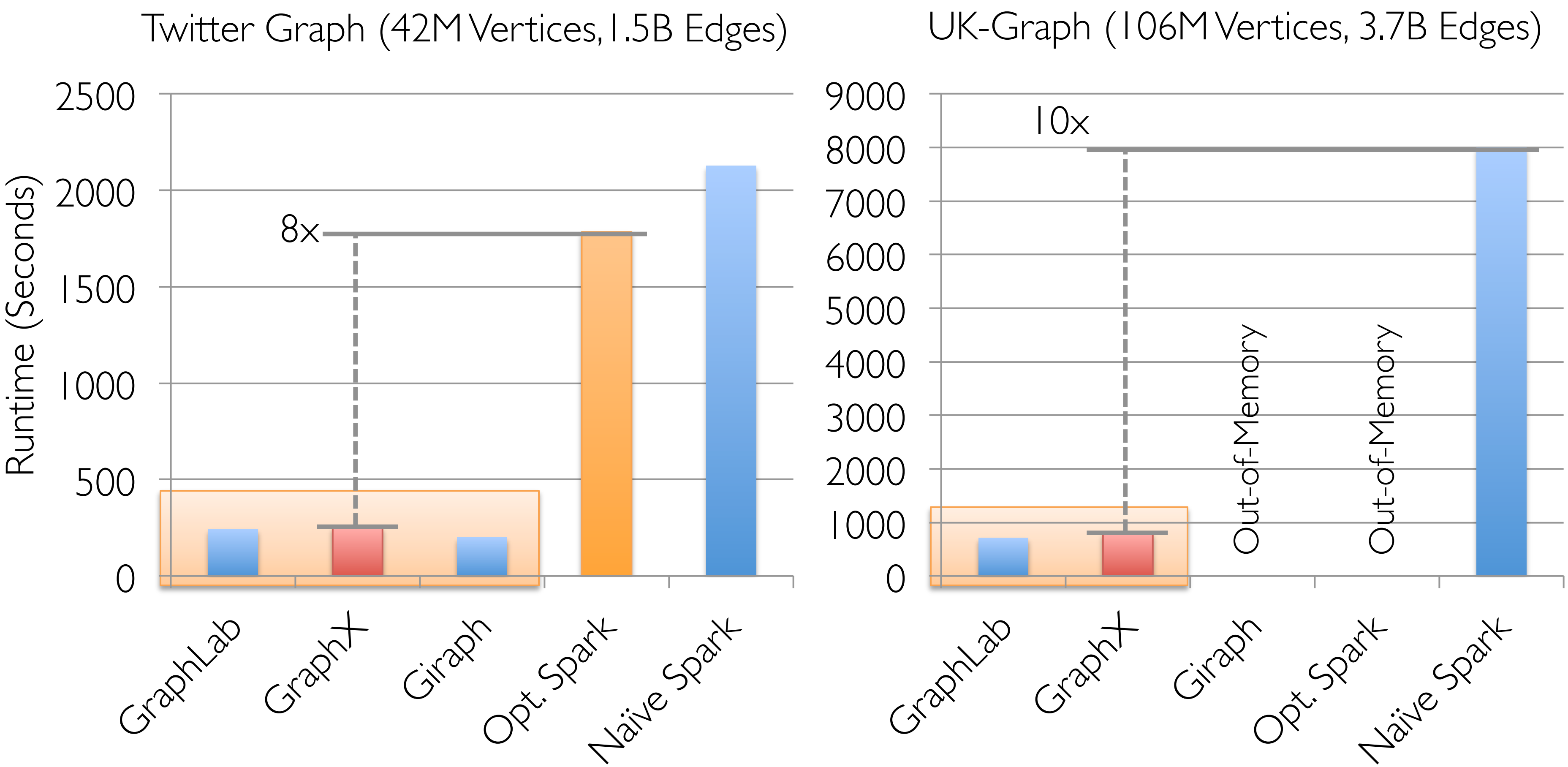




GraphX performs comparably to
state-of-the-art graph processing systems.

Connected Comp. Benchmark

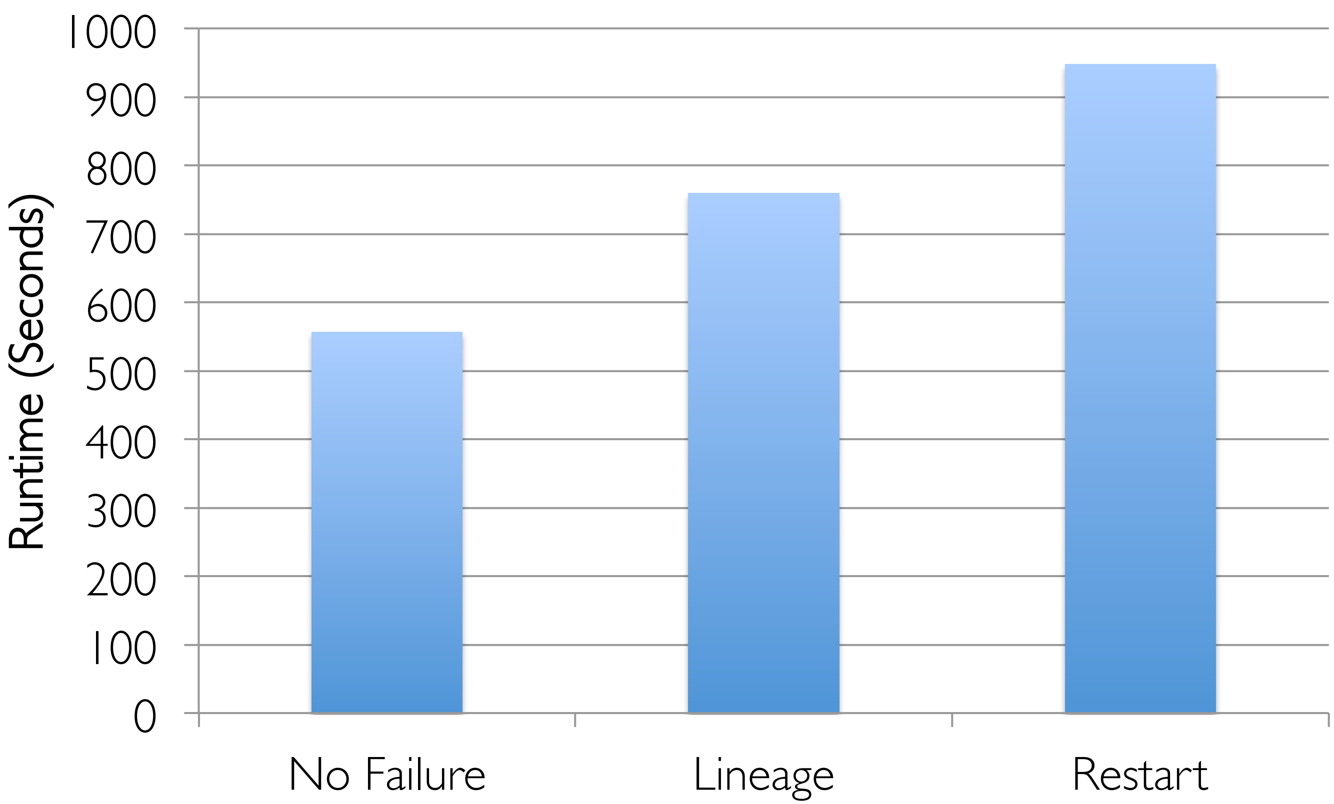
EC2 Cluster of 16 x m2.4xLarge Nodes + 1GigE



GraphX performs comparably to state-of-the-art graph processing systems.

Fault-Tolerance

Leverage Spark Fault-Tolerance Mechanism



Graph-Processing Systems

GraphLab

APACHE GIRAPH

Pregel oogle

Ligra CombBLAS X-Stream

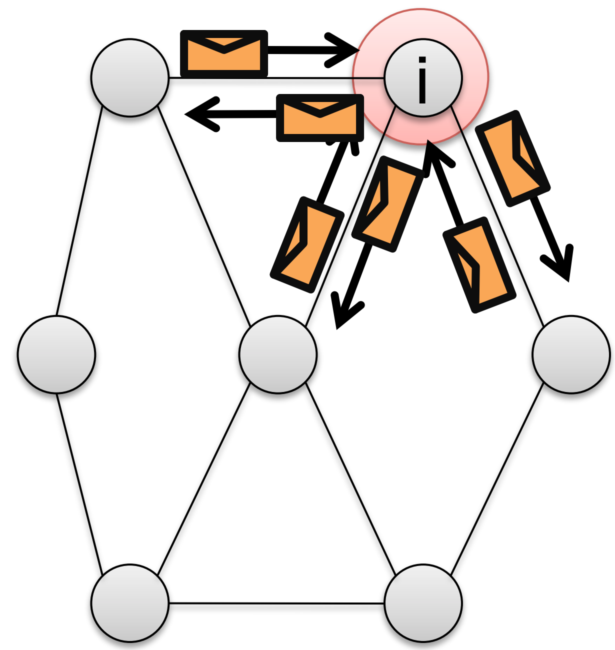
GraphChi GPS Kineograph

Representation

Expose specialized API to simplify graph programming.

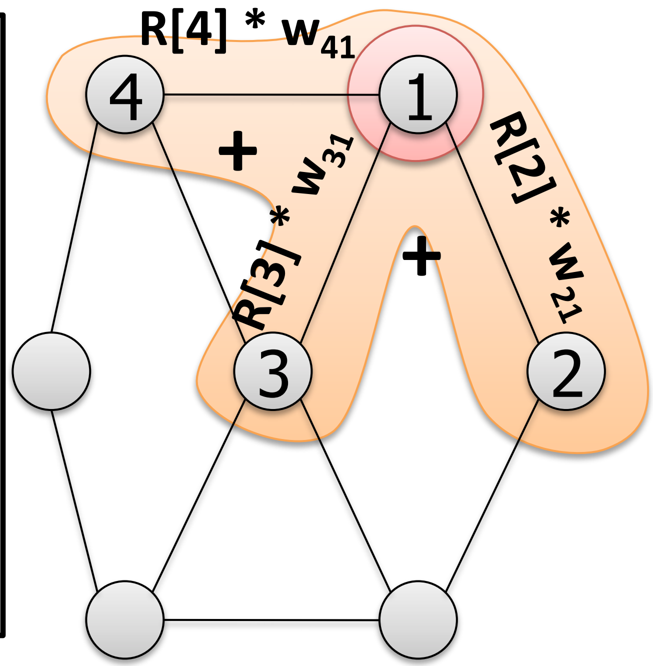
Vertex-Program Abstraction

```
Pregel_PageRank(i, messages) :
  // Receive all the messages
  total = 0
  foreach( msg in messages) :
    total = total + msg

  // Update the rank of this vertex
  R[i] = 0.15 + total

  // Send new messages to neighbors
  foreach(j in out_neighbors[i]) :
    Send  msg(R[i]) to vertex j
```



The Vertex-Program Abstraction

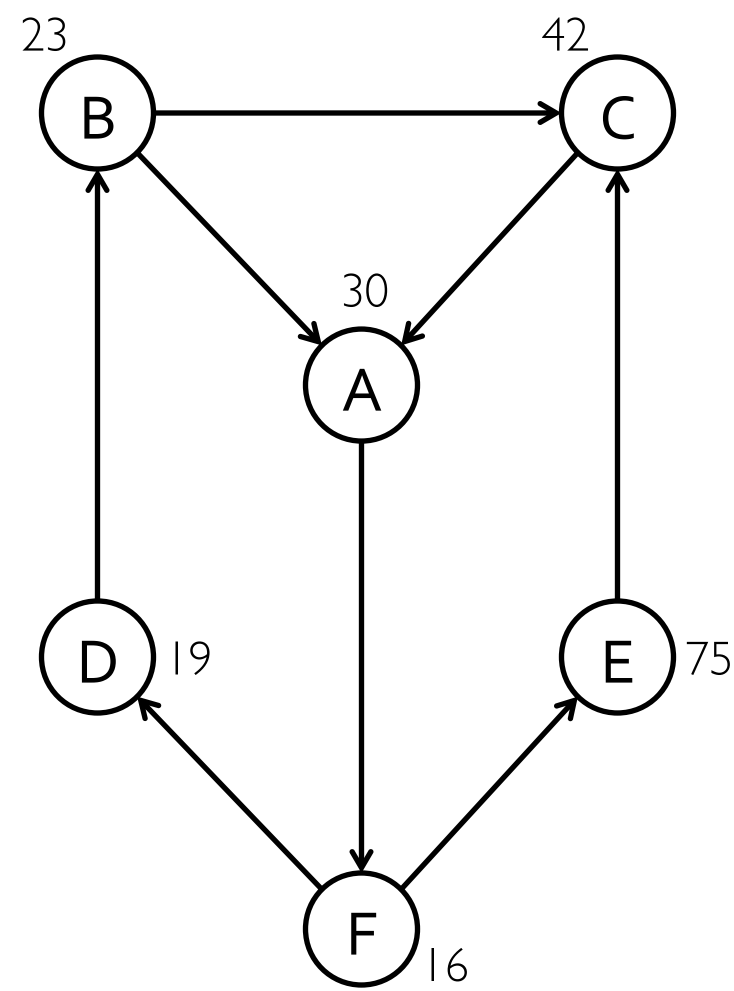
```
GraphLab_PageRank(i)
  // Compute sum over neighbors
  total = 0
  foreach( j in neighbors(i)):
    total += R[j] * w_ji

  // Update the PageRank
  R[i] = 0.15 + total
```



Example: Oldest Follower

Calculate the number of older followers for each user?

```
val olderFollowerAge = graph
  .mrTriplets(
    e => // Map
     if(e.src.age > e.dst.age) {
       (e.srcId, 1)
     else { Empty }
    ,
    (a,b) => a + b // Reduce
  )
  .vertices
```



Enhanced Pregel in GraphX

```
pregelPR(i, messageSum ):
  // Receive all the messages
  total = 0          messageSum
  foreach( msg in messageList) :
    total = total + msg

  // Update the rank of this vertex
  R[i] = 0.15 + total

  // Send new messages to neighbors
  foreach(j in out_neighbors[i]) :
    Send msg(R[i]/E[i,j]) to vertex
```

```
combineMsg(a, b):
  // Compute sum of two messages
  return a + b
```

```
sendMsg(i→j, R[i], R[j], E[i,j]):
  // Compute single message
  return msg(R[i]/E[i,j])
```

Require Message Combiners

Remove Message Computation from the Vertex Program

PageRank in GraphX

```
// Load and initialize the graph
val graph = GraphBuilder.text("hdfs://web.txt")
val prGraph = graph.joinVertices(graph.outDegrees)

// Implement and Run PageRank
val pageRank =
  prGraph.pregel(initialMessage = 0.0, iter = 10)(
    (oldV, msgSum) => 0.15 + 0.85 * msgSum,
    triplet => triplet.src.pr / triplet.src.deg,
    (msgA, msgB) => msgA + msgB)
```

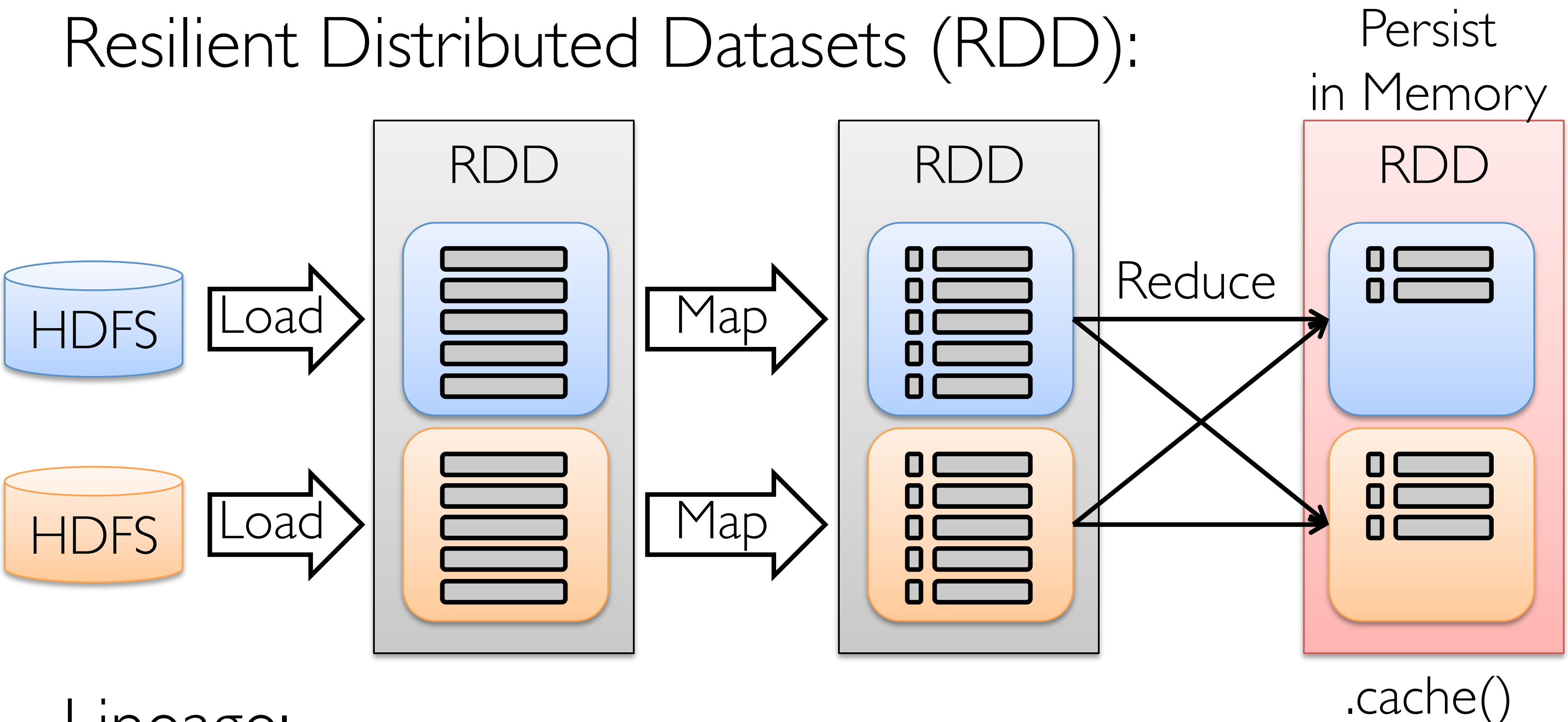
Example Analytics Pipeline

```
// Load raw data tables
val articles = sc.textFile(“hdfs://wiki.xml”).map(xmlParser)
val links = articles.flatMap(article => article.outLinks)
// Build the graph from tables
val graph = new Graph(articles, links)
// Run PageRank Algorithm
val pr = graph.PageRank(tol = 1.0e-5)
// Extract and print the top 20 articles
val topArticles = articles.join(pr).top(20).collect
for ((article, pageRank) <- topArticles) {
  println(article.title + ‘\t’ + pageRank)
}
```

Apache Spark Dataflow Platform

Zaharia et al., NSDI'12

Resilient Distributed Datasets (RDD):



Lineage: