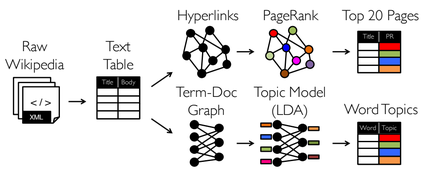
GraphX: Unified Data-Parallel and Graph-Parallel Analytics

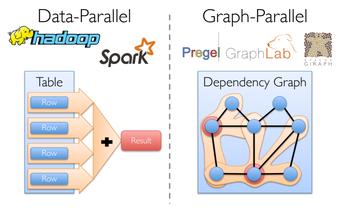
Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, Ion Stoica

MOTIVATION

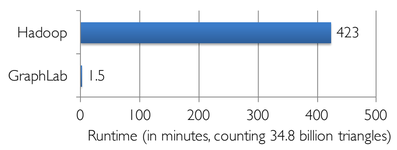
- Graph analytics involves viewing the same data as both graphs and tables



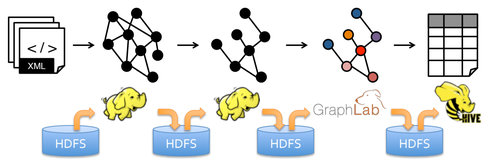
- We currently need separate systems to support each view: both specialized graph systems and general-purpose data-parallel systems



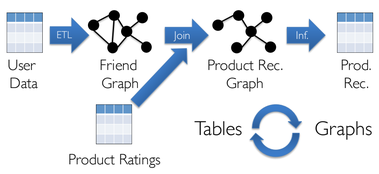
- Graph systems provide specialized graph-parallel APIs and exploit the graph structure to reduce communication



- But separate systems increase complexity, lead to unnecessary data movement, and hinder data structure reuse

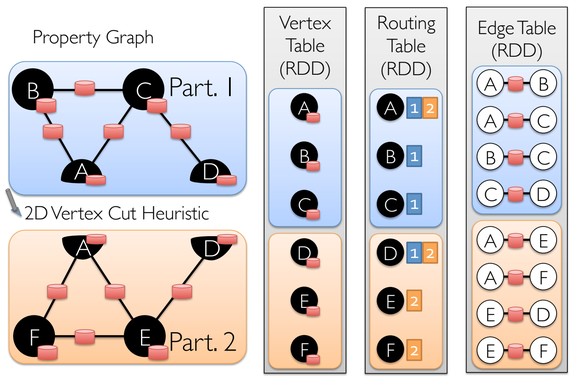


- GraphX embeds graph computation in Spark and implements a variety of join optimizations, enabling a much wider range of computation



GRAPHS AS TABLES

Horizontally partitioned vertex and edge tables with indexing and join site information



Extends the Spark RDDs

- *Immutable*, partitioned set of vertices and edges

Equivalent to GraphLab representation

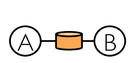
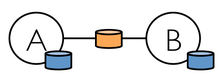
- *Vertex-cut* partitioning & vertex data movement pattern

THE GRAPHX API

```
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)]
  // Computation -----------------------------------------
  def mrTriplets(mapF: Edge[V, E] => List[(Id, T)],
                 reduceF: (T, T) => T): Graph[T, E]
  // Convenience functions -------------------------------
  def mapV(m: (Id, V) => T): Graph[T, E]
  def mapE(m: Edge[V,E] => T): Graph[V, T]
  def joinV(tbl: Table [(Id, T)]): Graph[(V, T), E]
  def joinE(tbl: Table [(Id, Id, T)]): Graph[V, (E, T)]
  def reverse: Graph[V, E]
  def subgraph(pV: (Id, V) => Boolean,
               pE: Edge[V, E] => Boolean): Graph[V, E]
}
```

The `triplets` operator joins vertices and edges:

SELECT src.Id, dst.Id, src.attr, e.attr, dst.attr
FROM edges **AS** e **JOIN** vertices **AS** src, vertices **AS** dst
ON e.srcId = src.Id **AND** e.dstId = dst.Id

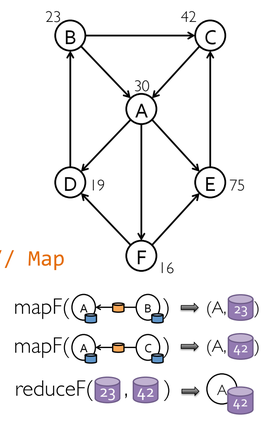
Vertices:  Edges:  Triplets: 

The `mrTriplets` operator sums adjacent triplets.

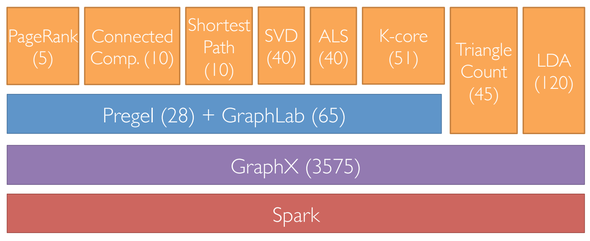
SELECT t.dstId, *reduceUDF*(*mapUDF*(t)) **AS** sum
FROM triplets **AS** t **GROUPBY** t.dstId

What is the age of the oldest follower for each user?

```
val oldestFollowerAge = graph
   .mrTriplets(
      e => (e.dst.id, e.src.age), // Map
      (a,b) => max(a, b) // Reduce
   )
   .vertices
```



CURRENT SYSTEM

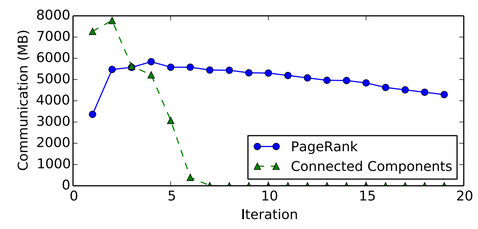


EXAMPLE: PAGERANK

```
// Load graph:
val vertices = spark.textFile("hdfs://path/pages.csv")
val edges = spark.textFile("hdfs://path/to/links.csv")
val g = new Graph(vertices, edges)
// Restrict to subgraph:
val g1 = g.subgraph(_.split('\t')(2) == "Berkeley")
println(g1.edges.count)
// Run PageRank:
val pg = Pregel.run(g1, numIter = 10)(
        (v, msg) => 0.15 + 0.85 * msg,
        (me_id, e) => e.src.data._2 * e.data,
        (a, b) => a + b)
// Get highest-ranking page:
val maxRank = pg.vertices.top
```

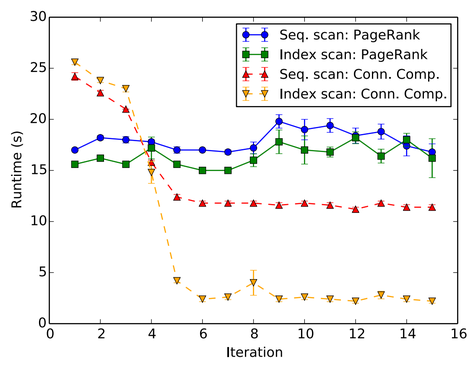
OPTIMIZATIONS

- *Incremental vertex replication.* Iterative graph algorithms have active vertex sets that shrink as the algorithm converges. We speed up *mrTriplets* by shipping only changed vertex attributes each iteration.



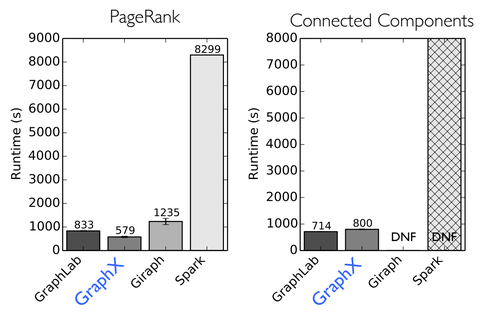
- *Structural index reuse.* Vertex and edge indices enable fast aggregations and joins. We exploit immutability by reusing vertex/edge indices across operations, including filters, speeding up PageRank by 41% (27 s to 16 s per iteration).

- *Sequential scan vs index scan.* Though the map phase in *mrTriplets* logically uses a sequential scan over edges, this is inefficient for small active vertex sets. When the active vertex fraction is less than 0.8, we instead scan the clustered index on vertices and filter by activeness.



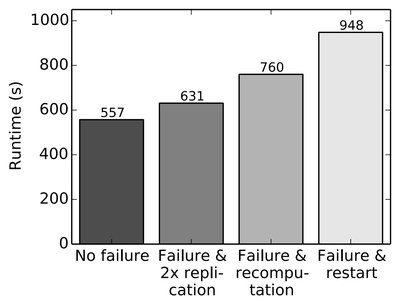
EVALUATION

- Comparable performance to fastest specialized systems



uk-2007-05 web graph (3.7B edges)

- Gains efficient fault tolerance from Spark



PageRank for 20 iterations on uk-2007-05 web graph

Failure in iteration 11

Intel Science & Technology Center for Cloud Computing
Carnegie Mellon University
Georgia Tech
intel
PRINCETON UNIVERSITY
UC Berkeley
UNIVERSITY of WASHINGTON