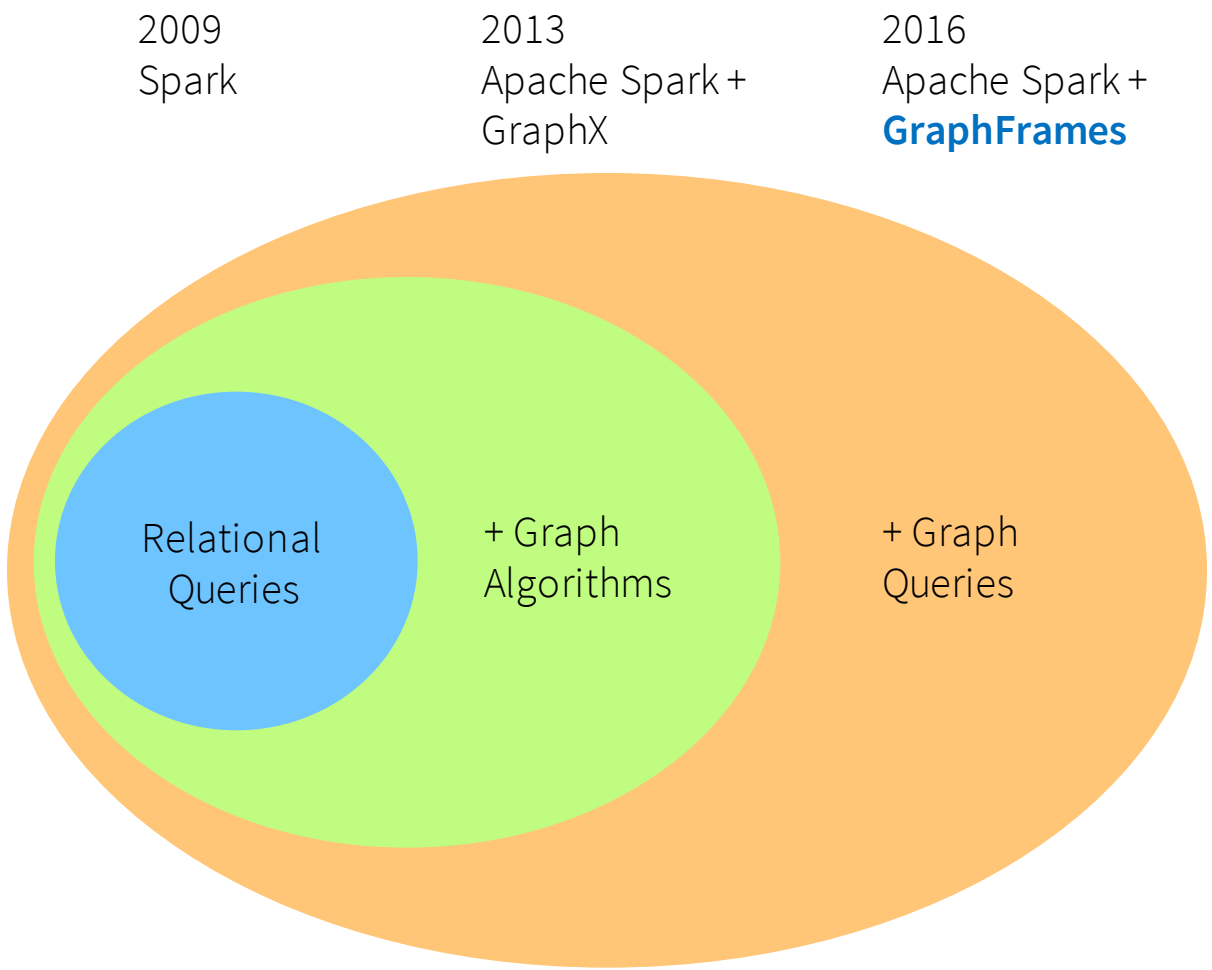
GraphFrames: An Integrated API for Mixing Graph and Relational Queries

Ankur Dave
UC Berkeley AMPLab

Joint work with Alekh Jindal (Microsoft), Li Erran Li (Uber), Reynold Xin (Databricks), Joseph Gonzalez (UC Berkeley), and Matei Zaharia (MIT and Databricks)

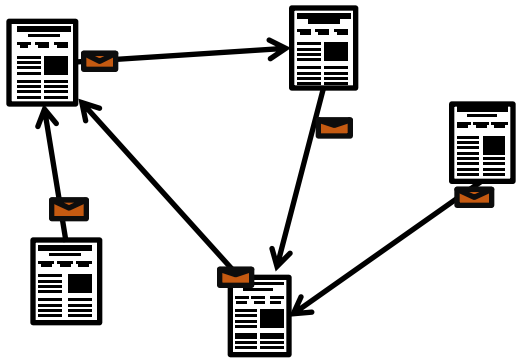
Trend: Unified Graph Analysis



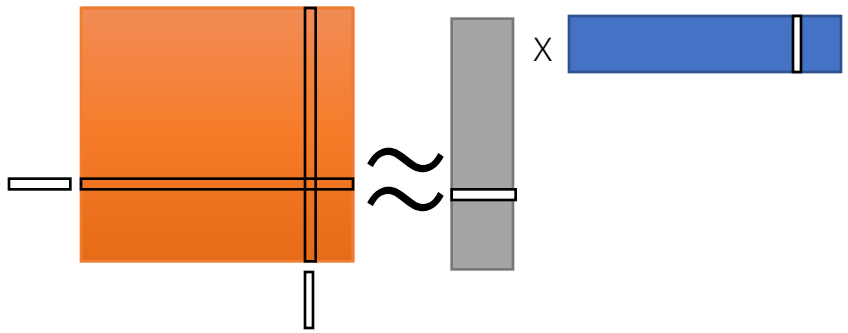
Graph Algorithms vs. Graph Queries

Graph Algorithms

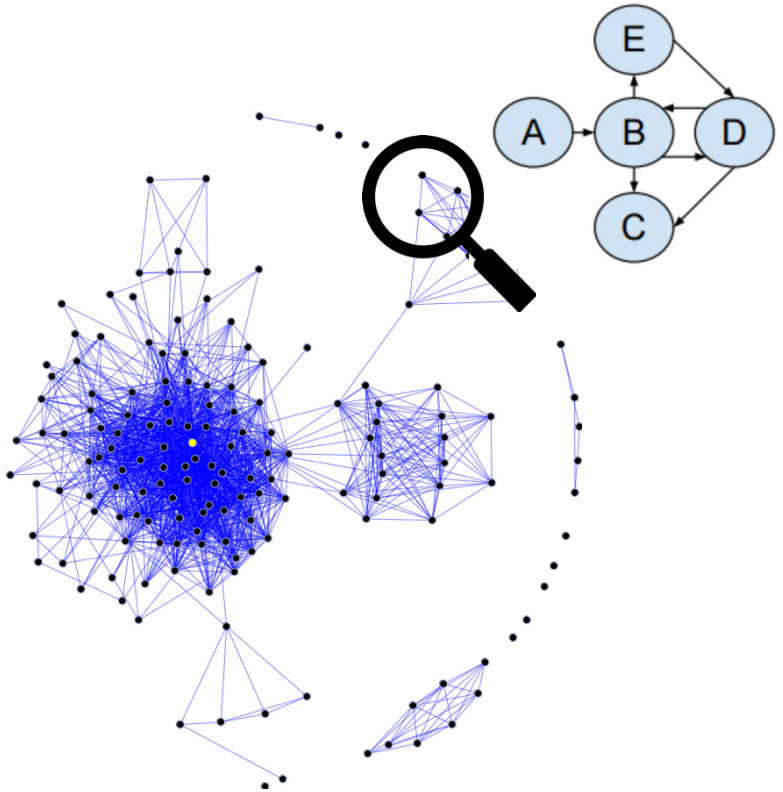
PageRank



Alternating Least Squares

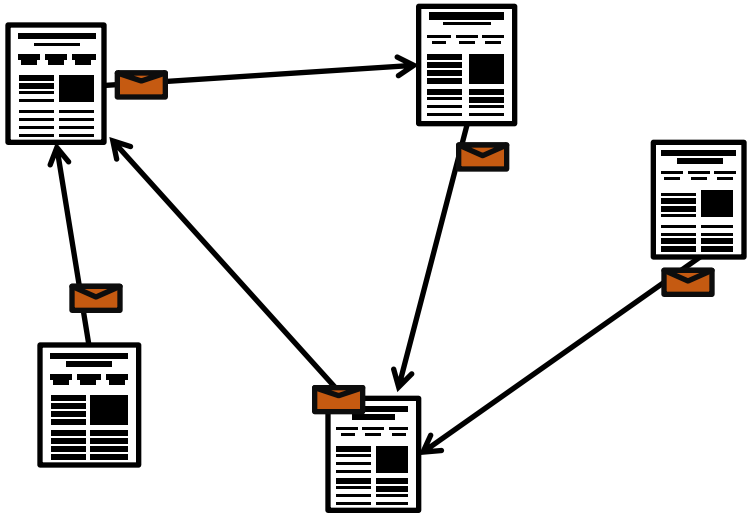


Graph Queries

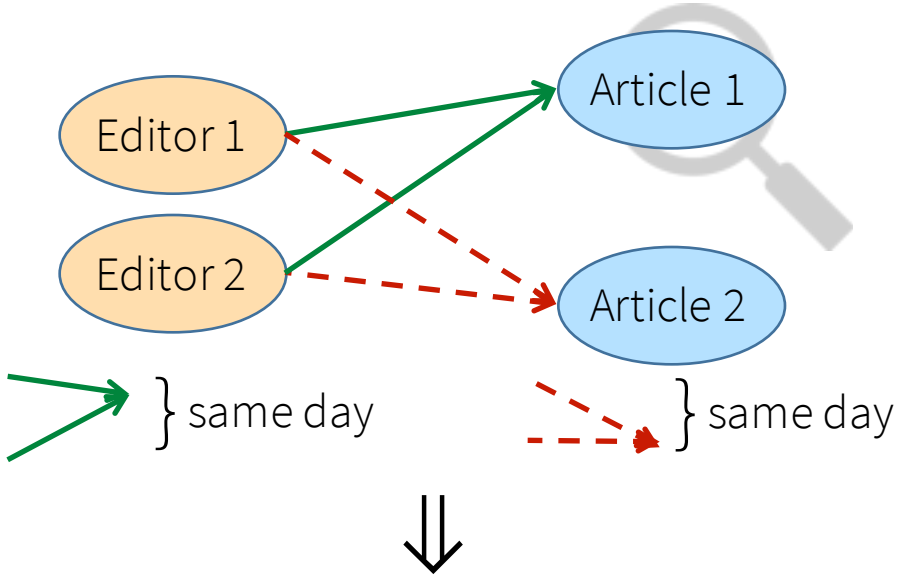


Graph Algorithms vs. Graph Queries

Graph Algorithm: PageRank



Graph Query: Wikipedia Collaborators



Editor 1	Editor 2	Article 1	Article 2

Graph Algorithms vs. Graph Queries

Graph Algorithm: PageRank

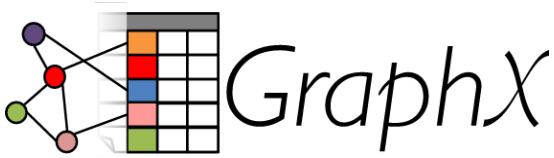
```
// Iterate until convergence
wikipedia.pregel(
  sendMsg = { e =>
    e.sendToDst(e.srcRank * e.weight)
  },
  mergeMsg = _ + _,
  vprog = { (id, oldRank, msgSum) =>
    0.15 + 0.85 * msgSum
  })
```

Graph Query: Wikipedia Collaborators

```
wikipedia.find(
  "(u1)-[e11]->(article1);
   (u2)-[e21]->(article1);
   (u1)-[e12]->(article2);
   (u2)-[e22]->(article2)")
 .select(
   "*",
   "e11.date - e21.date".as("d1"),
   "e12.date - e22.date".as("d2"))
 .sort("d1 + d2".desc).take(10)
```

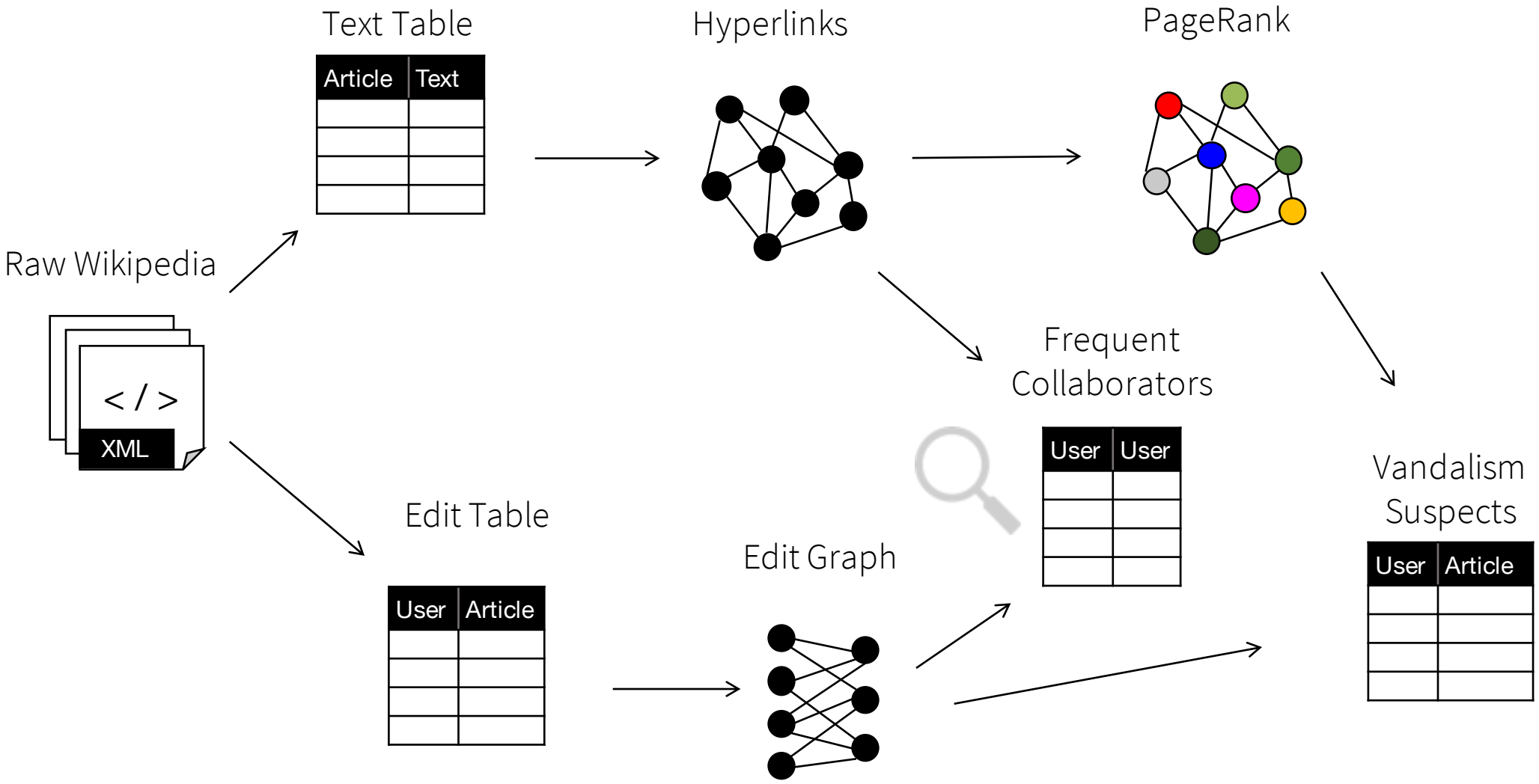
Separate Systems

Graph Algorithms

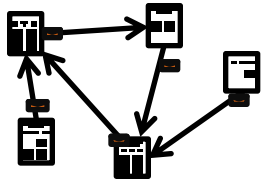


Graph Queries

Problem: Mixed Graph Analysis

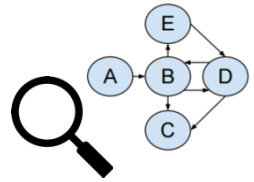


Solution: GraphFrames



Graph Algorithms

Graph Queries



GraphFrames API

Pattern Query Optimizer

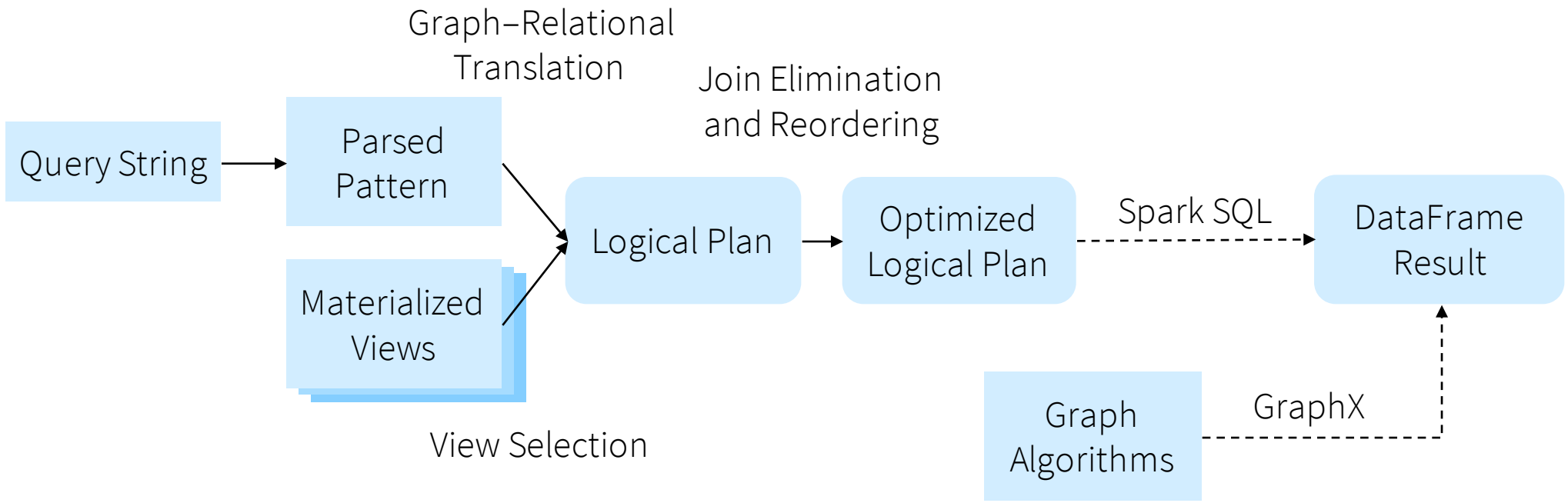
Spark SQL

GraphFrames API

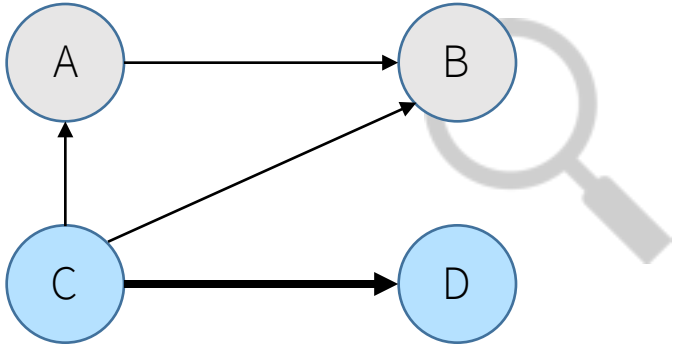
- Unifies graph algorithms, graph queries, and relational operations (DataFrames)
- Designed for interactive use
- Available in Scala, Java, and Python

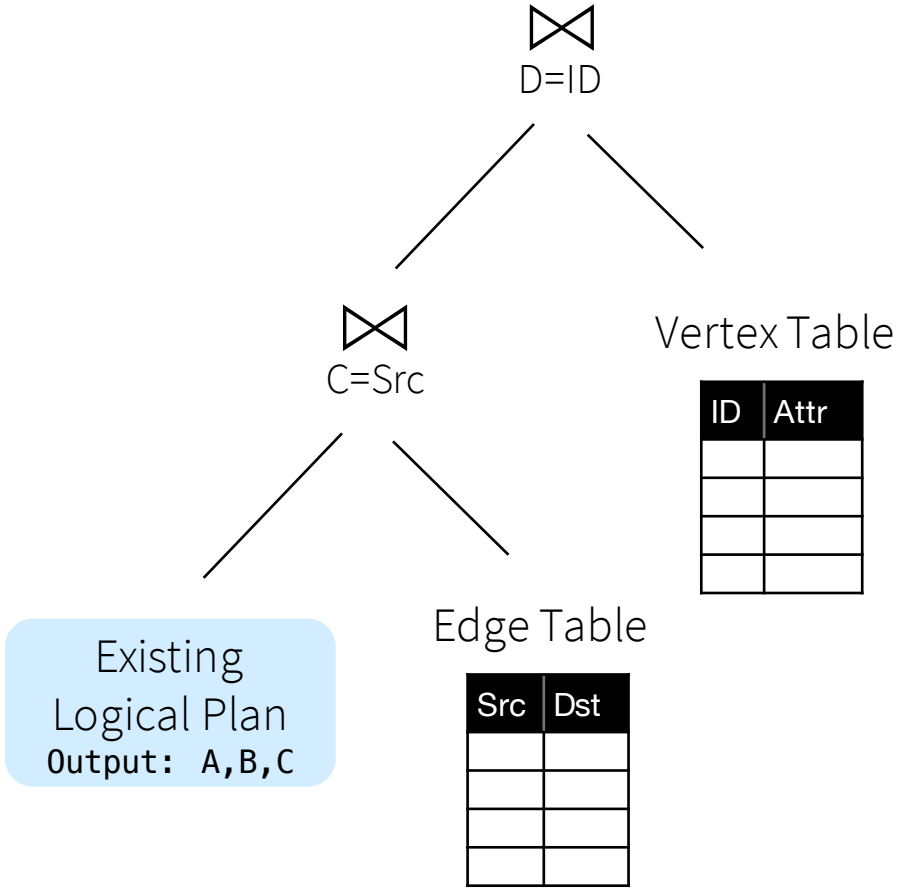
```
class GraphFrame {
  def vertices: DataFrame
  def edges: DataFrame

  def find(pattern: String): DataFrame
  def registerView(pattern: String, df: DataFrame): Unit

  def degrees(): DataFrame
  def pageRank(): GraphFrame
  def connectedComponents(): GraphFrame
  ...
}
```

Implementation



Graph–Relational Translation





Join Elimination

Edges

Src	Dst
1	2
1	3
2	3
2	5

Vertices

ID	Attr
1	A
2	B
3	C
4	D

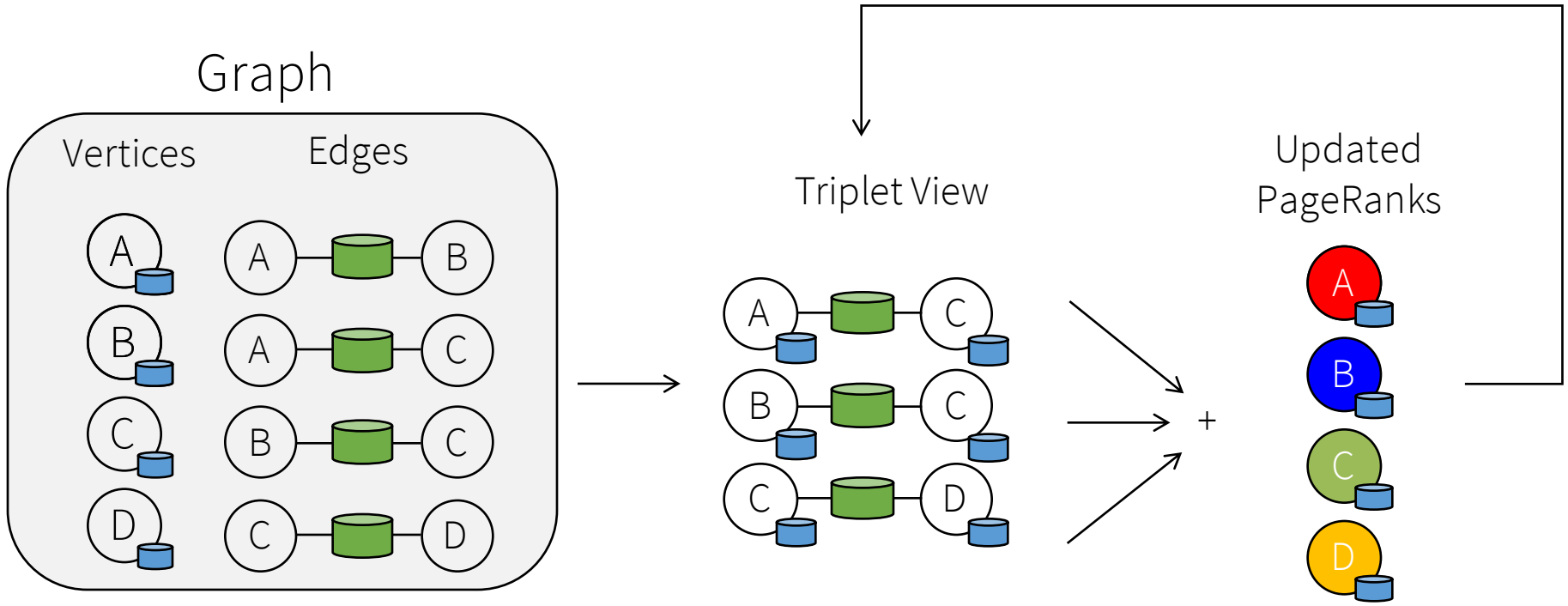
Unnecessary join

```
SELECT src, dst
FROM edges INNER JOIN vertices ON src = id;
```

can be eliminated if tables satisfy referential integrity, simplifying graph–relational translation:

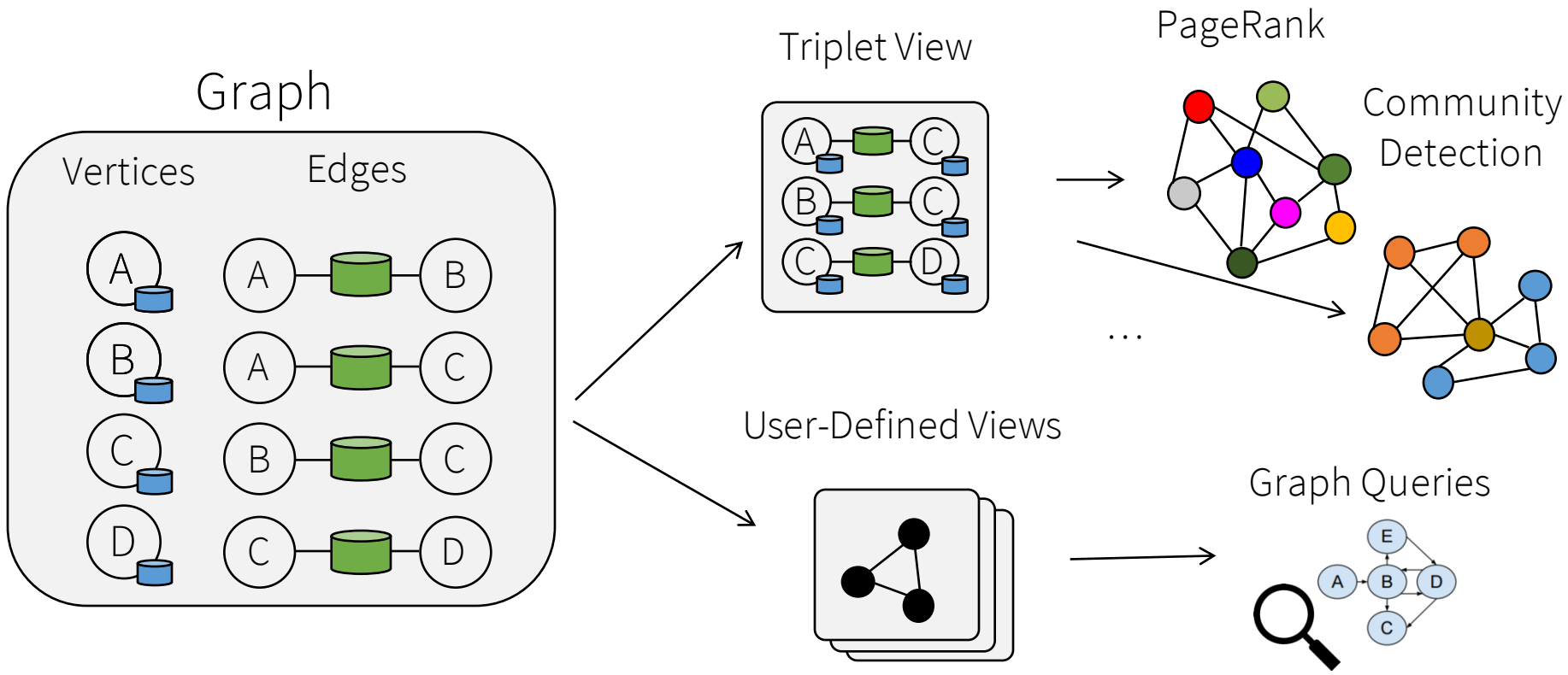
```
SELECT src, dst FROM edges;
```

Materialized View Selection



GraphX: Triplet view enabled efficient message-passing algorithms

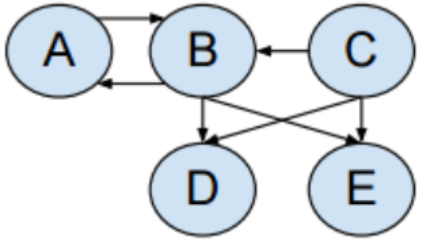
Materialized View Selection



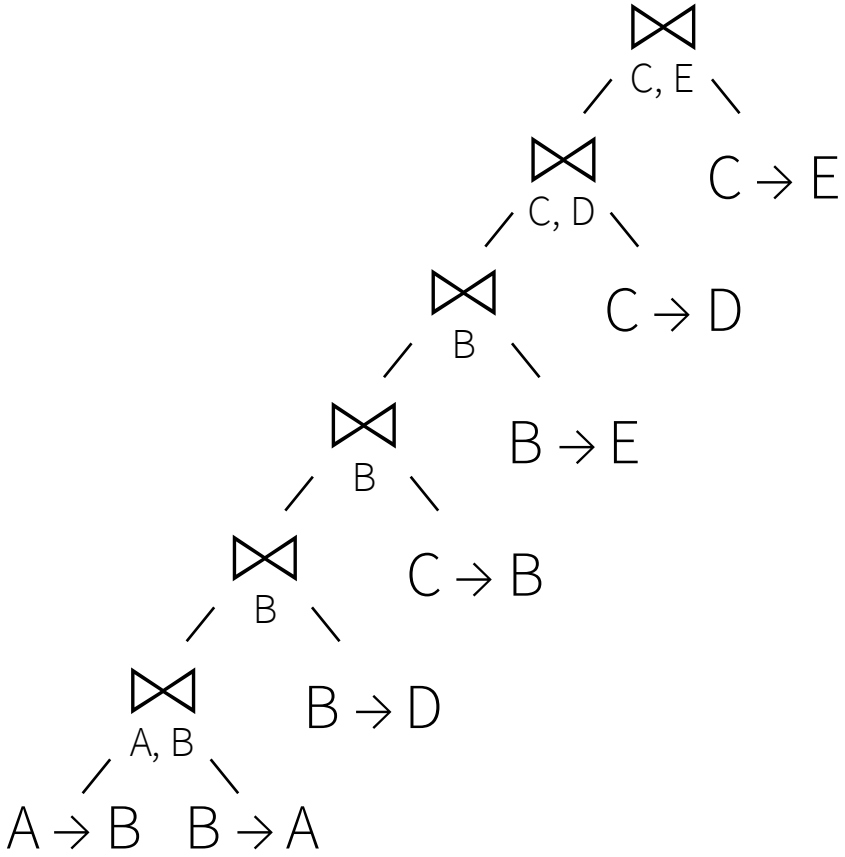
GraphFrames: User-defined views enable efficient graph queries

Join Reordering

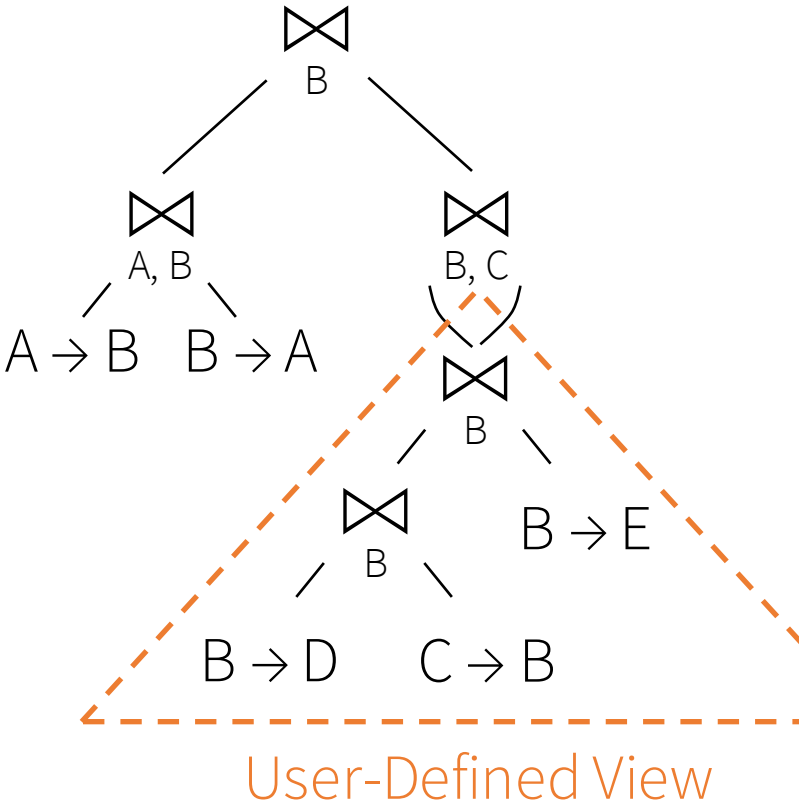
Example Query



Left-Deep Plan



Bushy Plan



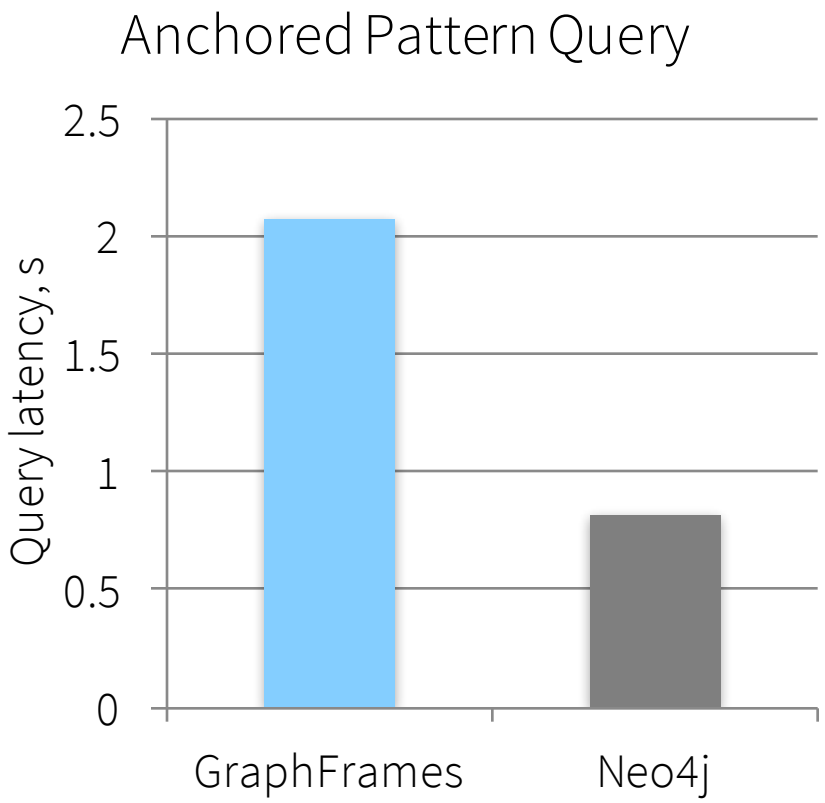
Query Planning Algorithm

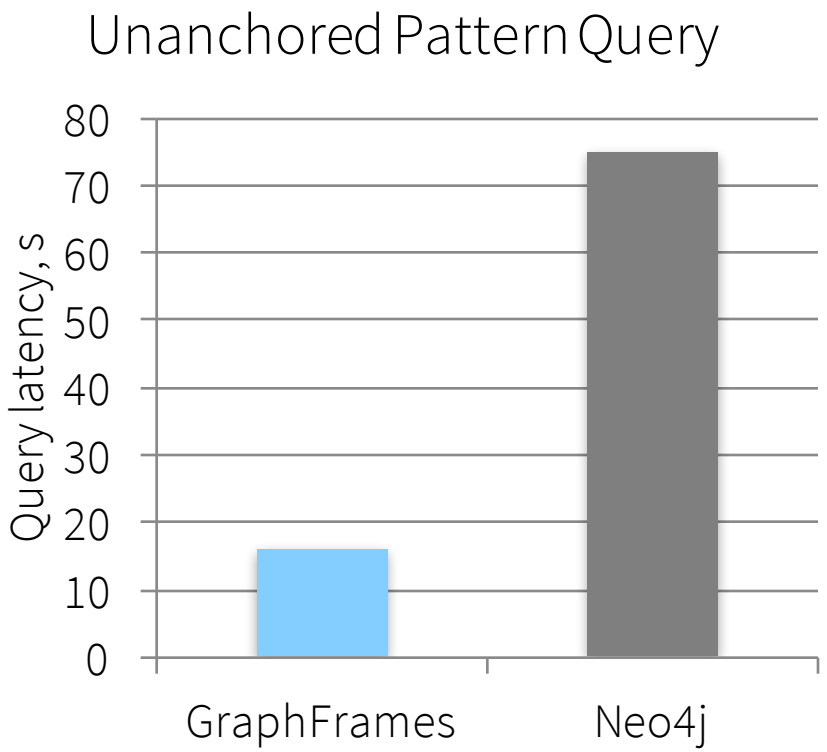
Dynamic programming algorithm based on:
J. Huang, K. Venkatraman, and D.J. Abadi. *Query optimization of distributed pattern matching.* In ICDE 2014.

1. Considers all left-deep plans, and a subset of bushy plans
   - Bushy plans to explore are chosen using layered-DAG and cycle-detection heuristics
2. Considers using each view that is exactly equivalent to a plan subtree
   - Result: Selects the largest of multiple hierarchically contained views

Evaluation

Faster than Neo4j for *unanchored* pattern queries

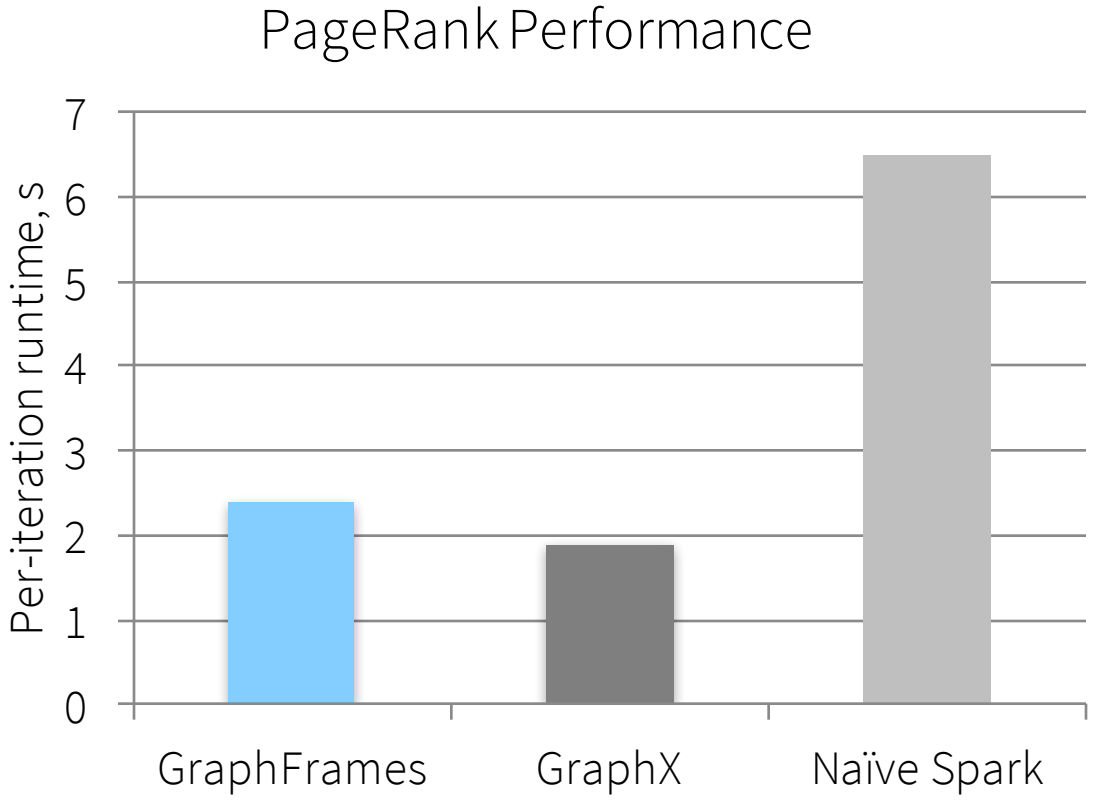




Triangle query on 1M edge subgraph of web-Google. Each system configured to use a single core.

Evaluation

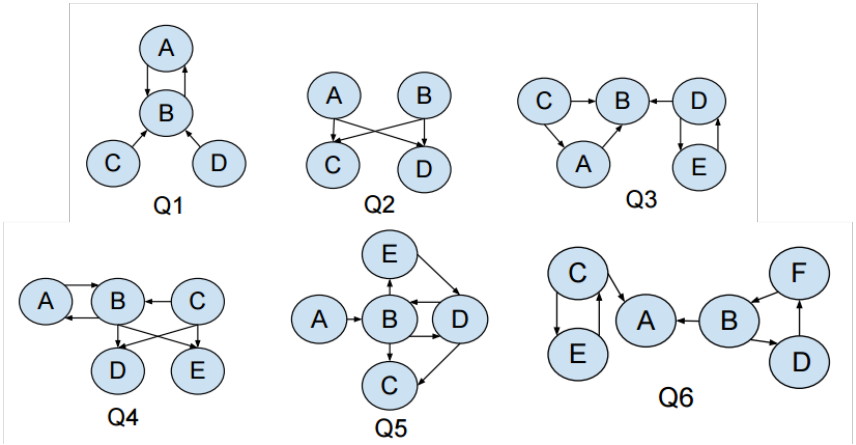
Approaches performance of GraphX for graph algorithms using Spark SQL whole-stage code generation



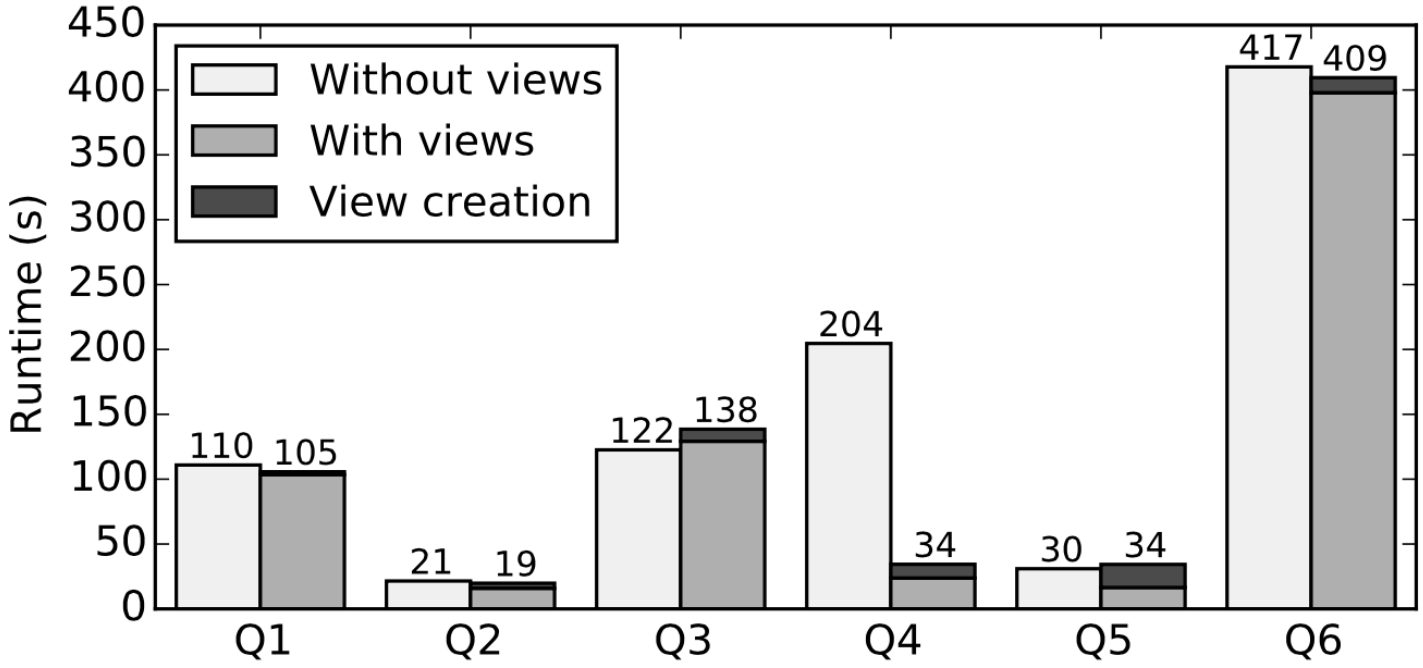
Per-iteration performance on web-Google, single 8-core machine. Naïve Spark uses Scala RDD API.

Evaluation

Registering the right views can greatly improve performance for some queries



View	Query	Size in Google graph
2-cycle	(a)->(b)->(a)	1,565,976
V	(c)<-(a)->(b)	67,833,471
Triangle	(a)<-(b)->(c)->(a)	28,198,954
3-cycle	(a)->(b)->(c)->(a)	11,669,313



Workload: J. Huang, K. Venkatraman, and D.J. Abadi. Query optimization of distributed pattern matching. In ICDE 2014.

Future Work

- Suggest views automatically
- Exploit attribute-based partitioning in optimizer
- Code generation for single node

Try It Out!

Released as a Spark Package at:

https://github.com/graphframes/graphframes

Thanks to Joseph Bradley, Xiangrui Meng, and Timothy Hunter.

ankurd@eecs.berkeley.edu