GraphiQL

Graph Intuitive Query Language for Relational Databases

at

IEEE BigData 2014

Talking on

Alekh Jindal

collaborate

Amol Deshpande

Supervisors

Sam Madden

Mike Stonebraker

work

University
of Maryland

sabbatical

work

work

MIT

Relational Database

GraphiQL

Graph Intuitive Query Language for Relational Databases

at

IEEE BigData 2014

Talking on

Alekh Jindal

collaborate

Amol Deshpande

work

University of Maryland

sabbatical

Supervisors

Sam Madden

work

Mike Stonebraker

work

MIT

Expensive!

Expensive!

Graph Analysis =

Graph Algorithms

+

Store
Extract
Preprocess
Update
Failover
Postprocess

Graph Analysis =

Relational Database

Graph Algorithms

+

Relational Database

Store
Extract
Preprocess
Update
Failover
Postprocess

“Counting Triangles with Vertica”

“Scalable Social Graph Analytics Using the Vertica Analytic Platform,”

“Graph Analysis: Do We Have to Reinvent the Wheel?”

“Query Optimization of Distributed Pattern Matching,”

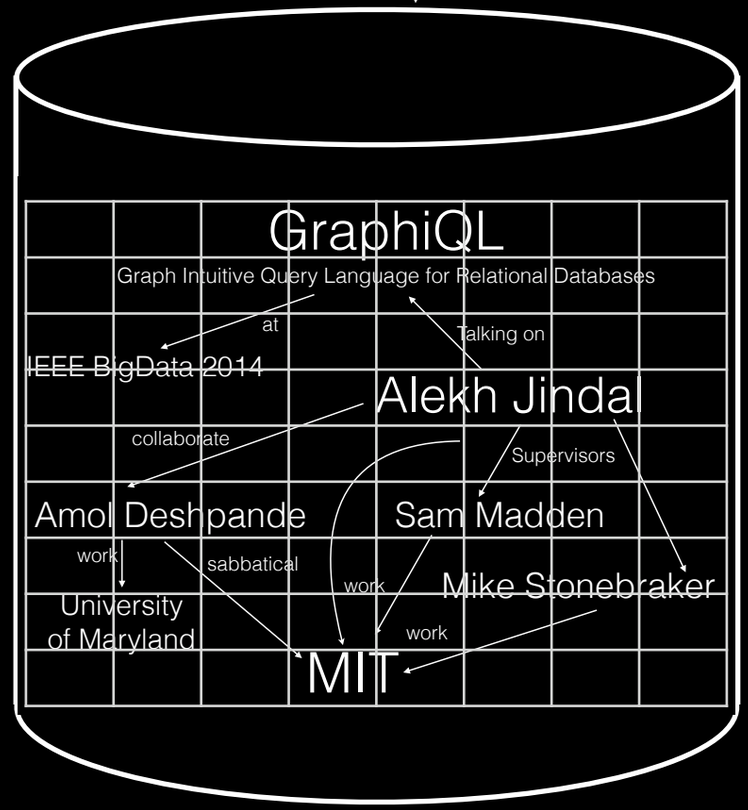
“GraphX: A Resilient Distributed Graph System on Spark,”

“Vertexica: Your Relational Friend for Graph Analytics!”

Problem !

GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT

SQL


GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT

SELECT

COUNT

UPDATE

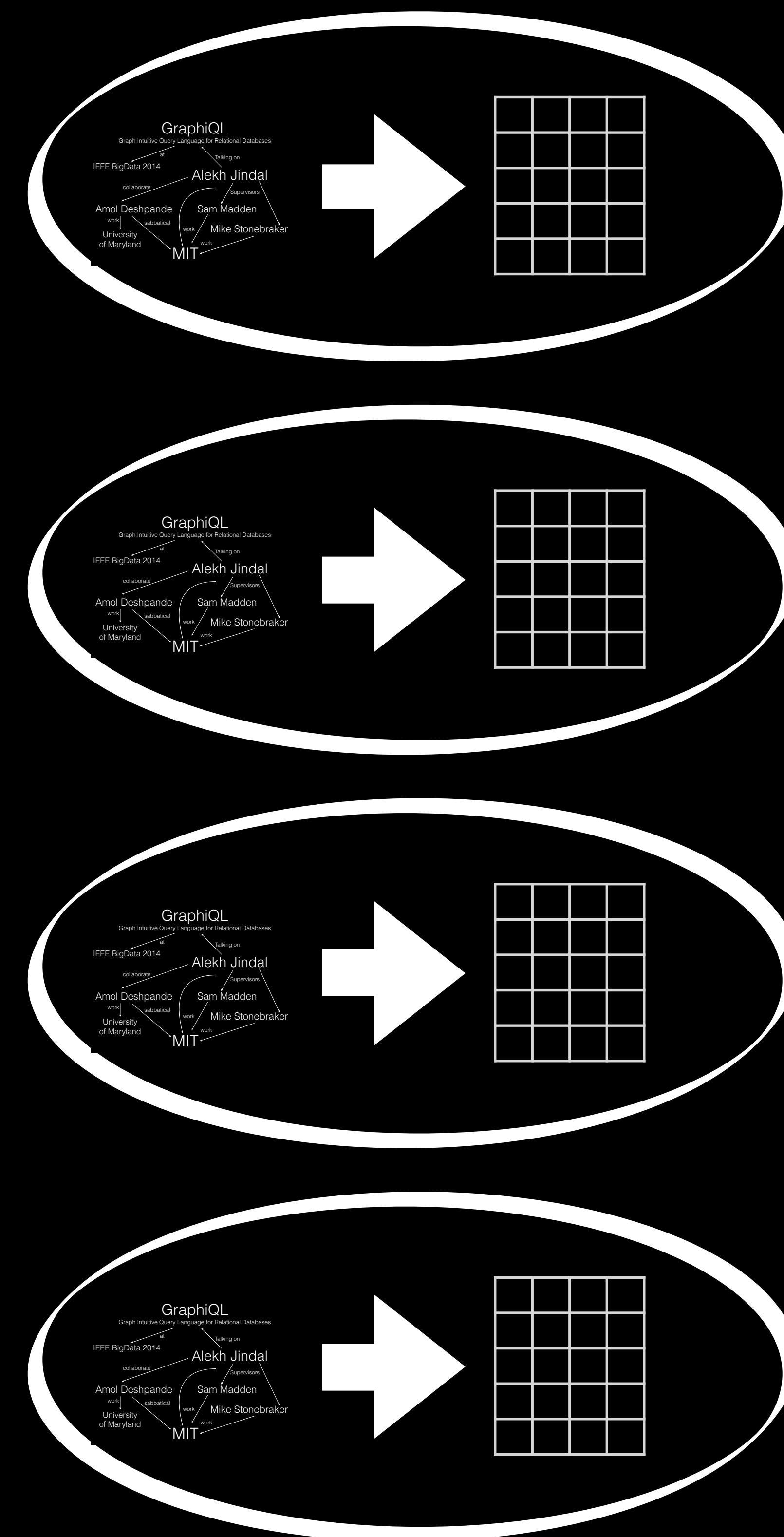
FROM

GROUP BY

SUM

WHERE

Redundant Effort

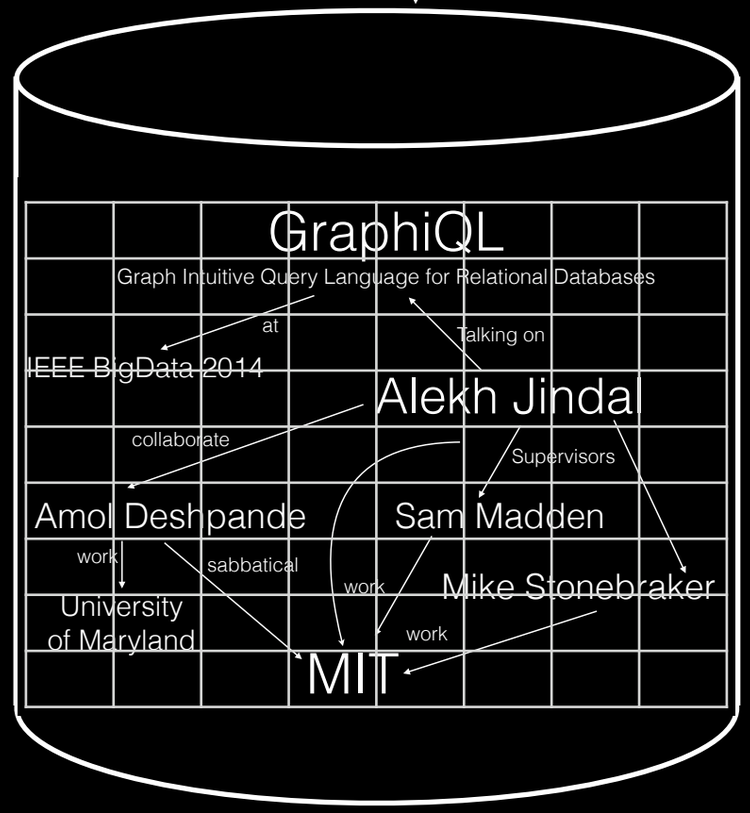


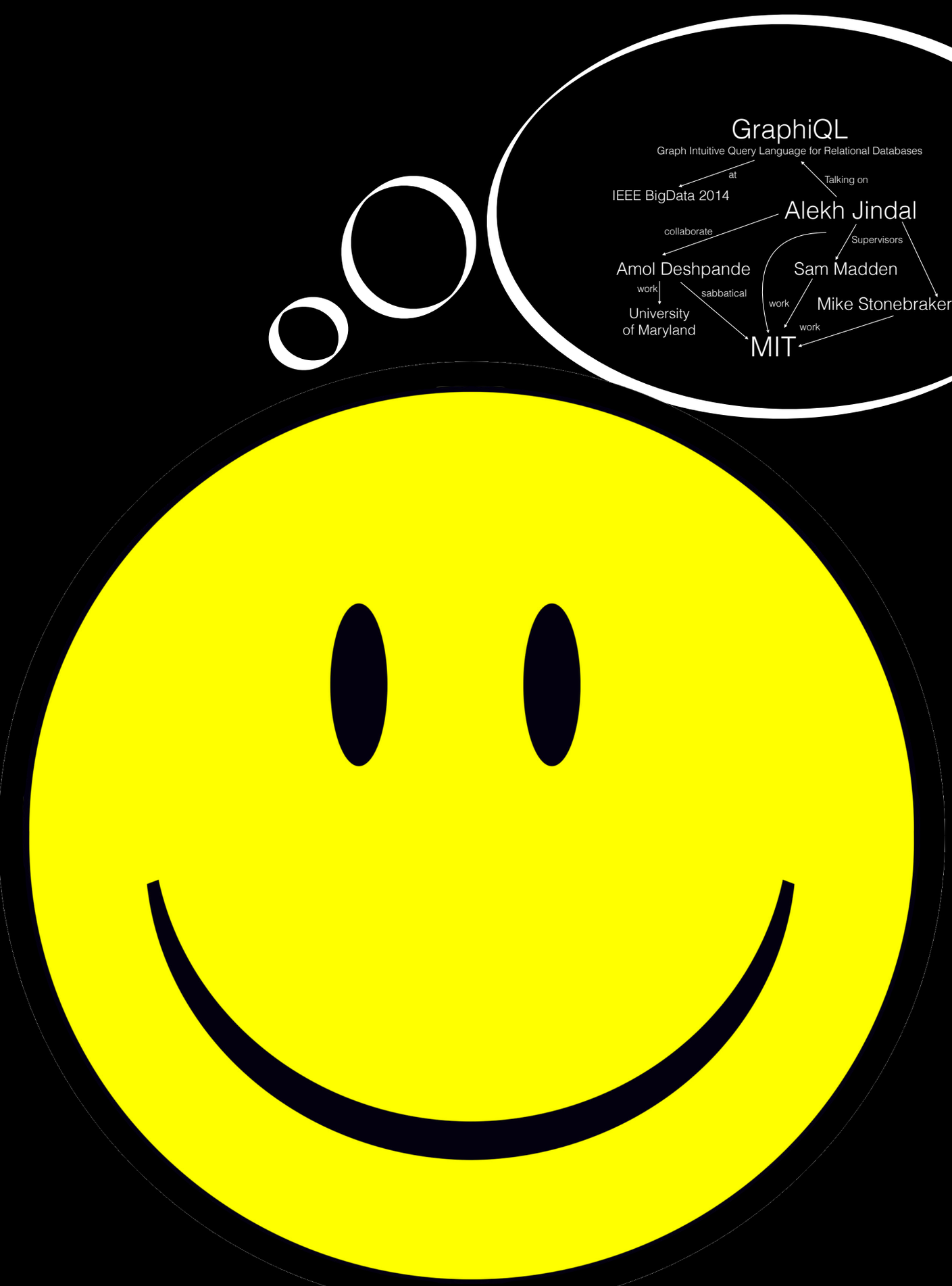
Optimizations?

GraphiQL

GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT

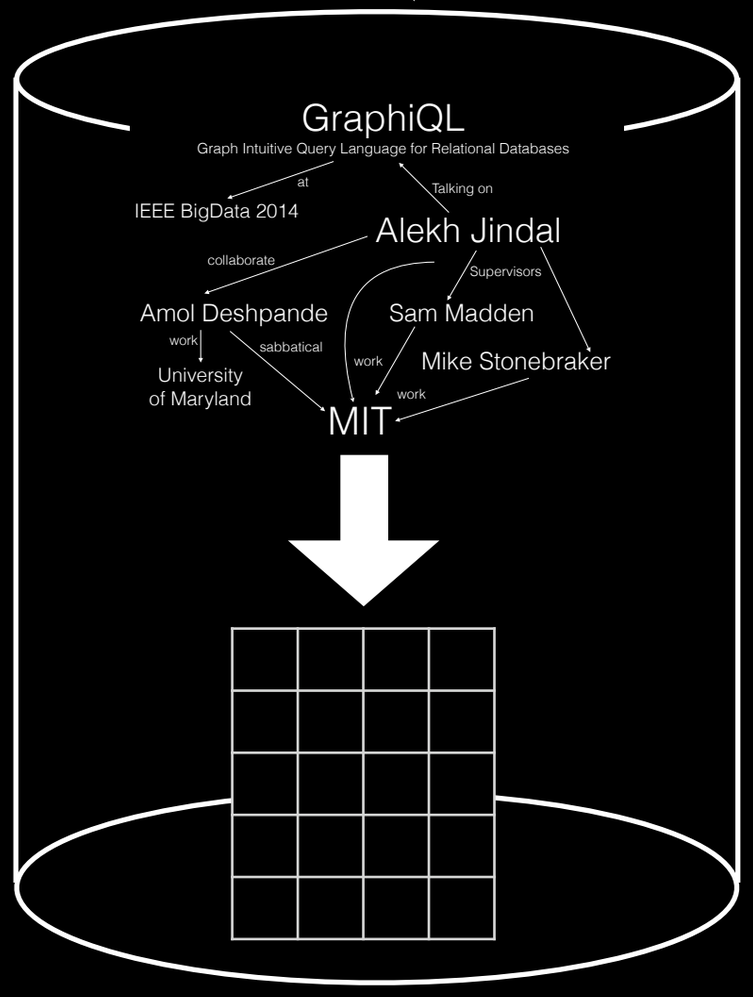
SQL


GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT



GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT

GraphiQL


GraphiQL
Graph Intuitive Query Language for Relational Databases
at
Talking on
IEEE BigData 2014
Alekh Jindal
collaborate
Supervisors
Amol Deshpande
Sam Madden
work
sabbatical
work
Mike Stonebraker
University
of Maryland
work
MIT

Key Features

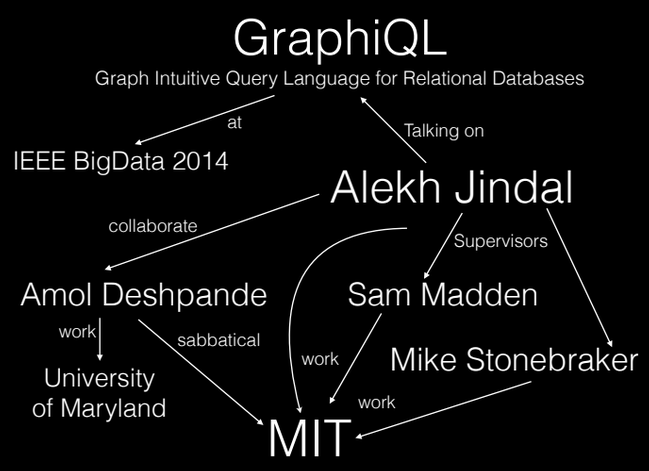
- Graph view of relational data; the system takes care of mapping to the relational world
- Inspired from PigLatin: right balance between declarative and procedural style language
- Key graph constructs: looping, recursion, neighborhood access
- Compiles to optimized SQL

Graph Table

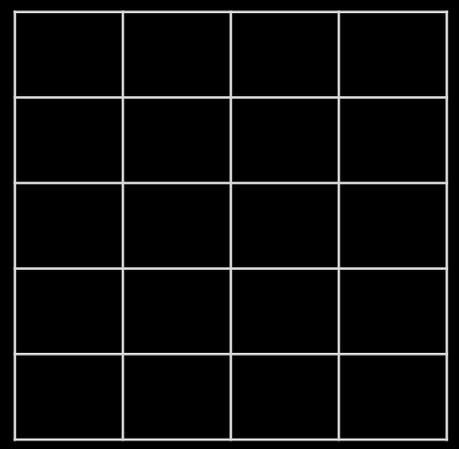
GraphiQL

SQL

Graph Table



Relational Table



Graph Table

node8		
node9		
edge7		
edge8		
edge9		

incoming

Graph Elements	id	weight	type
node1			
node2			
edge1			
edge2			
edge3			
node3			
node4			
node5			

outgoing

node6		
node7		
edge4		
edge5		
edge6		

Graph Table Definition

- Create

```
CREATE GRAPHTABLE g AS
    NODE (p1,p2,..)
    EDGE (q1,q2,..)
```

- Load

```
LOAD g AS
    NODE FROM graph_nodes DELIMITER d
    EDGE FROM graph_edges DELIMITER d
```

- Drop

```
DROP GRAPHTABLE g
```

Graph Table Manipulation

- Iterate

```
FOREACH element in g
  [WHILE condition]
```

- Filter

```
g' = g(k1=v1,k2=v2,…,kn=vn)
```

- Retrieve

```
GET expr1,expr2,…,exprn
[WHERE condition]
```

- Update

```
SET variable TO expr
[WHERE condition]
```

- Aggregate

```
SUM, COUNT, MIN, MAX, AVG
```

Nested Manipulation

outer \ *inner*	Iterate	Aggregate	Retrieve	Update
Iterate	✓	✓	✓	✓
Aggregate	✓	✓	✓	
Retrieve	✓	✓		
Update	✓	✓		

Example 1: PageRank

```
FOREACH n IN g(type=N)
  SET n.pr TO new_pr
```

Example 1: PageRank

```
FOREACH n IN g(type=N)
  SET n.pr TO 0.15/num_nodes + 0.85*SUM(pr_neighbors)
```

Example 1: PageRank

```
FOREACH n IN g(type=N)
  SET n.pr TO 0.15/num_nodes + 0.85*SUM(
    FOREACH n’ IN n.in(type=N)
      GET pr_n’
  )
```

Example 1: PageRank

```
FOREACH n IN g(type=N)
  SET n.pr TO 0.15/num_nodes + 0.85*SUM(
    FOREACH n' IN n.in(type=N)
      GET n'.pr/COUNT(n'.out(type=N))
  )
```

Example 1: PageRank

```
FOREACH iterations IN [1:10]
  FOREACH n IN g(type=N)
SET n.pr TO 0.15/num_nodes + 0.85*SUM(
  FOREACH n’ IN n.in(type=N)
    GET n’.pr/COUNT(n’.out(type=N))
)
```

Example 1: PageRank

Looping

```
FOREACH iterations IN [1:10]
  FOREACH n IN g(type=N)
    SET n.pr TO 0.15/num_nodes + 0.85*SUM(
      FOREACH n' IN n.in(type=N)
        GET n'.pr/COUNT(n'.out(type=N))
    )
```

Nested Manipulations

Reason about graph

Neighborhood Access

Example 2: SSSP

```
FOREACH n IN g(type=N)
  SET n.dist TO min_dist
```

Example 2: SSSP

```
FOREACH n IN g(type=N)
  SET n.dist TO MIN(n.in(type=N).dist)+1 AS dist'
    WHERE dist' < n.dist
```

Example 2: SSSP

```
WHILE updates > 0
  FOREACH n IN g(type=N)
    updates =
SET n.dist TO MIN(n.in(type=N).dist)+1 AS dist’
  WHERE dist’ < n.dist
```

Example 2: SSSP

```
SET g(type=N).dist TO inf
SET g(type=N,id=start).dist TO 0
WHILE updates > 0
  FOREACH n IN g(type=N)
    updates =
    SET n.dist TO MIN(n.in(type=N).dist)+1 AS dist’
      WHERE dist’ < n.dist
```

GraphiQL Compiler

- Graph Table manipulations to relational operators:
  - filter ↦ selection predicates
  - iterate ↦ driver loop
  - retrieve ↦ projections
  - update ↦ update in place
  - aggregate ↦ group-by aggregate

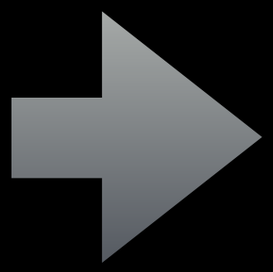
- Graph Tables to relational tables:
  - mapping

GraphiQL Compiler

```
g(type=N)   ↦   N

g(type=E)   ↦   E

g(type=N).out(type=E)   ↦   N ⋈ E

g(type=E).out(type=E)   ↦   E ⋈ E

g(type=N).out(type=N)   ↦   N ⋈ E ⋈ N

g.out.in = g.in

g.in.out = g.out
```

Example: SSSP

```
SET g(type=N).dist TO inf
SET g(type=N,id=start).dist TO 0
WHILE updates > 0
  FOREACH n IN g(type=N)
    updates =
    SET n.dist TO
    MIN(n.in(type=N).dist)+1 AS dist'
      WHERE dist' < n.dist
```



$$
\begin{aligned}
&l^{\text{updateCount}>0}\ (\\
&\quad n.\text{dist} \leftarrow \sigma_{n.\text{dist}>\text{dist}'}\ (\\
&\qquad \gamma_{\min(n'.\text{dist})+1}\ (\\
&\qquad\quad \Gamma_{n.\text{id}}\ (\\
&\qquad\qquad N \bowtie E \bowtie N'\\
&\qquad\quad)\\
&\qquad)\\
&\quad)\\
&)
\end{aligned}
$$

GraphiQL Optimizations

- De-duplicating graph elements
- Selection pushdown
- Cross-product as join
- Pruning redundant joins

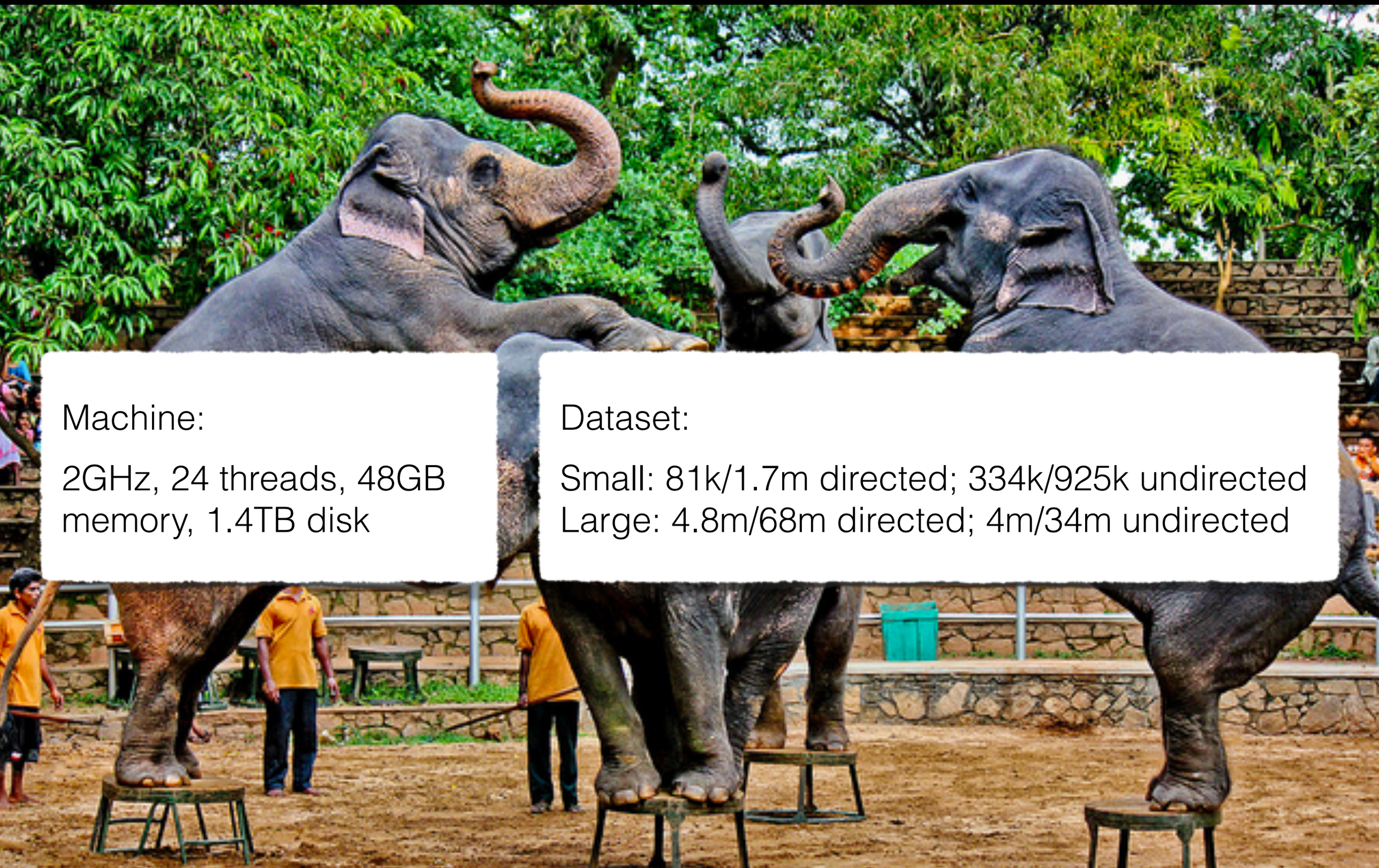
Performance

Performance

Machine:

2GHz, 24 threads, 48GB memory, 1.4TB disk

Performance



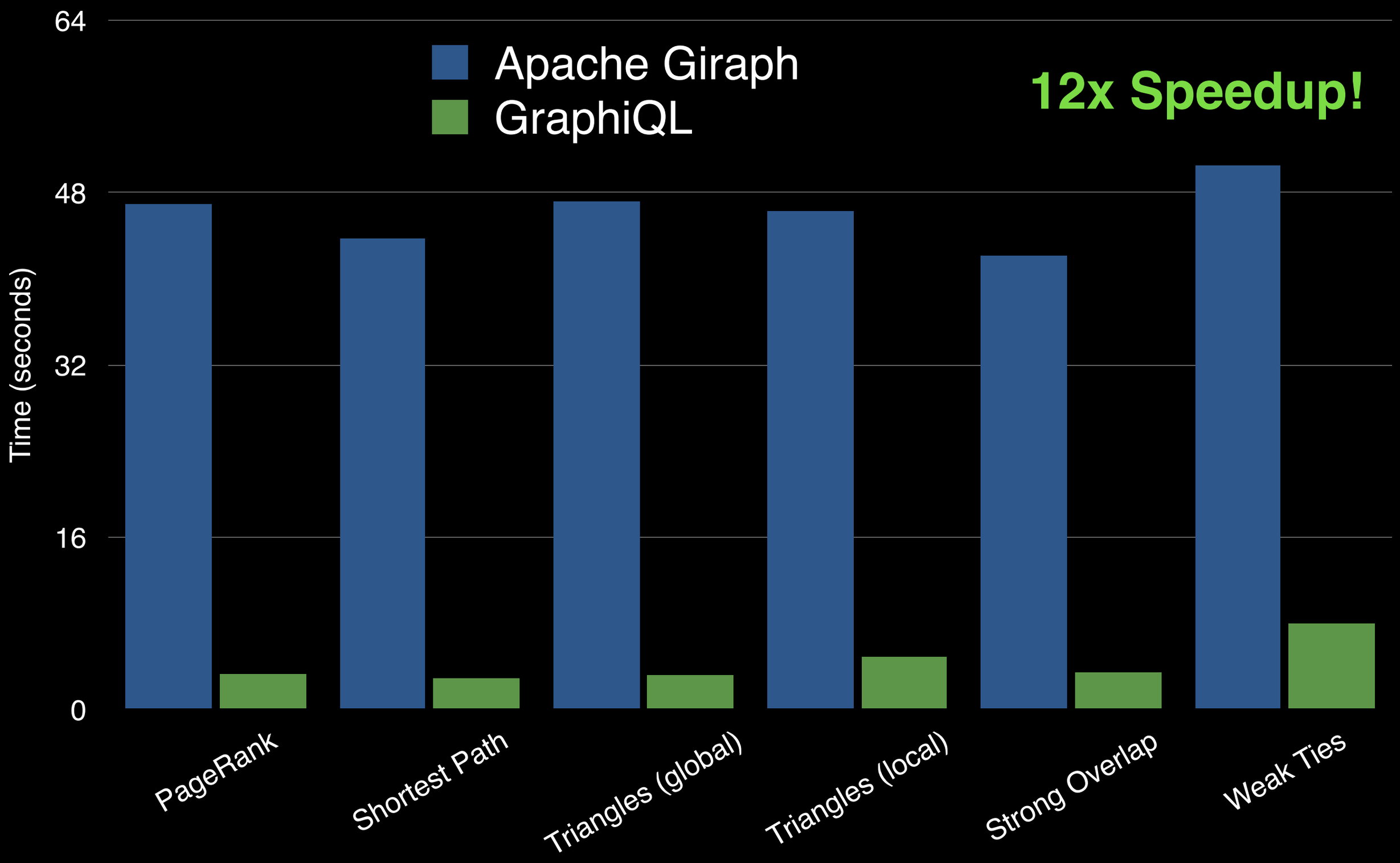
Machine:

2GHz, 24 threads, 48GB memory, 1.4TB disk

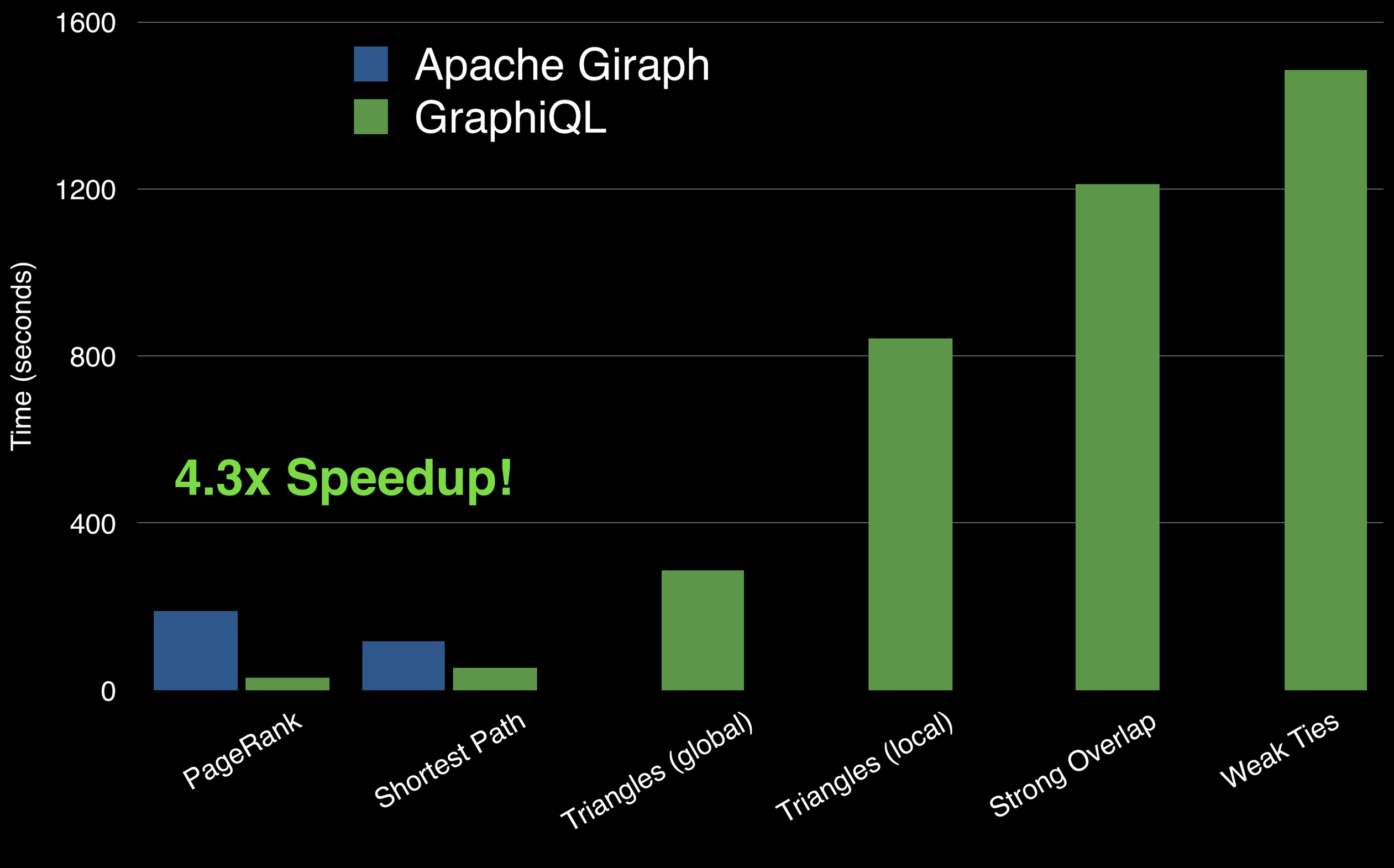
Dataset:

Small: 81k/1.7m directed; 334k/925k undirected
Large: 4.8m/68m directed; 4m/34m undirected

Performance - small graph



Performance - large graph



Summary

- Several real world graph analysis are better off in relational databases
- We need both the graph as well as relational view of data
- GraphiQL introduces Graph Tables to allows users to think in terms of graphs
- Graph Table supports recursive association, nested manipulations, and SQL compilation
- GraphiQL allows users to easily write a variety of graph analysis

Thanks!

Other Languages

Imperative languages: e.g. Green Marl
XPath: e.g. Cypher, Gremlin
Datalog: e.g. Socialite
SPARQL: Teradata blog
Procedural language: e.g. Vertex-centric