ORACLE

aDFS: An Almost Depth-First-Search Distributed Graph-Querying System

Vasileios Trigonakis, Jean-Pierre Lozi, Tomas Faltin, Nicholas P. Roth, Iraklis Psaroudakis, Arnaud Delamare, Vlad Haprian, Calin Iorgulescu, Petr Koupy, Jinsu Lee, Sungpack Hong, Hassan Chafi

Oracle Labs

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

GraalVM Native Image technology (including SubstrateVM) is early adopter technology. It is available only under an early adopter license and remains subject to potentially significant further changes, compatibility testing and certification.

Graphs and Graph Queries

- **Graphs are everywhere!**
  - Oracle Labs PGX.Distributed
- **Graph queries**
  - Allow flexible exploration of graphs
  - Similar to SQL for RDBMs
  - PGX.D's aDFS uses PGQL (pgql-lang.org)

Like SQL but adds **MATCH**

```
-- Info of authors who like each other and have < 10 years of age difference
SELECT a1.name, a2.name, a1.country = a2.country,
    ABS(a1.salary - a2.salary) AS salary_diff
MATCH (a1:author) -[:likes]-> (a2:author) -[:likes]-> (a1)
WHERE ABS(a1.age - a2.age) < 10
ORDER BY salary_diff DESC
```

Any user expression in projections and filters

Requires **homomorphic matching** and returns **all result permutations**

Complexities in Graph-Query Execution

- Limited locality (especially in a distributed system)
- Intermediate (and final) result explosion

Twitter graph

```
SELECT COUNT(*) MATCH (a)->()
+--------------------+
|     COUNT(*)       |
+--------------------+
| 1,468,365,182      |
+--------------------+
```
1 hop

```
SELECT COUNT(*) MATCH (a)->()->()
+----------------------+
|     COUNT(*)         |
+----------------------+
| 9,324,563,362,739 |
+----------------------+
```
2 hops

spoiler!
PGX.D aDFS
8 machines
~20 minutes
~8B matches/s

```
-- Info of authors who like each other and have < 10 years of age difference
SELECT a1.name, a2.name, a1.country = a2.country,
    ABS(a1.salary - a2.salary) AS salary_diff
MATCH (a1:author) -[:likes]-> (a2:author) -[:likes]-> (a1)
WHERE ABS(a1.age - a2.age) < 10
ORDER BY salary_diff DESC
```

Any user expression in projections and filters

Requires **homomorphic matching** and returns **all result permutations**

Agenda

1. Introduction / Motivation
2. aDFS Design
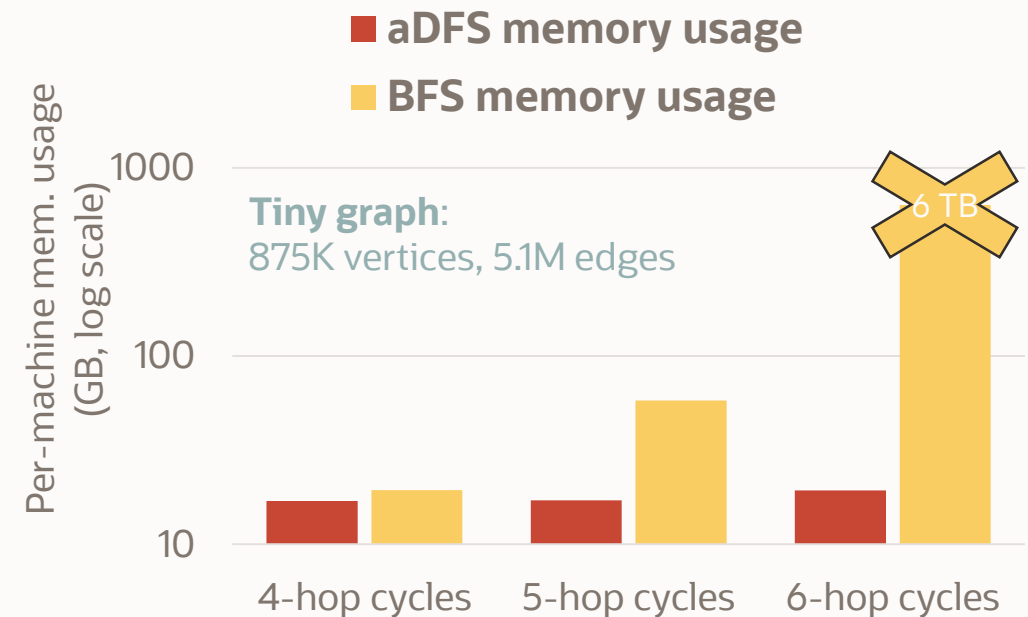3. Evaluation
4. Conclusions

aDFS Design Principles

1. Asynchronous operation

- Workers operate independently
  - on traversals where there is work
- Workers buffer and forget remote traversals

→Workers do not block due to remote communication

2. (Almost) Depth-first traversal

- Eager completion of matches
- Fine-grained flow control

→ Control memory consumption

■ **aDFS memory usage**
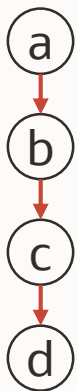■ **BFS memory usage**

Per-machine mem. usage (GB, log scale)

Tiny graph:
875K vertices, 5.1M edges

6 TB

1000

100

10

4-hop cycles   5-hop cycles   6-hop cycles

In-memory distributed execution with controllable memory usage
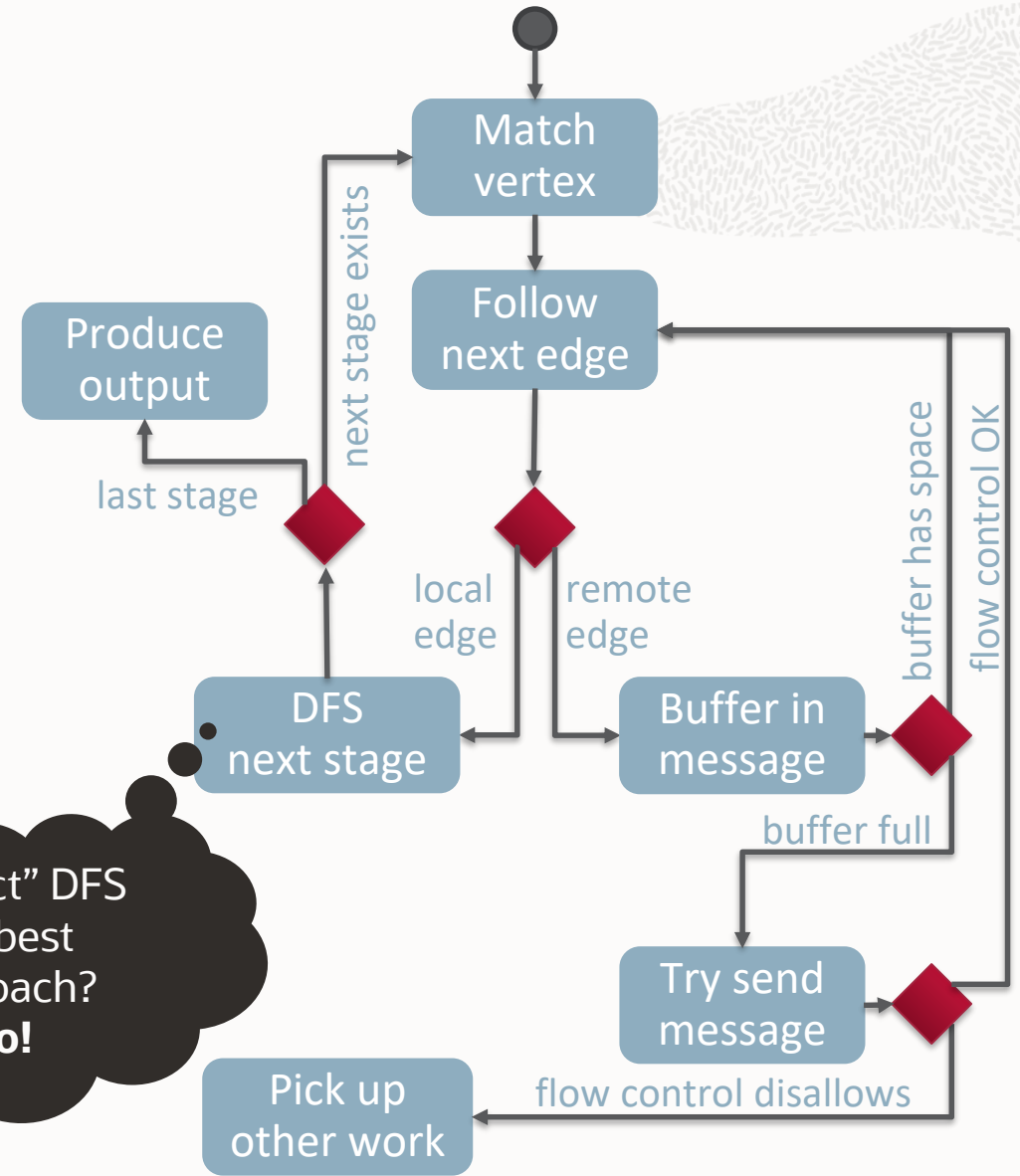
From a PGQL Query to an aDFS Execution Plan

PGQL	→	query plan	→	distributed query plan	→	distributed execution plan
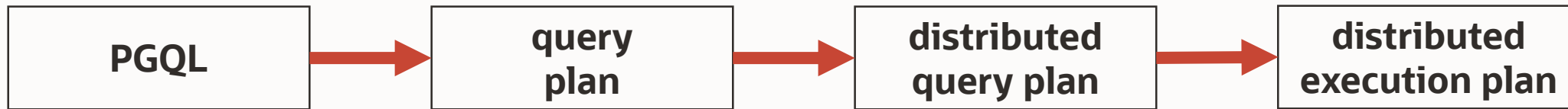
```
SELECT  ...
MATCH
  (a)->(b),
  (b)->(c),
  (c)->(d)
WHERE
  ...
```

a
↓
b
↓
c
↓
d

Stage 0 (a)	→	Stage 1 (b)	→	Stage 2 (c)	→	Stage 3 (d)

A list of stages that "know" how to
1. match a vertex
2. move to next stage

Asynchronous DFS/BFS Traversals

stage 0   stage 1   stage 2   stage 3

MATCH (a) → (b) → (c) → (d)
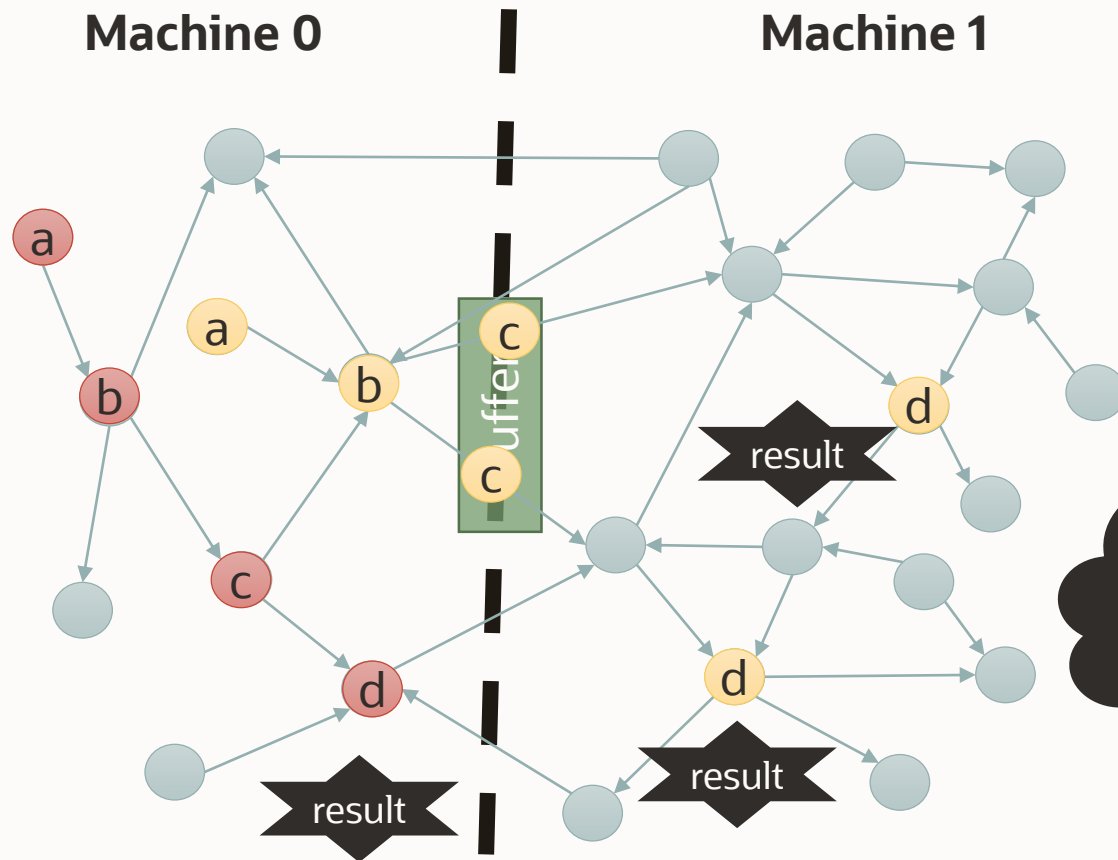
Machine 0   **Machine 1**

Is "strict" DFS the best approach? **No!**

Match vertex

Follow next edge

Produce output

next stage exists

last stage

local edge

remote edge

buffer has space

flow control OK

DFS next stage

Buffer in message

buffer full

Try send message

flow control disallows

Pick up other work

Experimental Evaluation

Schemaless Graphs and Queries

Legend: ■ aDFS ■ GraphFrames ▨ Neo4j (community edition) ■ MonetDB ■ PostgreSQL

Y-axis: Latency (s, log scale) — 100000, 1000, 10, 0.1

# Matches	52M	5.9B	106M	10B	500M	9.3T	645M	1.2T

Groups: Livejournal (Q1, Q2), Random 1B (Q1, Q2), Twitter (Q1, Q2), Webgraph-UK (Q1, Q2)
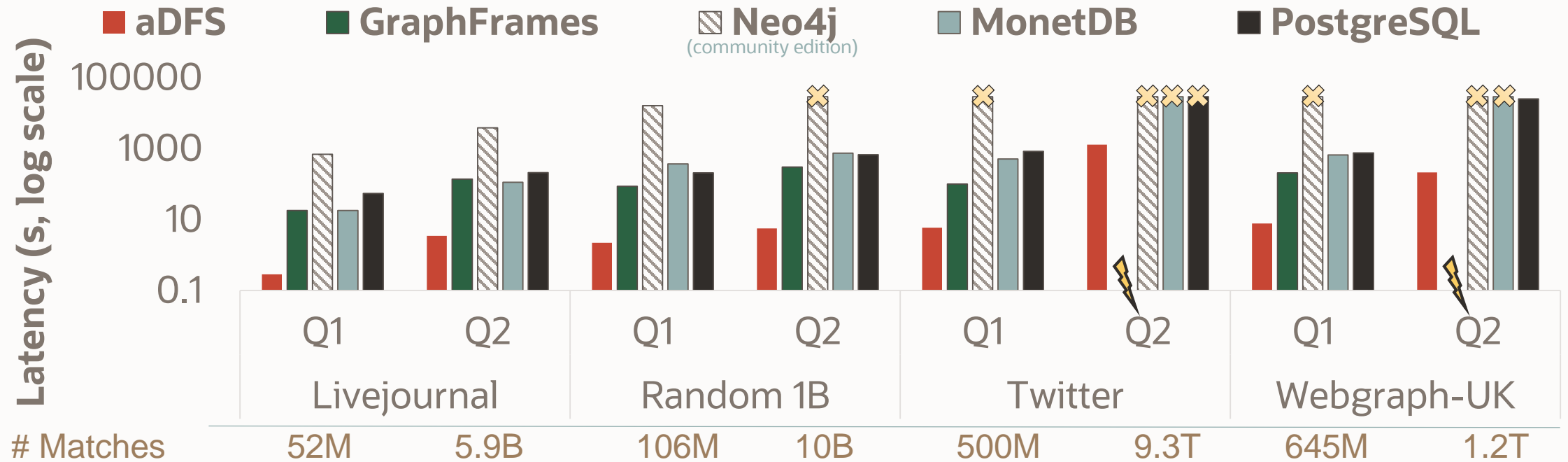
- **Q1: cycle (a)->(b)->(a)**   **Q2: 2-hops (a)->(b)->(c)**
- aDFS and GraphFrames with 8 machines / others single machine
- aDFS configured with 1GB memory per machine / others have access to whole machine memory (768 GB)
- ✖ Did not complete in 8 hours   ⚡ Hang due to out of memory

Only aDFS can handle the scale

Conclusions & Future Work

- **aDFS is a fast and scalable distributed graph querying engine**
  - Provides flexible PGQL querying
  - Combines BFS / DFS
  - Limits max memory usage

- **Current and future work** – extend PGQL capabilities, e.g.,
  - Reachability Path Queries (RPQs), e.g., (p1:person) -/:friend*/-> (p2:person)
  - Sub-queries
  - SHORTEST / CHEAPEST paths

Thank you!

Reach out to vasileios.trigonakis@oracle.com for questions, but also internship / job opportunities!