December 8th 2021

ORACLE

# Distributed Graph Processing with PGX.D

## And an overview of all the other things we do in Oracle Labs Zurich

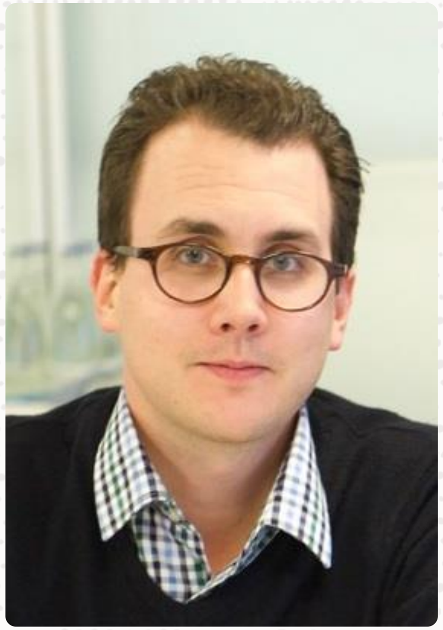
**Vasileios Trigonakis**
Principal Researcher
Oracle Labs Zurich

**Lucas Braun**
Program Manager
Oracle Labs Zurich

## Vasileios Trigonakis

- Principal Researcher @ Oracle Labs
- PhD in Computer Science from EPFL
- Started at Oracle in 2016
- Leading the PGX Distributed (PGX.D) project

in/vtrigonakis



ORACLE Database

## Lucas Braun

- Program Manager @ Oracle Labs
- BSc, MSc and PhD in Computer Science from ETH
- Started at Oracle in 2017
- Working on Oracle Database Multilingual Engine (MLE)

@lucasbraun87

/lucas-braun-277102153/

## Agenda

1 **Distributed Graph Processing with PGX.D**

- Graph Processing
- Graph Algorithms
- Graph Queries

2 A Quick Intro into Oracle Labs + Internships

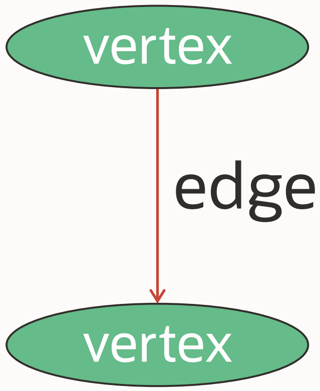
# *Graphs Are Everywhere!*

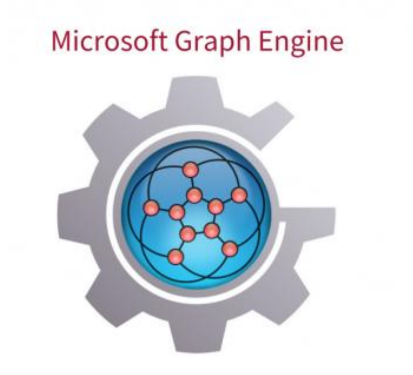
ORACLE

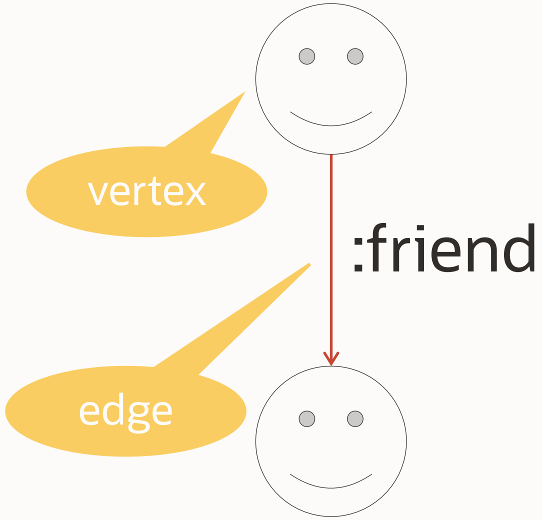
APACHE Spark

MariaDB

neo4j







Dragon

**Gartner's Top 10 Data and Analytics Technology Trends for 2020:**
**Trend No. 4: Graph Analytics**

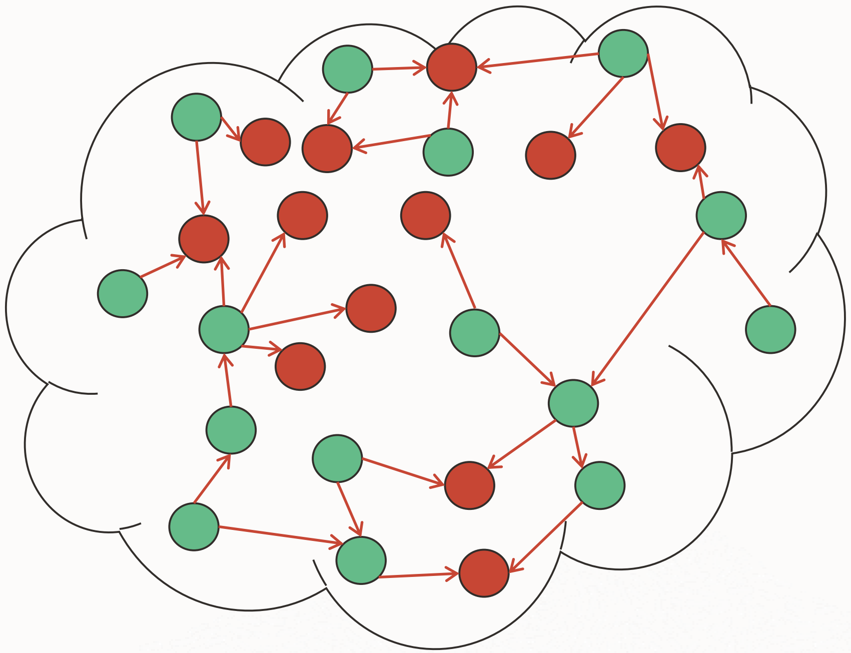
# Trend No. 4: Graph analytics

https://www.gartner.com/smarterwithgartner/gartner-top-10-data-analytics-trends/

“ Business users are asking increasingly complex questions across structured and unstructured data, often blending data from multiple applications, and increasingly, external data. Analyzing this level of data complexity at scale is not practical, or in some cases possible, using traditional query tools or query languages such as SQL.

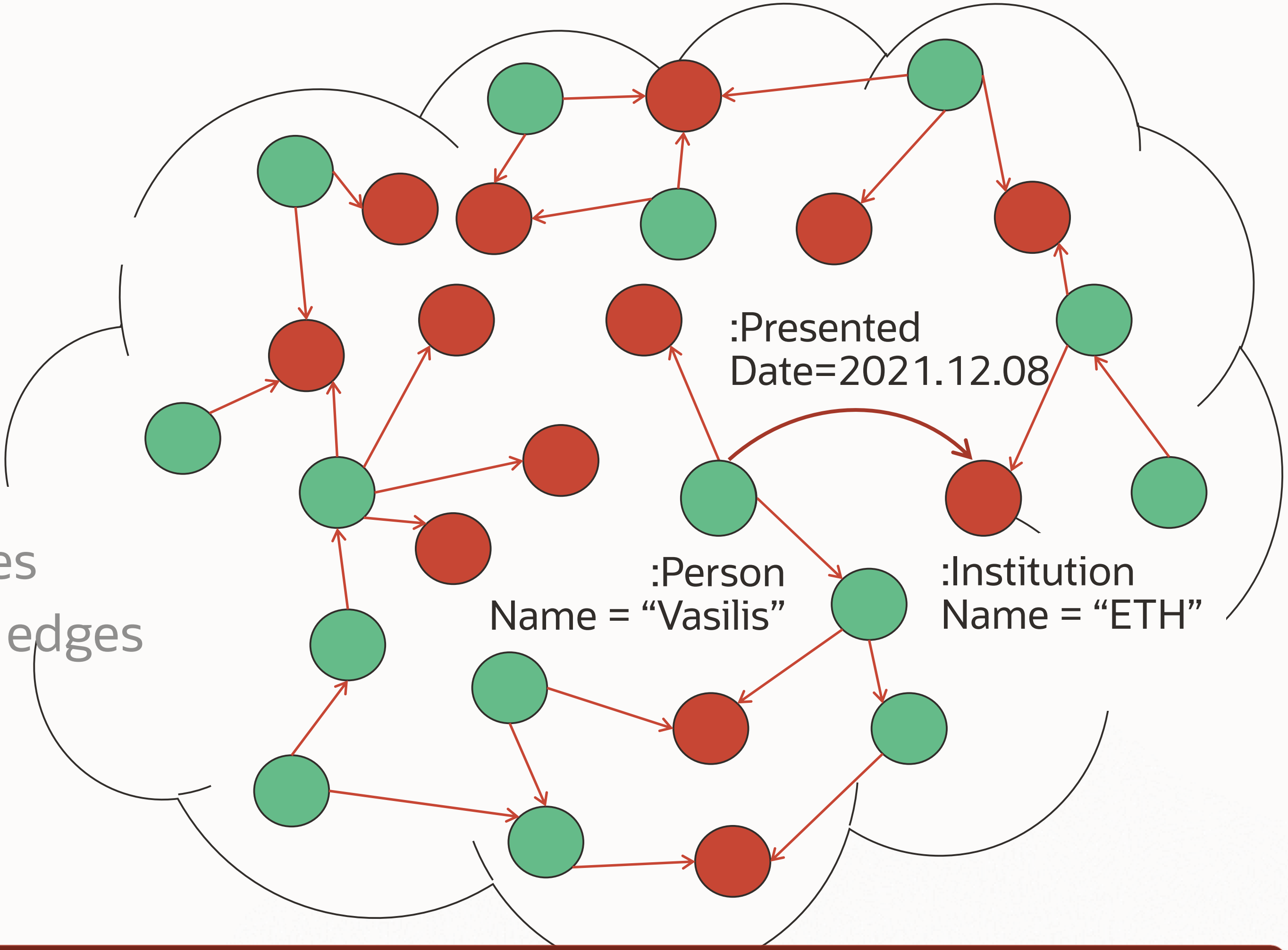
Graph analytics is a set of analytic techniques that shows how entities such as people, places and things are related to each other. Applications of the technology range from fraud detection, traffic route optimization and social network analysis to genome research.

Gartner predicts that the application of graph processing and graph databases will grow at 100% annually over the next few years to accelerate data preparation and enable more complex and adaptive data science.



# Your Data is a Graph!

- Represent it as a property graph
  - Entities are vertices
  - Relationships are edges
- Annotate your graph
  - Labels identify vertices and edges
  - Properties describe vertices and edges
- For the purpose of
  - Data modeling
  - Data analysis



Navigate multi-hop relationships quickly (instead of joins)

# Relational (Database) Model → Property Graph Model

| user_id (PK) | name |
|---|---|
| 0 | Vasilis |
| 1 | Lucas |
| … | … |

users

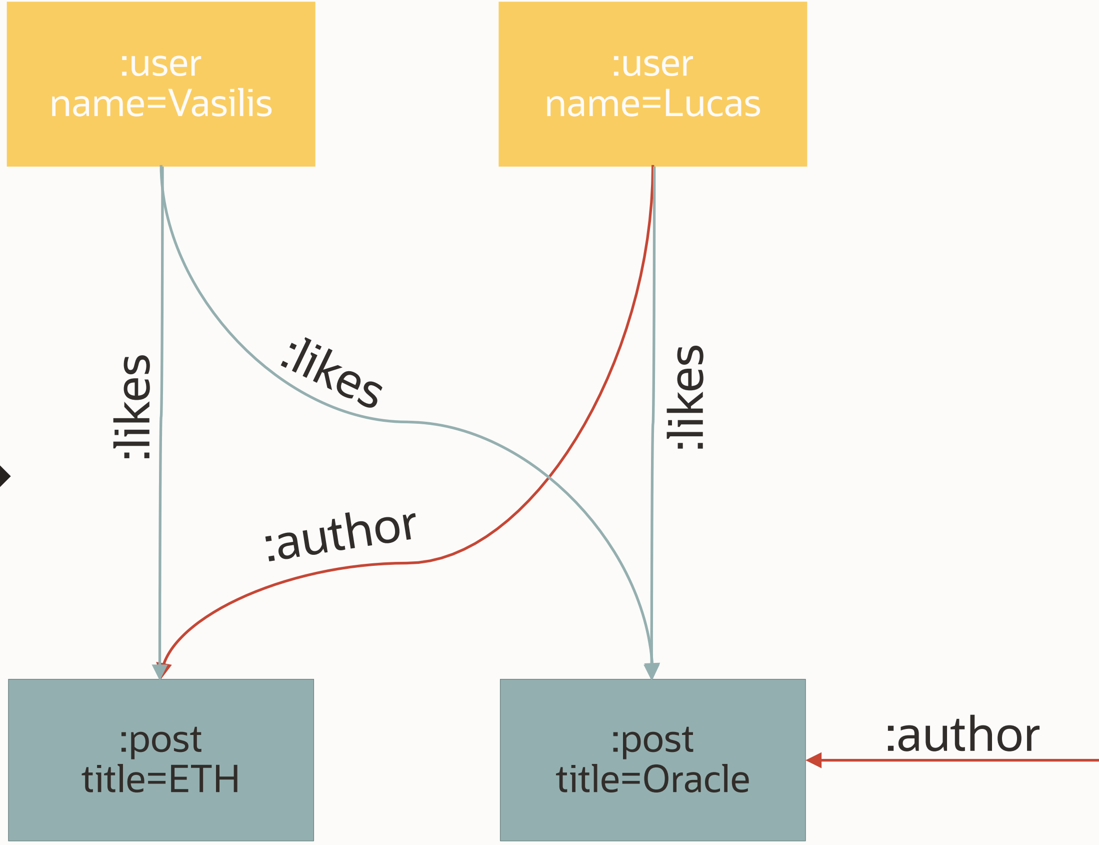
| user_id | post_id |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 1 |

user_likes

| author_id | post_id (PK) | title |
|---|---|---|
| 1 | 0 | ETH |
| 123 | 1 | Oracle |
| … | … | … |

posts

graph ♥



Essentially having "materialized joins"

# Example Query: Relational Model → Property Graph Model

*“Return any two people who like the same ‘Oracle’ post”*

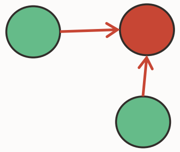
## SQL

```
SELECT u1.name, u2.name
FROM users u1, users u2, posts p,
     user_likes like1, user_likes like2
WHERE
     u1.user_id = like1.user_id AND
     u2.user_id = like2.user_id AND
     like1.post_id = like2.post_id AND
     p.post_id = like1.post_id AND
     p.title = "Oracle"
```

JOIN … JOIN … JOIN

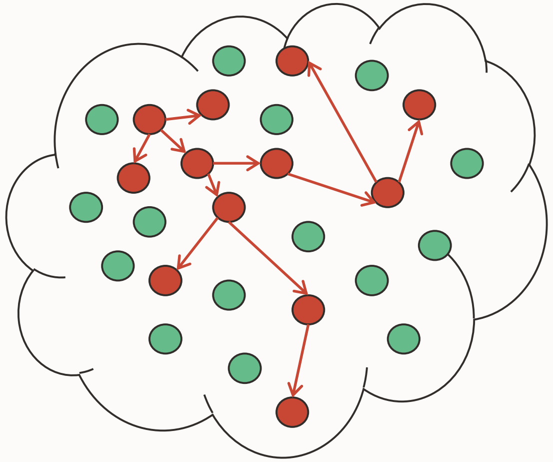
## PGQL

```
SELECT u1.name, u2.name
FROM graph_name
MATCH (u1:user)-[:likes]->(p:post),
      (u2:user)-[:likes]->(p:post)
WHERE
      p.title = "Oracle"
```

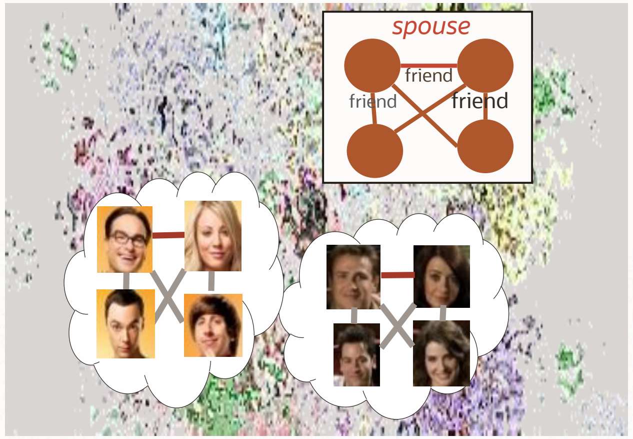


# Main Approaches of Graph Processing

1. Computational graph analytics [ASPLOS'12, VLDB'16]
   - Iterate the graph multiple times and compute mathematical properties using Greenmarl / PGX Algorithm (e.g., Pagerank)
   - e.g, `graph.getVertices().forEach(n -> …)`
2. Graph querying and pattern matching [GRADES'16/17, VLDB'16]
   - Query the graph using PGQL to find sub-graphs that match to the given relationship pattern
   - e.g., `SELECT … MATCH (a) -[edge]-> (b) …`
3. Graph ML (new)
   - Use the structural information latent in graphs
   - e.g., graph similarity

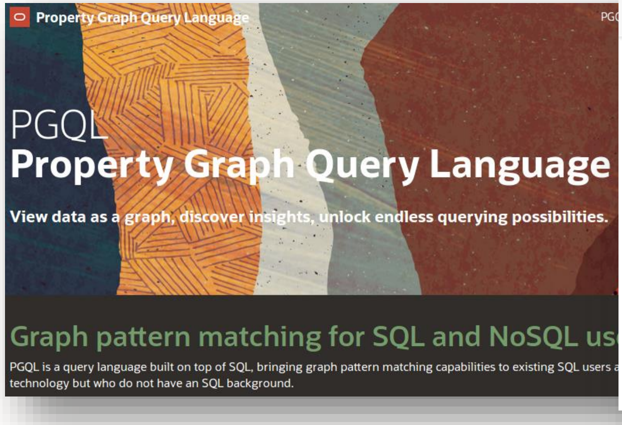


$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



# Oracle Labs PGX – Parallel Graph Analytix

- Fast, parallel, in-memory graph processing frameworks
- Efficient **graph analytics & queries**
  - 40+ built-in, graph analytics algorithms
- With **graph ML integrations**
  → one of the main focus points nowadays
- Embedded in Oracle products; active research project



Property Graph Query Language

PGQ

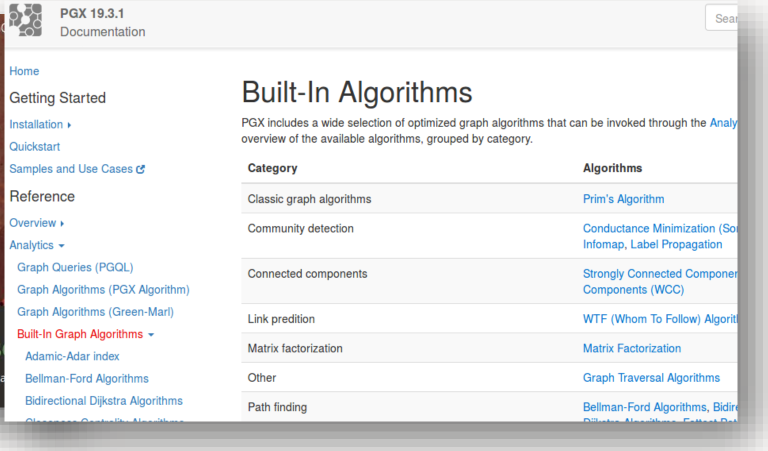
PGQL

**Property Graph Query Language**

**View data as a graph, discover insights, unlock endless querying possibilities.**

Graph pattern matching for SQL and NoSQL us

PGQL is a query language built on top of SQL, bringing graph pattern matching capabilities to existing SQL users a technology but who do not have an SQL background.



PGX 19.3.1
Documentation

Home

Getting Started

Installation ▸

Quickstart

Samples and Use Cases

Reference

Overview ▸

Analytics ▾

Graph Queries (PGQL)

Graph Algorithms (PGX Algorithm)

Graph Algorithms (Green-Marl)

Built-In Graph Algorithms ▾

Adamic-Adar index

Bellman-Ford Algorithms

Bidirectional Dijkstra Algorithms

## Built-In Algorithms

PGX includes a wide selection of optimized graph algorithms that can be invoked through the Analy overview of the available algorithms, grouped by category.

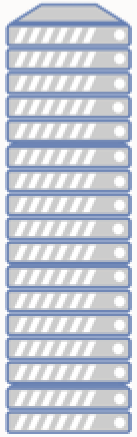
| Category | Algorithms |
|---|---|
| Classic graph algorithms | Prim's Algorithm |
| Community detection | Conductance Minimization (So Infomap, Label Propagation |
| Connected components | Strongly Connected Componen Components (WCC) |
| Link predition | WTF (Whom To Follow) Algorit |
| Matrix factorization | Matrix Factorization |
| Other | Graph Traversal Algorithms |
| Path finding | Bellman-Ford Algorithms, Bidire |

http://pgql-lang.org/

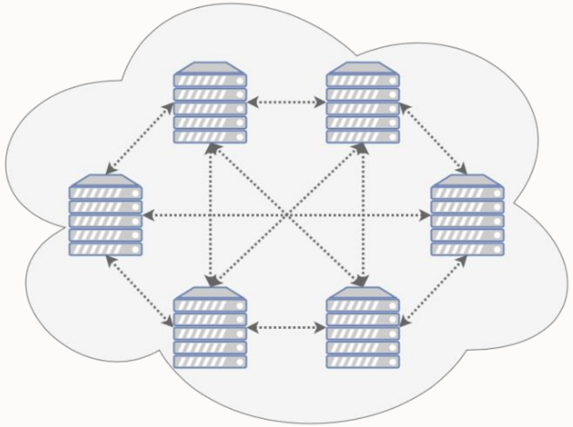
https://www.oracle.com/middleware/technologies/parallel-graph-analytix.html

(1) single machine (2) distributed

PGX.SM
Java based



PGX.D
Scalable, cloud oriented
C++ based



(3) Database

Graph-in-DB
Make graph a first class citizen in DB

# PGX Algorithm (VLDB'16)

- A Java Embedded DSL specially designed for graph data analysis
  - Easy development of algorithms – as simple as using your favorite Java IDE
  - A subset of Java is supported
  - Execution can be targeted for very different environment. (e.g. distributed)

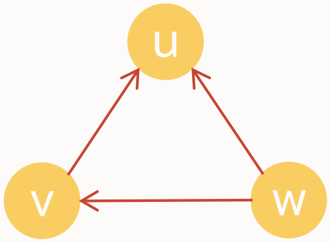
```
import com.oracle.pgx.api.beta.GraphAlgorithm;
import com.oracle.pgx.api.beta.PgxGraph;
import com.oracle.pgx.api.beta.VertexProperty;
import com.oracle.pgx.api.beta.annotations.Out;

@GraphAlgorithm
public class DegreeCentrality {
  void degree_centrality(PgxGraph g, @Out VertexProperty<Long> dc) {
    g.getVertices().forEach(n ->dc.set(n, n.getOutDegree() + n.getInDegree()));
  }
}
```

Distinguish input/output parameters

parallel loop over all nodes accepting a lambda

Similar naming as PGX API

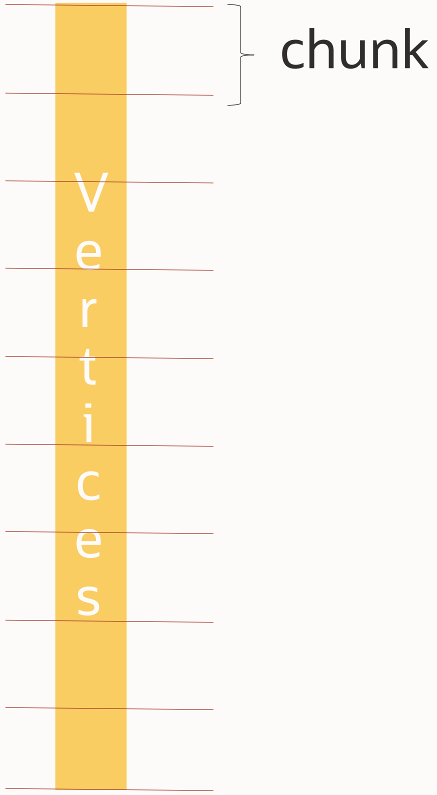


# From Algorithm to Efficient Execution (PGX.SM)

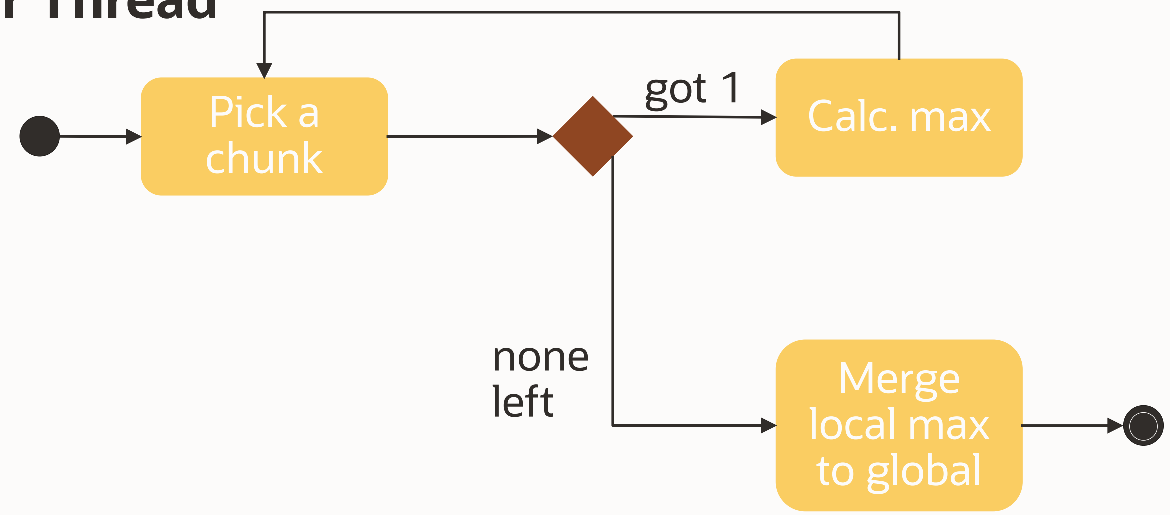
PGX algorithm is compiled to fast, parallel low-level code

- Uses Callisto-RTS parallel runtime [ATC’15]

```
double max_degree(PgxGraph g) {
    double maxDegree;
    g.getVertices().forEach(n ->
          Reduction.updateMaxValue(maxDegree, n.getDegree())
     );
     return maxDegree;
}
```



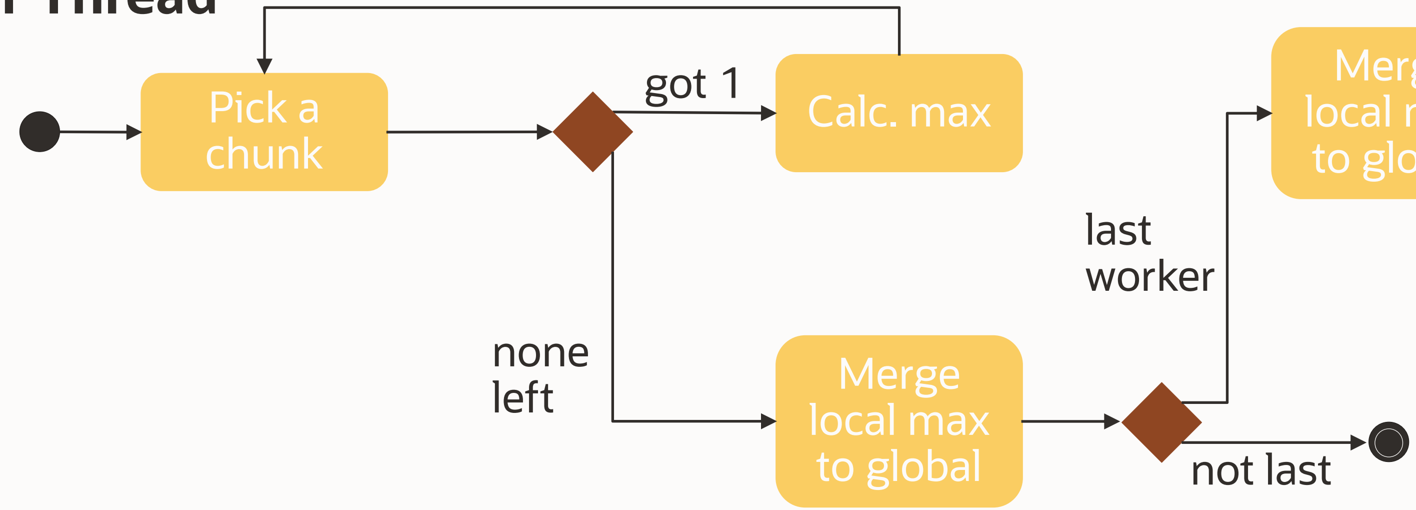
**Worker Thread**

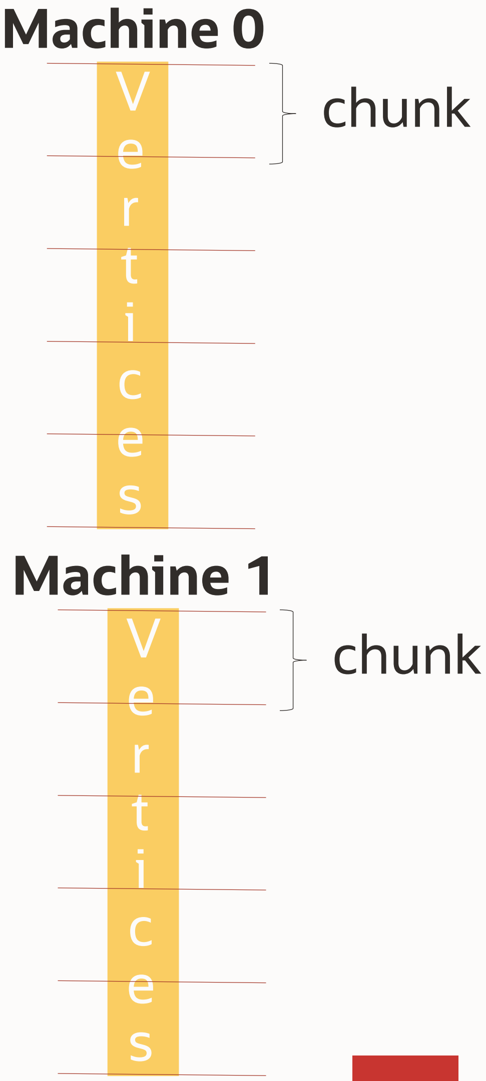


# From Algorithm to Efficient Execution (PGX.D)

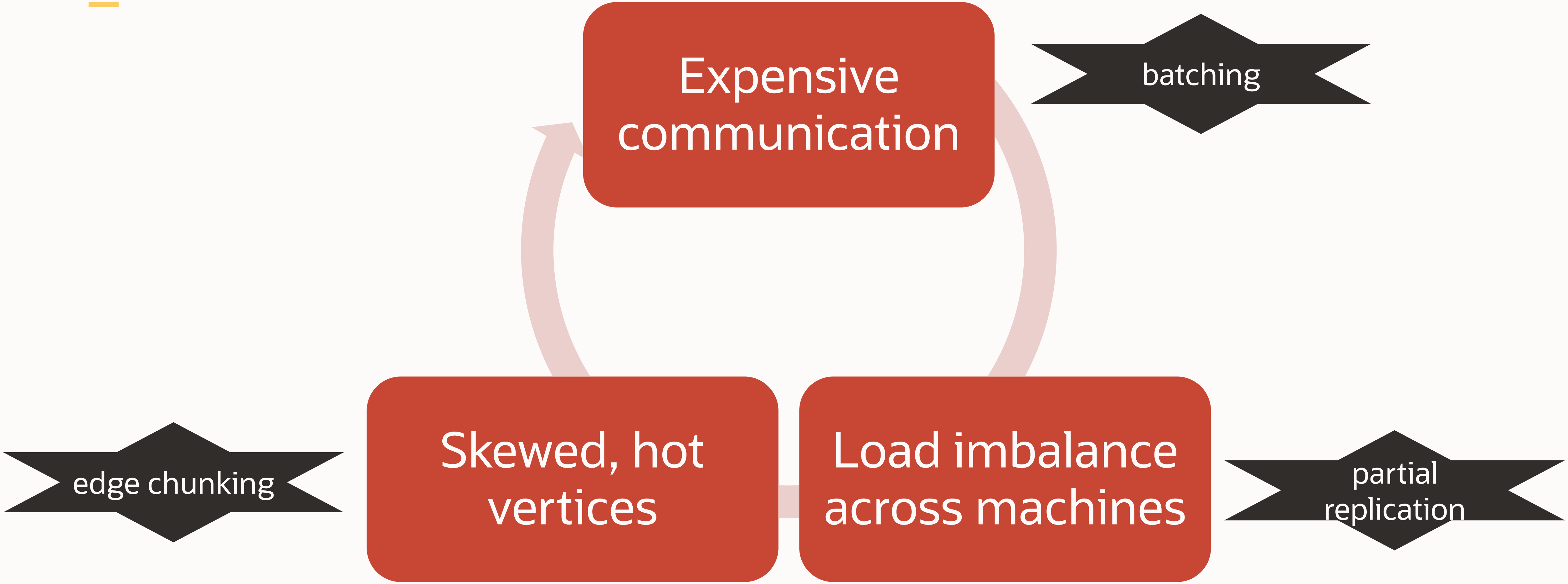
```
double max_degree(PgxGraph g) {
    double maxDegree;
    g.getVertices().forEach(n ->
          Reduction.updateMaxValue(maxDegree, n.getDegree())
     );
     return maxDegree;
}
```

Worker Thread

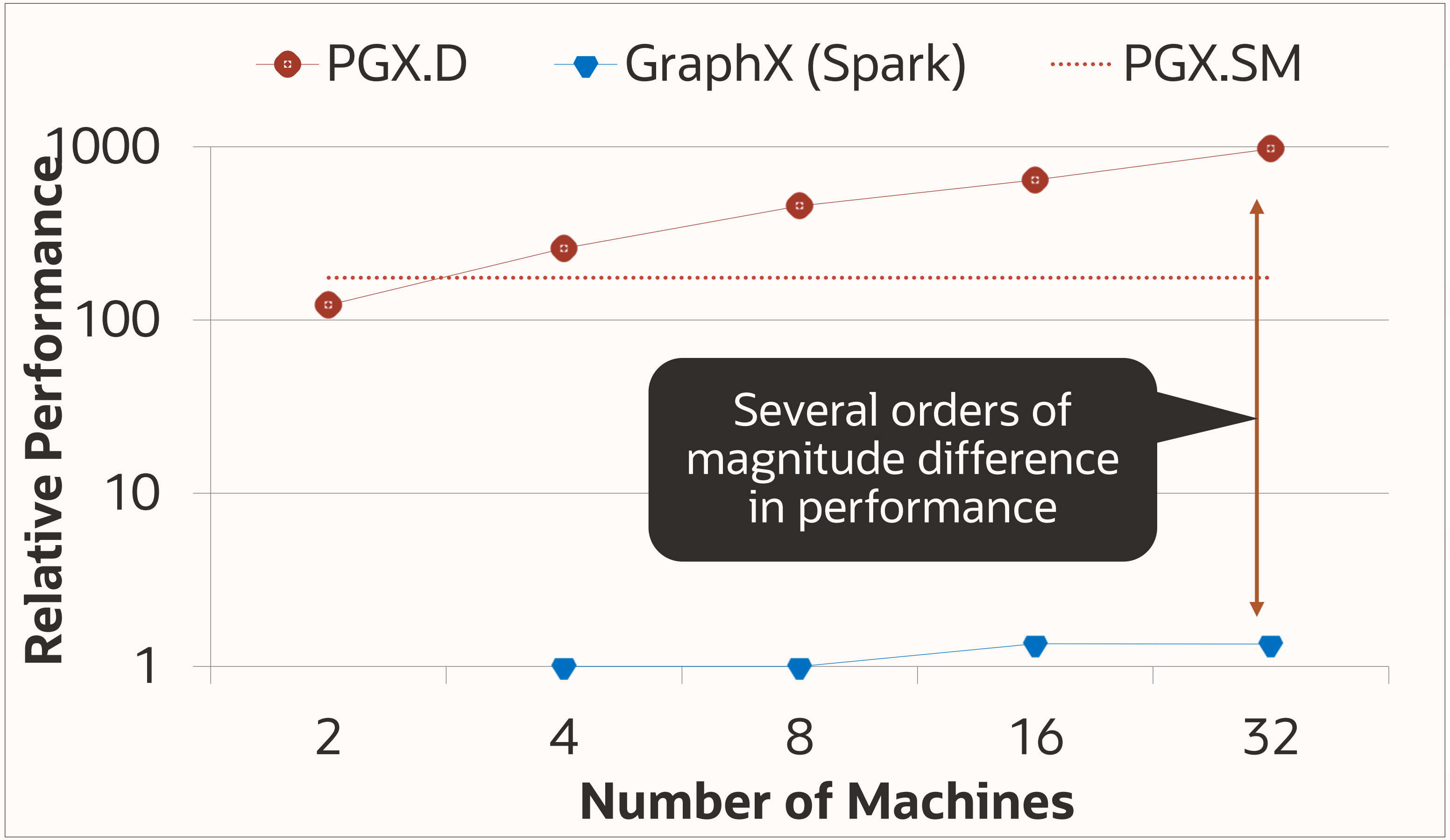




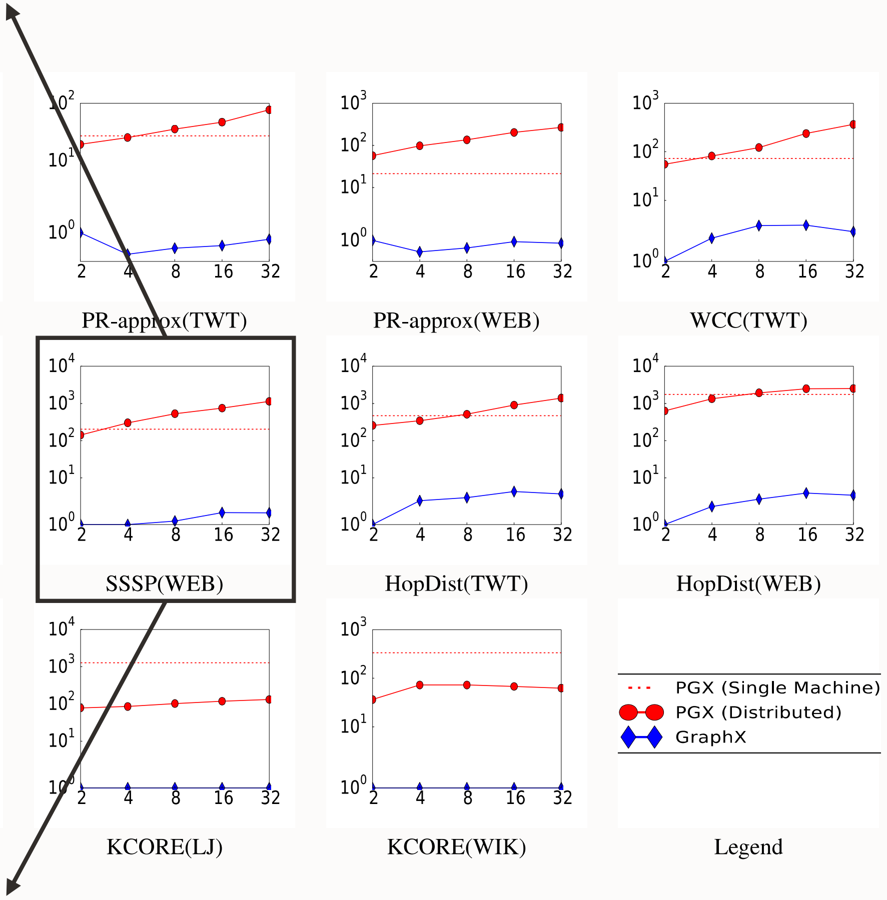
# Key Challenges For Distributed Graph Analytics (SC'15)



# PGX.D Performance: Graph Algorithm Computation



Several orders of magnitude difference in performance



```
Hardware:     Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Network:      Infiniband (40Gbps)
```

# Agenda

1 Distributed Graph Processing with PGX.D

- Graph Processing
- Graph Algorithms
- **Graph Queries**

2 A Quick Intro into Oracle Labs + Internships

# PGQL: Graph Query Language

- Query language for Property Graphs with SQL-like syntax
- Proposed and maintained by Oracle
- SQL-like operators: SELECT, WHERE, ORDER BY, GROUP BY, …
- Graph operators: graph pattern MATCH, PATH (reachability) and SHORTEST

Property Graph Query Language

PGQL 1.4

PGQL
Property Graph Query Language

View data as a graph, discover insights, unlock endless querying possibilities.

```
SELECT p.name, COUNT(*) AS num_movies
FROM movies_graph
MATCH (p:Person) -[:Directed]-> (m:Movie), (p) -[:Played_in]-> (m:Movie)
                       /* same person, same movie */
GROUP BY p
ORDER BY num_movies DESC
LIMIT 5
```

**Result**

```
+-----------------------------------+
| p.name              | num_movies  |
+-----------------------------------+
| Clint Eastwood      | 10          |
| Woody Allen         | 9           |
| Michael Moore       | 5           |
| David Hewlett       | 4           |
| Jay Chandrasekhar   | 3           |
+-----------------------------------+
```

# Distributed Graph Queries Are Very Difficult

- Intermediate (and final) result explosion

Twitter graph

```
SELECT COUNT(*) MATCH (a)
+-------------------------+
|    COUNT(*)             |
+-------------------------+
| 41,652,230              |
+-------------------------+
```

0 hops

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

1 hop

```
SELECT COUNT(*) MATCH (a)->()->()
```

?

2 hops

Distributed PGX
8 machines
~1200 seconds
~ 8B matches/s

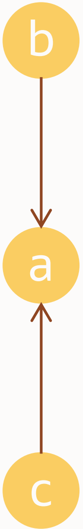
- Limited locality (especially with many machines)
- Do not want to do database JOINs

**We need an in-memory solution that can handle the scale**

| | PGX.SM | PGX.D |
|---|---|---|
| **Analytics** | BFS<br>(Parallel for) | BFS<br>(Bulk-synchronous) |
| **Queries** | BFS<br>(Parallel for) | almost-DFS<br>(Non-blocking) |

# Breadth-First vs. Depth-First Traversal Example

## BFT



for all

1. Match all ‘b’

for all

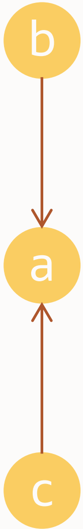
2. Match all ‘a’

for all

3. Match all ‘c’

{b0}
{b1} ⟶ {b0, a0}
{b0, a1}
{b1, a2}
{b1, a3} ⟶ {b0, a0, c1}
{b0, a0, c2}
{b1, a1, c3}
…

## DFT



1. Match one ‘b’
2. Match one ‘a’
3. Match one ‘c’

for all

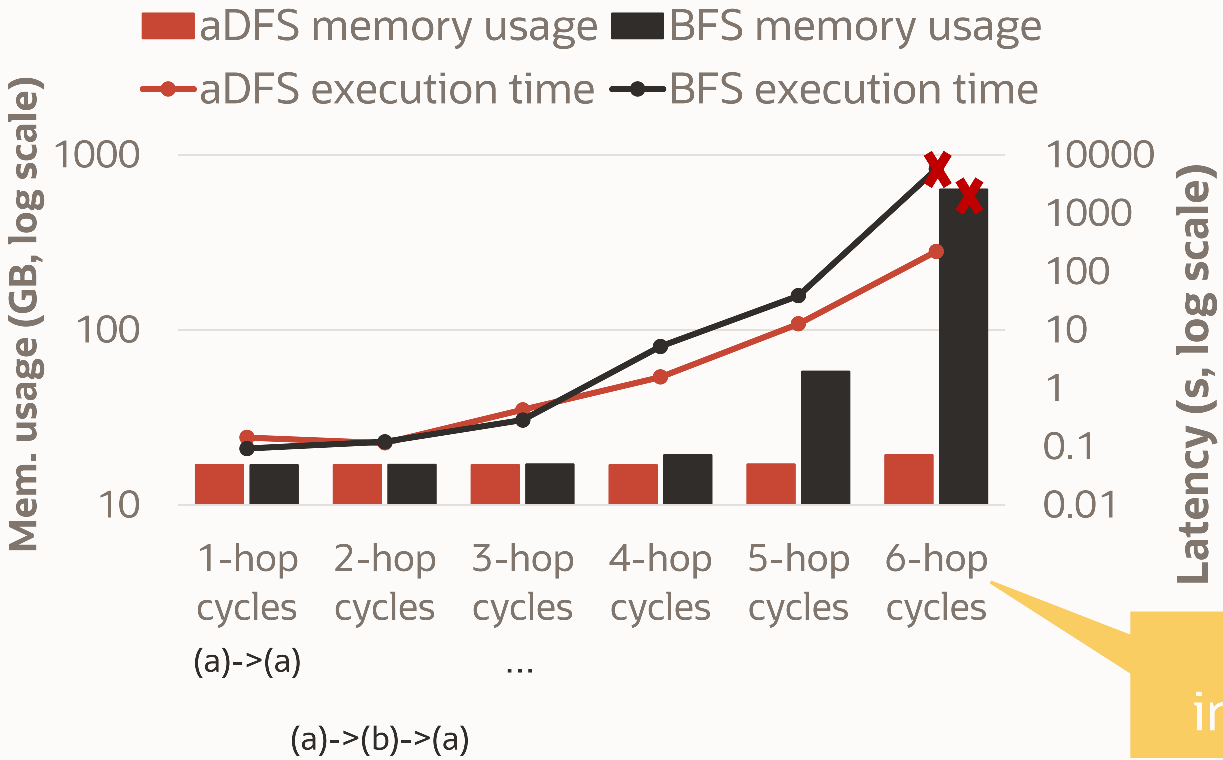
for all

for all

{b0} ⟶ {b0, a0} ⟶ {b0, a0, c1}
⟶ {b0, a0, c2}
{b0, a1} ⟶ {b0, a1, c3}

# BFS vs. Almost-DFS: Performance / Memory

- 875K vertices and 5.1M edges graph (2002 Google Programming Contest)
- 8 machines with 768GB memory each = 6TB of memory



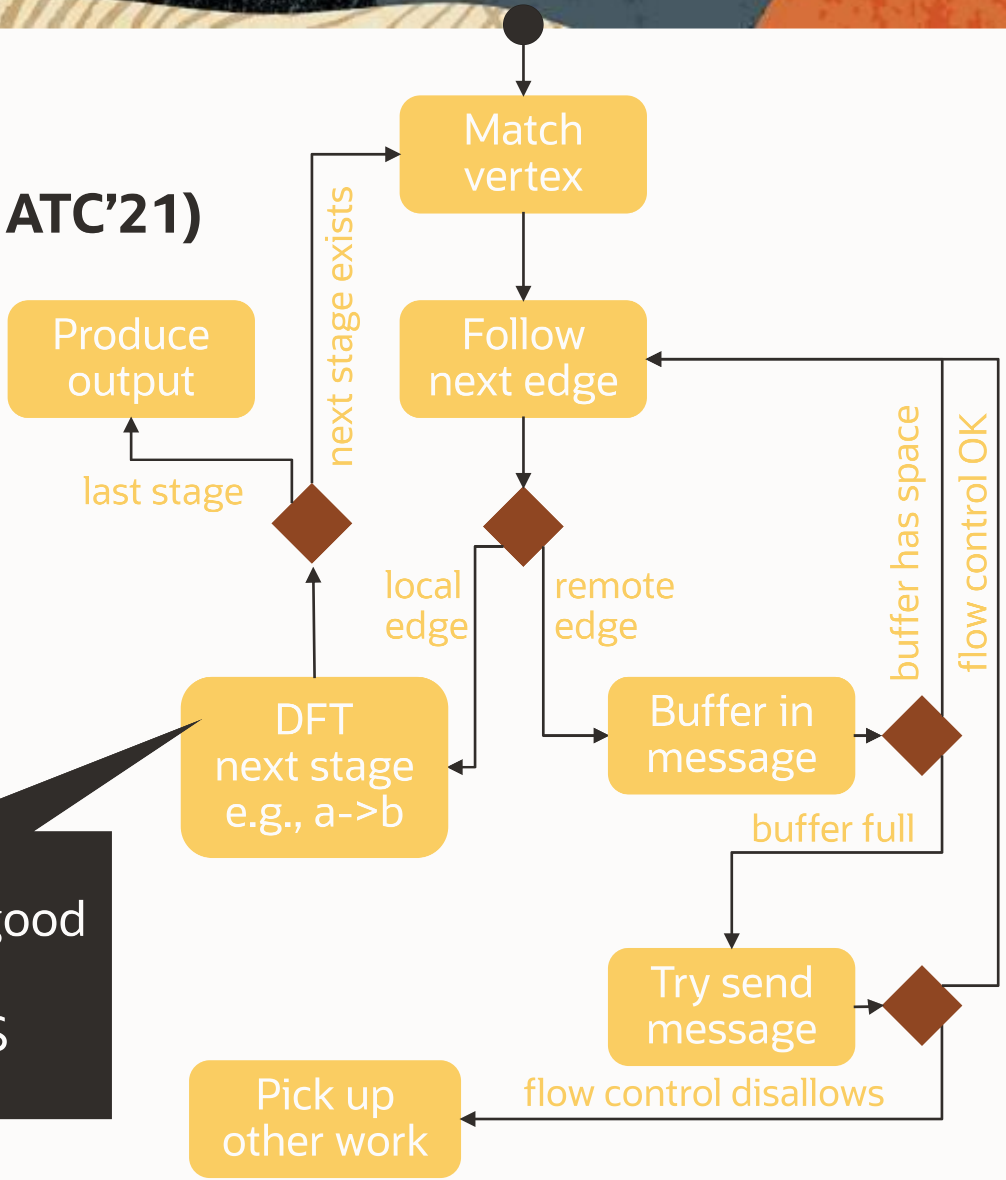
266 billion intermediate results

# PGX.D/Async Approach (USENIX ATC'21)

1. Asynchronous communication
   - Asynchronously send intermediate results
   - Avoid flooding by fined-grained flow control
   - Guaranteed to finish (and detect finish)
   - Workers do not block due to remote communication

2. Depth-first traversal (DFT)
   - Eager completion of matches
   - Allows for fine-grained flow control
   - Execution is bounded by allocated memory

→ Control memory / network consumption



Is strict DFT a good idea? **No** → Almost-DFS

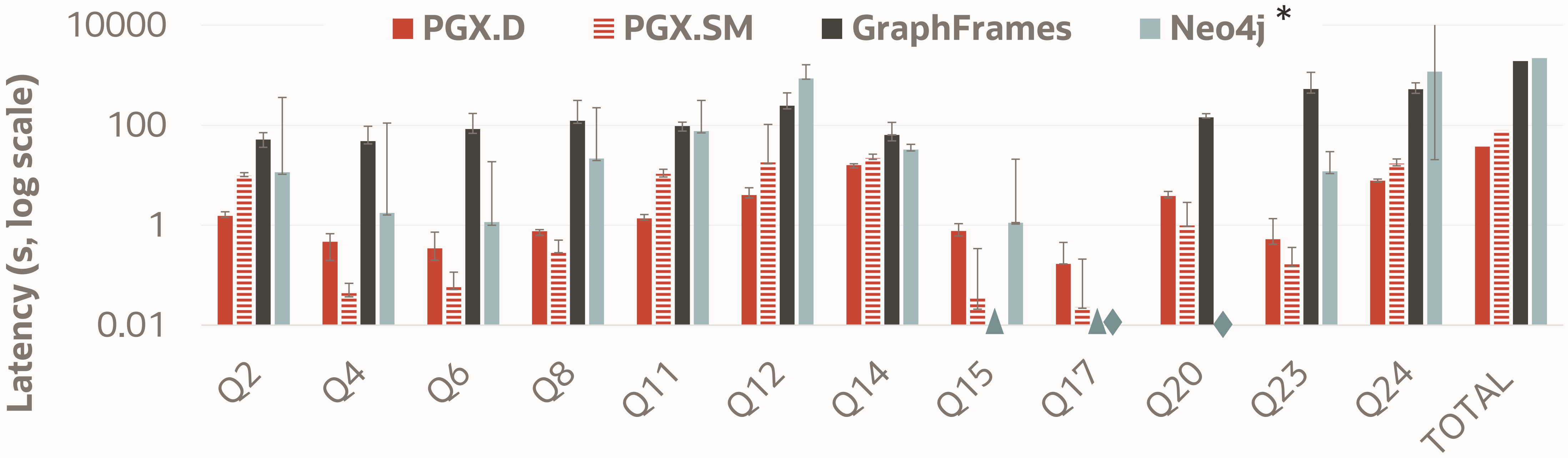
**In-memory distributed execution with controllable network/memory usage**

# PGQL Performance with PGX.D: LDBC

▲ missing feature
◆ incorrect results

- With hybrid depth-first/breadth-first execution runtime for PGX.D
- LDBC 100 Social Graph (283M vertices, 1.78B edges) and Queries
- PGX.D and Apache Spark GraphFrames on 8 machines

More in USENIX ATC'21 paper



**52x faster than Spark GraphFrames**
**66x faster than Neo4j**

* Neo4j community edition; the benchmarks have not been audited by the Neo4j team

# Agenda

1 Distributed Graph Processing PGX.D

2 **A Quick Intro Into Oracle Labs + Internships**

Our mission is to help people
see data in new ways, discover insights,
unlock endless possibilities.

**Identify**, **explore**, and **transfer** new technologies that have the potential to substantially improve Oracle's business.

**Oracle Labs Mission Statement**

# Oracle Labs's Four Pillars

### Exploratory Research

- New ideas within domains relevant to Oracle

### Consulting

- Provide expertise to product organizations

### Directed Research

- Difficult, future-looking problems
- Driven by product requirements
- In collaboration with product teams

### Product Incubation

- Grow new products from Oracle Labs research

# Oracle Labs’ Global Research Team

## Global research team

220+ researchers

Zurich: 80+ researchers

The geographic spread allows Oracle Labs to take advantage of a **tremendous pool of scientific and engineering talent** and enables Labs researchers to **collaborate with colleagues** from a **wide range of industries and universities**.

## Oracle Labs locations

- Zurich, Switzerland
- Prague & Brno, Czech Republic
- Casablanca, Morocco
- Linz, Austria
- Redwood Shores, USA
- Belgrade, Serbia
- Brisbane, Australia
- … and more!

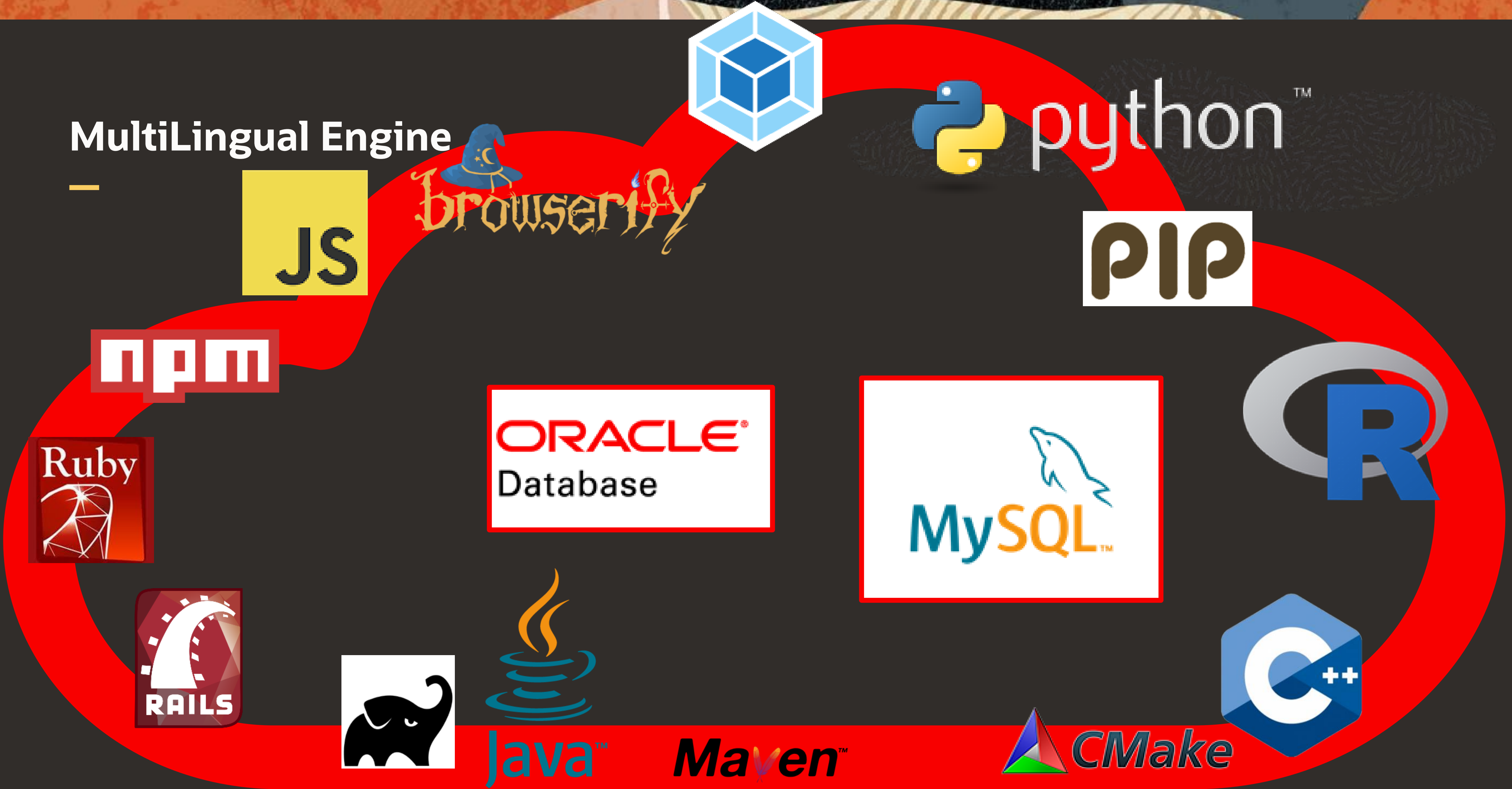
# Selection of Projects with Involvement of the Zurich Lab

- **Parallel Graph AnalytiX (PGX)** – High-performance graph toolkit
- **Data Studio (DS)** – Notebook technology for visualizing graphs and more
- **GraalVM** – A universal, polyglot VM environment
- **Active Libraries (AL)** – Self-optimizing Code based on runtime execution and data patterns
- **MultiLingual Engine (MLE)** – Bringing modern languages into the Oracle DB

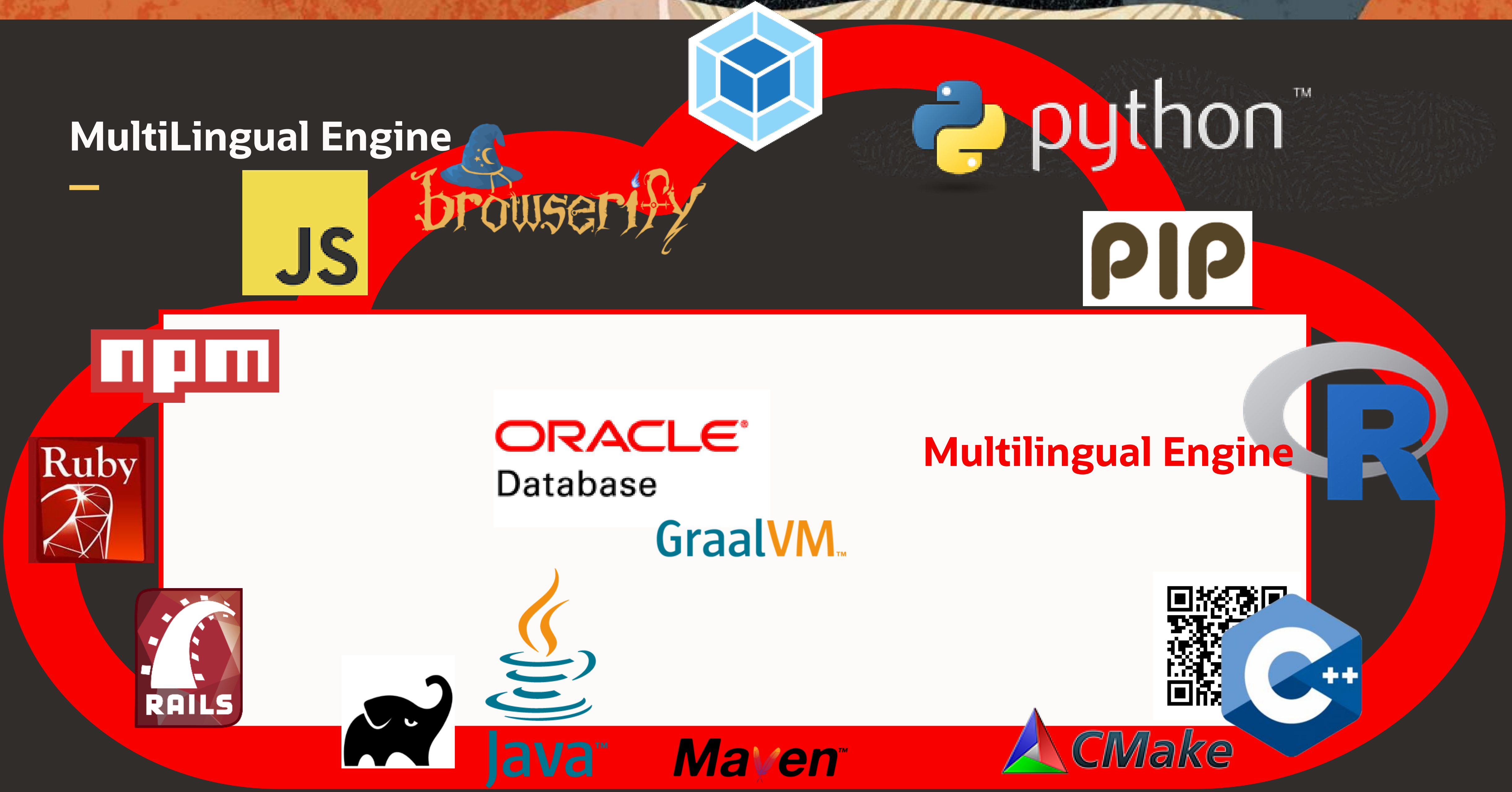
Several other topics across the other offices

- ML / AI applications, code analysis and security, concurrent programming, …

# MultiLingual Engine



# MultiLingual Engine


ORACLE Database

GraalVM

Multilingual Engine

# Internship and Job Opportunities

**Visit the Oracle Labs Internship Page, labs.oracle.com/pls/apex/labs/r/labs/internships**

- Automated Machine Learning with Explainability (AutoMLx)
- Automating OCA Verification of GitHub Pull Requests
- BPF Linux Schedulers
- Extending a Distributed Graph Engine (Oracle Labs PGX)
- Extending a Web-Based Enterprise Data Science Platform
- Graph Machine Learning at Oracle
- Graph Support in the Oracle Database
- Machine Learning and Data Analysis Techniques for Domain Global Graphs
- Machine Learning for Optimizing Oracle Database Performance
- Machine Learning Processing in DB Systems
- Oracle Database Multilingual Engine - Modern Programming Languages in the Database

**Interning at Oracle Labs** as part of the Data Studio team **was a great experience**. I was not only able to apply the **knowledge** gathered from my studies, but also **extend it through challenging tasks** in an environment of **very supportive and welcoming colleagues**.

**Nils Blach**

ETH student, 6-month intern with Oracle Labs in 2019/2020

# Internships at Oracle Labs Zurich*

Regular internships or MSc thesis

Typically 3 to 12 months

Competitive salary

Apply / get in touch with us via lucas.braun@oracle.com

*Currently remote due to COVID-19 (subject to change)

# Using the Oracle Cloud for free

**Everybody**
Oracle Cloud Always-Free Tier: oracle.com/cloud/free/

**Universities and Schools**
Oracle Academy: academy.oracle.com

**Research Institutions**
Oracle For Research: oracle.com/oracle-for-research/

# In Summary

- PGX.D is a highly-scalable distributed graph engine
  - Easy-to-write graph algorithms
  - Fast alsways in-memory distributed queries
- Oracle Labs is looking for you! Apply now by emailing to lucas.braun@oracle.com.

**Any questions?**

# Thank you.

# Have also a look at out our internship topics in the VIS Job Emails – we'd love to get your application.

**Stay healthy.**

ORACLE