ORACLE

14th of December 2022

Distributed Graph Processing with PGX.D

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

Calin Iorgulescu
Principal Researcher
Oracle Labs Zurich

Lucas Braun
Principal Program Manager
Oracle Labs Zurich

Calin Iorgulescu

- Principal Researcher @ Oracle Labs
- PhD in Computer Science from EPFL
- Started at Oracle in 2020
- Working on the PGX Distributed (PGX.D) project

in/ciorgulescu

ORACLE Database

Lucas Braun

- Principal 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

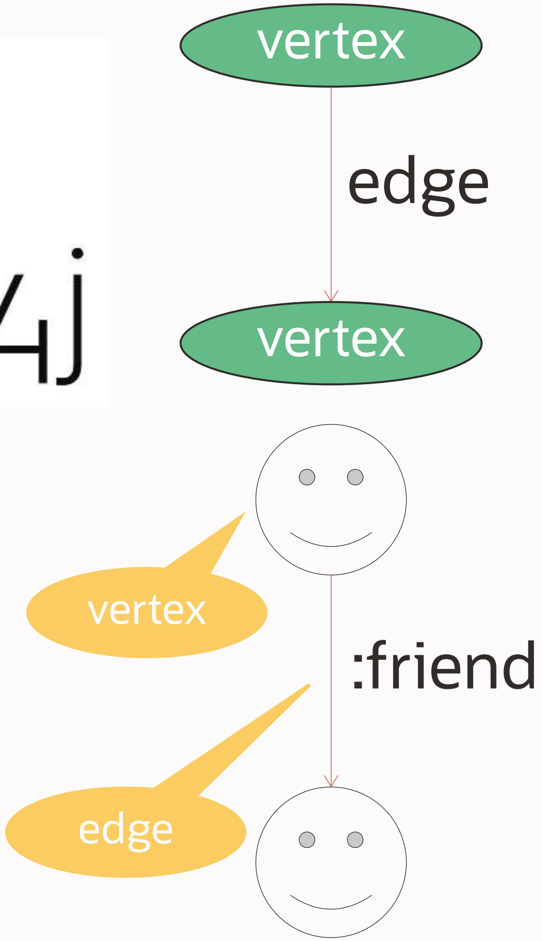
APACHE Spark™

MariaDB

neo4j

Dragon

Microsoft Graph Engine

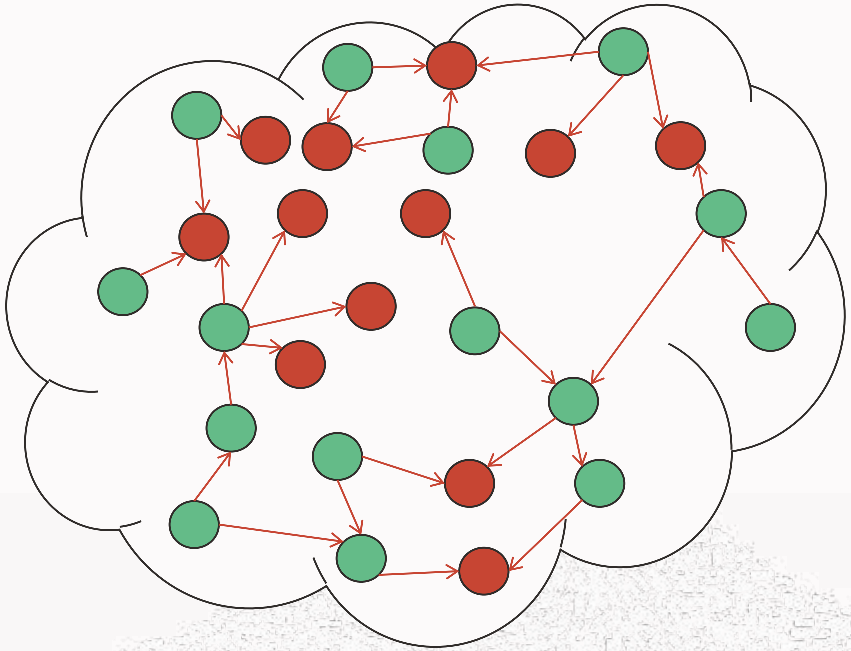


Gartner’s Top 12 Data and Analytics Technology Trends for 2022:
Trend No. 5: Context-enriched analysis built on graph

Trend No. 5: Context-enriched analysis

https://www.gartner.com/en/articles/12-data-and-analytics-trends-to-keep-on-your-radar

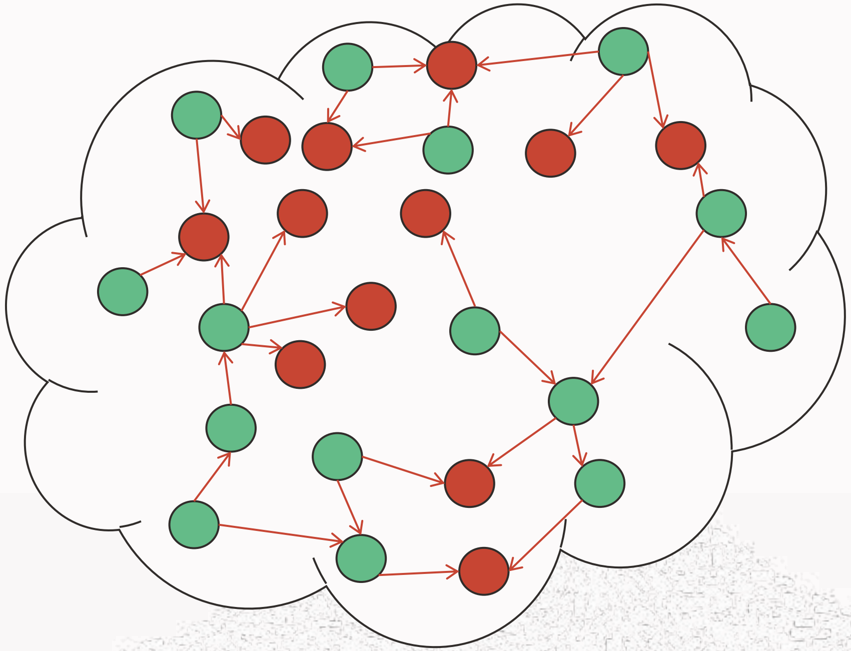
> “Context-enriched analysis builds on graph technologies. The information on the user’s context and needs is held in a graph that enables deeper analysis using the relationships between data points as much as the data points themselves. It helps identify and create further context based on similarities, constraints, paths and communities.
>
> Capturing, storing and using contextual data demands capabilities and skills in building data pipelines, X analytics techniques and AI cloud services that can process different data types. By 2025, context-driven analytics and AI models will replace 60% of existing models built on traditional data.



Trend No. 8: Graph Relates Everything (2021 Report)

https://www.gartner.com/smarterwithgartner/gartner-top-10-data-and-analytics-trends-for-2021/

> “Graph forms the foundation of modern data and analytics with capabilities to enhance and improve user collaboration, machine learning models and explainable AI. Although graph technologies are not new to data and analytics, there has been a shift in the thinking around them as organizations identify an increasing number of use cases. In fact, as many as 50% of Gartner client inquiries around the topic of AI involve a discussion around the use of graph technology.



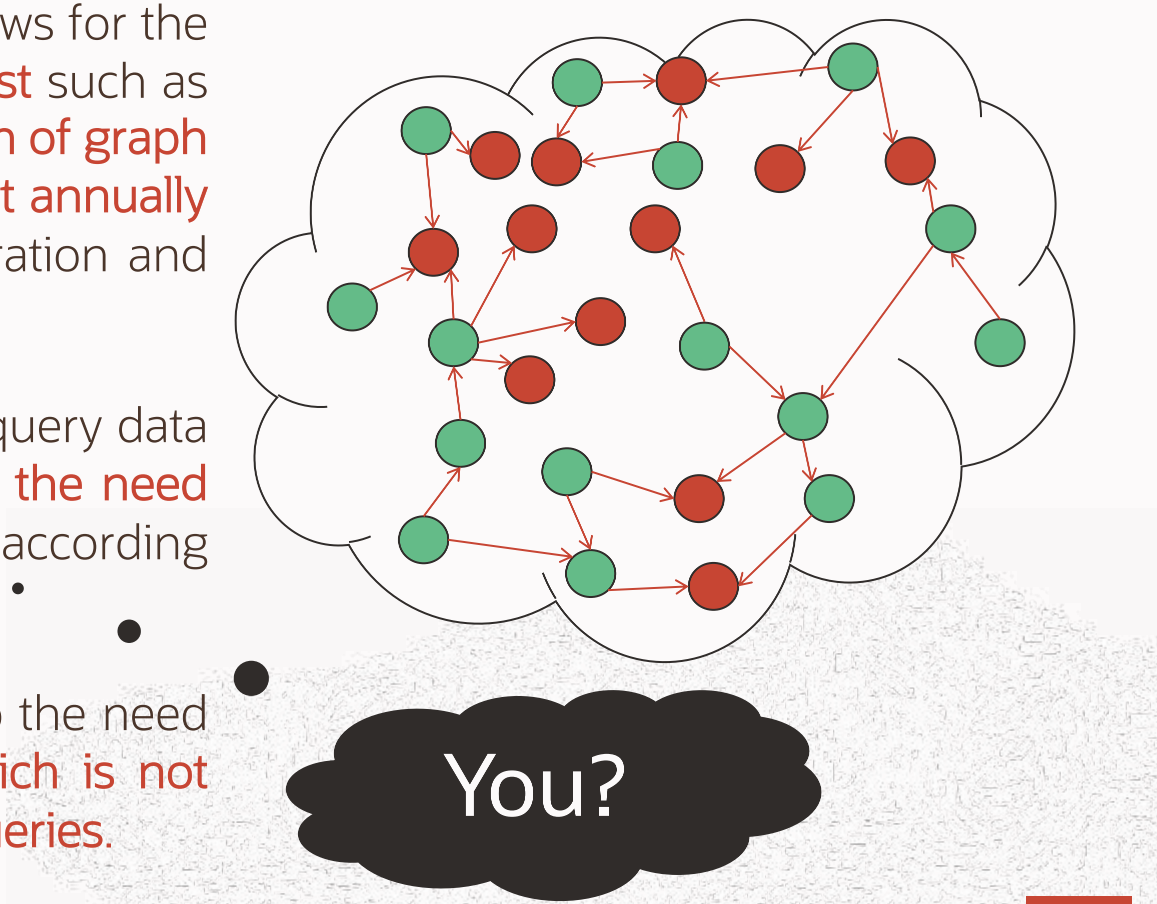
Trend No. 4: Graph analytics (2019)

https://www.gartner.com/en/newsroom/press-releases/2019-02-18-gartner-identifies-top-10-data-and-analytics-technolo

“Graph analytics is a set of analytic techniques that allows for the exploration of relationships between entities of interest such as organizations, people and transactions. The application of graph processing and graph DBMSs will grow at 100 percent annually through 2022 to continuously accelerate data preparation and enable more complex and adaptive data science.

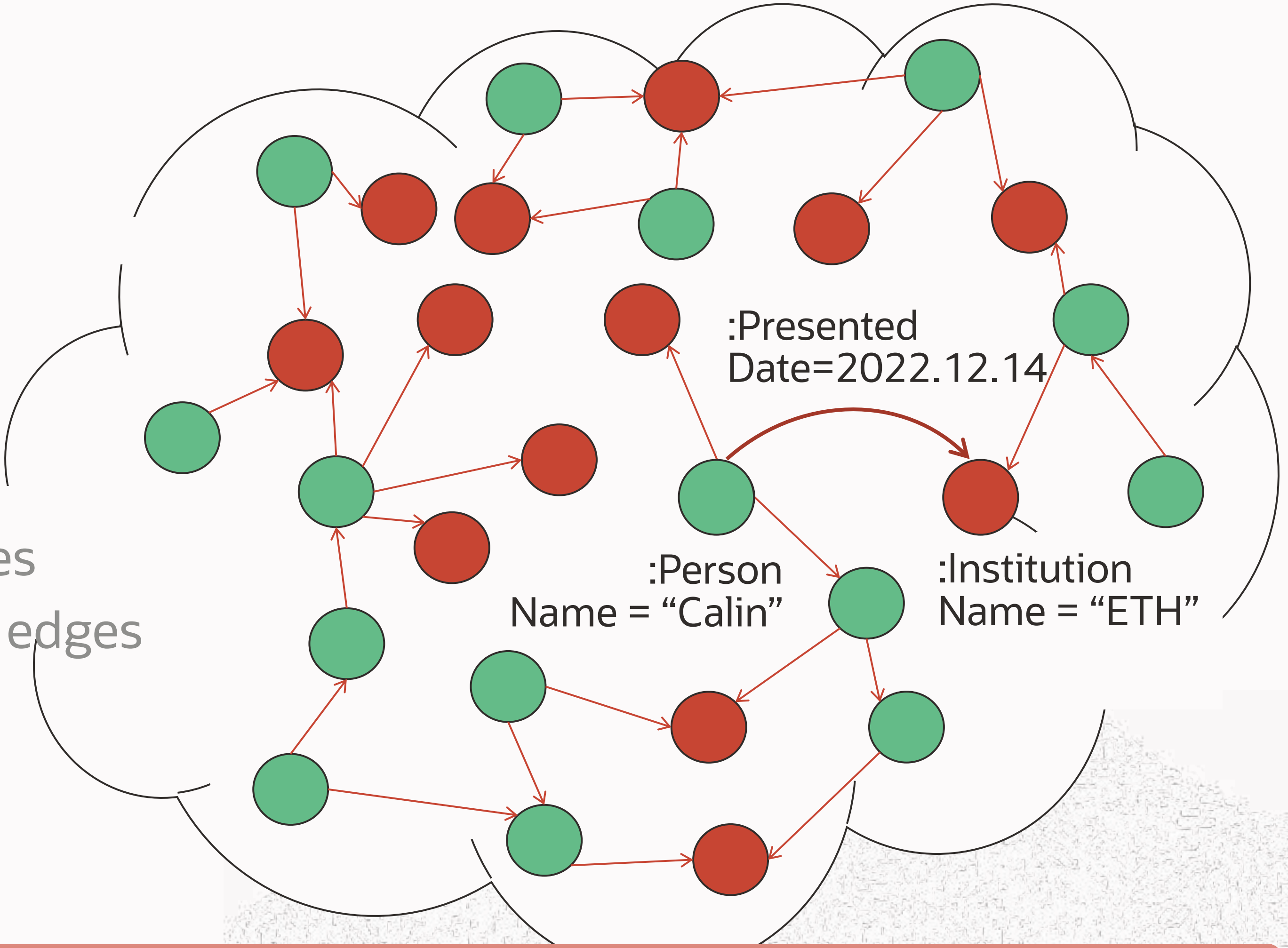
Graph data stores can efficiently model, explore and query data with complex interrelationships across data silos, but the need for specialized skills has limited their adoption to date, according to Gartner.

Graph analytics will grow in the next few years due to the need to ask complex questions across complex data, which is not always practical or even possible at scale using SQL queries.



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

users

user_id (PK)	name
0	Calin
1	Lucas
…	…

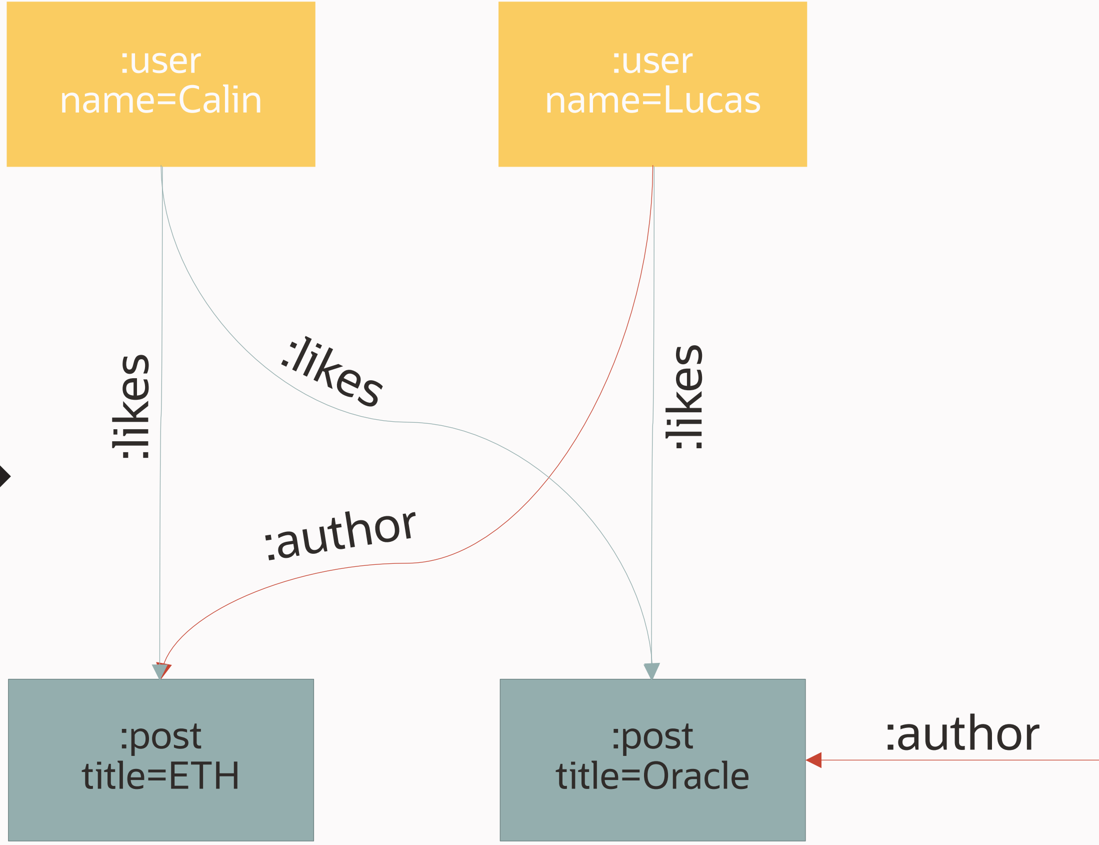
user_likes

user_id	post_id
0	0
0	1
1	1

posts

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

graph ❤️



Essentially having "materialized joins"

Example Query: Relational Model → Property Graph Model

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

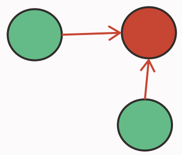
SQL

```
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"
```



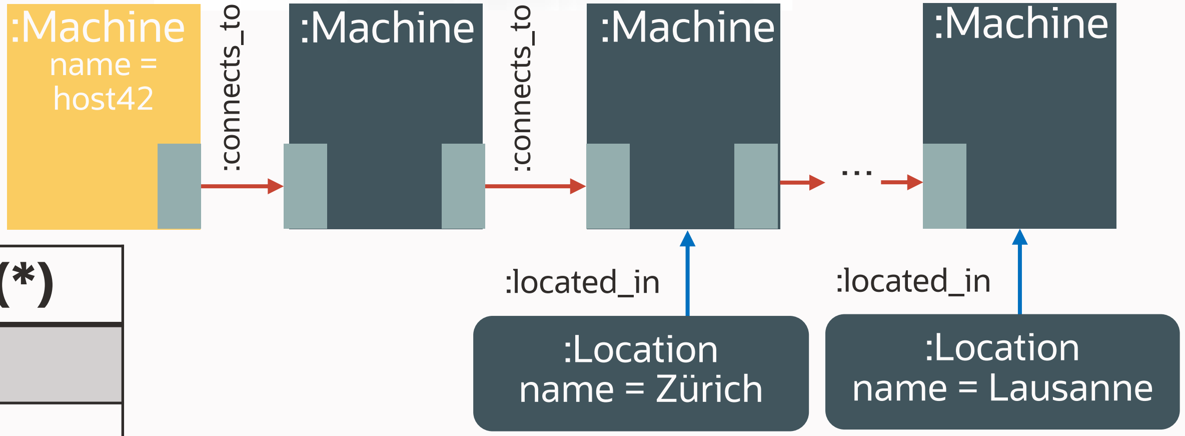
Example: Cloud Network Management

Use a graph to represent the machines and their interconnections

- Racks, machines, switches, ports, wires …

Looking for multi-hop connections

Count the number of Machines (indirectly) connected to host “host42”, group by Location name



loc.name	COUNT(*)
Zurich	1
Lausanne	1

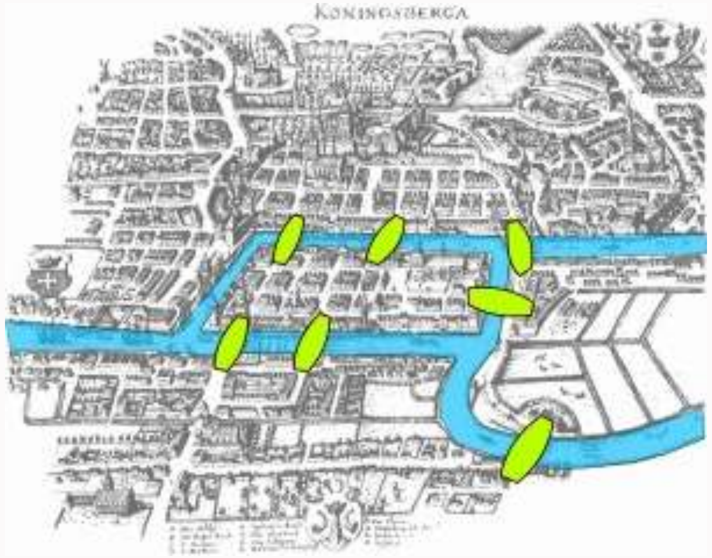
PGQL query

```
SELECT loc.name, COUNT(*)
MATCH (x) -/:connects_to*/-> (y:Machine),
      (y) <- [:located_in] - (loc:Location)
WHERE x.name = 'host42'
GROUP BY loc
```

Easy to write a query and fast to get the answer thanks to the graph model

Beyond Queries: Graph Algorithms for Powerful Analytics

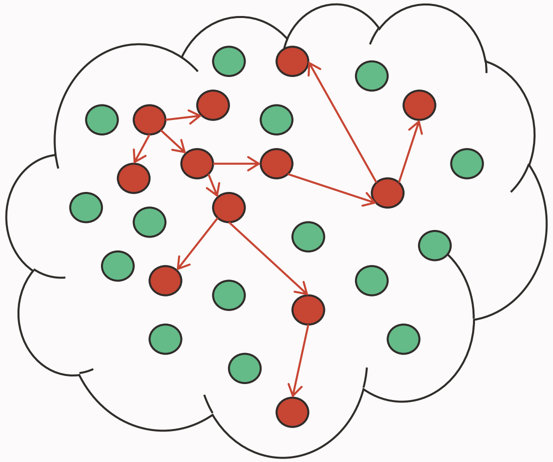
- **Graphs have been studied in Maths for centuries**
  - Since Euler's "Seven Bridges of Königsberg", 1736 [1]
- **Classic problems on graphs [1, 2]**
  - Graph isomorphism
  - Traveling salesman's problem
  - Max flow, min cut
  - …
- **More recent developments**
  - PageRank [3]
  - InfoMap [4]



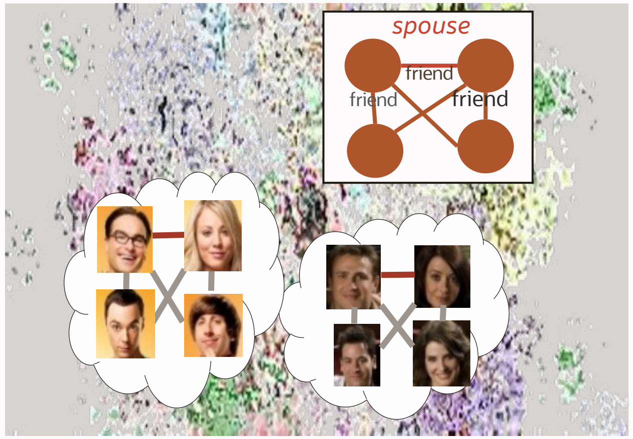
[1] https://en.wikipedia.org/wiki/Graph_theory#History
[2] https://docs.oracle.com/cd/E56133_01/latest/reference/algorithms/index.html
[3] https://en.wikipedia.org/wiki/PageRank
[4] http://www.mapequation.org/

Main Approaches of Graph Processing

1. Computational graph analytics [ASPLOS’12, VLDB’16]
   - Iterate the graph multiple times and compute mathematical prope
     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
   - e.g., `SELECT … MATCH (a) -[edge]-> (b) …`
3. Graph ML (new)
   - Use the structural information latent in graphs
   - e.g., graph similarity

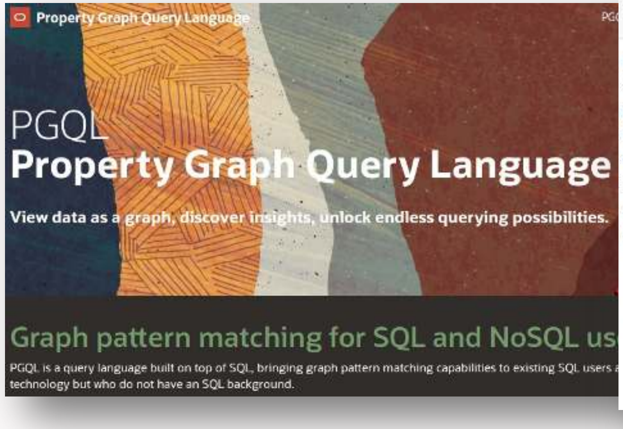


$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



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



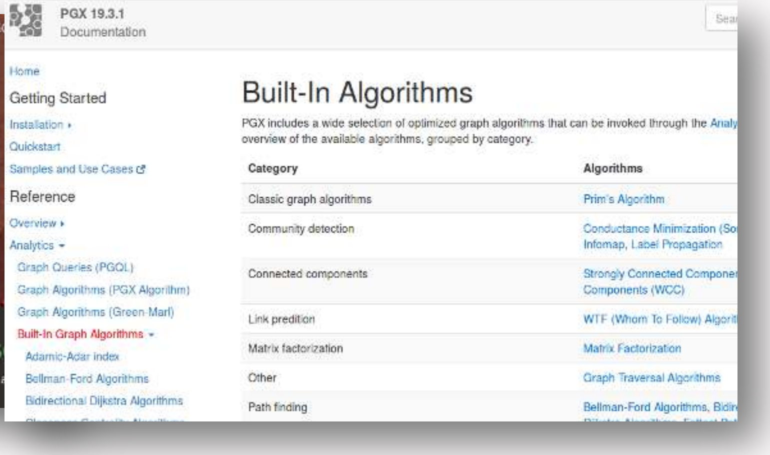
Property Graph Query Language

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.

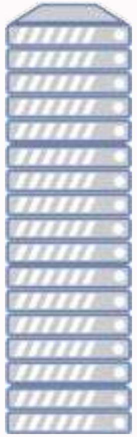
Category	Algorithms
Classic graph algorithms	Prim's Algorithm
Community detection	Conductance Minimization (So Infomap, Label Propagation
Connected components	Strongly Connected Componen Components (WCC)
Link predition	WTF (Whom To Follow) Algorit
Matrix factorization	Matrix Factorization
Other	Graph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

http://pgql-lang.org/

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

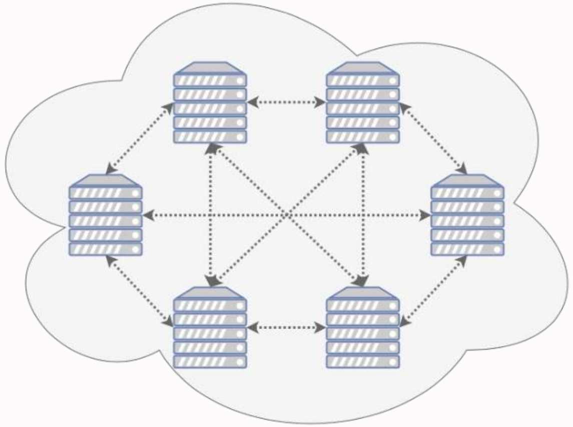
(1) single machine

PGX.SM
Java based



(2) distributed

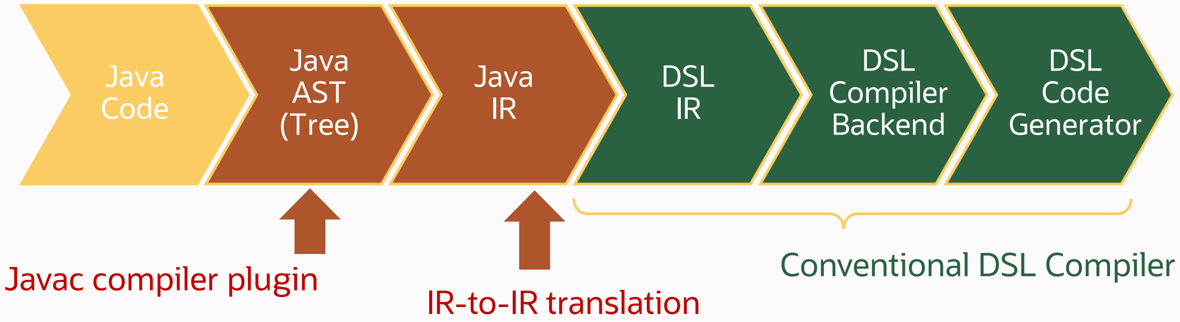
PGX.D
Scalable, cloud oriented
C++ based



(3) Database

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

PGX Algorithm [VLDB'16]



Java Code

Java AST (Tree)

Java IR

DSL IR

DSL Compiler Backend

DSL Code Generator

Javac compiler plugin

IR-to-IR translation

Conventional DSL Compiler

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 environments (e.g. distributed)

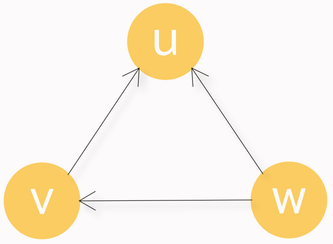
```
import com.oracle.pgx.api.beta.GraphAlgorithm;
import com.oracle.pgx.api.beta.PgxGraph;
import com.oracle.pgx.api.beta.VertexProperty;
import com.oracle.pgx.api.beta.annotations.Out;

@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 vertices accepting a lambda

Graph-friendly API



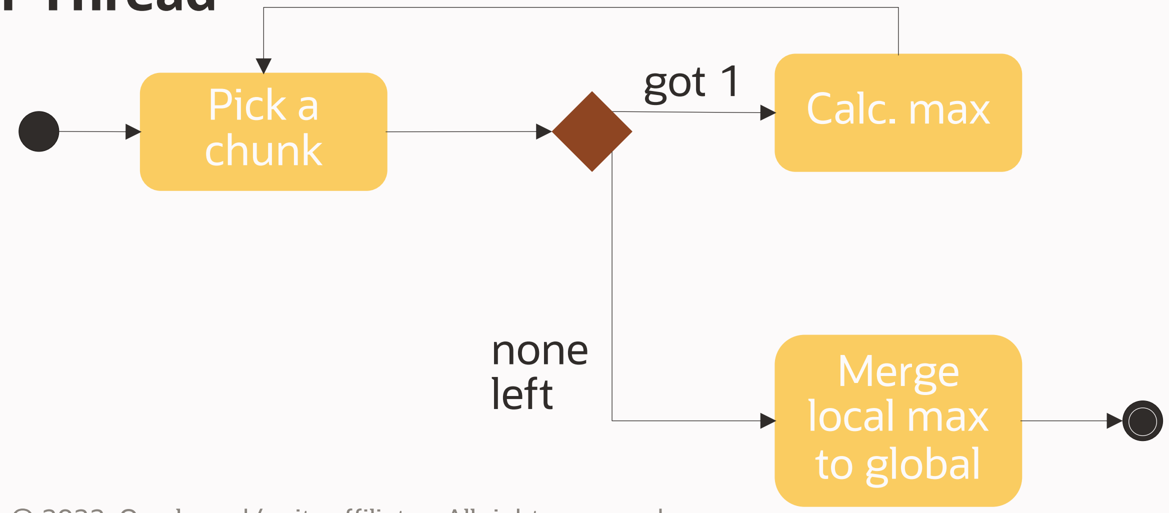
From Algorithm to Efficient Execution (PGX.SM)

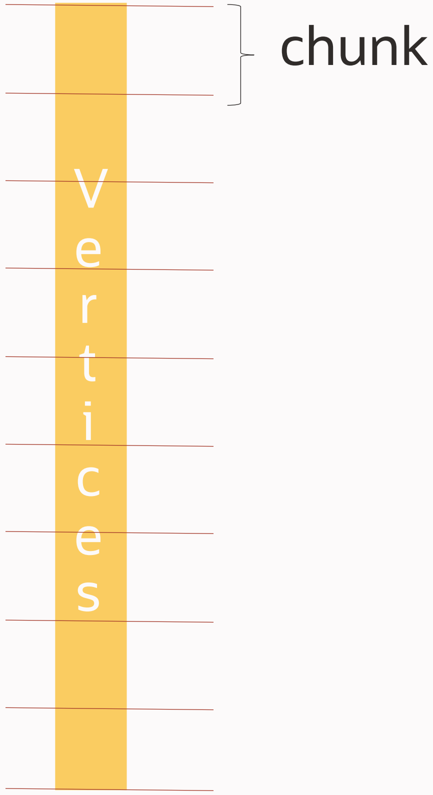
PGX algorithm is compiled to fast, parallel low-level code

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

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

Worker Thread

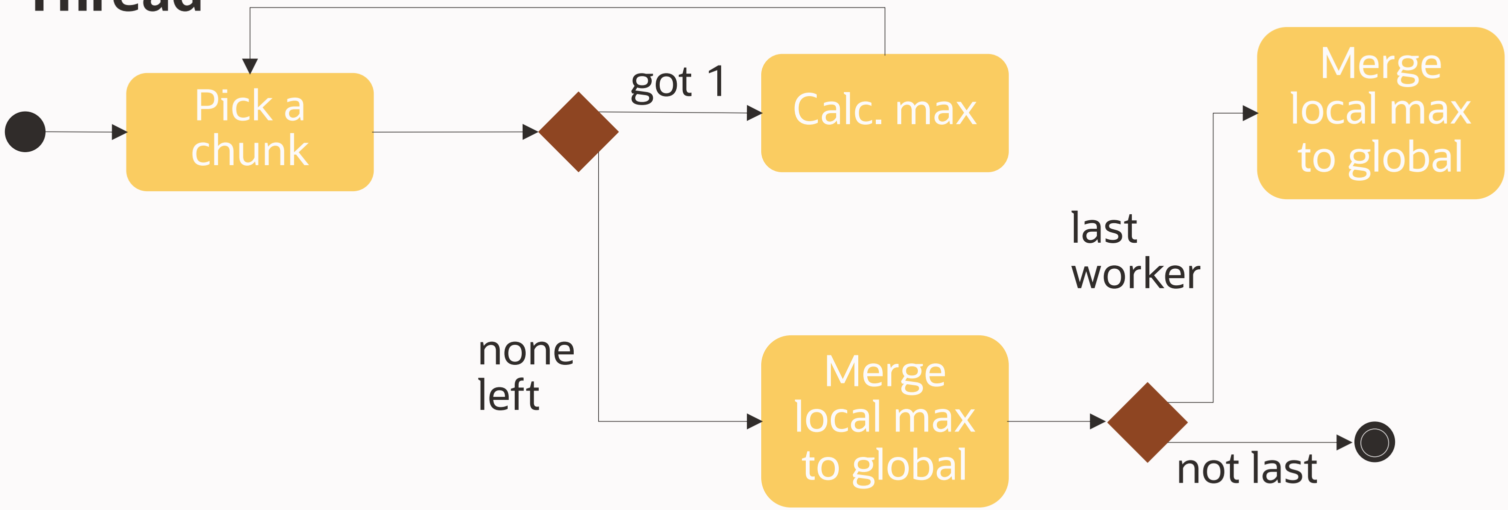


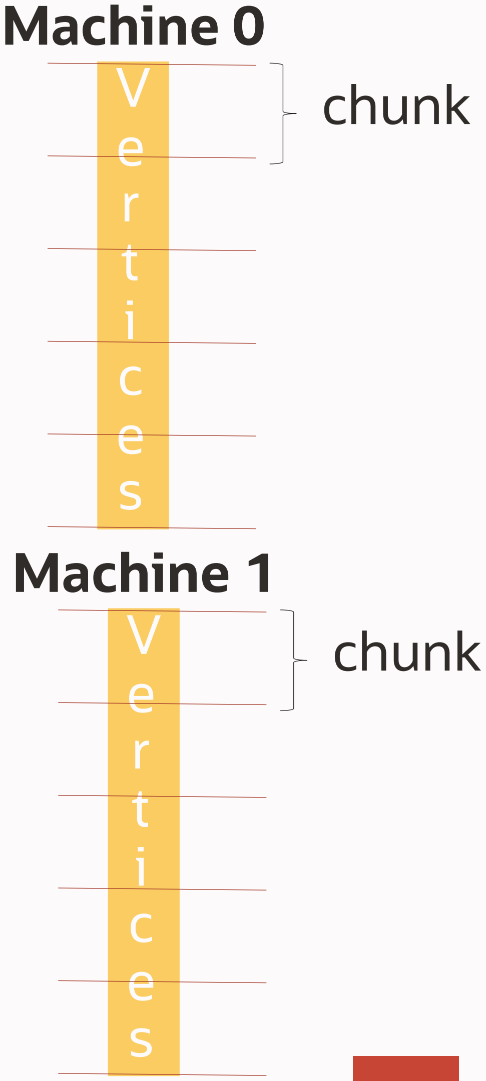


From Algorithm to Efficient Execution (PGX.D)

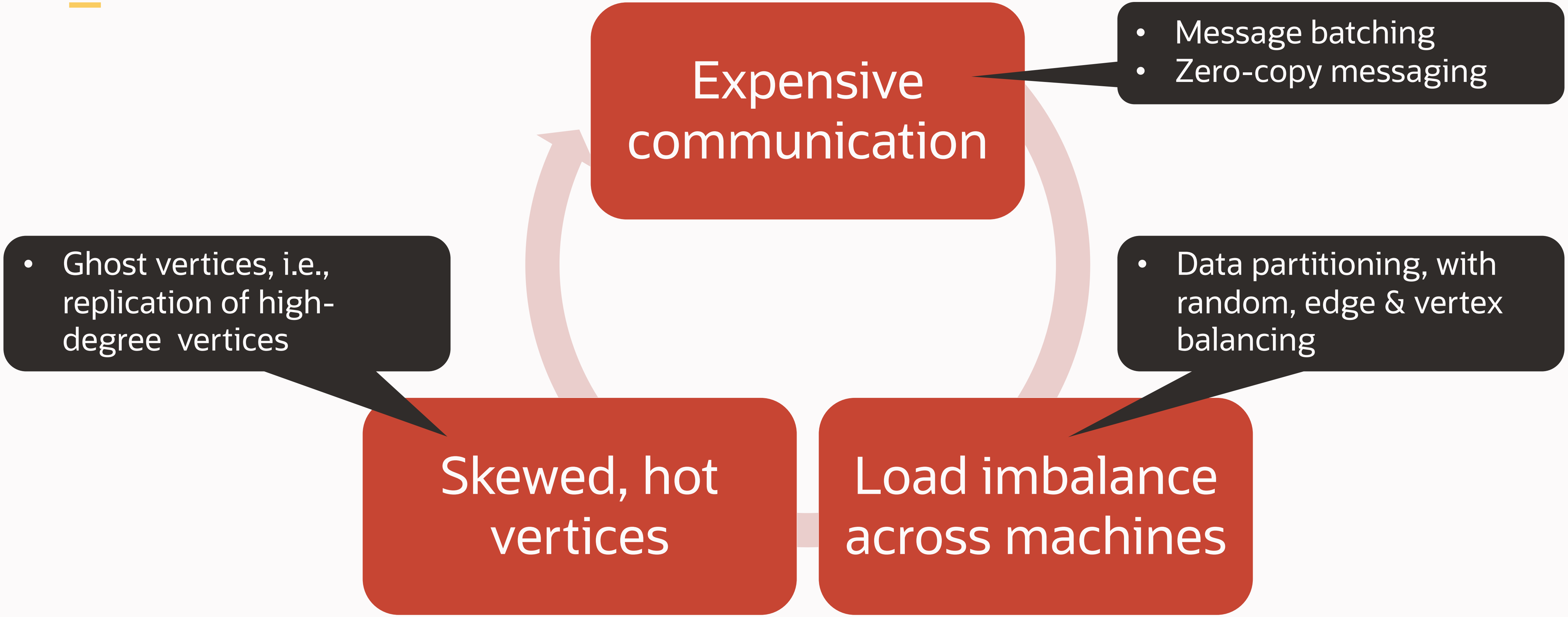
```
double max_degree(PgxGraph g) {
     double maxDegree;
     g.getVertices().forEach(n ->
          Reduction.updateMaxValue(maxDegree, n.getDegree())
     );
     return maxDegree;
}
```

Worker Thread





Key Challenges For Distributed Graph Analytics [SC' 15]



Expensive communication

- Message batching
- Zero-copy messaging

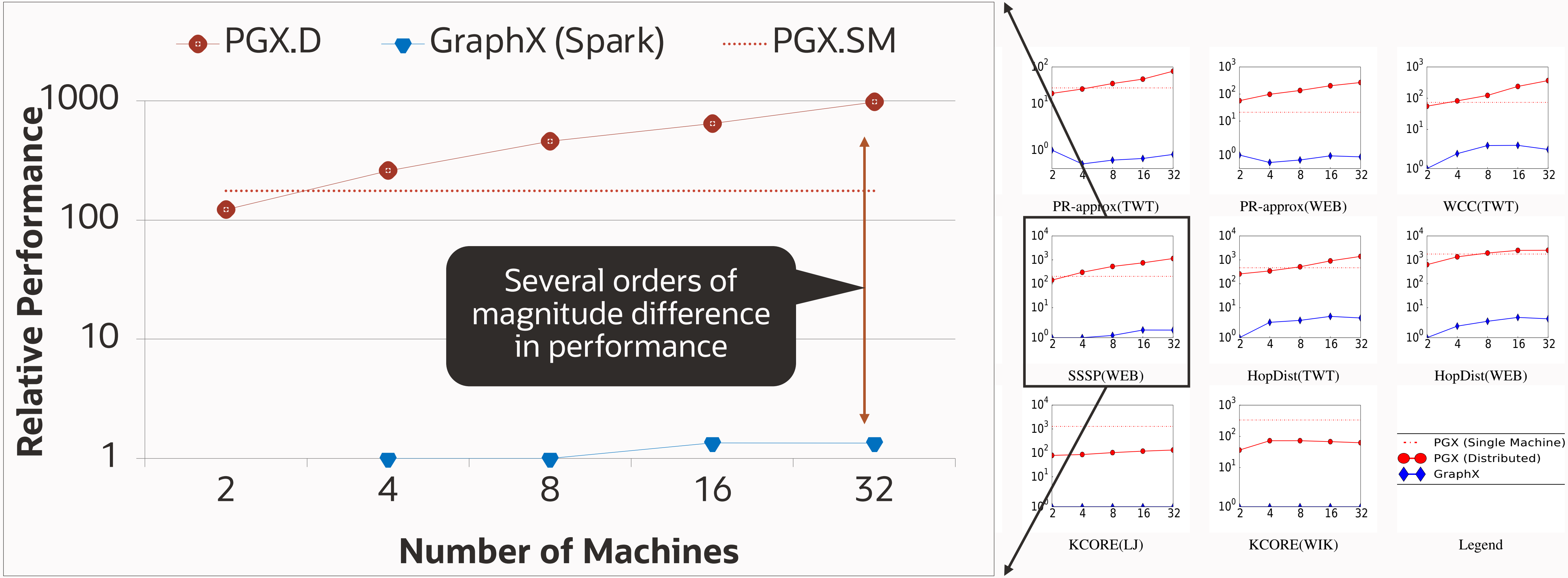
Skewed, hot vertices

- Ghost vertices, i.e., replication of high-degree vertices

Load imbalance across machines

- Data partitioning, with random, edge & vertex balancing

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 1.5
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

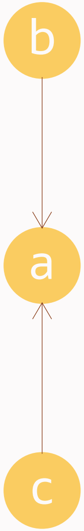
- Limited locality (especially with many machines)
- Do not want to do database JOINs

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

	PGX.SM	PGX.D
Analytics	BFS (Parallel for)	BFS (Bulk-synchronous)
Queries	BFS (Parallel for)	almost-DFS (Non-blocking)

Breadth-First vs. Depth-First Traversal Example

BFT



for all

1. Match all ‘b’

for all

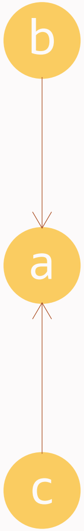
2. Match all ‘a’

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



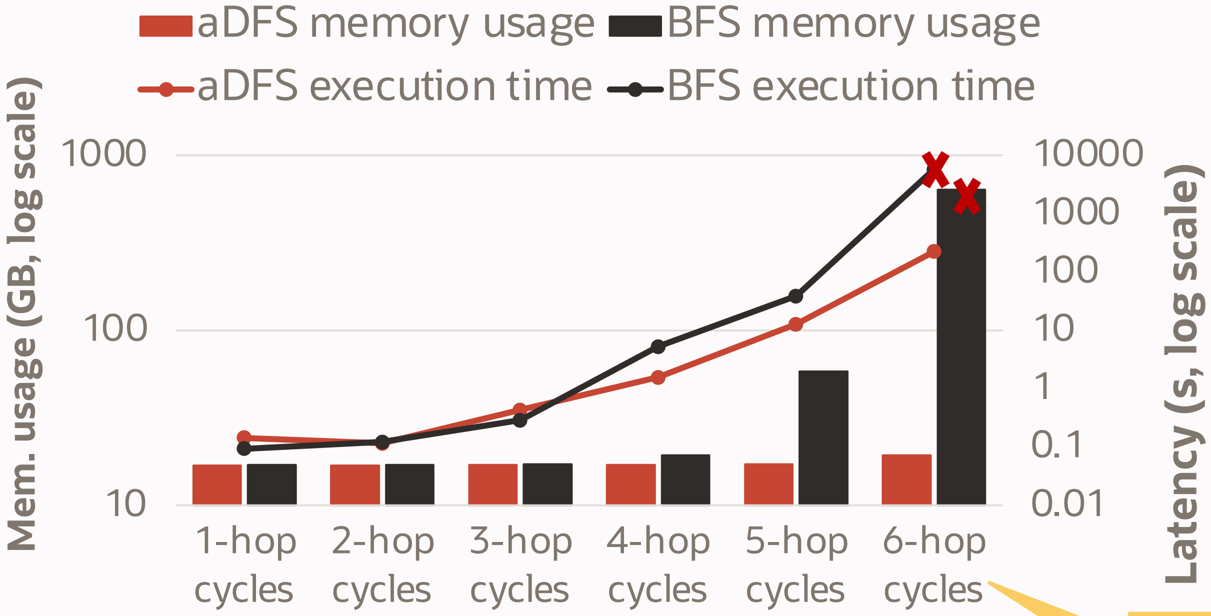
1. Match one ‘b’
2. Match one ‘a’
3. Match one ‘c’

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



(a)->(a)

…

(a)->(b)->(a)

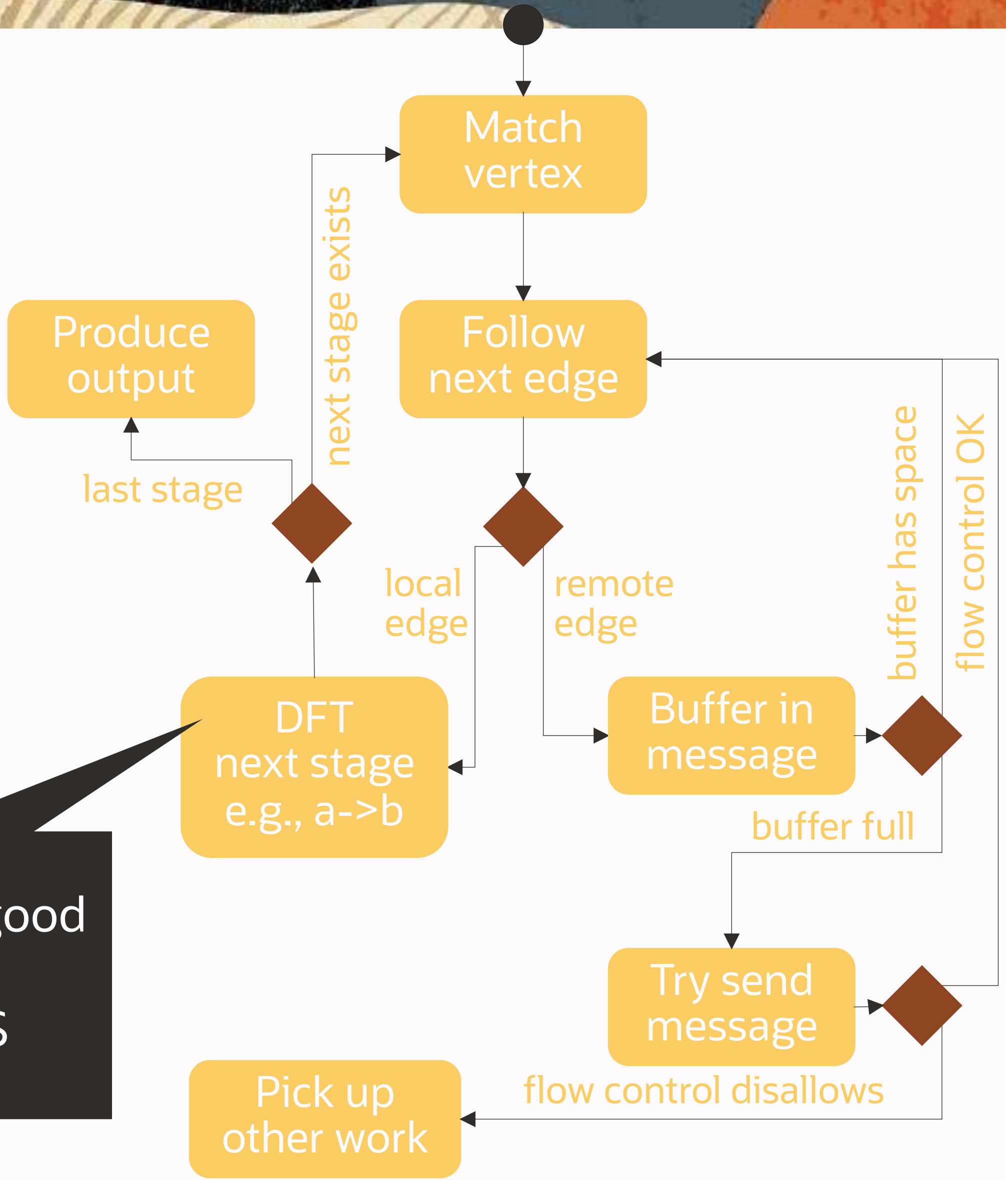
266 billion intermediate results

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

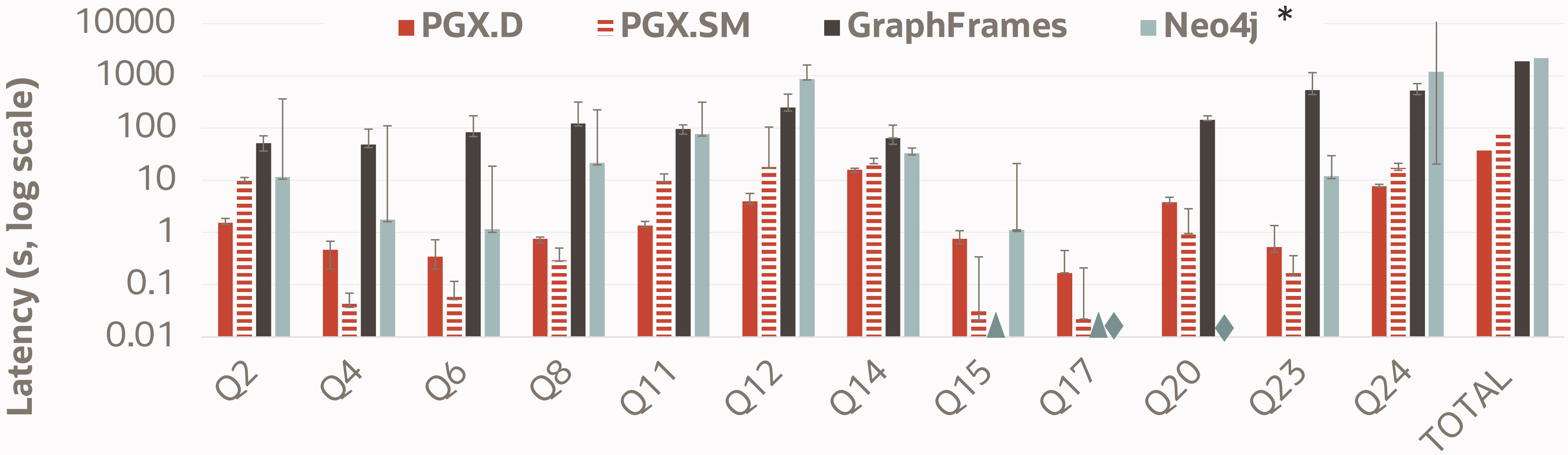
In-memory distributed execution with controllable network/memory usage

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**

Oracle Labs' Four Approaches to a Balanced Research Portfolio

Exploratory Research

- Pursue new ideas within domains relevant to Oracle

GraalVM™

Directed Research

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

ORACLE®
Database
Multilingual Engine

Consulting

- Provide unique expertise
- Small engagement across product organizations

Product Incubation

- Grow new products from Oracle Labs research

A global research team

Hundreds of researchers worldwide

Zurich: The biggest location with two floors at the Prime Tower (and growing!)

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, California, USA
- Austin, Texas, USA
- Belgrade, Serbia
- Brisbane, Australia
- … and more!

Selection of projects with involvement of the Zurich Lab

- **Parallel Graph AnalytiX (PGX)** – High-performance graph toolkit (single machine, distributed, in DB)
- **Data Studio (DS)** – Notebook technology for visualizing graphs and more
- **GraalVM** – A universal, polyglot VM environment
- **MultiLingual Engine (MLE)** – Bringing modern languages into the Oracle DB
- **Autonomous MiddleWare (AMW)** – Making middleware self-driving, self-patching & self-securing
- **BPF Linux Schedulers (BPF)** – OS scheduling for native Cloud applications

Several other topics across the other offices

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

If you are interested in Computer or Data Science, we have a great topic for you!

Check them out on labs.oracle.com/pls/apex/labs/r/labs/internships .

I initially joined **Oracle Labs for a short internship** where I was working on a distributed graph processing engine. I **designed and implemented major components** for the system in collaboration with well established Oracle Labs members and got the opportunity to **learn from very skilled people**. While I did enjoy the task, the highlight for me were the people. They were very **welcoming and helpful from the beginning** to the end of the internship. I ended up extending my internship and accepting a full time offer afterwards.

Irfan Bunjaku

ETH student, 6-month intern + MSc thesis
with Oracle Labs in 2022

Internships at Oracle Labs Zurich*

Regular internships or MSc theses

Typical duration of 3 to 12 months

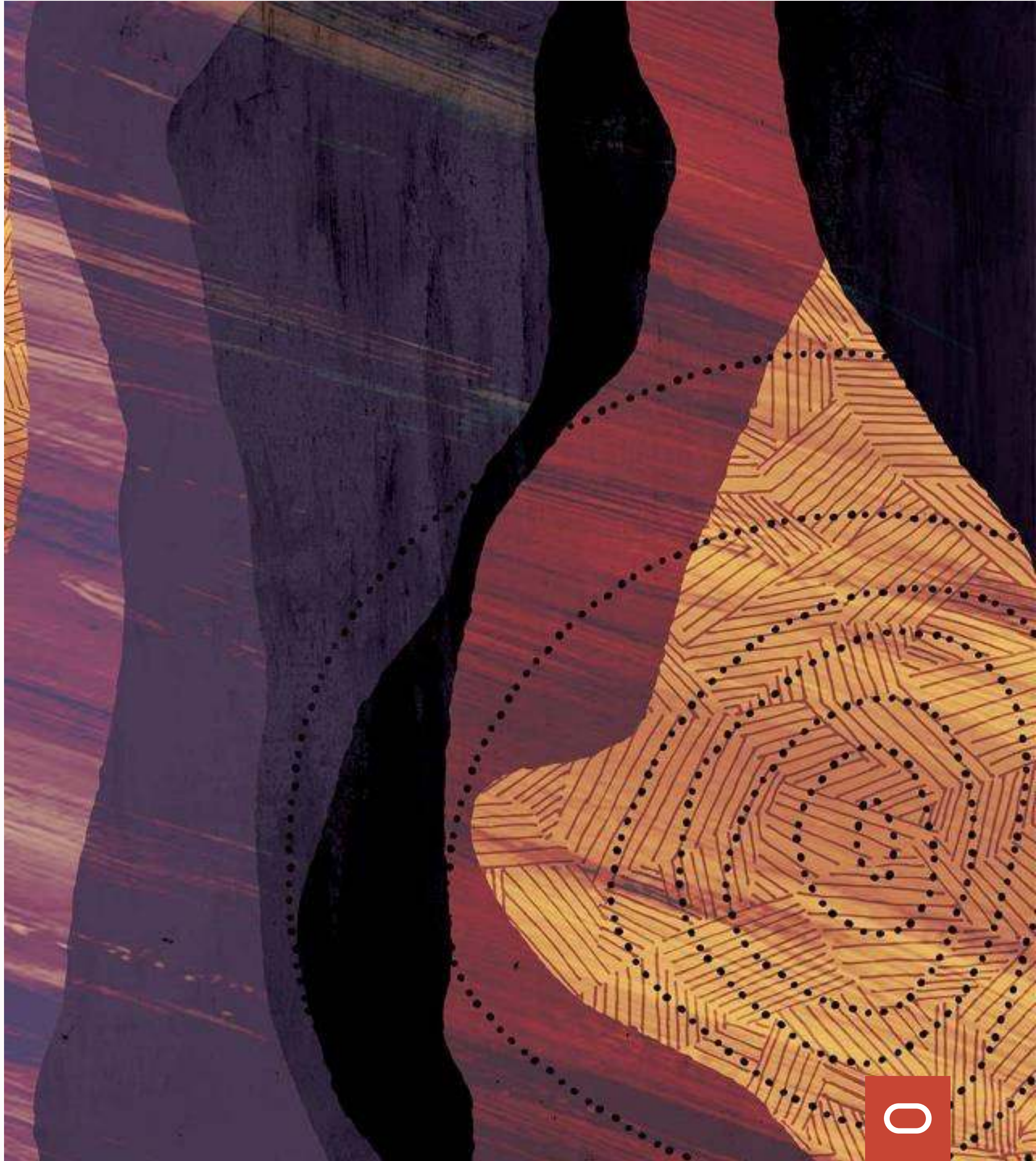
Competitive salary

Apply on
labs.oracle.com/pls/apex/labs/r/labs/internships
and/or contact us via lucas.braun@oracle.com!

*Back at the Prime Tower office since March 5th!

Thank you.

Also have a look at out our internship topics in the VIS Job Emails – we’d love to get your application.



ORACLE