Combining Graph and Machine Learning Technology using R

Hassan Chafi
Oracle Labs

Mark Hornick
Oracle Advanced Analytics

February 2, 2017

BIWA SUMMIT 2017
WITH SPATIAL SUMMIT

THE Big Data + Analytics + Spatial + Cloud + IoT + Everything Cool User Conference
January 31 - February 2, 2017

Copyright © 2017 Oracle and/or its affiliates. All rights reserved. |

Safe Harbor Statement

The following is intended to outline our research activities and 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Contents

- Graph Analysis and Machine Learning
  - Graph Analysis and Applications
  - Combining Graph Analysis with Machine Learning
- OAA.Graph
  - ORE/ORAAH and PGX
  - Integration
- Demo

ORACLE

Graph Analysis And Machine Learning

Big Data and Data Analysis

- The Big Data era is here
  - Volume
  - Velocity
  - Variety
- However, just storing and managing this data is not sufficient
  - Typically Big Data is low value per byte

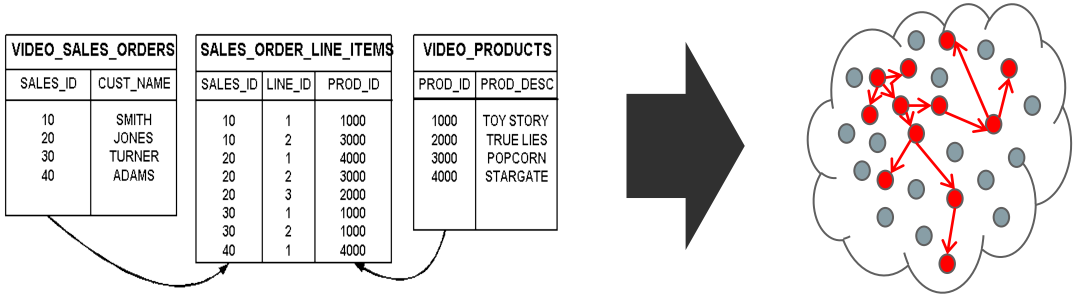
→ We want to get useful information out of the huge data sets

Methodologies:
- Classic OLAP
- Statistical analysis
- Machine learning
- Graph analysis

ORACLE

Graph Analysis

- A methodology in data analysis
- Represent your data as a graph
  - Data entities become nodes
  - Relationships become edges
- Analyze fine-grained relationships through the graph
  - Navigate multi-hop relationships quickly
  - Without computing expensive joins repeatedly

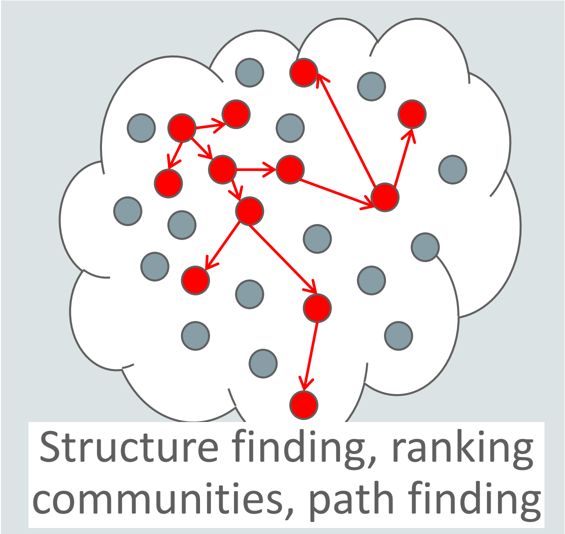


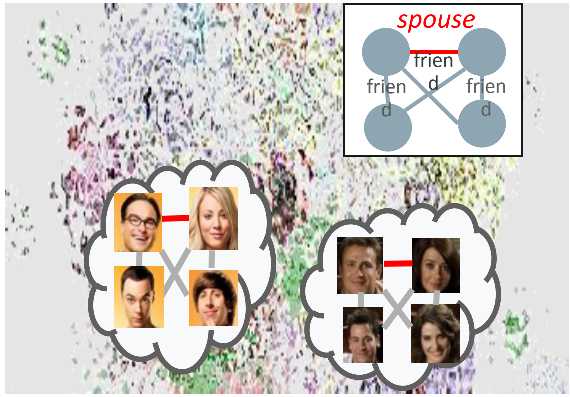
ORACLE

Graph Analysis

Inter-relationships between data and networks are growing in importance

- Graphs are everywhere
  - Facebook (friends of friends), Twitter, LinkedIn, etc.
    - Most data has inter-relationships that contain insights
- Two major types of graph algorithms
  - Computational Graph Analytics: Analysis of entire Graph
    - Influencer ID, community detect, patter machine, recommendations
  - Graph Pattern Matching
    - Queries that find sub-graphs fitting relationship patterns

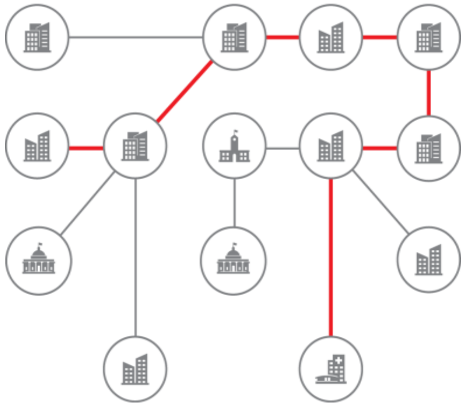




Graph Analysis Examples

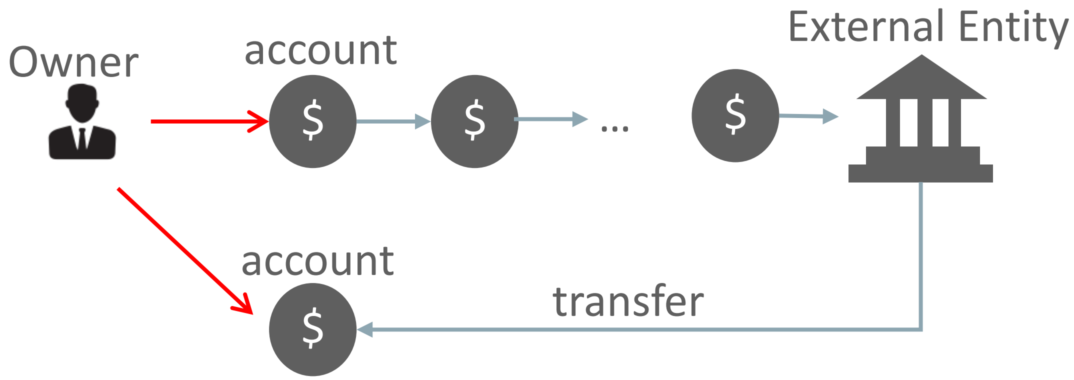
Reachability Analysis

Find out how data entities are connected with each other via multiple hops



- Example Application
  - Money laundering pattern detection in bank applications
  - Identify a chain of wire transfers, including an external entity, between two accounts of a single owner

➔ Graph pattern matching with cycle detection



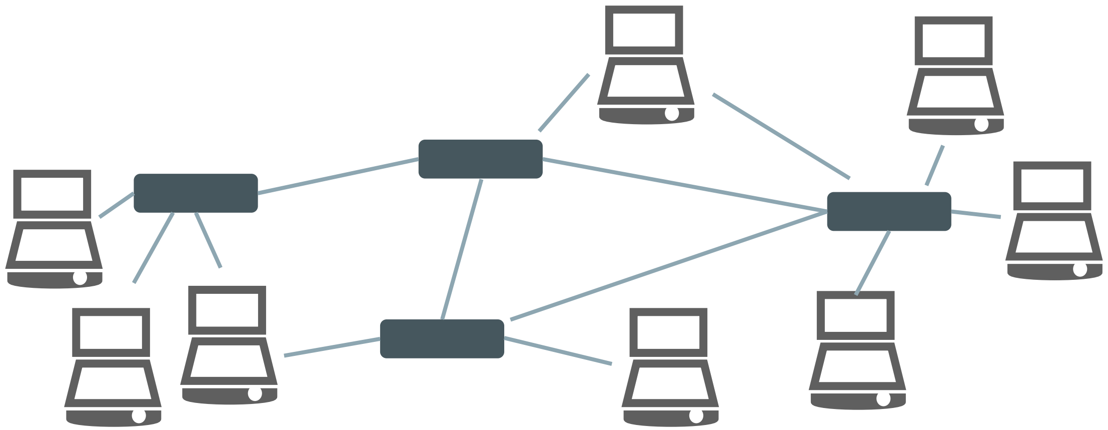
Graph Analysis Examples

Centrality Analysis

Identifying important entities from connections between data entities

- Example Application
  - Computer network vulnerability analysis
  - Identify network components whose failure would cause the largest damage

➔ *Betweenness Centrality* Computation

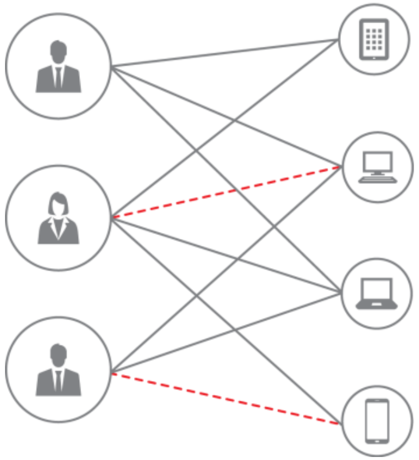


ORACLE

Graph Analysis Examples

Link Prediction

Identifying pairs of entities that are likely to have connections in future, due to their closeness or similarity

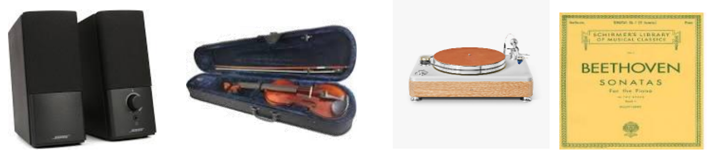


- Example Application
  - Product recommendation for retail
  - Given an item, identify *close* items from user-item or item-feature graph and recommend those items
  - Given a user, identify *close* users who purchased similar items and recommend items popular among those

➔ Matrix (Graph) Factorization, Personalized Pagerank, …

Items similar to this:

Other people also liked:

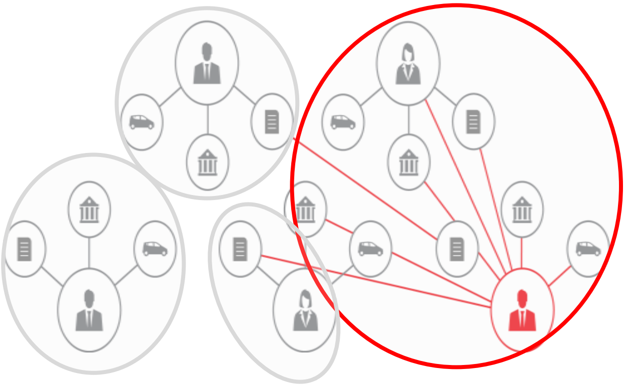


ORACLE

Graph Analysis Examples

Community Analysis

Identify grouping of data entities from their interconnection structure



- Example Application
  - Classification of data entities based on their relationship
  - E.g. classify students from the same department by the courses that they take

➔ Label propagation, Relax Map, …

Classification Result – For students of 'Math/CS department'

- Courses taken by community A

course_department	course_title	count
Dept of Math/CS	Data Structures	114
Dept of Math/CS	Computer Organization	104
Dept of Math/CS	Principles of Programming II	102
Dept of Math/CS	Database Design I	91
Dept of Math/CS	Operating Systems	89

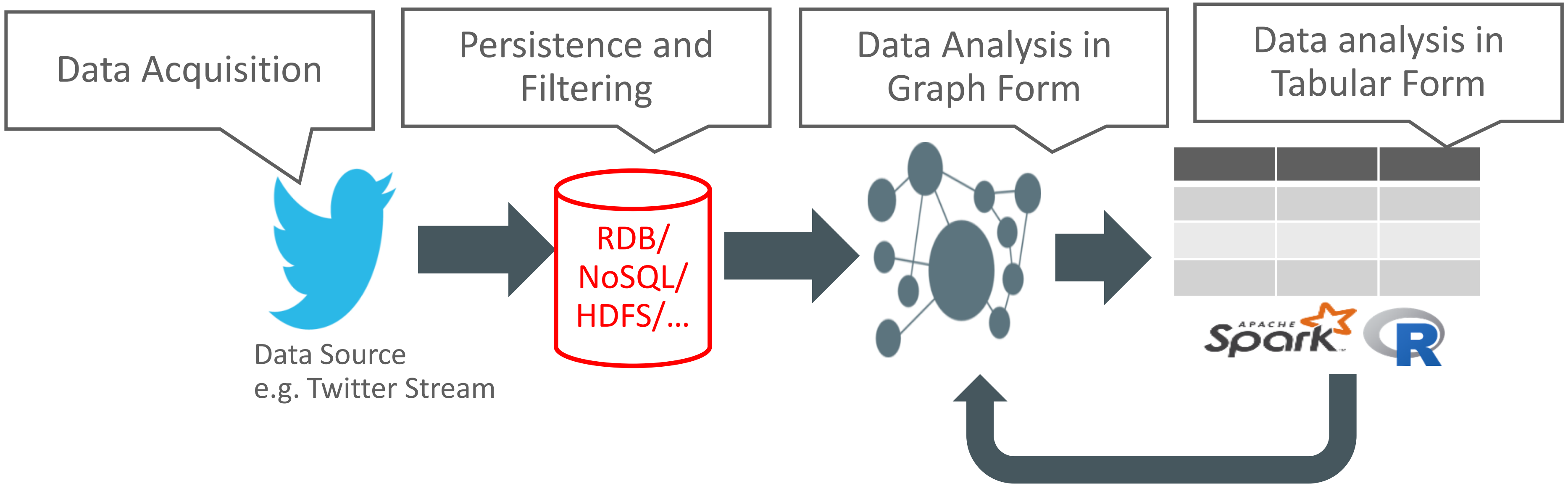
- Courses taken by community B

course_department	course_title	count
Dept of Math/CS	Ordinary Differential Equation	72
Dept of Math/CS	Set Theory	71
Dept of Math/CS	Probability and Statistics	65
Dept of Math/CS	Linear Algebra	55
Dept of Math/CS	Modern Algebra I	53

ORACLE

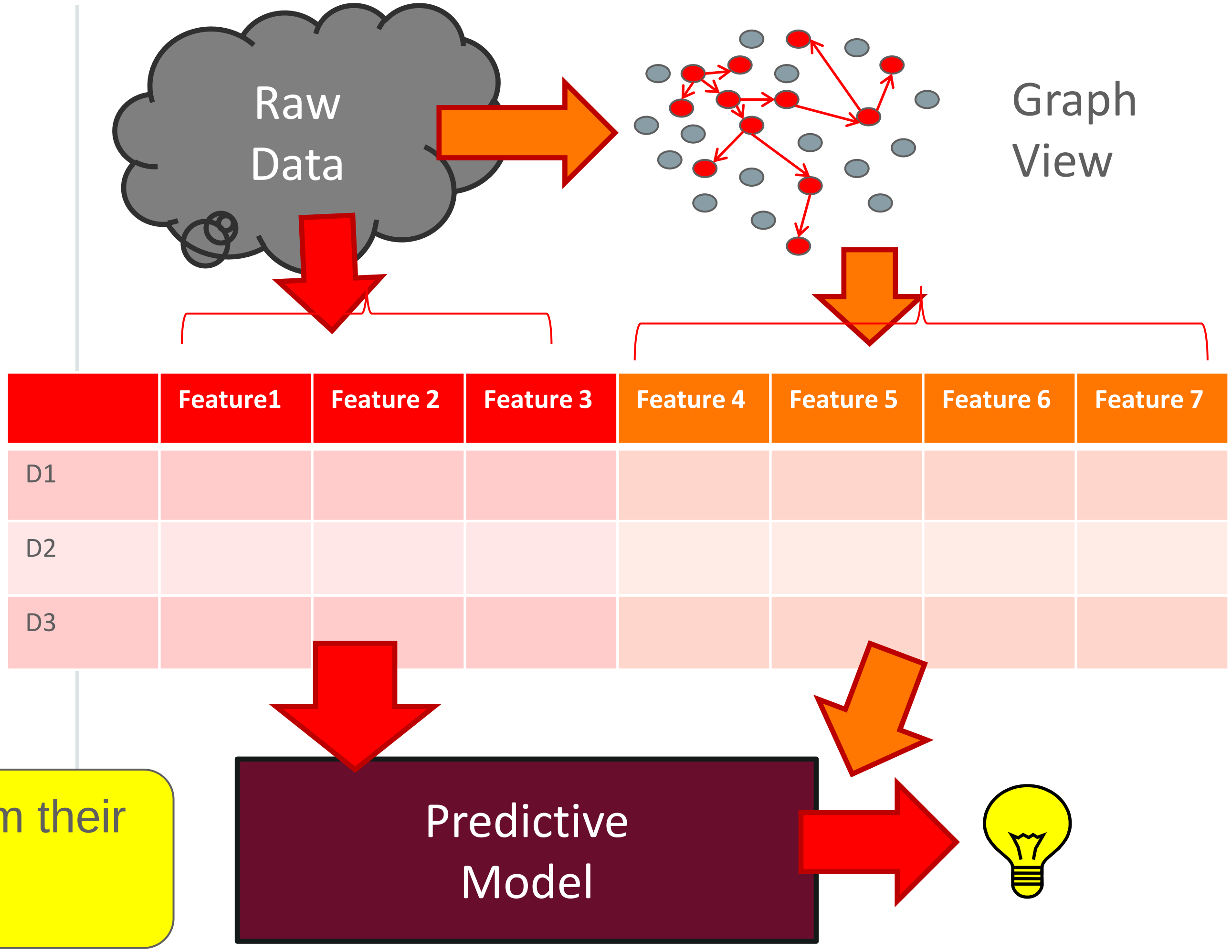
Graph Analysis and Other Data Analyses

- Naturally, graph analysis pairs well with other data analyses
  - Traditional analysis steps favors tabular data representation
  - Graph analysis can occur as a separate data processing step



Graph Analysis and Machine Learning

- Graph analysis can augment Machine Learning
  - Typical machine learning techniques create/train models based on observed features
  - Graph analysis can provide additional *strong* signals
  - That make predictions more accurate



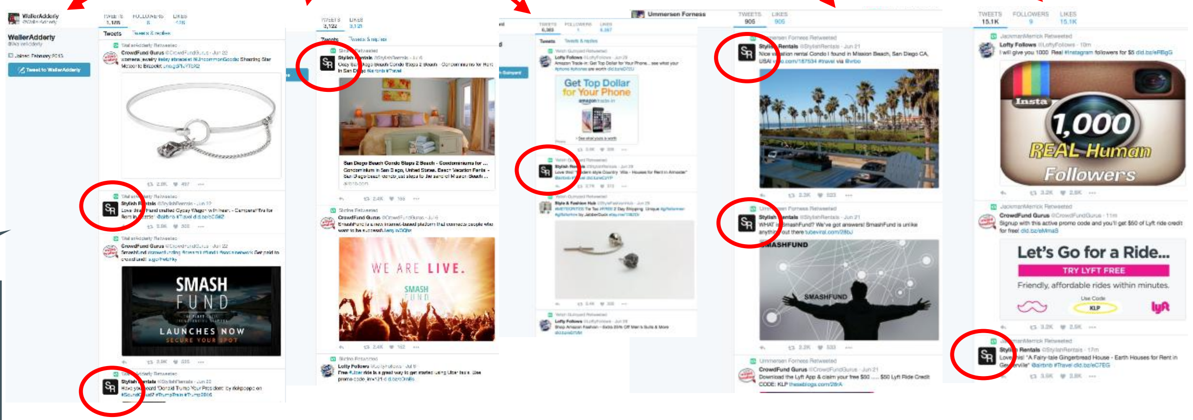
e.g. Can you identify groups of close customers from their call graph in order to predict customer churn?

Example – SNS Stream analysis

- Twitter streams can often be manipulated to achieve some goal
  - *Social and viral marketing (or alternative fact based news)*
  - True view on trends can be polluted by these streams
  - How can we eliminate such *noise*?

These are *bots* to make other accounts look more important

Different accounts re-tweet messages (mostly) from certain accounts



ORACLE

Data Processing Steps: Creating Graph

- Create graph representation from tables data set

authors

id	name	...
0x01	John	
0x02	Jane	
0x03	Bob	

messages

id	author_id	...
0x01	0x01	
0x02	0x02	
0x03	0x03	

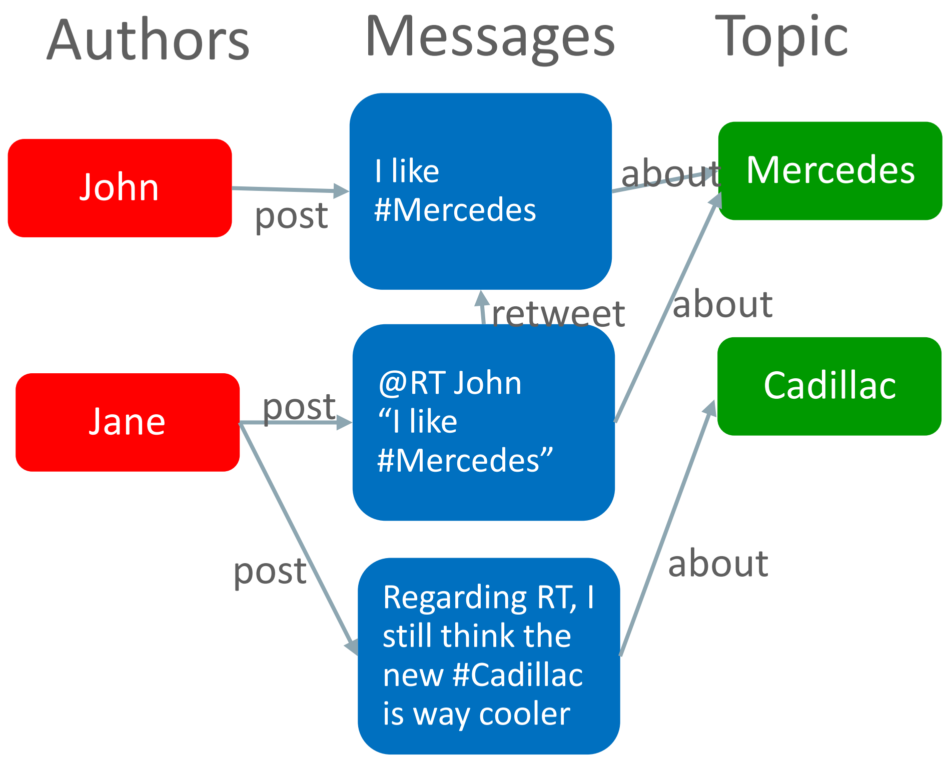
Message body

id	message_id	alaysis_group_id	snippet	...
0x01	0x01	0x01	I like #Mercedes	
0x02	0x02	0x01	RT @john "I like #...	
0x03	0x03	0x02	Regarding RT, I still ...	

topics

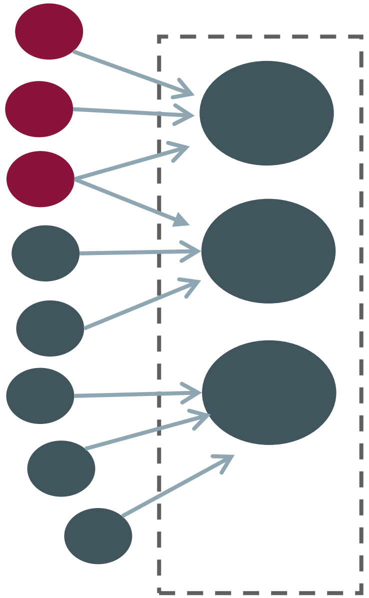
id	name	...
0x01	Mercedes	
0x02	Cadillac	

……

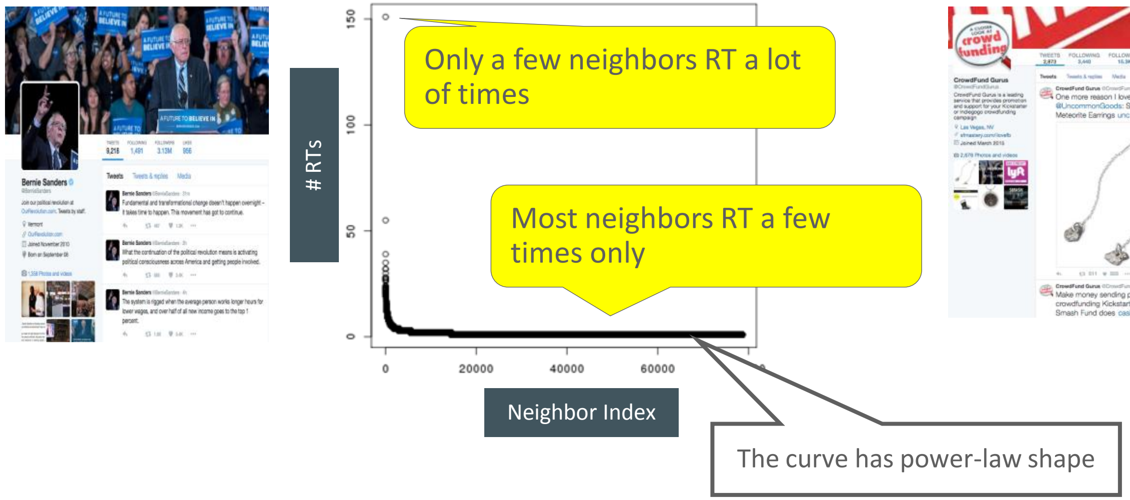


Data Processing Steps: Analyzing Graph

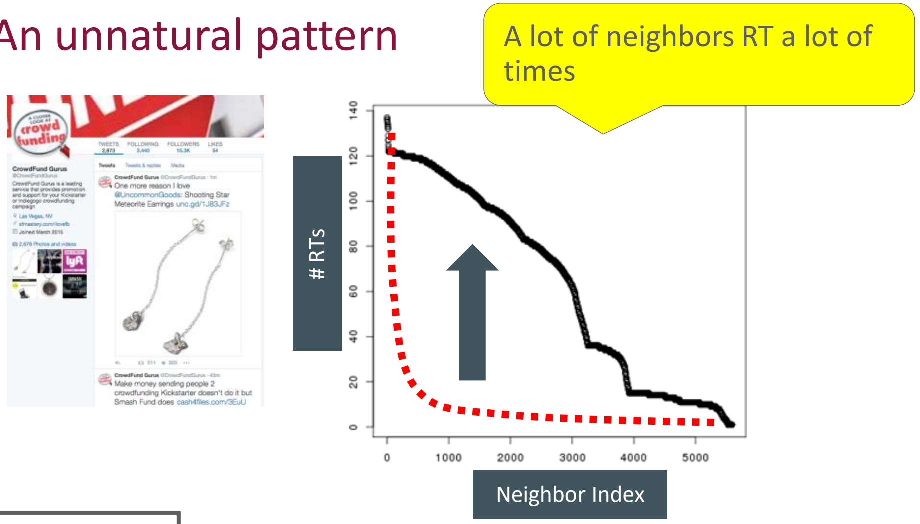
- Analyze Graph
  - Extract Retweet (RT) between accounts
  - Focus only on Total RT counts between accounts



A natural pattern



An unnatural pattern

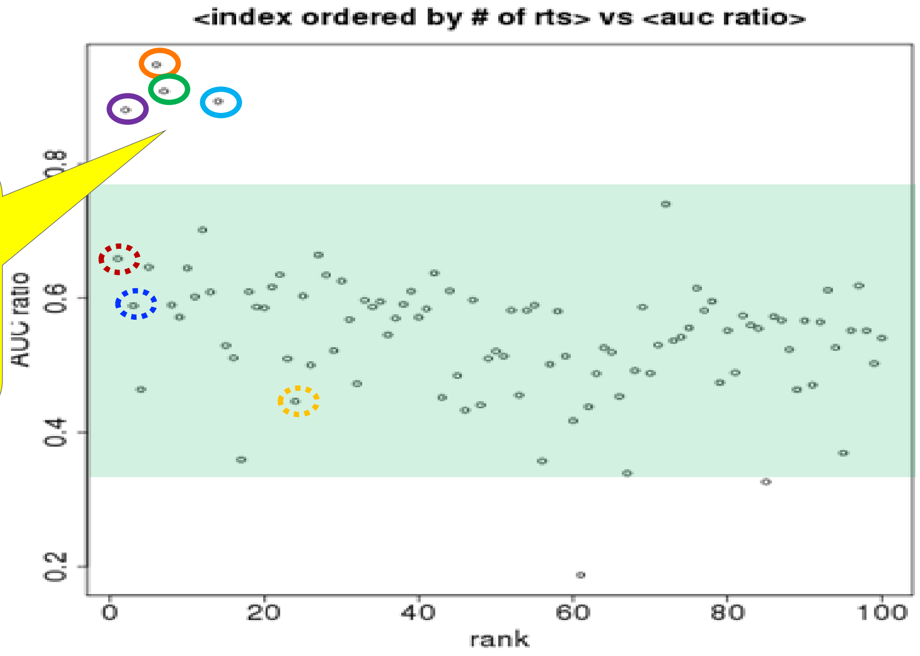


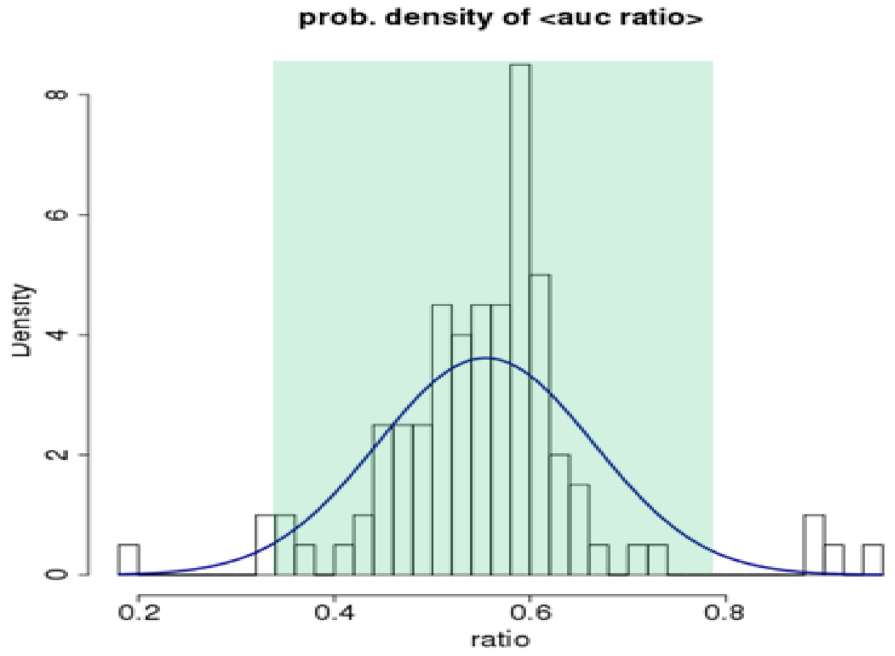
ORACLE

Data Processing Steps: Statistical Analysis

- Compute objective function for top-100 RT'ed accounts
- Identify anomalies from simple statistical methods

Certain accounts have obviously unnatural deviation

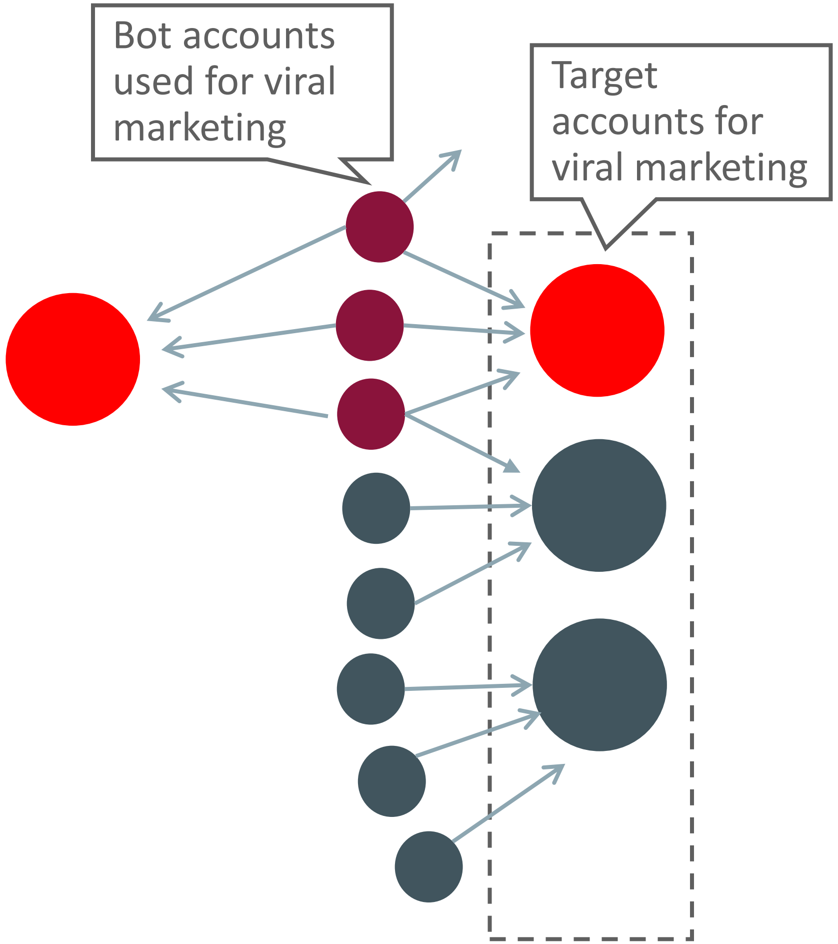




crowdFundGurus
carCrashesTV
LoftyFollows
StylishRentals
realDonalTrump
HillaryClinton
BernieSanders
$[\mu - 2\sigma, \mu + 2\sigma]$

Data Processing Steps: Analyzing Graph

- Resume from previous step's results
- Already identified targets (among top RT'ed accounts) and bots
- Analyze graph even further to identify more target accounts



Analysis Result

Data Set

- Data acquisition period: 2016-Feb (1 week)
- Number of topics: 675
- Number of messages: 2.6 million
- Number of accounts: 788,360

•# RT Bots: 3,092
•# RT Beneficiaries: 5
•# Removed messages: 551,177

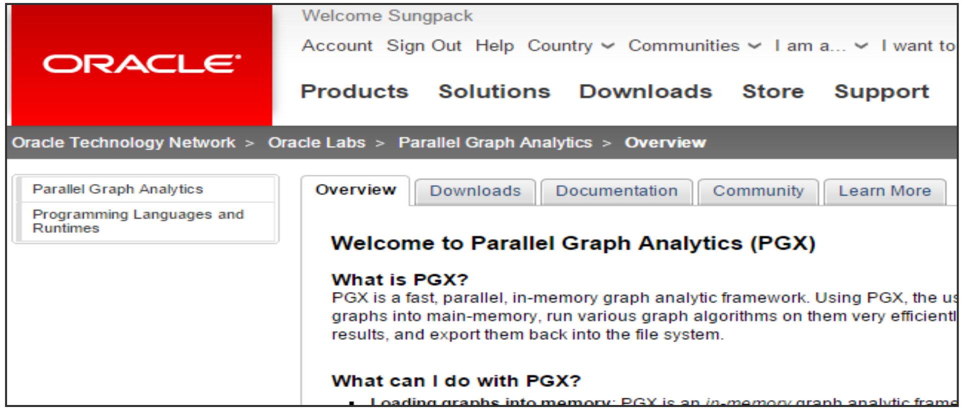
➔ Significant changes in important accounts and trends

ORACLE

What is PGX (part of BDSG)?

- PGX (Parallel Graph Analytics)
  - An in-memory graph analysis engine
  - Originated from Oracle Labs
  - Provides fast, parallel graph analysis
    - Built-in Algorithm Packages
    - Graph Query (Pattern-Matching)
    - Custom Algorithm Compilation (Advanced Use case)
  - Integrated with Oracle Product(s)
    - Oracle Big Data Spatial and Graph (with BDA)
    - Property Graph Support at RDBMS 12.2c (Planned)
  - 35+ graph algorithms
  - Exceeds open source tool capabilities

OTN Technology Preview



Oracle Big Data and Spatial and Graph

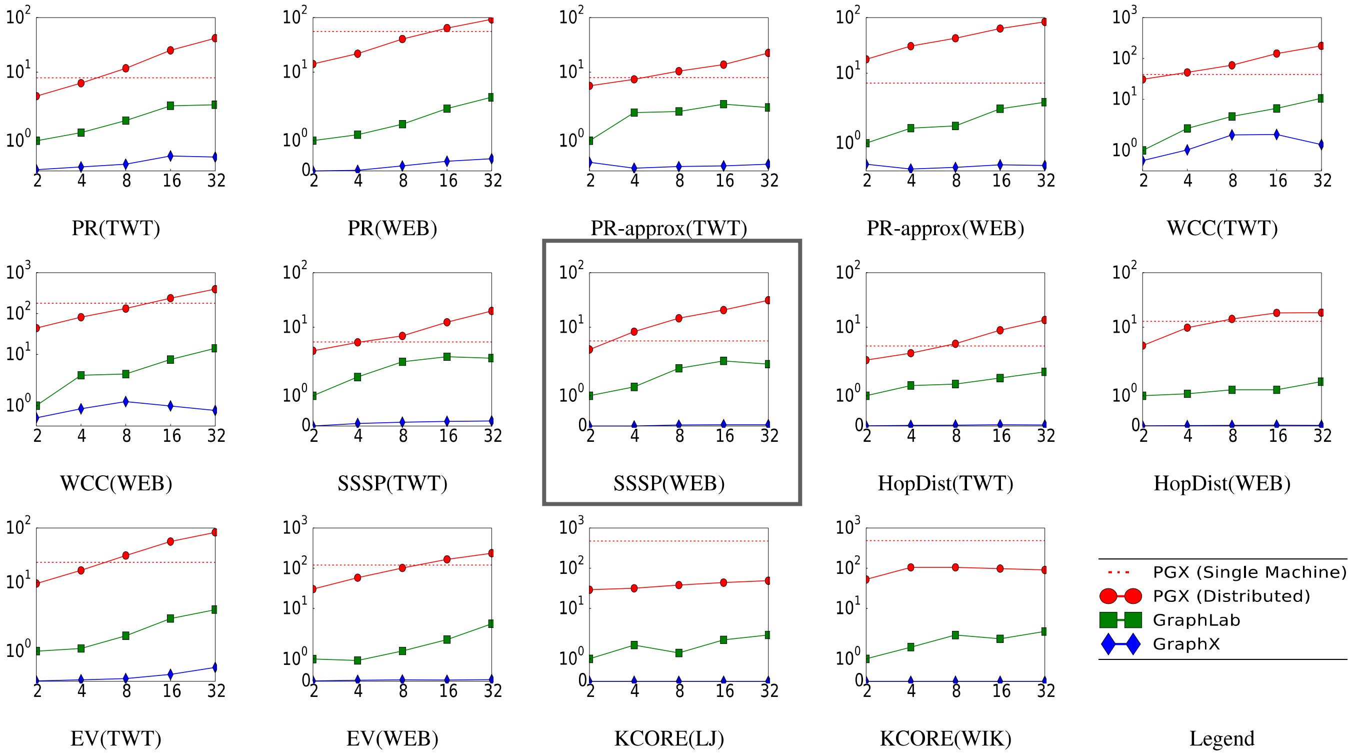
PGX Graph Algorithms

- Ranking
  - Pagerank (+ variants)
  - Vertex Betweenness Centrality (including approximations)
  - Closeness Centrality
  - Eigenvector Centrality
  - Degree Centrality
  - Hyperlink-Induced Topic Search (HITS)
- Path Finding
  - Dijkstra (+ variants)
  - Bellman Ford (+ variants)
  - Hop Distance (+ variants)
  - Fattest path
- Partitioning
  - Weakly and Strongly Connected Components
  - Conductance and Modularity
  - Community Detection
- Recommendation
  - Twitter's whom-to-follow
  - Matrix Factorization
- Other
  - Breadth First Search with filter
  - Triangle Counting
  - Degree Distribution
  - K-core
  - Adamic Adar

ORACLE

PGX Performance (Algorithm Computation)

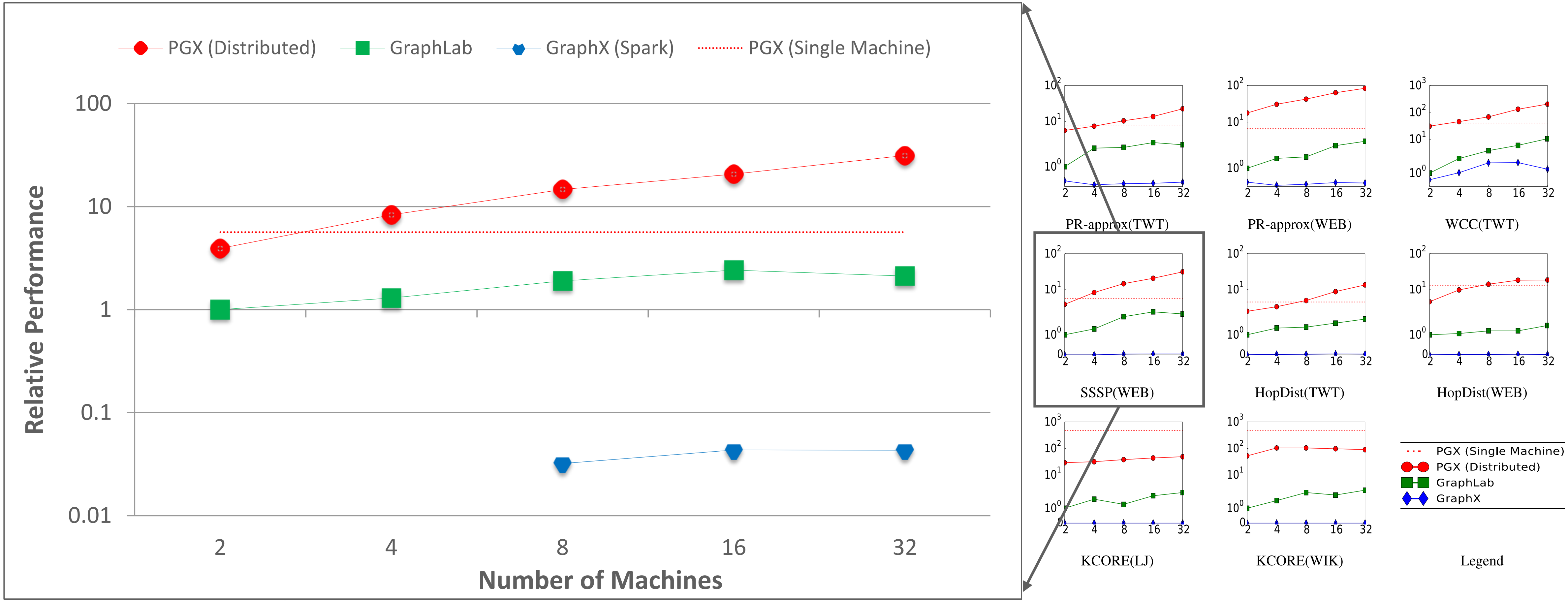
- Comparisons against existing graph engines
  - GraphX (Spark)
  - GraphLab (Dato)
- With seven popular algorithms
  - Pagerank (exact and approx), Weakly Connected Components , Single-Source Shortest Path, Hop-Distance (BFS), Eigen Vector, K-Core
- On Two Graph instances
  - Twitter Graph (TWT): 41 million nodes, 1.4 Billion edges
  - Web Graph (WEB): 77 millions nodes, 2.9 Billion edges




```
Hardware:       Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Network:        Melanox Infiniband (56Gbps)
```

ORACLE

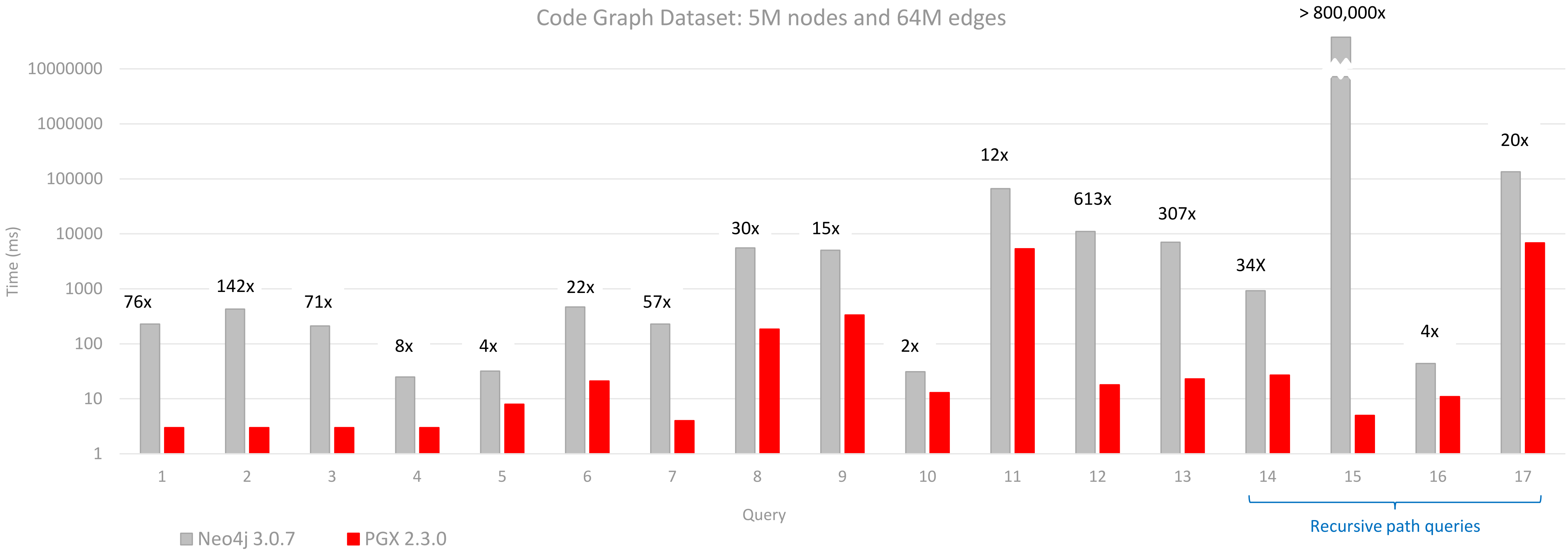
PGX Performance (Algorithm Computation)




```
Hardware:      Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Network:       Melanox Infiniband (56Gbps)
```

ORACLE

PGX Performance (Query) vs. Neo4j



Numbers shown are “hot” numbers:
Neo4j is no longer accessing disk.

```
Hardware:      88 x Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz - 256 RAM
Warmup:        ignore the first two runs, measure the third run
Index usage:   no special indexes (use defaults)
```

ORACLE

PGX (Single Node) Performance on SPARC

SPARC M7 up to 1.5x faster per core than x86

Graph Algorithm	*Workload Size*	4-chip X86 E5 v3	4-chip SPARC T7-4	SPARC per chip Advantage	SPARC per core Advantage
SSSP Bellman-Ford	448M vertices, 17.2B edges	39.2s	14.7s	2.7x	**1.5x**
	233M vertices, 8.6B edges	21.3s	8.5s	2.5x	**1.4x**
PageRank	448M vertices, 17.2B edges	136.7s	62.6s	2.2x	**1.2x**
	233M vertices, 8.6B edges	72.1s	27.6s	2.6x	**1.5x**

- Graph computations accelerated by SPARC's memory bandwidth
  - Bellman-Ford/SSSP (single-source shortest path) – optimal route or connection
  - PageRank - measuring website importance

ORACLE

OAAgraph

An R interface integrating PGX and ORE/ORAAH for Machine Learning

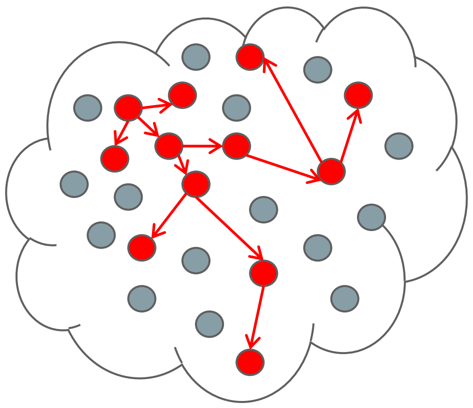
ORACLE®

Why an R interface to Graph?

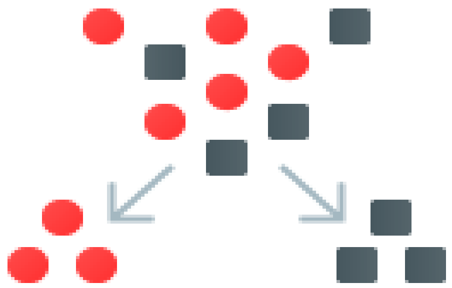
- Single, unified interface across complementary technologies
  - Work with R data.frames and convenient functions across ML and graph
  - Results returned as R data.frames allows further processing in R env
- R users take advantage of multiple, powerful technologies
  - Highly scalable PGX engine on both Oracle Database and Hadoop
  - Integrated with **Oracle R Enterprise**, part of Oracle Database Advanced Analytics option
  - Integrated with **Oracle R Advanced Analytics for Hadoop**, part of Oracle Big Data Connectors

ORACLE

Graph Analytics



Machine Learning



Compute graph metric(s)

Add to structured data →

Build predictive model using graph metric

Explore graph or compute new metrics using ML result

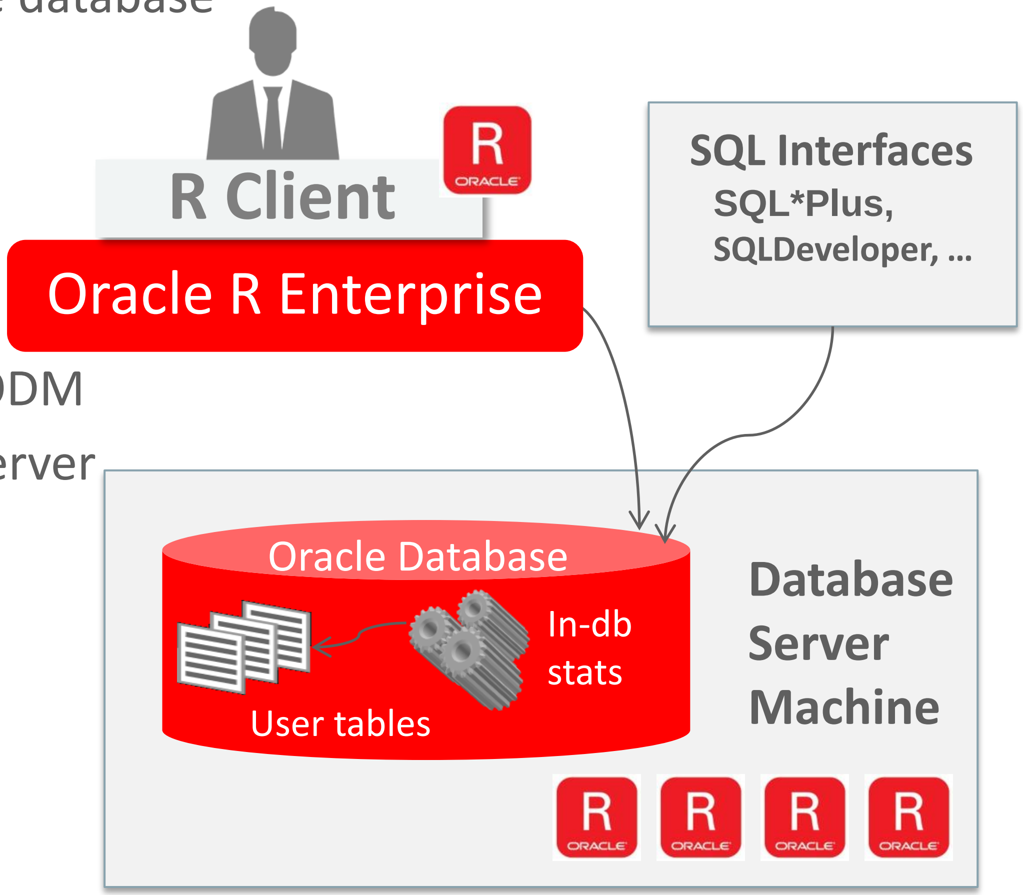
← Add to graph

Build model(s) and score or classify data

ORACLE

Oracle R Enterprise

- Use Oracle Database as a high performance compute environment
- Transparency layer
  - Leverage proxy objects (ore.frames) - data remains in the database
  - Overload R functions that translate functionality to SQL
  - Use standard R syntax to manipulate database data
- Parallel, distributed algorithms
  - Scalability and performance
  - Exposes in-database machine learning algorithms from ODM
  - Additional R-based algorithms executing and database server
- Embedded R execution
  - Store and invoke R scripts in Oracle Database
  - Data-parallel, task-parallel, and non-parallel execution
  - Use open source CRAN packages



OAA / Oracle R Enterprise 1.5

Machine Learning algorithms in-Database

…plus open source R packages for algorithms in combination with embedded R data- and task-parallel execution

Classification
- Decision Tree
- Logistic Regression
- Naïve Bayes
- Support Vector Machine
- RandomForest

Clustering
- Hierarchical k-Means
- Orthogonal Partitioning Clustering

Market Basket Analysis
- Apriori – Association Rules

Regression
- Linear Model
- Generalized Linear Model
- Multi-Layer Neural Networks
- Stepwise Linear Regression
- Support Vector Machine

Attribute Importance
- Minimum Description Length

Feature Extraction
- Nonnegative Matrix Factorization
- Principal Component Analysis
- Singular Value Decomposition

Anomaly Detection
- 1 Class Support Vector Machine

Time Series
- Single Exponential Smoothing
- Double Exponential Smoothing

ORACLE

Oracle R Advanced Analytics for Hadoop

Using Hadoop/Hive/Spark Integration, plus R Engine and Open-Source R Packages

Oracle R Advanced Analytics for Hadoop (ORAAH) on Hadoop Cluster

R interface to HQL Basic Statistics, Data Prep, Joins and View creation

HQL

HIVE

Parallel, distributed algorithms:

- ORAAH Spark algorithms: Deep Neural Network, GLM, LM
- Spark MLlib algorithms: LM, GLM, LASSO, Ridge Regression, Decision Trees, Random Forests, SVM, k-Means, PCA

Use of Open-source R packages via custom R Mappers / Reducers

R Client

R Analytics
Oracle R Advanced Analytics for Hadoop

Oracle Database
with Advanced Analytics option

SQL

Oracle Advanced Analytics

SQL Client

SQL Developer
Other SQL Apps

ORACLE

Oracle R Advanced Analytics for Hadoop 2.7.0
Machine Learning algorithms

Classification
- GLM ORAAH — hadoop MapReduce
- Logistic Regression ORAAH — Spark
- Logistic Regression — Spark MLlib
- Random Forests — Spark MLlib
- Decision Trees — Spark MLlib
- Support Vector Machines — Spark MLlib

Clustering
- Hierarchical k-Means — hadoop MapReduce
- Hierarchical k-Means — Spark MLlib
- Gaussian Mixture Models — Spark MLlib

Regression
- MLP Neural Networks ORAAH — Spark
- LASSO — Spark MLlib
- Ridge Regression — Spark MLlib
- Support Vector Machines — Spark MLlib
- Random Forest — Spark MLlib
- Linear Regression — Spark MLlib

Basic Statistics
- Correlation/Covariance — hadoop MapReduce

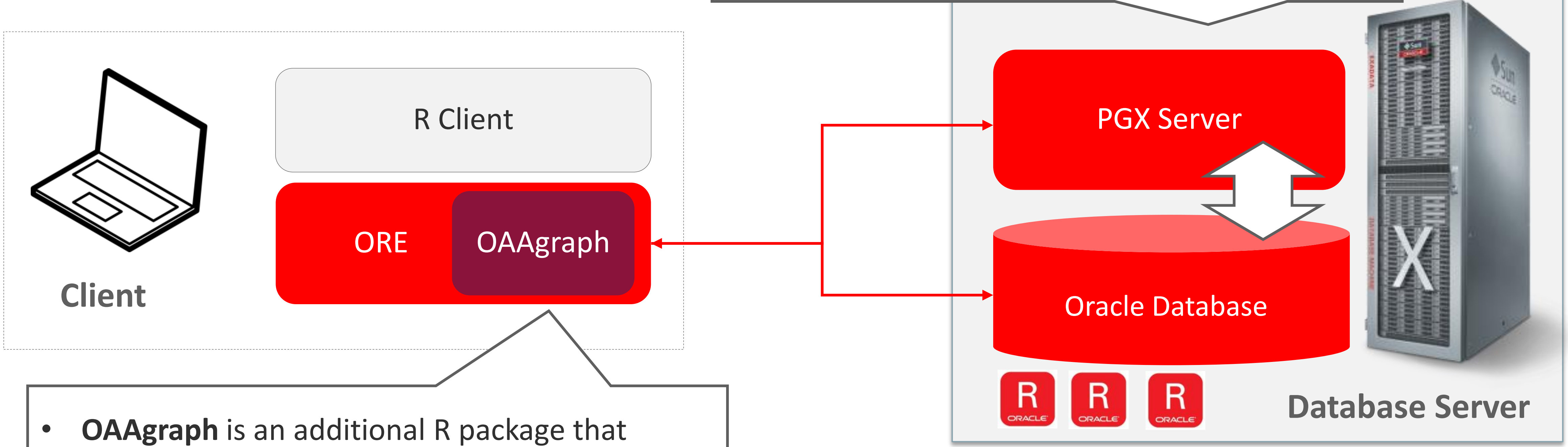
Feature Extraction
- Non-negative Matrix Factorization — hadoop MapReduce
- Collaborative Filtering (LMF) — hadoop MapReduce
- Singular Value Decomposition — Spark MLlib

Attribute Importance
- Principal Components Analysis — hadoop MapReduce
- Principal Components Analysis — Spark MLlib

ORACLE

OAAgraph Architecture with Oracle Database

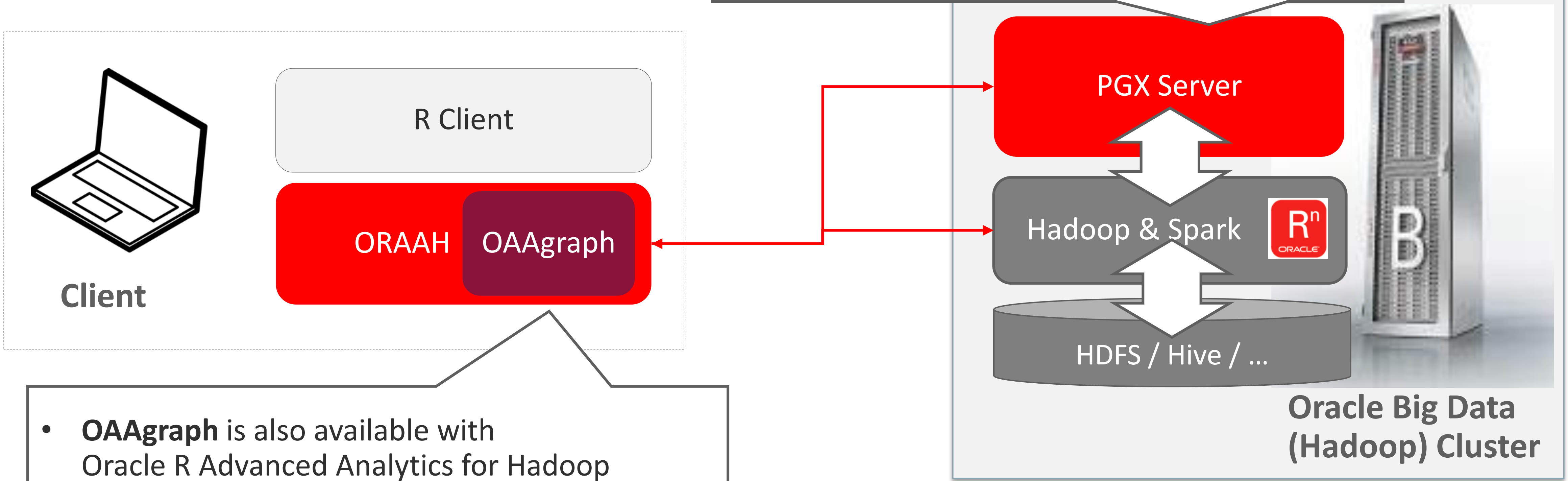
- **OAAgraph** gives remote control of PGX server
- PGX loads graph from database (ore.frames)



- **OAAgraph** is an additional R package that comes with Oracle R Enterprise

OAAgraph Architecture with Spark/Hadoop

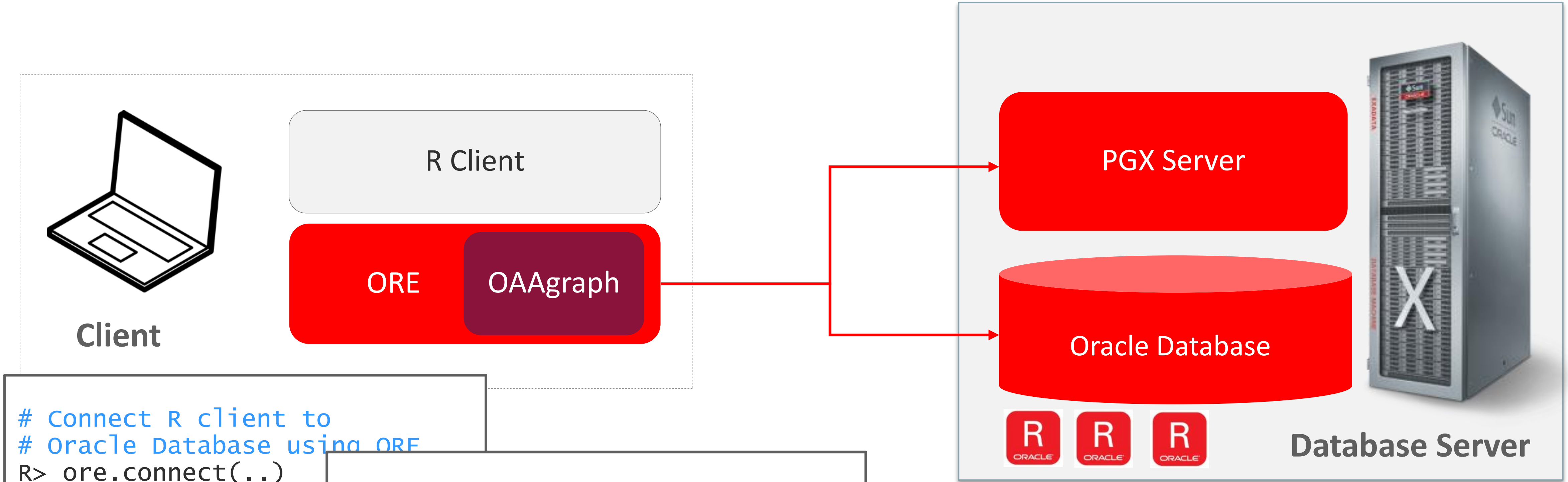
- **OAAgraph** gives remote control of PGX server
- PGX loads graph via SPARK data frames



- **OAAgraph** is also available with Oracle R Advanced Analytics for Hadoop

Execution Overview (ORE)

- Initialization and Connection




```
# Connect R client to
# Oracle Database using ORE
R> ore.connect(..)
```

```
# Connect to PGX server
# using OAAgraph
R> oaa.init(..)
R> oaa.graphConnect(...)
```

Execution Overview (ORE)

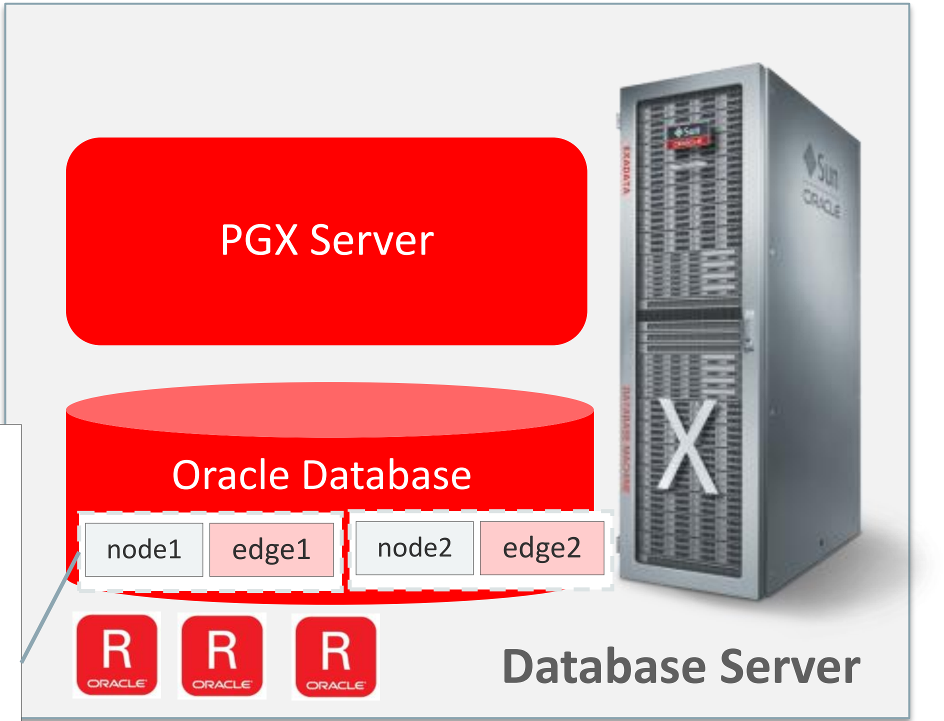
- Data Source
  - Graph data represented as two tables
    - Nodes and Edges
  - Multiple graphs stored in database
    - Using user-specified, unique table names

Node Table

Node ID	Node Prop 1 (name)	Node Prop 2 (age)	…
1238	John	39	…
1299	Paul	41	…
4818	…	…	…

Edge Table

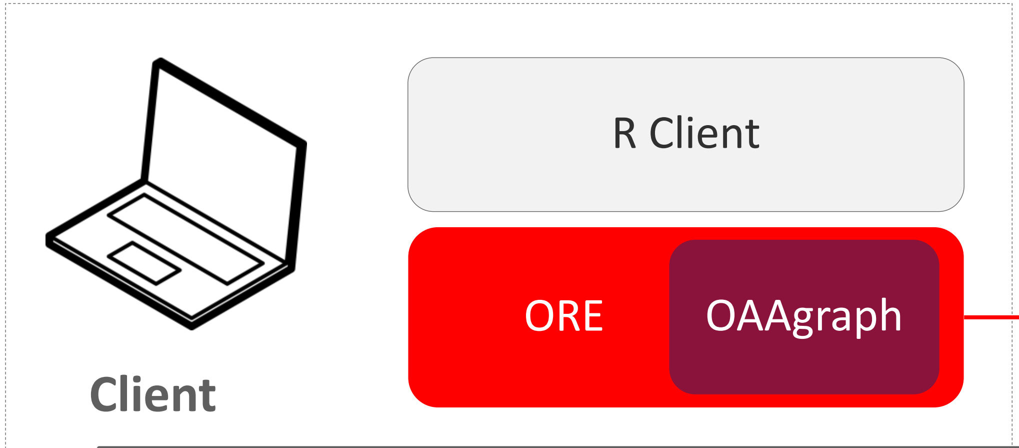
From Node	To Node	Edge Prop 1 (relation)	…
1238	1299	Likes	…
1299	4818	FriendOf	…
1299	6637	FriendOf	…

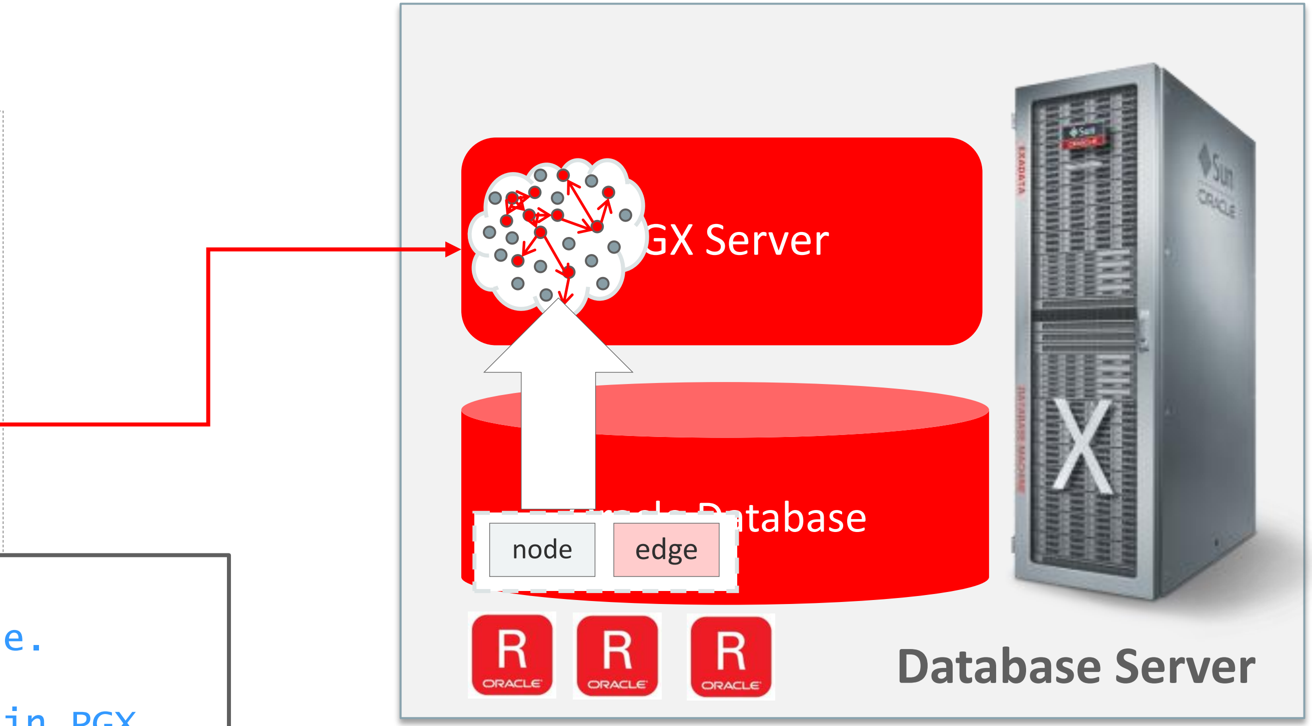


ORACLE

Execution Overview (ORE)

- Loading Graph

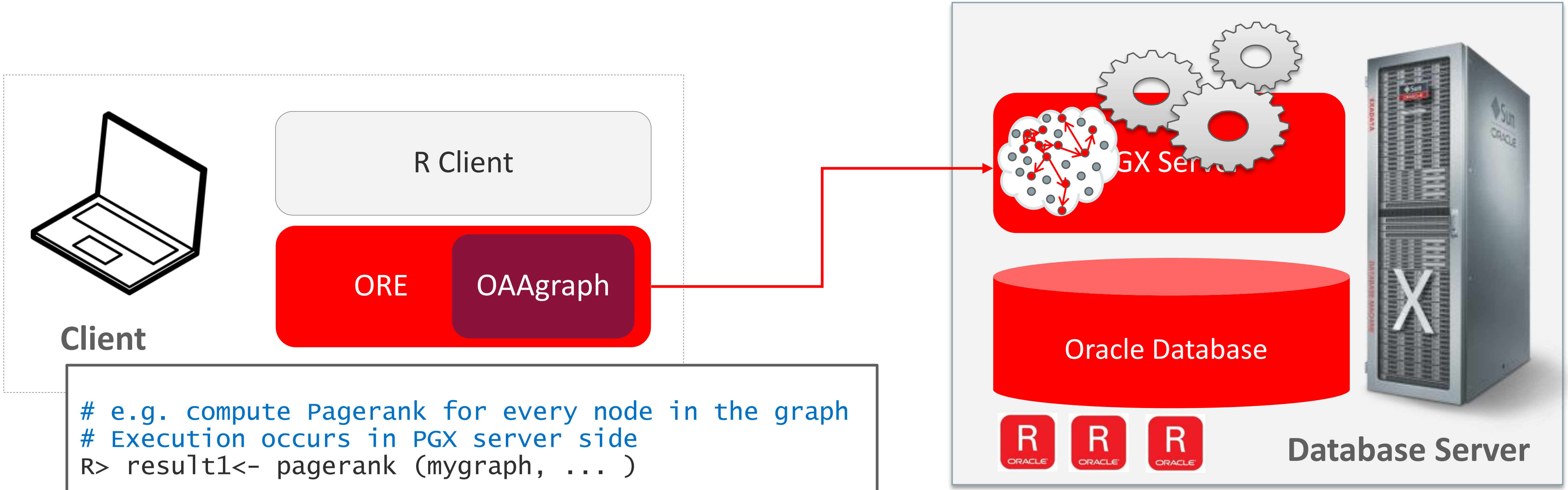





```
# Load graph into PGX:
#   Graph load happens at the server side.
#   Returns OAAgraph object, which is a
#   proxy (remote handle) for the graph in PGX
R> mygraph <-
    oaa.graph (NodeTable, EdgeTable, ...)
```

Execution Overview (ORE)

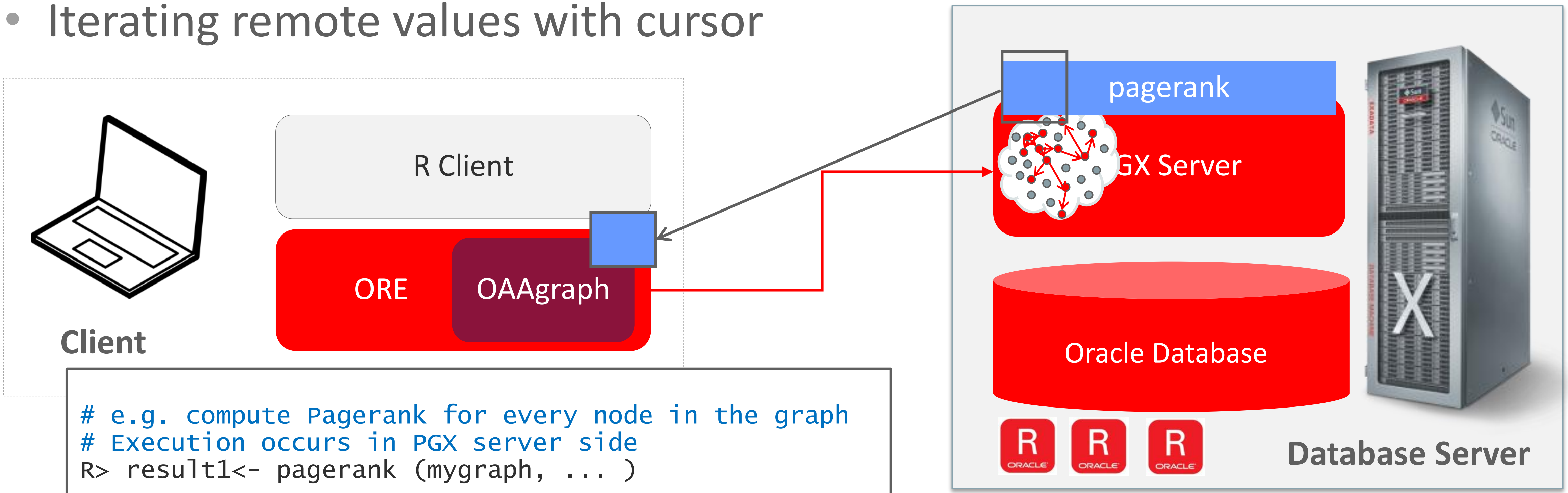
- Running Graph Algorithm




```
# e.g. compute Pagerank for every node in the graph
# Execution occurs in PGX server side
R> result1<- pagerank (mygraph, ... )
```

Execution Overview (ORE)

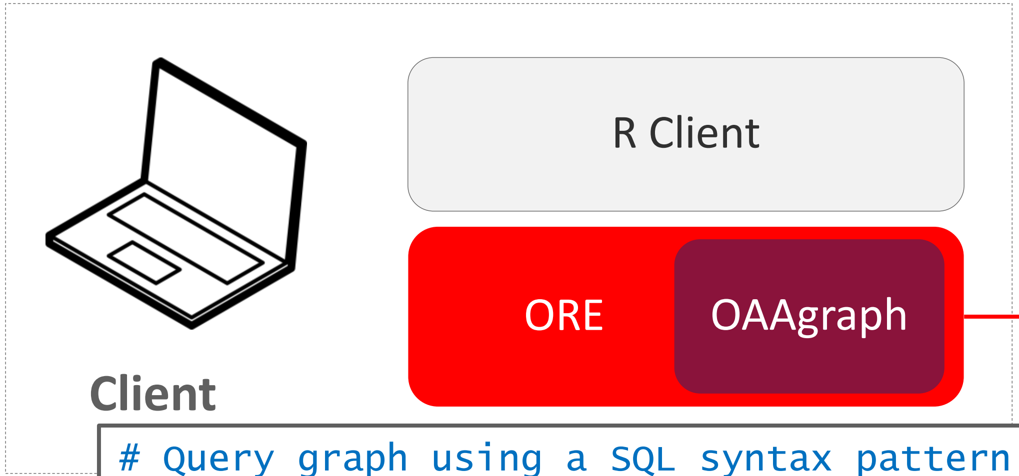
- Iterating remote values with cursor



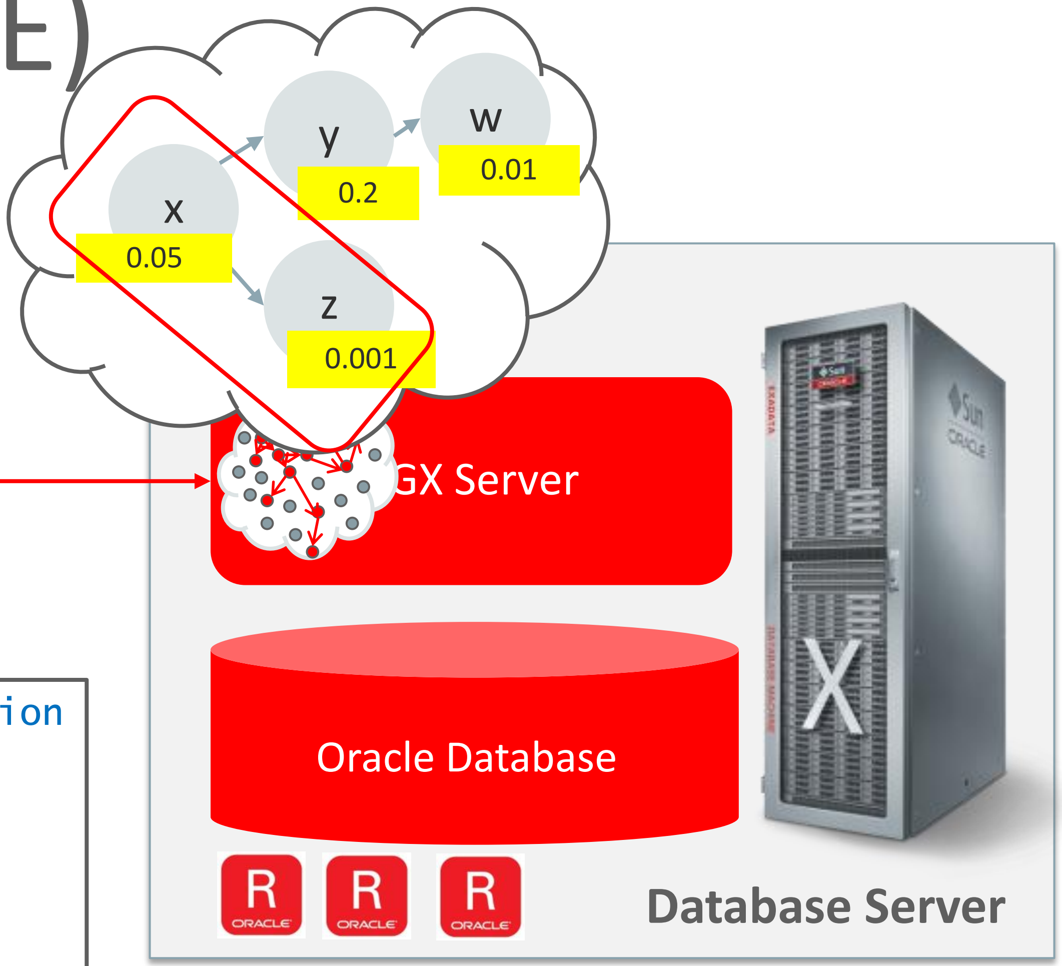

```
# e.g. compute Pagerank for every node in the graph
# Execution occurs in PGX server side
R> result1<- pagerank (mygraph, ... )

# Return value is a "cursor" object
# for the computed result:
#   client can get local data frames by oaa.next()
R> df <- oaa.next(result1, 10)
```

Execution Overview (ORE)

- Querying the graph

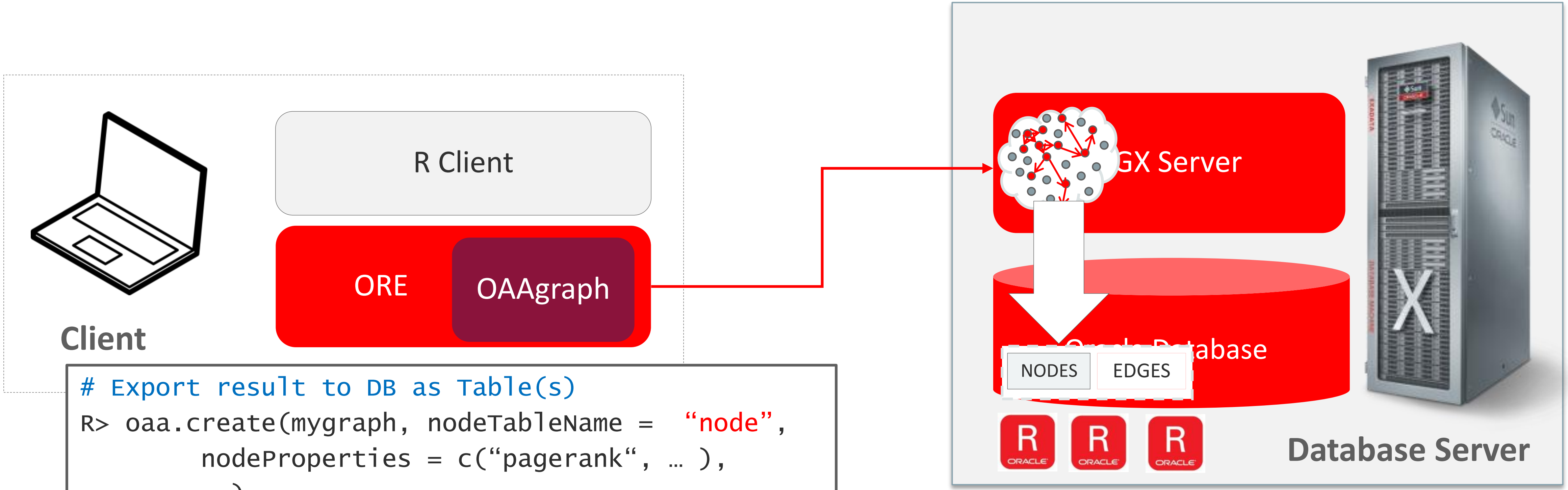



```
# Query graph using a SQL syntax pattern specification
R> q_result <- oaa.cursor(mygraph,
“SELECT n.name, m.name, n.pagerank, m.pagerank
 WHERE (n WITH pagerank < 0.1) -> (m),
       n.pagerank > m.pagerank
 ORDER BY n.pagerank”
)
# Returns a cursor to examine results
R> df <- oaa.next(q_result, 10)
```



Execution Overview (ORE)

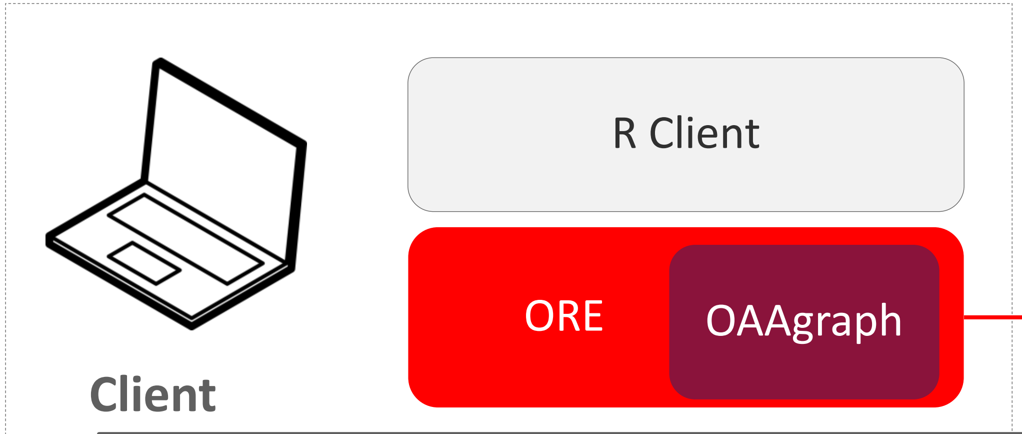
- Exporting the result to DB

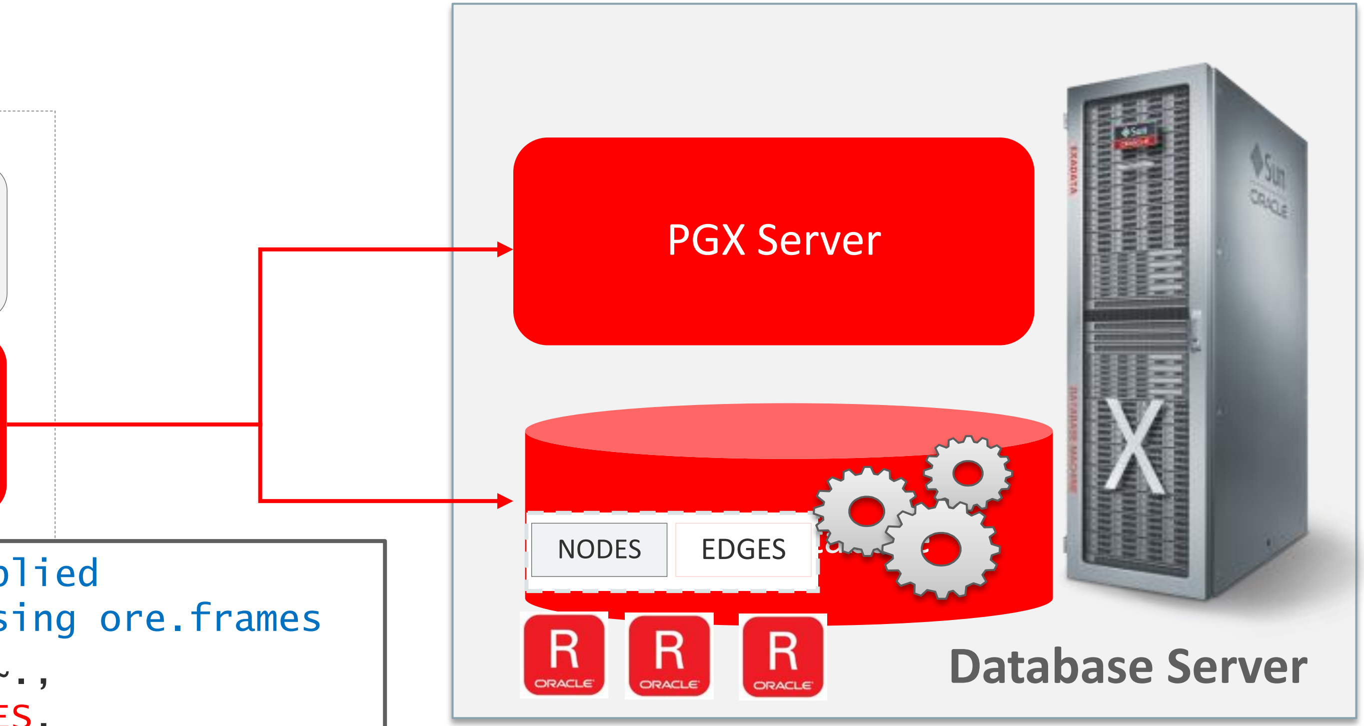



```
# Export result to DB as Table(s)
R> oaa.create(mygraph, nodeTableName = "node",
        nodeProperties = c("pagerank", … ),
        … )
```

Execution Overview (ORE)

- Continuing analysis with ORE Machine Learning



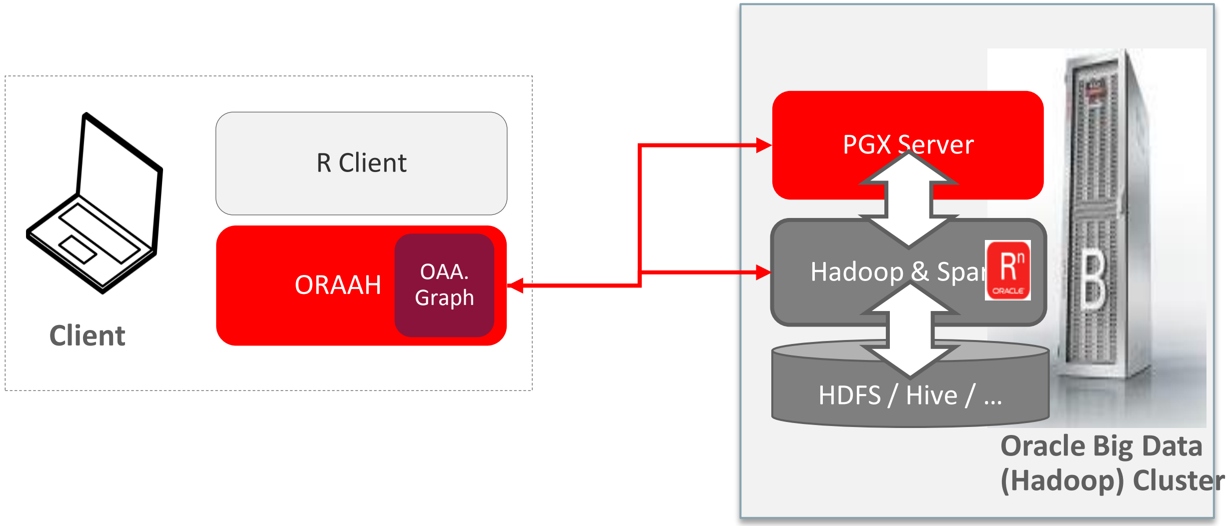



```
# Machine Learning analysis can be applied
# to the exported tables identified using ore.frames
R> model <- ore.odmKMmeans(formula = ~.,
                           data = NODES,
                           num.centers = 5,…)
R> scores <- predict(model, NODES, …)
…
```

Demo

Demo

- Environment
  - OAAgraph with ORAAH
  - PGX + SPARK + HDFS
- Dataset
  - Persons : name, age, zip, …
  - Calls: phone calls person-to-person

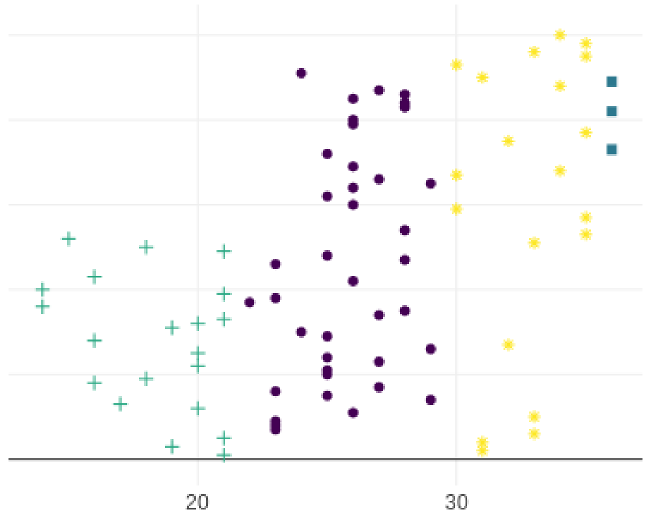


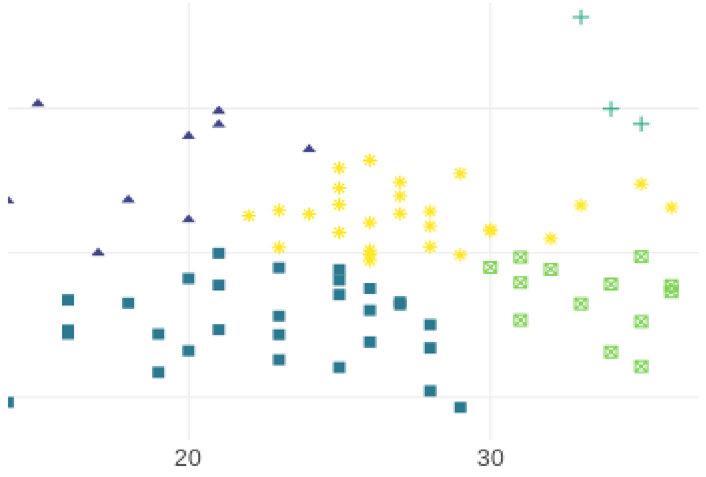
Demo Scenario

- Load *persons* data into ORAAH
- Check the data set
- Cluster *persons* by their age with K-means

- Load *calls* data into ORAAH
- Create an OAAgraph object with *persons* and *calls*
- Compute Pagerank and check results

- Export results back to ORAAH
- Cluster *persons* by their age AND pagerank values (with K-means)





ORACLE

Summary

- Powerful, scalable graph analytics enabled from R
- Cross-pollinate graph analytics and machine learning

Hardware and Software

Engineered to Work Together

ORACLE

ORACLE®

RDBMS queries (1)

Q1
```
SELECT n WHERE (n:macro WITH name = 'ksqget') ORDER BY n
```

Q2
```
SELECT n WHERE (n:macro WITH name =~ 'EVTDV\$') ORDER BY n
```

Q3
```
SELECT f.name WHERE (n:macro WITH name = 'ksqget') <-[c:expands]- (f:source_file) ORDER BY f.name LIMIT 10
```

Q4
```
SELECT c.use_file_id, c.use_end_line, c.name_file_id, c.use_start_line, c.name_start_line, c.name_start_column
WHERE (n:macro WITH name = 'ksqget') <-[c:expands]- (f:source_file WITH name = 'rdbms/src/server/vos/ksfd.c')
ORDER BY c.use_file_id, c.use_end_line, c.name_file_id, c.use_start_line, c.name_start_line, c.name_start_column
LIMIT 10
```

Q5
```
SELECT n1.id(), n2.id(), n3.id() WHERE (n1:field) -[:isa_type]-> (n2:function_type) -[:has_param_type]->
(n3:struct) -[:contains]-> (n1) ORDER BY n1, n2, n3 LIMIT 20
```

Q6
```
SELECT f.name, c.use_start_line
WHERE
  (n:macro WITH name = 'ksqget') <-[c:expands]- (f:source_file) -[e:expands]-> (m WITH name = 'KSQO_GLOBAL'),
  c.use_start_line <= e.use_start_line AND e.use_start_line <= c.use_start_line + 2
ORDER BY f.name, c.use_start_line
LIMIT 20
```

Q7
```
SELECT f.name WHERE (n:macro WITH name = 'KSQO_GLOBAL') <-[c:expands]- (f:source_file) -[e:expands]-> (n),
c.use_start_line = e.use_start_line ORDER BY f.name LIMIT 20
```

Q8
```
SELECT n1 WHERE (n1:function) -[e:calls]-> (n2:function)
ORDER BY e.use_start_line DESC, n1.name ASC, n2.name ASC LIMIT 10
```

ORACLE

RDBMS queries (2)

Q9
```
SELECT COUNT(*), MIN(e.use_end_line), MAX(e.use_start_line), AVG(e.name_start_line), SUM(e.name_start_column)
WHERE (:function) -[e:calls]-> (:function), e.name_start_line != -1
```

Q10
```
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
  (f:source_file WITH name = 'rdbms/src/generic/psm/kgfk.c') -[:file_contains|contains]-> () -
[r:calls|reads|writes]-> () <-[:file_contains|contains]- (o:source_file)
GROUP BY
  f, o, r.label()
ORDER BY f.name, o.name, r.label()
LIMIT 20
```

Q11
```
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
  (f:source_file) -[:file_contains|contains]-> () -[r:calls|reads|writes]-> () <-[:file_contains|contains]-
(o:source_file)
GROUP BY
  f, o, r.label()
ORDER BY f.name, o.name, r.label()
LIMIT 20
```

Q12
```
SELECT n.id(), n.outDegree()
WHERE (n:function)
ORDER BY n.outDegree() DESC
LIMIT 10
```

ORACLE

RDBMS queries (3)

Q13
```
SELECT n.id() WHERE (n WITH name = 'ksqget' OR name =~ 'EVTDV\$') ORDER BY n
```

Q14
```
PATH contains := () -[:file_contains|dir_contains]-> ()
SELECT m
WHERE
  (d:directory WITH name = 'rdbms/src/client') -/:contains*/-> (f),
  (f) -[:file_contains]-> (m:macro WITH name =~ 'EVTDV\$')
```

Q15
```
PATH includes := () -[:includes]-> ()
SELECT f.name
WHERE
  (f) -/:includes*/-> (h:source_file WITH name = 'rdbms/include/kge.h')
ORDER BY f.name
LIMIT 20
```

Q16
```
PATH contains := () -[:file_contains|contains]-> ()
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
  (f:source_file WITH name = 'rdbms/src/generic/psm/kgfk.c') -/:contains*/-> (x),
  (o:source_file)                                            -/:contains*/-> (y),
  (x) -[r:calls|reads|writes]-> (y)
GROUP BY
  f, o, r.label()
ORDER BY o.name
LIMIT 20
```

ORACLE

RDBMS queries (4)

Q17
```
PATH contains := () -[:file_contains|contains]-> ()
SELECT f.name, o.name, r.label(), COUNT(*)
WHERE
  (f:source_file) -/:contains*/-> (x),
  (o:source_file) -/:contains*/-> (y),
  (x) -[r:calls|reads|writes]-> (y)
GROUP BY
  f, o, r.label()
ORDER BY o.name
LIMIT 20
```

ORACLE