Session 7: Oracle R Enterprise 1.5.1
OAAgraph Package
Oracle Spatial and Graph PGX Graph Algorithms
Oracle R Technologies

Mark Hornick
Director, Oracle Advanced Analytics and Machine Learning
July 2017
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, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Topics

• Graph Analysis and Machine Learning
• ORE integration with Oracle Spatial and Graph option for PGX
• OAAtomain interface
• OAAtom examples
Graph Analysis And Machine Learning
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
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, pattern machine, recommendations
  – Graph Pattern Matching
    • Queries that find sub-graphs fitting relationship patterns
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

\[ \text{Feature 1} \quad \text{Feature 2} \quad \text{Feature 3} \quad \text{Feature 4} \quad \text{Feature 5} \quad \text{Feature 6} \quad \text{Feature 7} \]

\[
\begin{array}{cccccccc}
D1 & & & & & & \\
D2 & & & & & & \\
D3 & & & & & & \\
\end{array}
\]

*Predictive Model*

- e.g. Can you identify groups of close customers from their call graph in order to predict customer churn?
ORE integration with Oracle Spatial and Graph option for PGX
What is PGX?

• 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
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
OAAn graph
An R interface integrating PGX and ORE/ORAAH for Machine Learning
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
Graph Analytics

Compute graph metric(s)
Add to structured data
Build predictive model using graph metric
Build model(s) and score or classify data

Explore graph or compute new metrics using ML result
Add to graph
• **OAAgraph** is an additional R package that comes with Oracle R Enterprise

• **OAAgraph** gives remote control of PGX server

• PGX loads graph from database (ore.frames)
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

Client

R Client

ORAHH OAAGraph

PGX Server

Hadoop & Spark

HDFS / Hive / ...

Oracle Big Data (Hadoop) Cluster
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(...)

R Client

ORE

OAAGraph

PGX Server

Oracle Database

Database Server
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

<table>
<thead>
<tr>
<th>Node ID</th>
<th>Node Prop 1 (name)</th>
<th>Node Prop 2 (age)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>John</td>
<td>39</td>
</tr>
<tr>
<td>1299</td>
<td>Paul</td>
<td>41</td>
</tr>
<tr>
<td>4818</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>From Node</th>
<th>To Node</th>
<th>Edge Prop 1 (relation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>1299</td>
<td>Likes</td>
</tr>
<tr>
<td>1299</td>
<td>4818</td>
<td>FriendOf</td>
</tr>
<tr>
<td>1299</td>
<td>6637</td>
<td>FriendOf</td>
</tr>
</tbody>
</table>
Execution Overview (ORE)

• Loading Graph

```r
# 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

```r
# 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

```r
# 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

```r
# 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

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

<table>
<thead>
<tr>
<th>X86 Server</th>
<th>#Nodes</th>
<th>#Edges</th>
<th>Closeness Centrality (PGX)</th>
<th>Closeness Centrality (igraph)</th>
<th>Betweenness Centrality (PGX)</th>
<th>Betweenness Centrality (igraph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xeon E5-2660 2.2Ghz 2 socket x 8 cores x 2HT 256GB DRAM</td>
<td>75,879</td>
<td>811,480</td>
<td>17 sec</td>
<td>6.27 min</td>
<td>45 sec</td>
<td>11.5 min</td>
</tr>
<tr>
<td>Epinion</td>
<td>107,614</td>
<td>2,447,570</td>
<td>15 sec</td>
<td>132 min</td>
<td>30 min</td>
<td>290 min</td>
</tr>
<tr>
<td>Google</td>
<td>3,000,000</td>
<td>16,519,402</td>
<td>26 min</td>
<td>5+ days</td>
<td>20 hrs</td>
<td>5+ days</td>
</tr>
<tr>
<td>Churner</td>
<td>3,774,768</td>
<td>33,037,894</td>
<td>6.6 hrs</td>
<td>5+ days</td>
<td>48 hrs</td>
<td>5+ days</td>
</tr>
<tr>
<td>Patent</td>
<td>4,846,609</td>
<td>85,702,474</td>
<td>5.8 hrs</td>
<td>5+ days</td>
<td>60 hrs</td>
<td>5+ days</td>
</tr>
</tbody>
</table>

Effectively does not complete!
OAAGraph Examples
Example: Connecting to PGX

library(ORE)
library(OAAgraph)

dbHost <- "<DATABASE_HOST>"
dbUser <- "<DATABASE_USERNAME>"
dbPassword <- "<DATABASE_PASSWORD>"
dbSid <- "<DATABASE_SID>"
pgxBaseUrl <- "<PGX_BASE_URL>"
ore.connect(host = dbHost, user = dbUser, password = dbPassword, sid = dbSid)
oaa.graphConnect(pgxBaseUrl = pgxBaseUrl, dbHost = dbHost,
                 dbSid = dbSid, dbUser = dbUser, dbPassword = dbPassword)
Example: Create some data – node and edge tables

```r
#-- Create the node table in Oracle Database
VID <- c(1, 2, 3, 4, 5)
NP1 <- c("node1", "node2", "node3", "node4", "node5")
NP2 <- c(111.11, 222.22, 333.33, 444.44, 555.55)
NP3 <- c(1, 2, 3, 4, 5)
nodes <- data.frame(VID, NP1, NP2, NP3)
ore.drop(table="MY_NODES")
ore.create(nodes, table = "MY_NODES")

#-- Create the edge table in Oracle Database
EID <- c(1, 2, 3, 4, 5)
SVID <- c(1, 3, 3, 2, 4)
DVID <- c(2, 1, 4, 3, 2)
EP1 <- c("edge1", "edge2", "edge3", "edge4", "edge5")
EL <- c("label1", "label2", "label3", "label4", "label5")
edges <- data.frame(EID, SVID, DVID, EP1, EL)
ore.drop(table="MY_EDGES")
ore.create(edges, table = "MY_EDGES")
```
Example: Create some data – node and edge tables

```r
#-- Create a graph in PGX from the node and edge tables in the database
graph <- oaa.graph(MY_EDGES, MY_NODES, "myPgxGraph")
names(graph, "nodes")
names(graph, "edges")

#-- See result of countTriangles function, which gives an overview of the
#-- number of connections between nodes in neighborhoods
countTriangles(graph, sortVerticesByDegree=FALSE)

#-- See results from degree algorithm variants, note the graph nodes
#-- are augmented with new properties as indicated by the 'name' argument
degree(graph, name = "OutDegree")
degree(graph, name = "InDegree", variant = "in")
degree(graph, name = "InOutDegree", variant = "all")
```
Example: Load graph from database tables

```r
#-- Create a graph in PGX from the node and edge tables in the database
graph <- oaa.graph(MY_EDGES, MY_NODES, "myPgxGraph")
names(graph, "nodes")
names(graph, "edges")

#-- See result of countTriangles function, which gives an overview of the
#-- number of connections between nodes in neighborhoods
countTriangles(graph, sortVerticesByDegree=FALSE)

#-- See results from degree algorithm variants, note the graph nodes
#-- are augmented with new properties as indicated by the 'name' argument
degree(graph, name = "OutDegree")
degree(graph, name = "InDegree", variant = "in")
degree(graph, name = "InOutDegree", variant = "all")
```
Example: Create cursors to access the results

```r
#-- Create a cursor including the degree properties
cursor <- oaa.cursor(graph, c("OutDegree", "InOutDegree", "InDegree"), "nodes")
oaa.next(cursor, 5)
```

```r
#-- Create a cursor over the degree properties using
#-- the PGX SQL-like query language PGQL
cursor <- oaa.cursor(graph,
    query = "select n.OutDegree, n.InOutDegree, n.InDegree where (n)
            order by n.OutDegree desc")
```

```r
#-- View the first 5 entries from the cursor
oaa.next(cursor, 5)
```
Example: Create cursors to access the results

```r
#-- See results from the pagerank algorithm
pagerankCursor <- pagerank(graph, 0.085, 0.1, 100)
oaa.next(pagerankCursor, 5)

#-- Create a cursor over the pagerank property using PGQL
cursor <- oaa.cursor(graph,
    query = "select n.pagerank where (n) order by n.pagerank desc")
oaa.next(cursor, 5)

#-- This could be done using the R interface as well...
cursor <- oaa.cursor(graph, "pagerank", ordering="desc")
oaa.next(cursor, 5)
```
Example: More graph analytics and create snapshots

```r
#-- Compute the adamic adar index for edges
topEdges <- adamicAdarCounting(graph)
oaa.next(topEdges)

#-- List any graph snapshots available
oaa.graphSnapshotList()

#-- Export a binary snapshot of the whole graph into Oracle Database
#-- and view the listing again
oaa.graphSnapshotPersist(graph, nodeProperties = TRUE, edgeProperties = TRUE)
oaa.graphSnapshotList()
```
Example: Read snapshots and create tables from graphs

```r
#-- Read the snapshot back into memory
graph2 <- oaa.graphSnapshot("myPgxGraph")

#-- Export the graph nodes and specific node properties from memory into a database table
oaa.create(graph2, nodeTableName = "RANKED_NODES", nodeProperties = TRUE)

#-- Export both nodes and edges as tables from memory into the database,
#-- but only export the pagerank node property
oaa.create(graph2, nodeTableName = "RANKED_GRAPH_N",
            nodeProperties = c("NP1", "pagerank"),
            edgeTableName = "RANKED_GRAPH_E")

#-- Export graph edges and their properties from memory into a database table
oaa.create(graph2, edgeTableName = "RANKED_EDGES", edgeProperties = TRUE)

#-- Free the graphs at the PGX server
oaa.rm(graph)
oaa.rm(graph2)
```
Learn More about Oracle’s R Technologies...

http://oracle.com/goto/R