HP Big Data Conference
Deep dive into Haven Predictive Analytics

Powered by HP Distributed R and HP Vertica
Arash Fard, Vishrut Gupta / August 12, 2015
Outline

• Overview
• Architecture of Distributed R
• Out-of-box algorithms
• vertica.dplyr Package
• An end-to-end example
Overview
R is the language of data scientists

R is a programming language for statistical computing

- Popular
- Open source
- Flexible
- Extensible

Distributed R extends R language to provide distributed computing

- Scalable
- Growing number of distributed algorithms
- Very efficient Data Transfer from Vertica

Not scalable
Slow (single process)
Inefficient Data Transfer
HP Vertica and HP Distributed R

Delivering scale and high performance for predictive analytics

1. Ingest and prepare data by leveraging HP Vertica
2. Build and evaluate predictive models on large data sets using Distributed R
3. Deploy models to Vertica and use in-database scoring to produce prediction results for BI and applications
Architecture of Distributed R
The architecture of Distributed R

By default:
- Number of workers = number of physical nodes
- Number of executors = total number of logical cores
## Distributed data structures

<table>
<thead>
<tr>
<th>R - object</th>
<th>Distributed R - dobjecet</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix</td>
<td>darray</td>
</tr>
<tr>
<td>The same type elements:</td>
<td>Each partition is a matrix, no support for</td>
</tr>
<tr>
<td>numerical, character, or</td>
<td>character</td>
</tr>
<tr>
<td>logical</td>
<td></td>
</tr>
<tr>
<td>data.frame</td>
<td>dframe</td>
</tr>
<tr>
<td>Columns with different types</td>
<td>Each partition is a data.frame</td>
</tr>
<tr>
<td>list</td>
<td>dlist</td>
</tr>
<tr>
<td>Vectors of different types</td>
<td>Each partition is a list</td>
</tr>
<tr>
<td>and lengths</td>
<td></td>
</tr>
</tbody>
</table>
Distributed partitions of data

Partitions of a darray for example

darray

1

2

3

4
Distributed computing on partitions

foreach function: The main tool for running distributed jobs

- Express computations over partitions
- Execute across the cluster

\[ \text{foreach } f(x) \]
An example

```r
library(distributedR)  # Loads the package into R

distributedR_start()  # Starts up the cluster

X <- darray(dim=c(4,4), blocks=c(2,2))  # Creates a 4x4 darray, each partition 2x2

foreach(index, 1:npartitions(X),
  function( Xi= splits(X, index), i=index) {
    Xi <- matrix(i, nrow(Xi), ncol(Xi))
    update(Xi)
  })
```

© Copyright 2015 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice.
Running the example

```
foreach

f (x)
```

```
1 1
1 1
2 2
2 2
3 3
3 3
4 4
4 4
```
HP distributed R: deployment options

Multiple ways to achieve performant predict analytics using HP big data platform

- Bring Scale and Performance to Standard R
  - Distributed R + Standard R

- Bring Scale and Performance to R and Operationalize predictive analytics
  - Vertica + Distributed R + Standard R

- Ingest and prepare large data volumes and various data types
  - Hadoop (HDFS Storage) + JSON, CSV, AVRO etc.
Out-of-box algorithms
Available algorithms

- Similar signature, accuracy as R packages
- **Scalable** and **high performance**
- E.g., regression on billions of rows in a couple of minutes

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Use cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression (GLM)</td>
<td>Risk Analysis, Trend Analysis, etc.</td>
</tr>
<tr>
<td>Logistic Regression (GLM)</td>
<td>Customer Response modeling, Healthcare analytics</td>
</tr>
<tr>
<td></td>
<td>(Disease analysis)</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Customer churn, Market campaign analysis</td>
</tr>
<tr>
<td>K-Means Clustering</td>
<td>Customer segmentation, Fraud detection, Anomaly</td>
</tr>
<tr>
<td></td>
<td>detection</td>
</tr>
<tr>
<td>Page Rank</td>
<td>Identify influencers</td>
</tr>
</tbody>
</table>
Using Distributed R is easy

1. `distributedR_start()`
Using Distributed R is easy

1. `distributedR_start()`

2. In database data preparation (SQL)
Using Distributed R is easy

1. `distributedR_start()`

2. In database data preparation (SQL)

3. `db2dframe(...)"
Using Distributed R is easy

1. distributedR_start()
2. In database data preparation (SQL)
3. db2dframe(...)
4. hpdrandomForest(...)

Ship code to data

Distributed R

Vertica DB
Using Distributed R is easy

1. `distributedR_start()`
2. In database data preparation (SQL)
3. `db2dframe(...)`
4. `hpdrandomForest(...)`
5. `Deploy.model(...)`
Using Distributed R is easy

1. distributedR_start()

2. In database data preparation (SQL)

3. db2dframe(...)

4. hpdrandomForest(...)

5. Deploy.model(...)

6. In database scoring (SQL)
Performance benefits
Distributed R Vertica connector

Data access at Big Data scale - fast data transfer

Enabling predictive analytics in Vertica: Fast data transfer, distributed model creation and in-database prediction. Shreya Prasad, Arash Fard, Vishrut Gupta, Jorge Martinez, Jeff LeFevre, Vincent Xu, Meichun Hsu, Indrajit Roy. SIGMOD 2015, Melbourne, Australia.
Distributed model creation in Distributed R

Algorithm: K-Means
Setup: SL 390 servers, 12*2 HT cores/server, 196GB RAM

Dataset: 10Mx100 (~8GB), 1000 clusters
Vertica In-database SQL Prediction Functions

Near linear scalability and predictions on billions of records

Setup: 5*SL 390 servers, 12*2 HT cores/server, 196GB RAM
vertica.dplyr Package

An easy way to work with Vertica from R console
vertica.dplyr Package

An easy way to work with Vertica from R console

• An open source package based on dplyr project by “Hadley Wickham”
  https://github.com/vertica/vertica.dplyr
• Working with database tables from R-console with simple R commands
• Lazy execution
• Extensions for loading distributed objects of Distributed R
An end-to-end example
A classification problem

Use case for Marketing

- **Data source:** Publicly available marketing data from a banking institution. ([https://archive.ics.uci.edu/ml/datasets/Bank+Marketing](https://archive.ics.uci.edu/ml/datasets/Bank+Marketing))
- **Goal:** To determine if a phone-based marketing campaign will be successful

**Steps (assuming that the data is already in Vertica database)**

- Exploring and preparing the data in Vertica database
- Loading the Training set to Distributed R
- Training a logistic regression model in Distributed R
- Deploying the model into Vertica database
- In-database scoring of Test set
The data in Vertica

Demo screenshot - vsql console

Bank Marketing Data Set

- **Source:** [https://archive.ics.uci.edu/ml/datasets/Bank+Marketing](https://archive.ics.uci.edu/ml/datasets/Bank+Marketing)
- **Number of records:** 41188
- **Number of columns:** 21

```sql
condor=> \d bank_original

<table>
<thead>
<tr>
<th>Schema</th>
<th>Table</th>
<th>Column</th>
<th>List of Fields by Tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>public</td>
<td>bank_original</td>
<td>age</td>
<td>int</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>job</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>marital</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>education</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>&quot;default&quot;</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>housing</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>loan</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>contact</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>MONTH</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>day_of_week</td>
<td>char(5) 5</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>duration</td>
<td>int 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>campaign</td>
<td>int 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>pdays</td>
<td>int 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>previous</td>
<td>int 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>poutcome</td>
<td>varchar(80) 80</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>&quot;emp.var.rate&quot;</td>
<td>float 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>&quot;cons.price.idx&quot;</td>
<td>float 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>&quot;cons.conf.idx&quot;</td>
<td>float 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>euribor3m</td>
<td>float 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>&quot;nr.employed&quot;</td>
<td>float 8</td>
</tr>
<tr>
<td>public</td>
<td>bank_original</td>
<td>y</td>
<td>varchar(5) 5</td>
</tr>
</tbody>
</table>

(21 rows)
```
# Z-score normalization of numerical features

Comparing vsql and vertica.dplyr

<table>
<thead>
<tr>
<th>vsql</th>
<th>vertica.dplyr</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE VIEW bank_normalized AS</td>
<td>vertica_odbc &lt;- src_vertica(&quot;VerticaDSN&quot;)</td>
</tr>
<tr>
<td>SELECT (age – m_avg) / std_age AS age_z,</td>
<td>orig &lt;- tbl(vertica_odbc, “bank_original”)</td>
</tr>
<tr>
<td>...</td>
<td>m_sd &lt;- summarise(orig, m_age=mean(age), std_age=sd(age), ...)</td>
</tr>
<tr>
<td>FROM bank_original,</td>
<td>z &lt;- collect (m_sd)</td>
</tr>
<tr>
<td>(SELECT avg(age) AS m_age, stddev(age) AS</td>
<td>normalized &lt;- mutate(orig, age_z= (age -z[&quot;m_age&quot;]) /z[&quot;std_age&quot;], ...)</td>
</tr>
<tr>
<td>std_age FROM bank_original) AS tbl_mean_std;</td>
<td>norm_table &lt;- select(normalized, age_z, ...)</td>
</tr>
</tbody>
</table>
Top-N view of categorical features

Comparing vsq1 and vertica.dplyr

### vsq1
CREATE VIEW bank_top_n AS
SELECT DECODE( marital 'married', 'married', 'single', 'single', 'other') AS marital, ...
FROM bank_normalized;

### vertica.dplyr
decoded <- mutate(norm_table, marital2=decode(marital, 'married', 'married', 'single', 'single', 'other'), ...)
decoded <- select(decoded, marital=marital2, ...)
db_save_view(decoded,"bank_top_n")
Categorical columns to binary representation

Comparing vsql and vertica.dplyr

- hpdglm(), like many other algorithms, only accepts numerical features
- We have provided an R function for this purpose: cat2num

For example:

```r
# Example usage of cat2num function
# cat2num (srcTable='bank_top_n', dsn='VerticaDSN', dstTable='bank_top_n_num')
```

**poutcome** column has 3 categories: “nonexistent”, “failure”, “success”.

It will be converted to 2 columns: **poutcome_1** and **poutcome_2** where:

- `poutcome_1=0, poutcome_2=0` means “nonexistent”
- `poutcome_1=1, poutcome_2=0` means “failure”
- `poutcome_1=0, poutcome_2=1` means “success”

<table>
<thead>
<tr>
<th>poutcome</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>“nonexistent”</td>
<td>0</td>
</tr>
<tr>
<td>“failure”</td>
<td>1</td>
</tr>
<tr>
<td>“success”</td>
<td>2</td>
</tr>
</tbody>
</table>
Splitting data to Training and Testing sets

Comparing vsql and vertica.dplyr

Sampling
- 20% Testing Set
- 80% Training Set

<table>
<thead>
<tr>
<th>vsql</th>
<th>vertica.dplyr</th>
</tr>
</thead>
</table>
| CREATE TABLE testing_set AS (SELECT * FROM bank_top_n_num WHERE RANDOM() < 0.2); | top_tbl <- tbl(vertica,"bank_top_n_num")
| | testing_set <- filter(top_tbl, random() < 0.2)
| | testing_set <- compute(testing_set, name="testing_set")
| CREATE TABLE training_set AS (SELECT * FROM bank_top_n_num EXCEPT SELECT * FROM testing_set); | training_set <- compute(setdiff(top_tbl, testing_set), "training_set")
Model training and deployment

Demo screenshot – R console

- Loading data from Vertica database
- Training Logistic Regression
- Deploying the trained model into Vertica

```
library(HPdata)
distributedR_start()
LoadedData <- db2darray(tableName='bank_training', dsn='VerticaDSN', resp=list('y_1'))

Loading total 31068 rows from table bank_training from Vertica with approximate partition of 1942 rows
progress: 100%

library(HPdregression)
theModel <- glm(formula=Y ~ X, family=binomial)

deploy.model(model=theModel, dsn='SF', modelName='demoModel', modelComments='A logistic regression model for bank data')
```
In-database scoring and checking accuracy

Demo screenshot – vsql console

```
CREATE TABLE predict_test AS
  (SELECT v_1, 
   GlmPredict(age_z, job_1, job_2, job_3, job_4, job_5, marital_1, marital_2, 
   education_1, education_2, education_3, education_4, education_5, default_1, housing_1, 
   loan_1, contact_1, month_1, month_2, month_3, month_4, month_5, day_of_week_1, day_of_ 
   week_2, day_of_week_3, day_of_week_4, campaign_z, pdays_z, previous_z, poutcome_1, 
   poutcome_2, emp_var_rate_z, cons_price_idx_z, cons_conf_idx_z, euribor3m_z, nr_employed_z 
   USING PARAMETERS model='dbadmin/demoModel',TYPE='response')
  FROM bank_test);

=> SELECT sum(ROUND(1-abs(v_1 - GlmPredict))) from predict_test;
   sum
  ------
   7249
  (1 row)

=> SELECT count(*) from predict_test;
  count
  ------
   8367
  (1 row)

=> select 7294 / 8367 as accuracy;
  accuracy
  ------------
   0.873758097286960679
  (1 row)
```
Conclusion
Summary

• Distributed R is a powerful platform for developing distributed algorithms

• There are growing number of out-of-box Machine Learning and Data Mining algorithms in Distributed R product

• Integrating Distributed R and Vertica provides a very powerful system to harness the power of Big Data Analytics
Follow-up

• Distributed R is now a product. (www.distributedr.org)

• It is open source. (github.com/vertica/DistributedR)

• We are looking for people to use it and contribute to it.
THANK YOU