High Performance Predictive Analytics in R and Hadoop: Achieving Big Data Big Analytics

Presented by:
Mario E. Inchiosa, Ph.D.
US Chief Scientist

August 27, 2013
Polling Questions 1 & 2
Agenda

- Riding the Hadoop Wave
- Big Data Big Analytics
- R + Hadoop from Revolution Analytics
- Revolution R Enterprise ScaleR
- Getting Started
- Q&A
Riding the Hadoop Wave
Solve old problems in new ways
Solve new problems
If you want something you’ve never had, you must be willing to do something you’ve never done.
Major entertainment company integrates analytics across brands
Fraud detection interval reduced from 2 weeks to 7 hours
Predict mortgage default in time to avoid it
Recommend optimal growing plan
Big Data Big Analytics is different
Big Data is big, complex and messy
Big Analytics are compute intensive
Big Data Big Analytics rewards you
Innovate with R

- Most widely used data analysis software
  - Used by 2M+ data scientists, statisticians and analysts
- Most powerful statistical programming language
  - Flexible, extensible and comprehensive for productivity
- Create beautiful and unique data visualizations
  - As seen in New York Times, Twitter and Flowing Data
- Thriving open-source community
  - Leading edge of analytics research
- Fills the talent gap
  - New graduates prefer R
R is open source and drives analytic innovation but has some limitations for Enterprises

<table>
<thead>
<tr>
<th>Big Data</th>
<th>In-memory bound</th>
<th>Disk based scalability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed of Analysis</td>
<td>Single threaded</td>
<td>Parallel threading</td>
</tr>
<tr>
<td>Enterprise Readiness</td>
<td>Community support</td>
<td>Commercial support</td>
</tr>
<tr>
<td>Analytic Breadth &amp; Depth</td>
<td>5,000+ innovative analytic packages</td>
<td>Leverage open source packages plus Big Data-ready packages</td>
</tr>
</tbody>
</table>
Revolution R Enterprise
High Performance, Multi-Platform Analytics Platform

Revolution R Enterprise

DeployR
Web Services
Software Development Kit

DevelopR
Integrated Development Environment

ConnectR
High Speed & Direct Connectors
Teradata, Hadoop (HDFS, HBase), SAS, SPSS, CSV, OBDC

ScaleR
High Performance, Scalable, Portable, Parallelized, Full-Featured Big Data Analytics

DistributeR
Parallel & Distributed Computing Framework
LSF, HPC Server, Azure Burst, Hadoop

RevoR
Performance Enhanced Open Source R + CRAN packages
IBM PureData (Netezza), Platform LSF, MS HPC Server, MS Azure Burst, Cloudera, Hortonworks, IBM BigInsights, Intel Hadoop, SMP servers, Teradata
Big Data Speed @ Scale with Revolution R Enterprise

- In-Hadoop Execution
- In-Database Execution
- Parallelized User Code
- Parallelized Algorithms
- Multi-Core Execution
- Multi-Threaded Execution
- Memory Management
- Fast Math Libraries
Our Objectives with Respect to Hadoop

- Provide the first enterprise-ready, commercially supported, full-featured, out-of-the-box Predictive Analytics suite running in Hadoop
- Allow our customers to do predictive analytics as easily in Hadoop as they can using R on their workstations
- Scalable and High Performance
Simplicity Goal: Hadoop As An R Engine.

- Run Revolution R Enterprise Code In Hadoop Without Change
- Provide ScaleR Pre-Parallelized Algorithms
- No Need To “Think In MapReduce”
- Eliminate Movement to Slash Latencies
- Expanded Deployment Options
Revolution R Enterprise ScaleR

- An R package that adds capabilities to R:
  - Data Import/Clean/Explore/Transform
  - Analytics – Descriptive and Predictive
  - Parallel and distributed computing
  - Visualization
- Scales from small local data to huge distributed data
- Scales from workstation to server to cluster to cloud
- Portable – the same code works on small and big data, and on workstation, server, cluster, Hadoop
High Performance Big Data Analytics with Revolution R Enterprise ScaleR

R Data Step
Descriptive Statistics
Statistical Tests
Sampling
Predictive Models
Data Visualization
Machine Learning
Simulation
ScaleR: High Performance Scalable Parallel External Memory Algorithms

Data Prep, Distillation & Descriptive Analytics

R Data Step
- Data import – Delimited, Fixed, SAS, SPSS, OBDC
- Variable creation & transformation
- Recode variables
- Factor variables
- Missing value handling
- Sort
- Merge
- Split
- Aggregate by category (means, sums)
- **Use any of the functionality of the R language to transform and clean data row by row!**

Descriptive Statistics
- Min / Max
- Mean
- Median (approx.)
- Quantiles (approx.)
- Standard Deviation
- Variance
- Correlation
- Covariance
- Sum of Squares (cross product matrix for set variables)
- Risk Ratio & Odds Ratio
- Cross-Tabulation of Data (standard tables & long form)
- Marginal Summaries of Cross Tabulations

Statistical Tests
- Chi Square Test
- t-Test
- F-Test
- **Plus 1,000’s of other tests available in R!**

Sampling
- Subsample (observations & variables)
- Random Sampling
- High quality, fast, parallel random number generators
Revolution R Enterprise ScaleR: High Performance Big Data Analytics

### Statistical Modeling

- Covariance, Correlation, Sum of Squares (cross product matrix for set variables) matrices
- Multiple Linear Regression
- Generalized Linear Models (GLM)
  - All exponential family distributions: binomial, Gaussian, inverse Gaussian, Poisson, Tweedie. Standard link functions including: cauchit, identity, log, logit, probit. User defined distributions & link functions.
- Logistic Regression
- Classification & Regression Trees
- Decision Forests
- Predictions/scoring for models
- Residuals for all models

### Machine Learning

#### Data Visualization

- Histogram
- Line Plot
- Lorenz Curve
- ROC Curves (actual data and predicted values)

#### Variable Selection

- Stepwise Regression
- PCA

#### Cluster Analysis

- K-Means

#### Classification

- Decision Trees
- Decision Forests

#### Simulation

- Parallel random number generators for Monte Carlo
  - *Use the rich functionality of R for simulations*
ScaleR Scalability and Performance

- Handles an arbitrarily large number of rows in a fixed amount of memory
- Scales linearly with the number of rows
- Scales linearly with the number of nodes
- Scales well with the number of cores per node
- Scales well with the number of parameters
- Extremely high performance
GLM comparison using in-memory data: \texttt{glm()} and ScaleR’s \texttt{rxGlm()}

**GLM 'Gamma' Simulation Timings**

Independent Variables: 2 factors (100 and 20 levels) and one continuous

**Open Source R: glm()**
Single threaded and RAM-intensive

**RevoScaleR: rxGlm()**
Fast, parallelized, and scalable

Timings from a Windows 7, 64-bit quadcore laptop with 8 GB RAM
Allstate compares SAS, Hadoop and R for Big-Data Insurance Models

<table>
<thead>
<tr>
<th>Approach</th>
<th>Platform</th>
<th>Time to fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAS</td>
<td>16-core Sun Server</td>
<td>5 hours</td>
</tr>
<tr>
<td>rmr/MapReduce</td>
<td>10-node 80-core Hadoop Cluster</td>
<td>&gt; 10 hours</td>
</tr>
<tr>
<td>R</td>
<td>250 GB Server</td>
<td>Impossible (&gt; 3 days)</td>
</tr>
<tr>
<td>Revolution R Enterprise</td>
<td>5-node 20-core LSF cluster</td>
<td>5.7 minutes</td>
</tr>
</tbody>
</table>

Generalized linear model, 150 million observations, 70 degrees of freedom
http://blog.revolutionanalytics.com/2012/10/allstate-big-data-glm.html
## SAS HPA Benchmarking comparison*

### Logistic Regression

<table>
<thead>
<tr>
<th></th>
<th>SAS</th>
<th>Revolution R Enterprise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rows of data</td>
<td>1 billion</td>
<td>1 billion</td>
</tr>
<tr>
<td>Parameters</td>
<td>“just a few”</td>
<td>7</td>
</tr>
<tr>
<td>Time</td>
<td>80 seconds</td>
<td>44 seconds</td>
</tr>
<tr>
<td>Data location</td>
<td>In memory</td>
<td>On disk</td>
</tr>
<tr>
<td>Nodes</td>
<td>32</td>
<td>5</td>
</tr>
<tr>
<td>Cores</td>
<td>384</td>
<td>20</td>
</tr>
<tr>
<td>RAM</td>
<td>1,536 GB</td>
<td>80 GB</td>
</tr>
</tbody>
</table>

Revolution R Enterprise is faster on the same amount of data, despite using approximately a 20\textsuperscript{th} as many cores, a 20\textsuperscript{th} as much RAM, a 6\textsuperscript{th} as many nodes, and not pre-loading data into RAM.

Revolution R Enterprise Delivers Performance at **2% of the Cost**

*As published by SAS in HPC Wire, April 21, 2011*
Specific speed-related factors

- Efficient computational algorithms
- Efficient memory management – minimize data copying and data conversion
- Heavy use of C++ templates; optimal code
- Efficient data file format; fast access by row and column
- Models are pre-analyzed to detect and remove duplicate computations and points of failure (singularities)
- Handle categorical variables efficiently
ScaleR Parallel External Memory Algorithms (PEMA’s)

- The ScaleR analytics algorithms are all built on a platform (DistributeR) that efficiently parallelizes a broad class of statistical, data mining and machine learning algorithms.

- These Parallel External Memory Algorithms (PEMA’s) process data a chunk at a time in parallel across cores and nodes.

  1) Initialize, 2) Process Chunk, 3) Aggregate, 4) Finalize
Scalability and portability of Revolution Analytics’ implementation of PEMA’s

- These PEMA algorithms can process an unlimited number of rows of data in a fixed amount of RAM. They process a chunk of data at a time, giving linear scalability.

- They are independent of the “compute context” (number of cores, computers, distributed computing platform), giving portability across these dimensions.

- They are independent of where the data is coming from, giving portability with respect to data sources.
Simplified ScaleR Internal Architecture

Analytics Engine
PEMA’s are implemented here
(Scalable, Parallelized, Threaded, Distributable)

Inter-process Communication
MPI, RPC, Sockets, Files

Data Sources
HDFS, Teradata, ODBC, SAS, SPSS, CSV, Fixed, XDF
ScaleR on Hadoop

- Each pass through the data is one MapReduce job
- Prediction (Scoring), Transformation, Simulation:
  - Map tasks store results in HDFS or return to client
- Statistics, Model Building, Visualization:
  - Map tasks produce “intermediate result objects” that are aggregated by a Reduce task
  - Master process decides if another pass through the data is required
- Data can be cached or stored in XDF binary format for increased speed, especially on iterative algorithms
Sample code for logit on workstation

# Specify local data source
airData <- myLocalDataSource

# Specify model formula and parameters
rxLogit( ArrDelay>15 ~ Origin + Year + Month + DayOfWeek + UniqueCarrier + F(CRSDepTime), data=airData )
Sample code for logit on Hadoop

# Change the “compute context”
rxSetComputeContext(myHadoopCluster)

# Change the data source if necessary
airData <- myHadoopDataSource

# Otherwise, the code is the same
rxLogit(ArrDelay>15 ~ Origin + Year + Month + DayOfWeek + UniqueCarrier + F(CRSDepTime), data=airData)
Demo rxLinMod in Hadoop - Launching

```
# Set Compute Context
rxSetComputeContext(myHadoopCluster);

# Set Data Source
bigAirDS <- rxTextData(airDataName, useFastRead = TRUE, colInfo = airlineColInfo, varsToKeep = airlineVarsToKeep, fileSystem = hdfsFS)

# Run Linear Regression
delayArr <- rxLinMod(ArrDelay ~ DayOfWeek, data = bigAirDS, cube = TRUE)
summary(delayArr)
```

```
> # Set Compute Context
> rxSetComputeContext(myHadoopCluster);
>
> # Set Data Source
> bigAirDS <- rxTextData(airDataName, useFastRead = TRUE, colInfo = airlineColInfo,
>                        varsToKeep = airlineVarsToKeep, fileSystem = hdfsFS)
>
> # Run Linear Regression
> delayArr <- rxLinMod(ArrDelay ~ DayOfWeek, data = bigAirDS, cube = TRUE)
```
Demo **rxLinMod in Hadoop - In Progress**

```r
# Set Compute Context
taxSetComputeContext(myHadoopCluster);

# Set Data Source
bigAirDS <- RxTextData(airDataName, useFastRead = TRUE, colInfo = airlineColInfo, varsToKeep = airlineVarsToKeep, fileSystem = hdfsFS)

# Run Linear Regression
delayArr <- rxLinMod(ArrDelay ~ DayOfWeek, data = bigAirDS, cube = TRUE)
summary(delayArr)
```
Demo rxLinMod in Hadoop - Completed

```r
# Set Compute Context
rxSetComputeContext(myHadoopCluster);

# Set Data Source
bigAirDS <- rxTextData(airDataName, useFastRead = TRUE, colInfo =
                      airlineColInfo, varsToKeep = airlineVarsToKeep, fileSystem = hdfsFS)

# Run Linear Regression
delayArr <- rxLinMod(ArrDelay ~ DayOfWeek, data = bigAirDS, cube = TRUE)
summary(delayArr)
```

```
Revolution R Enterprise Console
Total independent variables: 7
Number of valid observations: 120947440
Number of missing observations: 2587529

Coefficients:

|         | Estimate | Std. Error | t value | Pr(>|t|) | Counts |
|---------|----------|------------|---------|----------|--------|
| DayOfWeek=1 | 6.669515 | 0.007288   | 915.1   | 2.22e-16 | 17750849 |
| DayOfWeek=2 | 5.960421 | 0.007310   | 815.4   | 2.22e-16 | 17643973 |
| DayOfWeek=3 | 7.091502 | 0.007299   | 971.6   | 2.22e-16 | 17697936 |
| DayOfWeek=4 | 8.945047 | 0.007302   | 1225.0  | 2.22e-16 | 17683723 |
```
Revolution R Enterprise 7 on Hadoop

- Revolution R Enterprise 7 on Hadoop and Analytics Clusters
  - “Right Tool For The Job”
  - RRE 7 “Inside” and “Beside” Hadoop
    - Connect a Compute Server or Cluster to Hadoop
- When To Use:
  - Production Hadoop Cluster
  - Need Parallelized Algorithms
  - Heavy Random Workloads
  - Extensive “Sandboxing”
  - Big Data Scoring
  - Data Security Constraints
  - Legacy Data Sources
- Advantages:
  - Independent Scalability
  - Flexibility
  - Low Latency
Consulting Services and Training

Services
Remote & On site
Projects & Staff Aug
Quick Start Programs
Entire project lifecycle

Training
Comprehensive Topics
Self Paced & Classroom
Customizable

✔️ Big Data Analytics Strategy
✔️ Design & Architecture
✔️ Use Case Definition
✔️ Model Development & Deployment
✔️ Support & Maintenance

✔️ R with Hadoop
✔️ R for SAS Users
✔️ Data Visualization
✔️ Parallel Computing with RRE
✔️ Big Data Analytics with RRE
Polling Question 3
Questions
Contact Revolution Analytics at info@revolutionanalytics.com