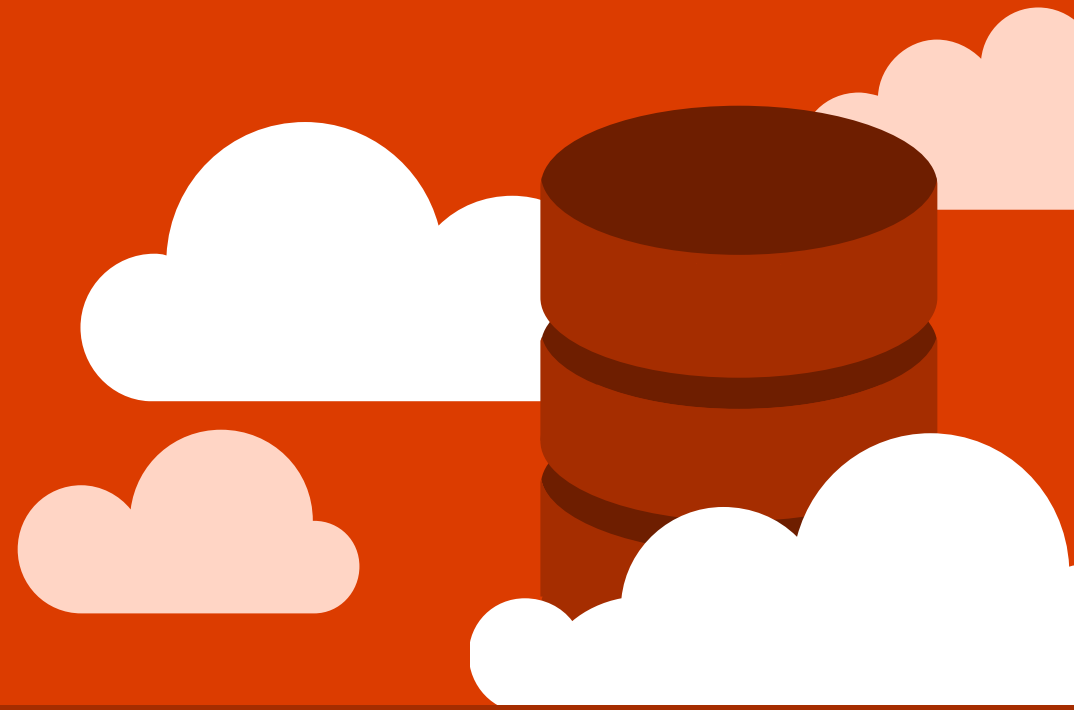


Using Microsoft R Server to Address Scalability Issues



February 4th, 2016 - Welcome!

Introduction

Microsoft R Open

Microsoft R Server

Demo

Derek McCrae Norton

Senior Data Scientist

Microsoft

@dermcnor



R – What is it?

Open Source
"lingua franca"



Analytics, Computing,
Modeling

Global Community



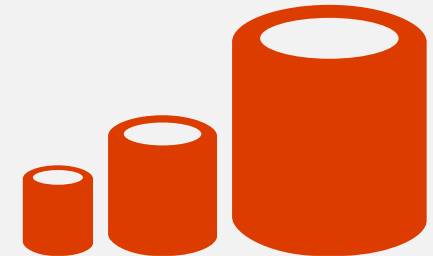
Millions of users

Ecosystem



7000+ Algorithms, Test
Data & Evaluations

Can be Scaled to
Big Data,
Big Analytics



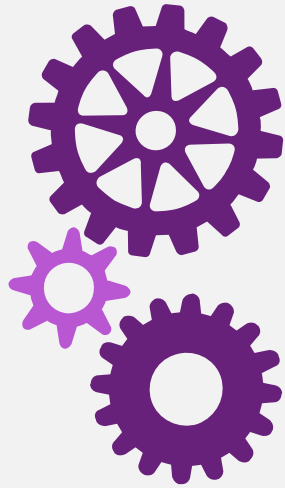
Scalability

R from Microsoft brings

Peace of
mind



Efficiency



Speed and
scalability



Flexibility
and agility



Microsoft R Products

Microsoft R Open

- Free and open source R distribution
- Enhanced and distributed by Revolution Analytics

SQL Server R Services

- Built in Advanced Analytics and Stand Alone Server Capability
- Leverages the Benefits of SQL 2016 Enterprise Edition

Microsoft R Server

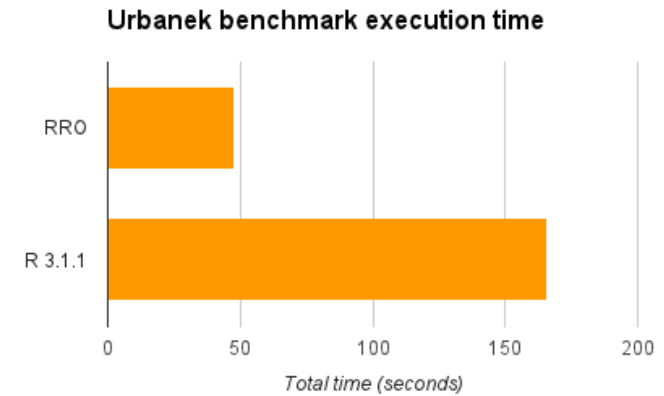
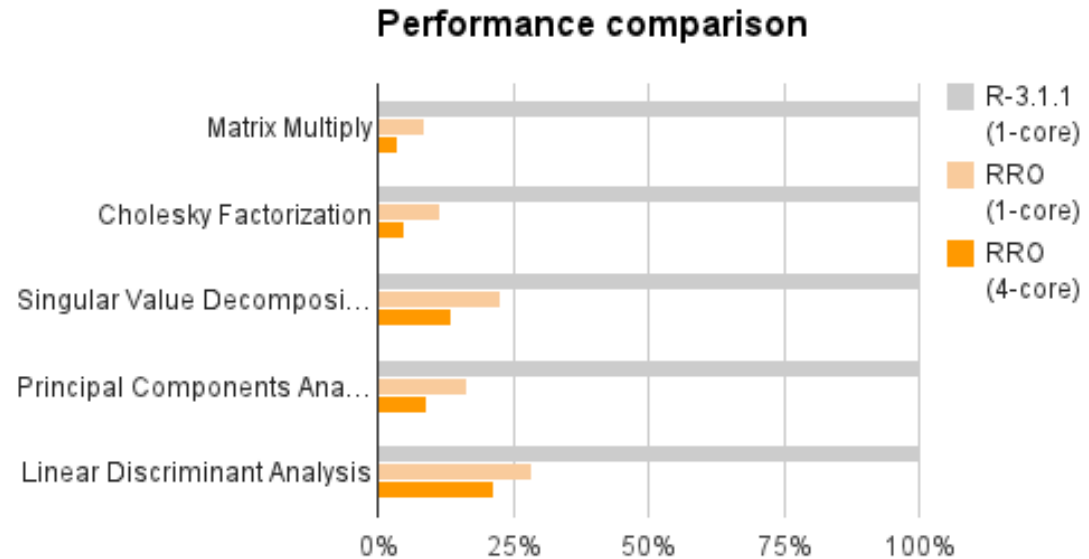
- Microsoft R Server for Redhat Linux
- Microsoft R Server for SUSE Linux
- Microsoft R Server for Teradata DB
- Microsoft R Server for Hadoop on Redhat

Microsoft R Open

Introducing Microsoft R Open

- Enhanced Open Source R distribution
 - Based on the latest Open Source R (3.2.2)
 - Built, tested and distributed by Microsoft
 - Enhanced by Intel MKL Library to speed up linear algebra functions
- Compatible with all R-related software
 - CRAN packages, RStudio, third-party R integrations, ...
- Revolutions Open-Source R packages
 - Reproducible R Toolkit – Checkpoint , miniCRAN
 - ParallelR – parallelise execution via 'foreach' loop
 - Rhadoop – rhdfs, rhbase, ravro, rmr2, plymr
 - AzureML – read/write data to AzureML, publish R code as ML API
- MRAN website mran.revolutionanalytics.com
 - Enhanced documentation and learning resources
 - Discover 6500 free add-on R packages
- Open source (GPLv2 license) - 100% free to download, use and share

CRAN R compared to Microsoft R Open



- More efficient and multi-threaded math computation.
- Benefits math intensive processing.
- No benefit to program logic and data transform

- Matrix calculation – up to 27x faster
- Matrix functions – up to 16x faster
- Programation – 0x faster

Microsoft R Server

CRAN, MRO, MRS Comparison



**Microsoft
R Open**

**Microsoft
R Server**

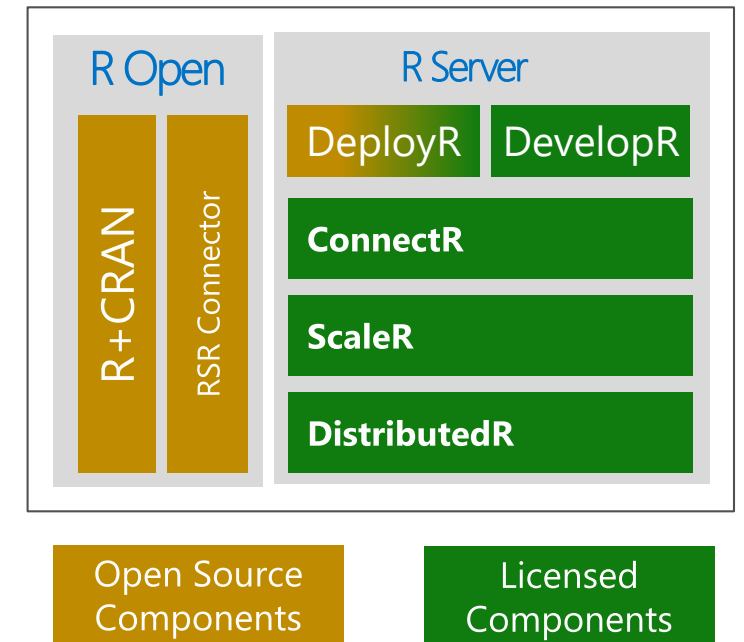
Datasize	In-memory	In-memory	In-Memory or Disk Based
Speed of Analysis	Single threaded	Multi-threaded	Multi-threaded, parallel processing 1:N servers
Support	Community	Community	Community + Commercial
Analytic Breadth & Depth	7500+ innovative analytic packages	7500+ innovative analytic packages	7500+ innovative packages + commercial parallel high-speed functions
License	Open Source	Open Source	Commercial license. Supported release with indemnity

Introducing Microsoft R Server

Microsoft R Server is a broadly deployable enterprise-class analytics platform based on R that is supported, scalable and secure. Supporting a variety of big data statistics, predictive modeling and machine learning capabilities, R Server supports the full range of analytics – exploration, analysis, visualization and modeling

High-performance open source R plus:

- Data source connectivity to big-data objects
- Big-data advanced analytics
- Multi-platform environment support
- In-Hadoop and in-Teradata predictive modeling
- Development and production environment support
 - IDE for data scientist developers
 - Secure, Scalable R Deployment



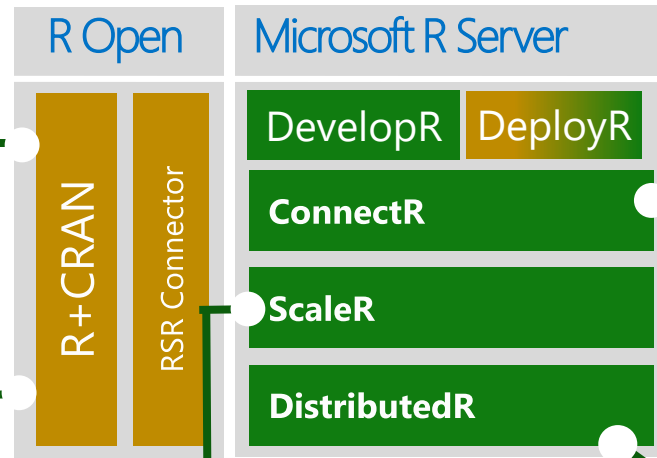
The Microsoft R Server Platform

R+CRAN

- Open source R interpreter
 - R 3.1.2
- Freely-available huge range of R algorithms
- Algorithms callable by Revor
- Embeddable in R scripts
- 100% Compatible with existing R scripts, functions and packages

MRO

- Performance enhanced R interpreter
- Based on open source R
- Adds high-performance math library to speed up linear algebra functions



ScaleR

- Ready-to-Use high-performance big data big analytics
- Fully-parallelized analytics
- Data prep & data distillation
- Descriptive statistics & statistical tests
- Range of predictive functions
- User tools for distributing customized R algorithms across nodes
- Wide data sets supported – thousands of variables

ConnectR

- High-speed & direct connectors

Available for:

- High-performance XDF
- SAS, SPSS, delimited & fixed format text data files
- Hadoop HDFS (text & XDF)
- Teradata Database & Aster
- EDWs and ADWs
- ODBC

DistributedR

- Distributed computing framework
- Delivers cross-platform portability

Scale R – Parallelized Algorithms & Functions

Data Preparation

- Data import – Delimited, Fixed, SAS, SPSS, ODBC
- Variable creation & transformation
- Recode variables
- Factor variables
- Missing value handling
- Sort, Merge, Split
- Aggregate by category (means, sums)

Descriptive Statistics

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

Statistical Tests

- Chi Square Test
- Kendall Rank Correlation
- Fisher's Exact Test
- Student's t-Test

Sampling

- Subsample (observations & variables)
- Random Sampling

Predictive Models

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

Variable Selection

- Stepwise Regression

Simulation

- Simulation (e.g. Monte Carlo)
- Parallel Random Number Generation

Cluster Analysis

- K-Means

Classification

- Decision Trees
- Decision Forests
- Gradient Boosted Decision Trees
- Naïve Bayes



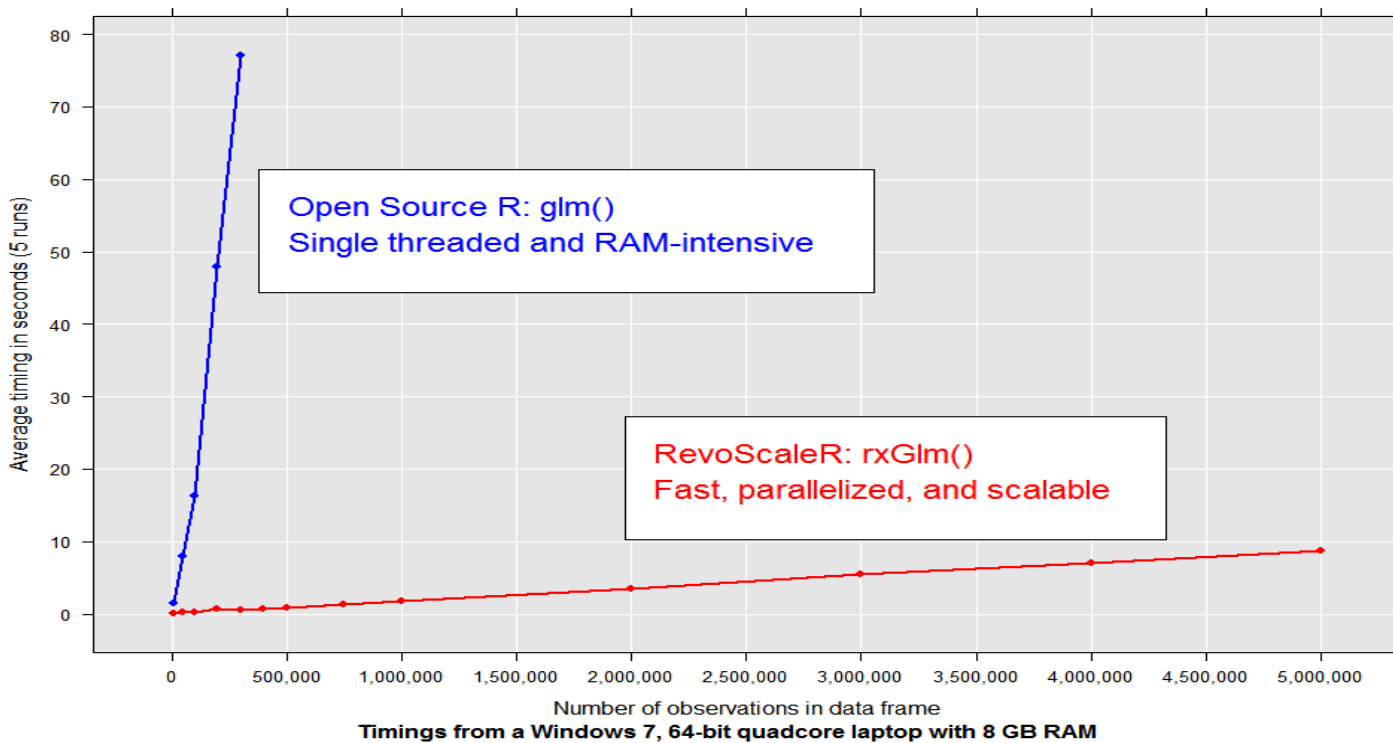
Combination

- rxDataStep
- rxExec
- PEMA-R API Custom Algorithms

ScaleR - Performance comparison

Microsoft R Server has no data size limits in relation to size of available RAM. When open source R operates on data sets that exceed RAM it will fail. In contrast Microsoft R Server scales linearly well beyond RAM limits and parallel algorithms are much faster.

GLM 'Gamma' Simulation Timings
Independent Variables: 2 factors (100 and 20 levels) and one continuous

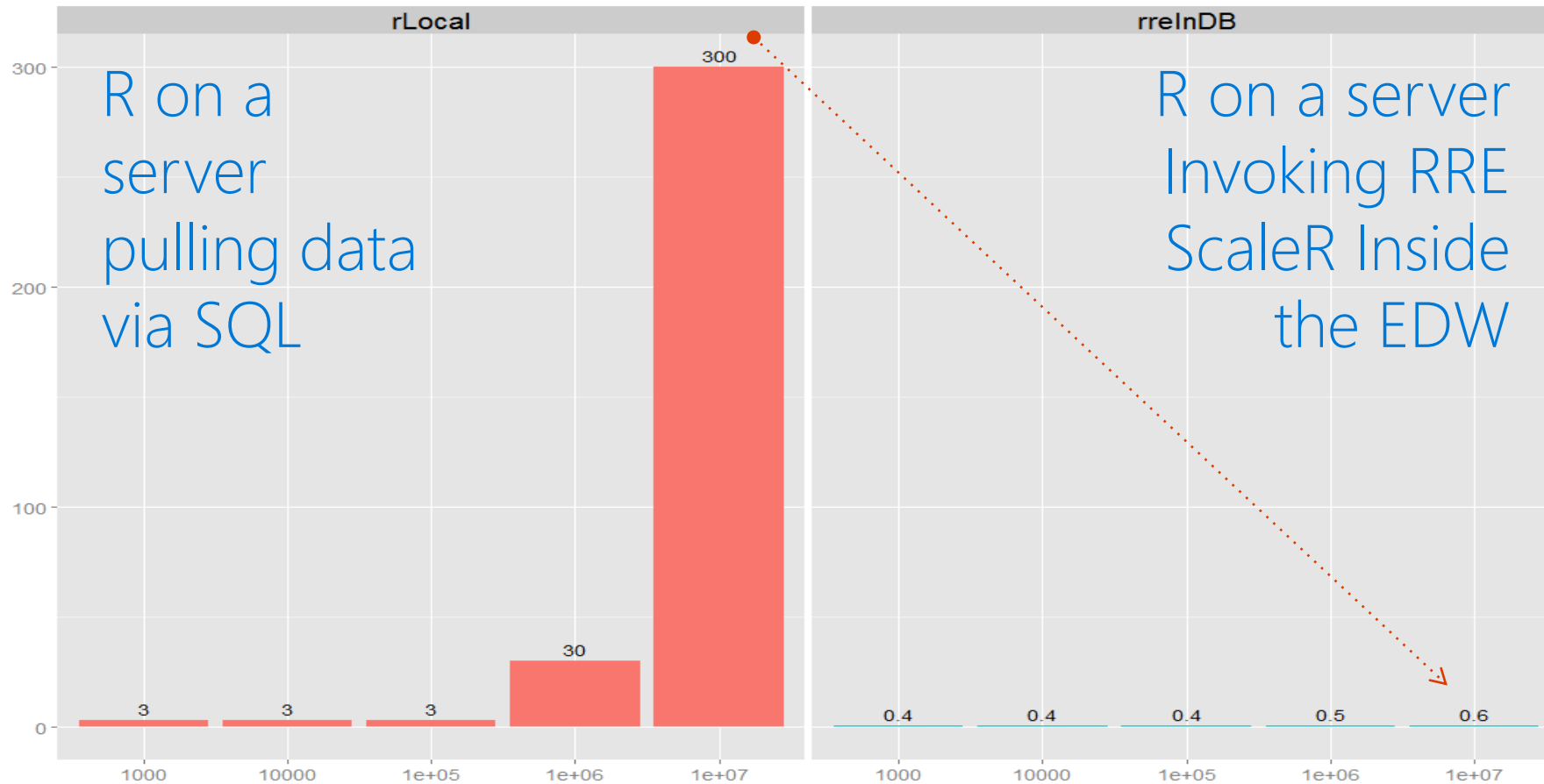


File Name	Compressed File Size (MB)	No. Rows	Open Source R (secs)	Revolution R (secs)
Tiny	0.3	1,235	0.00	0.05
V. Small	0.4	12,353	0.21	0.05
Small	1.3	123,534	0.03	0.03
Medium	10.7	1,235,349	1.94	0.08
Large	104.5	12,353,496	60.69	0.42
Big (full)	12,960.0	123,534,969	Memory!	4.89
V.Big	25,919.7	247,069,938	Memory!	9.49
Huge	51,840.2	494,139,876	Memory!	18.92

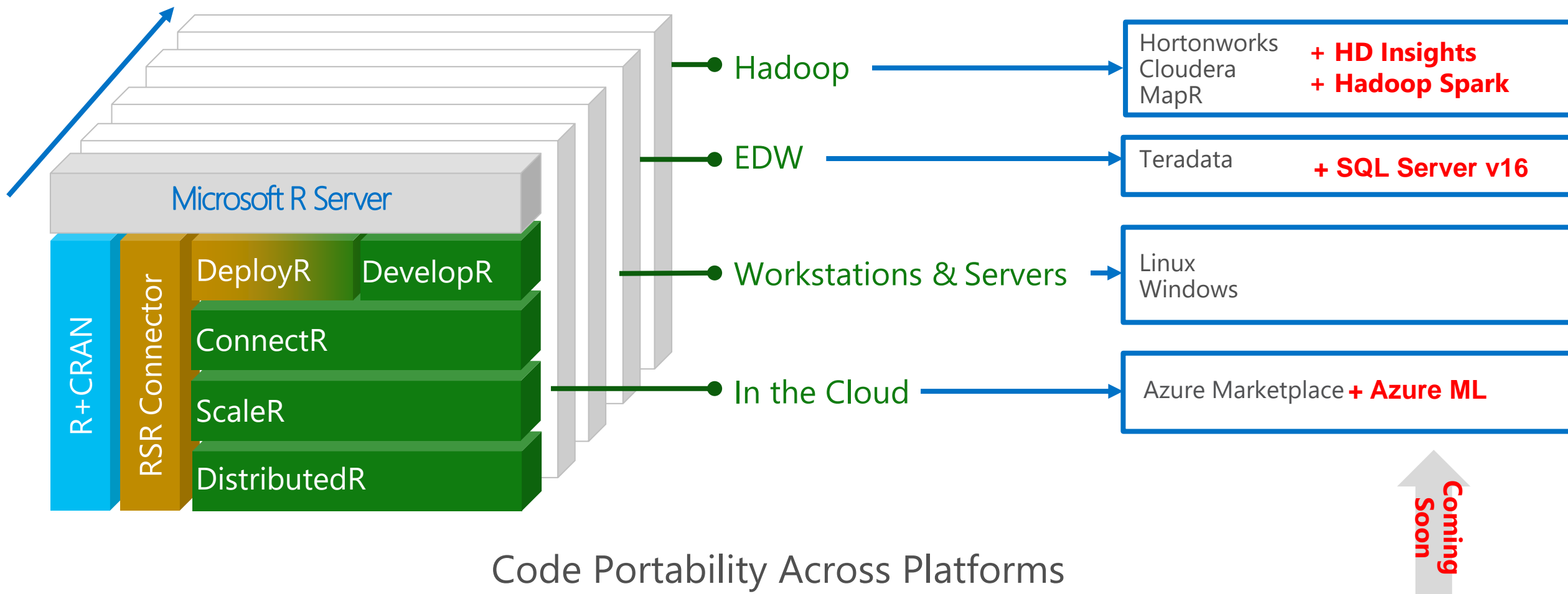
- US flight data for 20 years
- Linear Regression on Arrival Delay
- Run on 4 core laptop, 16GB RAM and 500GB SSD

Example of In-Database Acceleration

- 5+ hours to 40 seconds:



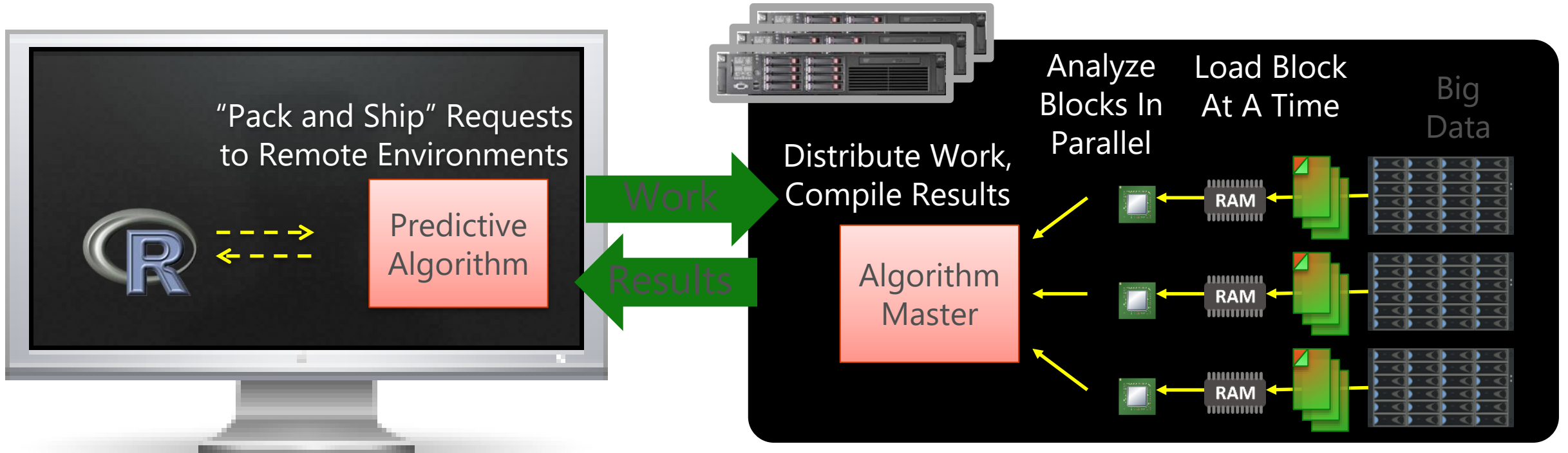
Distributed R - "Write Once. Deploy Anywhere."



Demo

Appendix

DistributedR - Remote Execution



Microsoft R Server functions

- A compute context defines remote connection
- Microsoft R functions prefixed with `rx`
- Current compute context determines processing location

The Results:

- Even Faster Computation
- Larger Data Set Capacity
- Fewer Security Concerns
- No Data Movement, No Copies

DistributedR - Revolution Code Portability

ScaleR models can be deployed **from a server or edge node to run in Hadoop** without any functional R model re-coding for map-reduce

Local Parallel processing – **Linux or Windows**

```
### SETUP LOCAL ENVIRONMENT VARIABLES ###
myLocalCC <- "localpar"

### LOCAL COMPUTE CONTEXT ###
rxSetComputeContext(myLocalCC)

### CREATE LINUX, DIRECTORY AND FILE OBJECTS ###
linuxFS <- RxNativeFileSystem() )
AirlineDataSet <-
RxXdfData("AirlineDemoSmall/AirlineDemoSmall.xdf",
fileSystem = linuxFS)
```

In – **Hadoop**

```
### SETUP HADOOP ENVIRONMENT VARIABLES ###
myHadoopCCC <- RxHadoopMR()

### HADOOP COMPUTE CONTEXT ###
rxSetComputeContext(myHadoopCC)

### CREATE HDFS, DIRECTORY AND FILE OBJECTS ###
hdfsFS <- RxHdfsFileSystem()
```

```
### ANALYTICAL PROCESSING ###
### Statistical Summary of the data
rxSummary(~ArrDelay+DayOfWeek, data= AirlineDataSet, reportProgress=1)

### CrossTab the data
rxCrossTabs(ArrDelay ~ DayOfWeek, data= AirlineDataSet, means=T)

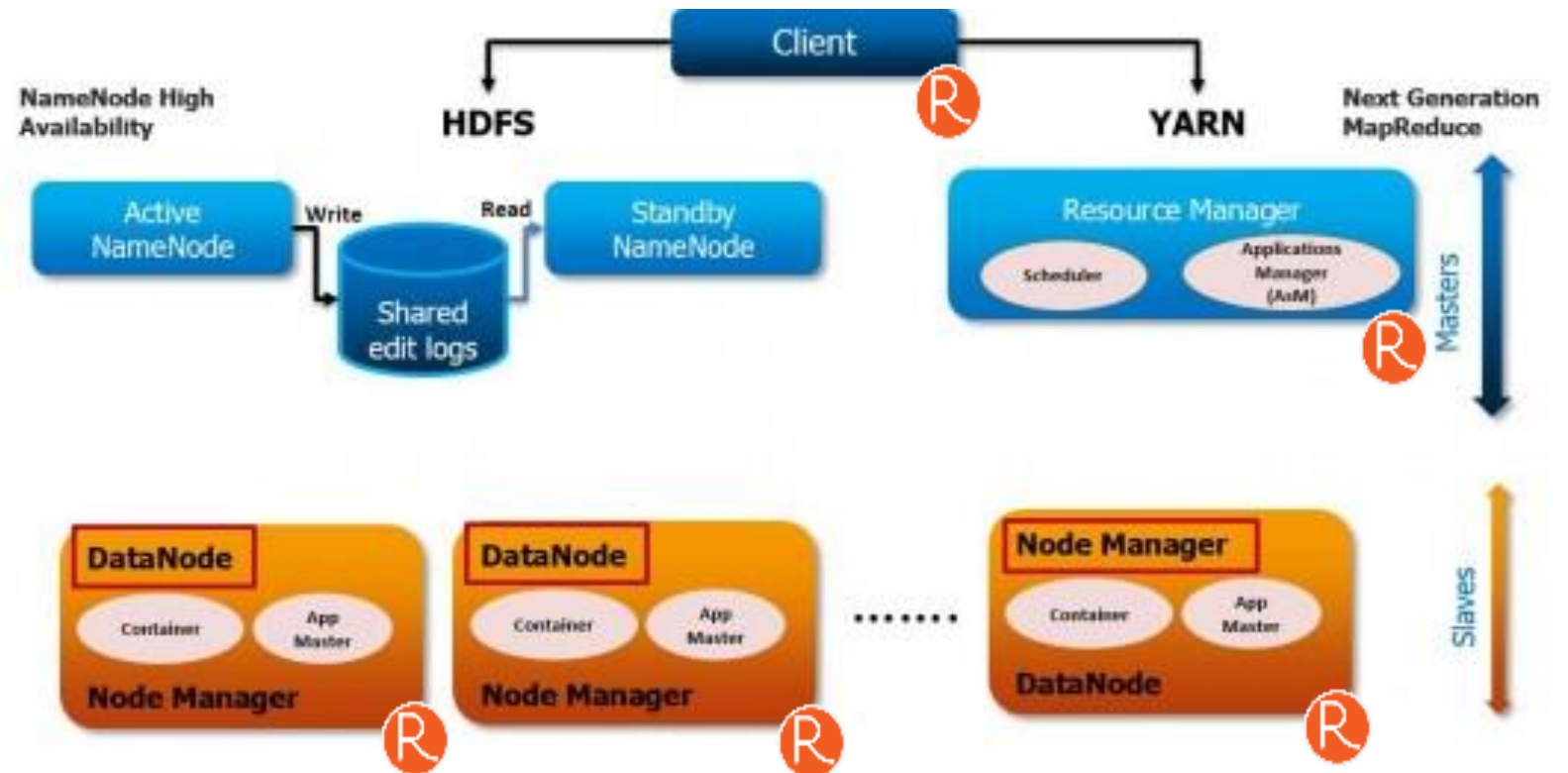
### Linear Model and plot
hdfsXdfArrLateLinMod <- rxLinMod(ArrDelay ~ DayOfWeek + 0 , data = AirlineDataSet)
plot(hdfsXdfArrLateLinMod$coefficients)
```

Compute
context R script
– sets where the
model will run

Functional
model R script –
does not need
to change to run
in Hadoop

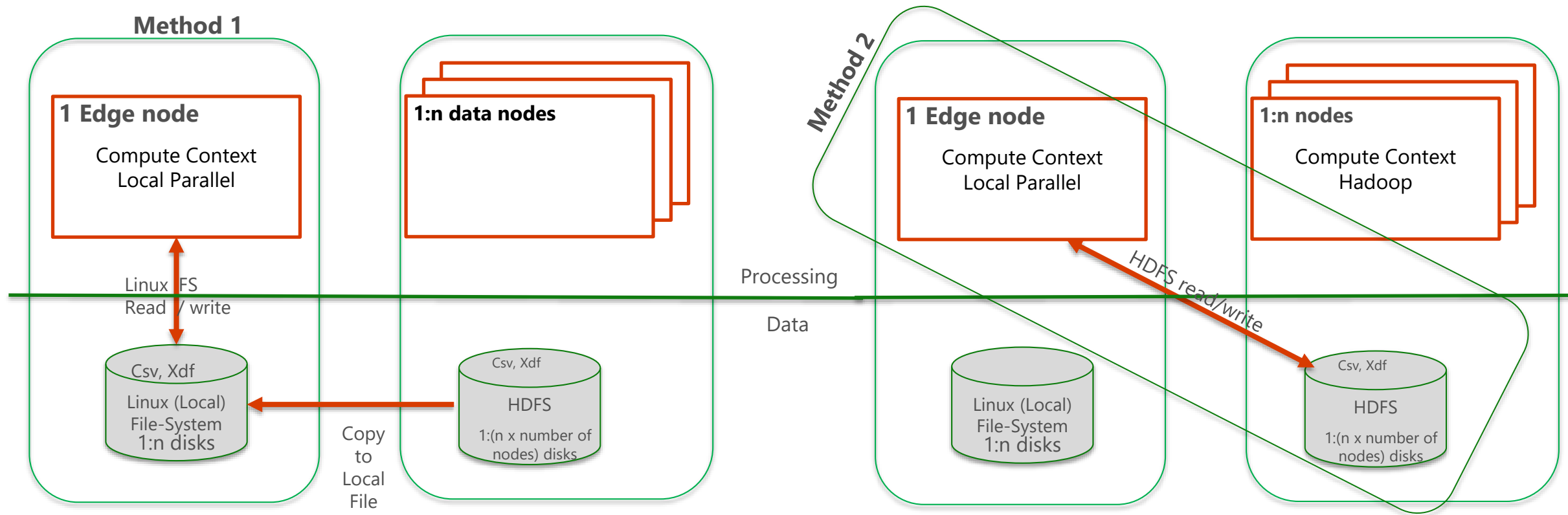
DistributedR - In-Hadoop

- Uses Hadoop nodes for R computations
- Eliminate data movement latency on very large data
- Remove data duplication
- Faster model development
- No MapReduce R coding
- Develop better models using all the data



 = Microsoft R Server

DistributedR - Hadoop Processing Methods

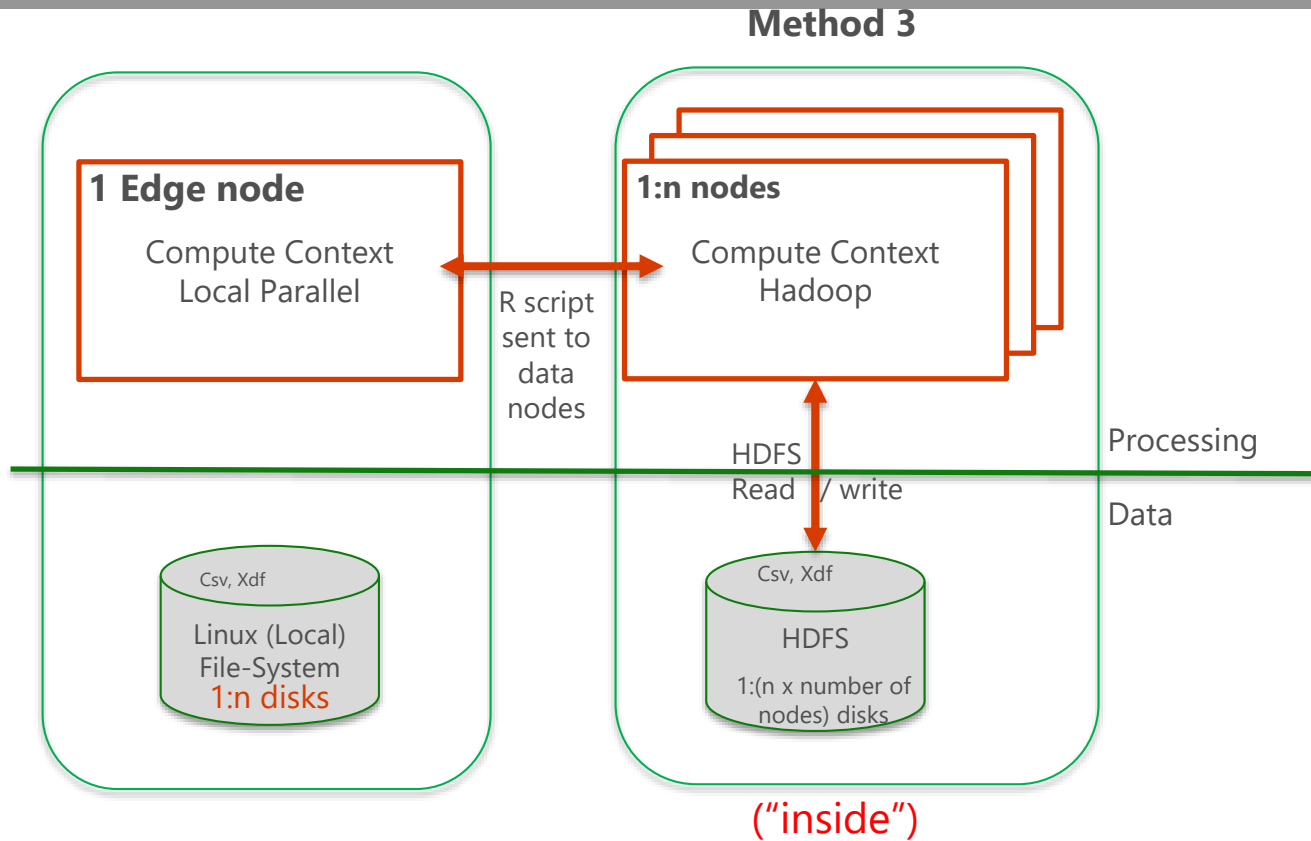


Method 1: Local (Linux) parallel processing using all cores on one node, **copying** data from HDFS to store in local Linux file-system.

Method 2: Local (Linux) parallel processing using all cores on one node, **streaming** data from / to HDFS

("Beside" or "Edge")

DistributedR - Hadoop Processing Methods



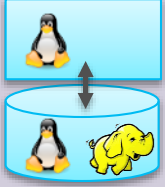
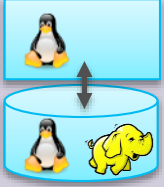
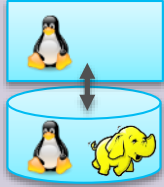
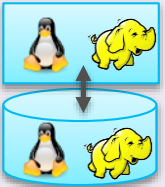
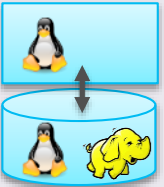
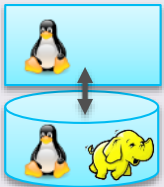
Method 3: Hadoop (Map-Reduce) parallel processing using all cores on n nodes, using HDFS data on each node

R model script sent to Master Node:

1. Starts a master process
2. Distribute work
3. Master tasks for each node
4. Master initiates distributed work
 1. Hadoop schedules mapper for each split
 2. Algorithm computes intermediate result
 3. Reducer combines intermediate results
5. Master process evaluates completion
6. Iterates as required by the algorithm
7. Returns consolidated answer to script

DistributedR - What processing mode to use?

Analytic data set size and processing complexity (e.g. simple summary statistics vs iterative algorithm) guide the use of Method 1 and 2 (Edge Node / Server Linux local processing) vs Method 3 (in-Hadoop processing)

Processing Complexity \ Data Size	Low	Medium	High
Small Data < 10GB			
Medium Data < 50GB			
Bigger Data > 50GB	