Scalable Data Science with Hadoop, Spark and R

Mario Inchiosa, PhD

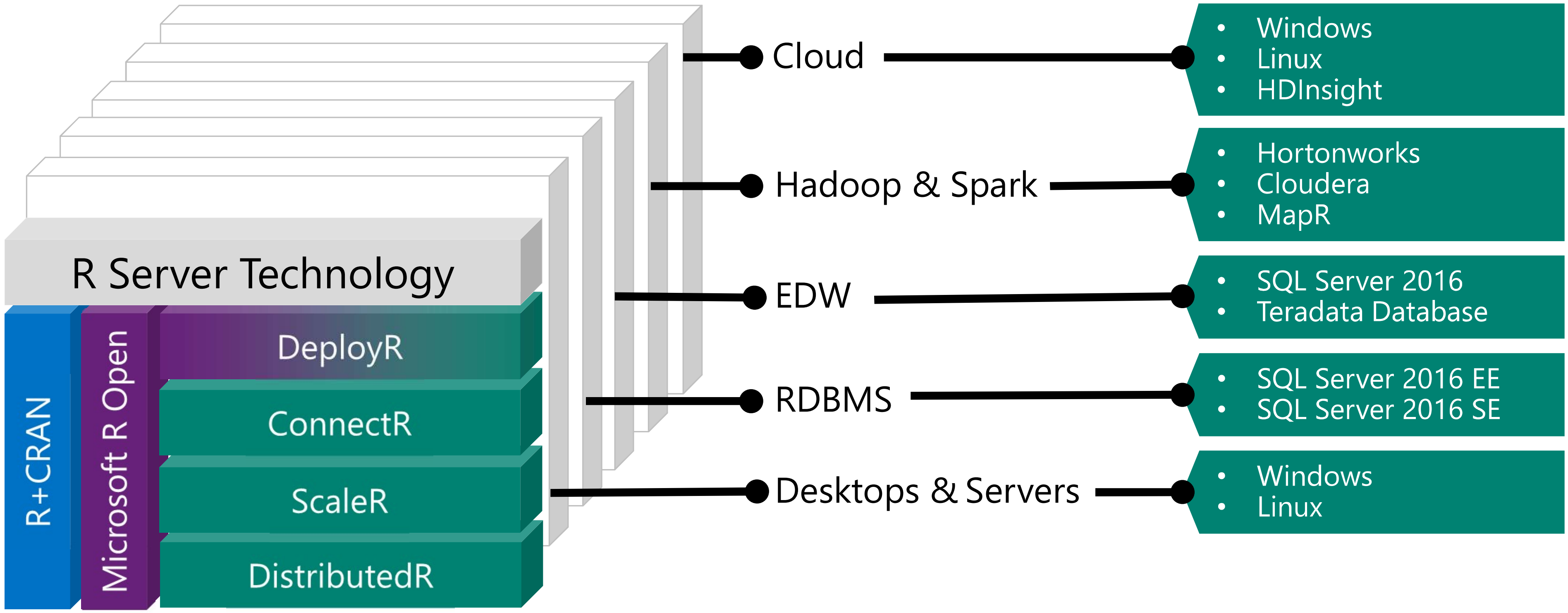
Principal Software Engineer

Microsoft Data Group

DSC 2016 July 2, 2016

Microsoft

Microsoft R Server



R Server Technology

R+CRAN

Microsoft R Open

DeployR

ConnectR

ScaleR

DistributedR

Cloud
- Windows
- Linux
- HDInsight

Hadoop & Spark
- Hortonworks
- Cloudera
- MapR

EDW
- SQL Server 2016
- Teradata Database

RDBMS
- SQL Server 2016 EE
- SQL Server 2016 SE

Desktops & Servers
- Windows
- Linux

R Server “Parallel External Memory Algorithms” (PEMAs)

- The initialize() method of the master Pema object is executed
- The master Pema object is serialized and sent to each worker process
- The worker processes call processData() once for each chunk of data
  - The fields of the worker’s Pema object are updated from the data
  - In addition, a data frame may be returned from processData(), and will be written to an output data source
  - When a worker has processed all of its data, it sends its reserialized Pema object back to the master (or an intermediate combiner)
- The master process loops over all of the Pema objects returned to it, calling updateResults() to update its Pema object
- processResults() is then called on the master Pema object to convert intermediate results to final results
- hasConverged(), whose default returns TRUE, is called, and either the results are returned to the user or another iteration is started

R Script for Execution in MapReduce

Define Compute Context

Sample R Script:

Define Data Source

```
rxSetComputeContext( RxHadoopMR(…) )

inData <- RxTextData(“/ds/AirOnTime.csv”, fileSystem = hdfsFS)

model <- rxLogit(ARR_DEL15 ~ DAY_OF_WEEK + UNIQUE_CARRIER, data = inData)
```

Train Predictive Model

Easy to Switch From MapReduce to Spark

Change the Compute Context

Sample R Script:

Keep other code unchanged

```
rxSetComputeContext( RxSpark(…) )

inData <- RxTextData(“/ds/AirOnTime.csv”, fileSystem = hdfsFS)

model <- rxLogit(ARR_DEL15 ~ DAY_OF_WEEK + UNIQUE_CARRIER, data = inData)
```

R Server: scale-out R

- 100% compatible with open source R
  - Any code/package that works today with R will work in R Server
- Wide range of scalable and distributed R functions
  - Examples: rxDataStep(), rxSummary(), rxGlm(), rxDForest(), rxPredict()
- Ability to parallelize any R function
  - Ideal for parameter sweeps, simulation, scoring

Parallelized & Distributed Algorithms

ETL

- 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)

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

Predictive Statistics

- 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
- Predictions/scoring for models
- Residuals for all models

Variable Selection

- Stepwise Regression

Machine Learning

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

Clustering

- K-Means

Sampling

- Subsample (observations & variables)
- Random Sampling

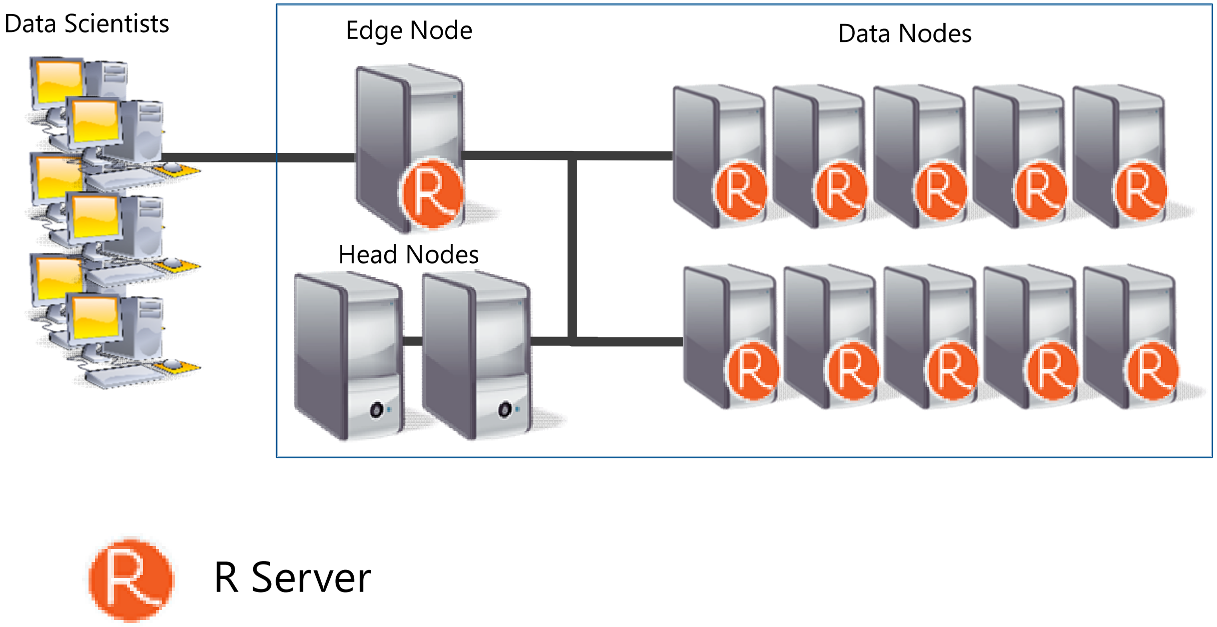
Simulation

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

Custom Parallelization

- rxDataStep
- rxExec
- PEMA-R API

R Server Hadoop Architecture



1. R Server Local Processing:

Data in Distributed Storage

↓

R process on Edge Node

2. R Server Distributed Processing:

Master R process on Edge Node

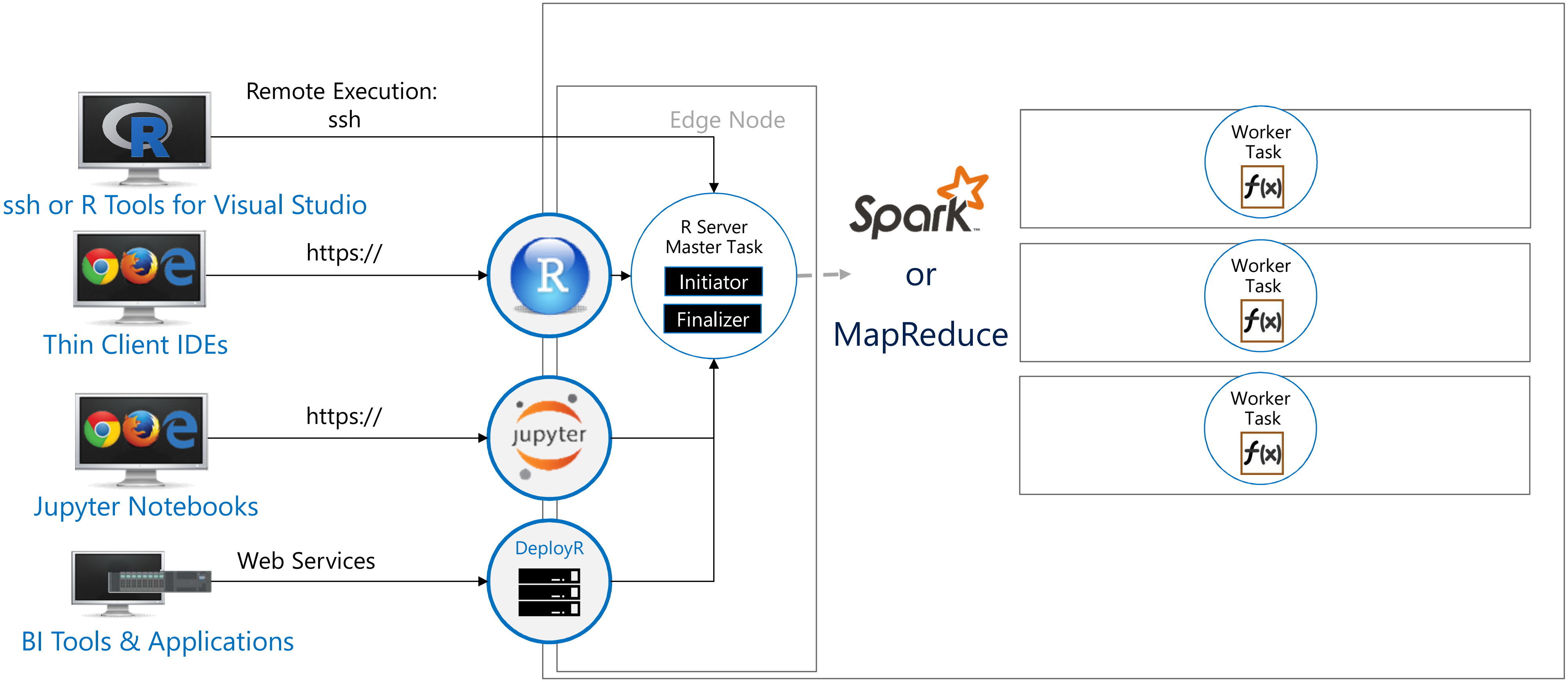
↓

Apache YARN and Spark

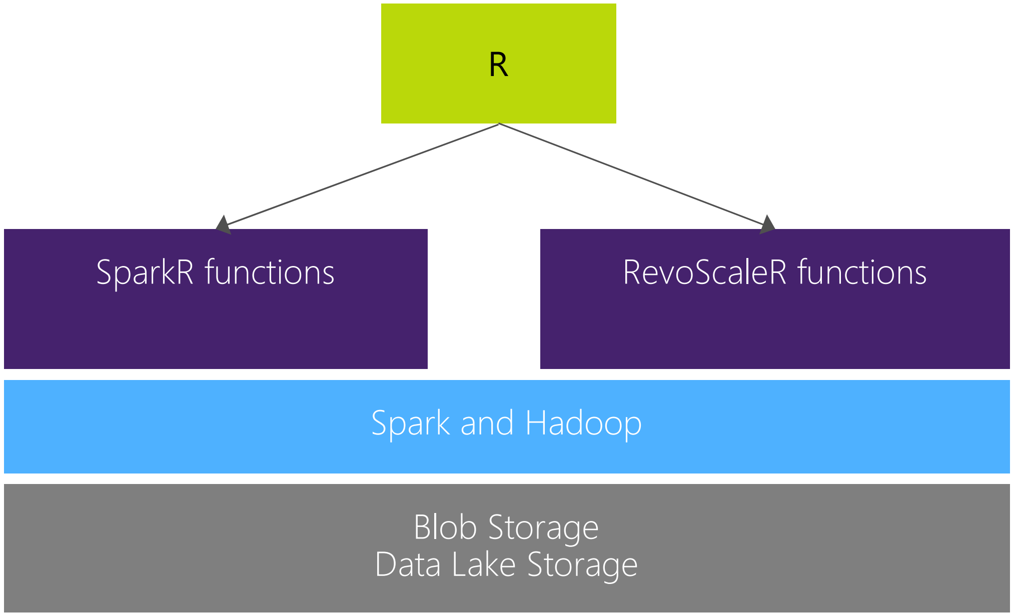
↓

Worker R processes on Data Nodes

R Server for Hadoop - Connectivity



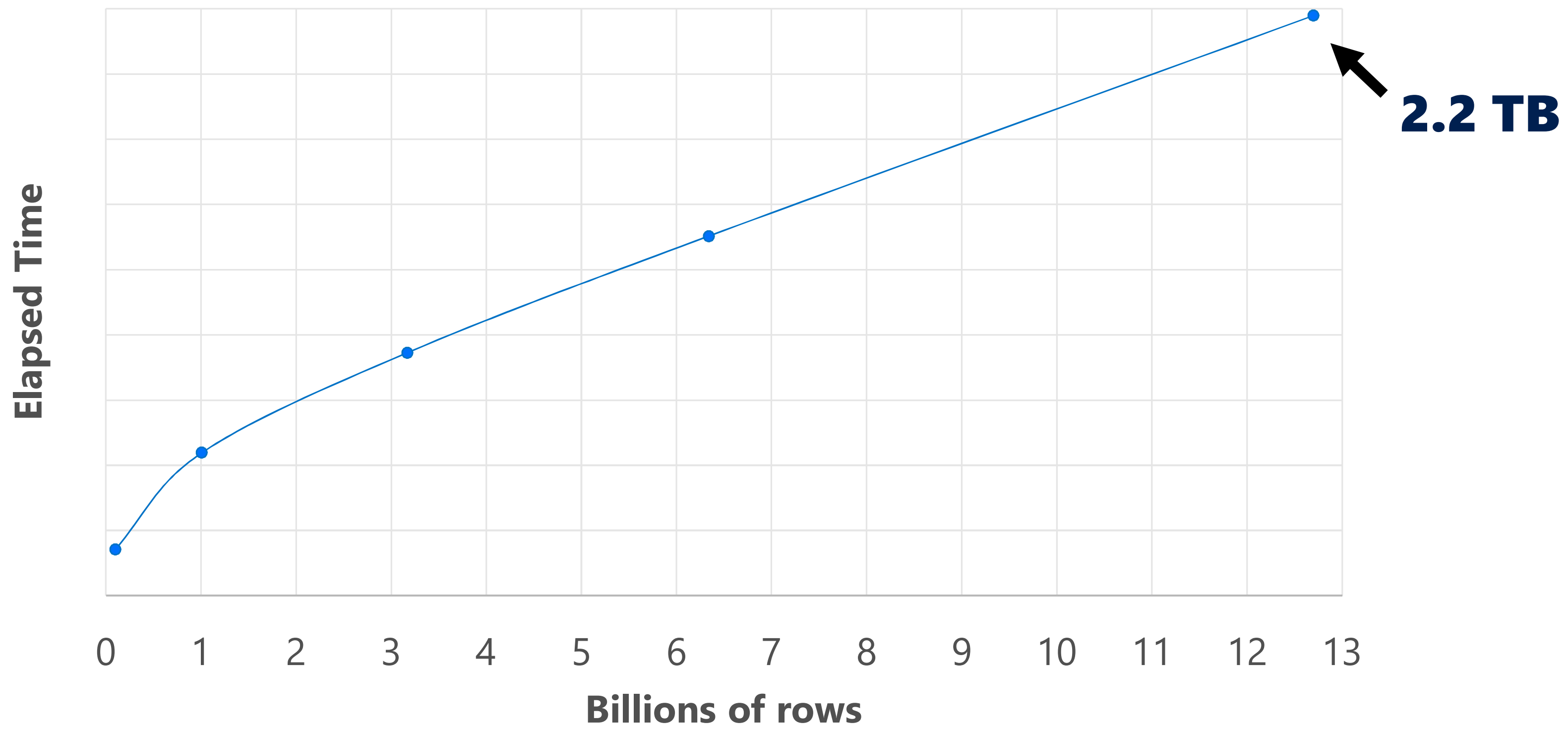
HDInsight + R Server: Managed Hadoop for Advanced Analytics in the Cloud



- Easy setup, elastic, SLA
- Spark
  - Integrated notebooks experience
  - Upgraded to latest Version 1.6.1
- R Server
  - Leverage R skills with massively scalable algorithms and statistical functions
  - Reuse existing R functions over multiple machines

R Server on Hadoop/HDInsight scales to hundreds of nodes, billions of rows and terabytes of data

Logistic Regression on NYC Taxi Dataset

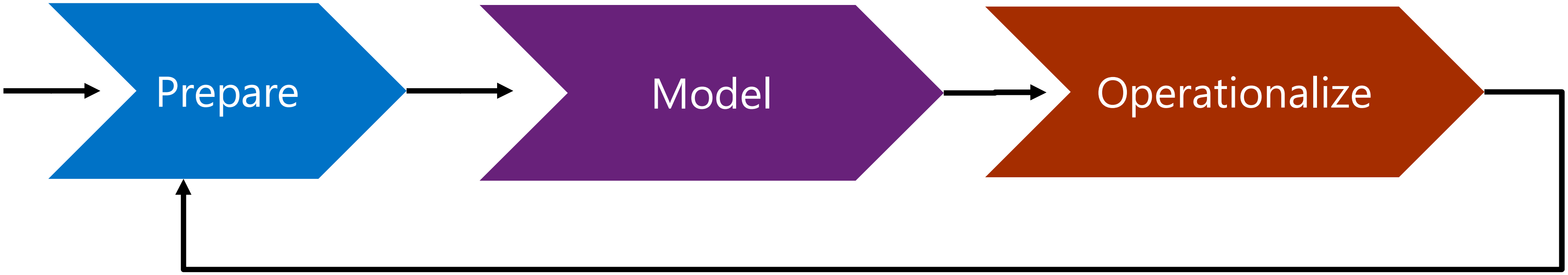


Typical advanced analytics lifecycle

Prepare: Assemble, cleanse, profile and transform diverse data relevant to the subject.

Model: Use statistical and machine learning algorithms to build classifiers and regression models

Operationalize: Make predictions and visualizations to support business applications



Airline Arrival Delay Prediction Demo

- Clean/Join – Using SparkR from R Server
- Train/Score/Evaluate – Scalable R Server functions
- Deploy/Consume – Using AzureML from R Server

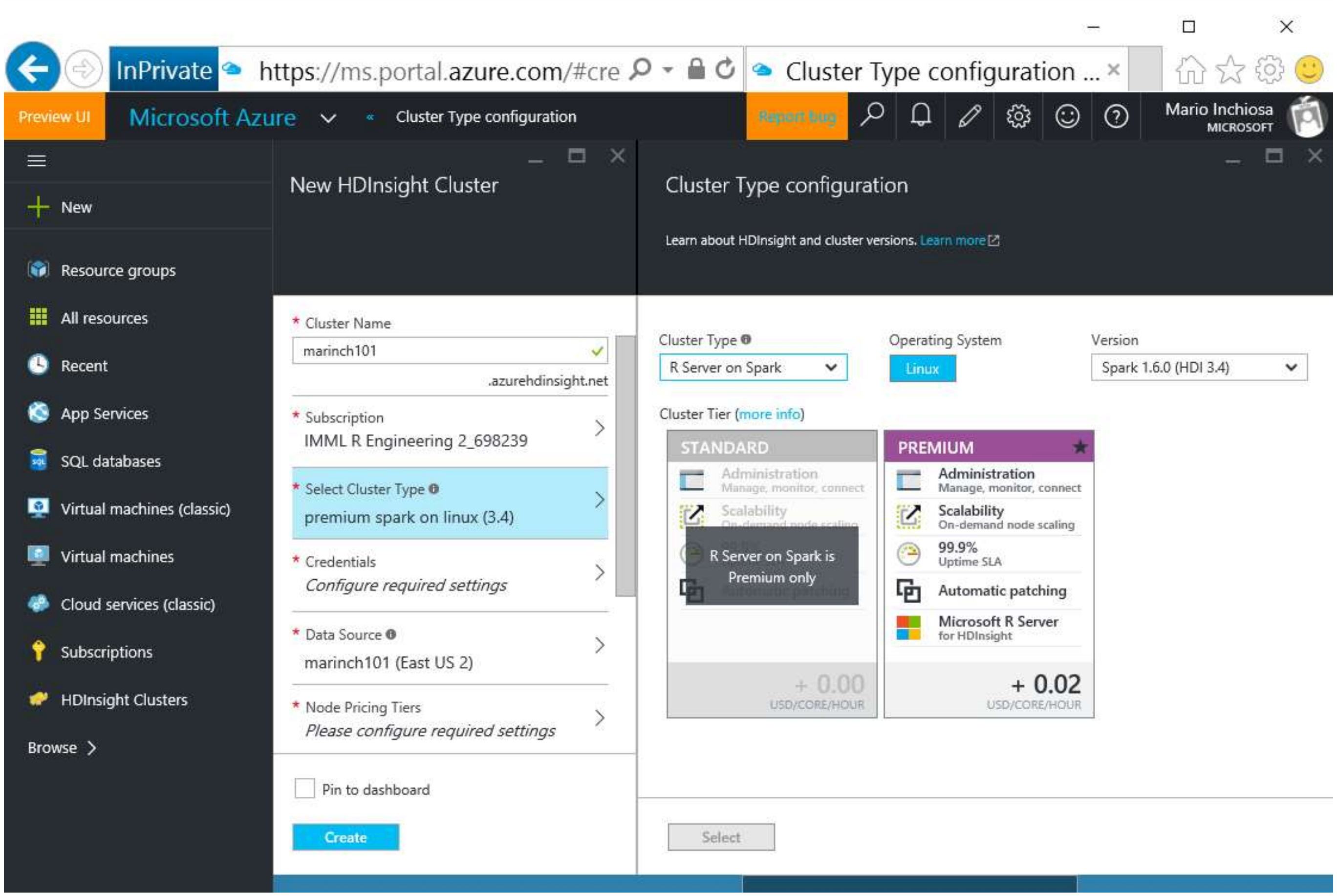
Airline data set

- Passenger flight on-time performance data from the US Department of Transportation's TranStats data collection
- >20 years of data
- 300+ Airports
- Every carrier, every commercial flight
- http://www.transtats.bts.gov

Weather data set

- Hourly land-based weather observations from NOAA
- > 2,000 weather stations
- http://www.ncdc.noaa.gov/orders/qclcd/

Provisioning a cluster with R Server



https://ms.portal.azure.com/#cre

Cluster Type configuration ...

Microsoft Azure « Cluster Type configuration

New HDInsight Cluster

* Cluster Name
marinch101 ✓
.azurehdinsight.net

* Subscription
IMML R Engineering 2_698239

* Select Cluster Type
premium spark on linux (3.4)

* Credentials
Configure required settings

* Data Source
marinch101 (East US 2)

* Node Pricing Tiers
Please configure required settings

□ Pin to dashboard

Create

Cluster Type configuration

Learn about HDInsight and cluster versions. Learn more

Cluster Type	Operating System	Version
R Server on Spark	Linux	Spark 1.6.0 (HDI 3.4)

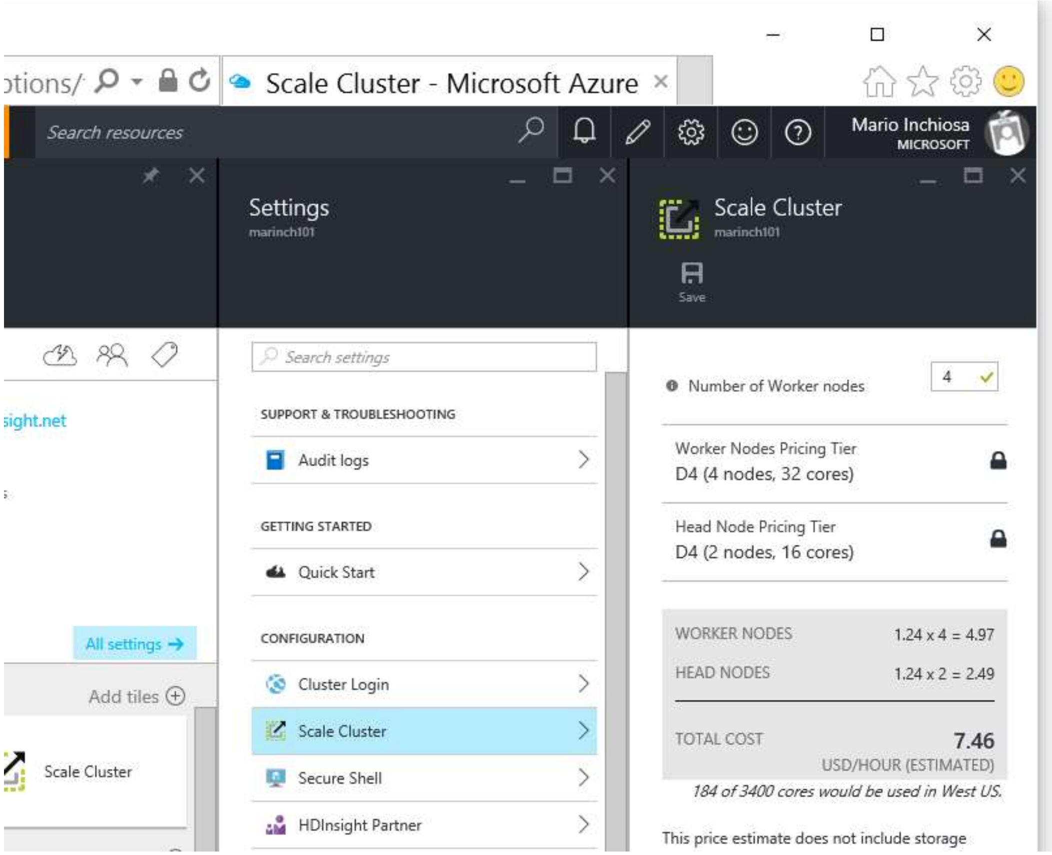
Cluster Tier (more info)

STANDARD	PREMIUM ★
Administration Manage, monitor, connect	Administration Manage, monitor, connect
Scalability On-demand node scaling	Scalability On-demand node scaling
99.9% Uptime SLA	99.9% Uptime SLA
Automatic patching	Automatic patching
	Microsoft R Server for HDInsight
+ 0.00 USD/CORE/HOUR	+ 0.02 USD/CORE/HOUR

R Server on Spark is Premium only

Select

Scaling a cluster



Number of Worker nodes: 4 ✓

Worker Nodes Pricing Tier
D4 (4 nodes, 32 cores)

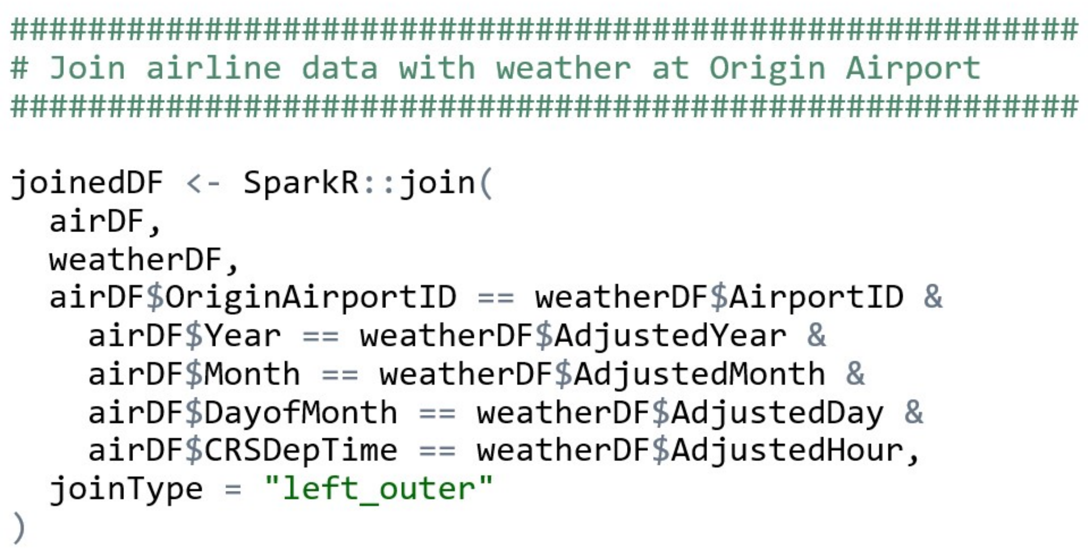
Head Node Pricing Tier
D4 (2 nodes, 16 cores)

WORKER NODES	1.24 x 4 = 4.97
HEAD NODES	1.24 x 2 = 2.49
TOTAL COST	7.46 USD/HOUR (ESTIMATED)

184 of 3400 cores would be used in West US.

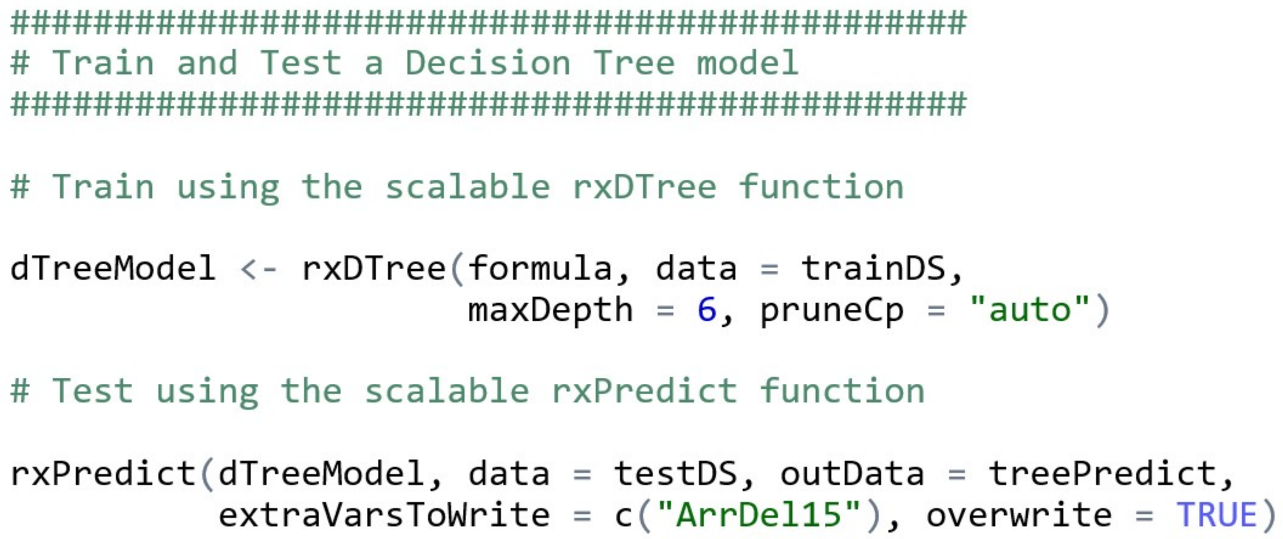
This price estimate does not include storage

Clean and Join using SparkR in R Server



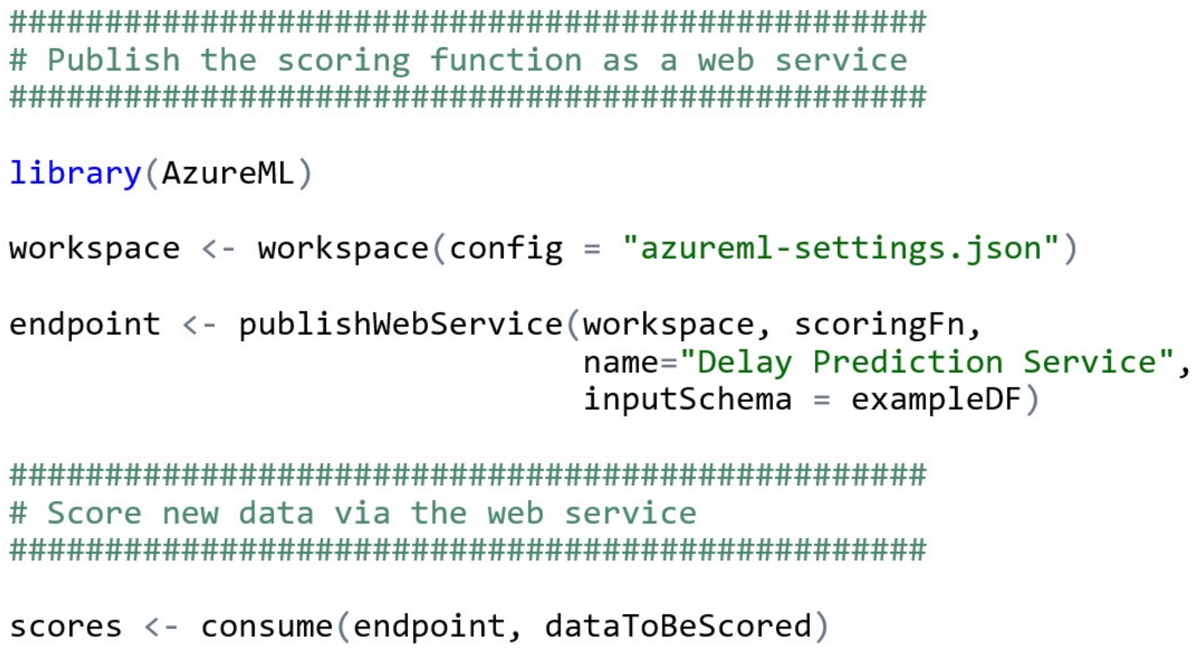
```
#######################################################
# Join airline data with weather at Origin Airport
#######################################################

joinedDF <- SparkR::join(
  airDF,
  weatherDF,
  airDF$OriginAirportID == weatherDF$AirportID &
    airDF$Year == weatherDF$AdjustedYear &
    airDF$Month == weatherDF$AdjustedMonth &
    airDF$DayofMonth == weatherDF$AdjustedDay &
    airDF$CRSDepTime == weatherDF$AdjustedHour,
  joinType = "left_outer"
)
```

Train, Score, and Evaluate using R Server



```
##############################################
# Train and Test a Decision Tree model
##############################################

# Train using the scalable rxDTree function

dTreeModel <- rxDTree(formula, data = trainDS,
                      maxDepth = 6, pruneCp = "auto")

# Test using the scalable rxPredict function

rxPredict(dTreeModel, data = testDS, outData = treePredict,
          extraVarsToWrite = c("ArrDel15"), overwrite = TRUE)
```

Publish Web Service from R

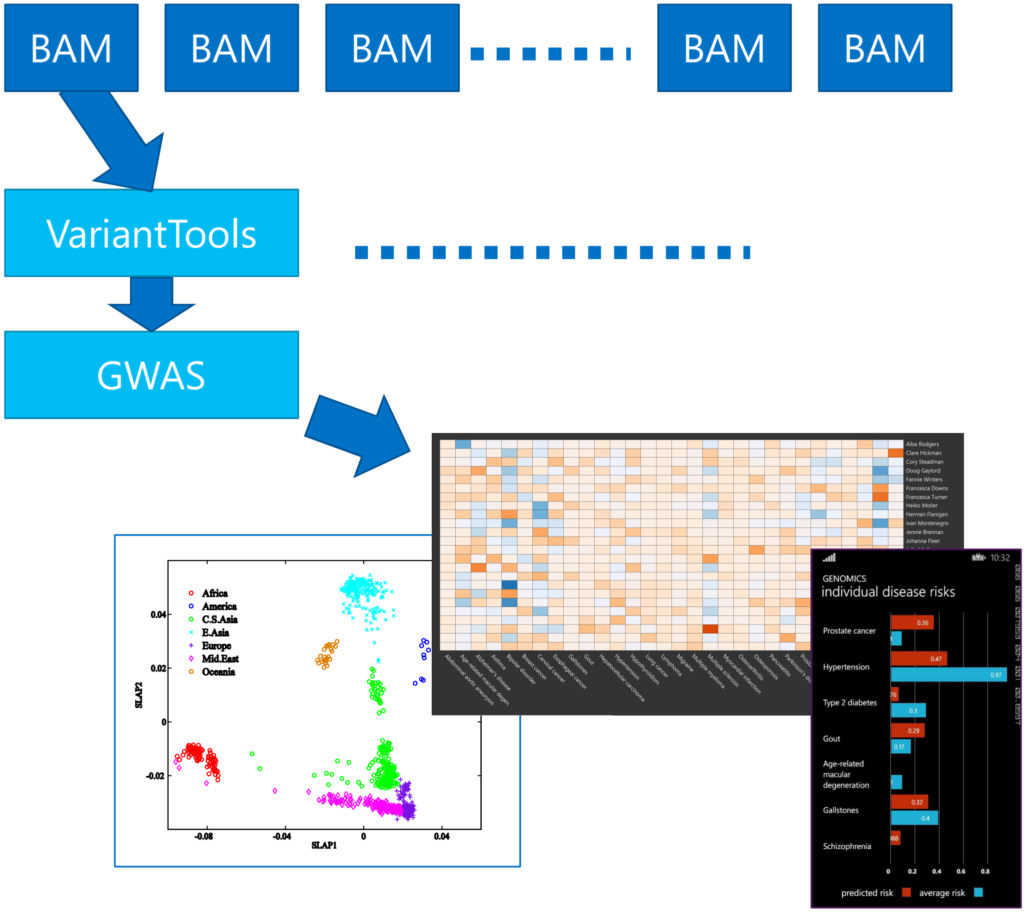


```r
################################################
# Publish the scoring function as a web service
################################################

library(AzureML)

workspace <- workspace(config = "azureml-settings.json")

endpoint <- publishWebService(workspace, scoringFn,
                              name="Delay Prediction Service",
                              inputSchema = exampleDF)

################################################
# Score new data via the web service
################################################

scores <- consume(endpoint, dataToBeScored)
```

Demo Technologies

- HDInsight Premium Hadoop cluster
- Spark on YARN distributed computing
- R Server R interpreter
- SparkR data manipulation functions
- RevoScaleR Statistical & Machine Learning functions
- AzureML R package and Azure ML web service

Building a genetic disease risk application with R


BAM BAM BAM BAM BAM

VariantTools

GWAS

Data

- Public genome data from 1000 Genomes
- About 2TB of raw data

Platform

- HDInsight Hadoop (8 clusters)
  - 1500 cores, 4 data centers
- Microsoft R Server

Processing

- VariantTools R package (Bioconductor)
- Match against NHGRI GWAS catalog

Analytics

- Disease Risk
- Ancestry

Presentation

- Expose as Web Service APIs
- Phone app, Web page, Enterprise applications

Microsoft

For more information…

R Server
microsoft.com/r-server

HDInsight Premium
microsoft.com/hdinsight