

Using R for Scalable Data Science

Robert Horton, Mario Inchiosa, Vanja Paunic, and Hang Zhang
Microsoft Corporation

Acknowledgements: Gopi Kumar, Paul Shealy, Katherine Zhao

KDD 2017
Halifax, NS

TUTORIAL MATERIAL & SLIDES:

<https://aka.ms/kdd2017r>

Tutorial Outline

- Introduction to R for Scalable Data Science
- R in SQL Server
- R in Apache Spark
- Distributed model training and parameter optimization use cases
 - Learning Curves: Detecting Gibberish
 - Grouped Time Series Forecasting
- Sentiment Analysis with Pretrained Deep Learning Model

What is



Language
Platform

- The most popular statistical programming language
- A data visualization tool
- Open source

Community

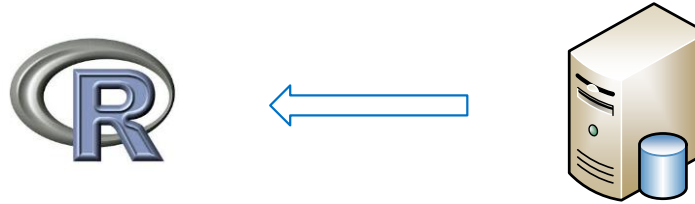
- 2.5+M users
- Taught in most universities
- Many common use cases across industry
- Thriving user groups worldwide
 - 5th in 2016 IEEE Spectrum rank
 - 42% pro analysts prefer R (highest amongst R, SAS, python)

Ecosystem

- 10,000+ contributed packages
- Rich application & platform integration

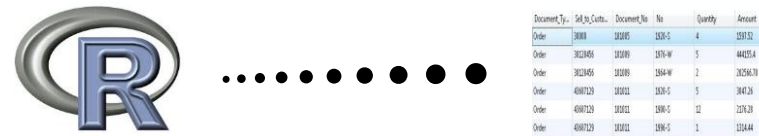
Challenges with Embracing Open Source R

Data Movement



Moving data from the database to the R Runtime becomes painful as data volumes grow and carries security risks

Scale/Performance



R runs single threaded and only accommodates datasets that fit into available memory

4 Operationalization



How do I call the R script from my production application?

Scalable R Solutions

- R packages for scaling on single machines
 - The **bigmemory** project
 - **ff** and related packages
 - **foreach** with **doParallel**
- R packages for scaling with distributed computing
 - **SparkR**
 - **sparklyr**
 - **RevoScaleR** (Microsoft R Server)
 - **h2o**, **rsparkling**
 - **foreach** with **doAzureParallel**, **doSNOW**

SQL Server R Services

Reduce or eliminate data movement with In-Database analytics

- SQL Server 2016 extensibility mechanism allows secure execution of R scripts on the SQL Server

Operationalize R scripts and models

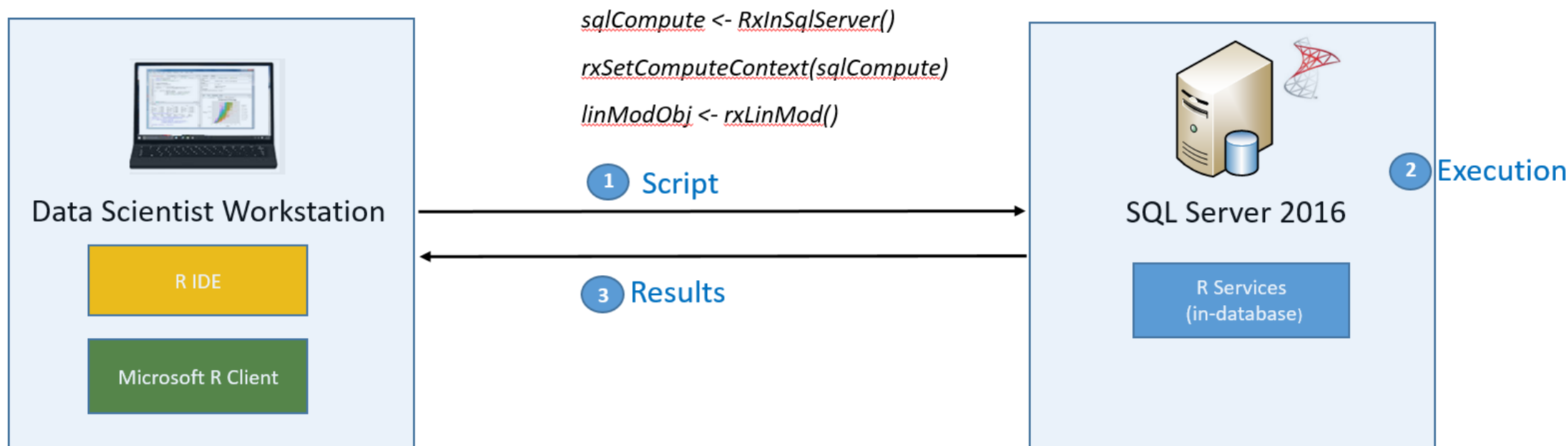
- Use familiar T-SQL stored procedures to invoke R scripts from your application
- Embed the returned predictions and plots in your application

Enterprise Performance and scale

- Use SQL Server's in-memory querying and Columnstore Indexes
- Leverage RevoScaleR support for large datasets and parallel algorithms
- Easily deploy to Spark cluster by just changing compute context

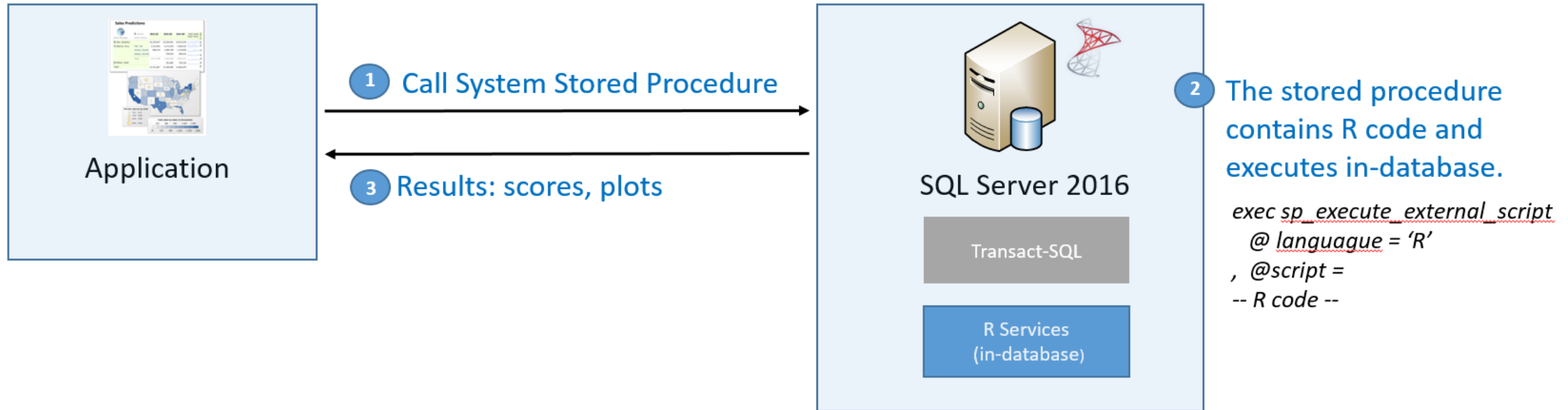
R Services in-database: Data Exploration and Predictive Modeling ('Data Scientist')

Working from my R IDE on my workstation, I can execute an R script that runs in-database, and get the results back.

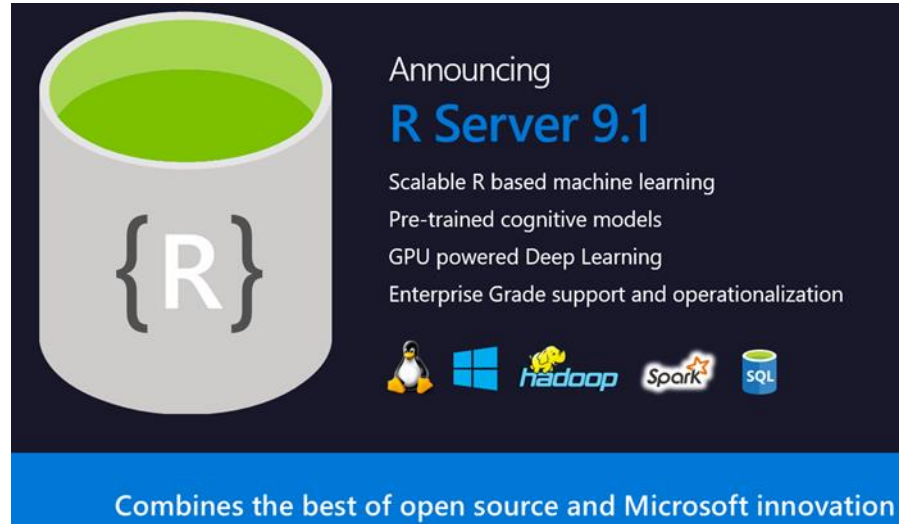


R Services in-database: Operationalizing R Code via T-SQL ('Developer')

I can call a T-SQL System Stored Procedure from my application and have it trigger R script execution in-database. Results are then returned to my application (predictions, plots, etc).



Free Developer's Versions Available

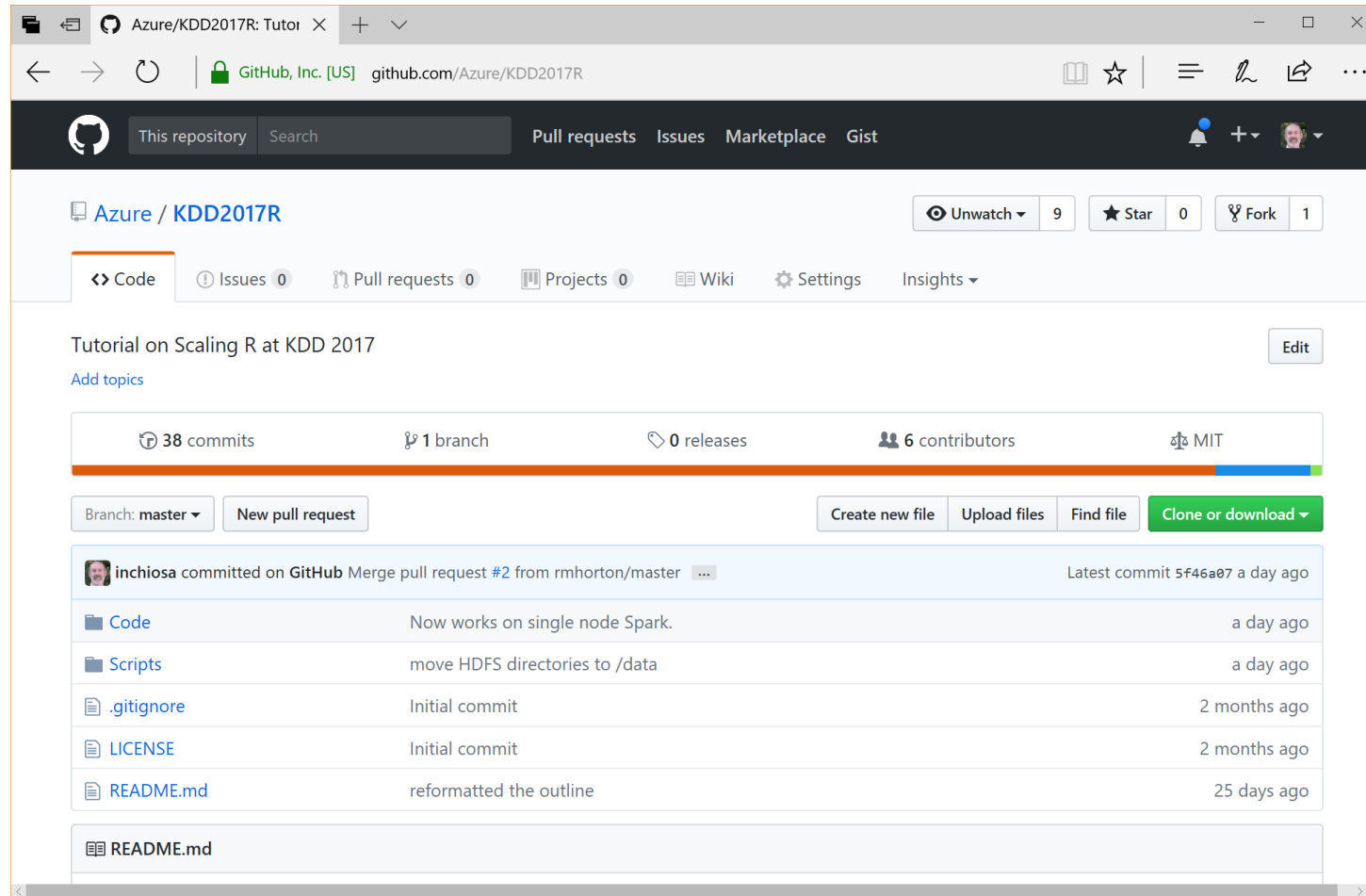


<https://aka.ms/freemrs>



<https://aka.ms/sqlserverdeveloper>

GitHub repository for all code and scripts



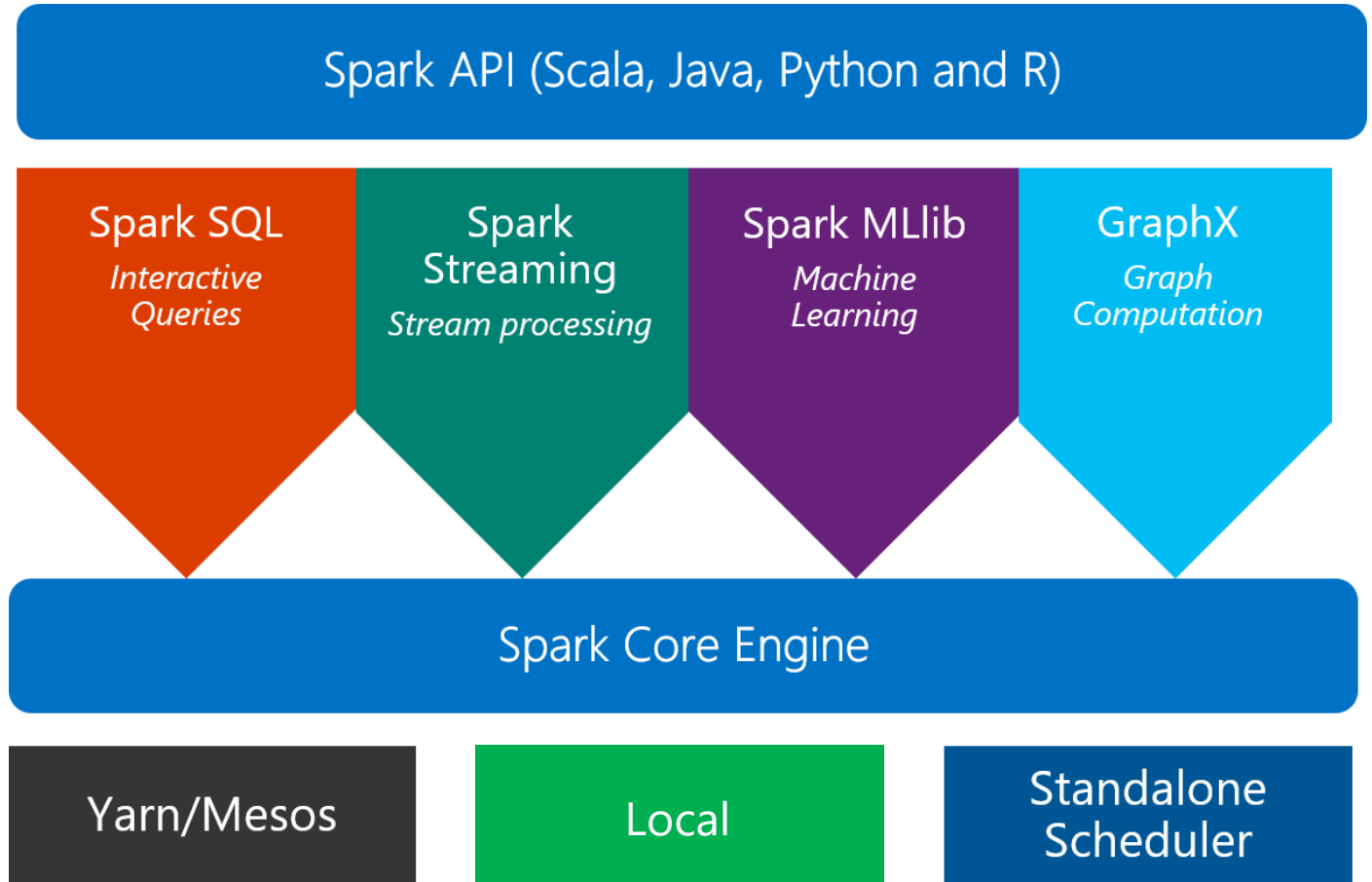
<https://aka.ms/kdd2017r>

Distributed computing on Spark

Brief intro to Spark, its APIs and OS R packages

Scaling R on Spark Clusters

- What is Spark?
 - An unified, open source, parallel, data processing framework for Big Data Analytics



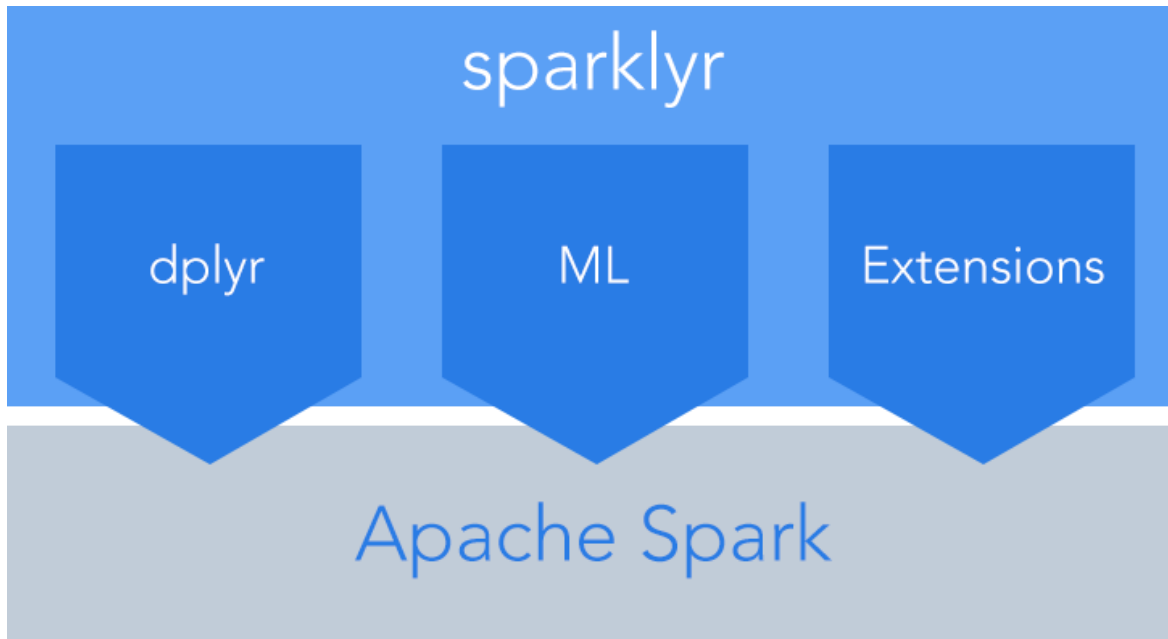
SparkR: R API included with Apache Spark

- An R package provides a light-weight frontend to use Apache Spark from R and allows data scientists to analyze large datasets.
- **SparkDataFrame** is distributed collection of data organized into named columns.
- **SparkR** can create `SparkDataFrames` from local R data frames, csv, json and parquet files.
- Access tables from **Hive** MetaStore.
- Pre-configured on Spark clusters in Azure HDInsight.

Data processing and modeling with SparkR

- Supports functions for processing structured data
 - Selections: select(), filter()
 - Grouping, Aggregations: summarize(), arrange()
 - Running local R functions distributed: spark.lapply()
 - Applying UDFs on each partition/group of a SparkDataFrame: dapply(), dapplyCollect(), gapply(), gapplyCollect()
- Uses **MLlib** to train models and allows model persistence
 - Generalized Linear Model
 - Survival regression
 - Naive Bayes
 - KMeans
 - Logistic Regression
 - Gradient Boosted Tree
 - Random Forest
 - ... others

sparklyr: R interface for Apache Spark



- Easy installation from CRAN

```
install.packages("sparklyr")
```

- Connect to both local instances of Spark and remote Spark clusters


```
library("sparklyr")  
# connect to local instance of Spark  
sc <- spark_connect(master = "local")  
# connect to remote Spark clusters  
sc <- spark_connect(master = "yarn-client")
```

- Loads data into **SparkDataFrame** from: local R data frames, Hive tables, CSV, JSON, and Parquet files.

dplyr and ML in sparklyr

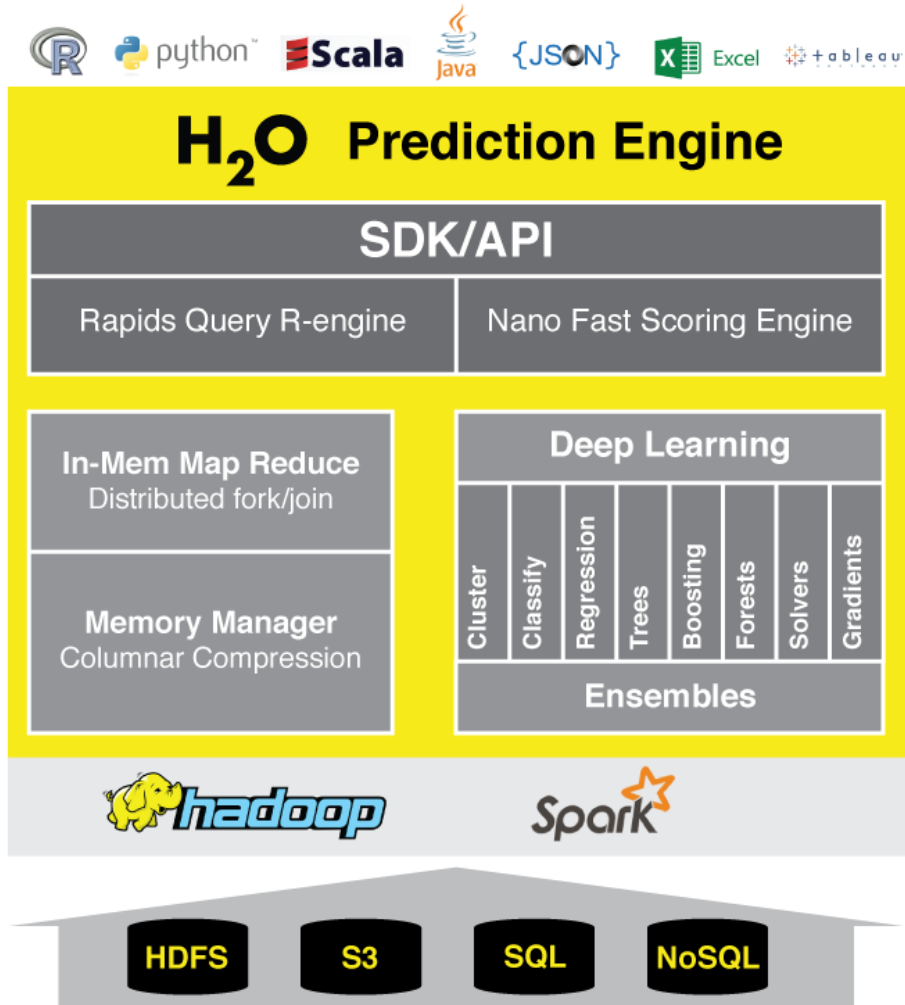
- Provides a complete **dplyr** backend for data manipulation, analysis and visualization

```
# manipulate data with dplyr
library("dplyr")
partitions <- airline_lyr %>%
  mutate(CRSDepTimeHour = floor(CRSDepTime/100)) %>%
  sdf_partition(training = 0.7, test = 0.3, seed = 1099)
```



- Includes 3 family of functions for machine learning pipeline
 - **ml_***: Machine learning algorithms for analyzing data provided by the spark.ml package.
 - K-Means, GLM, LR, Survival Regression, DT, RF, GBT, PCA, Naive-Bayes, Multilayer Perceptron, LDA
 - **ft_***: Feature transformers for manipulating individual features.
 - **sdf_***: Functions for manipulating SparkDataFrames.

h2o: prediction engine in R



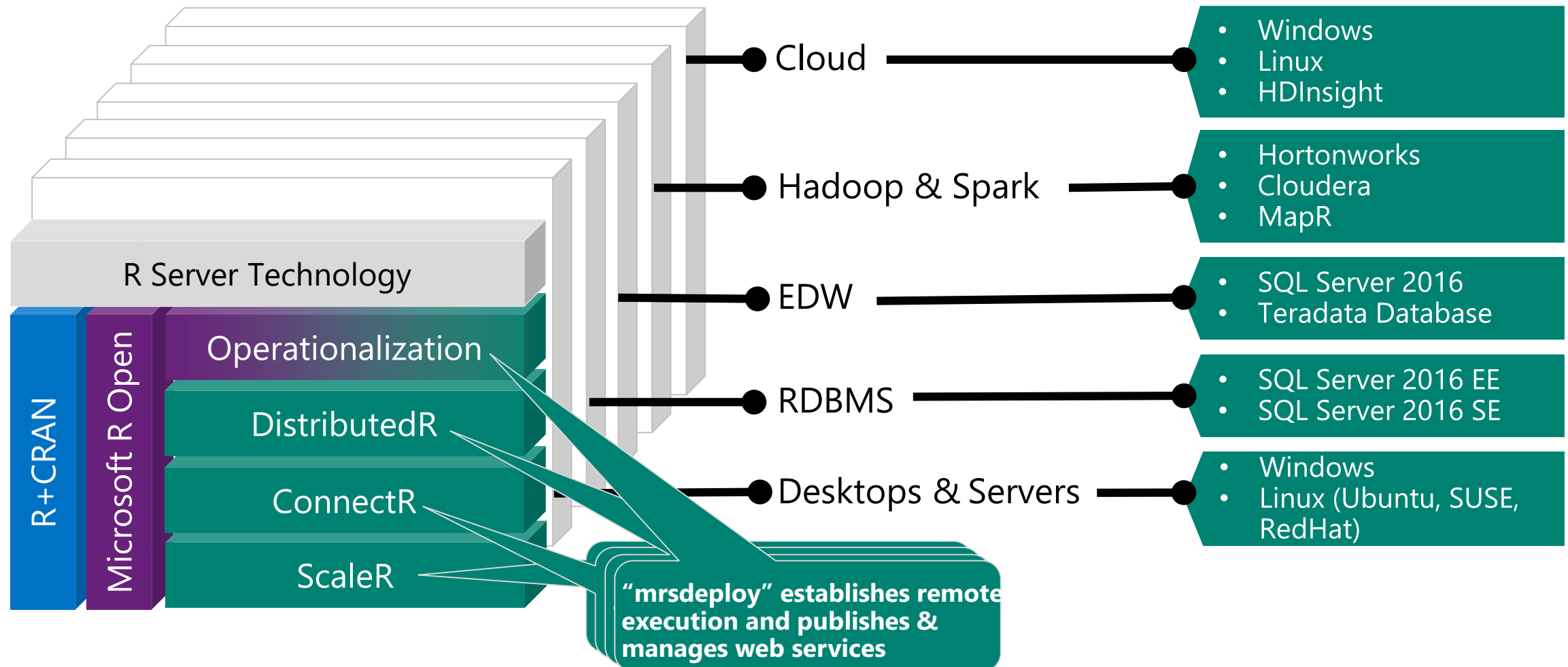
<http://www.h2o.ai/product/>

- Optimized for “in memory” processing of distributed, parallel machine learning algorithms on clusters.
- **Data manipulation and modeling on H2OFrame:** R functions + **h2o** pre-fixed functions.
 - Transformations: `h2o.group_by()`, `h2o.impute()`
 - Statistics: `h2o.summary()`, `h2o.quantile()`, `h2o.mean()`
 - Algorithms: `h2o.glm()`, `h2o.naiveBayes()`, `h2o.deeplearning()`, `h2o.kmeans()`
- **rsparkling** package: h2o on Spark
 - Provides bindings to **h2o**’s machine learning algorithms
 - Simple data conversion: `SparkDataFrame -> H2OFrame`

R Server 9.1: Scale-out R, Enterprise-Class Support

- 100% compatible with open source R
 - Virtually any code/package that works today with R will work in R Server.
- Ability to parallelize any R function
 - Ideal for parameter sweeps, simulation, scoring.
- Wide range of scalable and distributed **rx** pre-fixed functions in **RevoScaleR** package.
 - Transformations: rxDataStep()
 - Statistics: rxSummary(), rxQuantile(), rxChiSquaredTest(), rxCrossTabs()...
 - Algorithms: rxLinMod(), rxLogit(), rxKmeans(), rxBTrees(), rxDForest()...
 - Parallelism: rxSetComputeContext()
- ML featurizers and algorithms in **MicrosoftML** package

RevoScaleR: Portable across multiple platforms



Hands-on Tutorial: Airline Arrival Delay Prediction using R Server and sparklyr

Azure Data Science Virtual Machine (Ubuntu Linux)

Data-science virtual machine

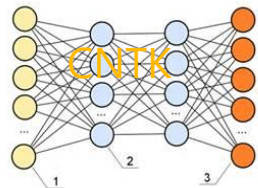


Vowpal Wabbit

xgboost Rattle



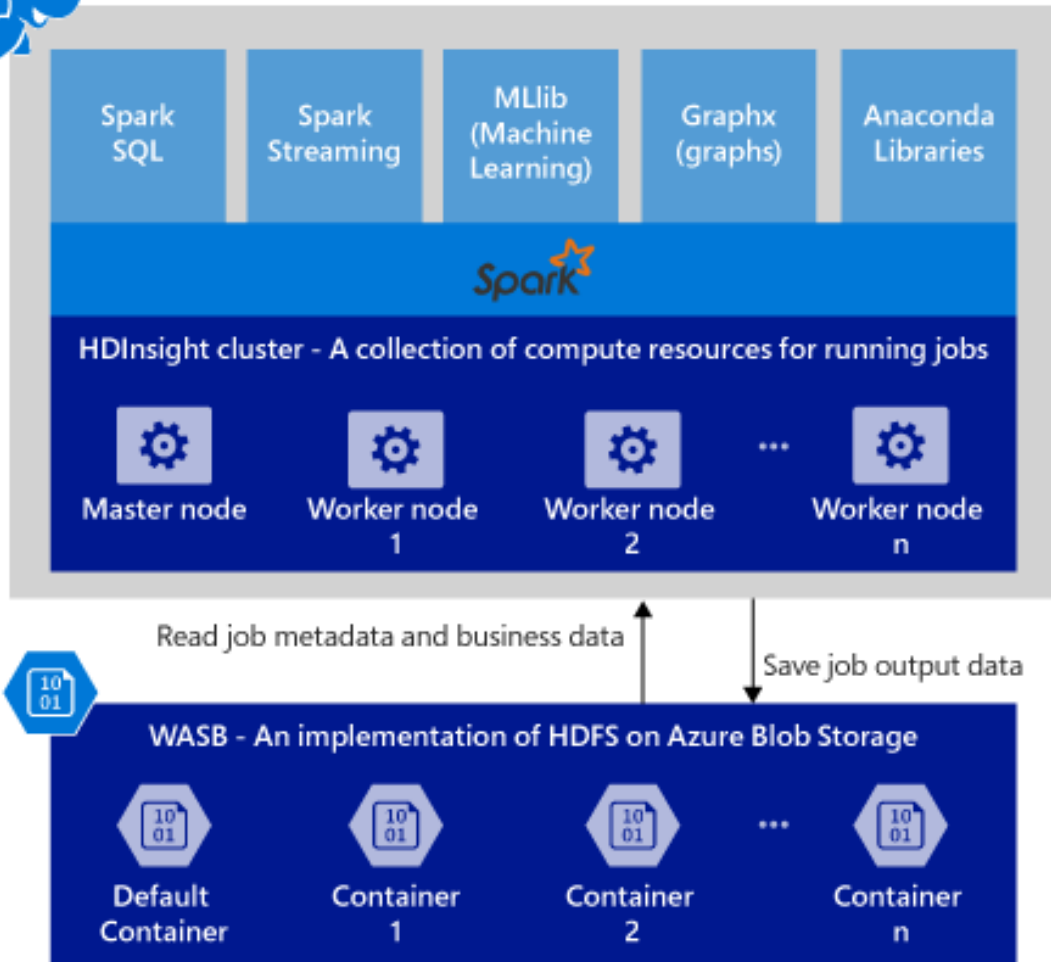
#!/bin/bash



- Spark 2.1.1
- HDFS
- YARN

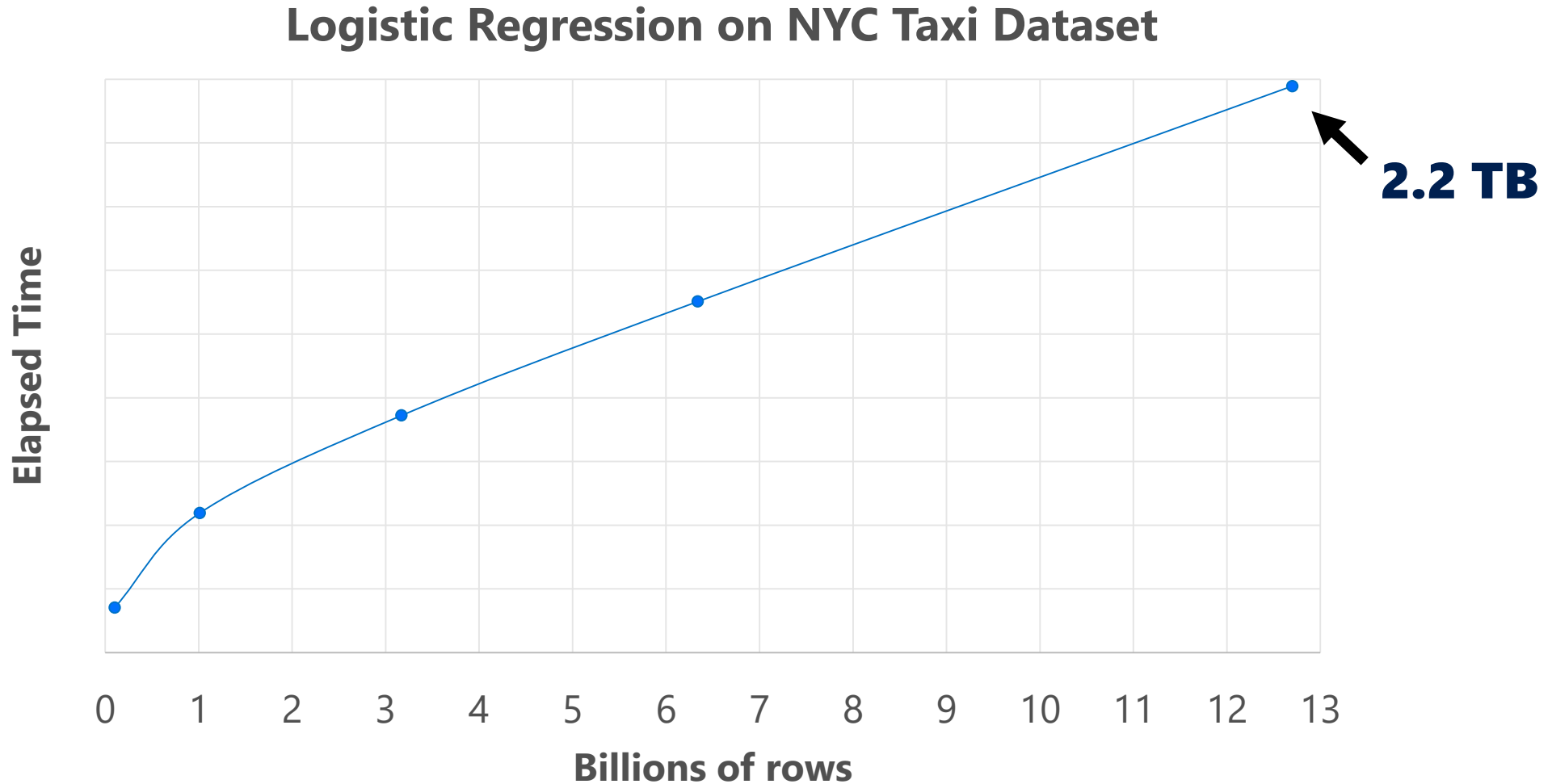
<http://aka.ms/dsvm>

Spark R Server Clusters in Azure HDInsight



- Provisions Azure compute resources with Spark and R Server installed and configured
- Supports multiple versions of Spark and R Server
- Stores data in Azure Blob storage (WASB), Azure Data Lake Store or Local HDFS.

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



Airline Arrival Delay Prediction in Spark

- Clean/Join – sparklyr
- Train/Score/Evaluate – RevoScaleR
- Deploy/Consume – mrsdeploy

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

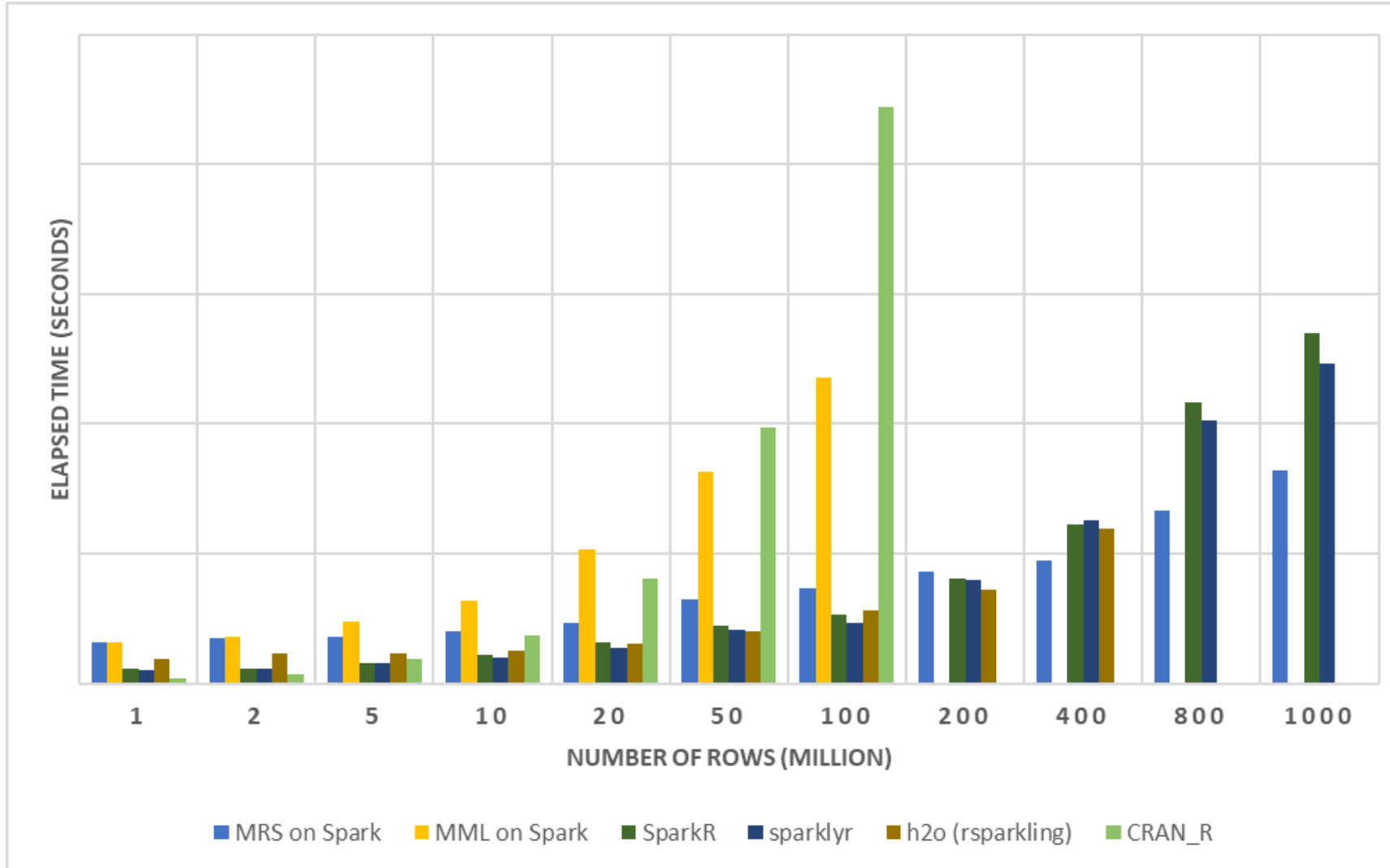
- Hourly land-based weather observations from NOAA
- > 2,000 weather stations
- <http://www.ncdc.noaa.gov/orders/qclcd/>

Comparisons

Base and scalable approaches comparison

Approach	Scalability	Spark	MapReduce	SQL Server	Teradata
Base R	Single machines				
SparkR	Single + Distributed computing	X			
sparklyr	Single + Distributed computing	X			
h2o	Single + Distributed computing	X	X		
RevoScaleR	Single + Distributed computing	X	X	X	X

R Server on Spark - faster and more scalable



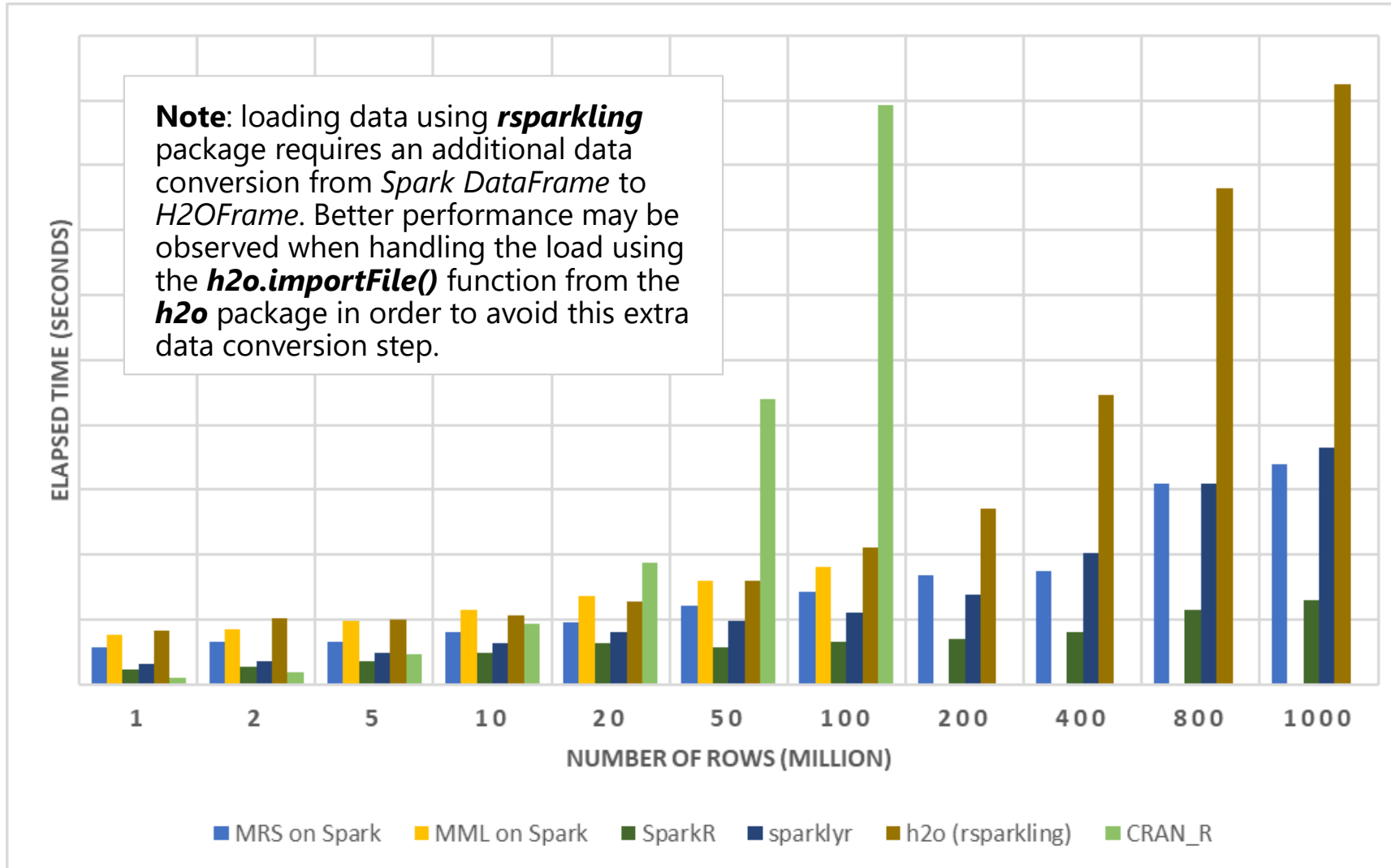
End-to-End Process:

- Load Data from .csv
- Transform Features
- Split Data: Train + Test
- Fit Model: Logistic Regression
- Predict and Write Outputs

Configuration:

- 1 Edge Node: 16 cores, 112GB
- 4 Worker Nodes: 16 cores, 112GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Fastest loading data: SparkR



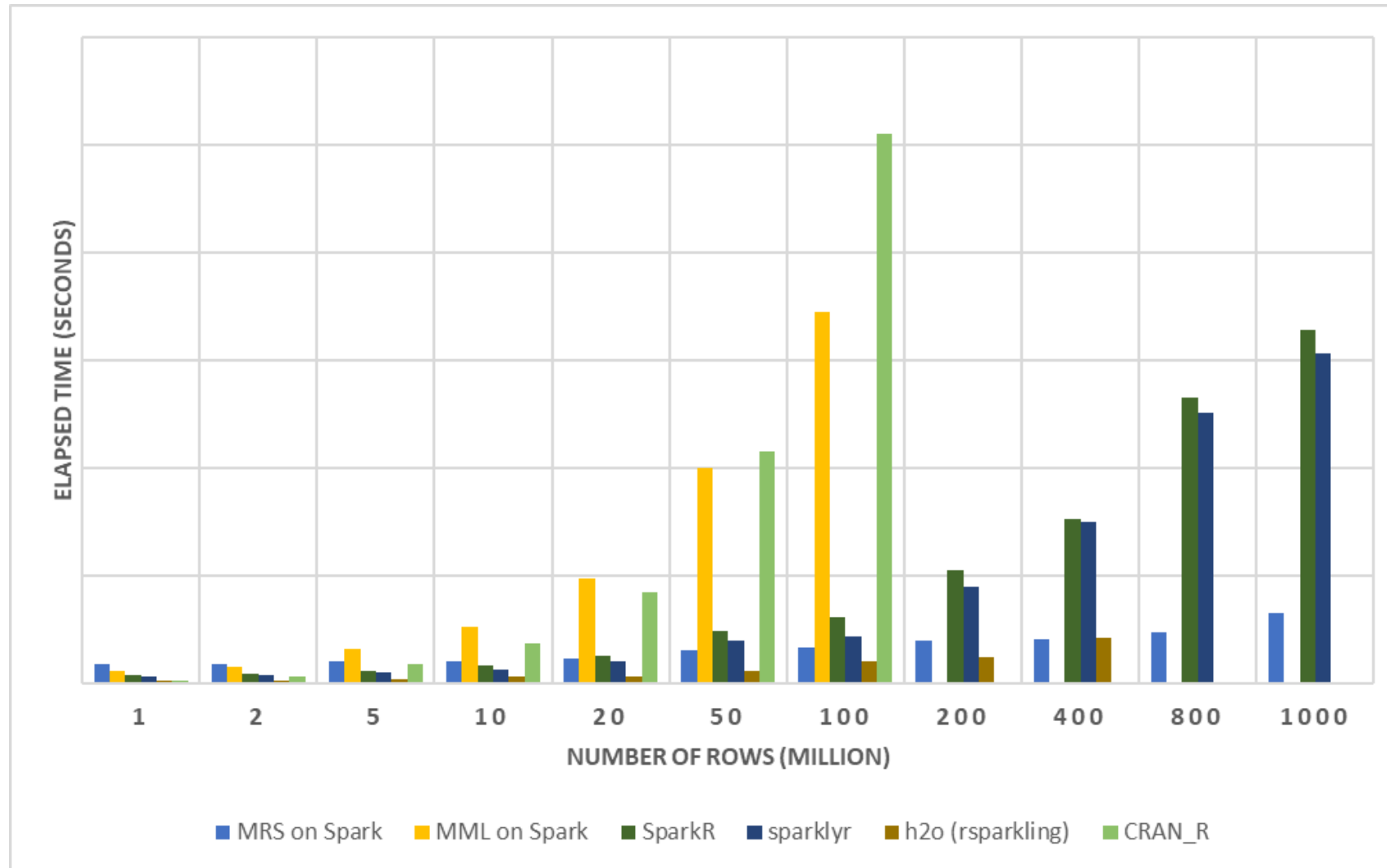
Load Data:

- MRS on Spark: **XDF**
- MML on Spark: **XDF**
- SparkR: **Spark DF**
- sparklyr: **Spark DF**
- h2o: **H2OFrame**
- CRAN R: **DF**

Configuration:

- 1 Edge Node: 16 cores, 112GB
- 4 Worker Nodes: 16 cores, 112GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Fastest fitting big data: R Server



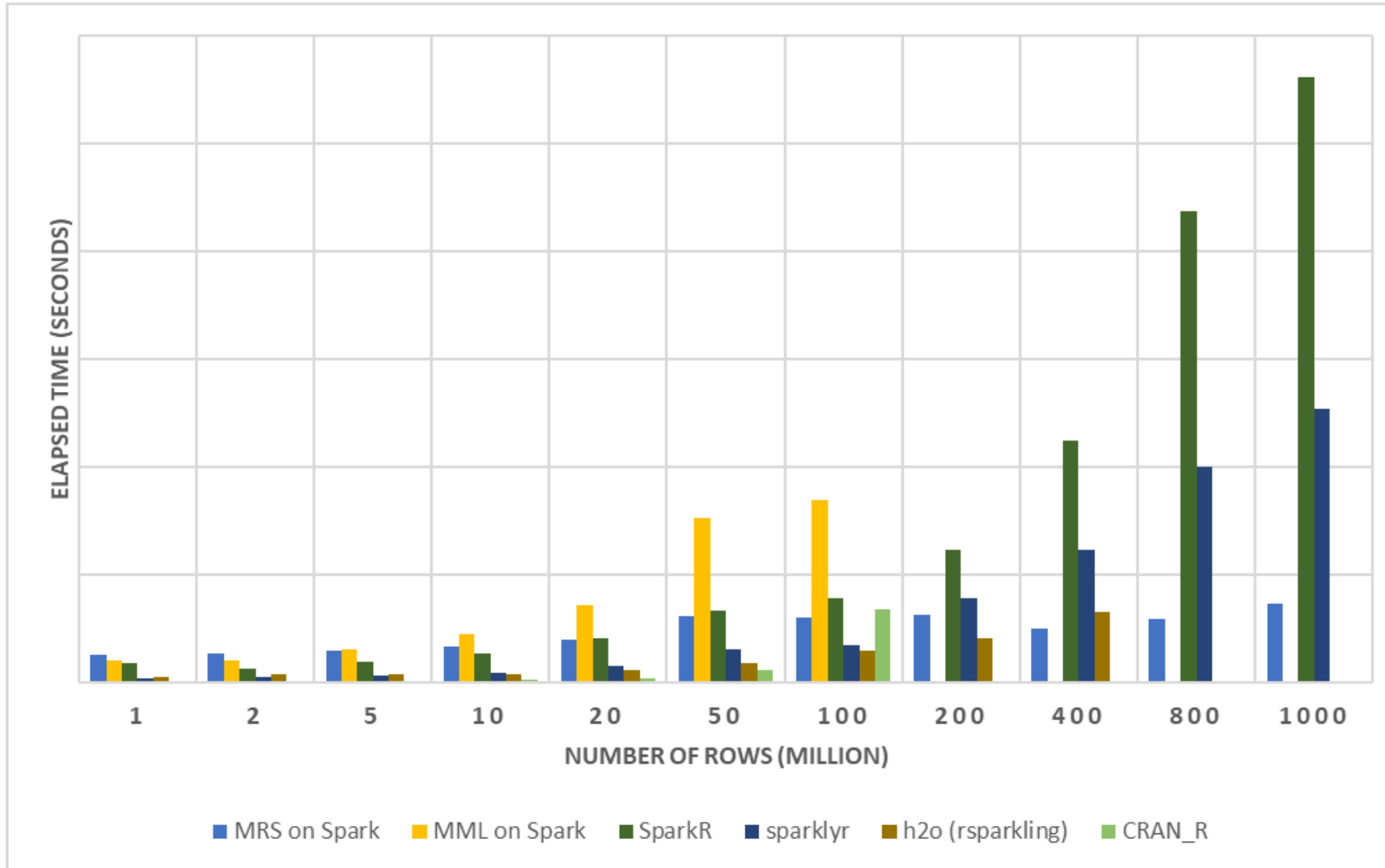
Fit model:

- Logistic regression

Configuration:

- 1 Edge Node: 16 cores, 112GB
- 4 Worker Nodes: 16 cores, 112GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Fastest making predictions: R Server



Predict:

- Outputs predictions into files in HDFS

Configuration:

- 1 Edge Node: 16 cores, 112GB
- 4 Worker Nodes: 16 cores, 112GB
- Dataset: Duplicated Airlines data (.csv)
- Number of columns: 26

Distributed model training and parameter
optimization:

Learning Curves on Big Data

Robert M. Horton, PhD MS
Senior Data Scientist

Learning Curve



Detecting Gibberish

Was the value entered on a web form a real name?

Training Set

real: unique names from babynames database
(71323)

pseudogibberish: simulated keymash
(712625)

random: randomly sampled lower case letters
(712695)

	name	category	is_real
1	UIOPUIO	pseudogibberish	FALSE
2	A A SD	pseudogibberish	FALSE
3	nvsfmxnkw	random	FALSE
4	jbhu	random	FALSE
5	ethjef	random	FALSE
6	opndgsc	random	FALSE
7	smly	random	FALSE
8	ASDASDAS	pseudogibberish	FALSE
9	sbzjo	random	FALSE
10	ellqnc	random	FALSE
11	rerereuy	pseudogibberish	FALSE
12	uswfs	random	FALSE
13	tztxksfl	random	FALSE
14	DSA }	pseudogibberish	FALSE
15	Kaylah	real	TRUE
16	rewrew	pseudogibberish	FALSE
17	uytuyt	pseudogibberish	FALSE
18	Semaiah	real	TRUE
19	iomnbm	random	FALSE
20	nqbthsh	random	FALSE
21	wwxnx	random	FALSE
22	bvgl	random	FALSE
23	Annabelle	real	TRUE

Text Featurization

```
fit <- rxFastLinear(  
  is_real ~ chagrams,  
  local_xdf,  
  type="binary", normalize="no",  
  l1Weight=0, l2Weight=1e-8,  
  mlTransforms = featurizeText(  
    vars = c(chagrams="name"),  
    case='lower',  
    keepNumbers=FALSE,  
    keepDiacritics=FALSE,  
    keepPunctuations=FALSE,  
    charFeatureExtractor=ngramCount(  
      ngramLength=3,  
      weighting="tf",  
      maxNumTerms=1e8  
    ),  
    wordFeatureExtractor=NULL  
  )  
)
```

Model Coefficients

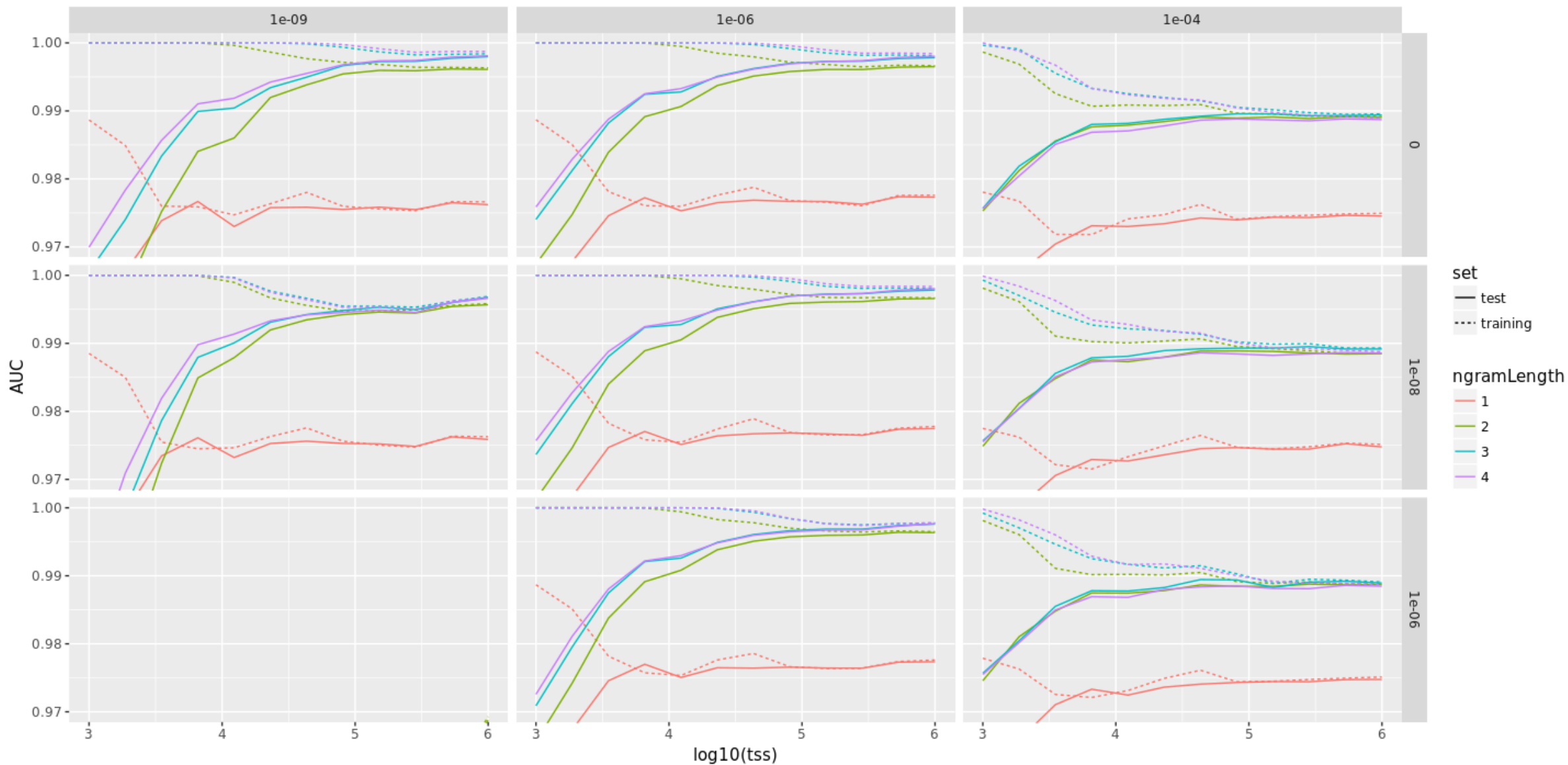
	coefficient
(Bias)	-10.71464
r e r	-17.37628
q	-15.18943
<P> a <P>	-14.60359
q u	13.91951
n a n	-13.82819
s a s	-13.79482
i i	-13.73330
f f	13.53813
<P> l u	13.33264
	...

Tuning Regularization Penalties

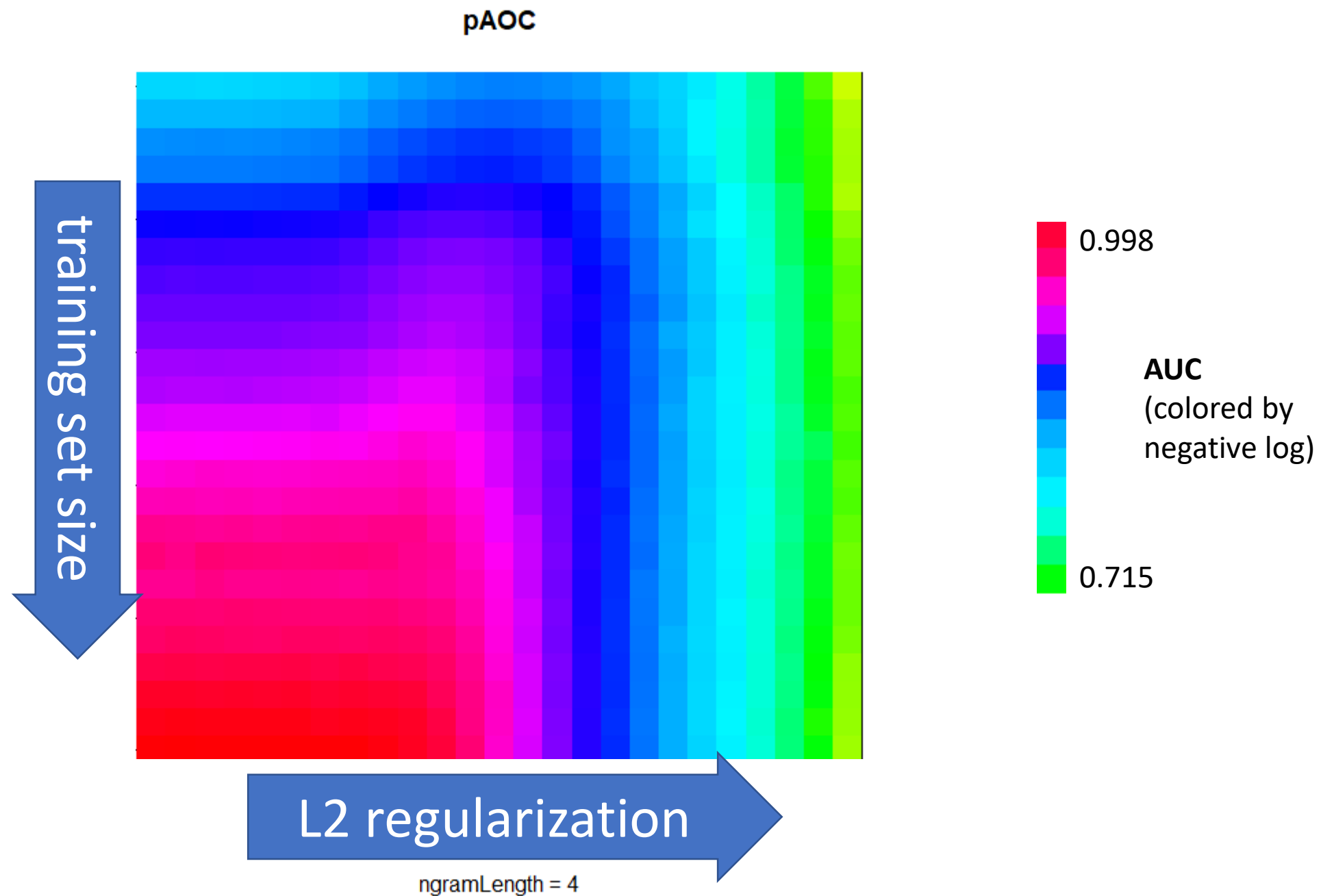
L2

faceting by regularization weights

L1



Error Surface



-Grouped Time Series
Forecasting

-Sentiment Analysis with
Pretrained Deep Learning

Deep Learning in R

KDD 2017 – Using R for Scalable Data Science tutorial

Package	Network Architecture	CPU / GPU support	Tensor backend	Reference
tensorflow TensorFlow for R	FFNN, CNN, RNN	multi-CPU, multi-GPU	Tensorflow	https://tensorflow.rstudio.com/
mxnet MXNet R API	FFNN, CNN, RNN	multi-CPU, multi-GPU	MXNet	http://mxnet.io/api/r/index.html
keras R interface for Keras	FFNN, CNN, RNN	multi-CPU, multi-GPU	Tensorflow, CNTK, Theano	https://rstudio.github.io/keras/
h2o - h2o.deepwater Deep Water R API	FFNN, CNN, RNN	multi-CPU, multi-GPU	Tensorflow, Caffe, MXNet	https://www.h2o.ai/deep-water/
h2o - h2o.deeplearning	FFNN	CPU	NA	http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/deep-learning.html
MicrosoftML Microsoft R Server	FFNN, CNN	CPU, GPU	NA	1) https://docs.microsoft.com/en-us/r-server/r-reference/microsoftml/rxneuralnet 2) https://blogs.msdn.microsoft.com/microsoftserververtigerteam/2017/03/10/get-started-with-microsoftmls-rxneuralnet-with-gpu-acceleration/

Other Options for Scaling R Scripts

The bigmemory project

- **bigmemory** supports large matrix-like objects in R
- Combines memory and file-backed data structures: analyze numerical data larger than RAM



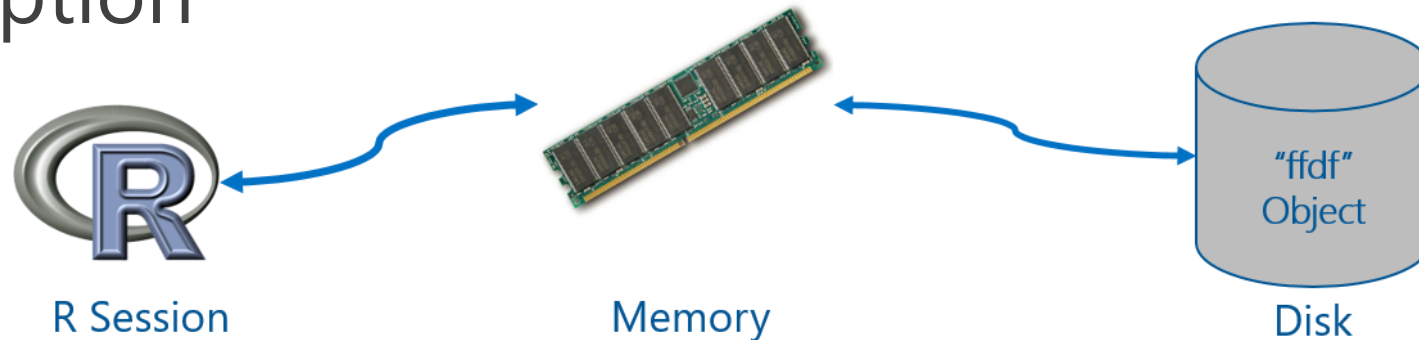
- The data structures may be allocated to shared memory

sister packages and related work

- **biganalytics**: provides exploratory data analysis functionality on `big.matrix`
- **bigtabulate**: adds `table`-, `tapply`-, and `split`-like behavior for `big.matrix`
- **bigalgebra**: performs linear algebra calculations on `big.matrix` and R matrix
- **synchronicity**: supports synchronization and may eventually support interprocess communication (ipc) and message passing
- **biglm**: provides linear and generalized linear models on `big.matrix`
- **Rdsm**: enables shared-memory parallelism with `big.matrix`

ff package

- Provides data structures that are stored on Disk, but behave as if they were in RAM
- Maps only a section in main memory for effective consumption



- Accepts numeric and characters as input data

ff related packages

- **ffbase**: adds basic statistical functionality to ff
 - Coercions: `as.character.ff()`, `as.Date_ff_vector()`, `as.ffdf.ffdf()`, `as.ram.ffdf()`
 - Selections: `subset.ffdf()`, `ffwhich()`, `transform.ffdf()`, `within.ffdf()`, `with.ffdf()`
 - Aggregations: `quantile.ff()`, `hist.ff()`, `sum.ff()`, `mean.ff()`, `range.ff()`, `tabulate.ff()`
 - Algorithms: `bigglm.ffdf()`
- **biglars**: provides least-angle regression, lasso and stepwise regression on ff

Parallel programming with `foreach`

- Provides a function `foreach` and two operators `%do%` and `%dopar%` that support parallel execution
- `%dopar%` operator relies on a pre-registered parallel backend – `doParallel()`, `doSNOW()`, `doAzureParallel()`, etc.

```
> library("doParallel")
> cl <- makeCluster(getOption("cl.cores", 4))
> registerDoParallel(cl)

> rf <- foreach(ntree=rep(250, 4), .combine=combine, .packages='randomForest') %dopar%
+   randomForest(x, y, ntree=ntree)
> rf

Call:
  randomForest(x = x, y = y, ntree = ntree)
              Type of random forest: classification
                Number of trees: 1000
No. of variables tried at each split: 2
```


Q & A

CONTACT INFORMATION

Robert Horton (rhorton@microsoft.com)

Mario Inchiosa (marioinc@yahoo.com)

Vanja Paunic (vanja.paunic@microsoft.com)

Hang Zhang (hangzh@microsoft.com)

THANK YOU