

Using R in Scalable Compute Contexts – R Server with Spark on HDInsight

Ali Zaidi

Microsoft Machine Learning and Data Science Team

learnanalytics.microsoft.com



R – What is it?

Open Source
"lingua franca"



Analytics, Computing,
Modeling

Global Community



Millions of users

Ecosystem

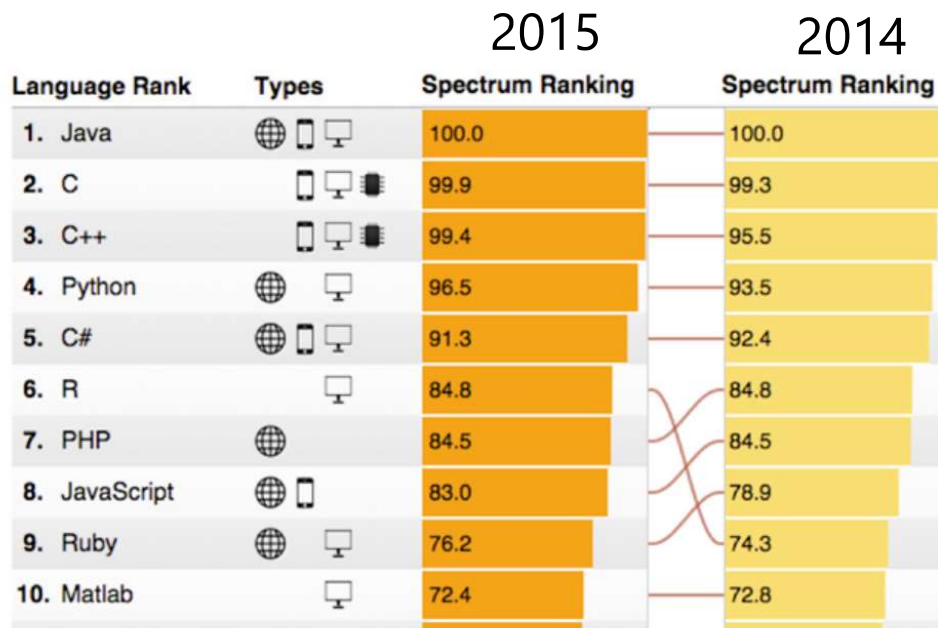


7000+ Algorithms, Test
Data & Evaluations

Common R use cases

Vertical	Sales & Marketing	Finance & Risk	Customer & Channel	Operations & Workforce
Retail	Demand Forecasting Loyalty Programs Cross-sell & Upsell Customer Acquisition	Fraud Detection Pricing Strategy	Personalization Lifetime Customer Value Product Segmentation	Store Location Demographics Supply Chain Management Inventory Management
Financial Services	Customer Churn Loyalty Programs Cross-sell & Upsell Customer Acquisition	Fraud Detection Risk& Compliance Loan Defaults	Personalization Lifetime Customer Value	Call Center Optimization Pay for Performance
Healthcare	Marketing Mix Optimization Patient Acquisition	Fraud Detection Bill Collection	Population Health Patient Demographics	Operational Efficiency Pay for Performance
Manufacturing	Demand Forecasting Marketing mix Optimization	Pricing Strategy Perf Risk Management	Supply Chain Optimization Personalization	Remote Monitoring Predictive Maintenance Asset Management

R has some challenges



IEEE Spectrum July 2015

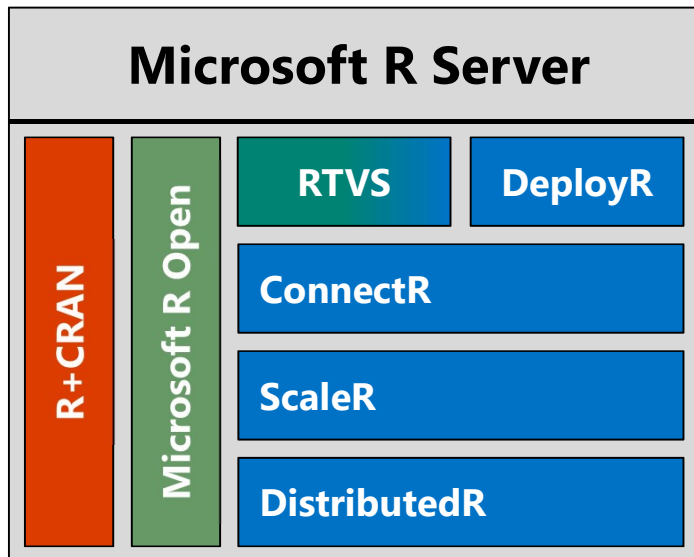
■ Data Flows Overwhelm Open Source R

- In-Memory Operation
- Lack of Implicit Parallelism
- Expensive Data Movement & Duplication

■ Not enterprise ready

- Inadequacy of Community Support
- Lack of Guaranteed Support Timeliness
- No SLAs or Support models

R Server: scale-out R, Enterprise Class!



- Enterprise Scale & Performance
 - Scales from workstations to large clusters
 - Can process terabytes of data faster
 - Growing portfolio of Parallelized algorithms
- Enterprise Class Support
- Secure R Deployment/Operationalization
- Write Once Deploy Anywhere for multiple platforms
 - Cloud: HDInsight and Marketplace
 - On Prem: SQL Server, Hadoop (HortonWorks, Cloudera, MapR) and TeraData DB
- IDE for data scientists and developers (R Tools for Visual Studio)

R Server: scale-out R, Enterprise Class!

- 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 or multiple runs

Parallelized & Distributed Algorithms



Data Step

- 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 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



Simulation

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



Cluster Analysis

- K-Means



Machine Learning

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



Custom Parallelization

- rxDataStep
- rxExec
- PEMA-R API

Hadoop and HDInsight



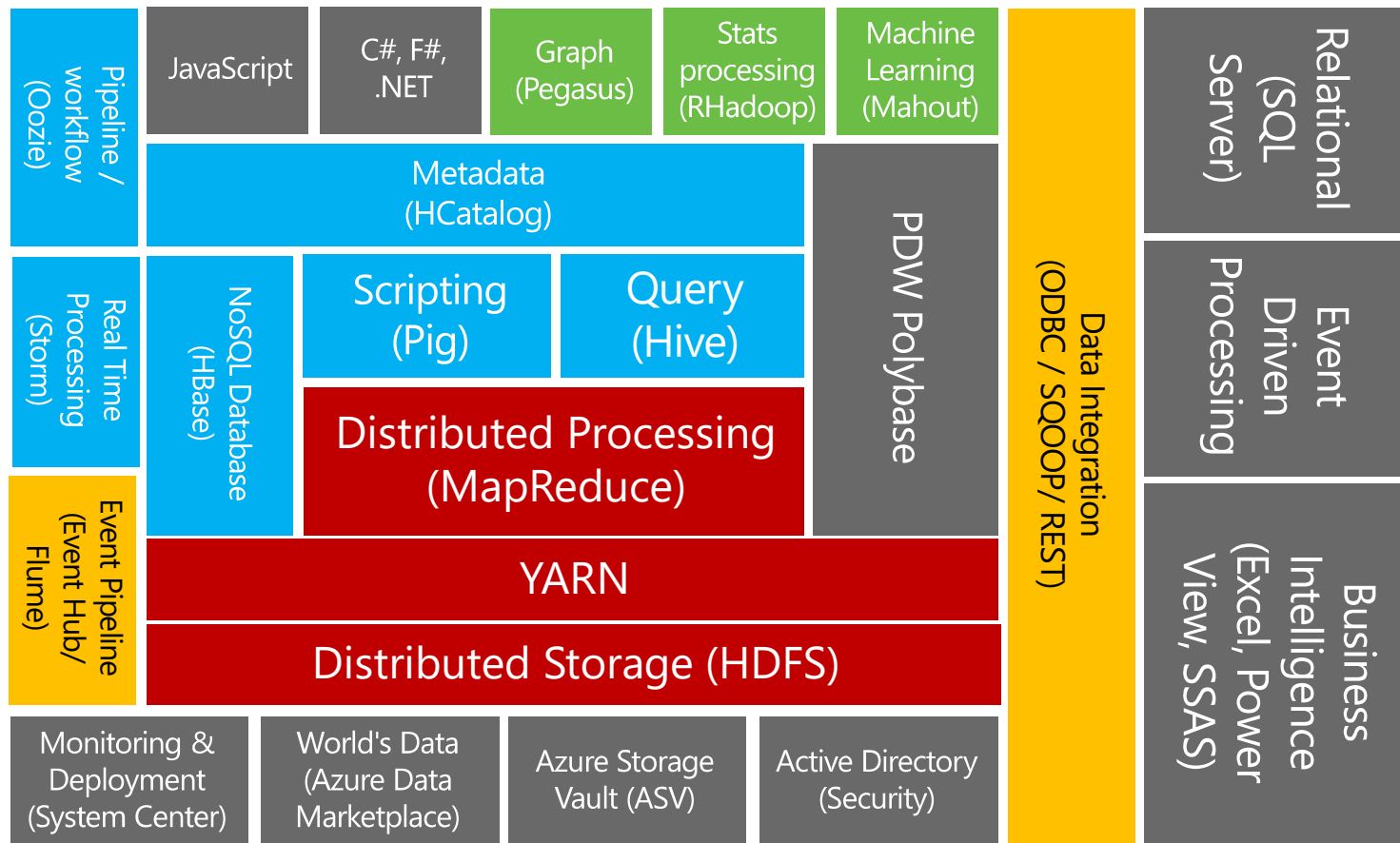
Using the Hadoop Ecosystem to
process and query data

Hadoop



- An ecosystem of components for distributed data processing and analysis
- Core components: MapReduce, HDFS, YARN
- Data is processed in the Hadoop Distributed File System (HDFS)
- Resource Management is performed by YARN
- Many other related projects

HDInsight and the Hadoop ecosystem



Core
Hadoop

Data
processing

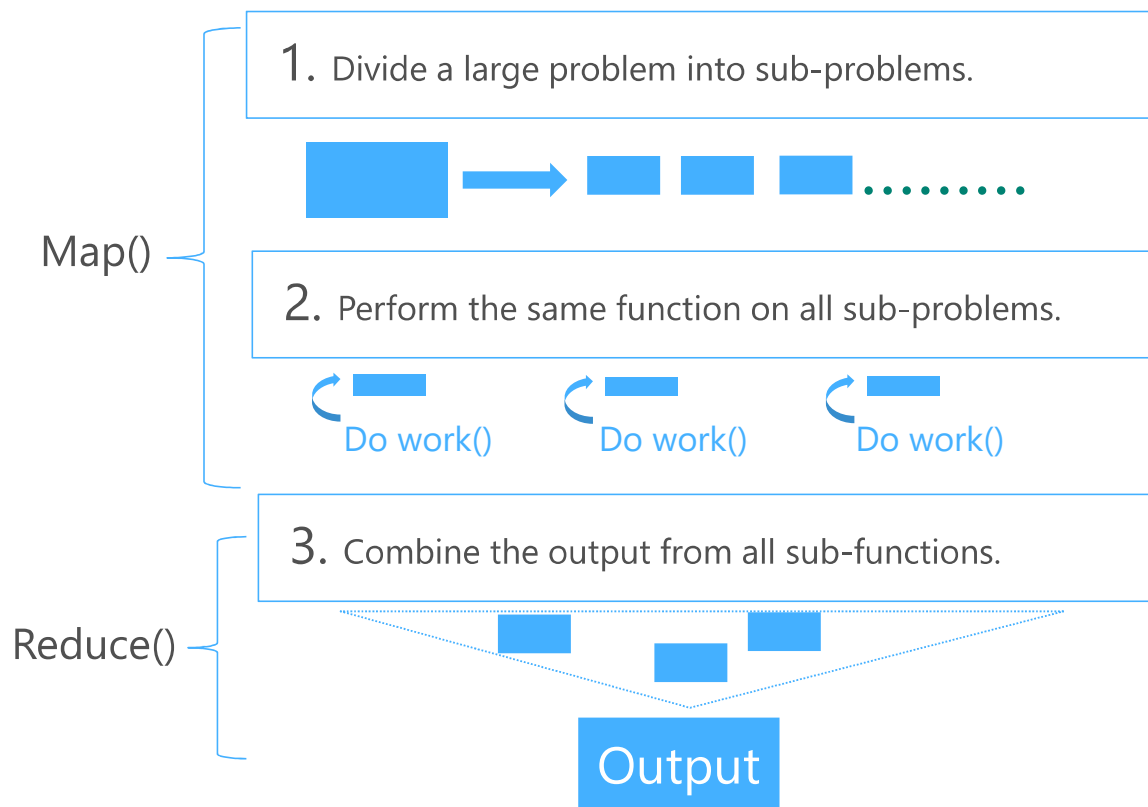
Microsoft
integration

Data
Movement

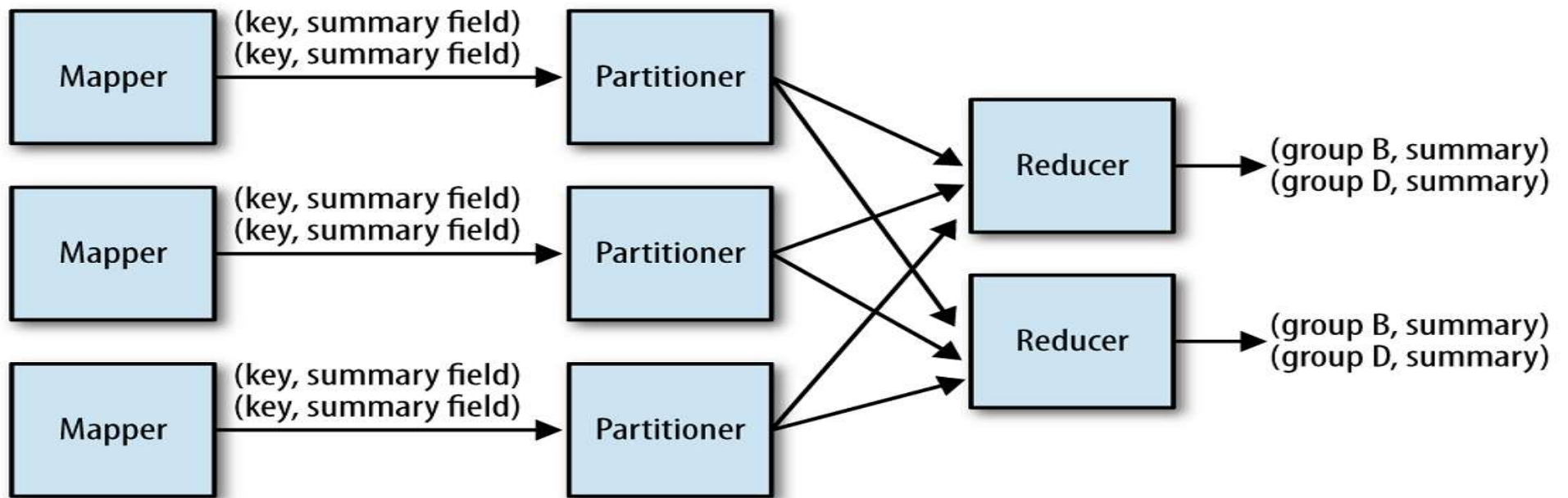
Packages

Hadoop MapReduce

- Programming framework (library and runtime) for analyzing datasets stored in HDFS
- Composed of user-supplied Map and Reduce functions:
 - Map() - subdivide and conquer
 - Reduce() - combine and reduce cardinality



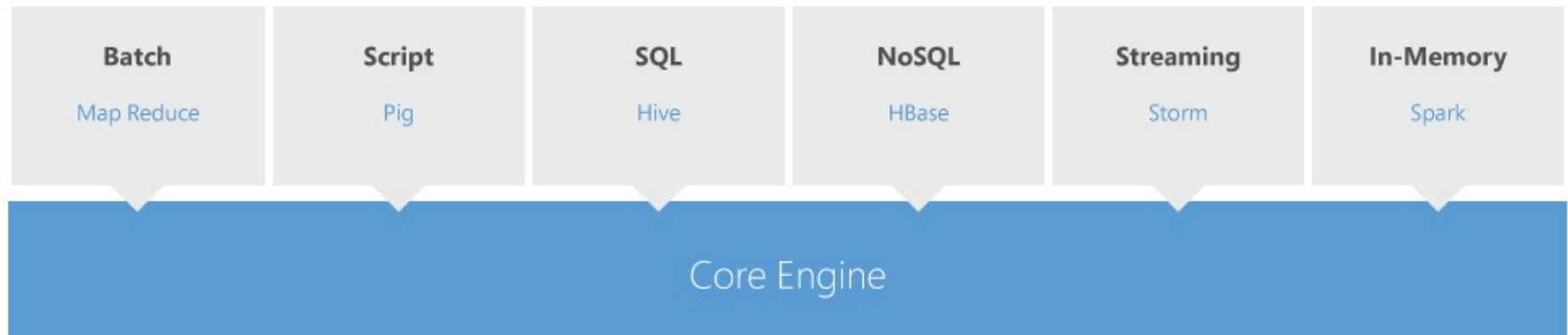
Another view of MapReduce



HDInsight



- 3 Modes: VM, Service, On-Demand
- Azure Storage or Azure Data Lake provides the HDFS layer
- Azure SQL Database stores metadata



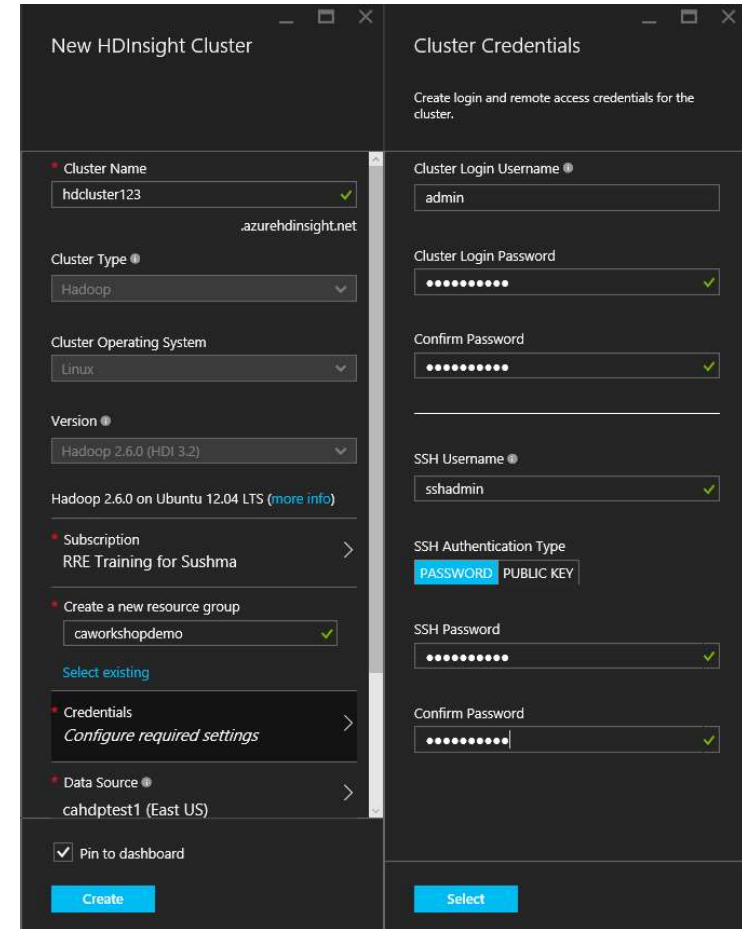
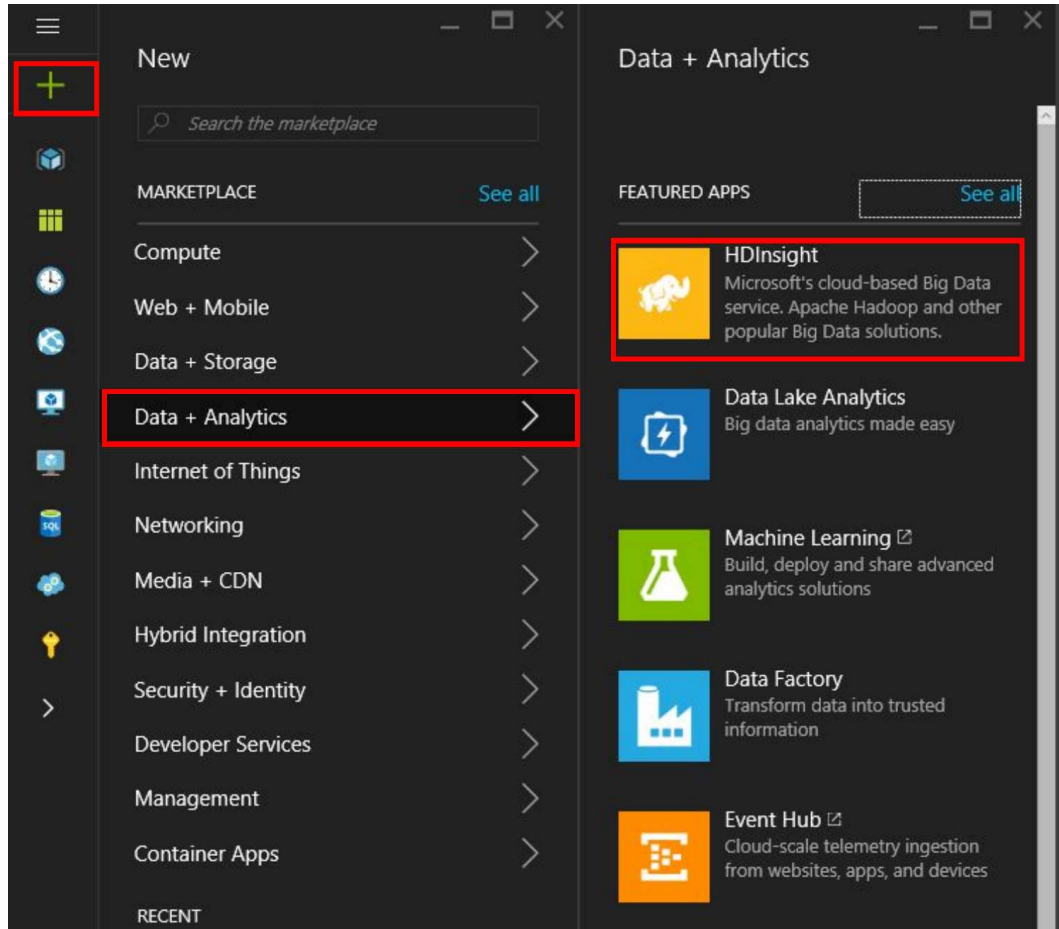
Deploying HDInsight Clusters

- Cluster Type: Hadoop, Spark, HBase and Storm.
 - Hadoop clusters: for query and analysis workloads
 - HBase clusters: for NoSQL workloads
 - Spark clusters: for in-memory processing, interactive queries, stream, and machine learning workloads
- Operating System: Windows or Linux
- Can be deployed from Azure portal, Azure Command Line Interface (CLI), or Azure PowerShell and Visual Studio
- A UI dashboard is provided to the cluster through Ambari.
- Remote Access through SSH, REST API, ODBC, JDBC.
 - Remote Desktop (RDP) access for Windows clusters

Components and Customization

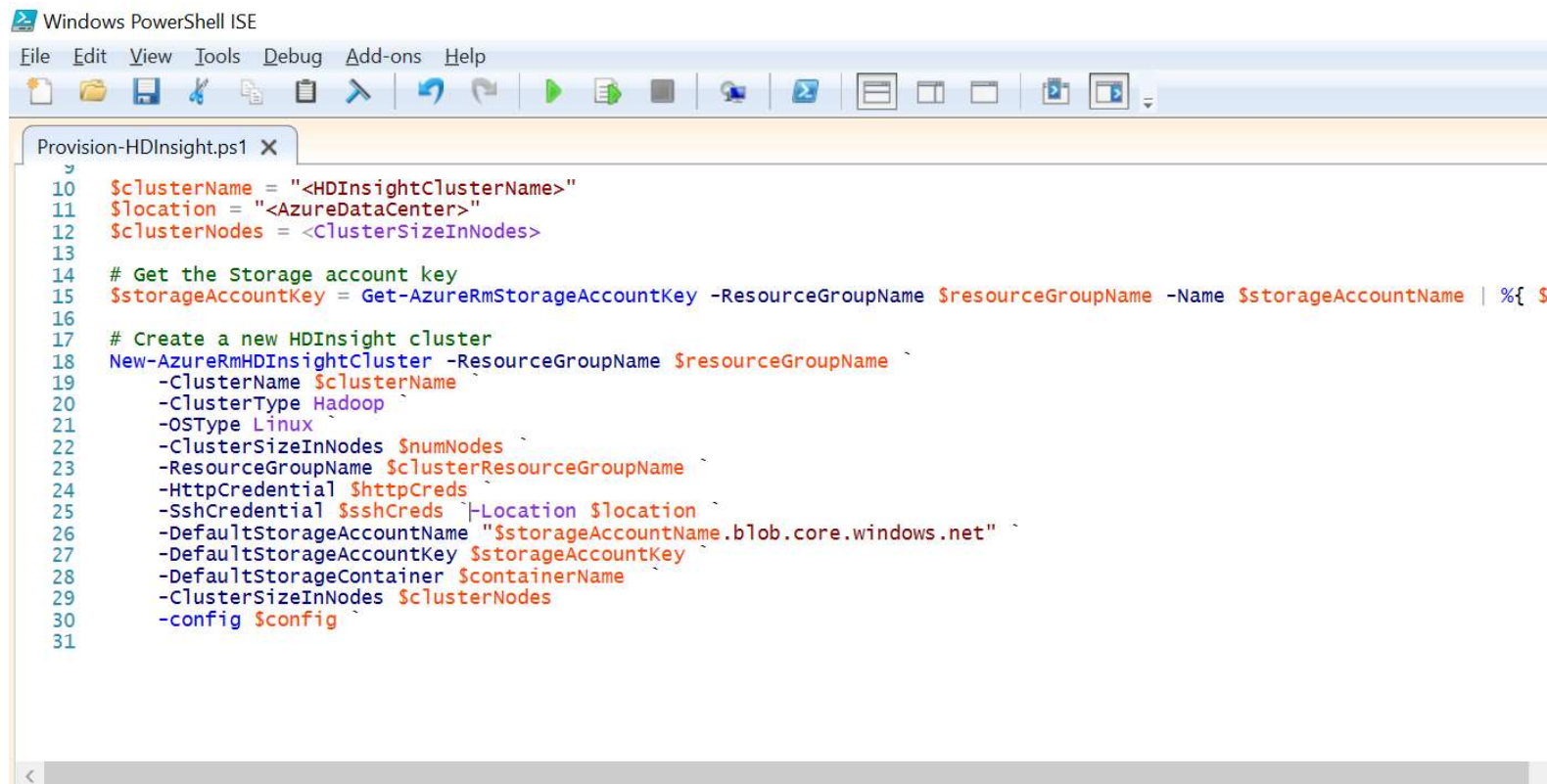
- Script Actions can be run during cluster provisioning to install additional components.
- You can use the sample script actions during deployment to install:
 - Solr
 - R
 - Giraph
- You can change the configurations of a cluster using Bootstrap with:
 - Azure PowerShell
 - .NET SDK
 - ARM Templates

Provisioning with Azure Portal



Provisioning Clusters Using PowerShell

- You can provision with Azure PowerShell



```
Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Provision-HDInsight.ps1 X
10 $clusterName = "<HDInsightClusterName>"
11 $location = "<AzureDataCenter>"
12 $clusterNodes = <ClusterSizeInNodes>
13
14 # Get the Storage account key
15 $storageAccountKey = Get-AzureRmStorageAccountKey -ResourceGroupName $resourceGroupName -Name $storageAccountName | %{ $
16
17 # Create a new HDInsight cluster
18 New-AzureRmHDInsightCluster -ResourceGroupName $resourceGroupName `
19     -ClusterName $clusterName `
20     -ClusterType Hadoop `
21     -OSType Linux `
22     -ClusterSizeInNodes $numNodes `
23     -ResourceGroupName $clusterResourceGroupName `
24     -HttpCredential $httpCreds `
25     -SshCredential $sshCreds -Location $location `
26     -DefaultStorageAccountName "$storageAccountName.blob.core.windows.net" `
27     -DefaultStorageAccountKey $storageAccountKey `
28     -DefaultStorageContainer $containerName `
29     -ClusterSizeInNodes $clusterNodes `
30     -config $config
31
```

Cluster Dashboard Access - Ambari

hdcluster123
HDInsight Cluster

Settings Dashboard Secure Shell Scale Cluster Delete Move

Essentials ^

Resource group: [caworkshopdemo](#) URL: [hdcluster123.azurehdinsig](#)




Status: **Running** Cluster Type: **Hadoop on Linux**

Location: **East US** Head Node, Worker Nodes: **D3 (x2), D3 (x1)**

Subscription name: [RRE Training for Sushma](#) Learn more: [Documentation](#)

Subscription id: [5be49961-ea44-42ec-8021-b728be90d58c](#) Getting Started: [Quickstart](#)

Quick Links

 **Cluster Dashboard**  **Ambari Views** 

Ambari hdcluster123 0 ops 0 alerts

Dashboard Services Hosts Alerts Admin admin

Metrics Heatmaps Config History

Metric Actions

HDFS Disk Usage
8%

DataNodes Live
1/1

HDFS Links
Active NameNode
Standby NameNode
1 DataNodes

Memory Usage
9.3 GB

Network Usage
488.2 KB

CPU Usage
50%

Cluster Load
5

NameNode Heap
20%

NameNode RPC
0.18 ms

NameNode CPU WIO
7.3%

NameNode Uptime
37.3 min

ResourceManager Heap
2%

ResourceManager Uptime
39.4 min

NodeManagers Live
1/1

YARN Memory
0%

YARN Links
ResourceManager
1 NodeManagers

What Is Spark?



Spark is

- an open-source project that performs rapid calculations on in-memory datasets
- Started at the University of California Berkeley AMPLab
- Open Source [Apache hosted & licensed]
- Developed by a large community of developers (currently the most active Apache project)

R Server in Hadoop



- Traditionally R Server supported Hadoop MR as a Hadoop compute context
- This meant R Server functions ran on Hadoop as a MapReduce Job under YARN
- Spark posits itself as a better execution engine for iterative jobs
- R Server on MR is slowed by startup overhead and file-based communications

R Server on Spark



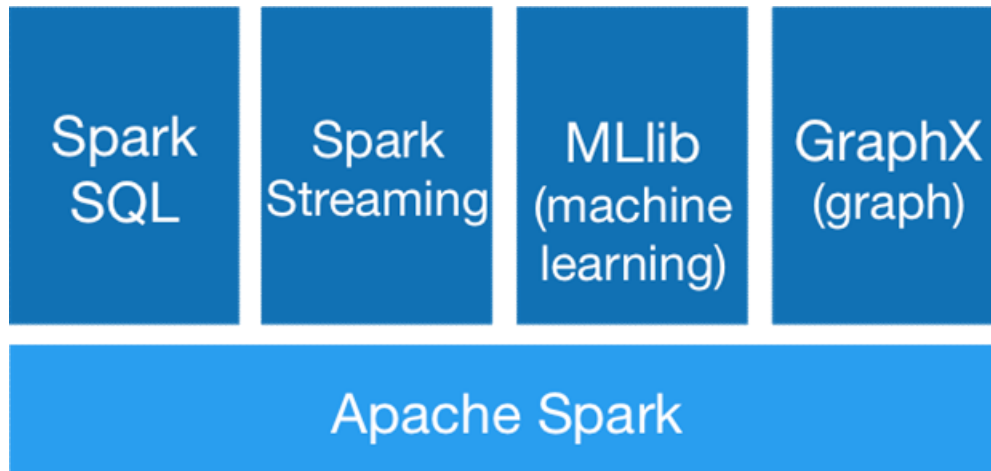
- Spark permits data to be retained in memory across iterations of ML pipelines through Resilient Distributed Datasets (RDD) and Spark Dataframes
- R Server on Spark is built on Spark's high performance distributed backend w/o the file-based overhead of MR
- How it works:
 - Use RDD's to load and retain data across iterations
 - Cache data in RDD between iterations
 - Master and worker nodes communicate with Scala API for communication
 - Saves temporary files in tmpfs (in-memory file system on Linux)
 - Supports persistent mode in Spark Context to reduce overhead of launching the process.

How to Write Spark Jobs – Spark Abstractions



- Spark supports popular language APIs such as: Scala, Python, R, Java
 - Spark SQL: Seamlessly mix SQL queries with Spark programs
 - Datasets – RDD
 - RDD (Resilient Distributed Data) is the basis for what Spark enables
 - Resilient – the models can be recreated on the fly from known state
 - Distributed – the dataset is often partitioned across multiple nodes for increased scalability and parallelism
- Data Scientists prefer to use DataFrames tabular derivative of RDDs

Spark Core



- Elegant Developer APIs: Data Frames/SQL, Machine Learning, Graph algorithms and streaming
 - High Level API in Scala, Python, Java and R
 - Promotes Code Reuse – APIs and data types are similar for batch and streaming jobs
-
- Workloads: MLlib and GraphX for Data Scientists, Spark SQL for Data Analysts, Spark Streaming for micro batch problems

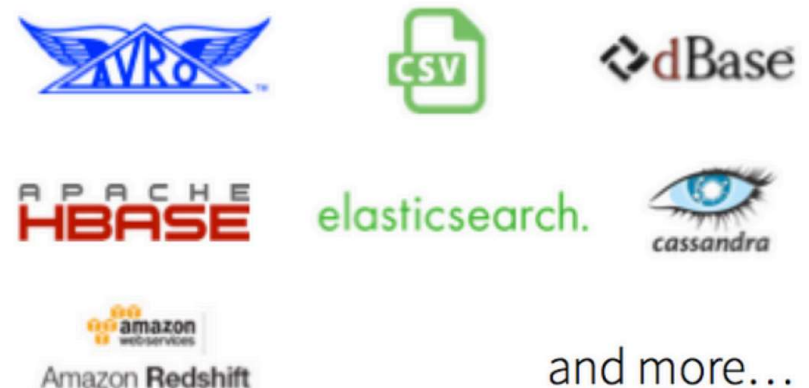
Spark Supported Data Types



Built-In



External



and more...

Spark v. MapReduce



- How??
 - Caching data to memory can avoid extra reads to disk
 - Scheduling of tasks from 15 – 20 s to 15 – 20 ms
 - Resources are dedicated to the entire life of the application
 - Can link multiple maps and reduces together without having to rewrite intermediate data to HDFS
 - Every reduce doesn't require a map

Spark Main Considerations



- Spark's Model of Parallel Computing
 - RDDs form the basis of data abstraction in Spark
 - RDDs are immutable distributed collections of objects
 - Spark, as a Scala derivative, is a functional programming paradigm
 - Functional abstractions are transformers and actions:

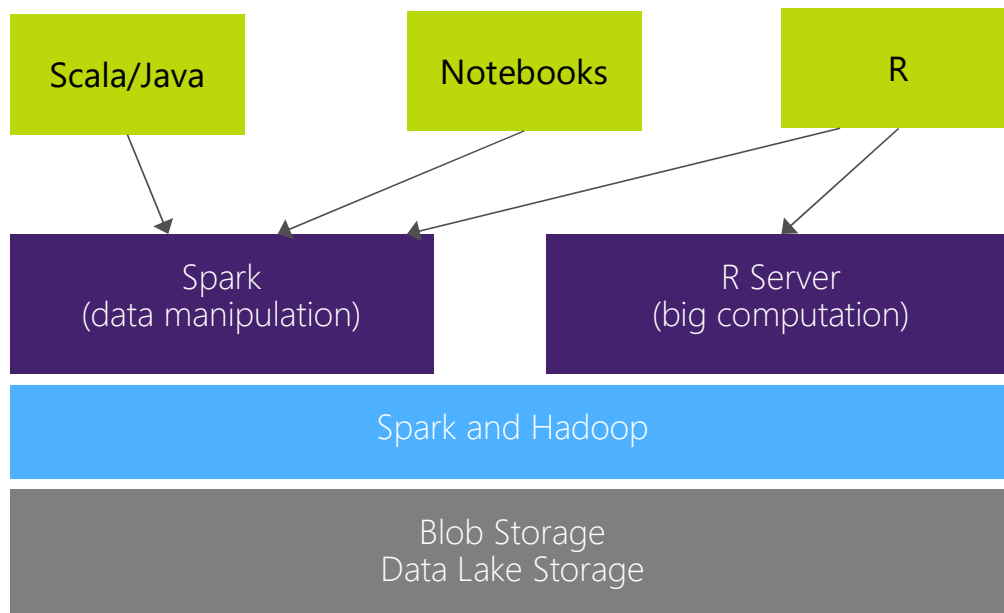
←

Applied to an RDD lazily,
creating a promise to
generate a new RDD

→

Triggers
scheduler,
retrieves results

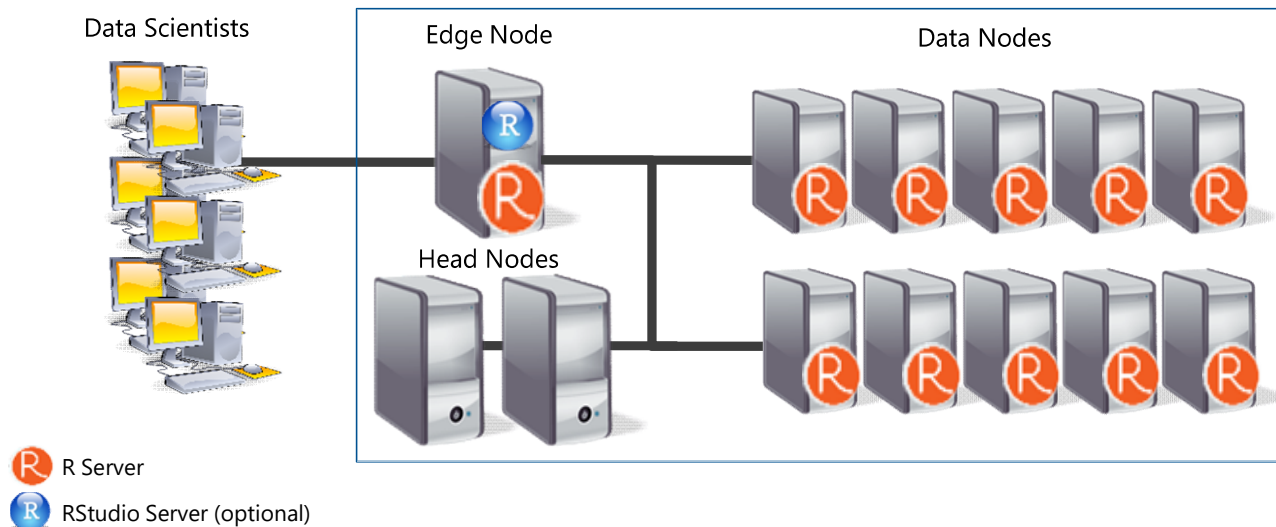
HDInsight + R Server: Managed Hadoop for advanced analytics



Hadoop: lingua franca for BigData

- Spark (Standard)
 - Integrated notebooks experience
 - Upgraded to latest Version 1.6
- R Server (Premium)
 - Leverage R skills with massively scalable algorithms and statistical functions
 - Reuse existing R functions over multiple machines

R Server HDInsight Architecture



R Server Distributed Processing:

Master R process on Edge node



Apache YARN and Spark



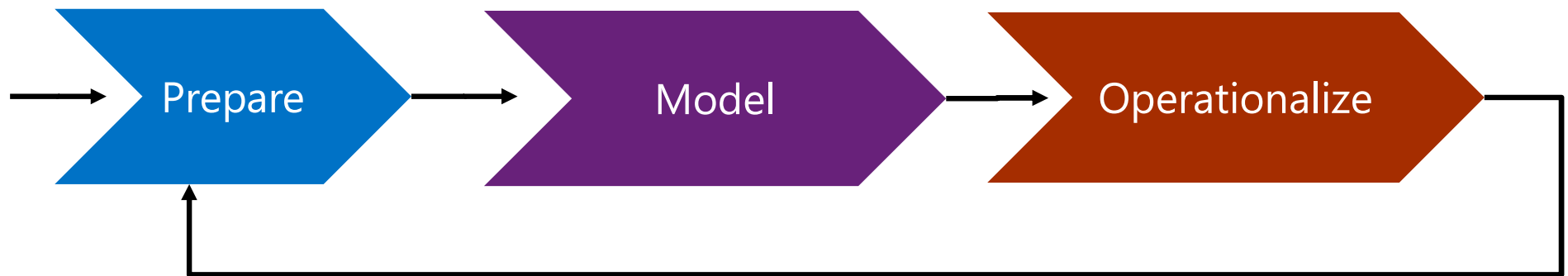
Worker R processes on Data nodes

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 make predictions

Operationalize: Apply 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

The screenshot displays the Microsoft Azure portal interface for provisioning a new HDInsight cluster. The browser address bar shows the URL <https://ms.portal.azure.com/#cre>. The page title is "Cluster Type configuration".

Left Sidebar (Navigation):

- New
- Resource groups
- All resources
- Recent
- App Services
- SQL databases
- Virtual machines (classic)
- Virtual machines
- Cloud services (classic)
- Subscriptions
- HDInsight Clusters
- Browse >

Main Content Area:

New HDInsight Cluster

- Cluster Name:** marinch101 (Validated)
- 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** button

Cluster Type configuration

Learn about HDInsight and cluster versions. [Learn more](#)

Cluster Type: R Server on Spark

Operating System: Linux

Version: 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
R Server on Spark is Premium only	99.9% Uptime SLA
	Automatic patching
	Microsoft R Server for HDInsight
+ 0.00 USD/CORE/HOUR	+ 0.02 USD/CORE/HOUR

Select button

Scaling a cluster

The screenshot shows the 'Scale Cluster' settings page in the Microsoft Azure portal. The page is divided into three main sections: a left sidebar with navigation links, a central settings pane, and a right-hand configuration pane.

Left Sidebar: Contains a search bar, a list of settings categories (SUPPORT & TROUBLESHOOTING, GETTING STARTED, CONFIGURATION), and a 'Scale Cluster' tile.

Central Settings Pane: Displays a search bar and a list of settings categories. The 'Scale Cluster' option is highlighted under the 'CONFIGURATION' section.

Right-hand Configuration Pane: Shows the current configuration for the cluster. The 'Number of Worker nodes' is set to 4. The 'Worker Nodes Pricing Tier' is D4 (4 nodes, 32 cores). The 'Head Node Pricing Tier' is D4 (2 nodes, 16 cores). A cost summary table shows the total cost of 7.46 USD/HOUR (ESTIMATED).

Configuration	Value
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)
```

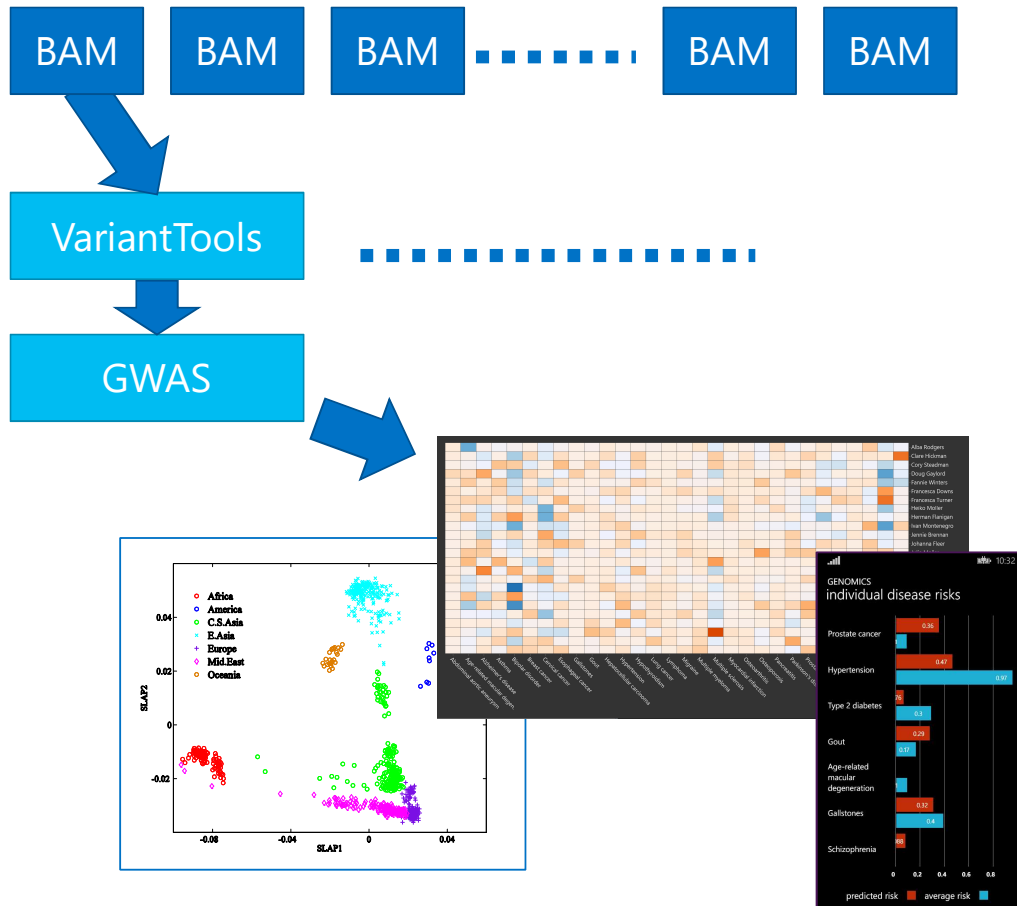
Publish Web Service from R

```
#####  
# 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



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

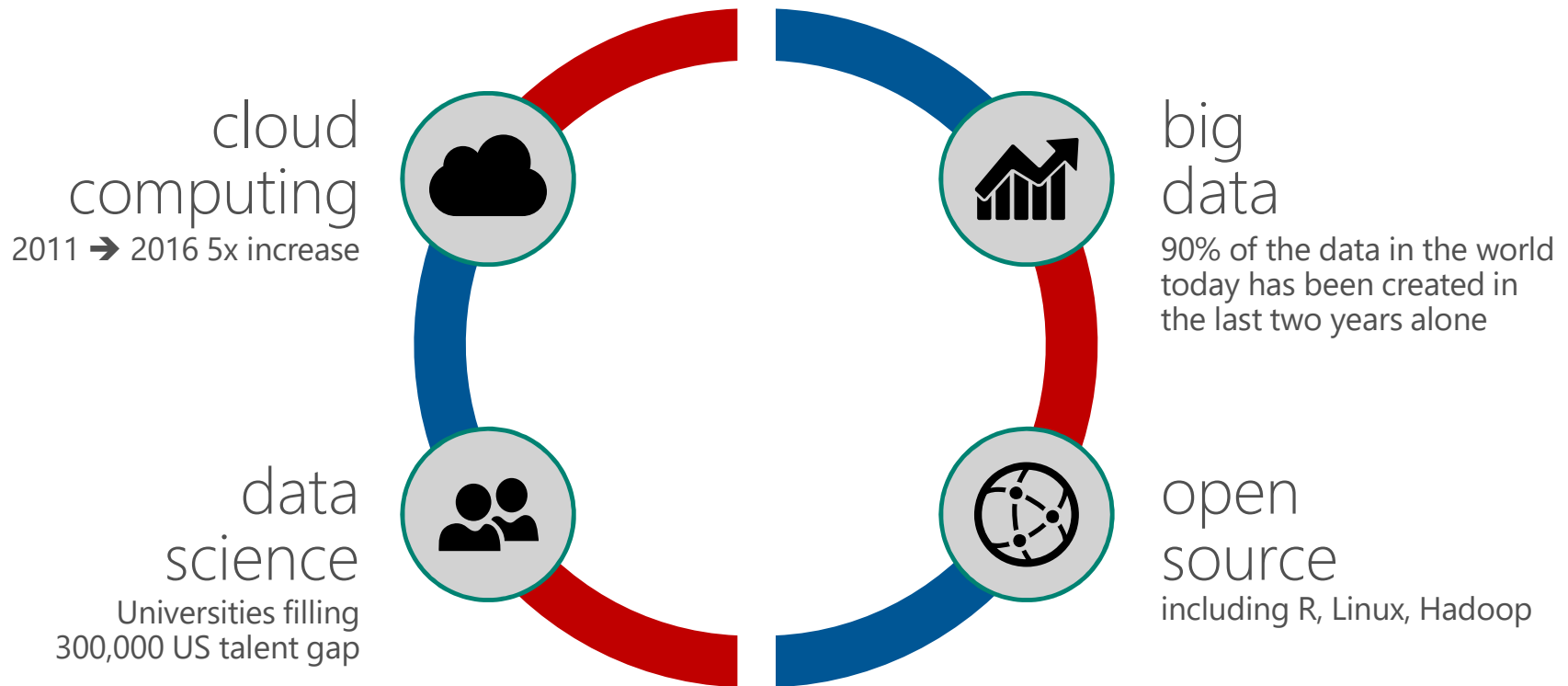
Analytics

- Disease Risk
- Ancestry

Presentation

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

The Four Transformational Trends





For more information...

HDInsight Premium
microsoft.com/hdinsight

R Server
microsoft.com/r-server