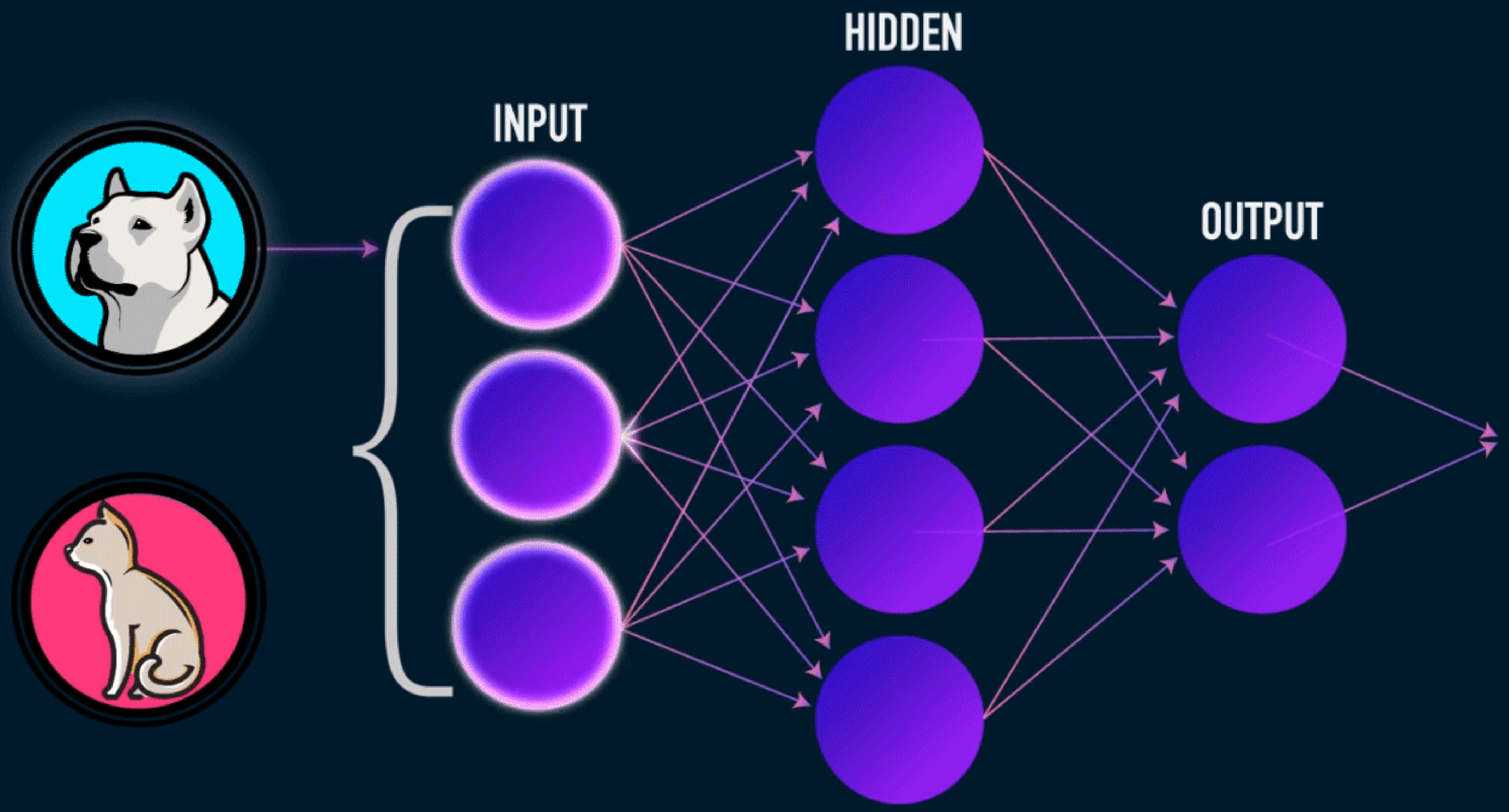
Deep Learning on Azure Databricks

Fahd . Rafi @ Microsoft .com

Global Azure Bootcamp 2019

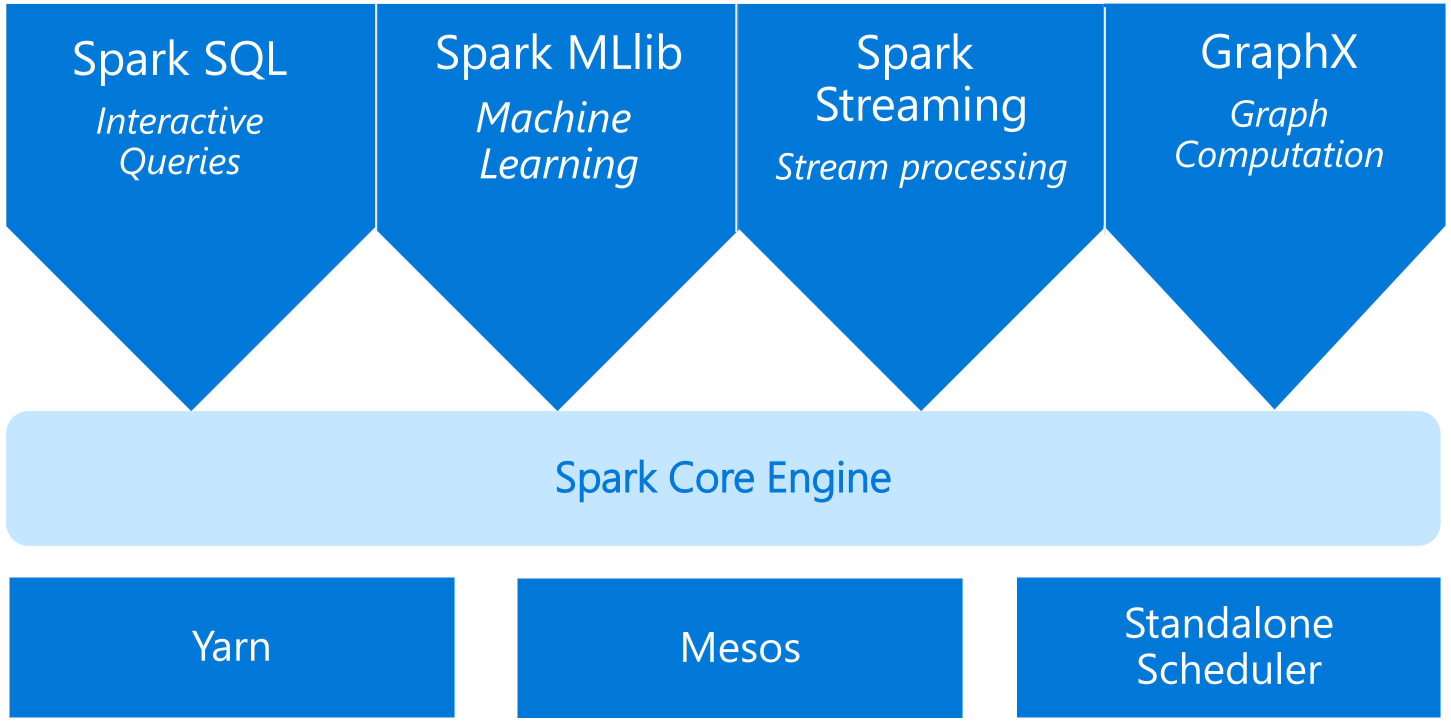

HIDDEN
INPUT
OUTPUT

APACHE SPARK

An unified, open source, parallel, data processing framework for Big Data Analytics

Spark Unifies:

- Batch Processing
- Interactive SQL
- Real-time processing
- Machine Learning
- Deep Learning
- Graph Processing

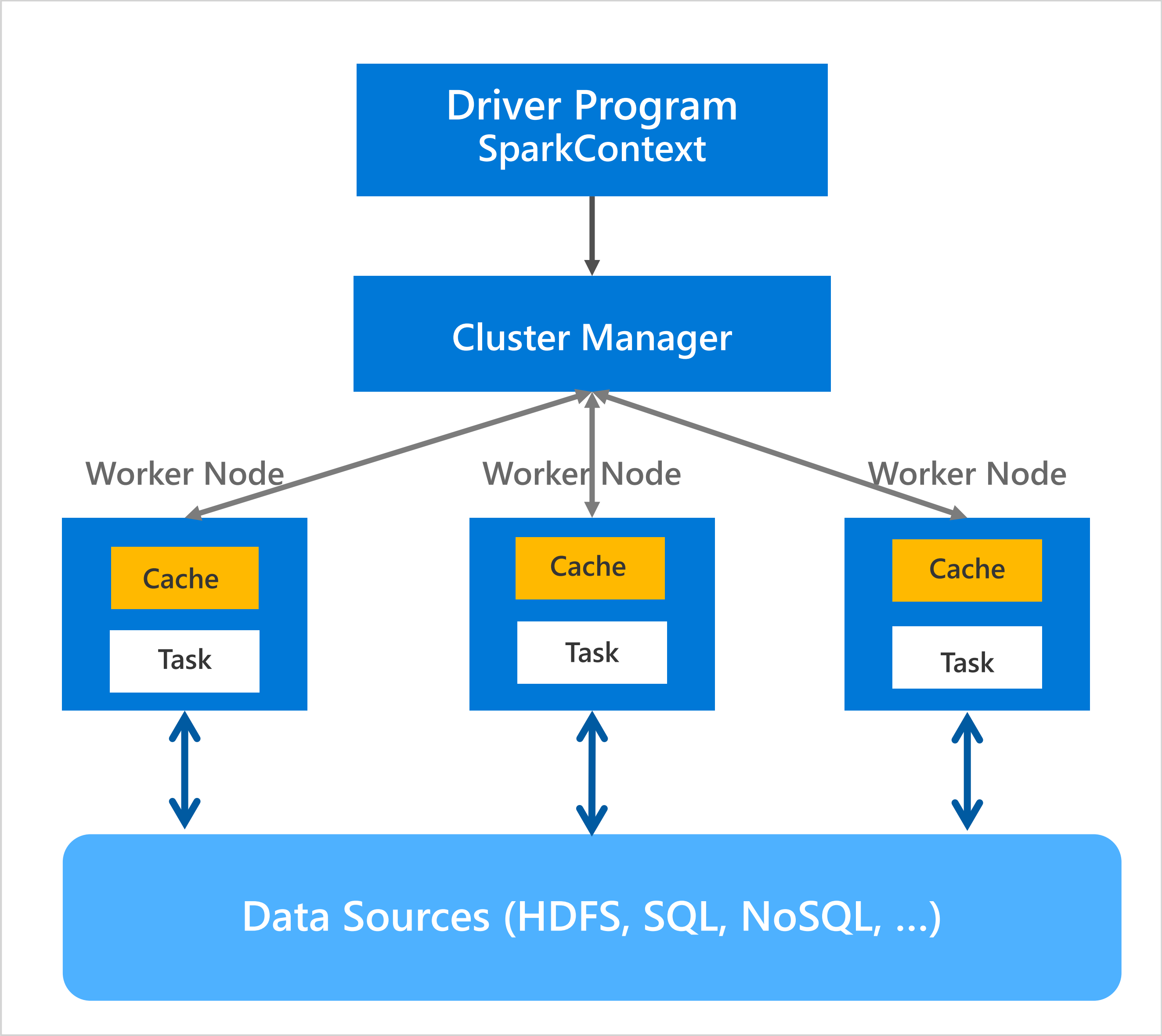


Azure Databricks

Databricks Spark as a managed service on Azure

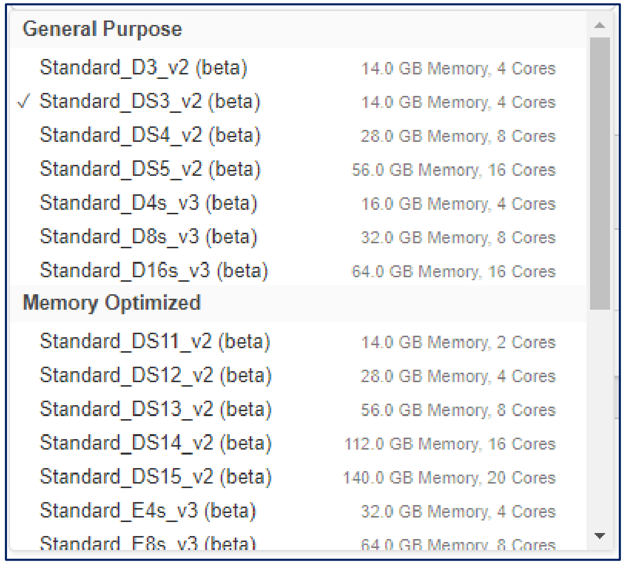
GENERAL SPARK CLUSTER ARCHITECTURE

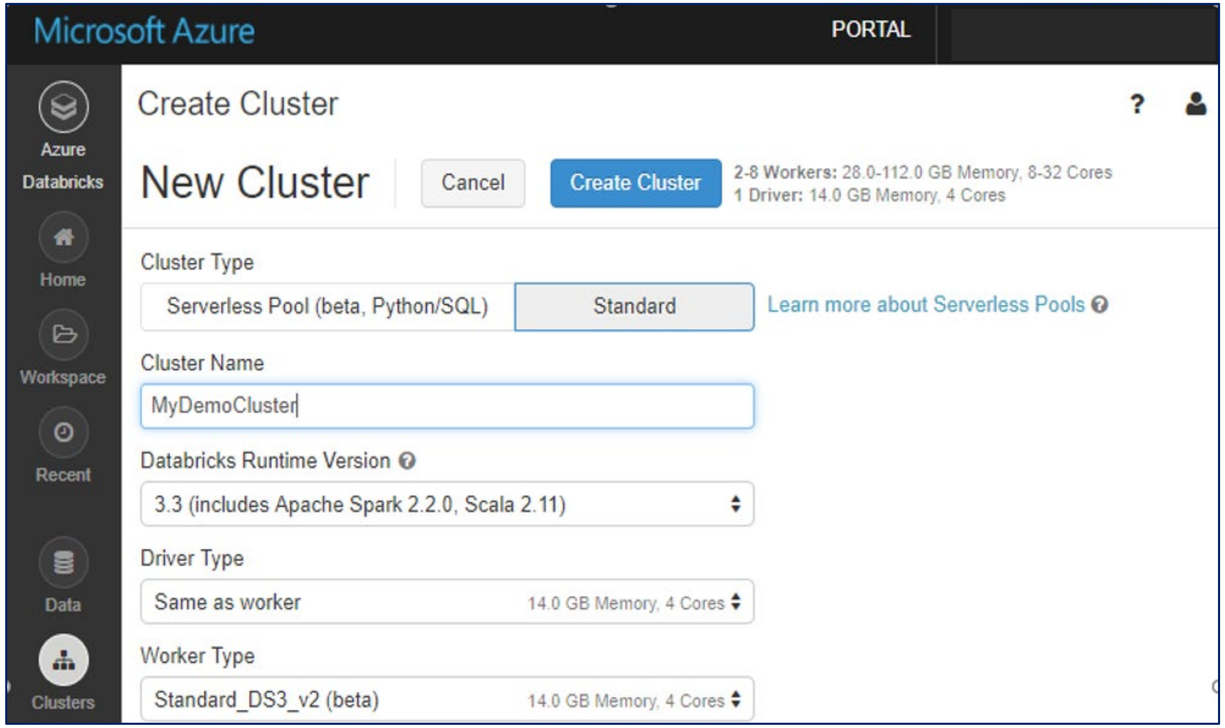
- ‘Driver’ runs the user’s ‘main’ function and executes the various parallel operations on the worker nodes.
- The results of the operations are collected by the driver
- The worker nodes read and write data from/to Data Sources including HDFS.
- Worker node also cache transformed data in memory as RDDs (Resilient Data Sets).
- Worker nodes and the Driver Node execute as VMs in public clouds (AWS, Google and Azure).



CLUSTER CREATION

- You can create two types of clusters – *Standard* and *Serverless Pool* (see next slide)
- While creating a cluster you can specify:
  - Number of nodes
  - Autoscaling and Auto Termination policy
  - Auto Termination policy
  - Spark Configuration details
  - The Azure VM instance types for the Driver and Worker Nodes





Graphical wizard in the Azure Databricks portal to create a Standard Cluster

CLUSTERS: AUTO SCALING AND AUTO TERMINATION

Simplifies cluster management and reduces costs by eliminating wastage

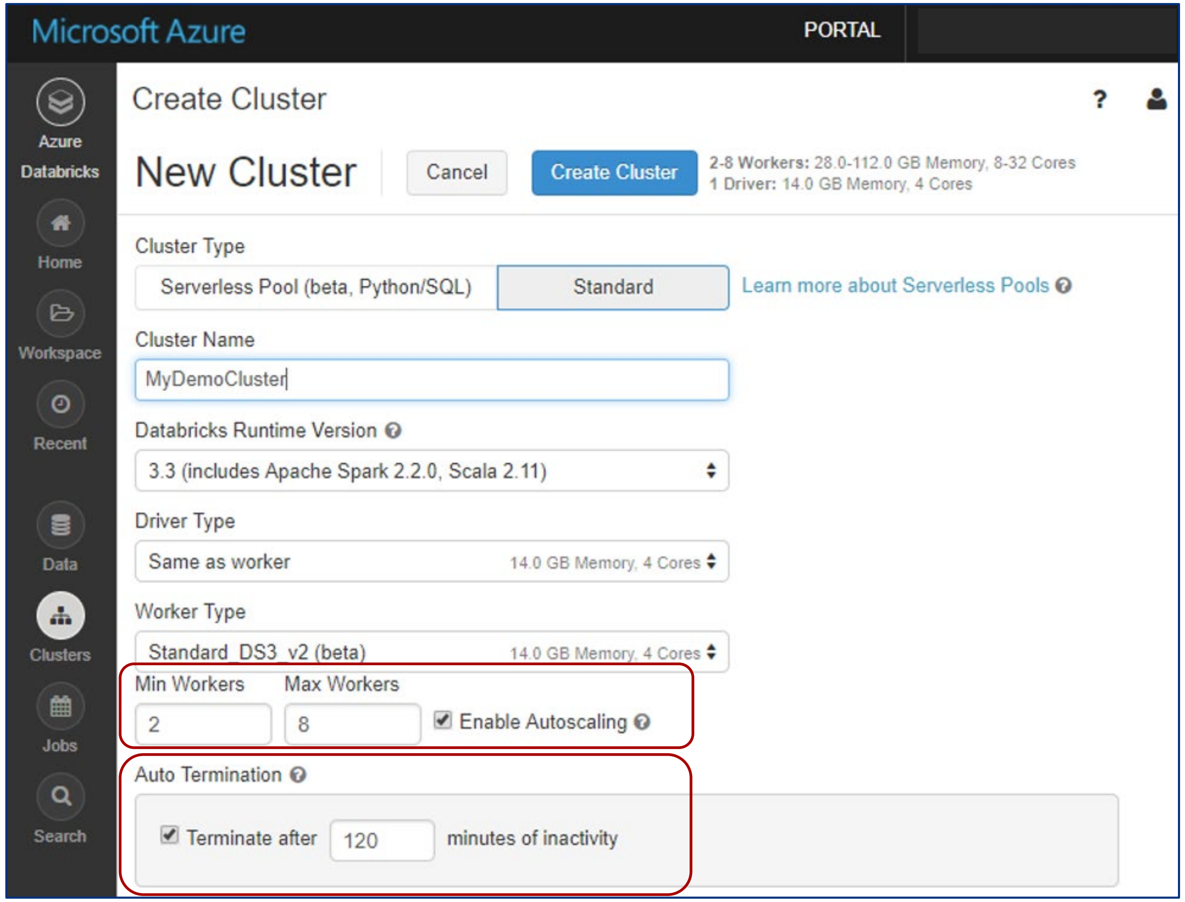
When creating Azure Databricks clusters you can choose Autoscaling and Auto Termination options.

Autoscaling: Just specify the min and max number of clusters. Azure Databricks automatically scales up or down based on load.

Auto Termination: After the specified minutes of inactivity the cluster is automatically terminated.

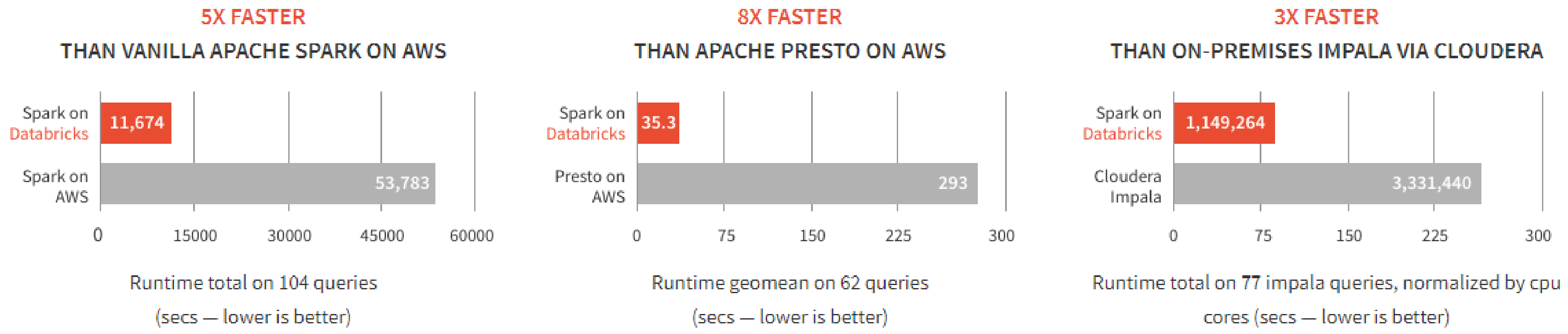
Benefits:

- You do not have to guess, or determine by trial and error, the correct number of nodes for the cluster
- As the workload changes you do not have to manually tweak the number of nodes
- You do not have to worry about wasting resources when the cluster is idle. You only pay for resource when they are actually being used
- You do not have to wait and watch for jobs to complete just so you can shutdown the clusters



DATABRICKS SPARK IS FAST

Benchmarks have shown Databricks to often have better performance than alternatives

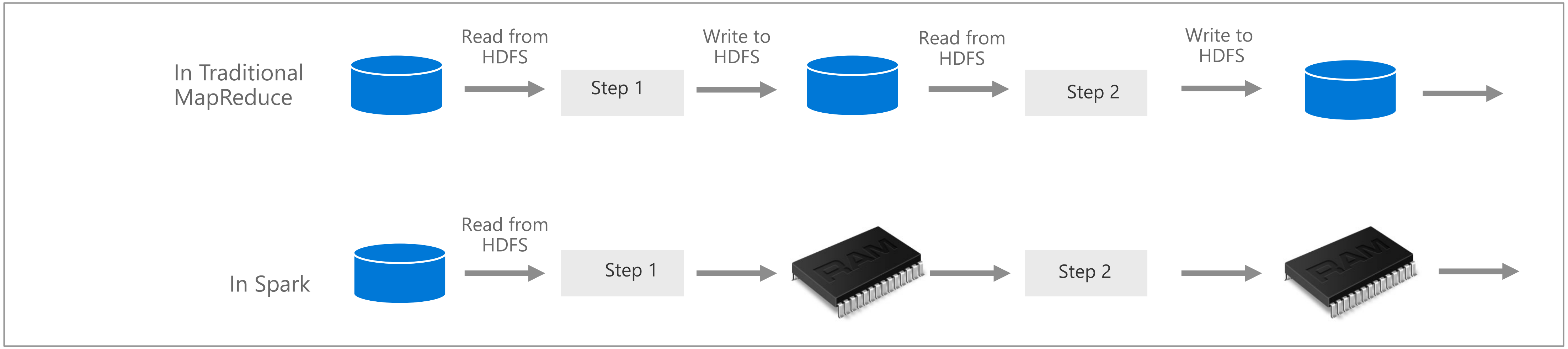


SOURCE: [Benchmarking Big Data SQL Platforms in the Cloud]

WHAT MAKES SPARK FAST?(1OF 2)

- **In-memory cluster computing:** Spark provides primitives for *in-memory* cluster computing. A Spark job can *load and cache* data into memory and query it repeatedly (iteratively) much quicker than disk-based systems.
- **Scala Integration:** Spark integrates into the Scala programming language, letting you manipulate distributed datasets like local collections. No need to structure everything as map and reduce operations
- **Faster Data-sharing:** Data-sharing between operations is faster as data is in-memory:
  - In (traditional) Hadoop data is shared through HDFS which is expensive. HDFS maintains three replicas.
  - Spark stores data in-memory *without any replication*.

Data Sharing between Steps of a Job



WHAT MAKES SPARK FAST? (2 OF 2)

Databricks IO Cache automatically caches 'remote' data on 'local nodes' to accelerate data reads

- A copy of the remote file is created in the node's local storage
  - Local data is stored in a fast intermediate format
  - Currently *Parquet* file format is supported
- Remote data is cached automatically
- Supports *DBFS, HDFS, Azure Blob Storage and Azure Data Lake store*
- DBIO Cache lets you"
  - Enable or disable caching at anytime
  - Cache only a select subset of the data
- DBIO Cache has to be configured during cluster creation. The '*max disk space per node reserved for cached data*' must be specified during cluster creation

You can Monitor the state of the DBIO cache in the Portal

Storage

Parquet IO Cache

Host	Disk Usage	Max Disk Usage Limit	Percent Disk Usage	Metadata Cache Size	Max Metadata Cache Size Limit	Percent Metadata Usage
10.0.185.226	8.3 GB	442.4 GB	1 %	6.8 MB	8.8 GB	0 %
10.0.194.201	8.2 GB	442.4 GB	1 %	6.8 MB	8.8 GB	0 %
10.0.199.229	8.2 GB	442.4 GB	1 %	6.9 MB	8.8 GB	0 %
10.0.215.147	8.1 GB	442.4 GB	1 %	7.0 MB	8.8 GB	0 %
Total	32.8 GB	1769.5 GB	1 %	27.4 MB	35.4 GB	0 %

RDDS AND DBIO CACHE - DIFFERENCES

DBIO cache and RDDs are both caches that can be used together

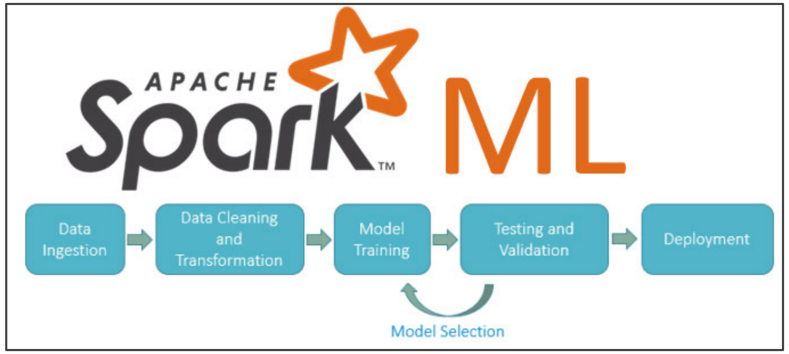
Capability	Comment
Availability	• RDD is part of Apache Spark • Databricks IO cache is available only to Databricks customers.
Type of data stored	• The RDD cache can be used to store the result of any subquery. • The DBIO cache is designed to speed-up scans by creating local copies of remote data. It can improve the performance of a wide range of queries, but cannot be used to store results of arbitrary subqueries.
Performance	• The data stored in the DBIO cache can be read and operated on faster than the data in the RDD cache. This is because the DBIO cache uses efficient decompression algorithms, and outputs data in the optimal format for further processing using whole-stage code generation.
Automatic vs manual control	• When using the RDD cache it is necessary to manually choose tables or queries to be cached. • When using the DBIO cache the data is added to the cache automatically whenever it has to be fetched from a remote source. This process is fully transparent and does not require any action from the user.
Disk vs memory-based	• Unlike the RDD cache, the DBIO cache is stored entirely on the local disk.

Machine Learning and Deep Learning

SPARK MACHINE LEARNING(ML) OVERVIEW

Enables Parallel, Distributed ML for large datasets on Spark Clusters

- Offers a set of parallelized machine learning algorithms (see next slide)
- Supports Model Selection (hyperparameter tuning) using Cross Validation and Train-Validation Split.
- Supports Java, Scala or Python apps using DataFrame-based API (as of Spark 2.0). Benefits include:
  - An uniform API across ML algorithms and across multiple languages
  - Facilitates ML pipelines (enables combining multiple algorithms into a single pipeline).
  - Optimizations through Tungsten and Catalyst
- Spark MLlib comes pre-installed on Azure Databricks
- 3rd Party libraries supported include: H20 Sparkling Water, SciKit-learn and XGBoost

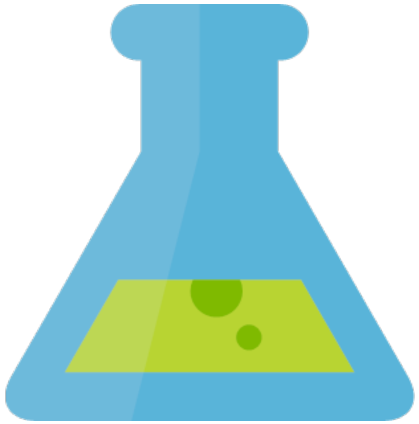


MMLSPARK

Microsoft Machine Learning Library for Apache Spark (MMLSpark) lets you easily create scalable machine learning models for large datasets.

It includes integration of SparkML pipelines with the Microsoft Cognitive Toolkit and OpenCV, enabling you to:

- Ingress and pre-process image data
- Featurize images and text using pre-trained deep learning models
- Train and score classification and regression models using implicit featurization



SPARK ML ALGORITHMS

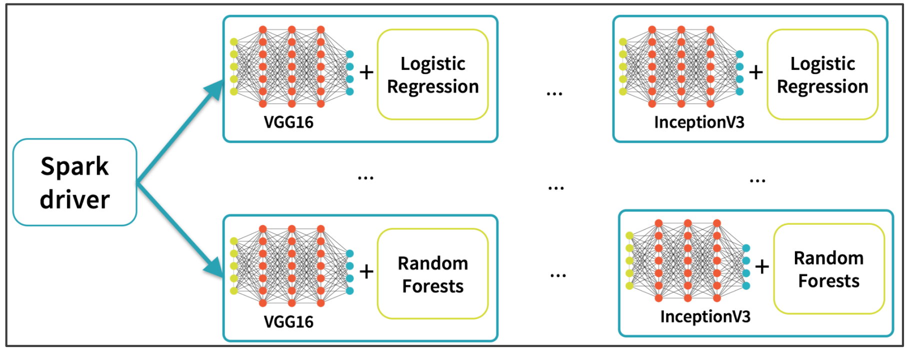
Spark ML Algorithms

Classification and Regression	• Linear Models (SVMs, logistic regression, linear regression) • Naïve Bayes • Decision Trees • Ensembles of trees (Random Forest, Gradient-Boosted Trees) • Isotonic regression
Clustering	• k-means and streaming k-means • Gaussian mixture • Power iteration clustering (PIC) • Latent Dirichlet allocation (LDA)
Collaborative Filtering	• Alternating least squares (ALS)
Dimensionality Reduction	• SVD • PCA
Frequent Pattern Mining	• FP-growth • Association rules
Basic Statistics	• Summary statistics • Correlations • Stratified sampling • Hypothesis testing • Random data generation

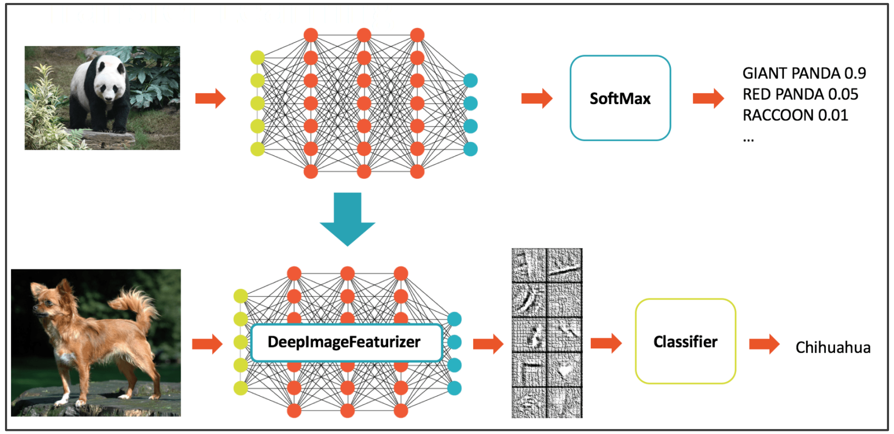
D E E P L E A R N I N G

Azure Databricks supports and integrates with a number of Deep Learning libraries and frameworks to make it easy to build and deploy Deep Learning applications

- Supports Deep Learning Libraries/frameworks including:
  - Microsoft Cognitive Toolkit (CNTK).
    - Article explains how to install CNTK on Azure Databricks.
  - TensorFlowOnSpark
  - BigDL
- Offers Spark Deep Learning Pipelines, a suite of tools for working with and processing images using deep learning using transfer learning. It includes high-level APIs for common aspects of deep learning so they can be done efficiently in a few lines of code:
  - Image loading
  - Applying pre-trained models as transformers in a Spark ML pipeline
  - Transfer learning
  - Distributed hyperparameter tuning
  - Deploying models in DataFrames and SQL



Distributed Hyperparameter Tuning



Transfer Learning

Resources and Demo

VM Types:
https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sizes-gpu

Azure Databricks Docs (Deep Learning Guide):
https://docs.azuredatabricks.net/applications/deep-learning/index.html#deep-learning-guide

Go to Azure Portal for Demo

Microsoft