

Big Data Fundamentals with Apache Spark

by: Dr Tassadaq Hussain

Mr. Muhammad Danish

**Professor Department of Electrical Engineering
Namal University Mianwali**

Collaborations:

Barcelona Supercomputing Center, Spain

European Network on High Performance and Embedded Architecture and Compilation

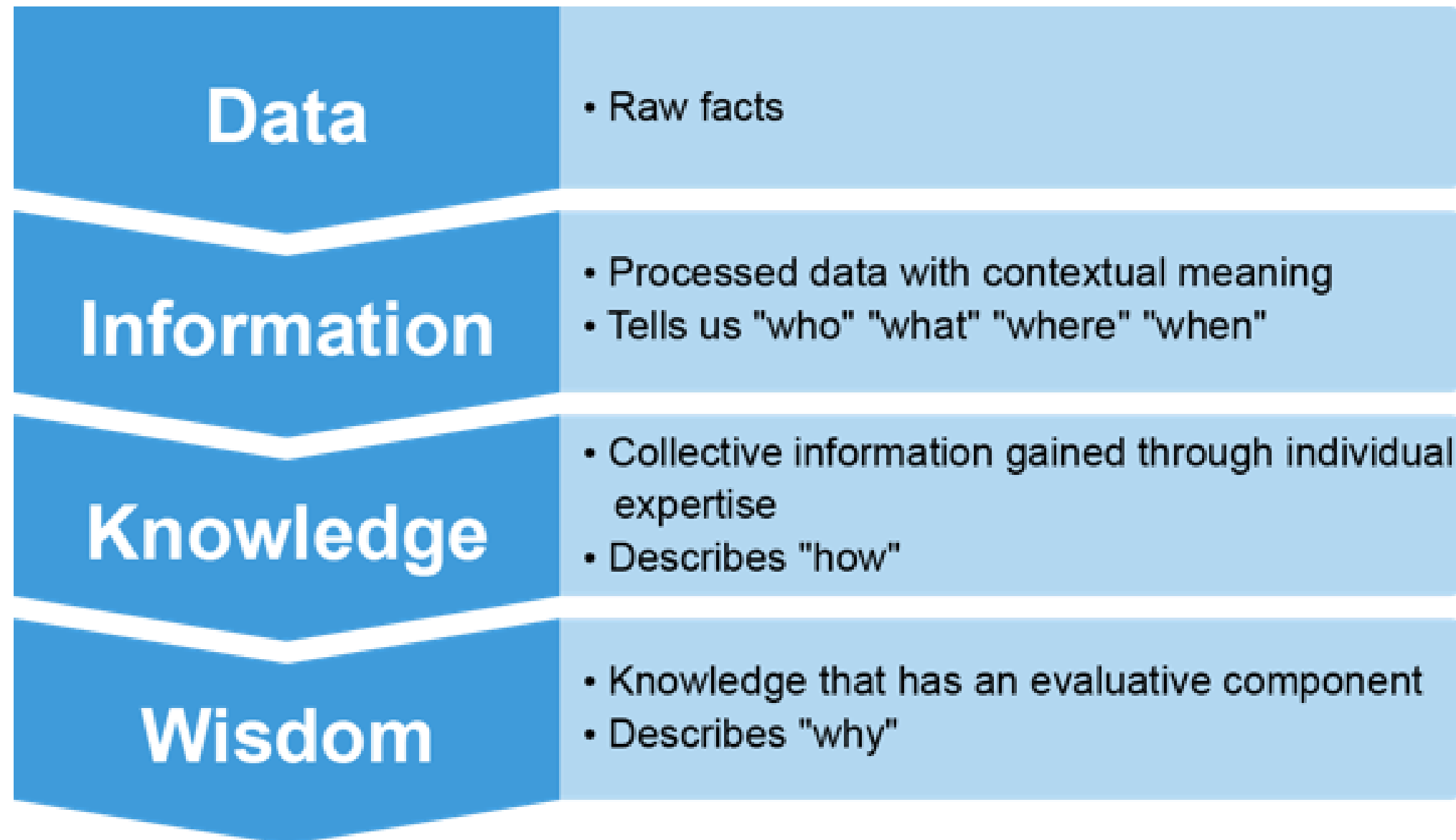
Pakistan Supercomputing Center

Making data work for you



Use data to better describe the past and present or better predict the future

The Data-Information-Knowledge-Wisdom

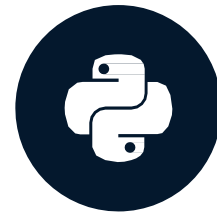


The data science workflow



Fundamentals of Big Data

B I G D A T A F U N D A M E N T A L S W I T H P
Y S P A R K

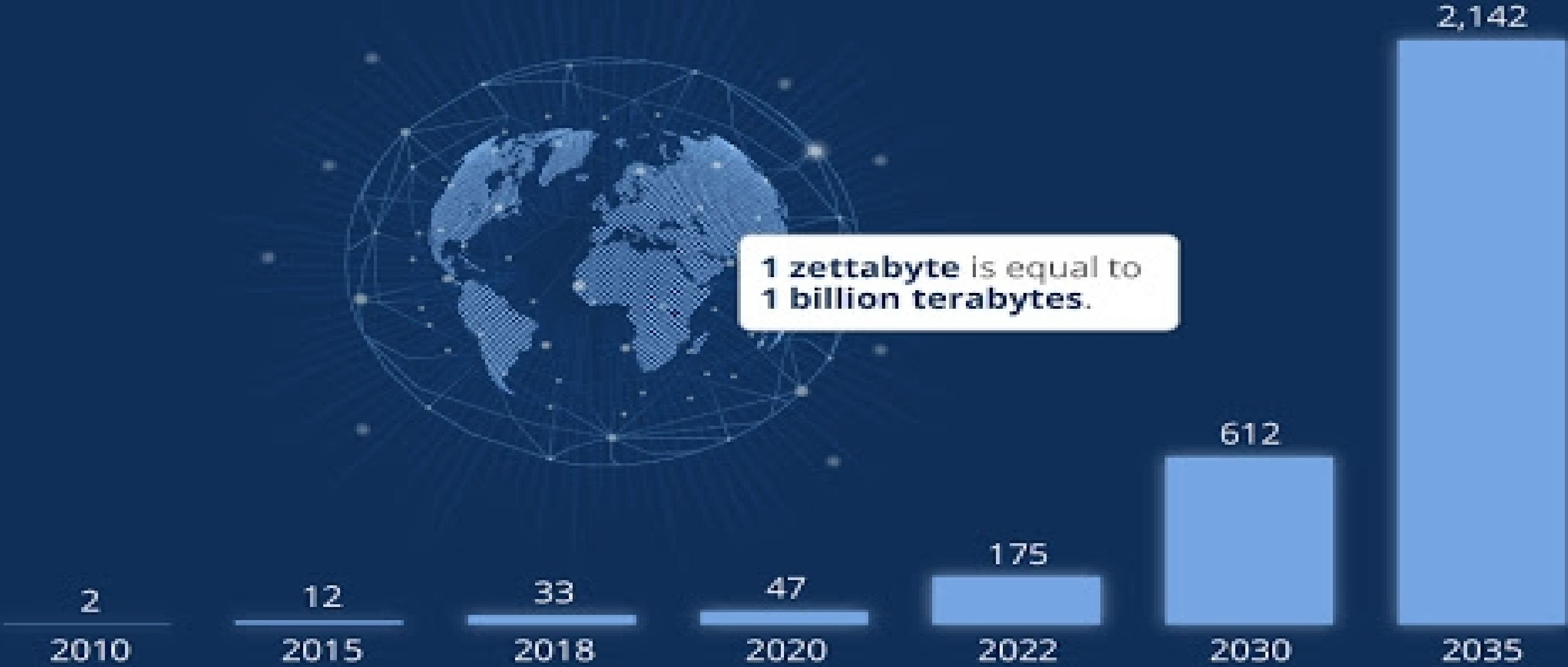


Global Data Creation is About to Explode

Actual and forecast amount of data created worldwide 2010-2035 (in zettabytes)



1 zettabyte is equal to
1 billion terabytes.



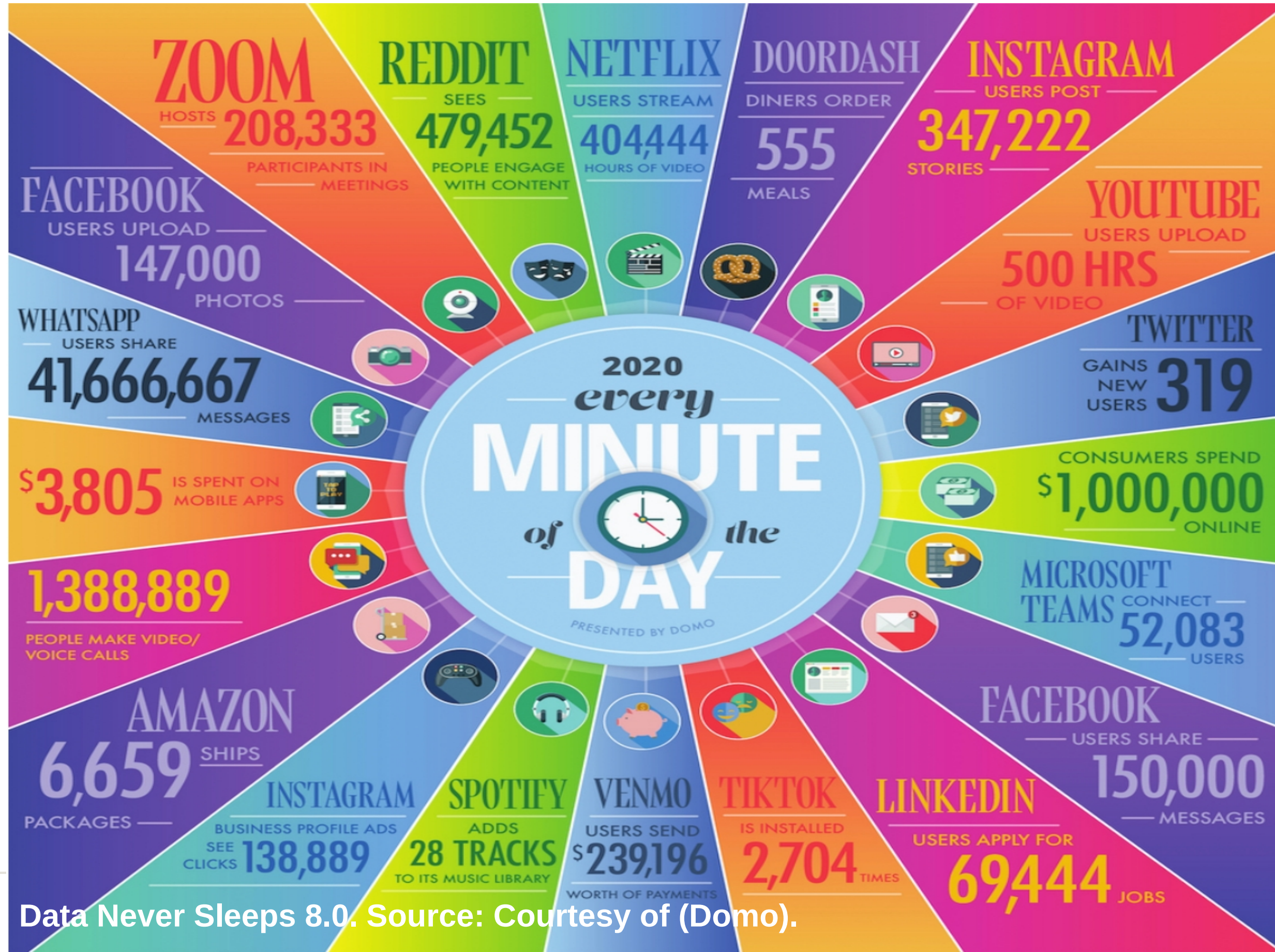
57.76 US\$

What is Big Data?

- Big data is a term used to refer to the study and applications of data sets that are too complex for traditional data-processing software - Wikipedia

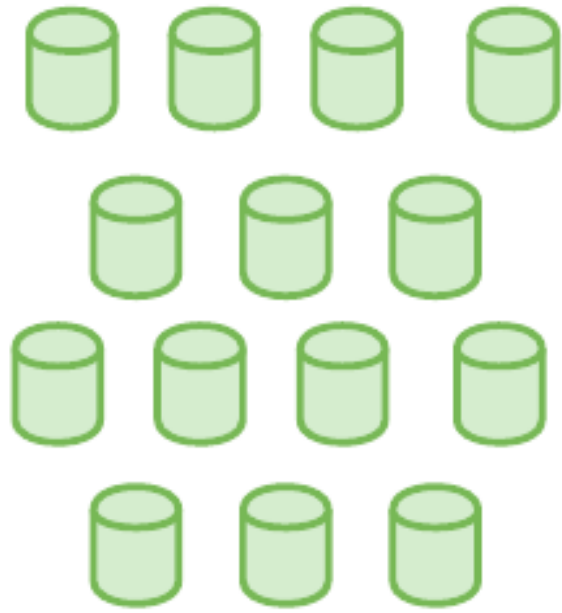
The 3 Vs of Big Data

- Volume: Size of the data
 - Variety: Different sources and formats
 - Velocity: Speed of the data
-



Data Never Sleeps 8.0. Source: Courtesy of (Domo).

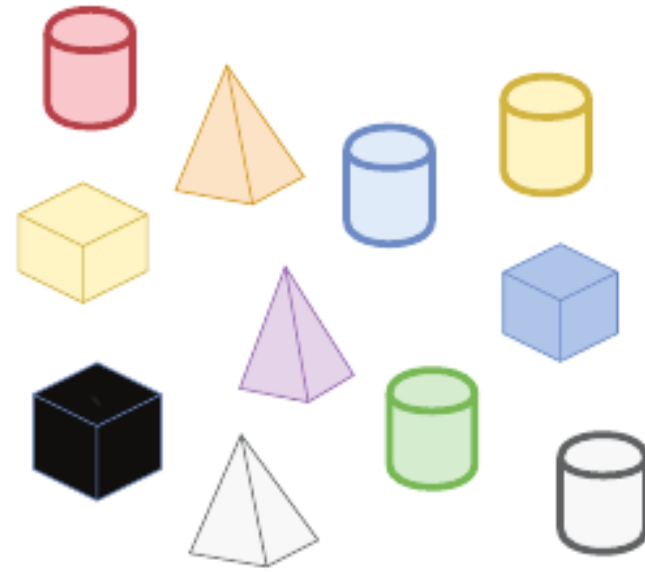
Data at rest



Terabytes to zettabytes
of data to process

Volume

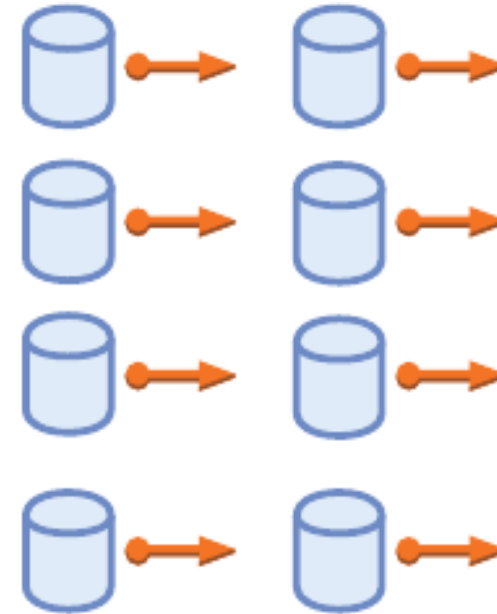
Data in many forms



Structured,
unstructured, and semi-
structured

Variety

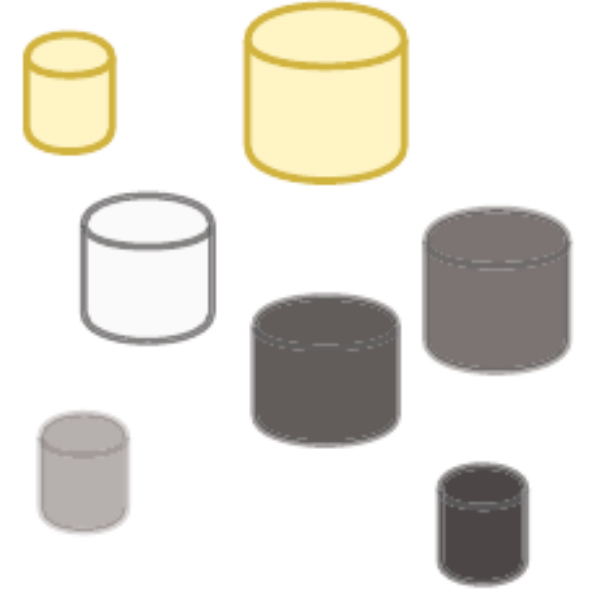
Data in motion



Streaming data,
microseconds to seconds
to respond

Velocity

Data in doubt



Uncertainty due to data
inconsistency, ambiguities,
deception, and model
approximations

Veracity

Some Candid Applications

- Recommendation Engines
- Media Analytics:
 - Sentiment Analysis
 - Trend Analysis

What recommendations look like



The Power of Recommendation Engines

powered by increasingly sophisticated models that analyze transaction data and digital footprints (for example, what topics are hot on social networks). **Already, 35 percent of what consumers purchase on Amazon and 75 percent of what they watch on Netflix come from product recommendations based on such algorithms.** Company-directed marketing is also competing for attention with recommendations through social networks, user-generated content, and word-of-mouth for the average

Ian Mackenzie, Chris Meyer, and Steve Noble
McKinsey & Company, October 2013

Big Data processing

systems

MapReduce: Scalable and fault tolerant framework written

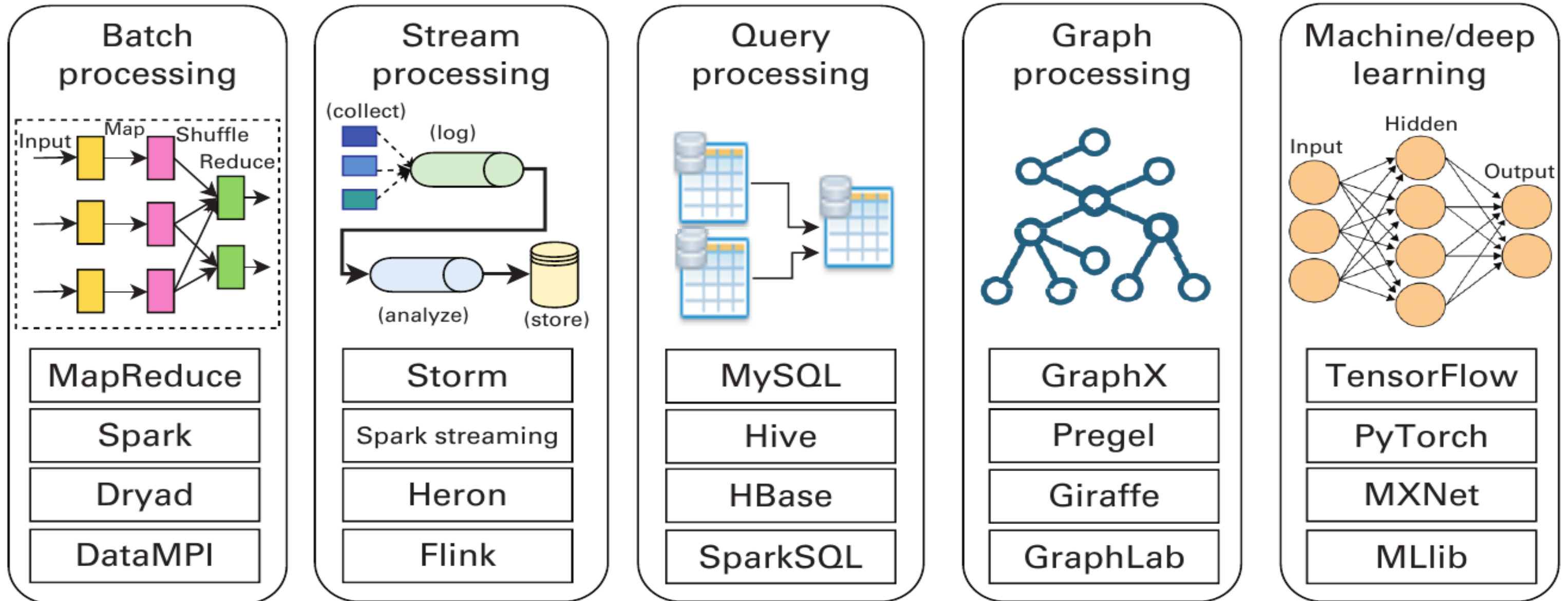
- in Java Open source

Batch processing

- Apache Spark: General purpose and lightning fast cluster computing
 - system Open source

Both batch and real-time data processing

Big data applications



Storage middleware and API interface

Features of Apache Spark

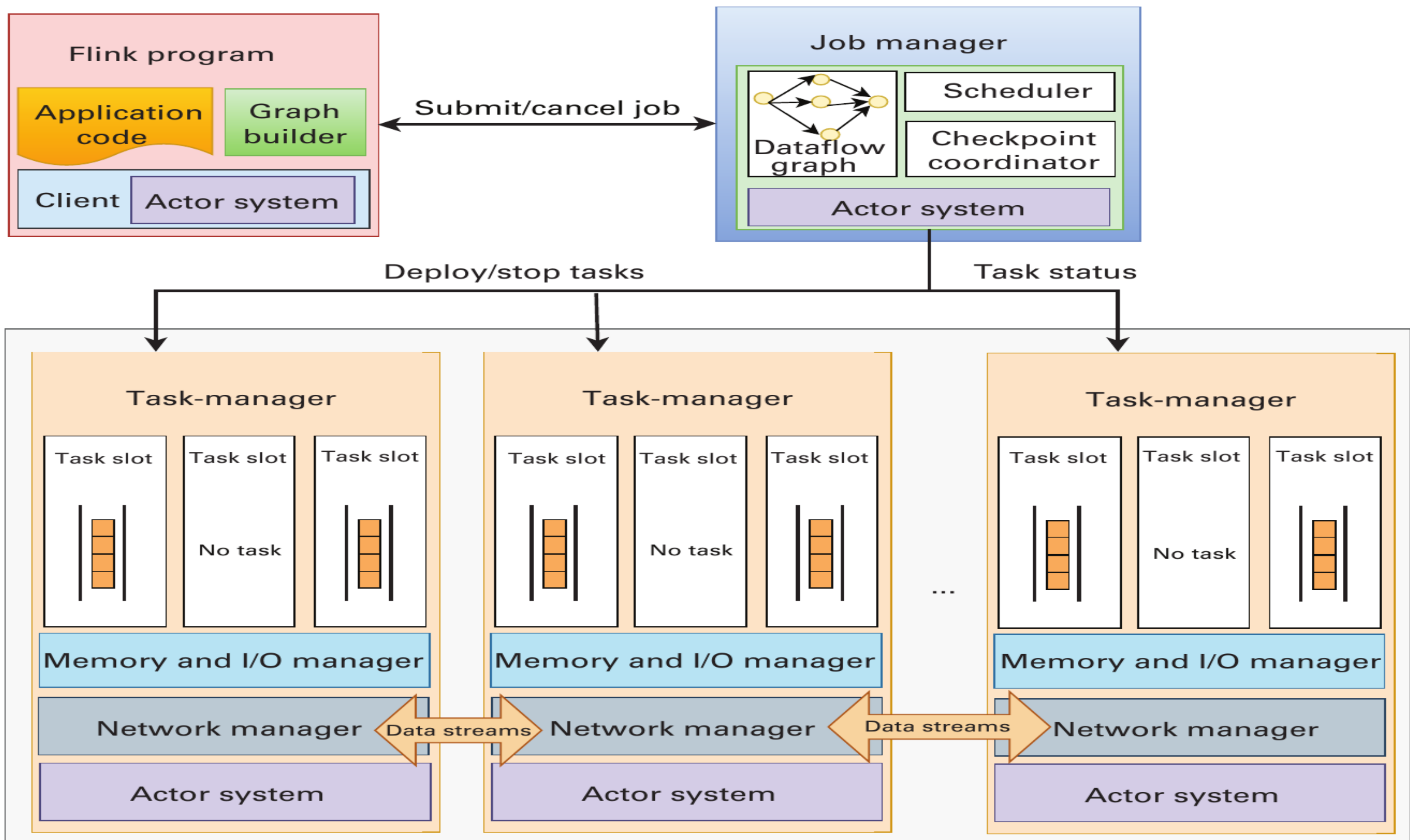
distributed cluster computing framework

- Efficient in-memory computations for large data Sets
- Lightning fast data processing
- framework

Provides support for Java, Scala, Python, R and SQL

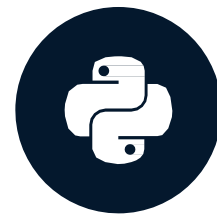
Spark modes of deployment

- Local mode: Single machine such as your laptop
 - Local mode is convenient for testing, debugging and demonstration
- Cluster mode: Set of pre-defined machines Good for production
- Workflow: Local -> clusters No code change necessary



PySpark: Spark with Python

B I G D A T A F U N D A M E N T A L S W I T H P
Y S P A R K



Overview of

PySpark

• Apache Spark is written in Scala

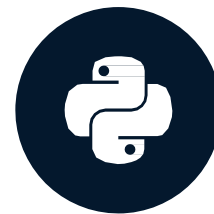
- To support Python with Spark, Apache Spark Community released
- PySpark Similar computation speed and power as Scala
- PySpark APIs are similar to Pandas and Scikit-learn

Understanding SparkSession

- SparkSession is an entry point into the world of Spark
- Spark An entry point is a way of connecting to Spark cluster
- Spark cluster An entry point is like a key to the house

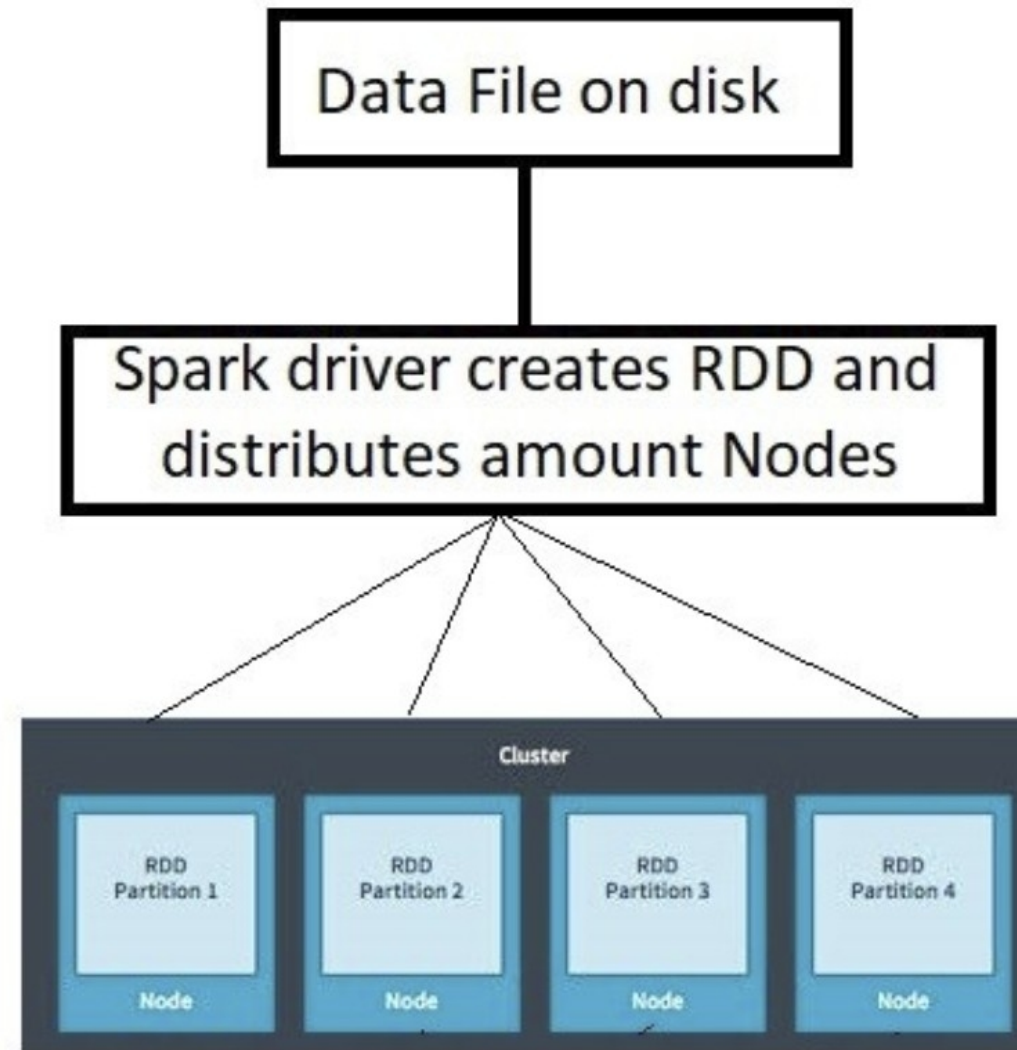
PySpark RDD

B I G D A T A F U N D A M E N T A L S
W I T H P Y S
P A R K



What is

RDD? Resilient Distributed Datasets



Decomposing RDDs

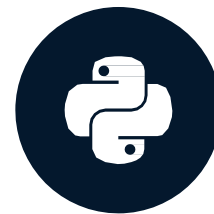
- Resilient Distributed Datasets
 - Resilient: Ability to withstand failures
 - Distributed: Spanning across multiple machines
 - Datasets: Collection of partitioned data e.g, Arrays, Tables, Tuples etc.,

Creating RDDs. How to do it?

- Parallelizing an existing collection of objects
- External datasets:
 - Files in HDFS
 - Objects in Amazon S3
 - bucket Lines in a text file

RDD operations in PySpark

B I G D A T A F U N D A M E N T A L S W I T H
H P Y S P A R K



Overview of PySpark operations

Spark Operations =


TRANSFORMATIONS

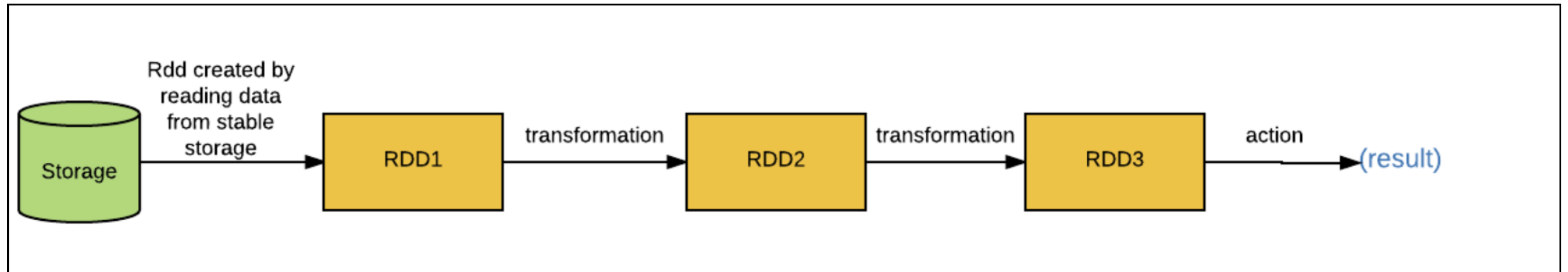
+


ACTIONS

- Transformations create new RDDs
- Actions perform computation on the RDDs

RDD Transformations

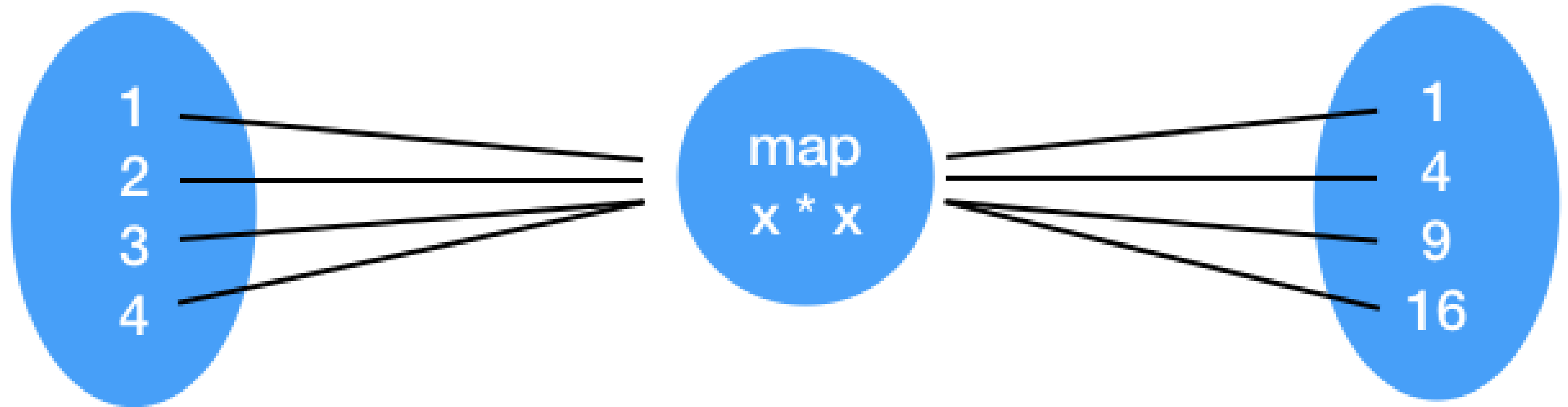
- Transformations follow Lazy evaluation



- Basic RDD Transformations
 - `map()` , `filter()` , `flatMap()`

map() Transformation

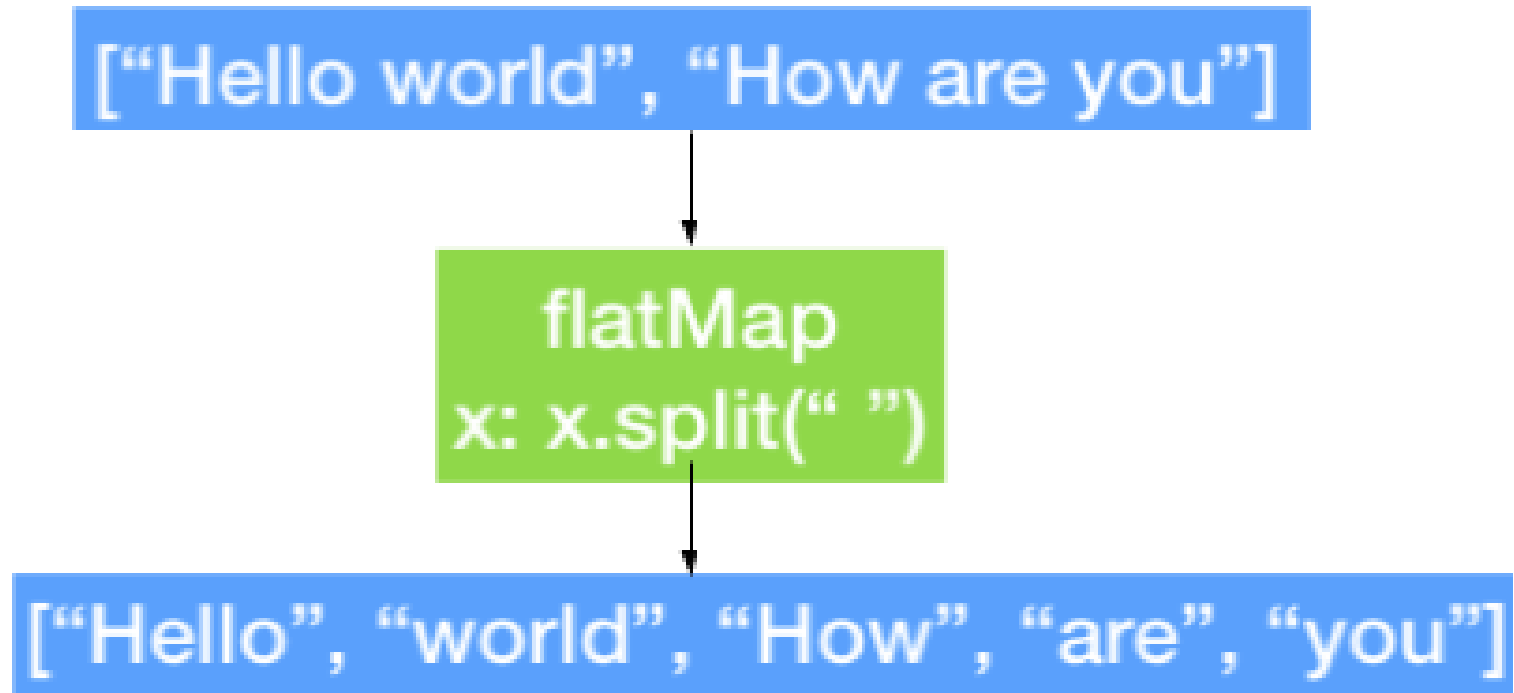
- map() transformation applies a function to all elements in the RDD



flatMap()

Transformation

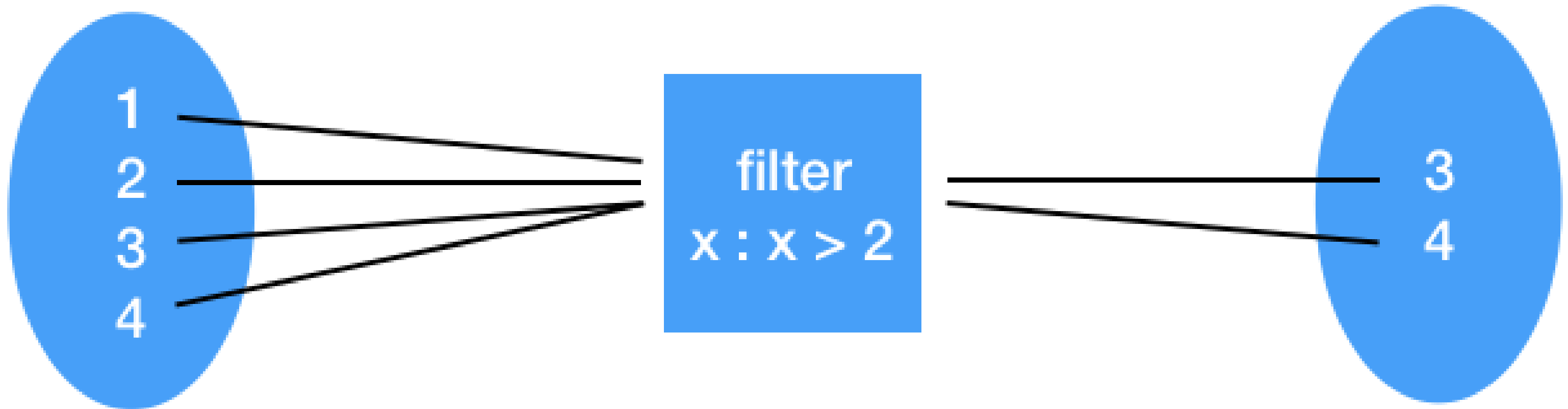
- flatMap() transformation returns multiple values for each element in the original RDD



filter()

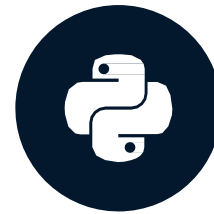
Transformation

- Filter transformation returns a new RDD with only the elements that pass the condition



Working with Pair RDDs in PySpark

B I G D A T A F U N D A M E N T A L S W I T H
H P Y S P A R K



Introduction to pair RDDs in PySpark

- Datasets are usually key/value pairs
- Each row is a key and maps to one or more values
- Pair RDD is a special data structure to work with this kind of datasets
- Pair RDD: Key is the identifier and value is data

Creating pair RDDs

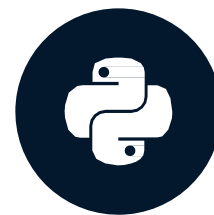
- Two common ways to create pair RDDs
 - From a list of key-value tuple
 - From a regular RDD
- Get the data into key/value form for paired RDD

Transformations on pair RDDs

- All regular transformations work on pair RDD
- Have to pass functions that operate on key value pairs rather than on individual elements
- Examples of paired RDD Transformations
 - `reduceByKey(func)`: Combine values with the same key
 - `groupByKey()`: Group values with the same key
 - `sortByKey()`: Return an RDD sorted by the key
 - `join()`: Join two pair RDDs based on their key

PySpark DataFrames

B I G D A T A F U N D A M E N T A L S
W I T H P Y S
P A R K



What are PySpark

DataFrames?

- PySpark SQL is a Spark library for structured data. It provides more information about the structure of data and computation
- PySpark DataFrame is an immutable distributed collection of data with named columns
- Designed for processing both structured (e.g relational database) and semi-structured data (e.g JSON)
- Dataframe API is available in Python, R, Scala, and Java
- DataFrames in PySpark support both SQL queries (`SELECT * from table`) or expression methods (`df.select()`)

Pandas DataFrame vs PySpark

DataFrame

- Pandas DataFrames are in-memory, single-server based structures and operations on PySpark run in parallel
- The result is generated as we apply any operation in Pandas whereas operations in PySpark DataFrame are lazy evaluation
- Pandas DataFrame as mutable and PySpark DataFrames are immutable
- Pandas API support more operations than PySpark Dataframe API

Creating DataFrames in PySpark

- Two different methods of creating DataFrames in PySpark
 - From existing RDDs using SparkSession's createDataFrame() method
 - From various data sources (CSV, JSON, TXT) using SparkSession's read method

Create a DataFrame from RDD

```
iphones_RDD = sc.parallelize([ ("XS",  
    2018, 5.65, 2.79, 6.24),  
("XR", 2018, 5.94, 2.98, 6.84),  
("X10", 2017, 5.65, 2.79, 6.13),  
("8Plus", 2017, 6.23, 3.07, 7.12)  
])
```

```
names = ['Model', 'Year', 'Height',  
    'Width', 'Weight']
```

```
iphones_df = spark.createDataFrame(iphones_RDD, schema=names)
```

Create a DataFrame from reading a CSV/JSON/TXT

```
df_csv = spark.read.csv("people.csv", header=True, inferSchema=True)
```

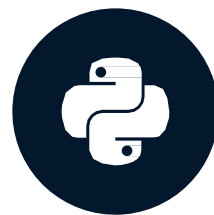
```
df_json = spark.read.json("people.json", header=True, inferSchema=True)
```

```
df_txt = spark.read.txt("people.txt", header=True, inferSchema=True)
```

- Path to the file and two optional parameters
- Two optional parameters
 - `header=True` , `inferSchema=True`

Interacting with PySpark DataFrames

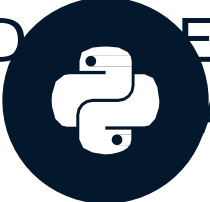
B I G D A T A F U N D A M E N T A L S
W I T H P Y S P A R K



DataFrame operators in PySpark

- DataFrame operations: Transformations and Actions
- DataFrame Transformations:
 - `select()`, `filter()`, `groupby()`, `orderBy()`,
`dropDuplicates()` and `withColumnRenamed()`
- DataFrame Actions :
 - `printSchema()`, `head()`, `show()`, `count()`, `columns`
and `describe()`

Interacting with DataFrames using PySpark SQL

BIG DATA FUNDAMENTALS WITH
H  ARK

DataFrame API vs SQL queries

- In PySpark You can interact with SparkSQL through DataFrame API and SQL queries
- DataFrame transformations and actions are easier to construct programmatically
- SQL queries can be concise and easier to understand and portable
- The operations on DataFrames can also be done using SQL queries

Executing SQL Queries

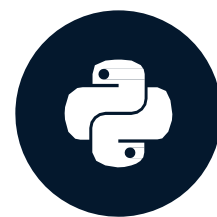
- The SparkSession `sql()` method executes SQL query
- `sql()` method takes a SQL statement as an argument and returns the result as DataFrame

```
df.createOrReplaceTempView("table1")
```

```
df2 = spark.sql("SELECT field1, field2 FROM  
table1") df2.collect()
```

Data Visualization in PySpark using DataFrames

B I G D A T A F U N D A M E N T A L S W I T H P
Y S P A R K



What is Data visualization?

• Data visualization is a way of representing your data in graphs or charts

- Open source plotting tools to aid visualization in Python:
 - Matplotlib, Seaborn, Bokeh etc.,
- Plotting graphs using PySpark DataFrames is done using three methods
 - pyspark_dist_explore library
 - toPandas()
 - HandySpark library

Using Pandas for plotting DataFrames

- It's easy to create charts from pandas DataFrames

```
test_df = spark.read.csv("test.csv", header=True,  
inferSchema=True)
```

```
test_df_sample_pandas =
```

```
test_df.toPandas()
```

```
test_df_sample_pandas.hist('Age')
```


Let's Get to Code!

B I G D A T A F U N D A M E N T A L
S W I T H P
Y S P A R K
