TOPIC F

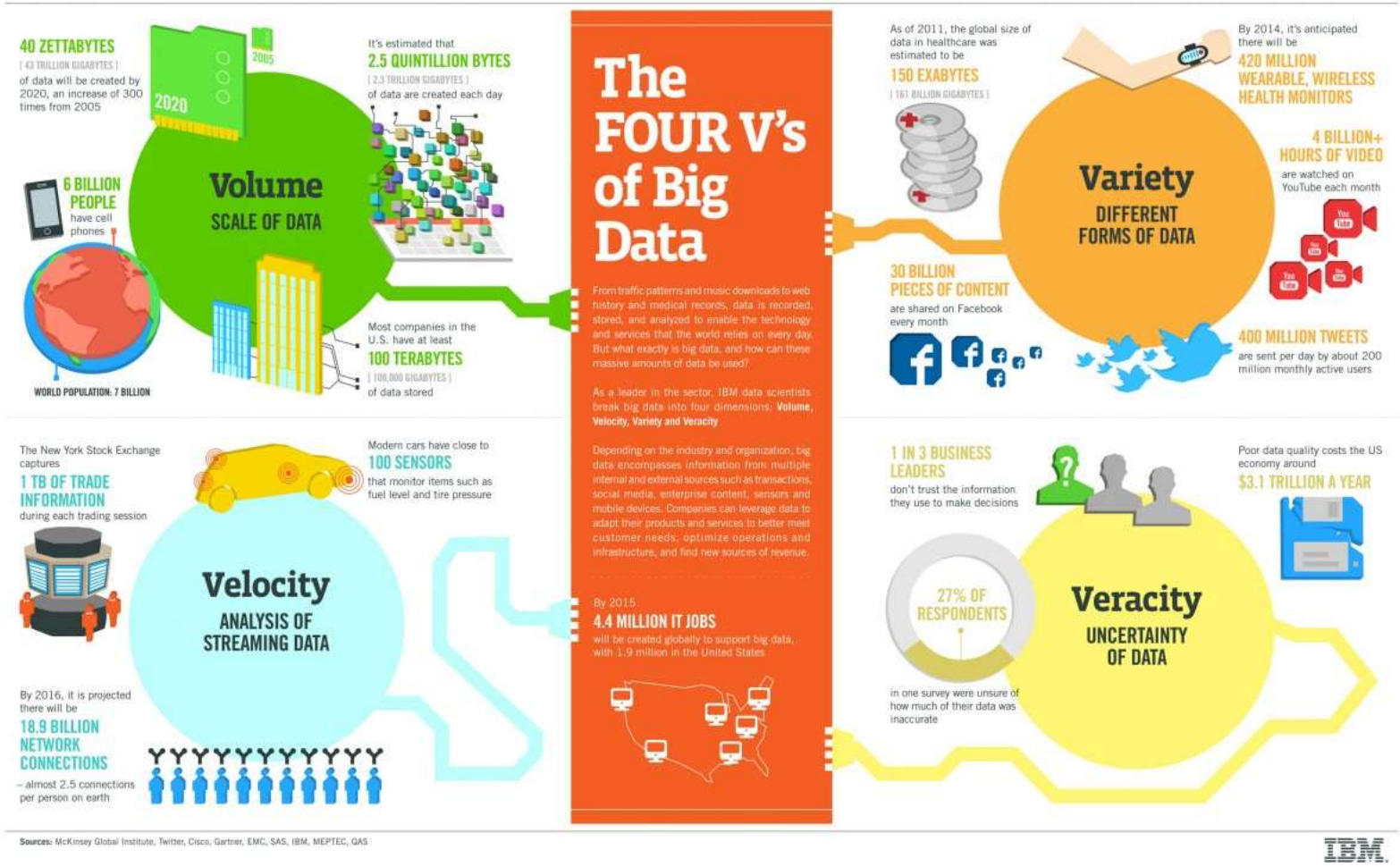
BIG DATA PROCESSING

BIG DATA AND PROCESSING

CONTENT

- Big Data Characteristics
- Data Engineering challenges with Big Data
- Apache Spark
- Introduction the Databricks Community Edition environment
- Data Analysis using Spark SQL

Reference: Bill Chambers and Matei Zaharia (2018), Spark: The Definitive Guide, O'Reilly


40 ZETTABYTES
[43 TRILLION GIGABYTES]
of data will be created by 2020, an increase of 300 times from 2005
2005
2020
6 BILLION PEOPLE
have cell phones
WORLD POPULATION: 7 BILLION
Volume
SCALE OF DATA
It's estimated that
2.5 QUINTILLION BYTES
[2.3 TRILLION GIGABYTES]
of data are created each day
Most companies in the U.S. have at least
100 TERABYTES
[100,000 GIGABYTES]
of data stored
The FOUR V's of Big Data
From traffic patterns and music downloads to web history and medical records, data is recorded, stored, and analyzed to enable the technology and services that the world relies on every day. But what exactly is big data, and how can these massive amounts of data be used?
As a leader in the sector, IBM data scientists break big data into four dimensions: Volume, Velocity, Variety and Veracity.
Depending on the industry and organization, big data encompasses information from multiple internal and external sources such as transactions, social media, enterprise content, sensors and mobile devices. Companies can leverage data to adapt their products and services to better meet customer needs, optimize operations and infrastructure, and find new sources of revenue.
By 2015
4.4 MILLION IT JOBS
will be created globally to support big data, with 1.9 million in the United States
As of 2011, the global size of data in healthcare was estimated to be
150 EXABYTES
[161 BILLION GIGABYTES]
30 BILLION PIECES OF CONTENT
are shared on Facebook every month
Variety
DIFFERENT FORMS OF DATA
By 2014, it's anticipated there will be
420 MILLION WEARABLE, WIRELESS HEALTH MONITORS
4 BILLION+ HOURS OF VIDEO
are watched on YouTube each month
400 MILLION TWEETS
are sent per day by about 200 million monthly active users
The New York Stock Exchange captures
1 TB OF TRADE INFORMATION
during each trading session
Modern cars have close to
100 SENSORS
that monitor items such as fuel level and tire pressure
Velocity
ANALYSIS OF STREAMING DATA
By 2016, it is projected there will be
18.9 BILLION NETWORK CONNECTIONS
– almost 2.5 connections per person on earth
1 IN 3 BUSINESS LEADERS
don't trust the information they use to make decisions
27% OF RESPONDENTS
in one survey were unsure of how much of their data was inaccurate
Veracity
UNCERTAINTY OF DATA
Poor data quality costs the US economy around
$3.1 TRILLION A YEAR
Sources: McKinsey Global Institute, Twitter, Cisco, Gartner, EMC, SAS, IBM, MEPTEC, QAS
IBM

BIG DATA CHARACTERISTICS

- **Volume**
  - The size of data sets,
  - frequently larger than terabytes and petabytes
- **Velocity**
  - Speed with which data is generated
  - High velocity data is generated with such a pace that it requires distinct (distributed) processing techniques.
- **Variety**
  - Big Data comes from a great variety of sources
  - generally is one out of three types: structured, semi structured and unstructured data
- **Veracity**
  - The quality of the data
  - High veracity data has many records that are valuable to analyze

VOLUME CHALLENGES

- Resources Requirement
- Need for Scalability in Design
- Maintaining Latency

SPEED CHALLENGES

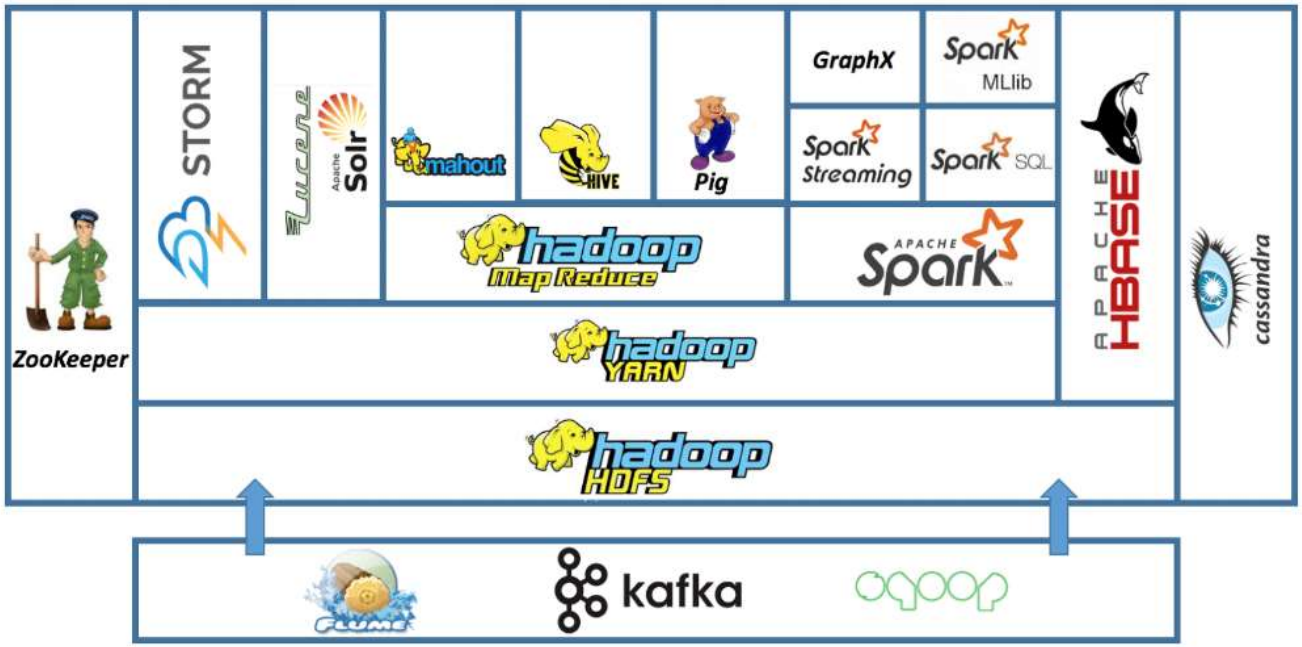
- Real-time event data handling
- Need for speed
- Handling lags

VARIETY CHALLENGES

- Text, audio, video and images
- More resources needed
- Serving at low latency

VARIETY CHALLENGES

- Spikes in load
- Decoupling need with buffering zones
- Maintaining latency



BIG DATA PLATFORM:

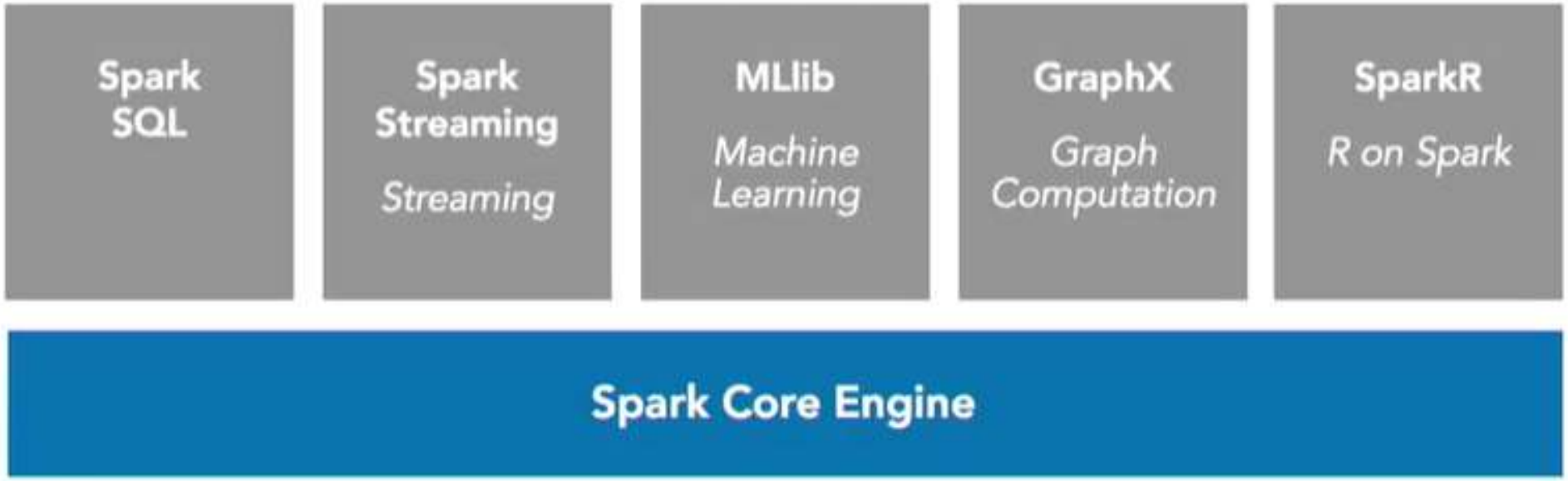
HADOOP

APACHE SPARK

Managing and coordinating the execution of tasks on **data across a cluster of computers**.

- Speed
- Ease of Use
- Generality
- Platform Agnostics

SPARK COMPONETS



SPARK BAISC ARCHITECTURE: SPARK APPLICATION

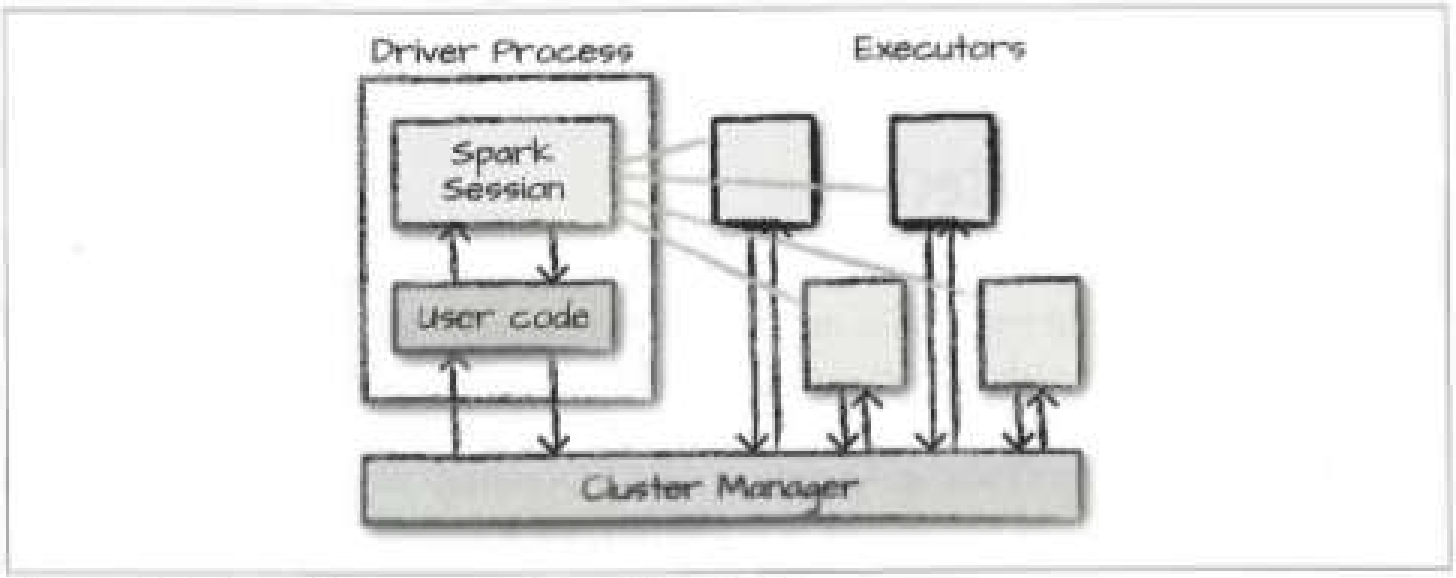


Figure 2-1. The architecture of a Spark Application

Source: Spark: The Definitive Guide, Bill Chambers and Matei Zaharia

SPARK DATA FRAME & PARTITION

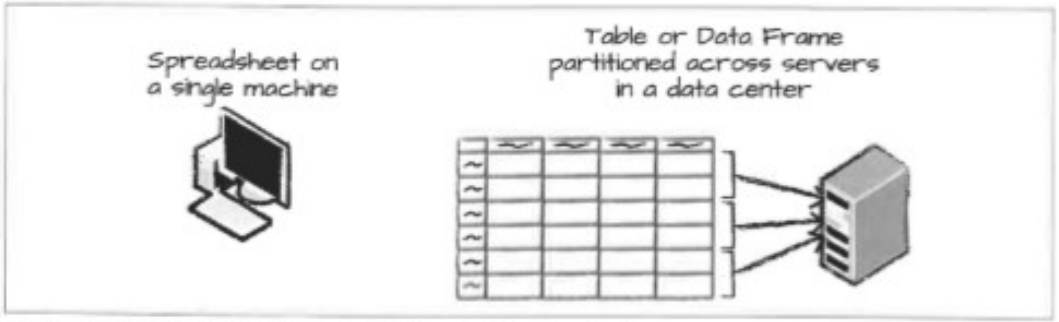


Figure 2-3. Distributed versus single-machine analysis

Source: Spark: The Definitive Guide, Bill Chambers and Matei Zaharia

- DataFrame (Immutable)
- Schema
- Partition

SPARK DATAFRAME VS PANDAS DATAFRAME

```python
import pandas as pd

d = {'weights' :[50,70.5,85.3,43.1],
     'heights' : [1.54,1.73,1.82,1.6]}

pandasDF = pd.DataFrame(d)
print(pandasDF)

# Pandas DataFrame is immutable
pandasDF['BMI'] = pandasDF['heights'] / (pandasDF['weights'] * pandasDF['weights']) #
print(pandasDF)
```

```
   weights  heights
0     50.0     1.54
1     70.5     1.73
2     85.3     1.82
3     43.1     1.60
   weights  heights       BMI
0     50.0     1.54  0.000616
1     70.5     1.73  0.000348
2     85.3     1.82  0.000250
3     43.1     1.60  0.000861
```

Command took 0.04 seconds -- by leong_fong_sow@sp.edu.sg at 12/13/2019, 4:03:53 PM on My Cluster

SPARK TRANSFORMATION & ACTION

Transformation

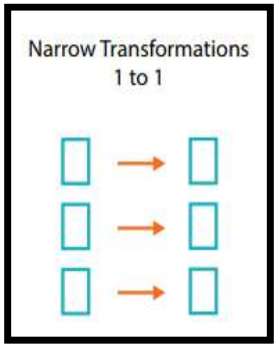
- Tell Spark how you would like to modify the data to do what you what.
- To build up our logical transformation plan
- Spark will not act on transformation until we call an action.

Action

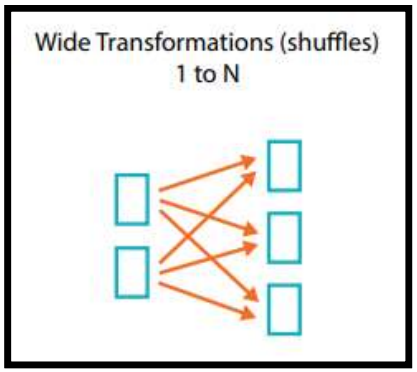
- Instruct Spark to compute a result from a series of transform.

SPARK TRANSFORMATION & ACTION

- Narrow Transformation
  - Each input partition will contribute to only one output partition



- Wide Transformation (Shuffle)
  - input partitions contributing to many output partitions.



```
1  flightData2015 = spark.read.option("inferSchema", "true").option("header", "true")\
2                    .csv("dbfs:/FileStore/tables/flight-data/csv/2015_summary-ebaee.csv")
3
4  flightData2015.sort("count").explain()
```

▸ (2) Spark Jobs

▸ flightData2015: pyspark.sql.dataframe.DataFrame = [DEST_COUNTRY_NAME: string, ORIGIN_COUNTRY_NAME: string ... 1 more fields]

```
== Physical Plan ==
Sort [count#627 ASC NULLS FIRST], true, 0
+- Exchange rangepartitioning(count#627 ASC NULLS FIRST, 5), [id=#1357]
   +- *(1) FileScan csv [DEST_COUNTRY_NAME#625,ORIGIN_COUNTRY_NAME#626,count#627] Batched: false, DataFilters:
y-ebaee.csv], PartitionFilters: [], PushedFilters: [], ReadSchema: struct<DEST_COUNTRY_NAME:string,ORIGIN_COUNTI
```

Command took 0.60 seconds -- by leong_fong_sow@sp.edu.sg at 12/13/2019, 5:00:08 PM on My Cluster

Transformation Plan

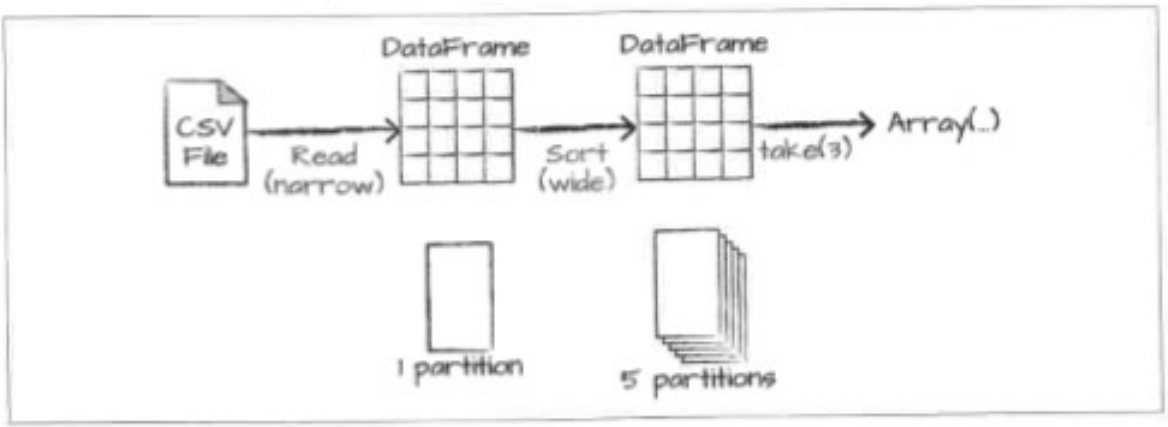


Figure 2-9. The process of logical and physical DataFrame manipulation

Cmd 10

```
%sql

SELECT DEST_COUNTRY_NAME, sum(count) as destination_total from flight_data_2015
   group by DEST_COUNTRY_NAME
   ORDER by sum(count) DESC
   LIMIT 10
```

▾ (1) Spark Jobs

▸ Job 59 View (Stages: 2/2)

DEST_COUNTRY_NAME
United States
Canada
Mexico
United Kingdom
Japan
Germany
Dominican Republic
South Korea
The Bahamas

Command took 0.63 seconds -- by leong_fong_sow@sp.edu.sg at 12/1

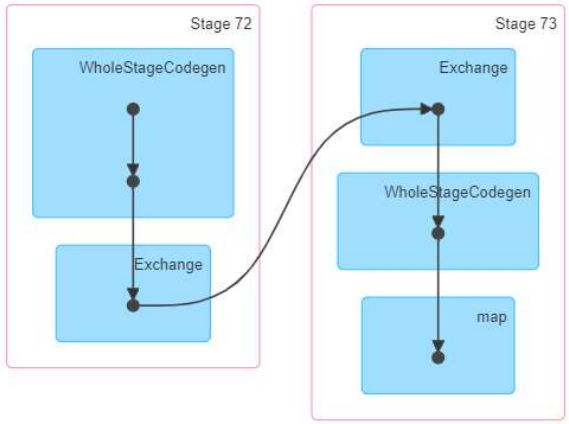
Jobs Stages Storage Environment Executors SQL JDBC/ODBC Server

Details for Job 59

Status: SUCCEEDED
Associated SQL Query: 66
Job Group: 5129141516862806075_8442325453850271231_d29c554c78c748818dcbf92a276c05c1
Completed Stages: 2

▸ Event Timeline

▾DAG Visualization



▾Completed Stages (2)

SPARK SQL – LAZY EVALUATION

- Execute transformation statements only when there is an action executed on the resulting RDDs.

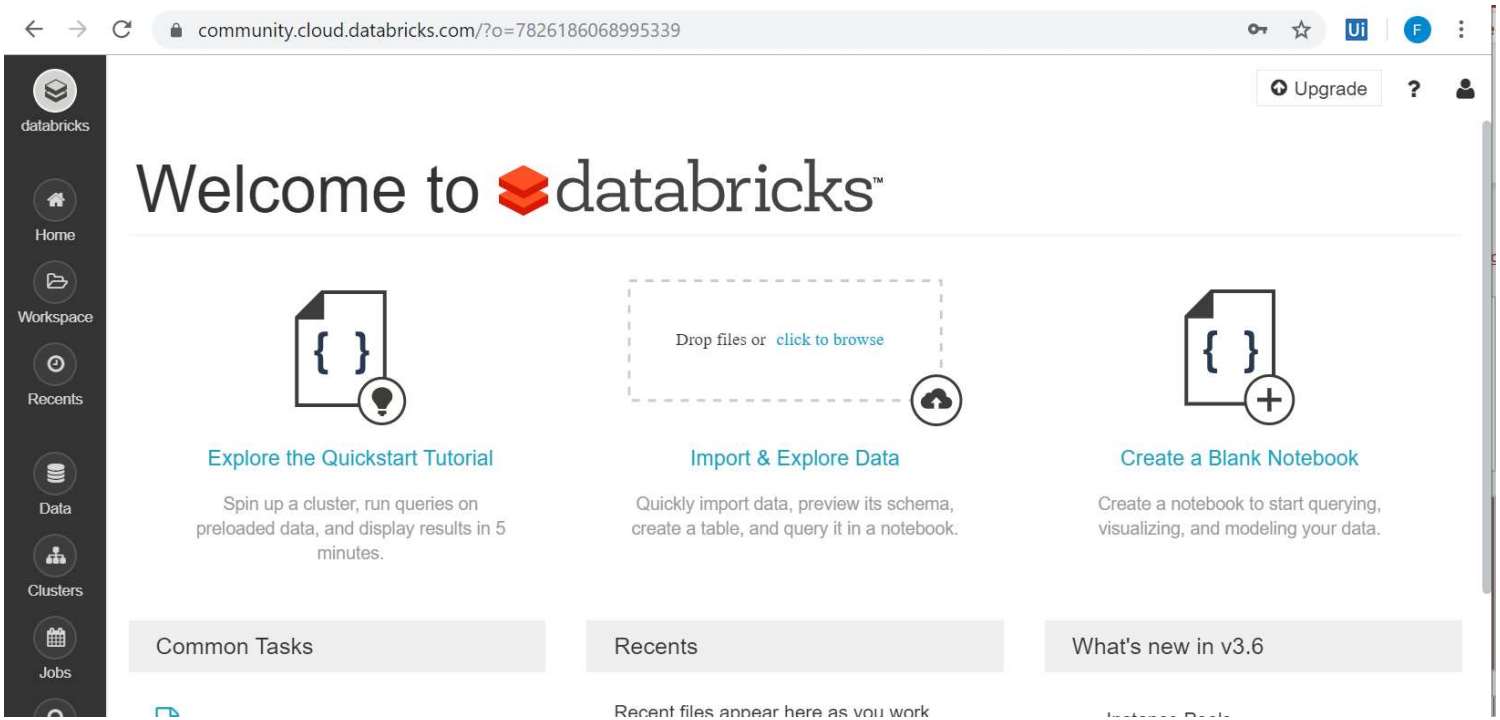
- Whenever Spark executes a batch, it comes up with an execution plan that optimizes full resources and memory based on the statements it has to execute.

OVERVIEW OF DATABRICKS COMMUNITY EDITION

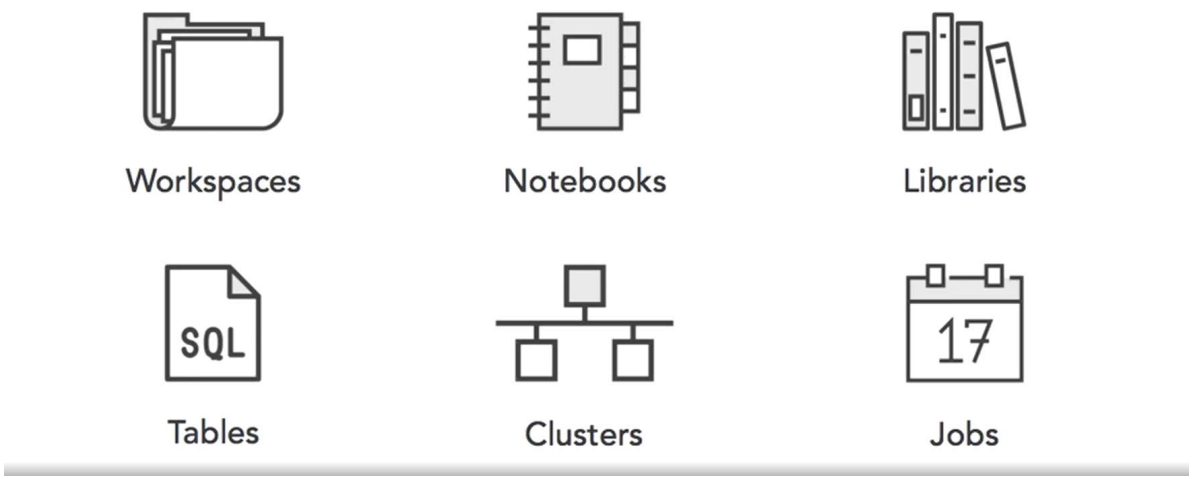
Implementation of Spark to help reduce complexity of setup and operation

100% open-source platform for distributed computing

DATABRICKS COMMUNITY EDITION



OVERVIEW OF DATABRICKS COMMUNITY EDITION



```
1  %timeit data = list(range(0,100))
```

```
2.33 µs ± 88.3 ns per loop (mean ± std. dev. of 7 runs,
```

```
1  data = list(range(0,10000))
2
3  %timeit print(data)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending outpu
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

JUPITER NOTEBOOK AT LOCAL MACHINE

Cmd 4

```
%timeit data = list(range(0,100))
```

```
1.32 µs ± 5.63 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

Command took 10.81 seconds -- by leong_fong_sow@sp.edu.sg at 12/10/2019, 9:11:11 AM on My Cluster

Cmd 5

```
# So far we've done just basic Python, now let's use Spark
# Start by using 'sc' to tell Spark we want to use the SparkContext
# Then we use parallelize() to create a Dataset and spread it across
# the cluster partitions

# Now let's create a simple list with 10000 integers
data = range(1, 10001)

# Then use 'sc' to tell Spark we want to use the SparkContext
# Then we use parallelize() to create a Dataset and spread it across
# the cluster partitions
ds = sc.parallelize(data, 8)
# more info on parallelize here
# help(sc.parallelize)

# show what we have in ds using the collect() action
%timeit print(ds.collect()) # we don't need to use "print" here, but it's better for formatting
```

▸ (8) Spark Jobs

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29
44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70
```

DATABRICKS NOTEBOOK(PYTHON)

SPARK SQL

- With Spark SQL, we can run SQL quires against views or table organization.
- Spark implements subset of ANSI SQL:2003.
- Spark is intended to operate as an online analytic processing (OLAP) database, not as an online transaction processing (OLTP).

SPARK SQL: CREATE TABLE

Create Table

Copy

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] [db_name.]table_name
  [(col_name1 col_type1 [COMMENT col_comment1], ...)]
  USING datasource
  [OPTIONS (key1=val1, key2=val2, ...)]
  [PARTITIONED BY (col_name1, col_name2, ...)]
  [CLUSTERED BY (col_name3, col_name4, ...) INTO num_buckets BUCKETS]
  [LOCATION path]
  [COMMENT table_comment]
  [TBLPROPERTIES (key1=val1, key2=val2, ...)]
  [AS select_statement]
```

SPARK SQL: WINDOW FUNCTIONS

- To carry out some unique aggregations by computing some aggregation on a specific "window" of data.
- The "window" of data is defined using a reference to the current data.
- The window specification is marked by the 'Over (…)' clause.

SPARK SQL: WINDOW FUNCTIONS – EXAMPLE

```
1  SELECT event_datetime, server_id, cpu_utilization,
2         avg(cpu_utilization)
3         OVER (
4           PARTITION BY server_id
5           ORDER BY event_datetime
6           ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) avg_server_util
7    FROM utilization
```

- Partition Specification
- Ordering Specification
- Frame Specification (Rows …)

SPARK SQL: WINDOW FUNCTIONS – EXAMPLE

▸ (1) Spark Jobs

event_datetime	server_id	cpu_utilization	avg_server_util
03/05/2019 08:06:34	112	0.71	0.745
03/05/2019 08:11:34	112	0.78	0.7866666666666666
03/05/2019 08:16:34	112	0.87	0.8233333333333333
03/05/2019 08:21:34	112	0.82	0.77
03/05/2019 08:26:34	112	0.62	0.7799999999999999
03/05/2019 08:31:34	112	0.9	0.8033333333333333
03/05/2019 08:36:34	112	0.89	0.8666666666666667

Showing the first 1000 rows.

THE END