Lab 7: Introduction to Apache Spark for Big Data Processing

Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance

Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since.

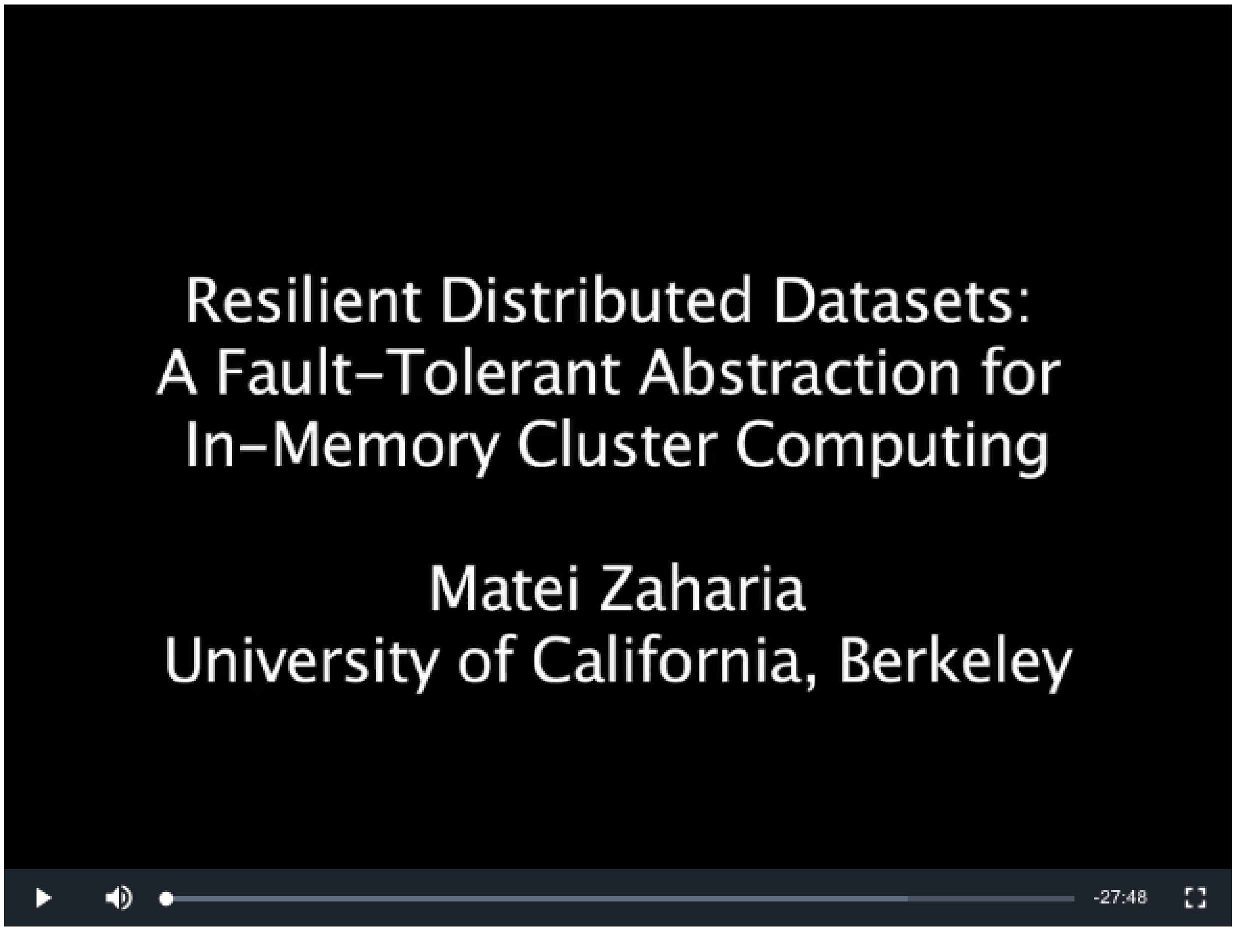
Apache Spark has its architectural foundation in the **resilient distributed dataset (RDD)**, a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. The Dataframe API was released as an abstraction on top of the RDD, followed by the Dataset API. In Spark 1.x, the RDD was the primary application programming interface (API), but as of Spark 2.x use of the Dataset API is encouraged even though the RDD API is not deprecated. The RDD technology still underlies the Dataset API.

Spark and its RDDs were developed in 2012 in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear data flow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.

References:

Website:
https://spark.apache.org/
Spark Quick Start:
http://spark.apache.org/docs/latest/quick-start.html
Resources:
https://databricks.com/spark/developer-resources
Spark Examples:
http://spark.apache.org/examples.html
Original paper:
Zaharia, Matei; Chowdhury, Mosharaf; Franklin, Michael J.; Shenker, Scott; Stoica, Ion. *Spark: Cluster Computing with Working Sets* (PDF). USENIX Workshop on Hot Topics in Cloud Computing (HotCloud).
Comparison of APIs:
Damji, Jules (2016-07-14). "A Tale of Three Apache Spark APIs: RDDs, DataFrames, and Datasets: When to use them and why".

Part 0: Watch the following video.


Resilient Distributed Datasets:
A Fault-Tolerant Abstraction for
In-Memory Cluster Computing

Matei Zaharia
University of California, Berkeley

https://c59951.ssl.cf2.rackcdn.com/nsdi12/zaharia.mp4

Part 1: Install PySpark

```
% conda create env pyspark
% conda activate pyspark
% conda install openjdk
% conda install -c conda-forge findspark
% conda install openjdk
% conda install -c conda-forge findspark
```

Validate:

```
% pyspark
Python 3.9.12 (main, Jun  1 2022, 06:34:44)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
23/04/26 12:15:40 WARN NativeCodeLoader: Unable to load native-hadoop library for your
platform... using builtin-java classes where applicable
23/04/26 12:15:40 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting
port 4041.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.4.0
      /_/
Using Python version 3.9.12 (main, Jun  1 2022 06:34:44)
Spark context Web UI available at http://jays-mbp:4041
Spark context available as 'sc' (master = local[*], app id = local-1682529340889).
SparkSession available as 'spark'.
>>> quit()
```

Tada!

Launch a jupyter and open the first lab notebook:

```
% jupyter notebook
```

Note: If you prefer using a terminal window, you can complete the lab using the terminal. You can save your python script history for your lab report as follows:

```
>>> import readline
>>> readline.write_history_file('x.txt')
```

Credits:
https://sparkbyexamples.com/pyspark/install-pyspark-in-anaconda-jupyter-notebook/