



Introduction to pySpark

Motivation for Spark

- * Data Analytics at Scale
- * Existing distributed solutions (MapReduce, Dryad, etc.)
 - * Pros
 - * Provide primitives for parallel computations
 - * Application programmers do not have to worry about work distribution or fault tolerance
 - * Cons
 - * Operate off of stable storage only
 - * Do not have abstractions for distributed memory
 - * Not suited for iterative and interactive applications
 - * Do not provide a generic API using which complex data pipelines can be built

What is *Spark*?

Apache Spark is an open source big data processing framework built around speed, ease of use, and sophisticated analytics.

Apache Spark™ is a fast and general engine for large-scale data processing

It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an **Apache** project.

What is *Spark*?

- * Spark is a distributed compute engine which provides
 - * Data-parallel computations
 - * Near real-time performance
 - * Fault tolerance
 - * High Scalability
 - * Data locality aware scheduling
 - * Load balancing
- * All of this on commodity hardware
- * Spark provides a unified engine across data **sources**, **workloads** and **environments**

Benefits of Spark

Speed

Engineered from the bottom-up for performance, Spark can be 100x faster than Hadoop for large scale data processing by exploiting in memory computing and other optimizations. Spark is also fast when data is stored on disk, and currently holds the world record for large-scale on-disk

sorting

Benefits of Spark

Speed

Ease of Use

Spark has easy-to-use APIs for operating on large datasets. This includes a collection of over 100 operators for transforming data and familiar data frame APIs for manipulating semi-structured data.

Benefits of Spark

Speed

Ease of Use

A Unified Engine

Spark comes packaged with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows.

Spark Vs Hadoop

They do different things.

Hadoop and Apache Spark are both big-data frameworks, but they don't really serve the same purposes.

Hadoop distributes massive data collections across multiple nodes within a cluster of commodity servers

Spark, on the other hand, is a data-processing tool that operates on those distributed data collections; it doesn't do distributed storage

Spark Vs Hadoop

You can use one without the other.

Hadoop includes not just a storage component, known as the Hadoop Distributed File System, but also a processing component called MapReduce

You can also use Spark without Hadoop. Spark does not come with its own file management system, though, so it needs to be integrated with one -- if not HDFS, then another cloud-based data platform.

Spark was designed for Hadoop, so many agree they're better together.

Spark Vs Hadoop

Spark is speedier

Spark is generally a lot faster than MapReduce because of the way it processes data, While MapReduce operates in steps, Spark operates on the whole data set

The MapReduce workflow looks like this: read data from the cluster, perform an operation, write results to the cluster, read updated data from the cluster, perform next operation, write next results to the cluster, etc

Spark, on the other hand, completes the full data analytics operations in-memory and in near real-time: "Read data from the cluster, perform all of the requisite analytic operations, write results to the cluster, done"

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk

Spark Vs Hadoop

You may not need Spark's speed

MapReduce's processing style can be just fine if your data operations and reporting requirements are mostly static and you can wait for batch-mode processing.

But if you need to do analytics on streaming data, like from sensors on a factory floor, or have applications that require multiple operations, you probably want to go with Spark

Most machine-learning algorithms, for example, require multiple operations. Common applications for Spark include real-time marketing campaigns, online product recommendations, cyber security analytics and machine log monitoring.

Spark Vs Hadoop

Failure recovery: different, but still good

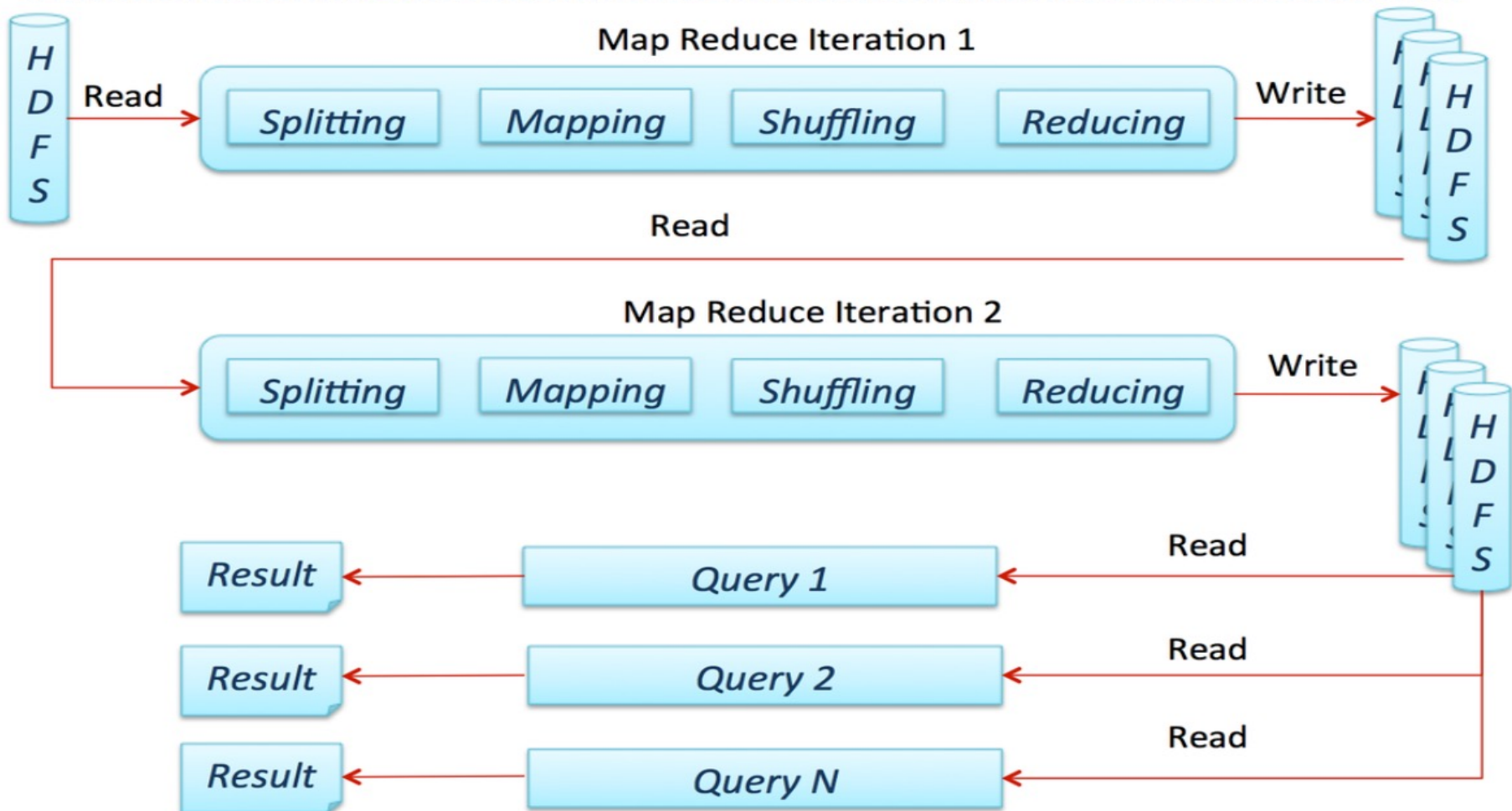
Hadoop is naturally resilient to system faults or failures since data is written to disk after every operation,

but Spark has similar built-in resiliency by virtue of the fact that its data objects are stored in something called resilient distributed datasets distributed across the data cluster.

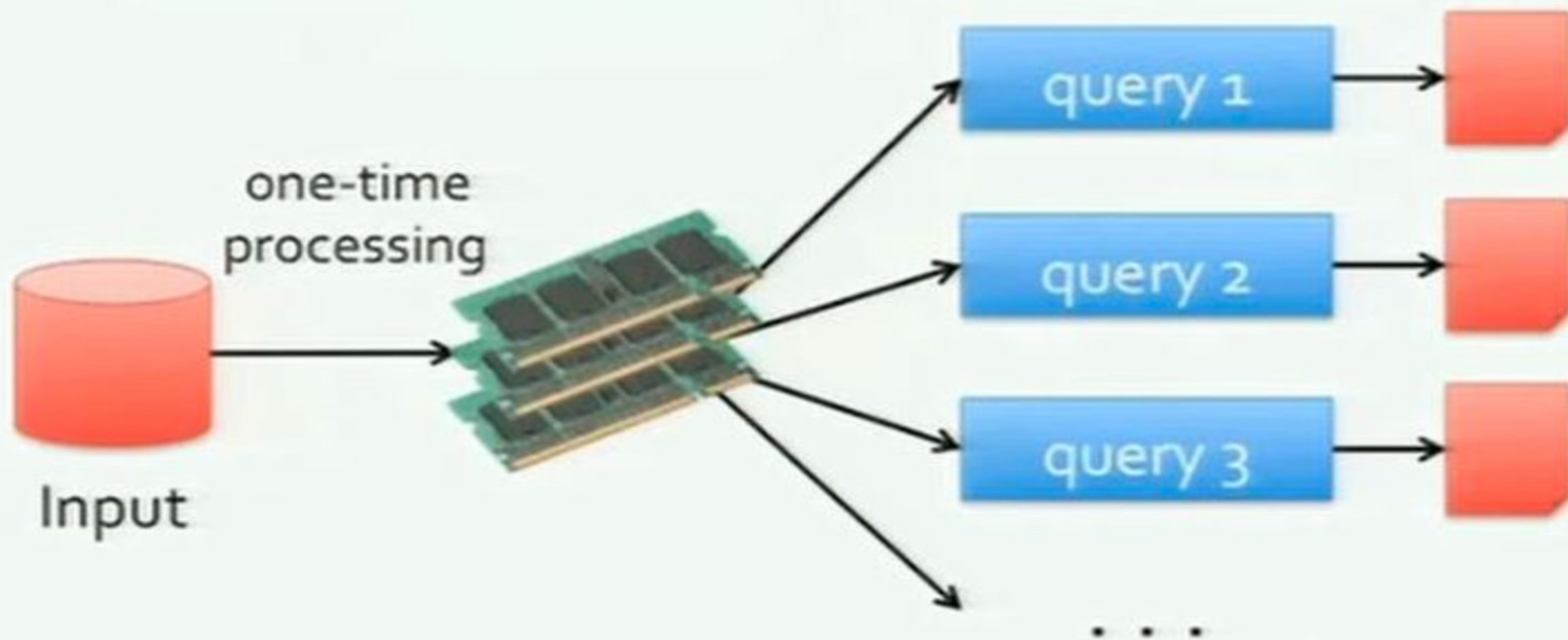
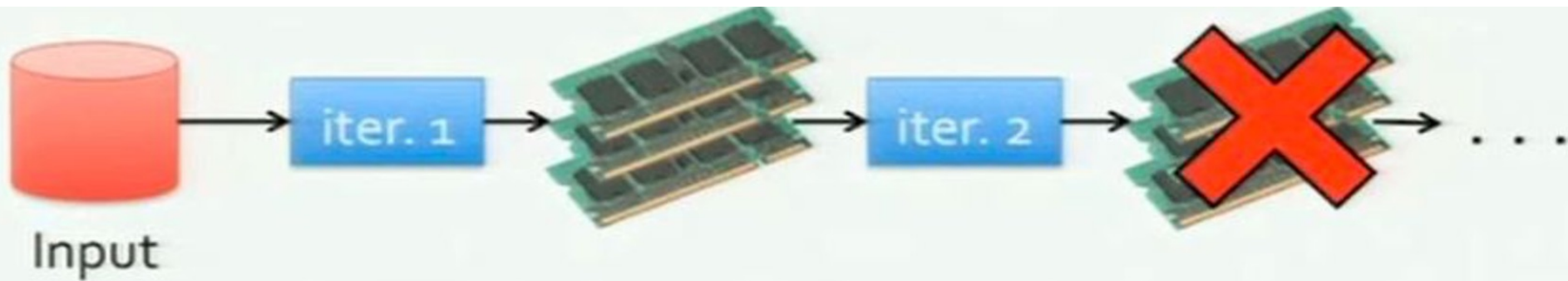
These data objects can be stored in memory or on disks, and RDD provides full recovery from faults or failures

Data Sharing in Map-Reduce

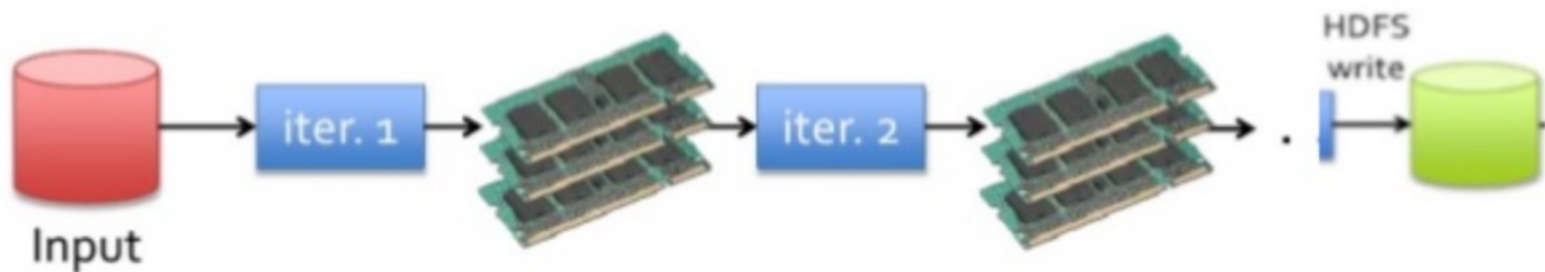
Data Sharing in Map Reduce



Spark Iterative Processing



Spark vs. Hadoop Map Reduce



Spark vs. Hadoop Map Reduce

- * More generic API
 - * Variety of operations – not limited to Map and Reduce
 - * Built-in libraries for SQL, Graph Processing and Machine Learning
 - * Unified API for batch and streaming analytics
- * More efficient runtime
 - * In-memory processing
 - * Long living workers
 - * More efficient scheduling
 - * Highly efficient shuffle
 - * No fixed slot types

Spark Use cases

applications	sensors	web	mobile phones
intrusion detection	malfunction detection	site analytics	network metrics analysis
fraud detection	dynamic process optimisation	recommendations	location based ads
log processing	supply chain planning	sentiment analysis	...

Spark Users and Distributors

Users



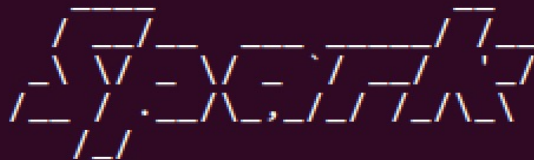
Distributors & Apps



Spark Execution Environment

pyspark terminal

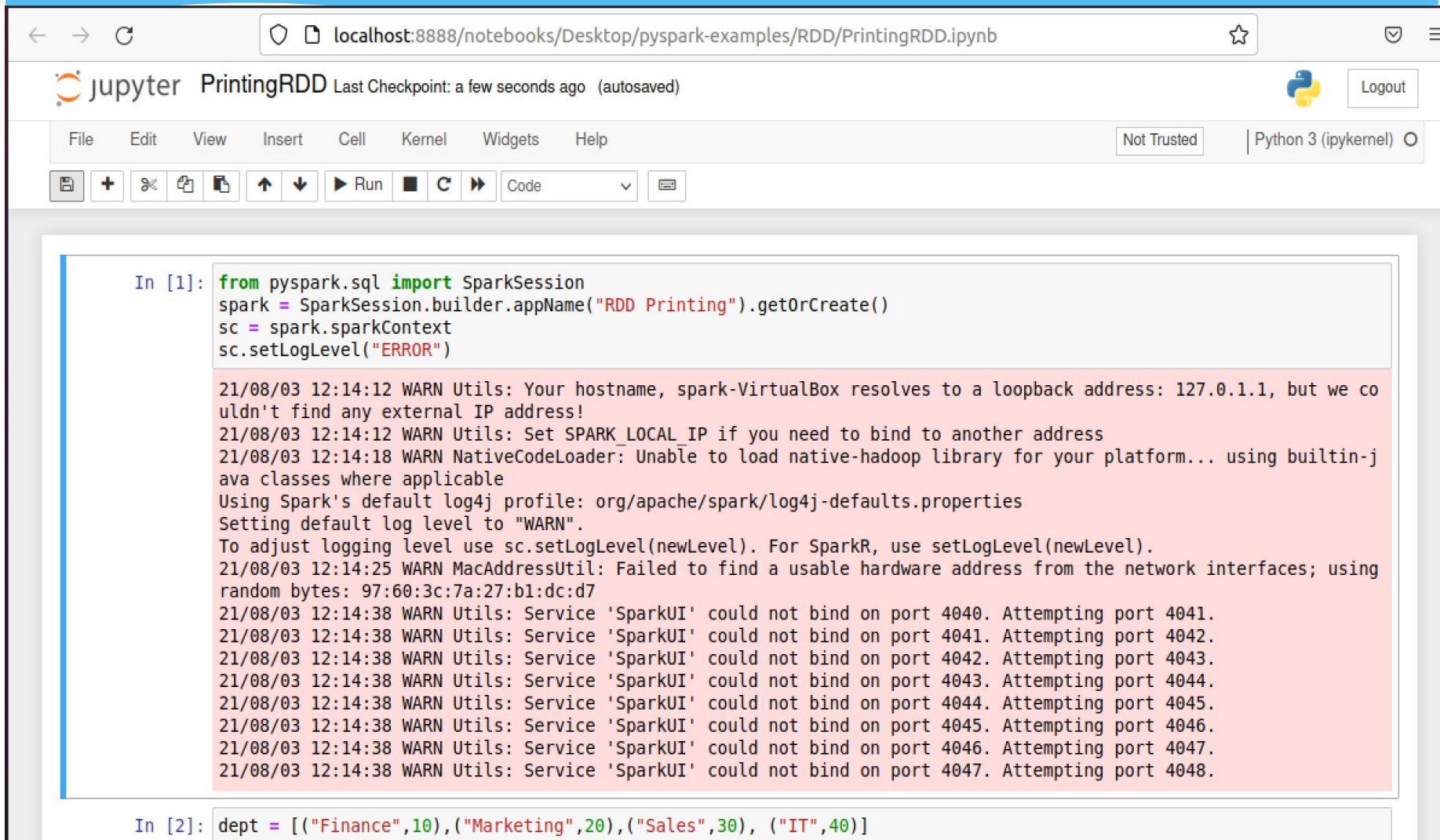
```
[GCC 9.4.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
21/08/03 20:31:30 WARN Utils: Your hostname, spark-VirtualBox resolves to a loop  
back address: 127.0.1.1; using 192.168.231.146 instead (on interface ens33)  
21/08/03 20:31:30 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another  
address  
21/08/03 20:31:32 WARN NativeCodeLoader: Unable to load native-hadoop library fo  
r your platform... using builtin-java classes where applicable  
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLeve  
l(newLevel).  
Welcome to
```



```
version 3.0.3  
  
Using Python version 3.8.10 (default, Jun 2 2021 10:49:15)  
SparkSession available as 'spark'.  
>>> spark  
<pyspark.sql.session.SparkSession object at 0x7f38ebffb370>  
>>>
```

Spark Execution Environment

Jupyter Notebooks



The screenshot displays a Jupyter Notebook interface in a web browser. The address bar shows the URL: `localhost:8888/notebooks/Desktop/pyspark-examples/RDD/PrintingRDD.ipynb`. The notebook title is "PrintingRDD" with a status message "Last Checkpoint: a few seconds ago (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, undo, redo, and running code. The main area shows a code cell with the following content:

```
In [1]: from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("RDD Printing").getOrCreate()
sc = spark.sparkContext
sc.setLogLevel("ERROR")
```

Below the code, the execution output is displayed, showing various warning messages from Spark's Utils and NativeCodeLoader. The messages indicate that the hostname resolves to a loopback address, that the SPARK_LOCAL_IP is not set, and that the native-hadoop library cannot be loaded. It also shows the SparkUI service attempting to bind to various ports (4040 to 4048) and failing.

```
21/08/03 12:14:12 WARN Utils: Your hostname, spark-VirtualBox resolves to a loopback address: 127.0.1.1, but we couldn't find any external IP address!
21/08/03 12:14:12 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
21/08/03 12:14:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/08/03 12:14:25 WARN MacAddressUtil: Failed to find a usable hardware address from the network interfaces; using random bytes: 97:60:3c:7a:27:b1:dc:d7
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4041. Attempting port 4042.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4042. Attempting port 4043.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4043. Attempting port 4044.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4044. Attempting port 4045.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4045. Attempting port 4046.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4046. Attempting port 4047.
21/08/03 12:14:38 WARN Utils: Service 'SparkUI' could not bind on port 4047. Attempting port 4048.
```

At the bottom, the start of a second code cell is visible:

```
In [2]: dept = [("Finance",10),("Marketing",20),("Sales",30), ("IT",40)]
```

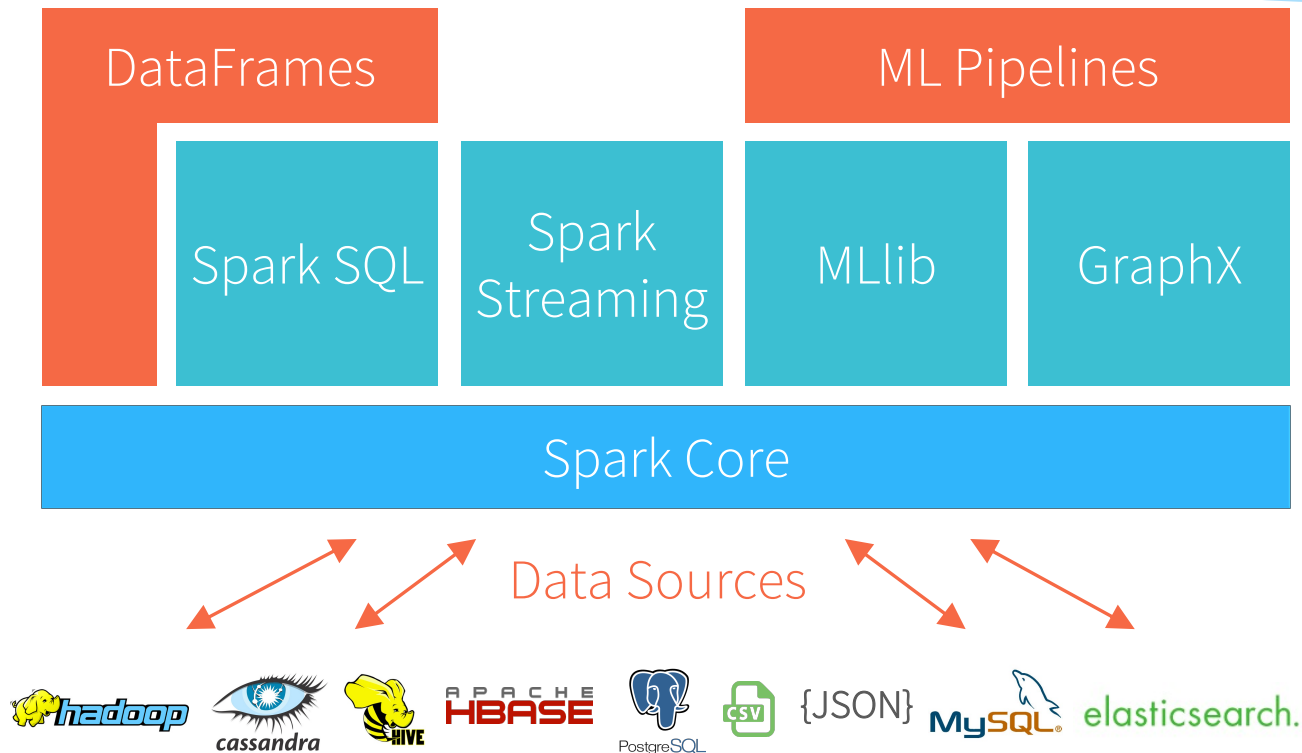

Spark Execution Environment

By script

```
spark@spark-VirtualBox: ~/Desktop
spark@spark-VirtualBox:~/Desktop$ spark-submit sample.py
21/08/03 20:38:48 WARN Utils: Your hostname, spark-VirtualBox resolves to a loop
back address: 127.0.1.1; using 192.168.231.146 instead (on interface ens33)
21/08/03 20:38:48 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another
address
21/08/03 20:38:49 WARN NativeCodeLoader: Unable to load native-hadoop library fo
r your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
21/08/03 20:38:50 INFO SparkContext: Running Spark version 3.0.3
21/08/03 20:38:50 INFO ResourceUtils: =====
=====
21/08/03 20:38:50 INFO ResourceUtils: Resources for spark.driver:

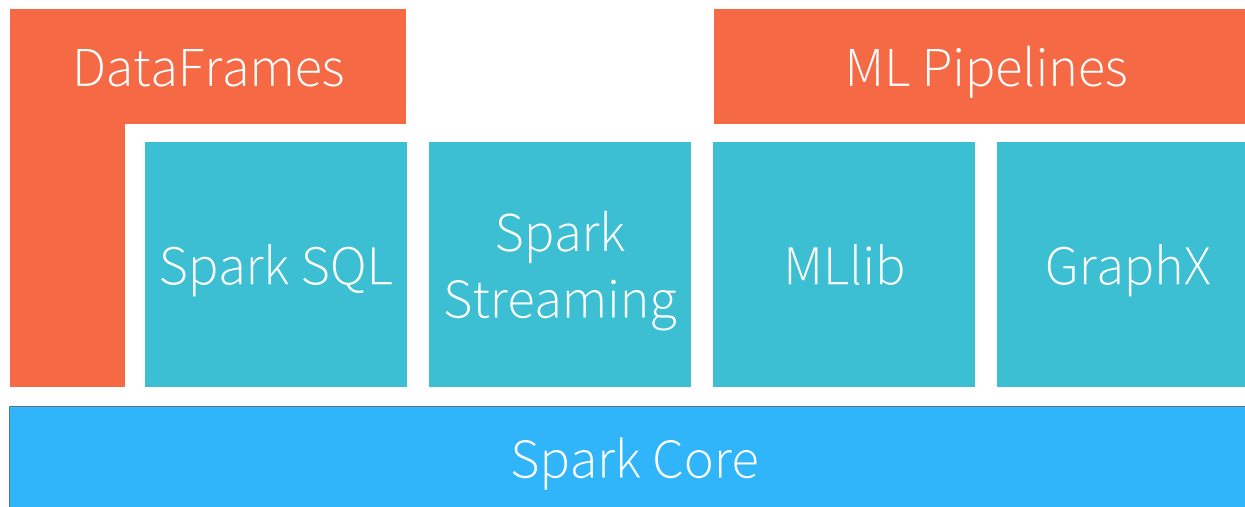
21/08/03 20:38:50 INFO ResourceUtils: =====
=====
21/08/03 20:38:50 INFO SparkContext: Submitted application: MyApp
21/08/03 20:38:50 INFO SecurityManager: Changing view acls to: spark
21/08/03 20:38:50 INFO SecurityManager: Changing modify acls to: spark
21/08/03 20:38:50 INFO SecurityManager: Changing view acls groups to:
21/08/03 20:38:50 INFO SecurityManager: Changing modify acls groups to:
21/08/03 20:38:50 INFO SecurityManager: SecurityManager: authentication disabled
; ui acls disabled; users with view permissions: Set(spark); groups with view p
ermissions: Set(); users with modify permissions: Set(spark); groups with modif
y permissions: Set()
```

Spark – Eco system



Spark - Eco system

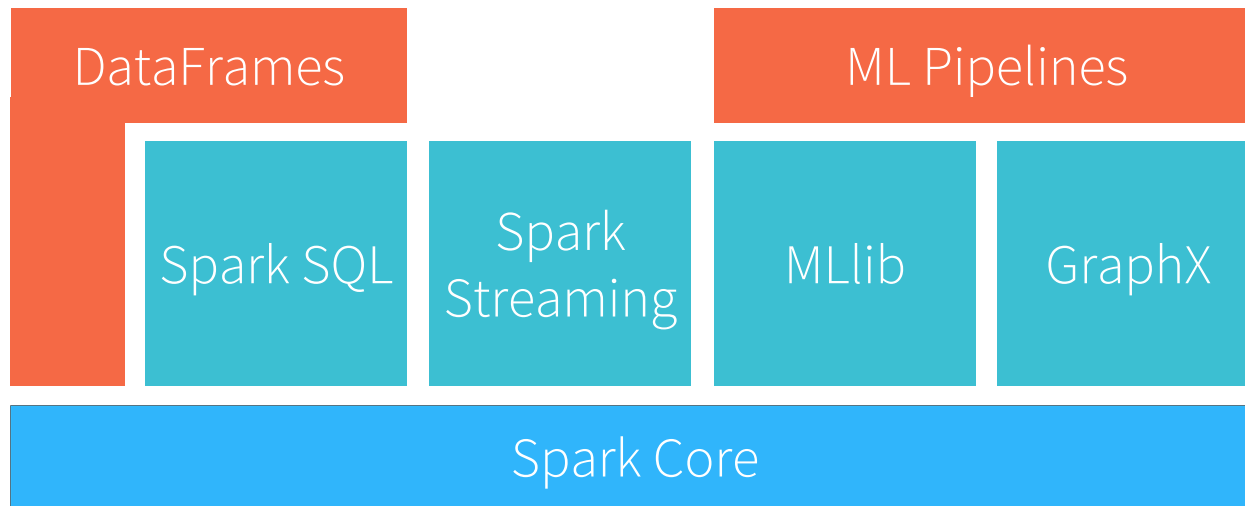
Spark Packages



Data Sources



Spark - Eco system



100TB Daytona Sort Competition 2014

Spark sorted the same data **3X faster** using **10X fewer machines** than Hadoop MapReduce in 2013.

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min

More info:

<http://sortbenchmark.org>

<http://databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html>

Spark Resources

- * <http://spark.apache.org>
 - * OSS Spark release information, downloads, documentation, etc.
- * <https://databricks.com/blog>
 - * Spark committer blogs
- * <https://sparkhub.databricks.com/>
 - * Spark community site
- * <http://spark-packages.org/>
 - * Community index of packages for Spark
- * <http://blog.cloudera.com/blog/category/spark/>
 - * Cloudera's blogs on Spark