CSC 261/461 – Database Systems
Spark!

Spring 2018

SPARK

What is SPARK?

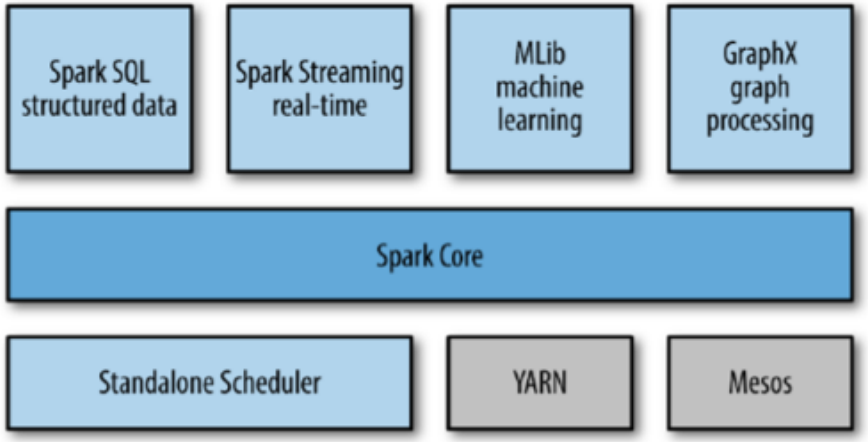
- Apache Spark™ is a cluster computing platform designed to be fast and general purpose.
- Spark extends the popular MapReduce model to efficiently support more types of computations, including interactive queries and machine learning.
- Spark is also more efficient than MapReduce for complex applications running on disk.
- Spark makes it easy and inexpensive to combine different processing types. In addition, it reduces the management burden of maintaining separate tools.

Why are we using Spark?

- Spark is designed to be highly accessible, offering simple APIs in Python, Java, Scala, and SQL, and rich built-in libraries.
- Spark can run in Hadoop or General Parallel File System (**GPFS**) clusters.

- Also, CIRC is giving us a great interface to access Spark.

Spark Stack



One of the largest advantages of tight integration is the ability to build applications that seamlessly combine different processing models.

For example, in Spark, you can write one application that uses machine learning to classify data in real time as it is ingested from streaming sources. Simultaneously, analysts can query the resulting data, also in real time, via SQL (e.g., to join the data with unstructured logfiles).

In addition, more sophisticated data engineers and data scientists can access the same data via the Python shell for ad hoc analysis.

Spark Core

- Spark Core contains the basic functionality of Spark, including components for:
  - task scheduling,
  - memory management,
  - fault recovery,
  - interacting with storage systems,
- Spark Core is also home to the API that defines resilient distributed datasets (RDDs), which are Spark's main programming abstraction.

Resilient distributed datasets (RDD)

- RDDs represent a collection of items distributed across many compute nodes that can be manipulated in parallel.
- Spark Core provides many APIs for building and manipulating these collections.

Spark Library/Package

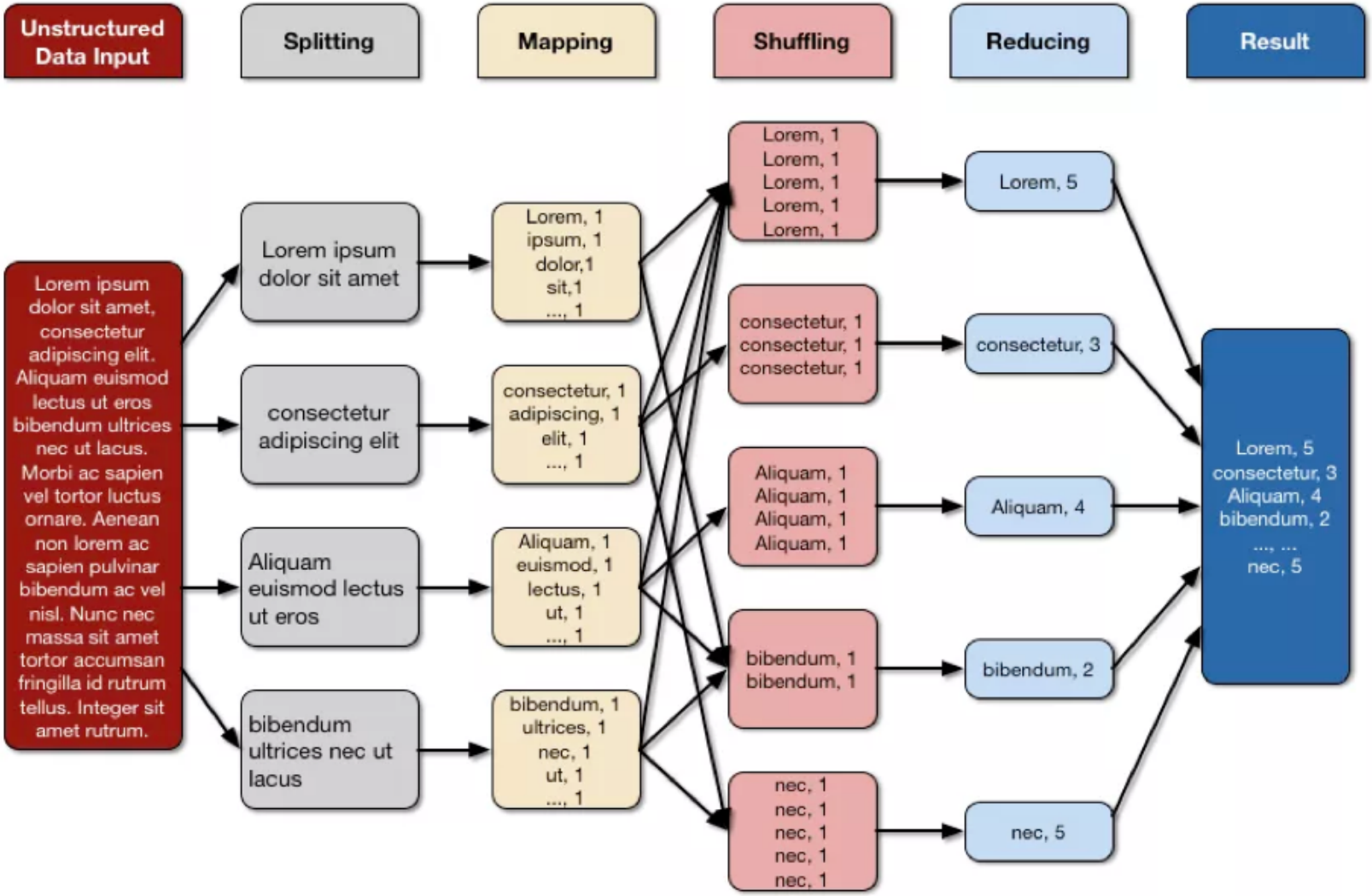
- Spark SQL
  - Spark's package for working with structured data. It allows querying data via SQL
- Spark Streaming
  - Spark component that enables processing of live streams of data
- MLlib
  - A library containing common machine learning (ML) functionality
- GraphX
  - A library for manipulating graphs (e.g., a social network's friend graph)

Cluster Managers

- Spark is designed to efficiently scale up from one to many thousands of compute nodes.
- Spark can run over a variety of cluster managers, including
  - Hadoop YARN,
  - Apache Mesos,
  - Standalone Scheduler (A cluster manager included in Spark)

MAP-REDUCE

MapReduce Data and Process Flow of Word Count



Python Lambda Functions

- Python supports the creation of anonymous functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda".
- This is not exactly the same as lambda in functional programming languages,
- It is a very powerful concept that's often used in conjunction with typical functional concepts like filter(), map() and reduce().

filter(*function*, *iterable*)

- Construct a list from those elements of *iterable* for which *function* returns true.

- `filter(function, iterable)` is equivalent to
- `[item for item in iterable if function(item)]`

map(*function*, *iterable*, ...)

- Apply *function* to every item of *iterable* and return a list of the results

reduce(*function*, *iterable*[, *initializer*])

- Apply `function` of two arguments cumulatively to the items of `iterable`, from left to right, so as to reduce the iterable to a single value.
- For example,

- `reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])`
- calculates
- `(((((1+2)+3)+4)+5)`.

- Note: Different from `reduceByKey()`

Example

```
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>>
>>> print filter(lambda x: x % 3 == 0, foo)
[18, 9, 24, 12, 27]
>>>
>>> print map(lambda x: x * 2 + 10, foo)
[14, 46, 28, 54, 44, 58, 26, 34, 64]
>>>
>>> print reduce(lambda x, y: x + y, foo)
139
```

JUPYTER NOTEBOOK

Website

- Jupyter notebook: jupyter.circ.rochester.edu/
- Log in (NETID credentials)
  - Needs Duo credentials
- 8 CPU 16GB Memory should be good enough.
- Select New -> Python 2 (Spark) Kernel
- You are all set to go.
- (also, you may would like to download `lecture24.ipynb` and `taleOfTwoCities.txt` from the course website.)

Acknowledgement

- Spark slides and material: Jonathan Carroll-Nellenback (CIRC, UofR)

- **Learning Spark** Lightning-Fast Big Data Analysis
  - By Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia
  - Publisher: O'Reilly Media