DISTRIBUTED AND CLOUD COMPUTING

LAB 11: APACHE SPARK

Plan for today

Introduction , set-up and experiment with using Apache Spark

Assignment 3 walkthrough

Introduction to Spark

Started at UC Berkeley in 2009 & became an Apache project in 2013

Utilised by a lot of the worlds largest organisations to processes massive datasets

- NASA relies on Spark to processes communication from spacecrafts in the deep space!

SPARK is an in-memory distributed data processing engine

- Written in Scala -> a functional programming language that runs on the JVM
- Offers API's in various languages -> we will use Python's API, **PySpark**

Why Spark?

- The key limitation of a MapReduce lies in its reliance on the disk
- This increases the latency of operations

Introduction to Spark: RRD's

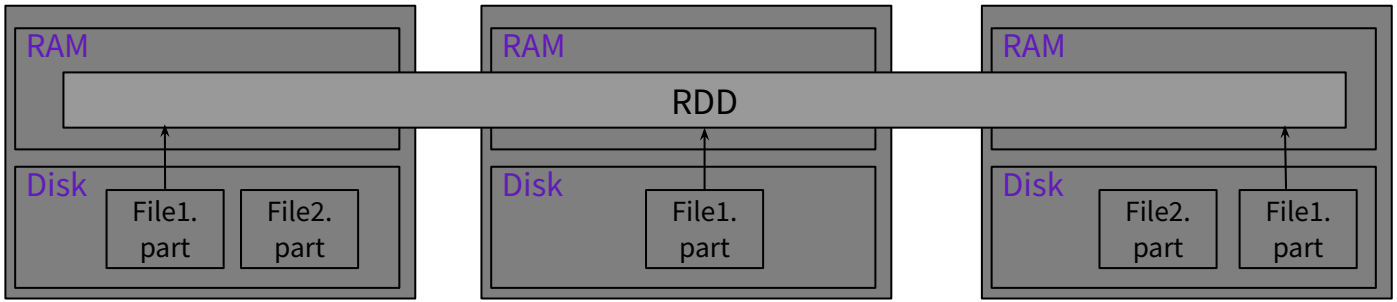
Spark utilises a Master Slave architecture for distributed data processing
Spark introduces the notion of a Resilient Distributed Dataset (RRD)

- **In memory data structure** that is distributed among the cluster

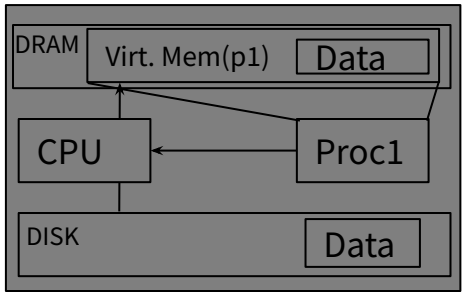
All data processing in Spark is done using RDD's

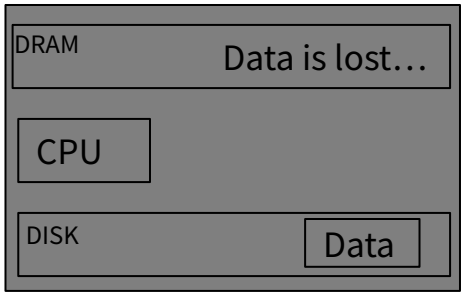
Spark RRD's allow for dramatically higher efficiency that MapReduce

1) **RRD caching / persistent RRD's**
2) **Lazy execution**

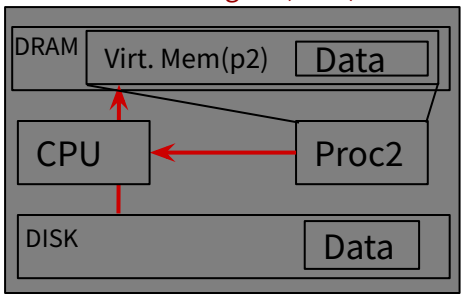


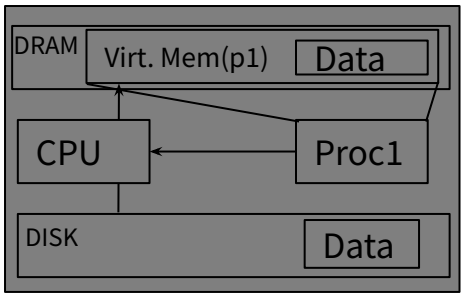
Introduction to Spark: Persistent RDD's

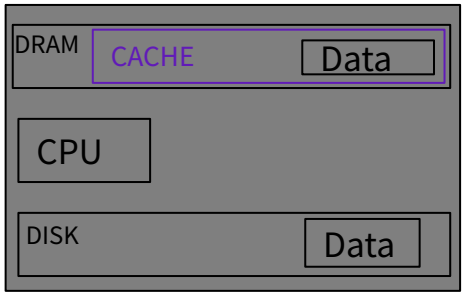




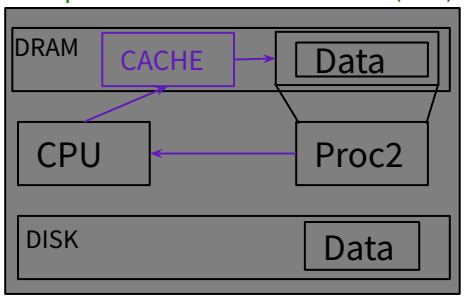
Load from disk again (slow)







RRD persistent RDD from cache (fast)



Introduction to Spark: Laziness

Spark is **LAZY**

- some expressions are **not** evaluated when executed

Two types of **RRD operations** exist:

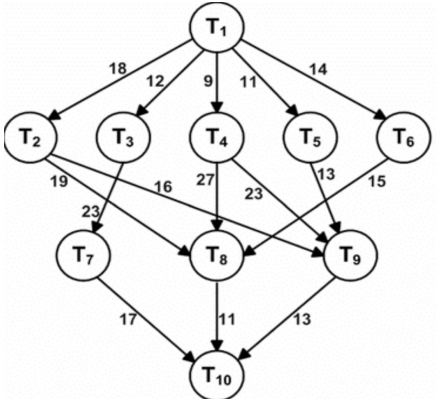
- **Transformations** and **Actions**

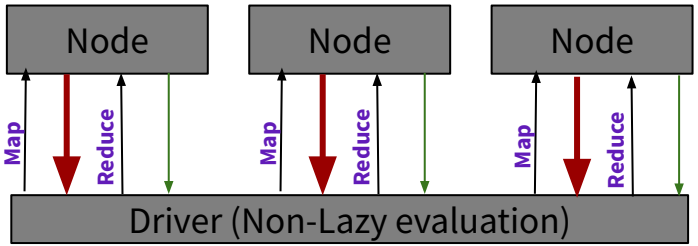
Transformations (lazy):

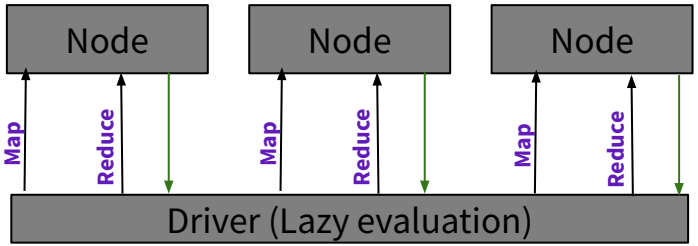
- They **DO NOT** execute their logic immediately
- Instead:
  - reserve memory for produced RRD
  - Add the computation as a node in a **DAG** workload

Actions:

- Execute the DAG workload consisting of all unevaluated transformations so far!







Introduction to Spark: Getting Data

Spark does NOT come with a file system or any specification for accessing data
How data makes it to the nodes is flexible and up the user:

- Client can upload the data along with the job
- Nodes can download the data from a database or the cloud

Common approach: Run Spark in an HDFS cluster

- Data is already stored in a distributed manner by the nodes

Introduction to Spark: Spark's components

Spark Core

- Contains the definition of and transformations and actions for RDD's

Spark SQL

- Defines a DataFrame structure based on an RRD
- This allows the use SQL

Introduction to Spark: Scheduling Spark Jobs

By default:

- Jobs run in FIFO order and attempt to use all available nodes in the cluster
- In standalone mode Spark creates as many workers are available CPU cores

Spark can run on a **Mesos** managed cluster and use the Mesos' built in scheduler

- Mesos: Apaches cluster manager which is basically a distributed version of the linux kernel

Spark jobs can be submitted and managed by **YARN**

- This is common as Spark is often used in clusters running HDFS and YARN

Spark environments can also be managed by **Kubernetes**

- Peter tell you more about Kubernetes in the final module of the lab starting next week.

Task 1: Installing Spark (PySpark)

Installing PySpark is very simple
Just like installing any other python library

```
$ pip install pyspark
$ pyspark
```

It is recommended to utilise a virtual environment linke Venv or Miniconda

Venv: Same commands

Conda: Activate the environment and use

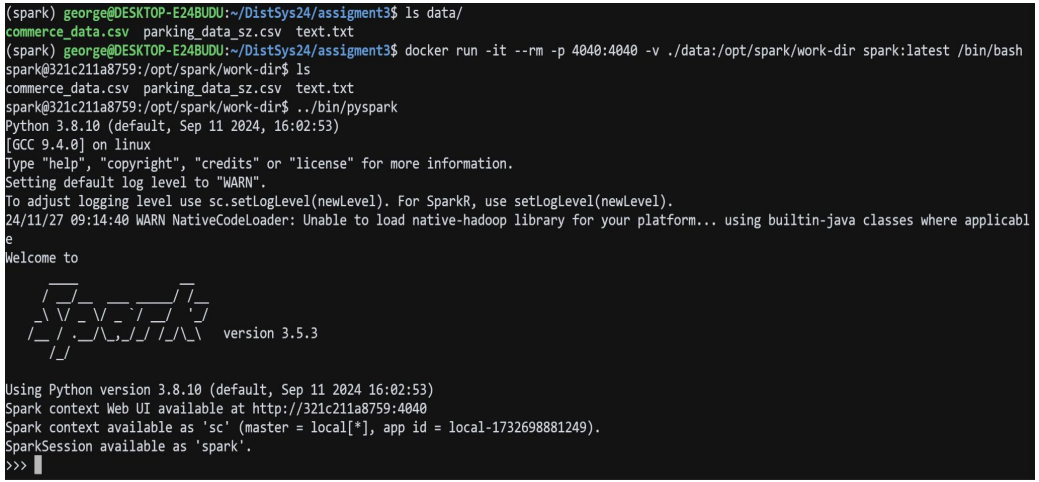
```
$ conda install -c conda-forge pyspark
```

If the local installation is not working you can pull the official Spark docker image from Apache by

```
$ docker pull spark
```

Create a directory called “data” with and put the input data there (.txt, .csv) etc.. and use the commands in the picture

Now let's run some basic commands together



-it	Interactive session
–rm	deletes container after exiting
-p 4040:4040	Links container port 4040 to local port 4040 to allow accessing SparkUI
-v ./data:/opt/spark/work-dir	Mounts the ./data to the container at opt/spark/working-dir

Task 2: PySpark for distributed data processing

TASK DESCRIPTION:

Using the file ‘commerce_data_cleaned.csv’ and PySpark write code to achieve the following:

a) The invoice with the the most unique items
b) The invoice with the most total items
c) The incoide with the highest total price

CSV column labels

```
InvoiceNo,StockCode,Description,Quantity,InvoiceDate,UnitPrice,CustomerID,Country
536365,85123A,WHITE HANGING HEART T-LIGHT HOLDER,6,12/1/2010 8:26,2.55,17850,United Kingdom
536365,71053,WHITE METAL LANTERN,6,12/1/2010 8:26,3.39,17850,United Kingdom
```