Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

intel

AI and Predictive Analytics in Data-Center Environments

Distributed Computing using Spark

Executions on Spark

Josep Ll. Berral @BSC

Intel Academic Education Mindshare Initiative for AI

Presentation

We have the environment configured now!

- We can now submit a job…
- … or open an interactive session
- … to a Spark cluster

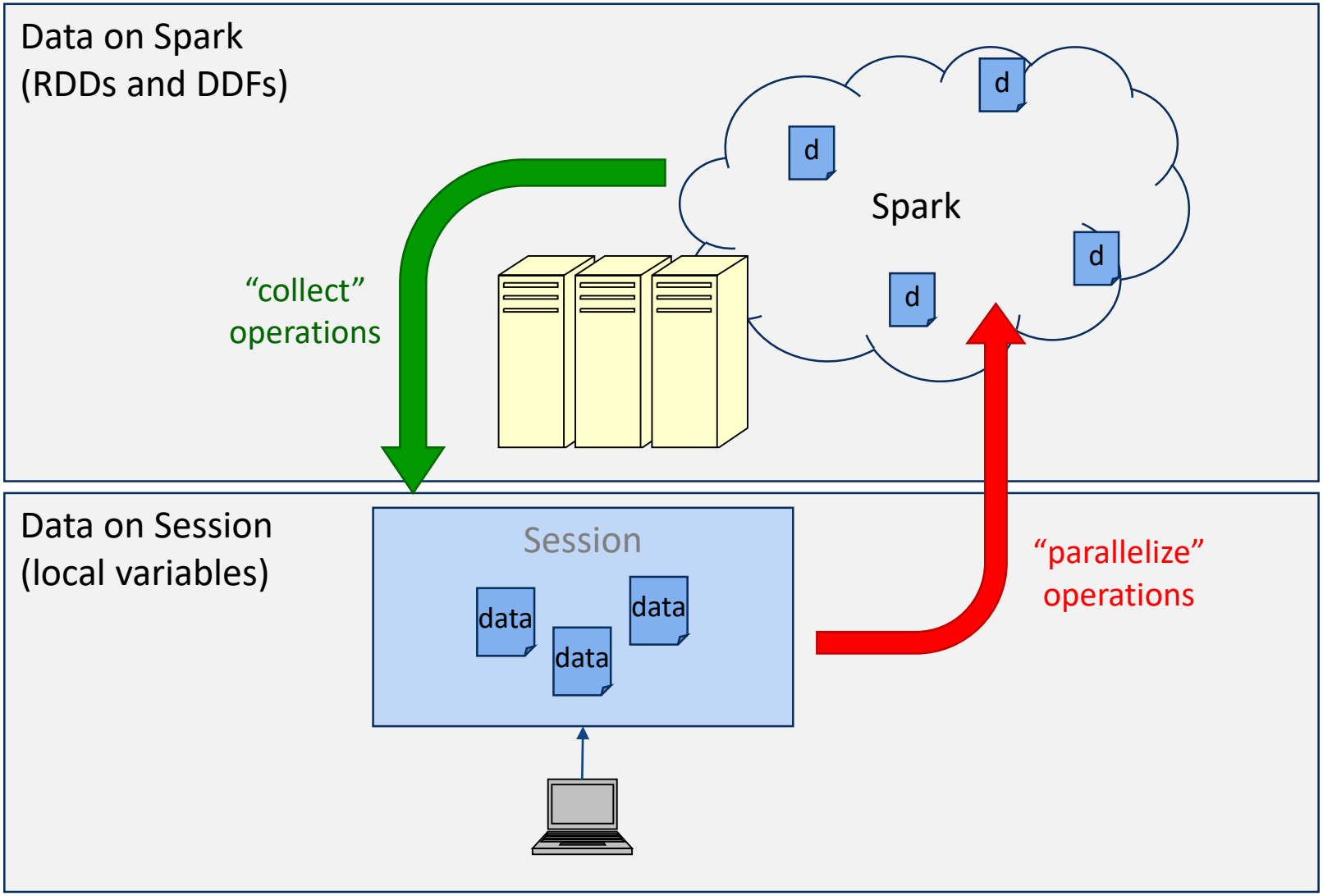
Introduction

- Running stuff on Spark
  - Submitting commands
  - Building the pipeline
  - Knowing where is my data and processes

Executing on Spark

- Two places where data can be:
  - “Uploaded” in Spark (Distributed)
    - We **parallelize** data, or read data “using Spark”
    - The data is now distributed along to be used
  - “Downloaded” on the Session (not distributed)
    - We **collect** data from Spark
    - The data is now on our session

Executing on Spark



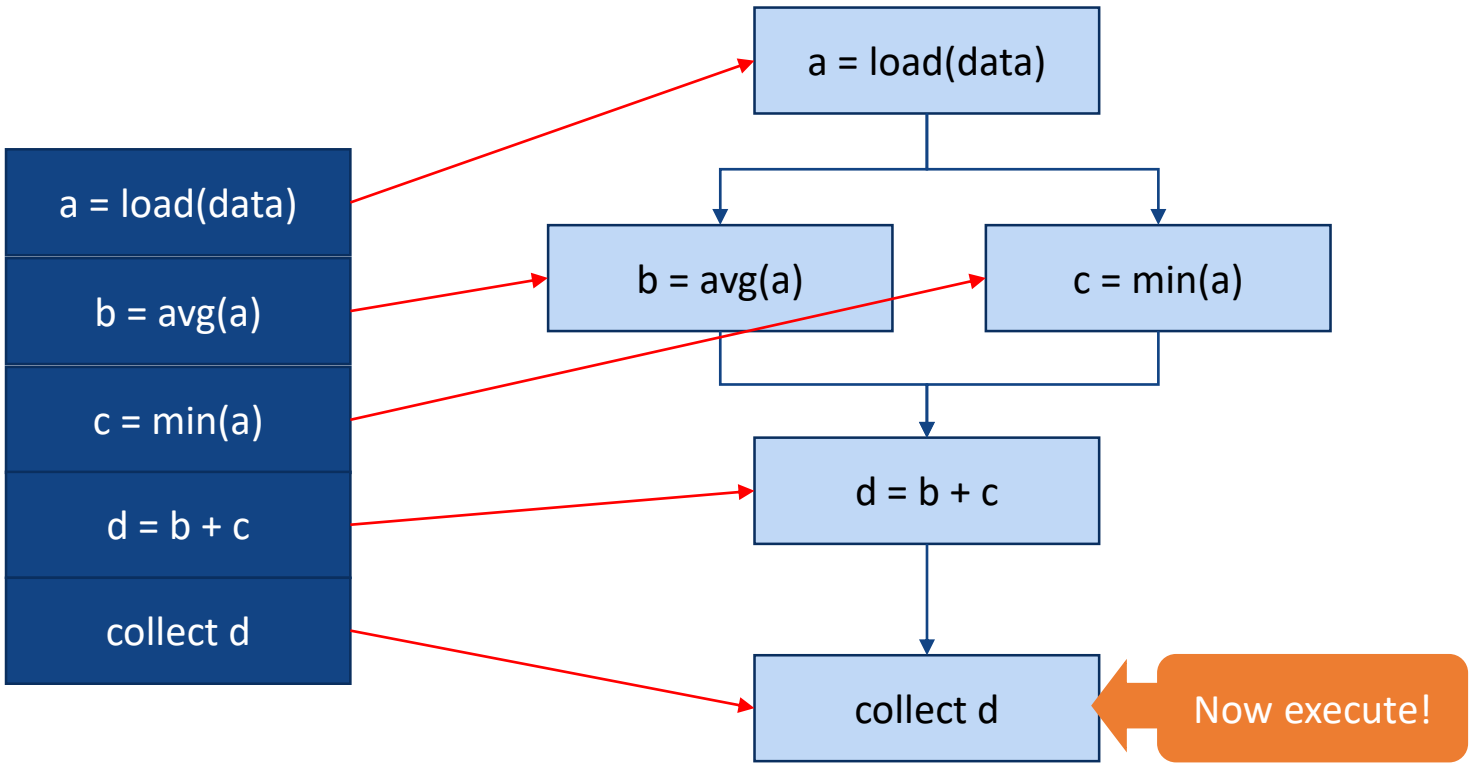
Execution Tree

- Spark Execution
  - Commands sent to Spark are put in an “Execution Tree”
  - The (relevant part of the) tree is executed when:
    - Results are collected (to local)
    - Results are saved (in local or distributed FS)
    - Results are passed to other processes (e.g. Spark Stream)

“Results are Consumed”

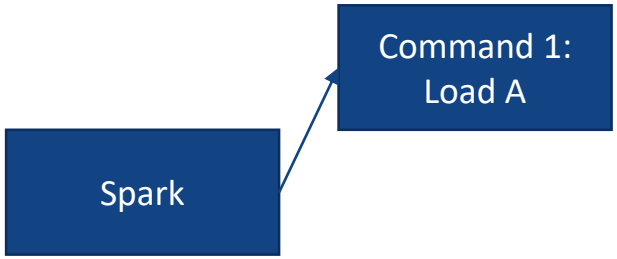
  - Non-syntactic errors are not detected until the computing is done
    - E.g.: a file to read is missing
    - E.g.: the content of a variable is not the correct type

Execution Tree



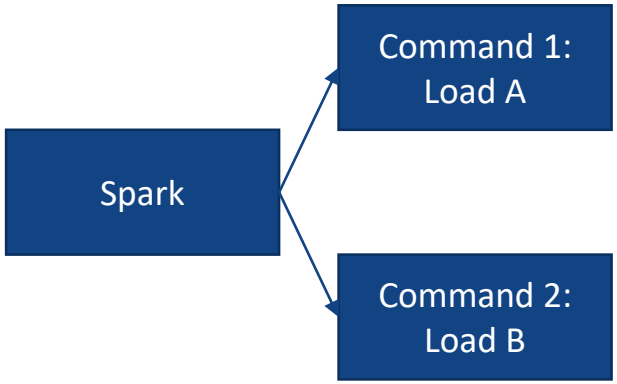
Execution Tree

- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]



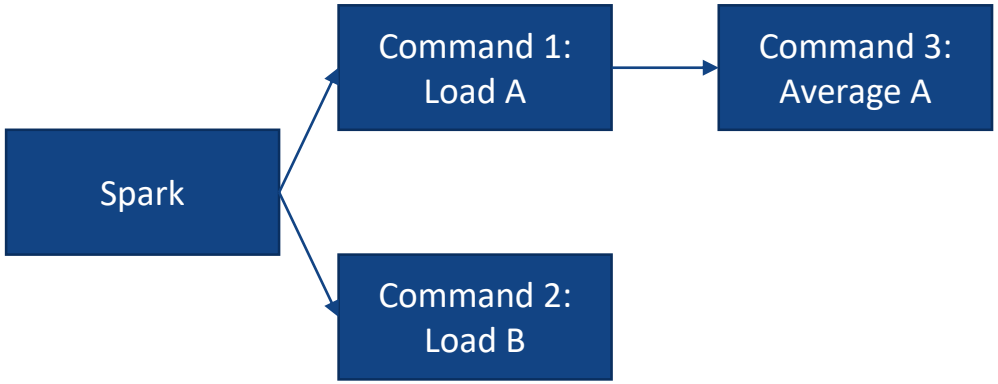
Execution Tree

- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]
  - Command 2: “Load data B on Spark” → OK [Not Executed]



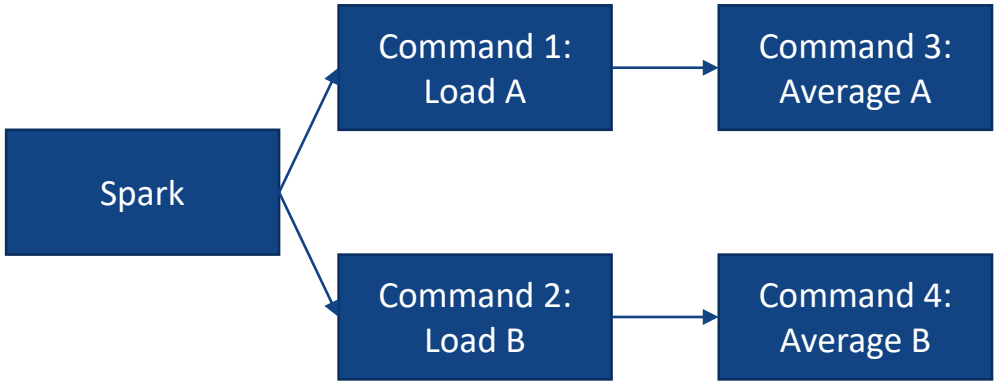
Execution Tree

- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]
  - Command 2: “Load data B on Spark” → OK [Not Executed]
  - Command 3: “Compute average on A” → OK [Not Executed]



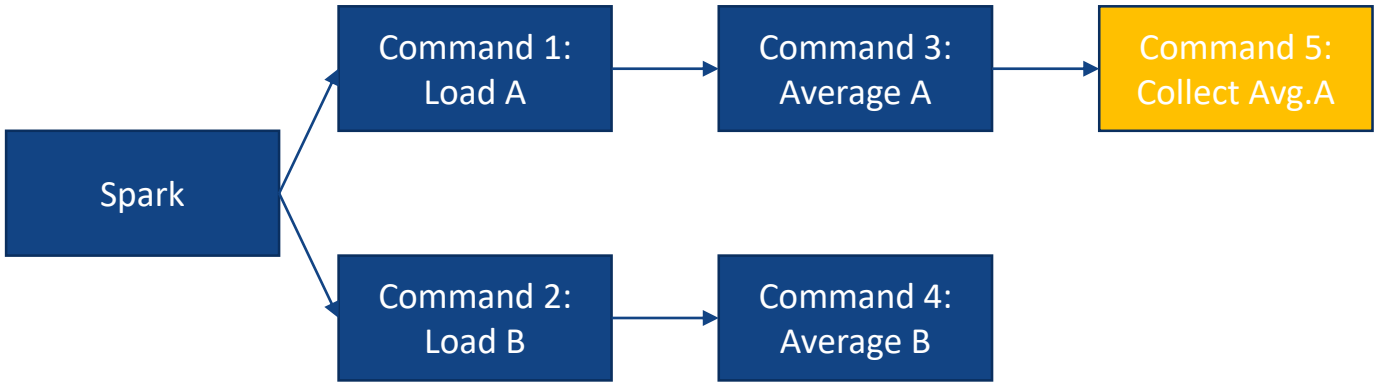
Execution Tree

- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]
  - Command 2: “Load data B on Spark” → OK [Not Executed]
  - Command 3: “Compute average on A” → OK [Not Executed]
  - Command 4: “Compute average on B” → OK [Not Executed]



Execution Tree

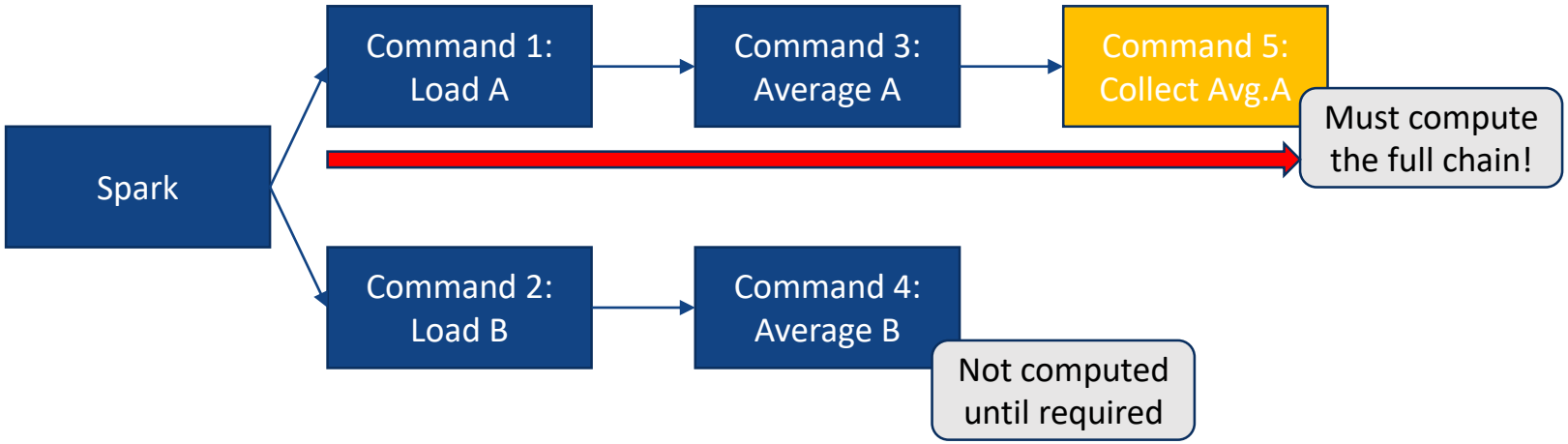
- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]
  - Command 2: “Load data B on Spark” → OK [Not Executed]
  - Command 3: “Compute average on A” → OK [Not Executed]
  - Command 4: “Compute average on B” → OK [Not Executed]
  - Command 5: “**Collect** the computed average of A” → OK...



Execution Tree

- Another example:
  - Command 1: “Load data A on Spark” → OK [Not Executed]
  - Command 2: “Load data B on Spark” → OK [Not Executed]
  - Command 3: “Compute average on A” → OK [Not Executed]
  - Command 4: “Compute average on B” → OK [Not Executed]
  - Command 5: “**Collect** the computed average of A” → OK...

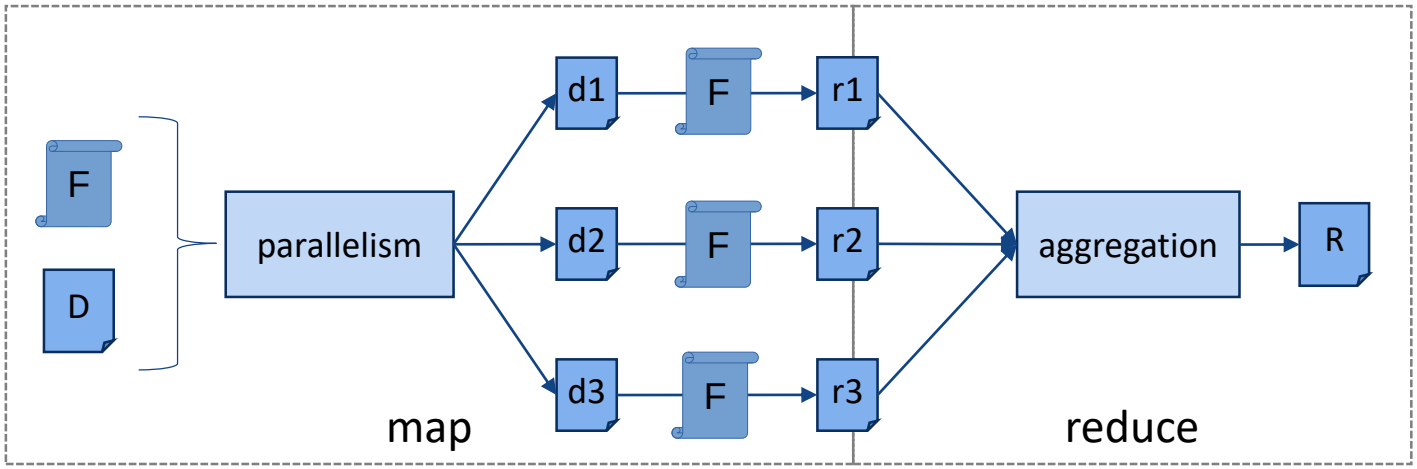
...[Execute Commands 1,3,5]



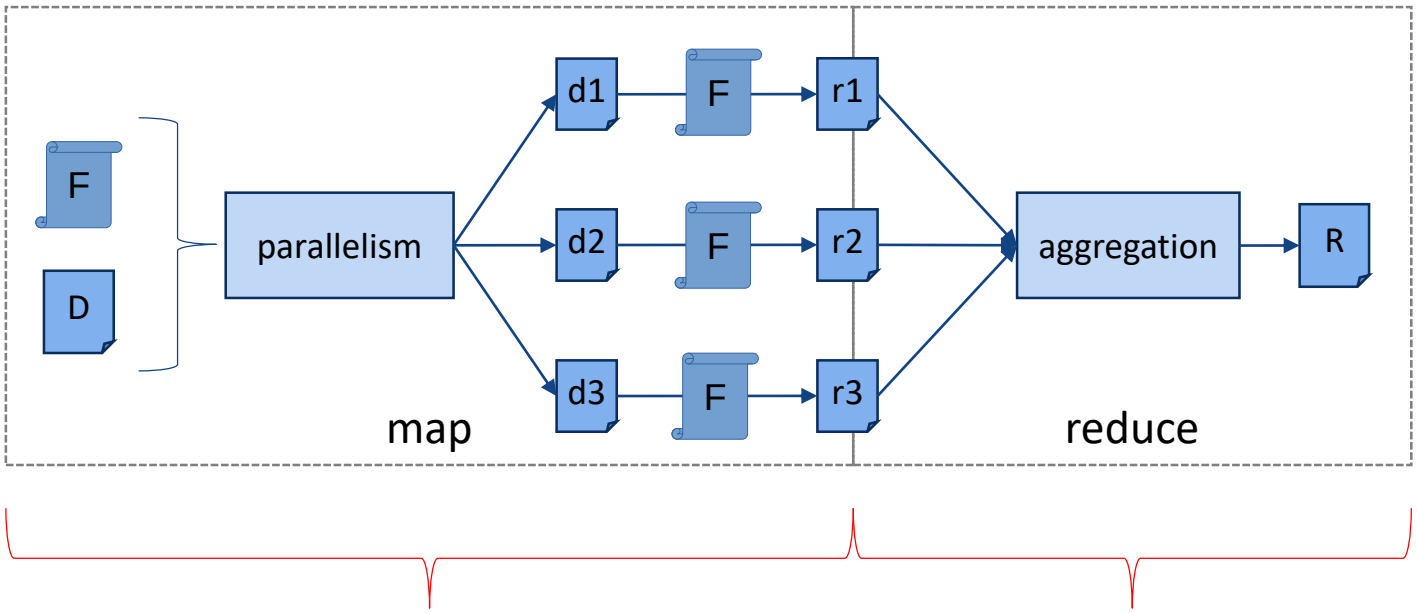
Map-Reduce on Spark

- Map-Reduce is the idea behind low-level Spark operations
  - "Apply function F over data D, and aggregate results into R"
  - Map:
    - "Apply function F to each element of dataset D"
  - Reduce:
    - "Aggregate the individual results of funtion F into a result R"

Map-Reduce on Spark



Map-Reduce on Spark



Summary

- Important concepts of Spark:
  - Distributed Data
  - Execution Tree
  - Map-Reduce

- Next Hands-On:
  - Let's run SPARK!

```
$ $SPARK_HOME/bin/pyspark
Python 3.6.5 (default, Mar 31 2018, 19:45:04) [GCC] on linux
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).
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\ version 2.4.1
      /_/
Using Python version 3.6.5 (default, Mar 31 2018 19:45:04)
SparkSession available as 'spark'.
>>>
```