PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE

Spark Cluster Overview

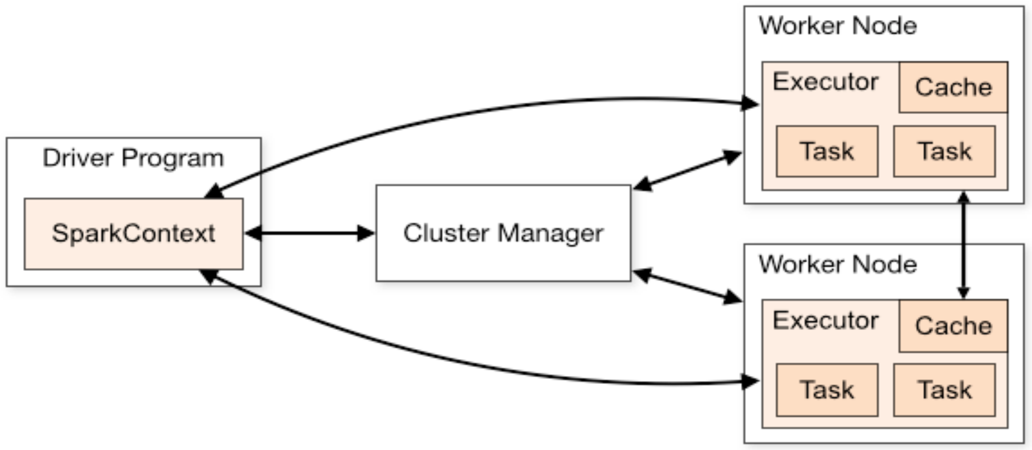
Amy Krause, Andreas Vroutsis

EPCC, The University of Edinburgh

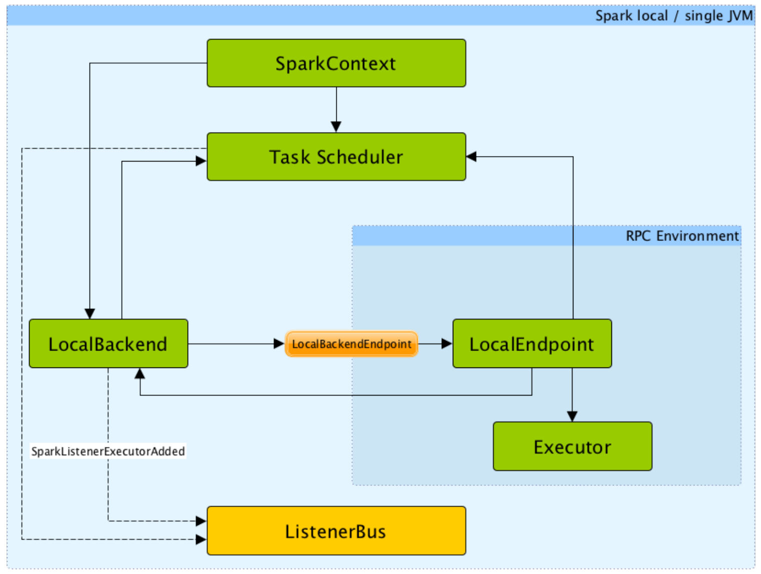
Spark Execution modes

It is possible to run a Spark application using cluster mode, local mode (pseudo-cluster) or with an interactive shell (pyspark or spark-shell).

Cluster mode

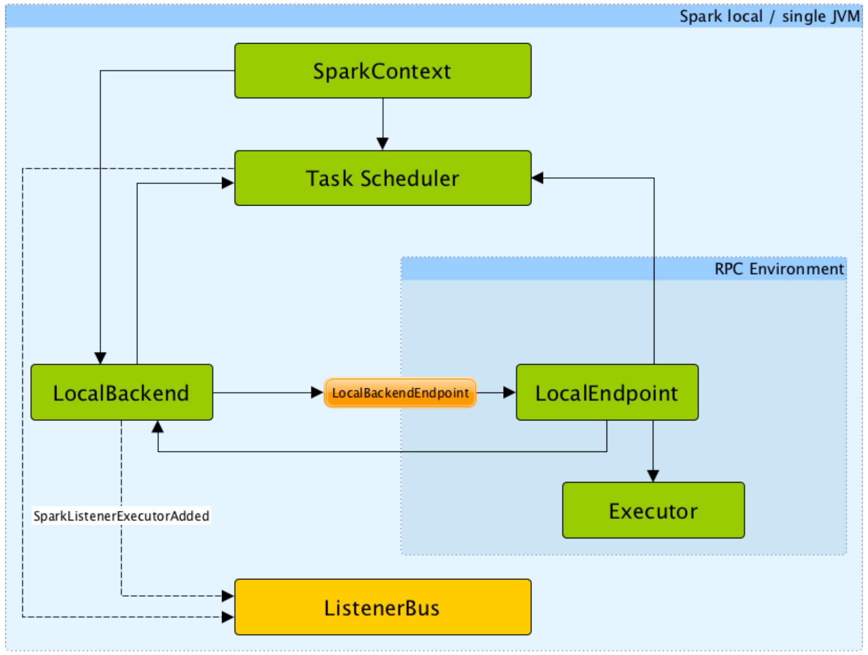


Local mode



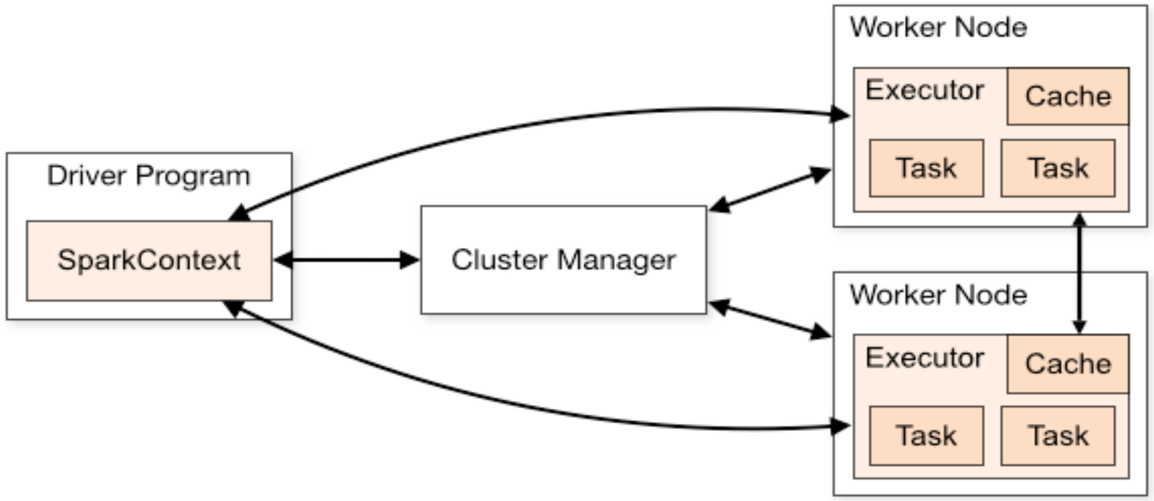
Spark Execution – Local Mode

- In this non-distributed single-JVM deployment mode.
- Spark spawns all the execution components - driver, executor, LocalSchedulerBackend, and master - in the same single JVM.
- The default parallelism is the number of threads as specified in the master URL.



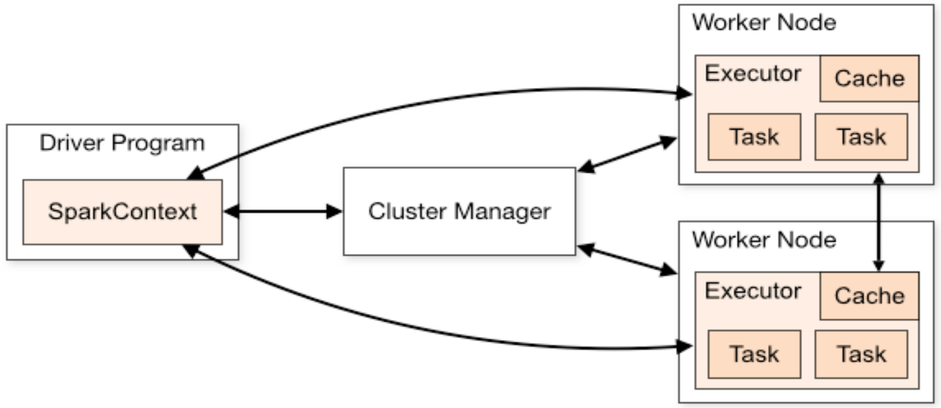
Standalone Deploy Mode

- Simplest way to deploy Spark on a private cluster
  - Apache Mesos
  - Hadoop YARN
  - Kubernetes



- Spark is agnostic to the underlying cluster manager

Spark Execution – Cluster mode

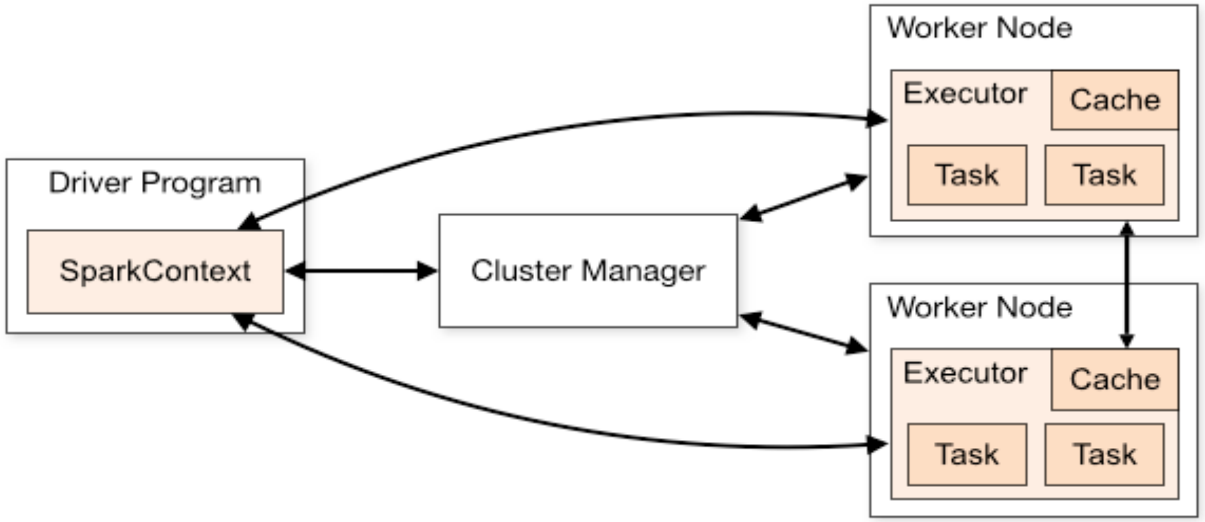


- Spark applications are run as independent sets of processes, coordinated by a SparkContext in a *driver* (*) program.
- The *context* connects to the cluster manager *which allocates resources*.
- Each *worker* in the cluster is managed by an *executor.*
- The *executor* manages computation as well as storage and caching on each machine.

(*) driver → process running the main() function of the application and creating the SparkContext

Spark Execution – Cluster mode

- The application code is sent from the *driver* to the *executors*, and the executors specify the context and the various *tasks* to be run.
- The *driver* program must listen for and accept incoming connections from its executors throughout its lifetime.



Spark App

Each SparkContext creates a Spark application, which includes a lot of scheduling components.

Upon an **Action**, the driver program submits the job to the cluster manager.

Cluster manager

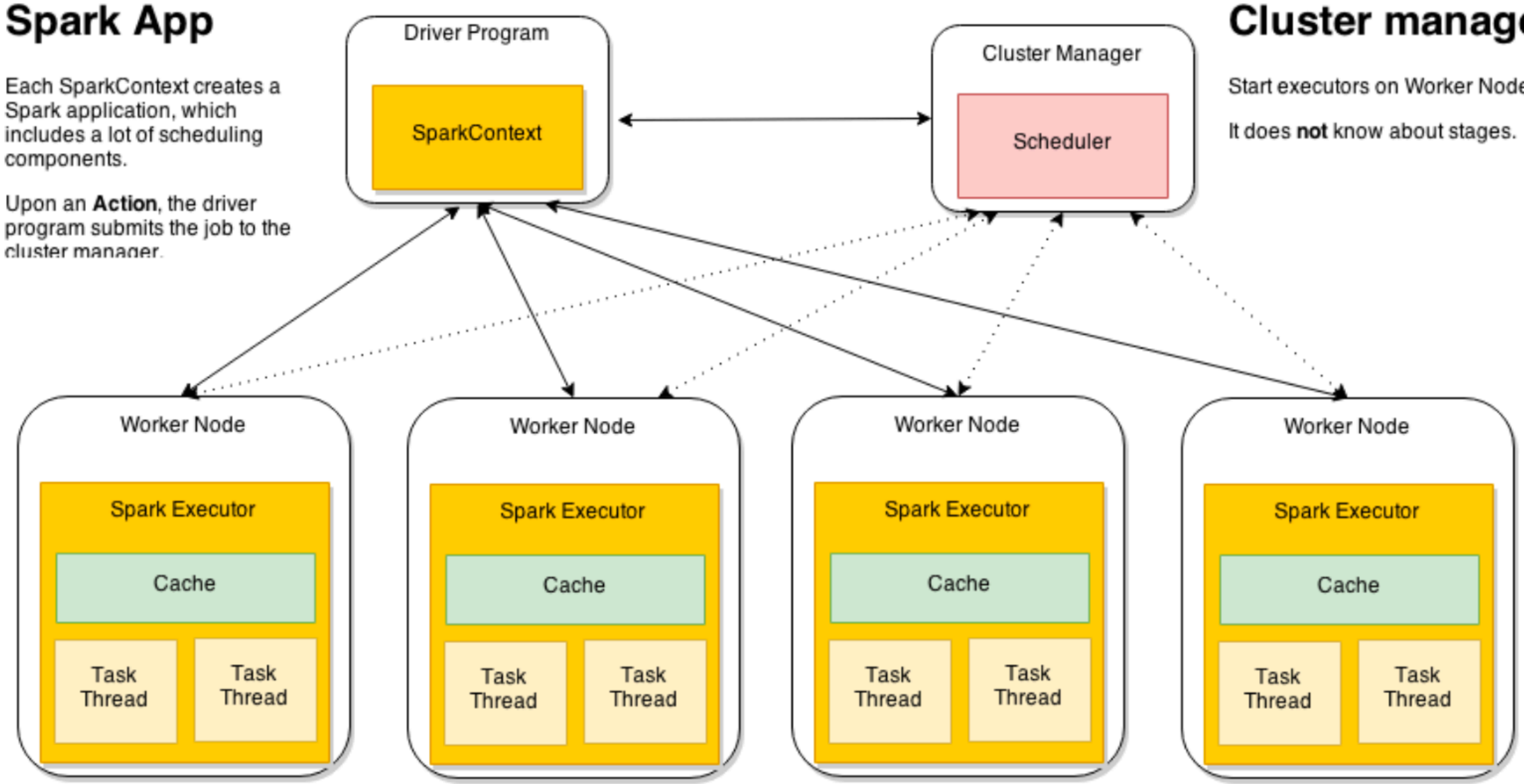
Start executors on Worker Nodes.

It does **not** know about stages.

Worker

Launch Spark Executor in a process.

Tasks are launched in separate threads, one per each core on the worker node (can be configured)

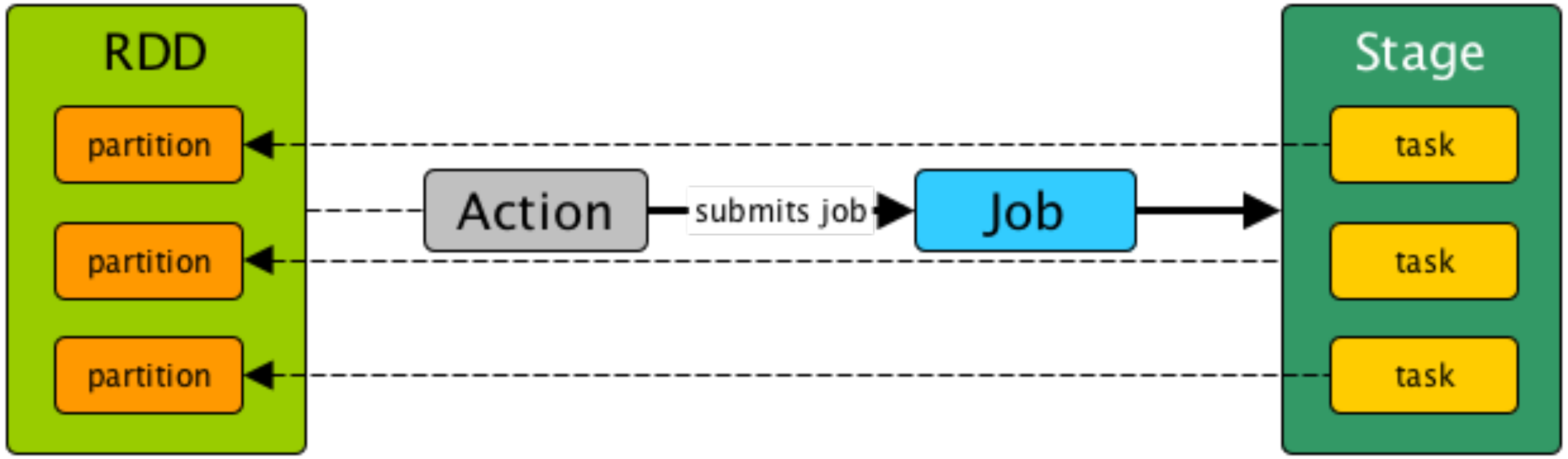


Spark: Standalone cluster – deploy modes

- For standalone clusters supports two deploy modes. They distinguish where the driver process runs:
  - *Client mode* (by default): the driver is launched in the same process as the client that submits the application.
  - *Cluster mode*: the driver is launched from one of the Worker processes inside the cluster.
    - The client process exits as soon as it fulfils its responsibility of submitting the application without waiting for the application to finish.
- Note: Currently, the **standalone mode** does not support **cluster mode** for **Python applications**.

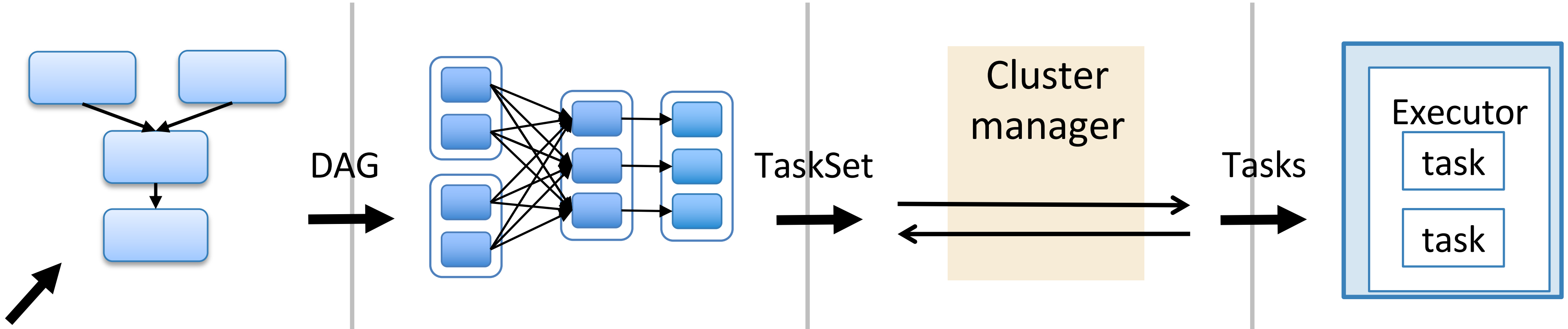
Spark Components

- Task: individual unit of work sent to one executor over a sequences of partitions
- Job : set of tasks executed as a result of an action
- Stage: set of tasks in a job that can be executed in parallel – at partition level
- RDD: Parallel dataset with partitions
- DAG: Logical Graph of RDD operations



Job scheduling

RDD Objects | DAGScheduler | TaskScheduler | Worker



```
rdd1.join(rdd2)
    .groupBy(…)
    .filter(…)
```

build operator DAG

split graph into *stages* of tasks

submit each stage as ready

launch tasks via cluster manager

retry failed or straggling tasks

execute tasks

Spark Application – wordcount.py

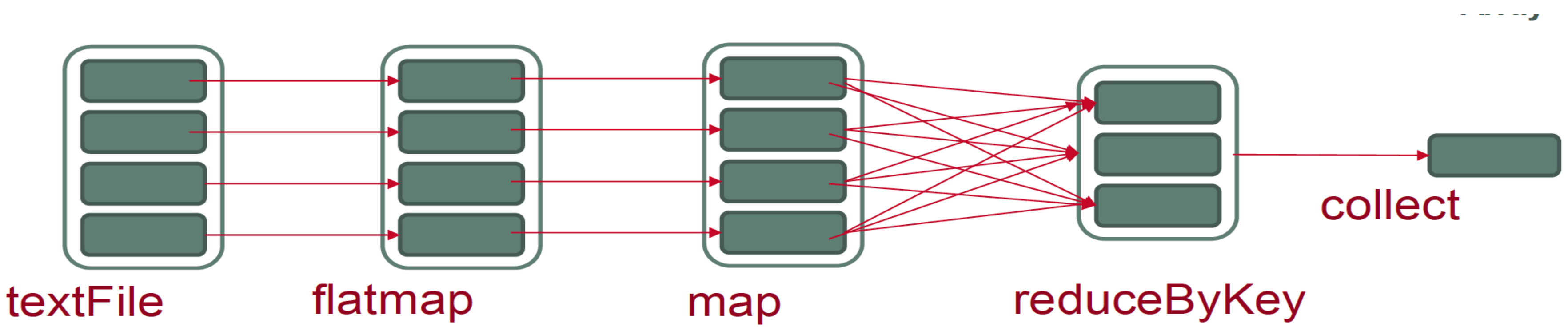
- The application that we are going to create is a simple “wordcount”:
  - Performs a **textFile** operation to read an input file in HDFS
  - **flatMap** operation to split each line into words
  - **map** operation to form (word, 1) pairs
  - **reduceByKey** operation to sum the counts (all the ’1’) for each word

Spark Application – wordcount.py

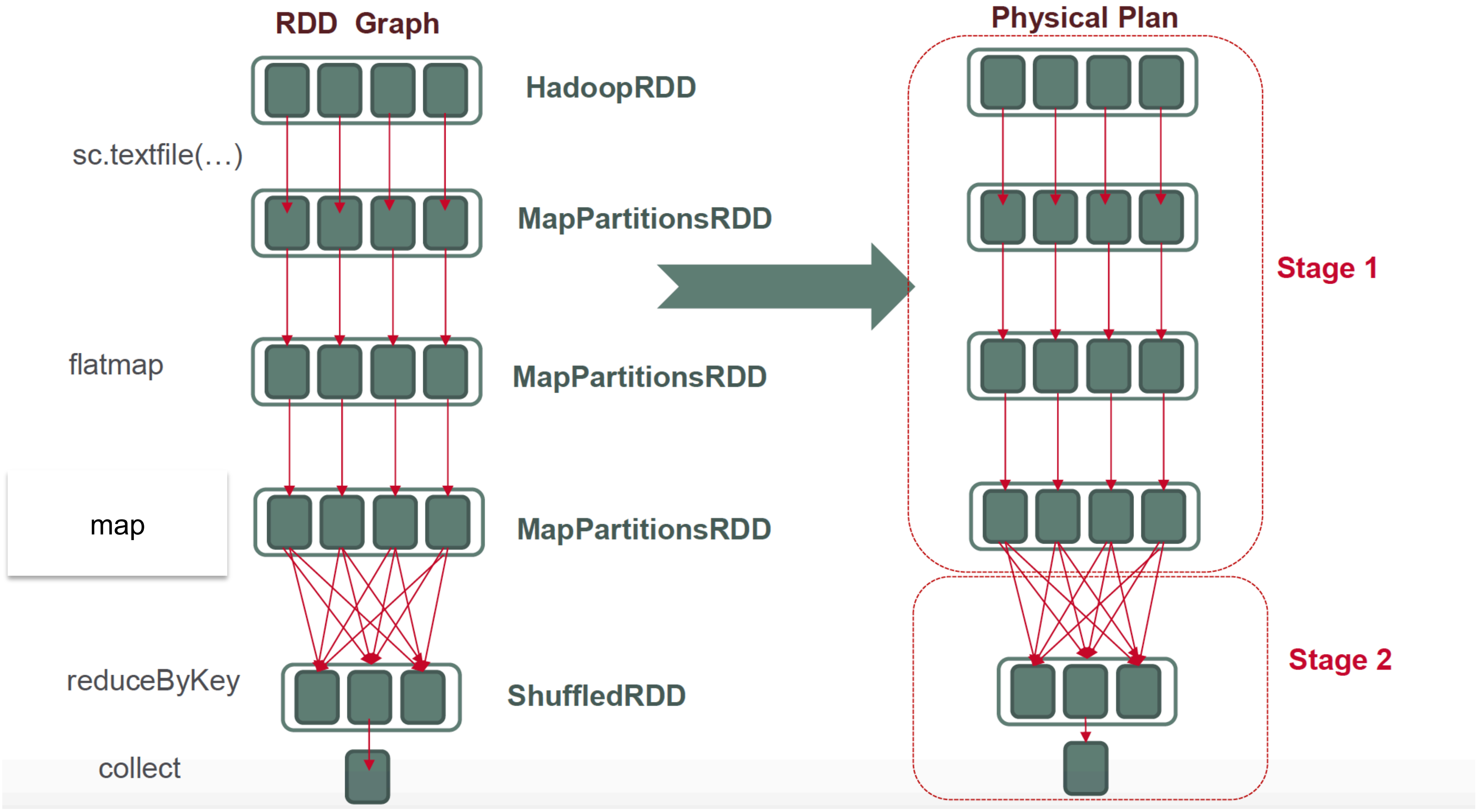
```
import sys
from pyspark import SparkContext, SparkConf


if __name__ == "__main__":
    conf = SparkConf().setAppName("Spark Count")
    sc = SparkContext(conf=conf)
    inputFile = sys.argv[1]
    textFile = sc.textFile(inputFile)
    wordCounts = textFile.flatMap(lambda line: line.split()).\
        map(lambda word: (word, 1)).reduceByKey(lambda a, b: a+b)
    output=wordCounts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))
```

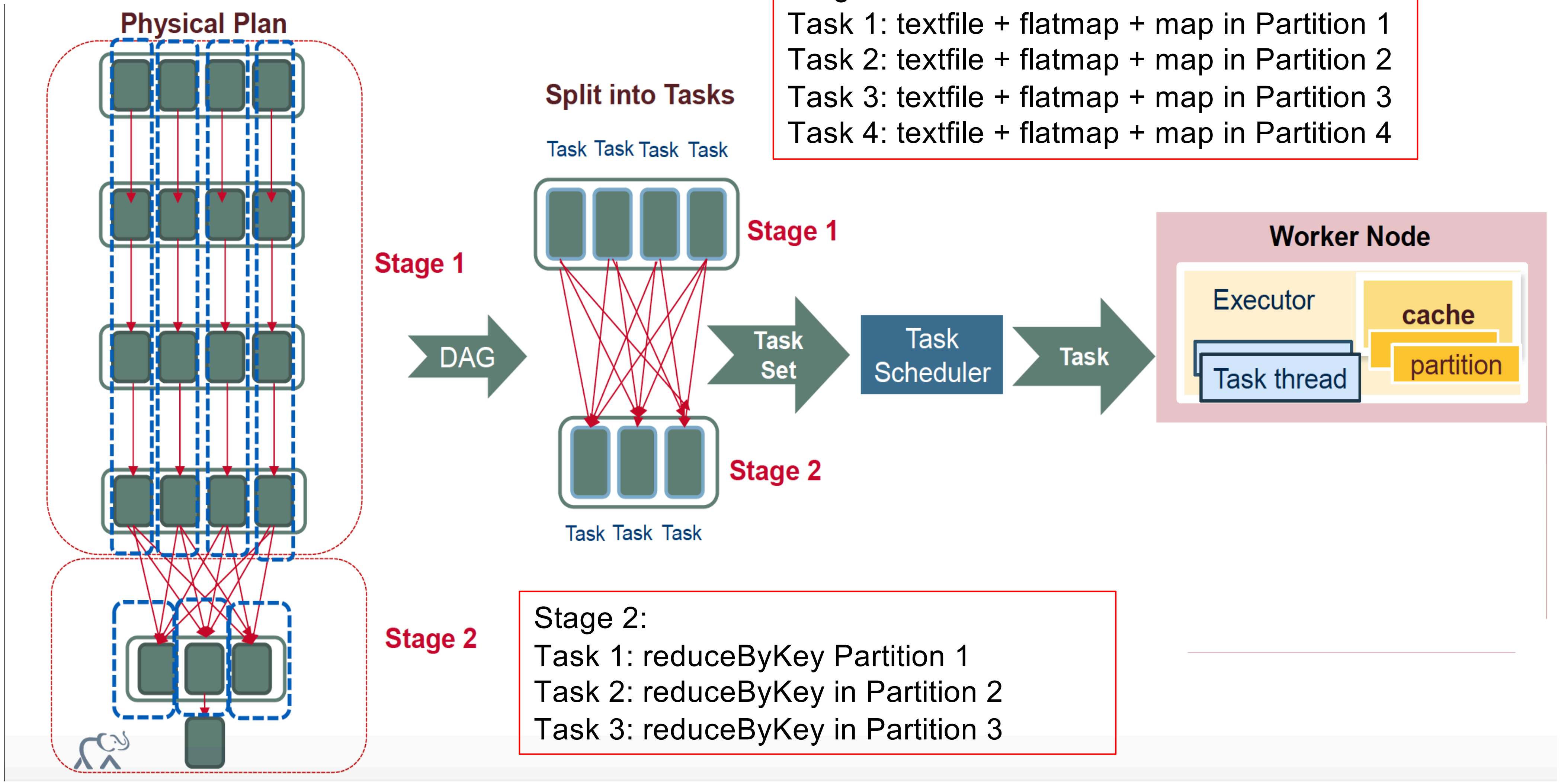
Spark Application – wordcount.py



RDD DAG -> Physical Execution plan



Initial RDD distributed among 4 partitions. Final RDD distributed among 3 partitions



Stage 1:
Task 1: textfile + flatmap + map in Partition 1
Task 2: textfile + flatmap + map in Partition 2
Task 3: textfile + flatmap + map in Partition 3
Task 4: textfile + flatmap + map in Partition 4

Stage 2:
Task 1: reduceByKey Partition 1
Task 2: reduceByKey in Partition 2
Task 3: reduceByKey in Partition 3

Operations that can run on the same partition are executed in stages

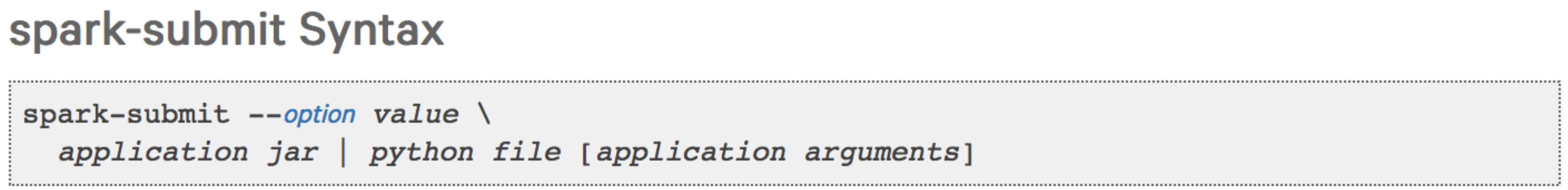
Running Spark Applications

- **Notebooks** are great for:
  - developing and testing quickly experiment with the data
  - demos and collaborating with other people
- **Spark-submit** jobs are more likely to be used in **production**.

Running Spark with Jupyter Notebooks

- We are going to use Jupyter Notebooks for running our walkthroughs & lab exercises.
- First we need to do the following steps:
  - Copying all the necessary material in our accounts in Cirrus
  - Starting an interactive session in a node
  - Starting a spark cluster (standalone) in that node
  - Starting a Jupyter session connected with pyspark
- All the information can be found in “Get_Started_Notebooks_Cirrus”:
  https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Get_Started_Notebooks_Cirrus.pdf

Submit job via spark-submit


spark-submit Syntax

```
spark-submit --option value \
  application jar | python file [application arguments]
```

Check the guide - Submitting Spark Applications:
https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Spark_Applications/Submitting_Spark_Applications.pdf

Submit job via spark-submit

```
 $SPARK_HOME/bin/spark-submit \
--class <main-class> \
--master <master-url> \
--deploy-mode <deploy-mode>  \
--conf \
….
<application-jar> [arguments] |
<python file >[arguments]
```

Some spark-submit options

- master – Determines how to run the job:
  - spark://r1i2n5:7077
  - local
- driver-memory
  - amount memory available for the driver process.
- executor-memory
  - amount of memory allocated to the executor process
- executor-cores
  - total number of cores allocated to the executor process
- total-executor-cores
  - Total number of cores available for all executors.

See: https://spark.apache.org/docs/latest/submitting-applications.html

Cirrus

- High-performance computing cluster
- One of the EPSRC Tier-2 National HPC Services.
- 280 nodes: 36 Intel Xeon CPUs, hyper threading, 256GB
  - Each node has (virtually) 72 cores
- 406 TB of storage – Lustre file system
- Link: http://www.cirrus.ac.uk/

https://cirrus.readthedocs.io/en/latest/user-guide/connecting.html

Cirrus

- Connecting to Cirrus

  ```
  ssh [userID]@login.cirrus.ac.uk
  ```

- Two types of nodes:
  - Login – access to outside network
  - Computing – only network between nodes (no access to outside world)

- For cloning the repository -> use the login node

- https://cirrus.readthedocs.io/en/latest/user-guide/connecting.html

Running jobs on Cirrus

- PBSPro to schedule jobs
  - Submission script to submit a job a queue
  - Interactive jobs are also available
    - To submit a request for an interactive job reserving 1 nodes (72 physical cores) for 1 hour you would issue the following qsub command from the command line

```
qsub -IVl select=3:ncpus=36,walltime=01:00:00,place=scatter:excl -A y15 -
q <reservation number> -j oe
```

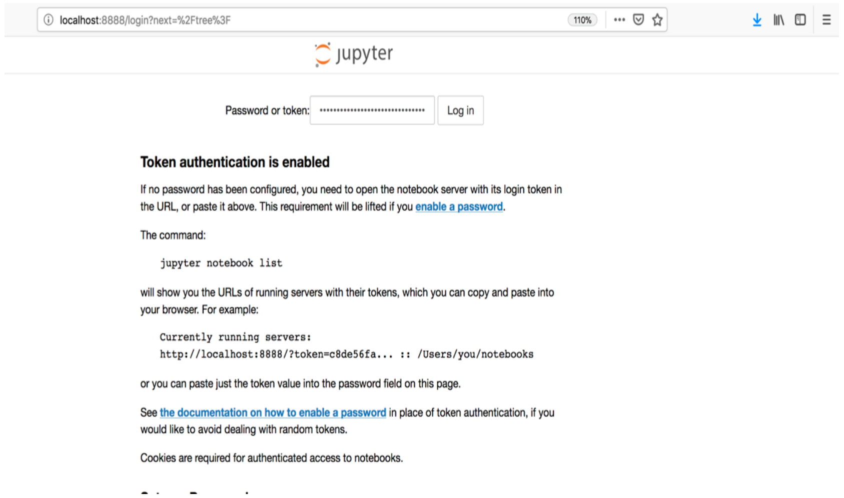
- Your session will end:
  - It hits the requested walltime
  - Typing exit command within the session
- https://cirrus.readthedocs.io/en/latest/user-guide/batch.html#interactive-jobs

Jupyter notebooks

- Start the jupyter server:
  - ./start_Jupyter_local.sh will give you a token, like this one:

    http://0.0.0.0:8888/?token=2d5e554b2397355c334b8c3367503b06c4f6f95a26151795
- Open another terminal and type the following command

  ssh USER@login.cirrus.ac.uk -L8888:MASTER NODE:8888
- Got to a Web browser at http://localhost:8888


localhost:8888/login?next=%2Ftree%3F

jupyter

Password or token: •••••••••••••••••••••••••••••••• Log in

Token authentication is enabled

If no password has been configured, you need to open the notebook server with its login token in the URL, or paste it above. This requirement will be lifted if you enable a password.

The command:

```
jupyter notebook list
```

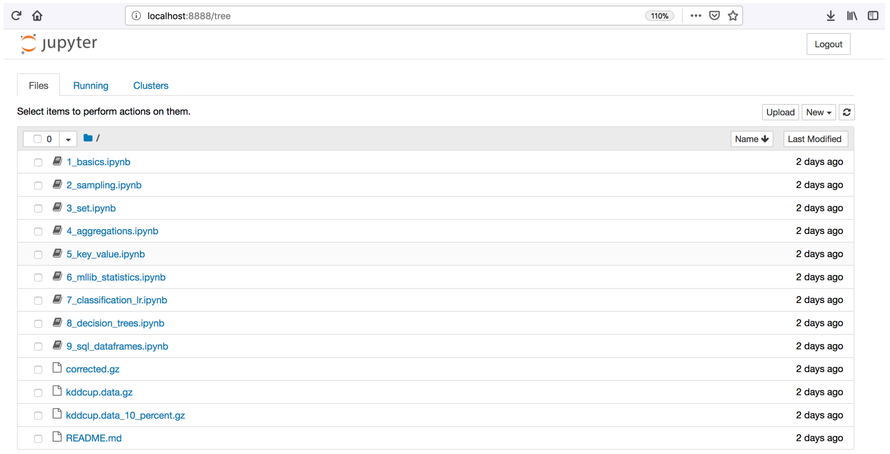
will show you the URLs of running servers with their tokens, which you can copy and paste into your browser. For example:

```
Currently running servers:
http://localhost:8888/?token=c8de56fa... :: /Users/you/notebooks
```

or you can paste just the token value into the password field on this page.

See the documentation on how to enable a password in place of token authentication, if you would like to avoid dealing with random tokens.

Cookies are required for authenticated access to notebooks.


localhost:8888/tree

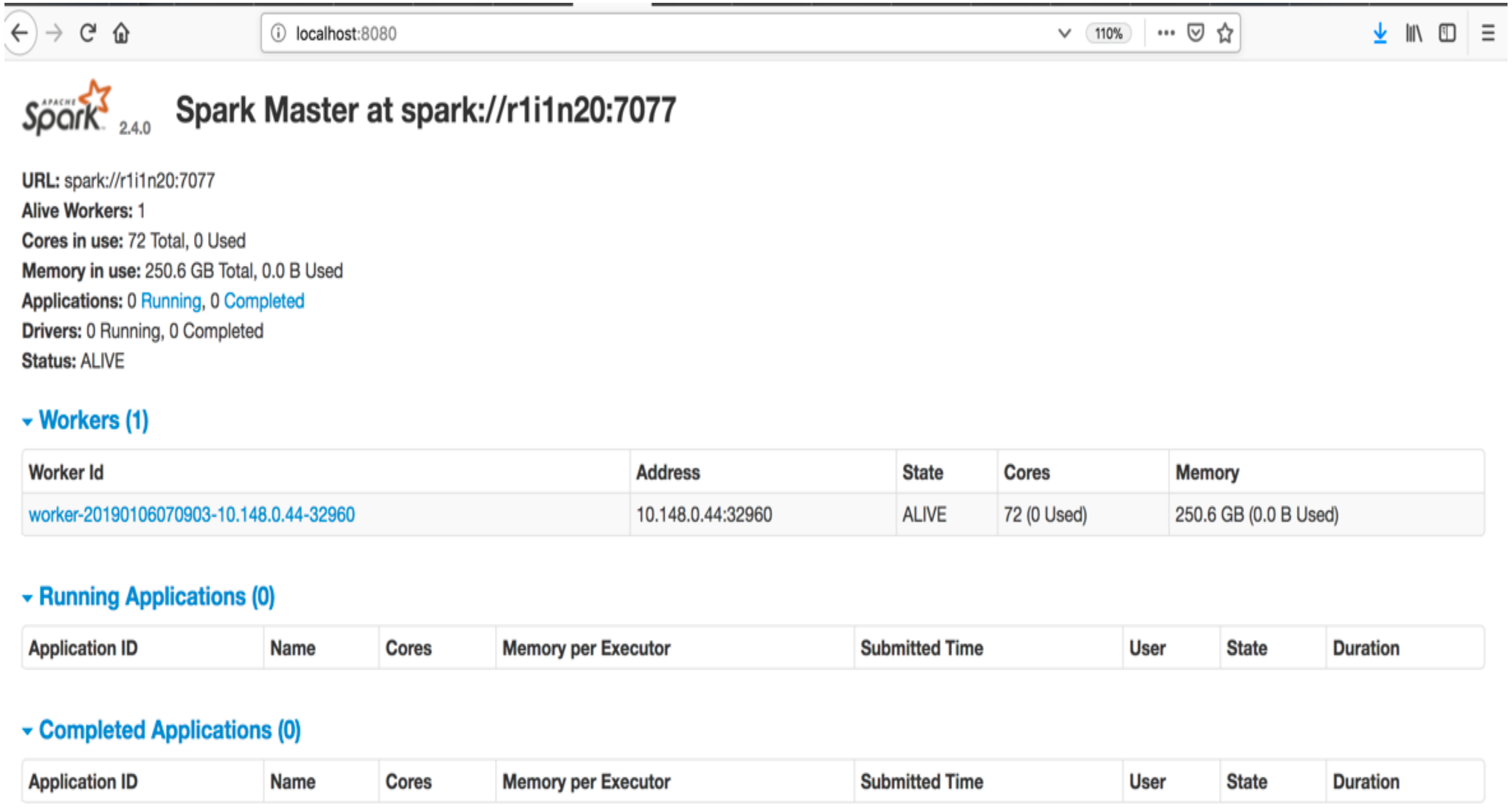
jupyter

Select items to perform actions on them.

1_basics.ipynb	2 days ago
2_sampling.ipynb	2 days ago
3_set.ipynb	2 days ago
4_aggregations.ipynb	2 days ago
5_key_value.ipynb	2 days ago
6_mllib_statistics.ipynb	2 days ago
7_classification_lr.ipynb	2 days ago
8_decision_trees.ipynb	2 days ago
9_sql_dataframes.ipynb	2 days ago
corrected.gz	2 days ago
kddcup.data.gz	2 days ago
kddcup.data_10_percent.gz	2 days ago
README.md	2 days ago

All the information can be found at “Get_Started_Notebooks_Cirrus”:
https://github.com/EPCCed/prace-spark-for-data-scientists/blob/master/Get_Started_Notebooks_Cirrus.pdf

Master Spark UI



localhost:8080

Spark 2.4.0 **Spark Master at spark://r1i1n20:7077**

URL: spark://r1i1n20:7077
Alive Workers: 1
Cores in use: 72 Total, 0 Used
Memory in use: 250.6 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20190106070903-10.148.0.44-32960	10.148.0.44:32960	ALIVE	72 (0 Used)	250.6 GB (0.0 B Used)

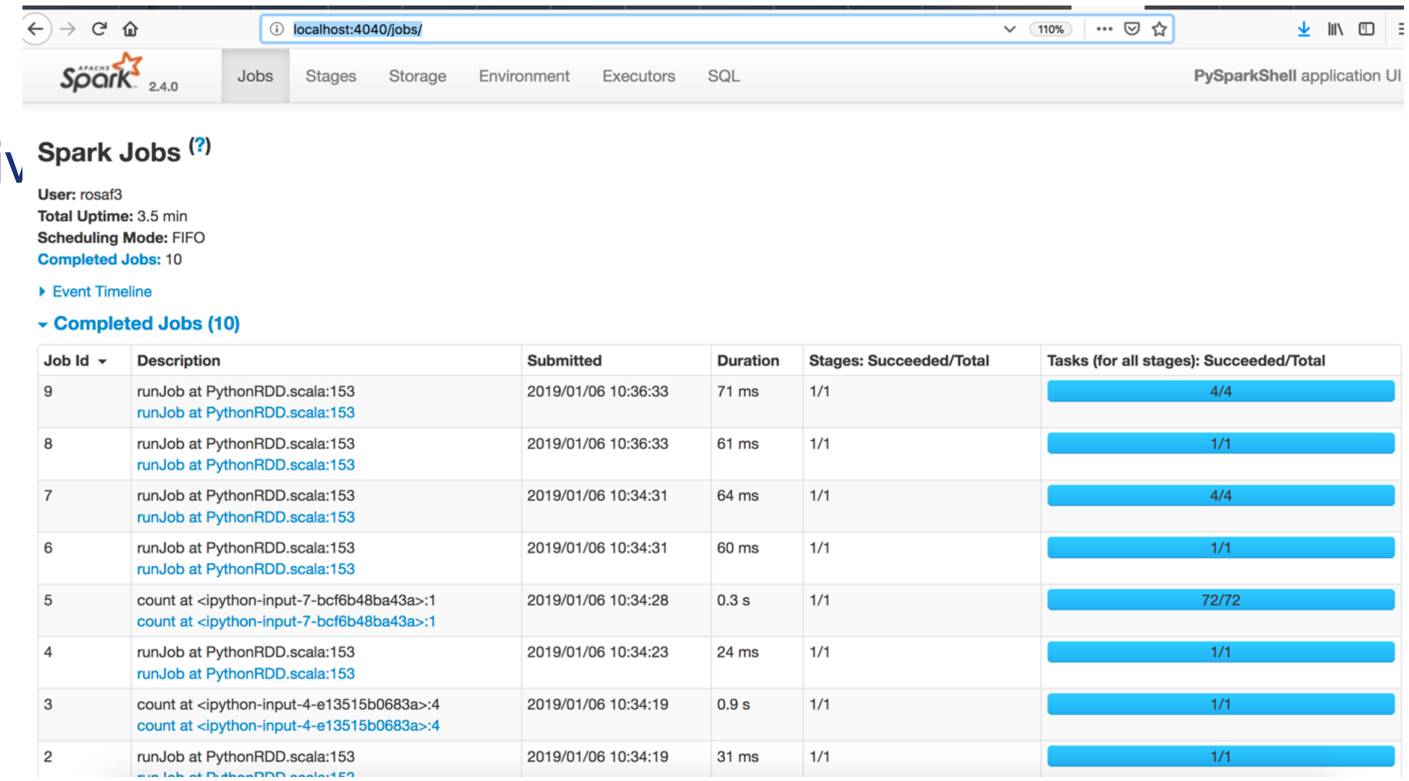
Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Driv



Spark Jobs (?)

User: rosaf3
Total Uptime: 3.5 min
Scheduling Mode: FIFO
Completed Jobs: 10

▸ Event Timeline

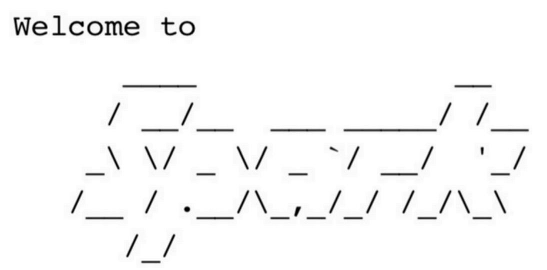
▾ **Completed Jobs (10)**

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
9	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:36:33	71 ms	1/1	4/4
8	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:36:33	61 ms	1/1	1/1
7	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:31	64 ms	1/1	4/4
6	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:31	60 ms	1/1	1/1
5	count at <ipython-input-7-bcf6b48ba43a>:1 count at <ipython-input-7-bcf6b48ba43a>:1	2019/01/06 10:34:28	0.3 s	1/1	72/72
4	runJob at PythonRDD.scala:153 runJob at PythonRDD.scala:153	2019/01/06 10:34:23	24 ms	1/1	1/1
3	count at <ipython-input-4-e13515b0683a>:4 count at <ipython-input-4-e13515b0683a>:4	2019/01/06 10:34:19	0.9 s	1/1	1/1
2	runJob at PythonRDD.scala:153	2019/01/06 10:34:19	31 ms	1/1	1/1

Every SparkContext launches a web UI (Spark driver's web UI),
by default on port 4040, that displays useful information about the application.

ssh USER@login.cirrus.ac.uk -L4040:DRIVER NODE:4040
web browser → localhost:4040

Running notebooks in your laptop



```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\
      /_/
```

- **Prerequisites: Anaconda, Python3**
- Get Spark from the downloads page of the project websi
  (https://blog.sicara.com/get-started-pyspark-jupyter-guide-tutorial-ae2fe84f594f)
- Check if pyspark is properly install → type `pyspark` in a terminal

```
>> git clone https://github.com/EPCCed/prace-spark-for-data-
scientists.git
>> cd walkthrough_examples
>> export SPARK_HOME=[INSTALLATION_PATH]/spark-2.4.0-bin-hadoop2.7/
>> PYSPARK_DRIVER_PYTHON=jupyter PYSPARK_DRIVER_PYTHON_OPTS='notebook’ \
$SPARK_HOME/bin/pyspark
```

THANK YOU FOR YOUR ATTENTION

www.prace-ri.eu