

SPARK ON HIPERGATOR

Ying Zhang
yingz@ufl.edu

November 6th, 2018

RESEARCH COMPUTING STAFF

- Dr. Matt Gizendanner
 - Bioinformatics Specialist
- Dr. Justin Richardson
 - Research Facilitator

AGENDA

- Introduction
 - Apache Spark
 - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises
- All slides are available at:

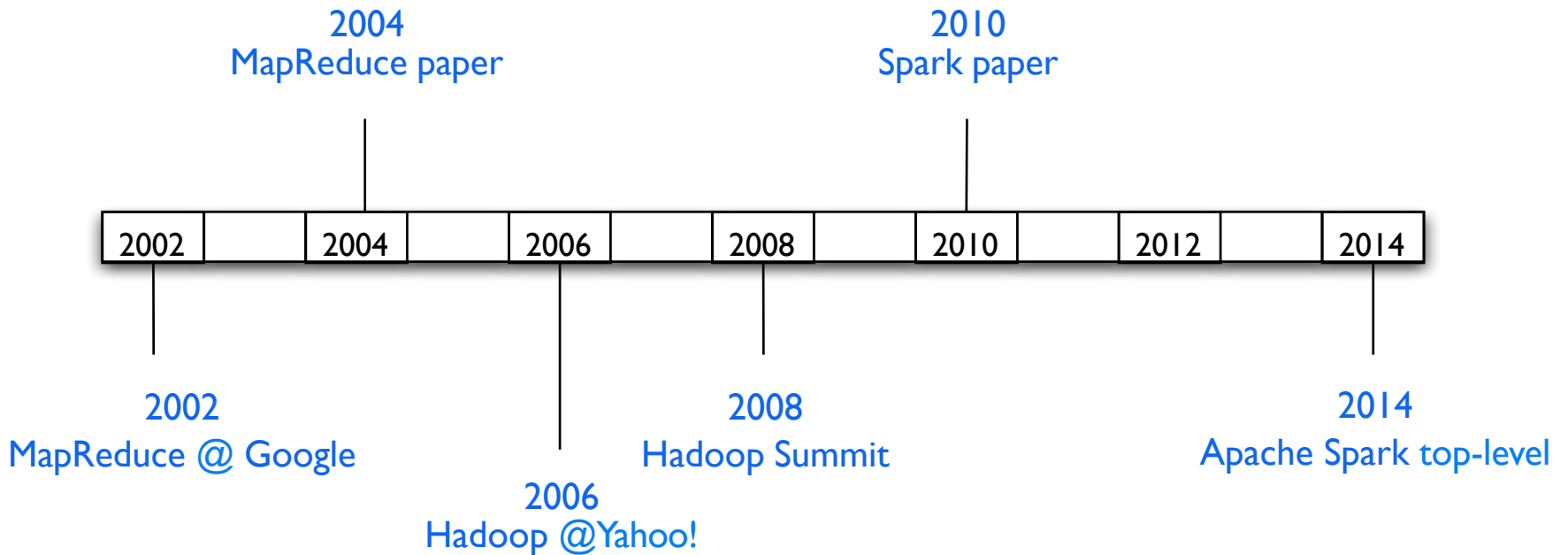
https://help.rc.ufl.edu/doc/Spark_Workshop

AGENDA

- Introduction
 - Apache Spark
 - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises

APACHE SPARK

- A brief history



MAP-REDUCE

- Data-parallel model
 - One operation, run it on all of the data
- A simple programming model that applies to many large-scale computing problems
- Typical problem
 - Read a lot of data
 - Map: extract desired information from each record
 - Shuffle/sort
 - Reduce: aggregate, summarize, filter, or transform
 - Write the results

MAP-REDUCE

- Word count example:

“Deer, Bear, River, Car, Car, River, Dear, Car, Beer”

- *Map* function:

- key/value pairs: 9 pairs

“Deer”, “1”

“Bear”, “1”

“River”, “1”

“Car”, “1”

“Car”, “1”

“River”, “1”

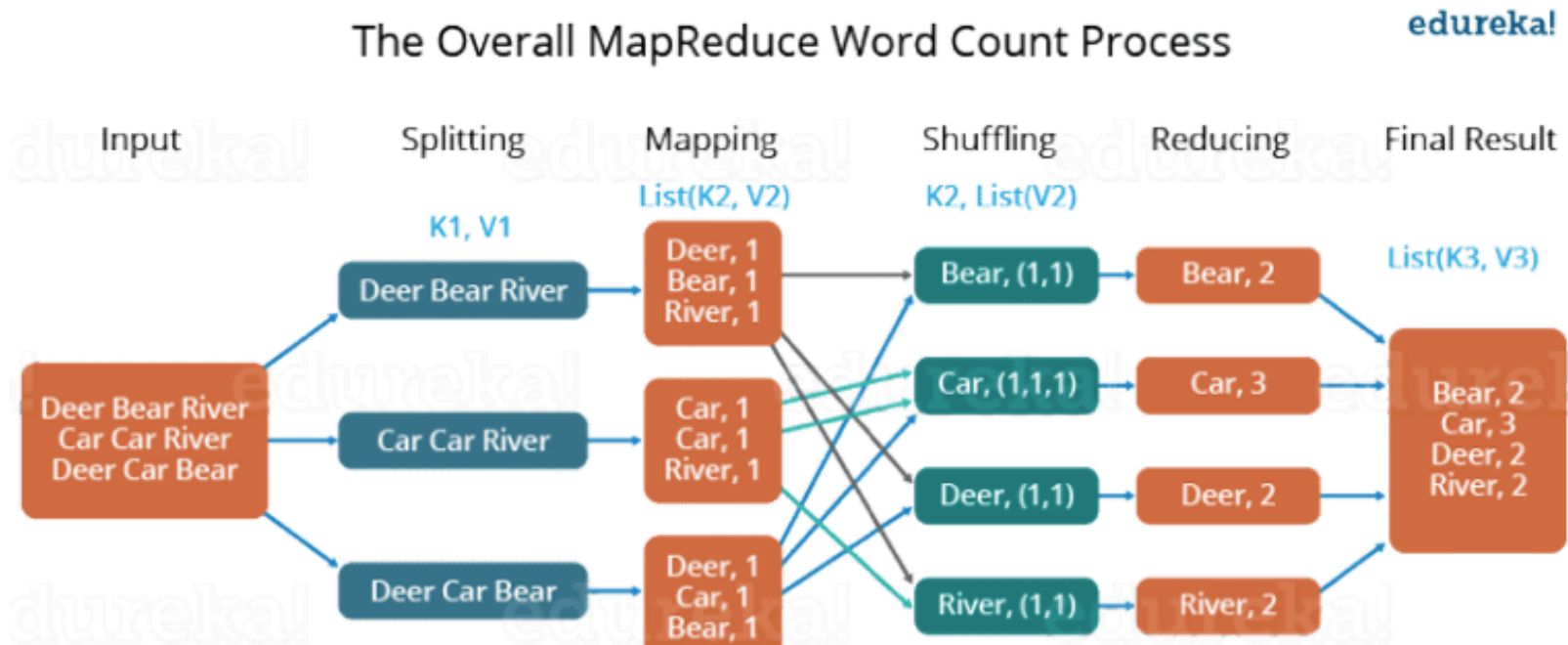
“Deer”, “1”

“Car”, “1”

“Bear”, “1”

MAP-REDUCE

- Word count example :
“Dear, Bear, River, Car, Car, River, Dear, Car, Bear”
- *Shuffle/sort*: gathers all pairs with the same key value
- *Reduce* function combines the values for a key



MAP-REDUCE

- Major limitations:
 - Difficulty to program directly
 - Performance bottlenecks
- Higher level frameworks, e.g. Hive, Pregel, Dremel, etc.

HADOOP & SPARK

- Hadoop
 - Started in 2006 at Yahoo
 - HDFS: Hadoop File System
 - YARN: a scheduler coordinates application runs
 - Built in JAVA, support Python and others
- Spark
 - Started in 2008 at AMPLab at UC Berkeley
 - Resilient Distributed Dataset (RDD), in memory process
 - Run in standalone mode or with Hadoop cluster
 - Directed Acyclic Graph (DAG), visualize the order of operations and relationships of operations
 - Written in Scala, support Java, Python and R

SPARK

- Handles batch, interactive, and real-time within a single framework
- Written in SCALA
- Integration with Java, Python, and R
- Programming at a higher level of abstraction
- More general and beyond map/reduce

HADOOP VS. SPARK

● apache hadoop
Search term

● apache spark
Search term

+ Add comparison

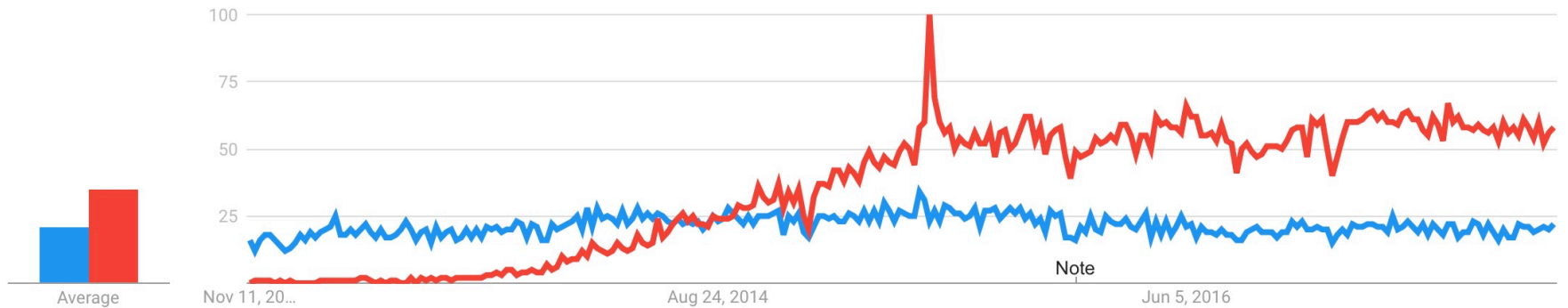
Worldwide ▼

Past 5 years ▼

All categories ▼

Web Search ▼

Interest over time ?



Note

SPARK PROGRAMMABILITY

WordCount in 50+ lines of Java

WordCount in 3 lines of Spark Scala

```
1 public class WordCount {
2     public static class TokenizerMapper
3         extends Mapper<Object, Text, Text, IntWritable> {
4
5         private final static IntWritable one = new IntWritable(1);
6         private Text word = new Text();
7
8         public void map(Object key, Text value, Context context
9             ) throws IOException, InterruptedException {
10            StringTokenizer itr = new StringTokenizer(value.toString());
11            while (itr.hasMoreTokens()) {
12                word.set(itr.nextToken());
13                context.write(word, one);
14            }
15        }
16    }
17
18    public static class IntSumReducer
19        extends Reducer<Text, IntWritable, Text, IntWritable> {
20        private IntWritable result = new IntWritable();
21
22        public void reduce(Text key, Iterable<IntWritable> values,
23            Context context
24                ) throws IOException, InterruptedException {
25            int sum = 0;
26            for (IntWritable val : values) {
27                sum += val.get();
28            }
29            result.set(sum);
30            context.write(key, result);
31        }
32    }
33
34    public static void main(String[] args) throws Exception {
35        Configuration conf = new Configuration();
36        String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
37        if (otherArgs.length < 2) {
38            System.err.println("Usage: wordcount <in> [<in>...] <out>");
39            System.exit(2);
40        }
41        Job job = new Job(conf, "word count");
42        job.setJarByClass(WordCount.class);
43        job.setMapperClass(TokenizerMapper.class);
44        job.setCombinerClass(IntSumReducer.class);
45        job.setReducerClass(IntSumReducer.class);
46        job.setOutputKeyClass(Text.class);
47        job.setOutputValueClass(IntWritable.class);
48        for (int i = 0; i < otherArgs.length - 1; ++i) {
49            FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
50        }
51        FileOutputFormat.setOutputPath(job,
52            new Path(otherArgs[otherArgs.length - 1]));
53        System.exit(job.waitForCompletion(true) ? 0 : 1);
54    }
55 }
```

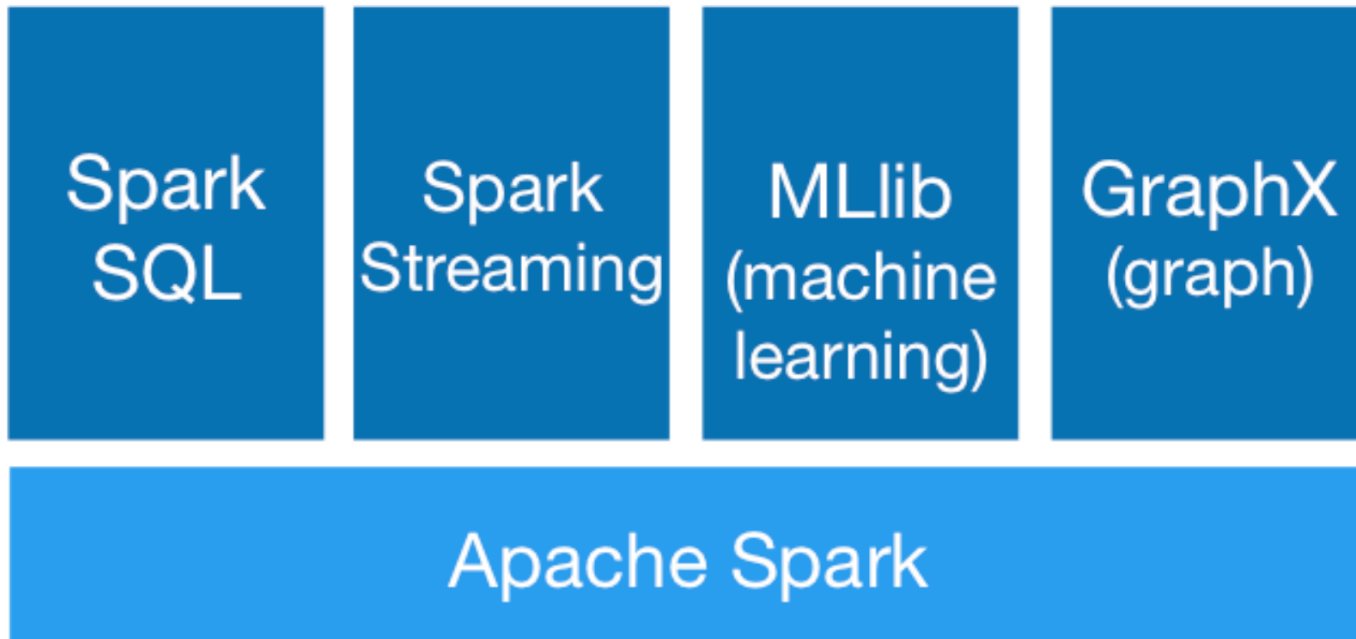
```
1 val f = sc.textFile(inputPath)
2 val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
3 w.reduceByKey(_ + _).saveAsText(outputPath)
```

SPARK PERFORMANCE

Sort 100TB of data with 1 Trillion records

	Hadoop MR Record	Spark Record
Data Size	102.5TB	100TB
Elapsed Time	72 minutes	23 minutes
Number of Nodes	2100	206
Number of Cores	50400 physical	6592 virtualized
Sort Rate	1.42 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min

SPARK ECOSYSTEM



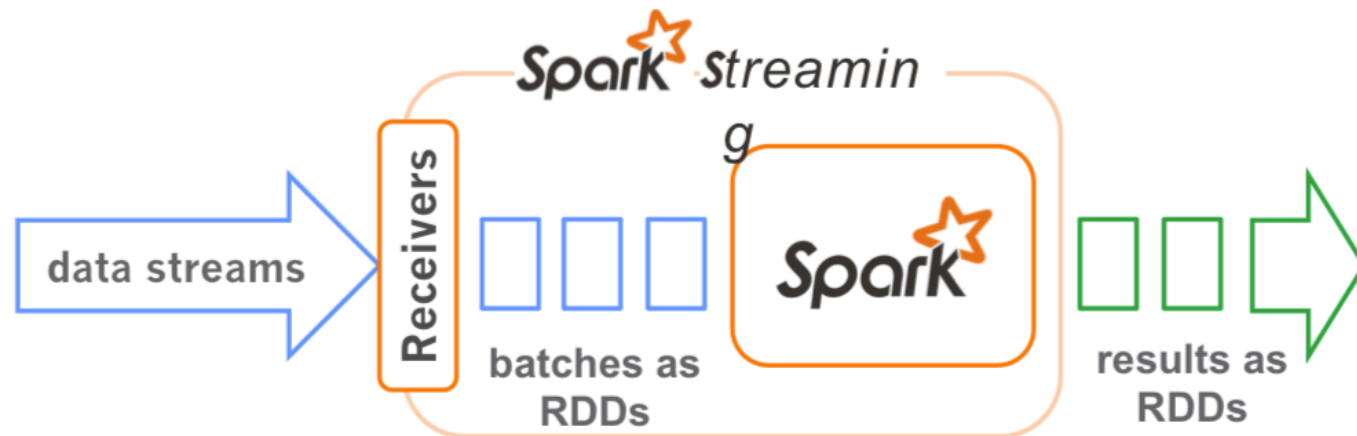
SPARK SQL AND DATAFRAMES

- SparkSQL
 - Allows SQL-like commands on distributed data sets
- Spark DataFrames
 - Developed in Spark 2.0
 - Organizes data into named columns (i.e. RDD with schema)
- SparkSQL allows querying DataFrames
- Support Python, Scala, Java, and R

```
spark.sql("SELECT * FROM people");
```


SPARK STREAMING

- What is it?
 - Receive data streams from input source
 - Break the data streams into small batches as RDDs (Dstream)
 - Process the batches using RDD operations in parallel
 - Output to databases/dashboards
 - Fault tolerant, second-scale latency
 - Support Scala, Java, and Python



SPARK MLLIB

- Provide machine learning primitives
 - Shipped with Spark since version 0.8
- Algorithms
 - Classification: Multilayer Perceptron Classifier, PC, linear SVM, Naïve Bayes
 - Regression: generalized linear regression (GLM)
 - Collaborative filtering: alternating least squares (ALS)
 - Clustering: k-means
 - Decomposition: single value decomposition (SVD), and principal component analysis (PCA)
- Support Java, Scala, and Python

SPARK GRAPHX

- Graph analytics
 - Examples: social networks, page rank, fraud detection, etc.
 - Graph data modeling
 - Graph data processing
- GraphX
 - API for graphs and graph-parallel computation
 - A growing library of graph algorithms
 - Performance comparable to the fastest specialized graph processing systems

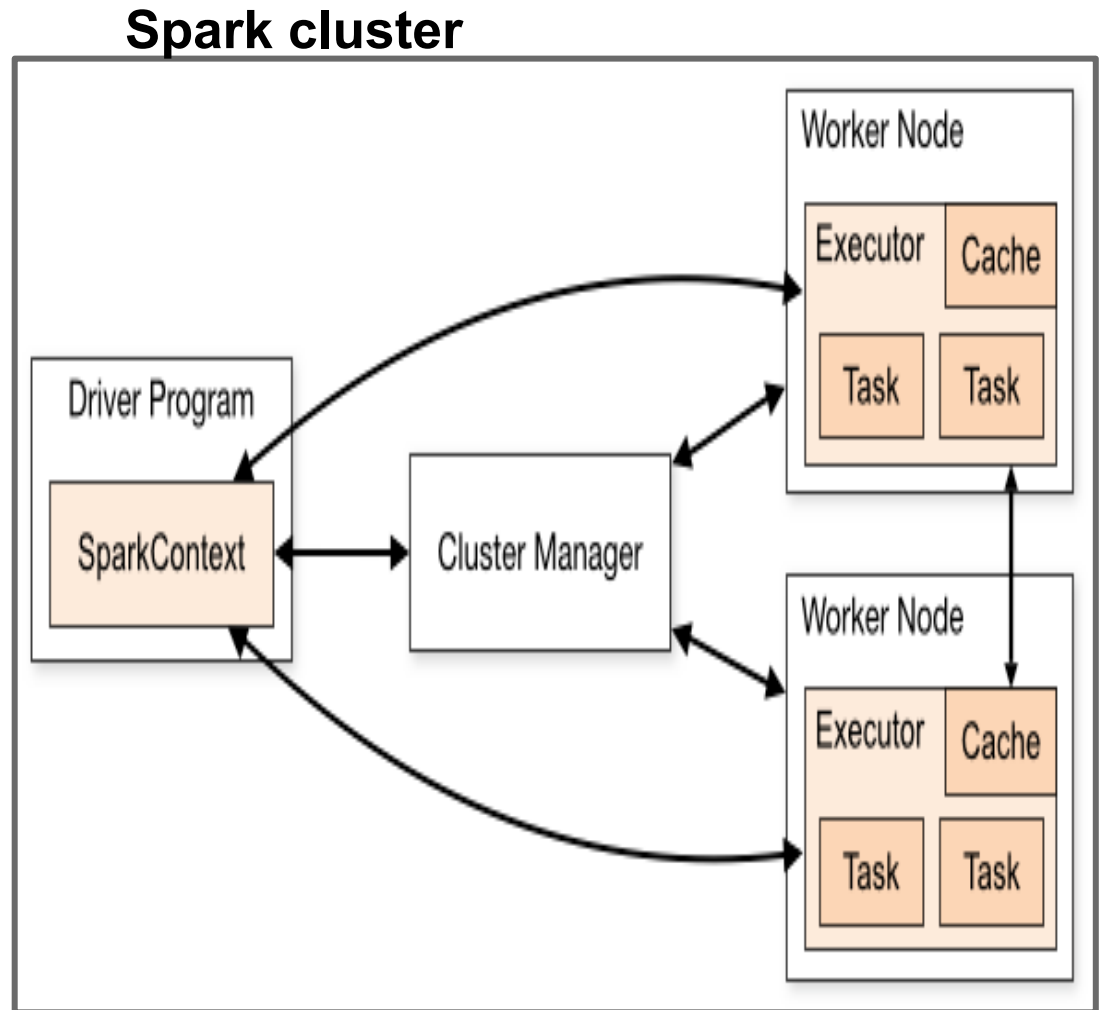


SPARK ARCHITECTURE OVERVIEW

- A master/slave paradigm
 - Master Daemon - driver process
 - Schedule the job executions
 - Negotiate with the cluster manager for resources
 - Translate RDD's into the execution graph (DAG)
 - Translate the user code into actual spark jobs (tasks)
 - Slave Daemon - worker process
 - Distributed agents to execute jobs (tasks)
 - Perform all the data processing

SPARK ARCHITECTURE OVERVIEW

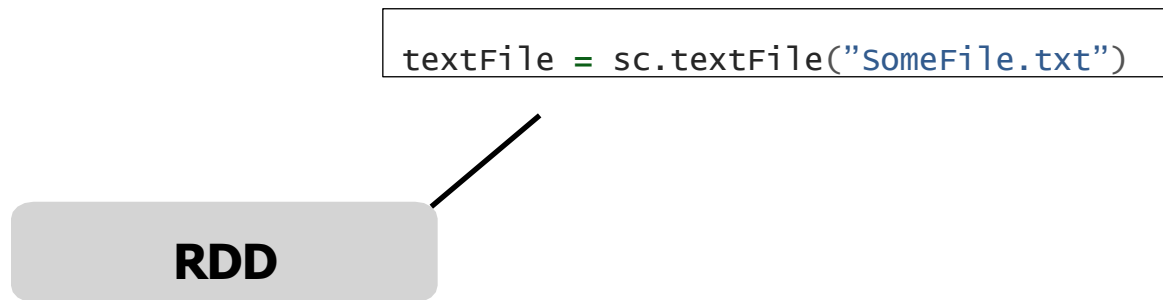
- **Cluster manager (master):** resource manager (standalone manager)
- **Worker node:** any node running application.
- **Application:** user program built on Spark. Driver program + executors
- **Driver program:** process running the main() function of the application
- **Executor:** process launched for an application on a worker node. it runs tasks.
- **Task:** a unit of work that will be sent to one executor



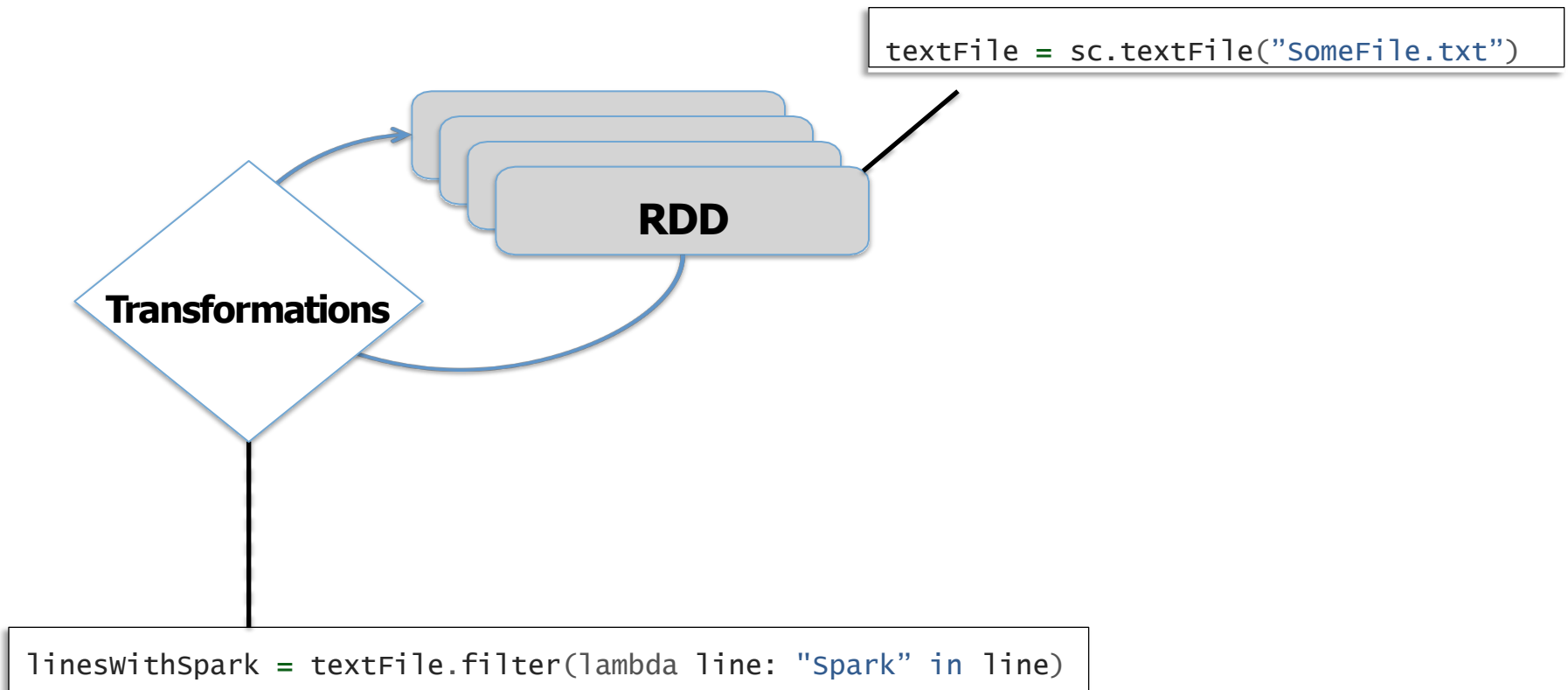
RDD: RESILIENT DISTRIBUTED DATASETS

- “*A fault-tolerant abstraction for in-memory cluster computing*”
- Collection of data items that can be operated on in parallel
 - Transformations
 - Actions
- Fault tolerance: track the series of transformations used to build them (lineage)

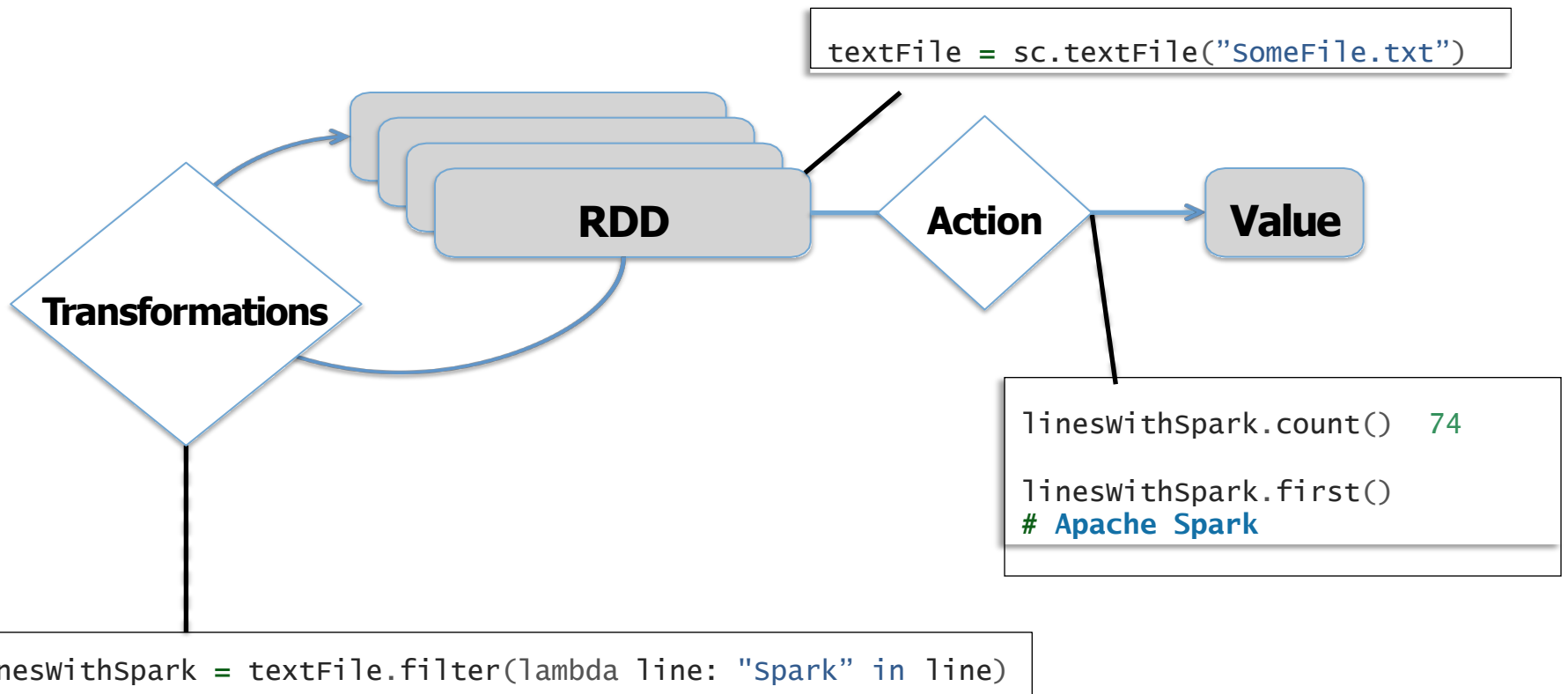
RDD: HOW DOES IT WORK?



RDD: HOW DOES IT WORK?



RDD: HOW DOES IT WORK?



AGENDA

- Introduction
 - Apache Spark
 - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises

HIPERGATOR

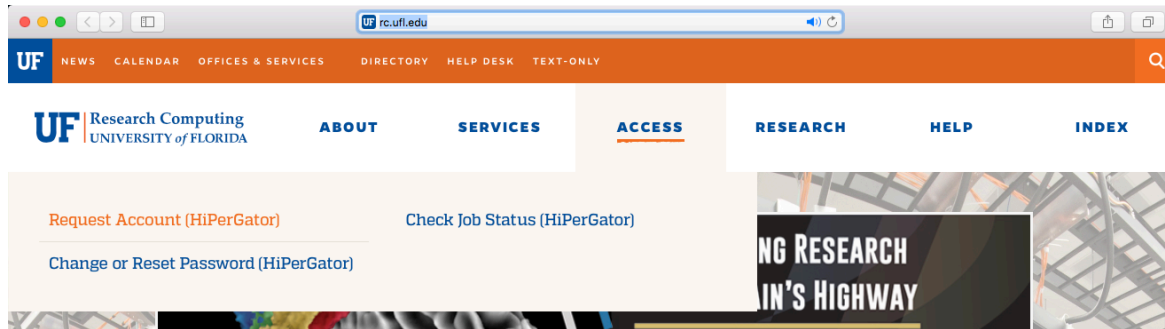


HIPERGATOR LOGISTICS

- **Hardware**
 - Over 50,000 computing cores
 - 3 PB of data storage
 - 180 TB of memory
 - GPU partition
 - Big memory partition
- **Software**
 - Over 1000 software applications installed
 - Covering wide range of research disciplines

HIPERGATOR ACCOUNTS

- Apply for a user account at: <http://rc.ufl.edu>



- Need faculty sponsor
- GatorLink ID

HIPERGATOR ENVIRONMENT

- A Linux-based system
- Interactive session for development and testing
- Production runs handled by job scheduler – **SLURM**



USING HIPERGATOR

- <https://help.rc.ufl.edu>

The screenshot shows a web browser window with the URL help.rc.ufl.edu. The page title is "UFRC Help and Documentation". The main content area is titled "UFRC Help and Documentation" and contains a welcome message and a list of help topics organized into sections. The left sidebar contains navigation and tools links. The main content area is divided into sections: "Getting Started", "Software and Libraries", "Batch System", "Connecting and Data Transfer", and "Specific Research Areas". Each section contains a list of links to specific help topics.

HiPerGator
RESEARCH COMPUTING

Page Discussion Read View source View history Search Go

UFRC Help and Documentation

Welcome to the [University of Florida](#) Research Computing Help and Documentation site. The information here is focused on particular applications, services, and usage examples and complements more general policies and information found on our main web site. It is used for information that changes more frequently and might become quickly dated or incorrect on the web site. This site is edited by individual UFRC staff members. If you find inaccuracies, errors, or omissions on this site please let us know.

Getting Started

- [Getting Started](#)
- [Mailing Lists](#)
- [Events Calendar](#)
- [Training](#)
- [Non-Batch System Resources](#)
- [Interactive Development and Testing](#)
- [GUI Programs](#)
- [SSH Key Creation \(Windows\)](#)
- [Change Your Password](#)
- [Submit a Support Request](#)
- [Storage Types](#)

Software and Libraries

- [Installed Applications](#)
- [Environment Modules System](#)
- [Available Compilers](#)
- [GPU Access](#)

Batch System

- [SLURM Commands](#)
- [Using Variables in SLURM](#)
- [SLURM Job Arrays](#)
- [Account and QOS Limits Under SLURM](#)
- [GUI Partition](#)
- [Big Memory Partition](#)
- [Annotated Job Script Walk-through](#)
- [Sample SLURM Scripts](#)

Connecting and Data Transfer

- [Samba Access](#)
- [Globus](#)
- [Transfer Data](#)

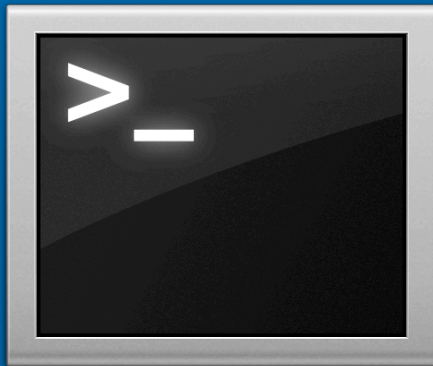
Specific Research Areas

- [Biological Research Computing](#)
- [Bioinformatics Software](#)
- [R \(BioConductor\)](#)
- [Galaxy Genomics Framework](#)
- [Blast Databases](#)

Category: Docs

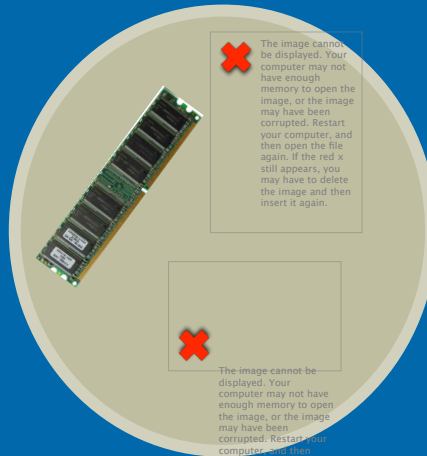
CLUSTER BASICS

User
interaction



Login node
(Head node)

Scheduler



Tell the
scheduler what
you want to do

Compute
resources



Your job
runs on the
cluster

SPARK ON HIPERGATOR

- Version 2.1.0 and 2.2.0
- Programming in **Scala**, **Java**, **Python**, or **R**
- Running standalone Spark jobs via SLURM
- Use spark module

```
module load spark/2.1.0
```

or

```
module load spark/2.2.0
```

- Use programming modules

```
module load scala
```

or

```
module load python (or java, or R)
```

CONNECTING TO HIPERGATOR

- [https://help.rc.ufl.edu/doc/Getting Started](https://help.rc.ufl.edu/doc/Getting_Started)

BREAK!

AGENDA

- Introduction
 - Apache Spark
 - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises

SPARK MODULE IN HIPERGATOR

```
$> module spider spark
```

```
spark:
```

```
Description:
```

```
  general-purpose cluster computing system
```

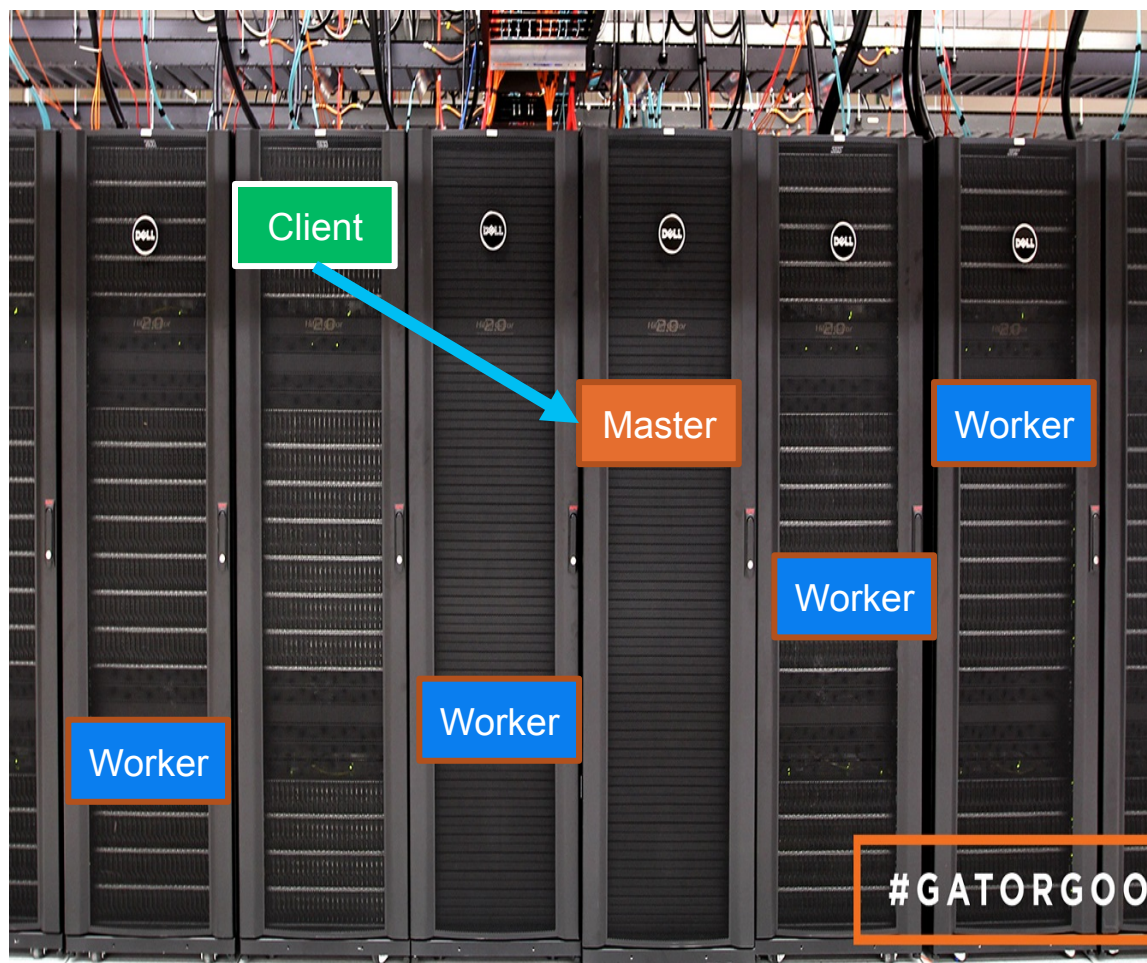
```
Versions:
```

```
  spark/2.1.0
```

```
  spark/2.2.0
```

“NO” SPARK CLUSTER IN HIPERGATOR

- SLURM (resource allocation, job scheduler, workload management) on HiPerGator
- Submit a SLURM job for Spark cluster



SET UP YOUR OWN SPARK CLUSTER

- Set SLURM parameters for Spark Cluster
 - How many nodes? How many CPUs per node? How long ?
 - Amount of memory? Output/error files?

```
#SBATCH --job-name=spark_cluster
#SBATCH --nodes=1 # nodes allocated to the job
#SBATCH --ntasks=1 # tasks to be created for the job
#SBATCH --cpus-per-task=64 # the number of CPUs allocated per task
#SBATCH --exclusive # not sharing the node with other running jobs
#SBATCH --time=03:00:00
#SBATCH --output=spark_cluster.log
#SBATCH --error=spark_cluster.err
##SBATCH --mail-type=ALL
##SBATCH --mail-user=YOUR-EMAIL-ADDRESS
#SBATCH --reservation=spark_workshop
#SBATCH --account=spark_workshop
#SBATCH --qos=spark_workshop
#SBATCH --mem=200gb
```

SET UP YOUR OWN SPARK CLUSTER

- Load spark module

```
module load spark
```

- Set Spark parameters for Spark Cluster
 - What is the working directory?
 - What is the port for communication between components?
 - What is the directory for logfiles?

```
export SPARK_LOCAL_DIRS=$HOME/spark/tmp
export SPARK_WORKER_DIR=$SPARK_LOCAL_DIRS
export SPARK_WORKER_CORES=$SLURM_CPUS_PER_TASK
export SPARK_MASTER_PORT=7077
export SPARK_MASTER_WEBUI_PORT=8080
export SPARK_NO_DAEMONIZE=true
export SPARK_LOG_DIR=$SPARK_LOCAL_DIRS
```


SET UP YOUR OWN SPARK CLUSTER

- Set Spark Master and Workers
 - Spark Master is a daemon for cluster management
 - The master waits for workers to connect with
 - Spark worker is a daemon for a node management
 - The workers need to register to the master

```
# start master
$SPARK_HOME/sbin/start-master.sh &

# start workers

$SPARK_HOME/sbin/start-slave.sh spark://$SPARK_MASTER_NODE:
$SPARK_MASTER_PORT
```

START SPARK CLUSTER ON HIPERGATOR

- Submit the SLURM job script to SLURM
 - Submit the job using “sbatch”
 - The job script, `spark-local-cluster.sh` is provided in `/ufrc/spark_workshop/share`

```
$> sbatch spark-local-cluster.sh
Submitted batch job 27482845
```

- Check your job status using `squeue` command

```
$> squeue -u yingz
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
27482845	hpg1-comp	spark_cl	yingz	R	0:23	1	c2a-s23

DIY1: 1-NODE SPARK CLUSTER

- Step 1: Login to HiPerGator

https://help.rc.ufl.edu/doc/Getting_Started

- Step 2: Copy the files in `/ufrc/spark_workshop/share/` to your directory and edit it

```
$> cp /ufrc/spark_workshop/share/* ~/
$> cd ~/
$> ls
```

- Step 3: Submit the job script to HiPerGator using *sbatch*

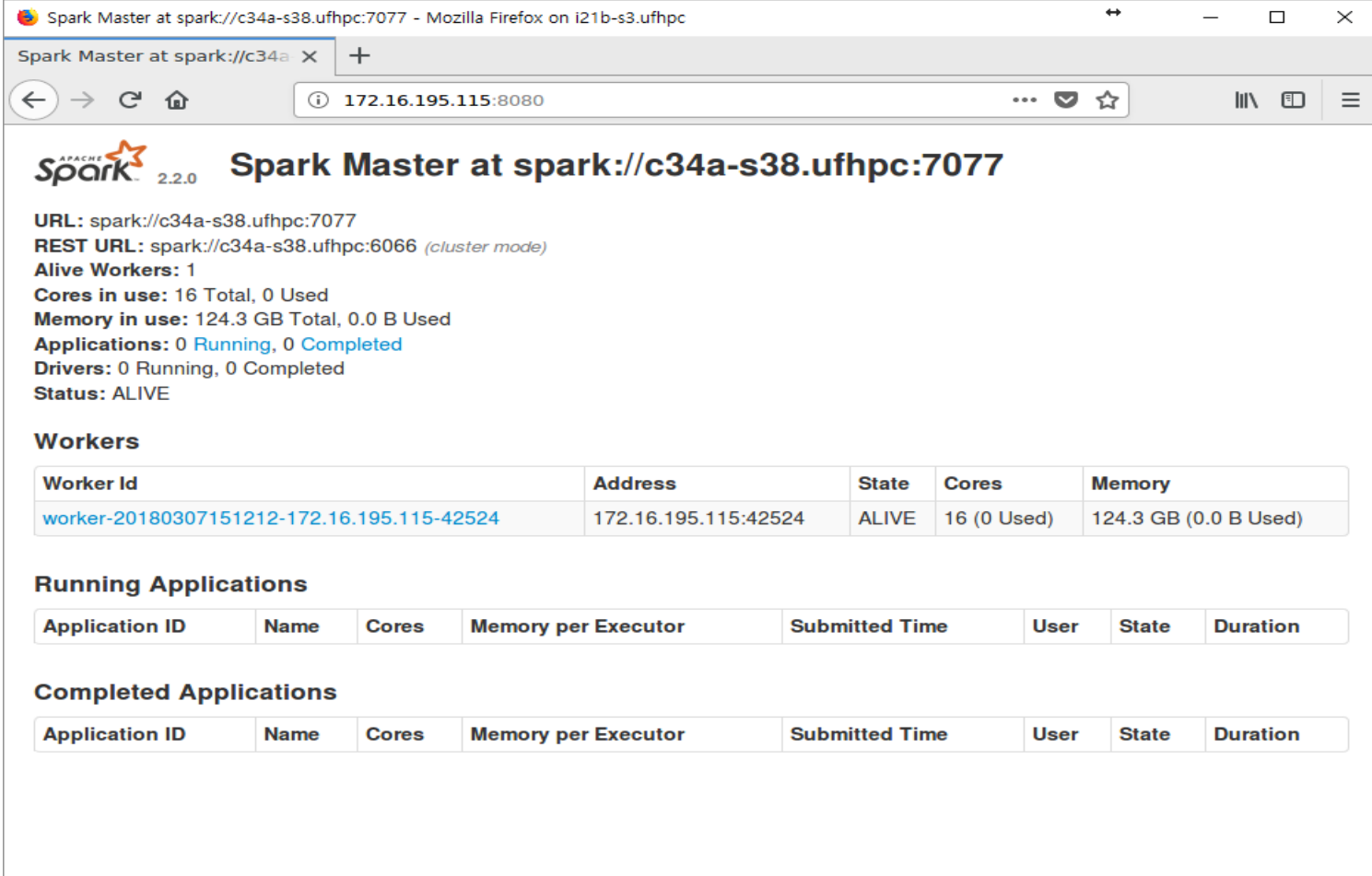
```
[$> sbatch spark-local-cluster.sh
```

- Step 4: Check the status of your job using *squeue*

```
[$> squeue -u <your ID>
```

DIY2: SPARK CLUSTER MONITORING

- Spark provides a web-interface to monitor its resource usage and job histories



The screenshot shows the Spark Master web interface in a Mozilla Firefox browser. The page title is "Spark Master at spark://c34a-s38.ufhpc:7077". The interface displays the following information:

- URL:** spark://c34a-s38.ufhpc:7077
- REST URL:** spark://c34a-s38.ufhpc:6066 (cluster mode)
- Alive Workers:** 1
- Cores in use:** 16 Total, 0 Used
- Memory in use:** 124.3 GB Total, 0.0 B Used
- Applications:** 0 Running, 0 Completed
- Drivers:** 0 Running, 0 Completed
- Status:** ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20180307151212-172.16.195.115-42524	172.16.195.115:42524	ALIVE	16 (0 Used)	124.3 GB (0.0 B Used)

Running Applications

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

Completed Applications

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

DIY2: SPARK CLUSTER MONITORING

- Get the IP address for the web interface of the master node

```
$> grep MasterWebUI spark_cluster.err  
18/11/04 11:50:37 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0,  
and started at http://172.16.198.119:8080
```

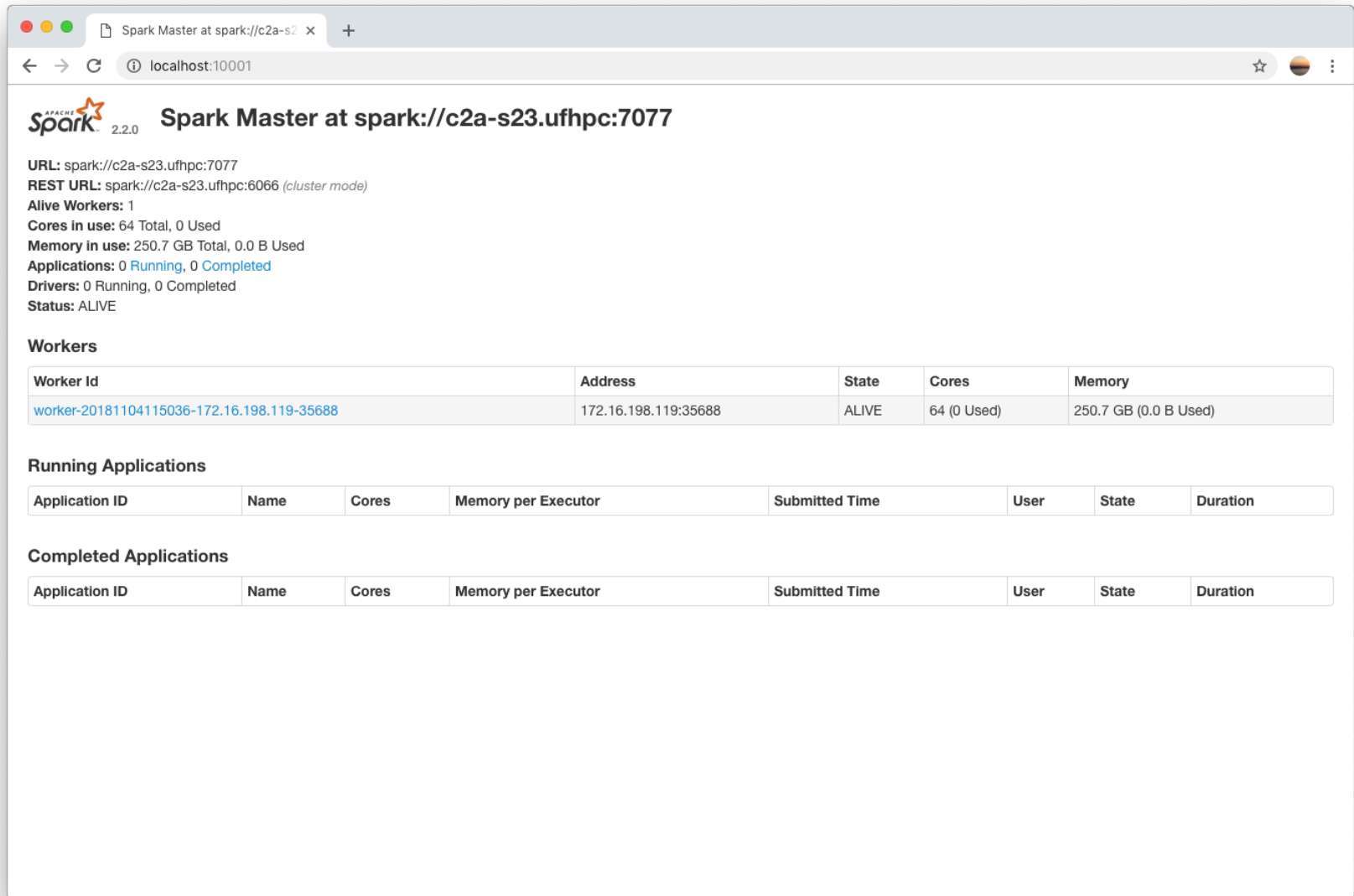
- Open a new terminal on your laptop. In the new terminal, type

```
ssh -L 10001:172.16.198.119:8080 your_ID@hpc.rc.ufl.edu
```

- On your laptop, open a browser, and type the following web address

localhost:10001

DIY2: SPARK CLUSTER MONITORING



The screenshot shows a web browser window displaying the Spark Master monitoring interface. The browser's address bar shows 'localhost:10001'. The page title is 'Spark Master at spark://c2a-s23.ufhpc:7077'. The interface includes the Apache Spark logo and version 2.2.0. Key metrics are listed: URL, REST URL, 1 alive worker, 64 total cores (0 used), 250.7 GB total memory (0.0 B used), 0 running applications, 0 completed applications, 0 running drivers, and 0 completed drivers. The status is 'ALIVE'. Below the metrics are three sections: 'Workers' with a table showing one worker, 'Running Applications' with an empty table, and 'Completed Applications' with an empty table.

Spark Master at spark://c2a-s23.ufhpc:7077

URL: spark://c2a-s23.ufhpc:7077
REST URL: spark://c2a-s23.ufhpc:6066 (cluster mode)
Alive Workers: 1
Cores in use: 64 Total, 0 Used
Memory in use: 250.7 GB Total, 0.0 B Used
Applications: 0 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20181104115036-172.16.198.119-35688	172.16.198.119:35688	ALIVE	64 (0 Used)	250.7 GB (0.0 B Used)

Running Applications

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

Completed Applications

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

BREAK!

SPARK INTERACTIVE SHELLS - SCALA

- To use *Scala*, load *spark* module

```
$> module load spark
$> module list
```

Currently Loaded Modules:

```
 1) java/1.8.0_31      4) libgdal/1.11.1    7) glpk/4.55
10) geos/3.3.8        13) scala/2.11.5
 2) netcdf/4.2         5) libxslt/1.1.28   8) lzo/2.09
11) libmpfr/3.1.1     14) spark/2.2.0
 3) hdf5/1.8.9        6) libxml2/2.9.1    9) mkl/2013.sp1.3.174
12) python/2.7.6 (H)
```

Where:

H: Hidden Module

DIY3: PI ESTIMATION VIA INTERACTIVE SHELL - PYTHON

- Estimate Pi (π) by "throwing darts" at a circle. Points in the unit square ((0, 0) to (1,1)) are randomly picked and observed how many fall in the unit circle. The fraction should be $\pi / 4$, so this is used to get the estimation.

DIY3: PI ESTIMATION VIA INTERACTIVE SHELL - PYTHON

```
from operator import add
from random import random

partitions = 10
n = 100000 * partitions

def f(_):
    x = random() * 2 - 1
    y = random() * 2 - 1
    return 1 if x ** 2 + y ** 2 <= 1 else 0

count = sc.parallelize(range(1, n + 1), partitions).map(f).reduce(add)

print("Pi is roughly %f" % (4.0 * count / n))
```

SPARK INTERACTIVE SHELLS - PYTHON

- Spark interactive shell in Python
 - `$> pyspark --master $SPARK_MASTER`

```
$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
$> pyspark --master $SPARK_MASTER
```

```
<<omitted>>
```

```
Welcome to
```

```
  ____
 /  _ \ /  _ \ /  _ \ /  _ \
 \  __/ \  __/ \  __/ \  __/
  \___/  \___/  \___/  \___/   version 2.2.0
  /  _ \
 /  _ \
 \  __/
  \___/
```

```
Using Python version 2.7.6 (default, Feb 5 2014 11:52:59)
```

```
SparkSession available as 'spark'.
```

```
>>>
```

DIY3: PI ESTIMATION VIA INTERACTIVE SHELL IN PYTHON

- Start Spark interactive shell in Python (pyspark)

```
[$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
[$> pyspark --master $SPARK_MASTER
```

```
Welcome to
```

```
  /_/_/  _/_/_/  /_/_/  /_/_/
 _\ \ \  _\ \ \  /_/_/  /_/_/
/_/_/  .\_/_/  /_/_/  /_/_/  \_/_/
  /_/_/
version 2.2.0
```

```
Using Python version 2.7.6 (default, Feb 5 2014 11:52:59)
```

```
SparkSession available as 'spark'.
```

```
>>> from random import random
```

```
>>> from operator import add
```

```
>>> partitions = 100
```

```
>>> n = 100000 * partitions
```

```
>>> def f(_):
```

```
...     x = random() * 2 - 1
```

```
...     y = random() * 2 - 1
```

```
...     return 1 if x ** 2 + y ** 2 <= 1 else 0
```

```
... 
```

```
>>>
```

```
>>> count = sc.parallelize(range(1, n + 1), partitions).map(f).reduce(add)
```

```
18/11/04 14:05:01 WARN TaskSetManager: Stage 0 contains a task of very large size (363 KB).
```

```
The maximum recommended task size is 100 KB.
```

```
>>> print("Pi is roughly %f" % (4.0 * count / n))
```

```
Pi is roughly 3.140654
```

DIY4: PI ESTIMATION FROM FILE WITH PYSPARK

- As of Spark 2.0, Python scripts can not be loaded directly to Spark interactive shell.
- Execute Python script via *pyspark* command line:
 - Set “PYTHONSTARTUP”, a python environmental variable.

```
$> PYTHONSTARTUP=diy4.py pyspark --master $SPARK_MASTER
```

DIY4: PI ESTIMATION FROM FILE WITH PYSPARK

```
[$> cp /ufrc/spark_workshop/share/diy4.py .  
[$> cat diy4.py  
  
from random import random  
def inside(p):  
    x, y = random(), random()  
    return x*x + y*y < 1  
  
NUM_SAMPLES = 1000000  
  
count = sc.parallelize(xrange(0, NUM_SAMPLES)).filter(inside).count()  
print "Pi is roughly %f" % (4.0 * count / NUM_SAMPLES)
```

DIY4: PI ESTIMATION FROM FILE WITH PYSPARK

```
$> module load spark
$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
$> PYTHONSTARTUP=diy4.py pyspark --master $SPARK_MASTER
Picked up _JAVA_OPTIONS:
Python 2.7.6 (default, Feb 5 2014, 11:52:59)
```

<< Omitted lines >>

```
Welcome to
```

```
  _____
 /  _  /  _  _  _  _  /  /  _
- \  \  -  \  -  \  /  _  /  \
/_  /  .  /  \  ,  /  /  /  \  \
  /  /
version 2.2.0
```

```
Using Python version 2.7.6 (default, Feb 5 2014 11:52:59)
SparkSession available as 'spark'.
Pi is roughly 3.147776
```


SUBMIT SPARK JOBS VIA SPARK-SUBMIT

- A script which provides unified interface for Spark jobs

```
./bin/spark-submit \  
  --class <main-class> --master <master-url> \  
  --deploy-mode <deploy-mode> --conf <key>=<value> \  
  ... # other options <application-jar> [application-arguments]
```

- `--class`: The entry point for your application (e.g. `org.apache.spark.examples.SparkPi`)
- `--master`: The [master URL](#) for the cluster (e.g. `spark://123.45.67.890:7077`)
- `--deploy-mode`: Whether to deploy your driver on the worker nodes (cluster) or locally as an external client (client) (default: client)
- `--conf`: Arbitrary Spark configuration property in key=value format. For values that contain spaces wrap “key=value” in quotes (as shown).
- `<application-jar>`: Path to a bundled jar including your application and all dependencies. The URL must be globally visible inside of your cluster, for instance, an `hdfs://` path or a `file://` path that is present on all nodes.
- `<application-arguments>`: Arguments passed to the main method of your main class, if any
- For further details about spark-submit, refer to <https://spark.apache.org/docs/2.2.0/submitting-applications.html>.

DIY5: PI ESTIMATION USING SPARK-SUBMIT

```
[$> module load spark
[$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
[$> spark-submit --master $SPARK_MASTER $SPARK_HOME/examples/src/main/python/pi.py 10

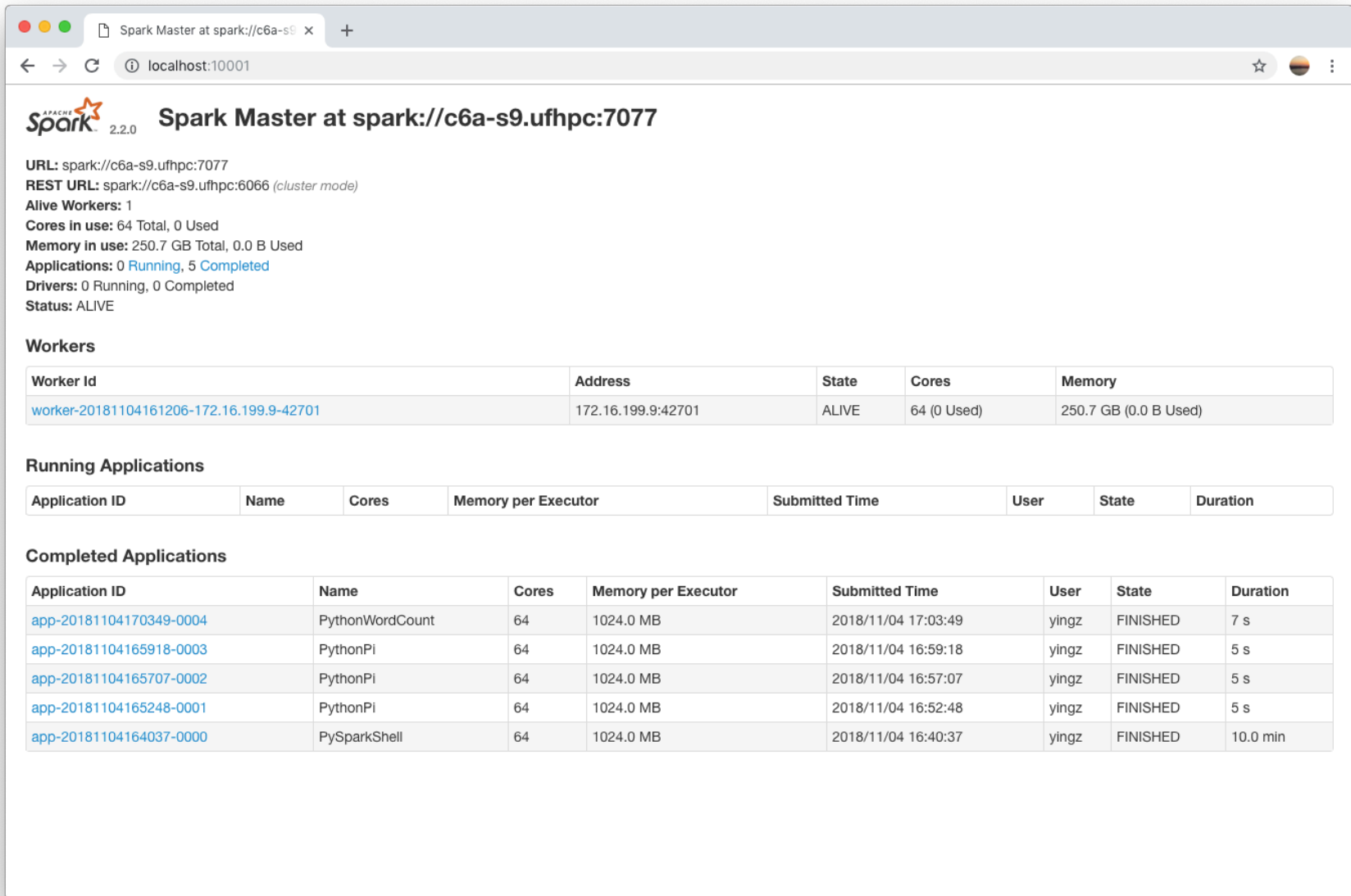
18/11/04 16:59:23 INFO TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 1975
ms on 172.16.199.9 (executor 0) (10/10)
18/11/04 16:59:23 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all c
ompleted, from pool
18/11/04 16:59:23 INFO DAGScheduler: ResultStage 0 (reduce at /apps/spark/2.2.0-hadoo
p2.7/examples/src/main/python/pi.py:43) finished in 3.970 s
18/11/04 16:59:23 INFO DAGScheduler: Job 0 finished: reduce at /apps/spark/2.2.0-hado
op2.7/examples/src/main/python/pi.py:43, took 4.124699 s
Pi is roughly 3.143048
18/11/04 16:59:23 INFO SparkUI: Stopped Spark web UI at http://172.16.206.9:4040
18/11/04 16:59:23 INFO StandaloneSchedulerBackend: Shutting down all executors
18/11/04 16:59:23 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each exec
utor to shut down
18/11/04 16:59:23 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint
stopped!
18/11/04 16:59:23 INFO MemoryStore: MemoryStore cleared
18/11/04 16:59:23 INFO BlockManager: BlockManager stopped
18/11/04 16:59:23 INFO BlockManagerMaster: BlockManagerMaster stopped
18/11/04 16:59:23 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: Outpu
tCommitCoordinator stopped!
18/11/04 16:59:23 WARN Dispatcher: Message RemoteProcessDisconnected(172.16.199.9:543
28) dropped. RpcEnv already stopped.
18/11/04 16:59:23 WARN Dispatcher: Message RemoteProcessDisconnected(172.16.199.9:543
28) dropped. RpcEnv already stopped.
```

DIY6: WORDCOUNT USING SPARK-SUBMIT

```
$> module load spark
$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
$> spark-submit --master $SPARK_MASTER $SPARK_HOME/examples/src/main/python/wordcount.py
spark_cluster.err > wc.result
```

```
[$> cat wc.result
: 33
command:: 5
app-20181104165248-0001: 5
Master:: 40
Unable: 2
kill: 4
ALIVE: 1
ExecutorRunner:: 13
/apps/spark/2.2.0-hadoop2.7: 1
41: 1
bootstraps): 1
'sparkWorker': 1
cleanupLocalDirs: 4
172.16.199.9:7077...: 1
NativeCodeLoader:: 2
app-20181104164037-0000: 5
with: 42
0.0.0.0,: 2
RAM: 2
Set(yingz);: 14
state: 4
created: 1
PythonWordCount: 3
```

SPARK JOB HISTORY



The screenshot shows the Spark Master web interface. At the top, the browser address bar shows 'localhost:10001'. The page title is 'Spark Master at spark://c6a-s9.ufhpc:7077'. Below the title, there is a summary of cluster status: URL, REST URL, Alive Workers (1), Cores in use (64 Total, 0 Used), Memory in use (250.7 GB Total, 0.0 B Used), Applications (0 Running, 5 Completed), Drivers (0 Running, 0 Completed), and Status (ALIVE). The 'Workers' section contains a table with one worker. The 'Running Applications' section is empty. The 'Completed Applications' section contains a table with five rows of job history.

Spark Master at spark://c6a-s9.ufhpc:7077

URL: spark://c6a-s9.ufhpc:7077
REST URL: spark://c6a-s9.ufhpc:6066 (cluster mode)
Alive Workers: 1
Cores in use: 64 Total, 0 Used
Memory in use: 250.7 GB Total, 0.0 B Used
Applications: 0 Running, 5 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers

Worker Id	Address	State	Cores	Memory
worker-20181104161206-172.16.199.9-42701	172.16.199.9:42701	ALIVE	64 (0 Used)	250.7 GB (0.0 B Used)

Running Applications

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

Completed Applications

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20181104170349-0004	PythonWordCount	64	1024.0 MB	2018/11/04 17:03:49	yingz	FINISHED	7 s
app-20181104165918-0003	PythonPi	64	1024.0 MB	2018/11/04 16:59:18	yingz	FINISHED	5 s
app-20181104165707-0002	PythonPi	64	1024.0 MB	2018/11/04 16:57:07	yingz	FINISHED	5 s
app-20181104165248-0001	PythonPi	64	1024.0 MB	2018/11/04 16:52:48	yingz	FINISHED	5 s
app-20181104164037-0000	PySparkShell	64	1024.0 MB	2018/11/04 16:40:37	yingz	FINISHED	10.0 min

ADVANCED TOPICS

- Deep learning with TensorFlow on Apache Spark
 - <https://databricks.com/blog/2016/01/25/deep-learning-with-apache-spark-and-tensorflow.html>
- Genome analysis with ADAM and Apache Spark
 - <https://github.com/bigdatagenomics/adam>
- GPU acceleration on Apache Spark
 - <http://www.spark.tc/gpu-acceleration-on-apache-spark-2/>
- RDMA (remote direct memory access)-based Apache Spark
 - <http://hibd.cse.ohio-state.edu/#spark>
- Etc.