SPARK ON HIPERGATOR

Ying Zhang
yingz@ufl.edu

September 18th, 2018

UF | Information Technology

HiPerGator
The University of Florida Supercomputer for Research

RESEARCH COMPUTING STAFF

- Dr. Matt Gizendanner
  - Bioinformatics Specialist
- Maksym Prokopenko
  - Application 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:

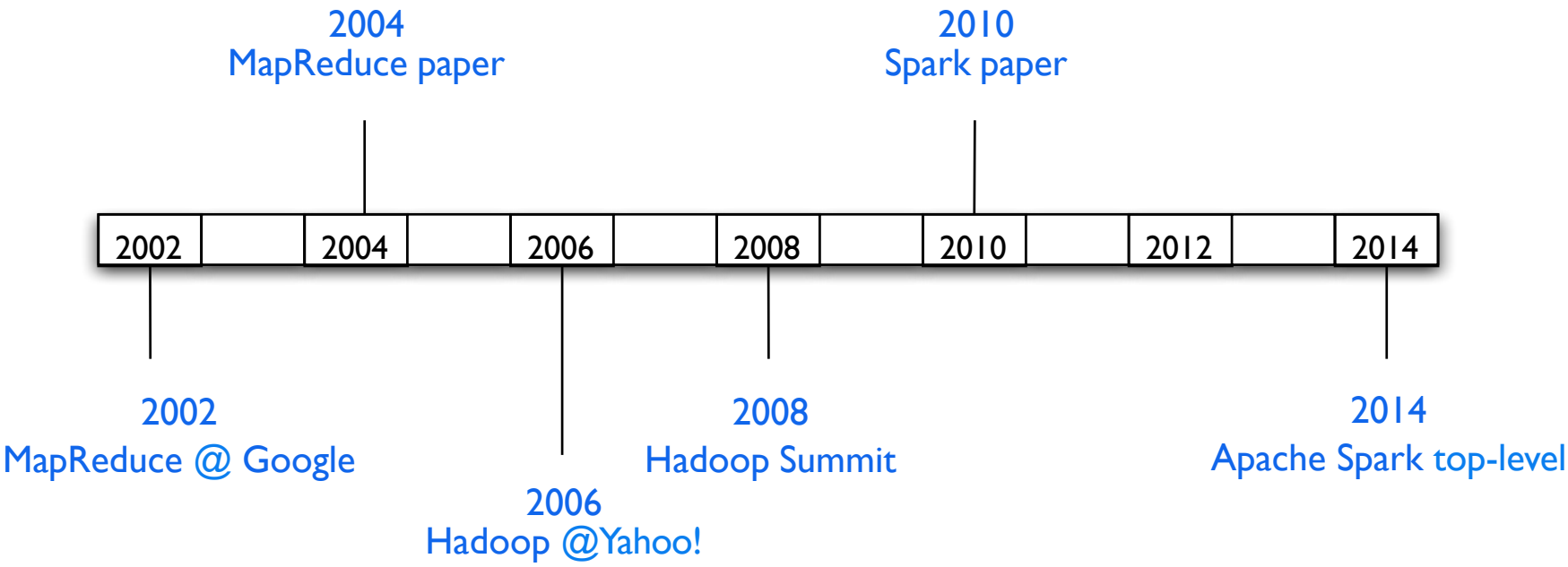
  https://help.rc.ufl.edu/doc/Spark_Workshop

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:
  - *Map* function:

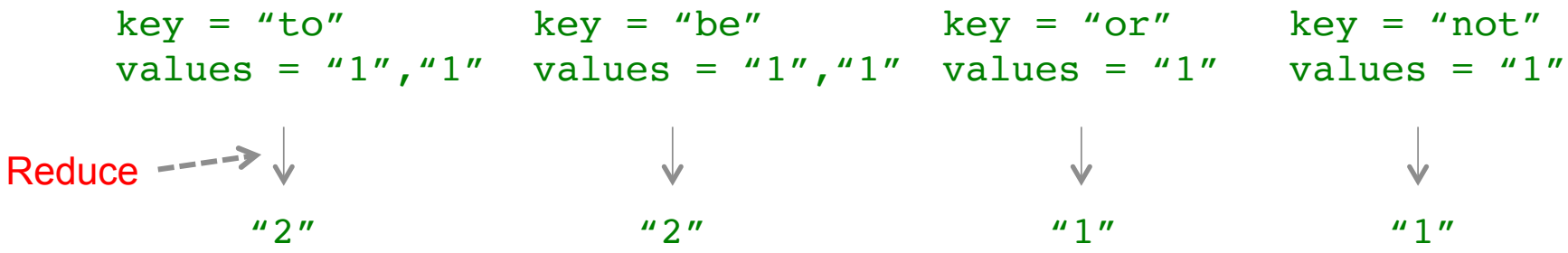
    "to be or not to be"

    - key/value pairs

      "to", "1"
      "be", "1"
      "or", "1"
      "not", "1"
      "to", "1"
      "be", "1"

MAP-REDUCE

- Word count example: *"to be or not to be"*
  - *Shuffle/sort*: gathers all pairs with the same key value
  - *Reduce* function combines the values for a key



- Output:

```
"to", "2"
"be", "2"
"or", "1"
"not", "1"
```

MAP-REDUCE

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

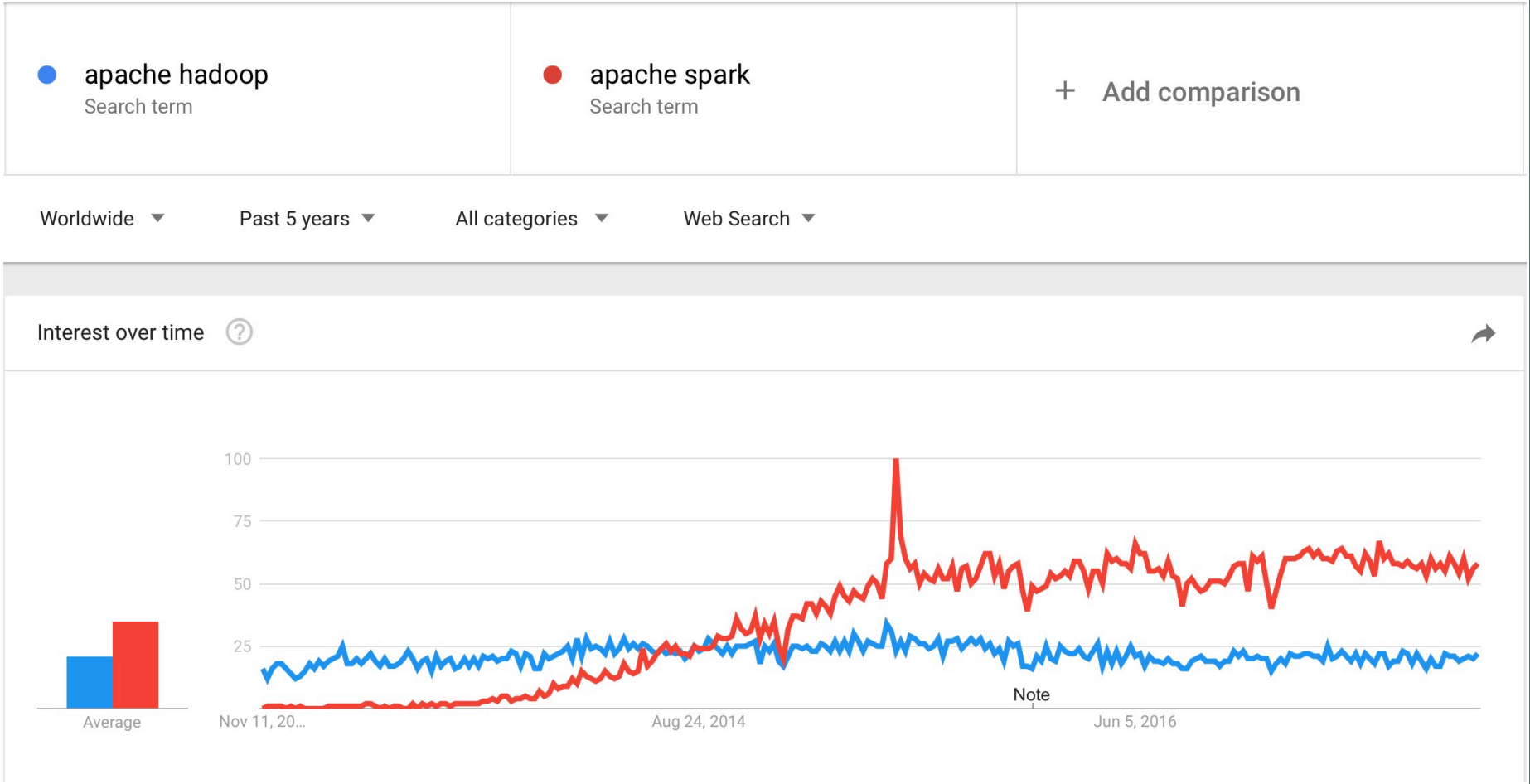
HADOOP & SPARK

- 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



SPARK PROGRAMMABILITY

WordCount in 50+ lines of Java

```
public class WordCount {
  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = new Job(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

WordCount in 3 lines of Spark Scala

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

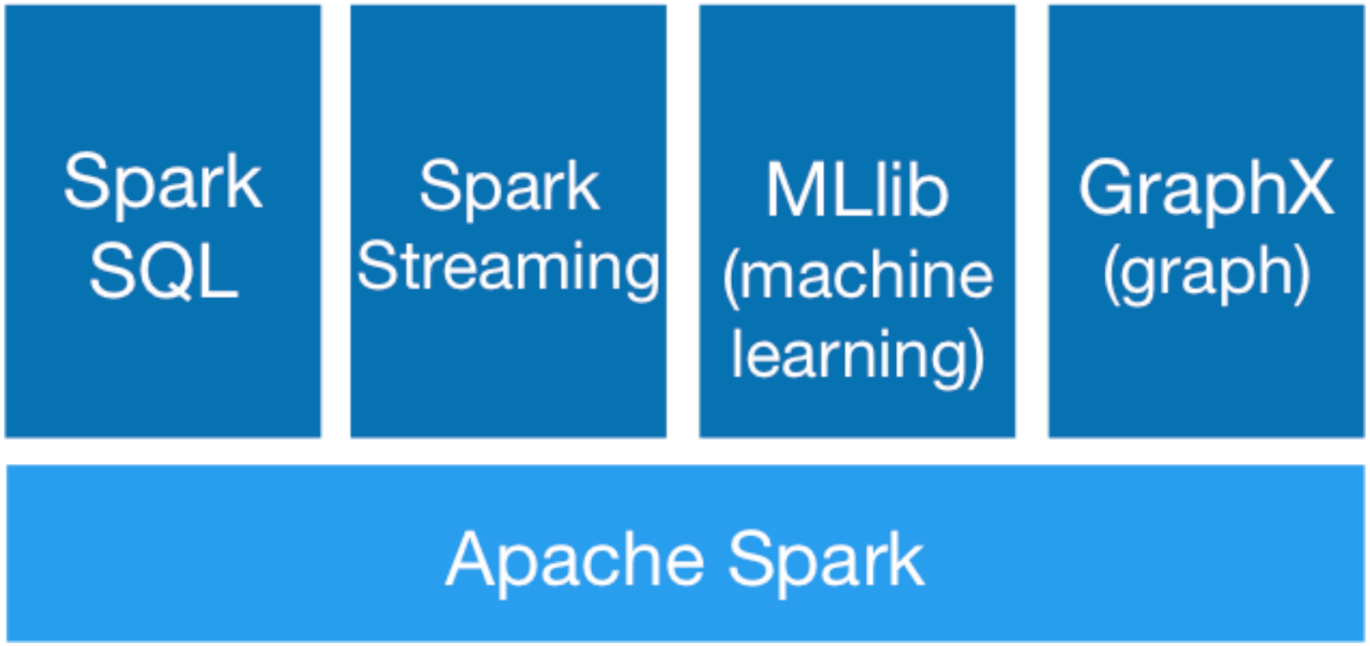
SPARK PERFORMANCE

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

Source: Daytona GraySort Competition 2014, https://databricks.com

SPARK ECOSYSTEM

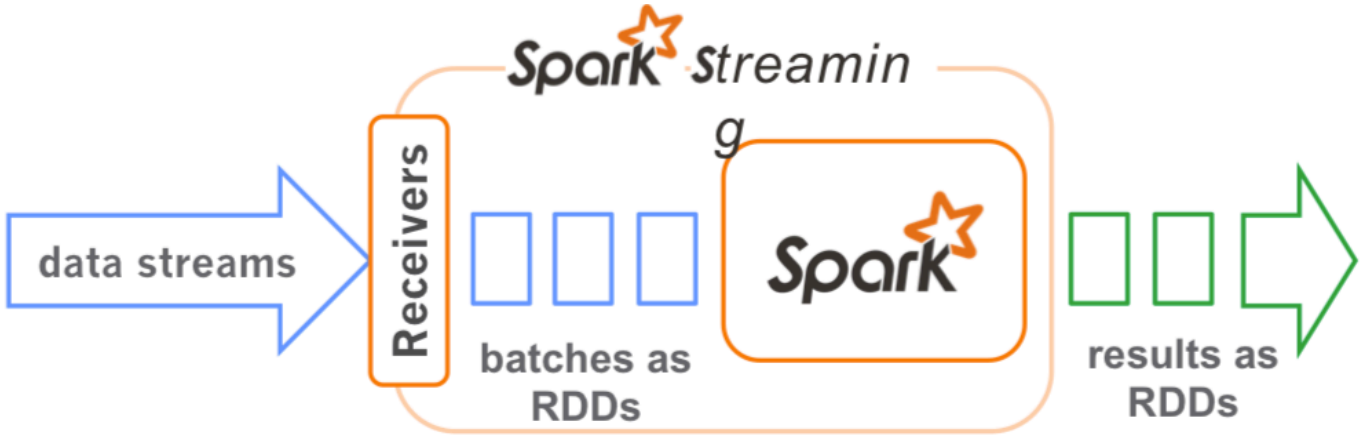


SPARK SQL AND DATAFRAMES

- 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 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: logistic regression, 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

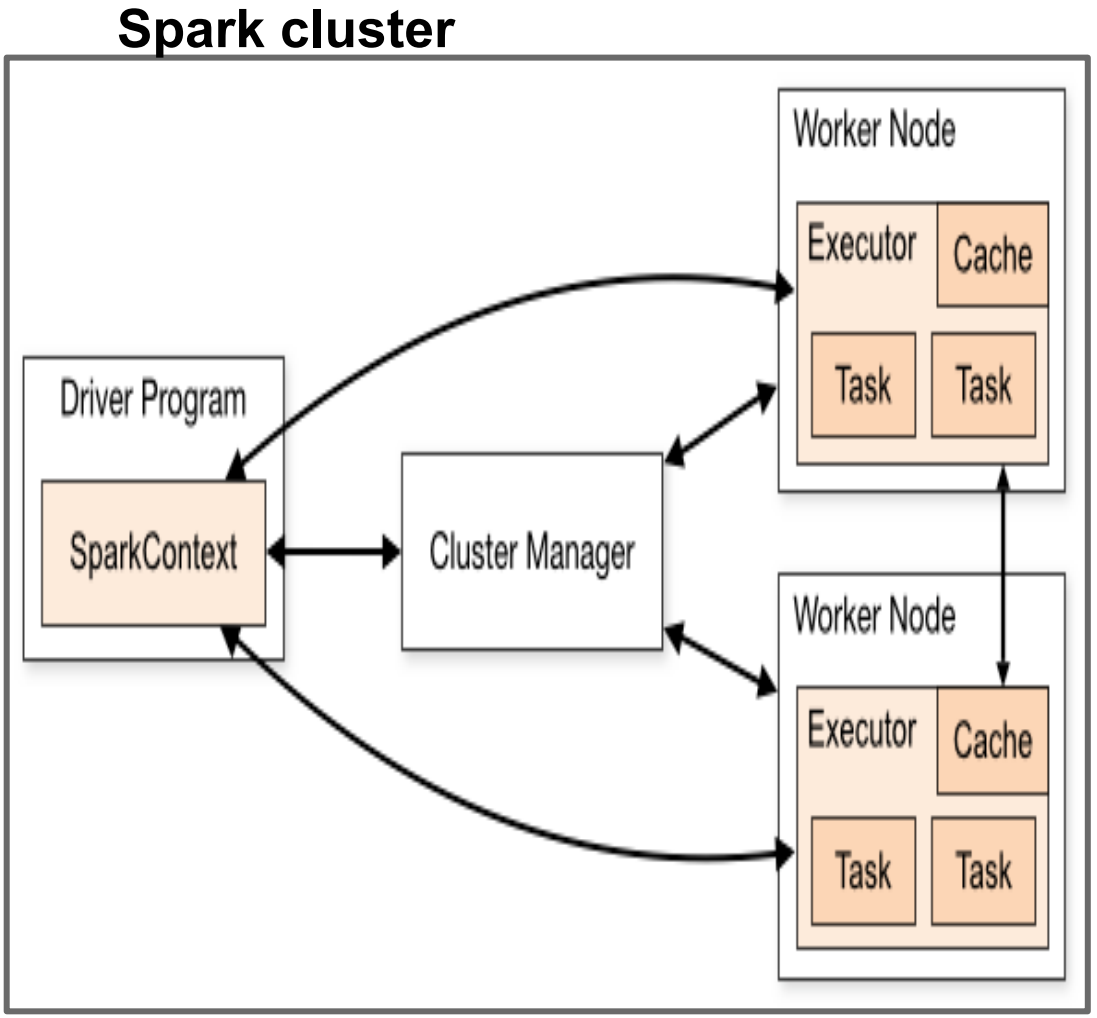
- Graph analytics
  - Examples: social networks, page rank, fraud detection, etc.
  - Graph data modeling
  - Graph data processing
- GraphX
  - API for graphs and graph-parallel computation
  - A growing library of graph algorithms
  - Performance comparable to the fastest specialized graph processing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

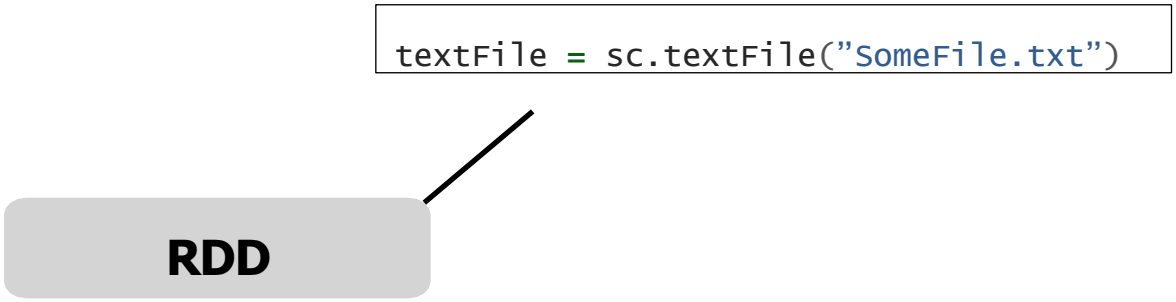
- **Cluster manager (master)**: resource manager (standalone manager)
- **Worker node**: any node running application.
- **Application**: user program built on Spark. Driver program + executors
- **Driver program**: process running the main() function of the application
- **Executor**: process launched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



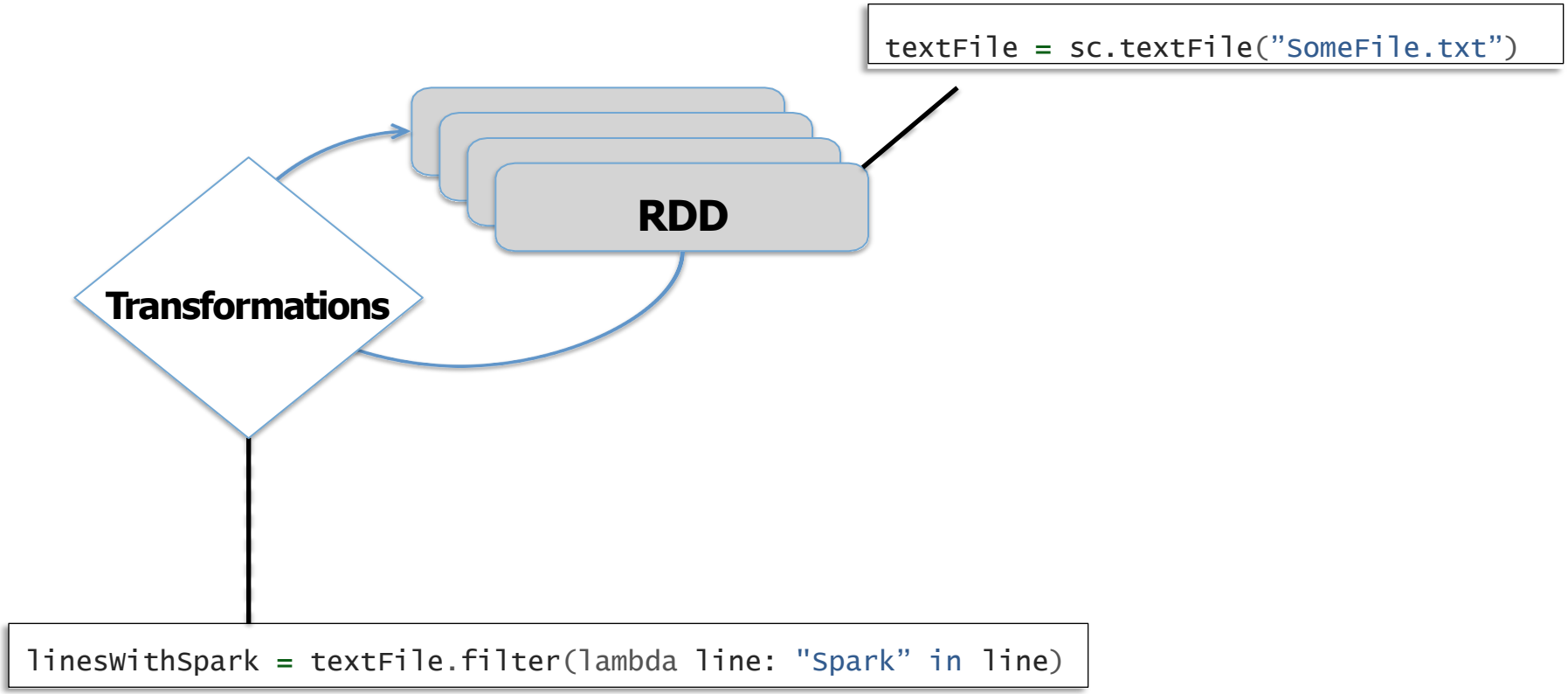
RDD: RESILIENT DISTRIBUTED DATASETS

- "*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)

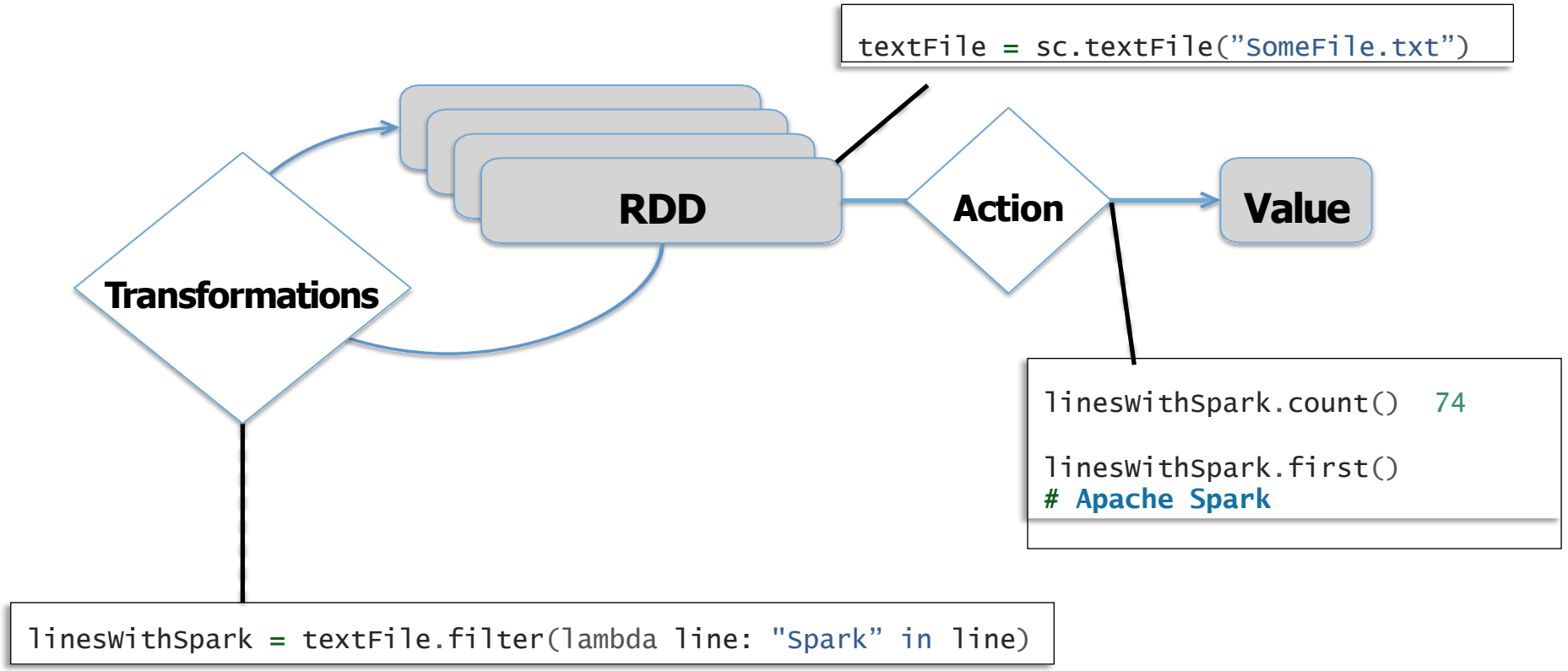
RDD: HOW DOES IT WORK?



RDD: HOW DOES IT WORK?



RDD: HOW DOES IT WORK?



AGENDA

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

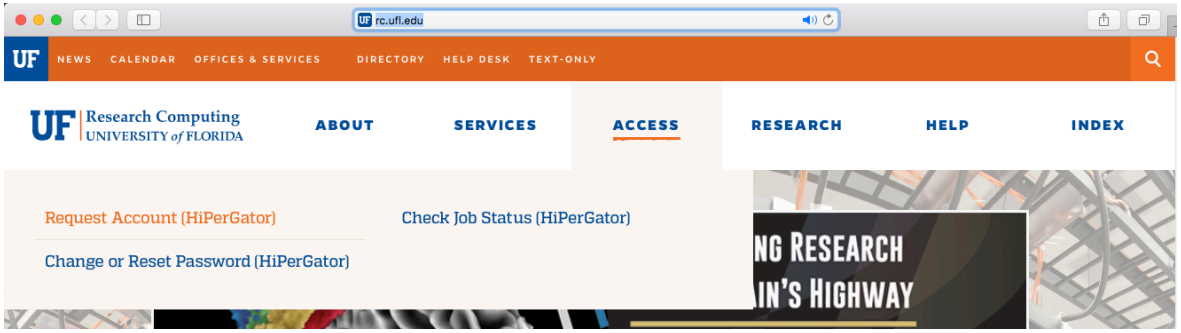
HIPERGATOR

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



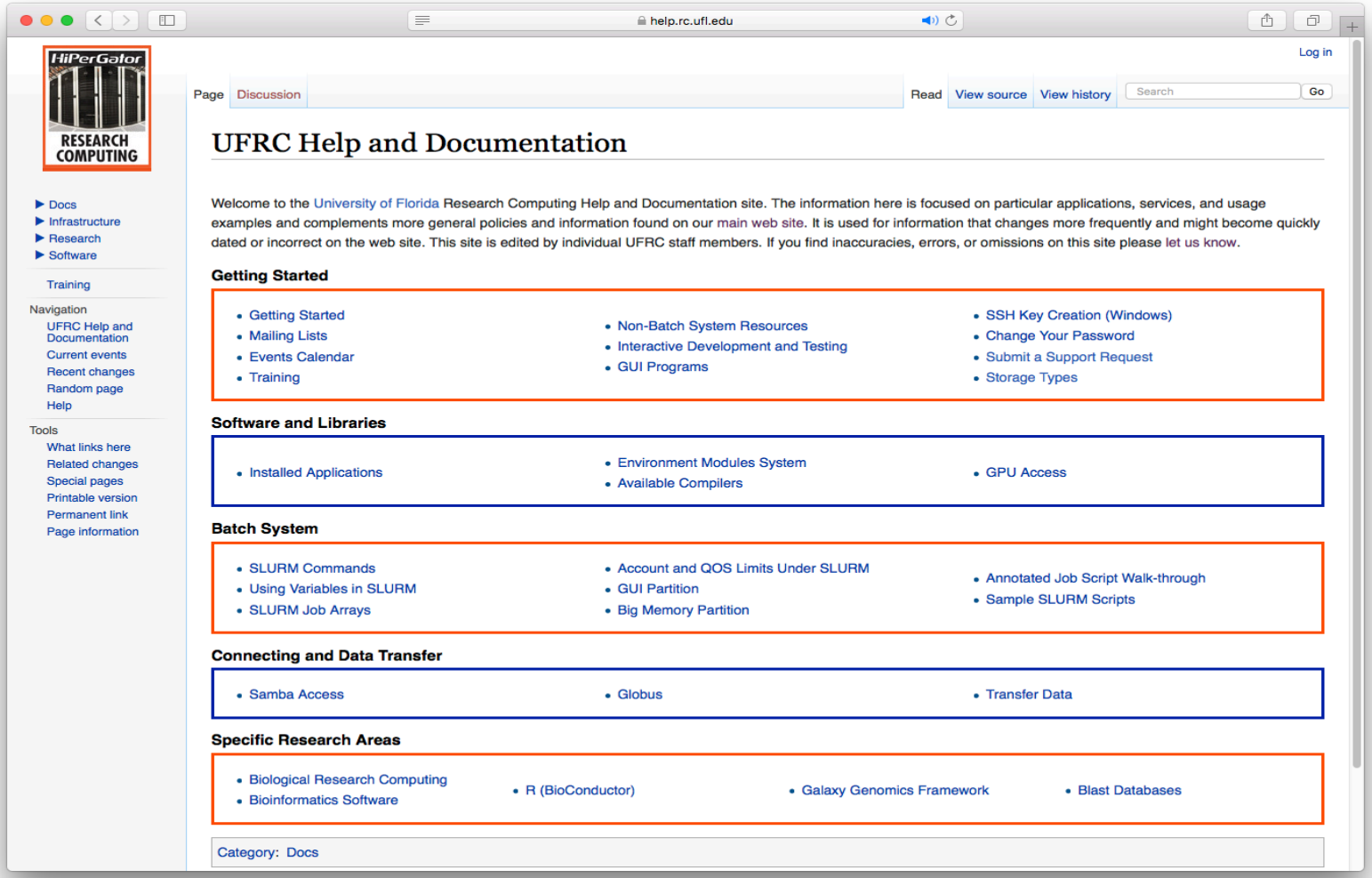
  - Need faculty sponsor
  - GatorLink ID

HIPERGATOR ENVIRONMENT

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

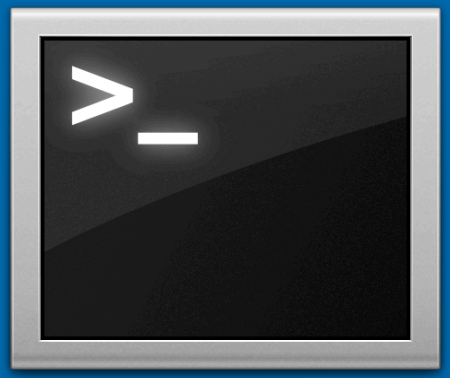
USING HIPERGATOR

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



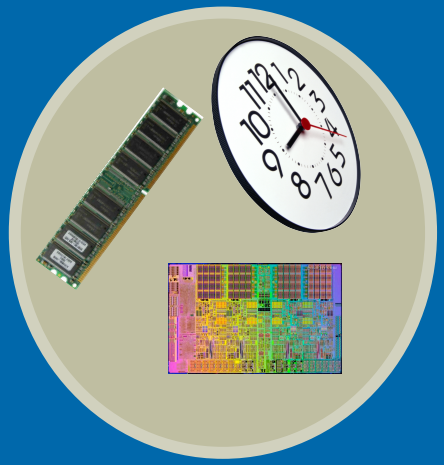
CLUSTER BASICS

User interaction



Login node (Head node)

Scheduler



Tell the scheduler what you want to do

Compute resources

Your job runs on the cluster

Source: Matt Gitzendanner, "Intro to Research Computing and HiPerGator"

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

BREAK!

AGENDA

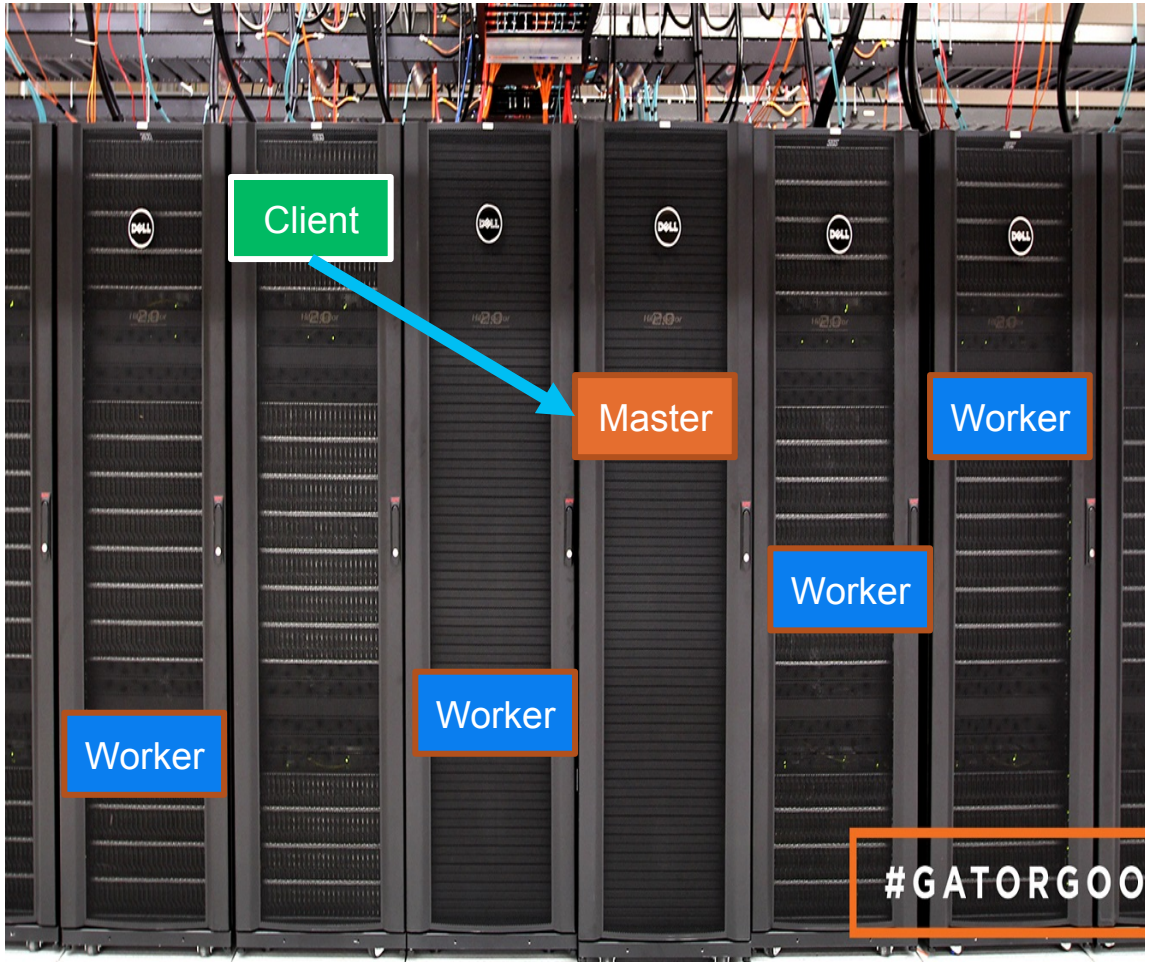
- Introduction
  - Apache Spark
  - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises

SPARK MODULE IN HIPERGATOR

```
$> ml spider spark

----------------------------------------------------------------------------------------------------
  spark:
----------------------------------------------------------------------------------------------------
    Description:
      general-purpose cluster computing system

     Versions:
        spark/2.1.0
        spark/2.2.0

----------------------------------------------------------------------------------------------------

<<omitted>>
```

"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 ?
  - ...

```
#SBATCH --job-name=spark_cluster
#SBATCH --nodes=1
#SBATCH --cpus-per-task=64
#SBATCH --exclusive # not sharing with other running jobs
#SBATCH --time=03:00:00
#SBATCH --output=spark_cluster.log
#SBATCH --error=spark_cluster.err
…
```

SET UP YOUR OWN SPARK CLUSTER

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

START SPARK CLUSTER ON HIPERGATOR

- Submit the SLURM job script to SLURM
  - "sbatch" is used to submit the script
    - The script, spark-local-cluster.sh is provided in /ufrc/spark_workshop/shared
  - "squeue" is used to check your job status

```
$> sbatch spark-local-cluster.sh
Submitted batch job 18070836

$> squeue –u giljael
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      18070836 hpg2-comp spark_cl  giljael  R       0:04      1 c29a-s42
```

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/shared/ to your directory and edit it

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

- Step 3: Submit the script to HiPerGator using sbatch

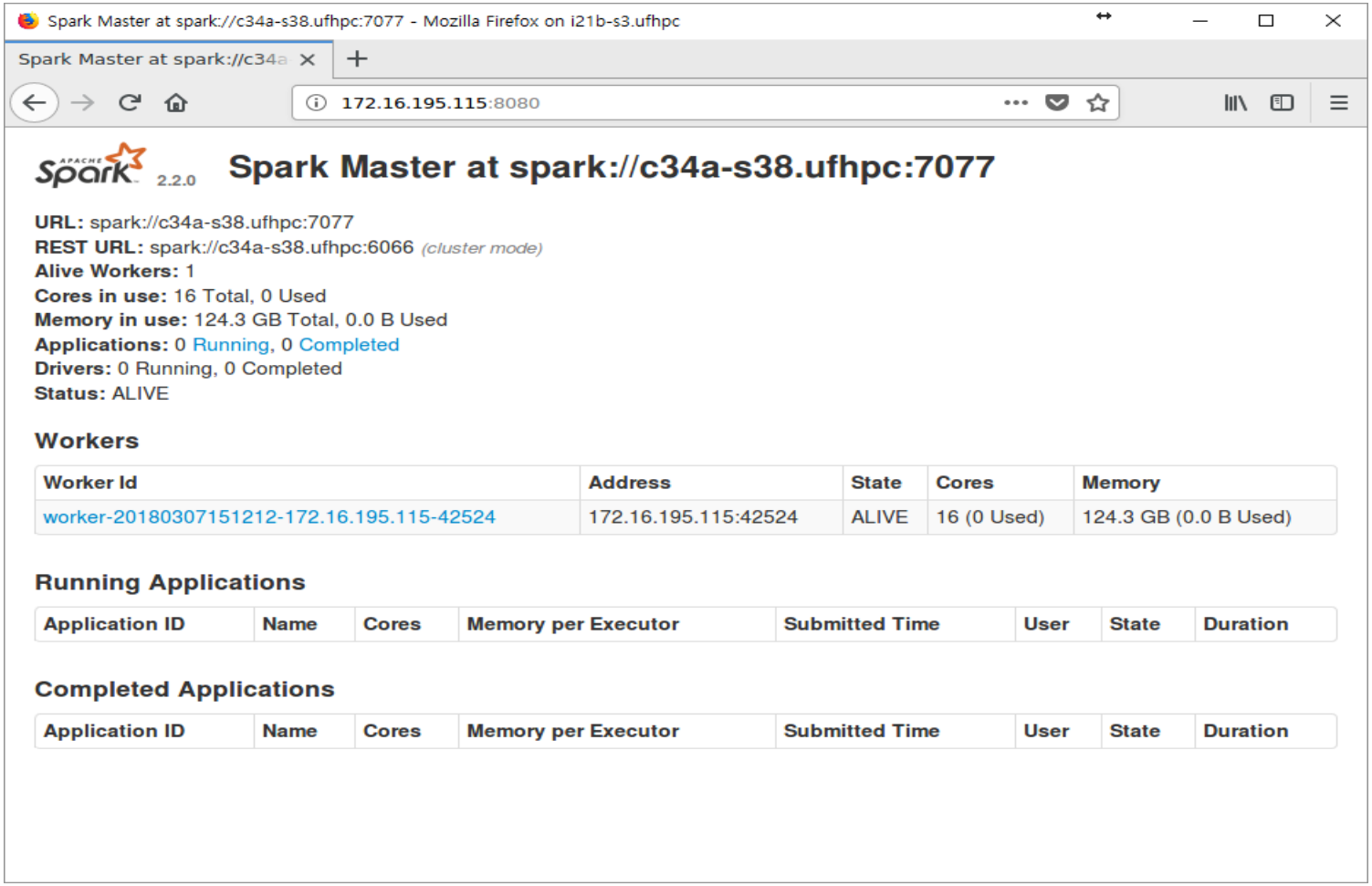
```
$> sbatch spark-local-cluster.sh
```

- 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

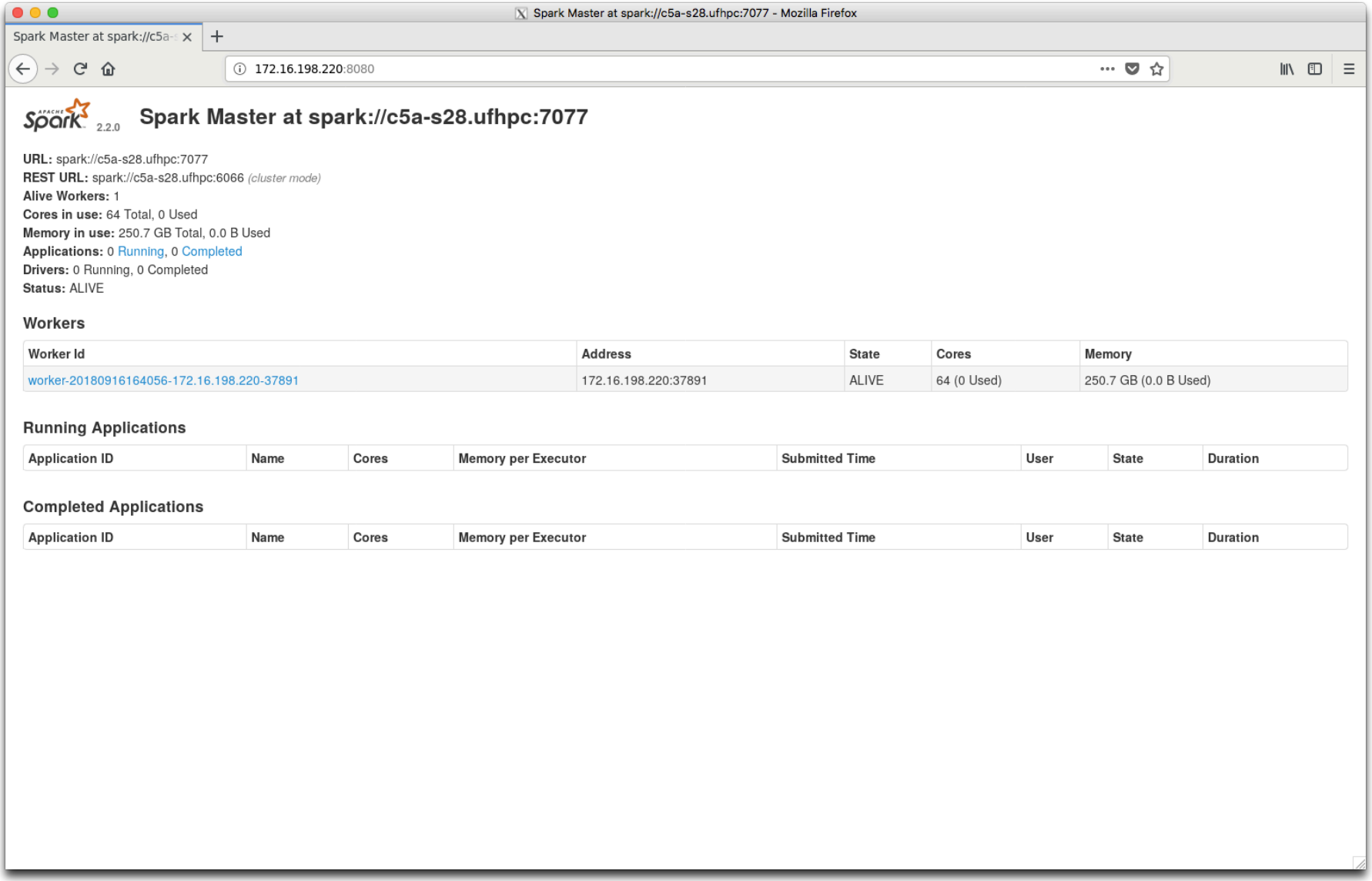


DIY2: SPARK CLUSTER MONITORING

- Get the IP address for the web interface of the master node and launch *firefox* in the X-terminal

```
$> grep MasterWebUI spark_cluster.err
18/09/16 16:40:56 INFO MasterWebUI: Bound MasterWebUI to 0.0.0.0,
and started at http://172.16.198.220:8080
$> module load ubuntu
$> firefox &
[1] 26681
```

DIY2: SPARK CLUSTER MONITORING



BREAK!

SPARK INTERACTIVE SHELLS - SCALA

- Spark interactive shell in Scala
  - $> spark-shell --master $SPARK_MASTER

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

<<omitted>>
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_31)
Type in expressions to have them evaluated.
Type :help for more information.
scala>
```

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 π / 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 # any value
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 operator import add
>>> from random import random
>>> 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)
18/02/27 14:01:30 WARN TaskSetManager: Stage 0 contains a task of very large size (363 KB). The maxim
um recommended task size is 100 KB.
>>> print("Pi is roughly %f" % (4.0 * count / n))
Pi is roughly 3.143667
>>>
```

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/shared/diy4.py .
$>  cat diy4.py

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))
```

DIY4: PI ESTIMATION FROM FILE WITH PYSPARK

```
$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
$> PYTHONSTARTUP=diy4.py pyspark --master $SPARK_MASTER

Picked up _JAVA_OPTIONS:
Python 2.7.6 …

<<omitted>>

Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.2.0
      /_/

Using Python version 2.7.6 (default, Feb  5 2014 11:52:59)
SparkSession available as 'spark'.
18/03/07 16:32:59 WARN TaskSetManager: Stage 0 contains a task of very large
size (363 KB). The maximum recommended task size is 100 KB.
Pi is roughly 3.143180
>>>
```

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

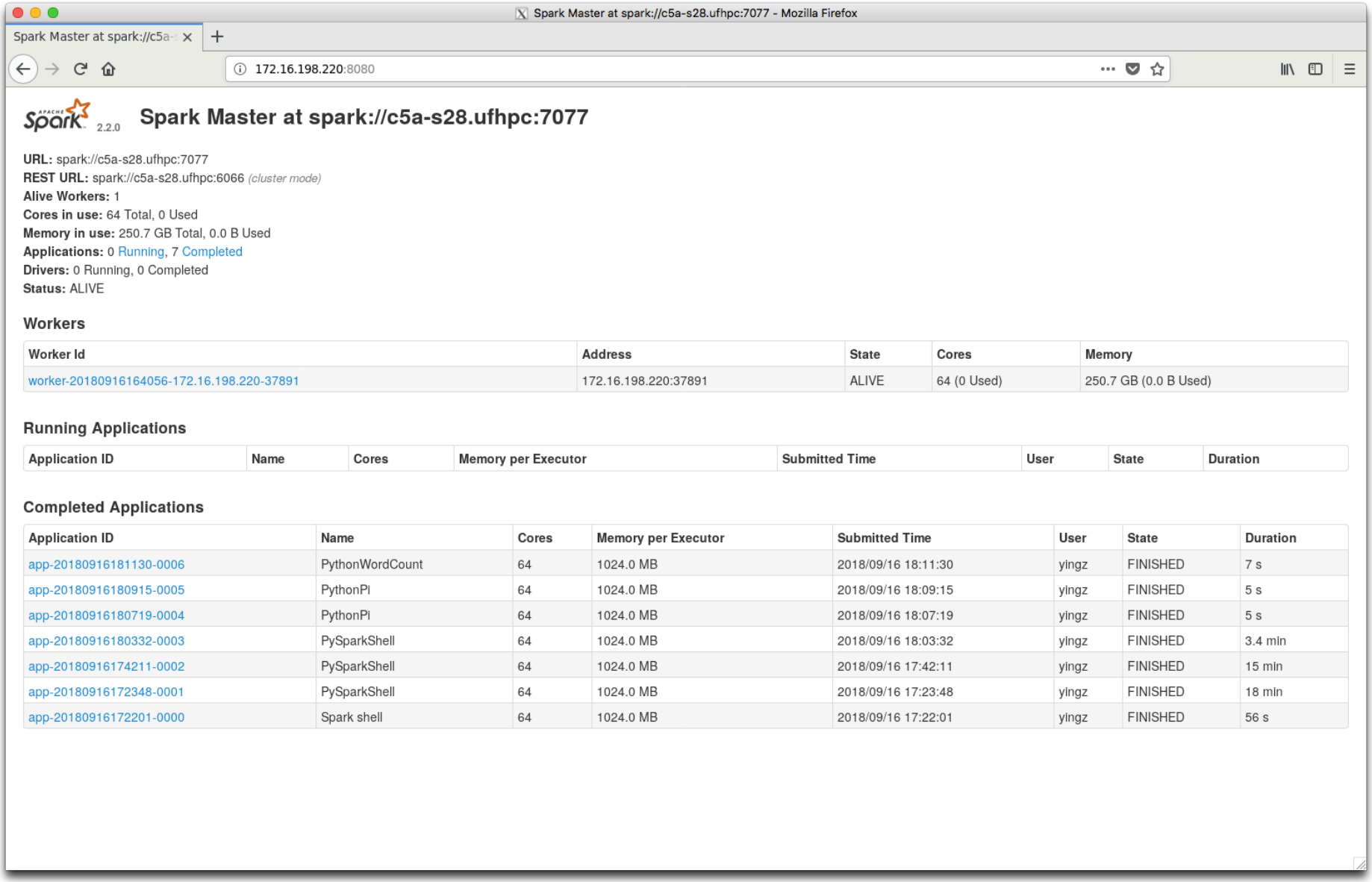
```
$> SPARK_MASTER=$(grep "Starting Spark master" *.err | cut -d " " -f 9)
$> spark-submit --master $SPARK_MASTER $SPARK_HOME/examples/src/main/python/pi.py 10
```

```
5.58 (executor 0) (99/100)
18/03/08 14:00:46 INFO TaskSetManager: Finished task 99.0 in stage 0.0 (TID 99) in 91 ms on 172.16.19
5.58 (executor 0) (100/100)
18/03/08 14:00:46 INFO TaskSchedulerImpl: Removed TaskSet 0.0, whose tasks have all completed, from p
ool
18/03/08 14:00:46 INFO DAGScheduler: ResultStage 0 (reduce at /apps/spark/2.2.0-hadoop2.7/examples/sr
c/main/python/pi.py:43) finished in 1.621 s
18/03/08 14:00:46 INFO DAGScheduler: Job 0 finished: reduce at /apps/spark/2.2.0-hadoop2.7/examples/s
rc/main/python/pi.py:43, took 1.818336 s
Pi is roughly 3.140117
18/03/08 14:00:46 INFO SparkUI: Stopped Spark web UI at http://172.16.206.4:4040
18/03/08 14:00:46 INFO StandaloneSchedulerBackend: Shutting down all executors
18/03/08 14:00:46 INFO CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut dow
n
18/03/08 14:00:46 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
18/03/08 14:00:46 INFO MemoryStore: MemoryStore cleared
18/03/08 14:00:46 INFO BlockManager: BlockManager stopped
18/03/08 14:00:46 INFO BlockManagerMaster: BlockManagerMaster stopped
18/03/08 14:00:46 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinat
or stopped!
18/03/08 14:00:46 INFO SparkContext: Successfully stopped SparkContext
18/03/08 14:00:47 INFO ShutdownHookManager: Shutdown hook called
18/03/08 14:00:47 INFO ShutdownHookManager: Deleting directory /tmp/spark-d1076f8b-abd6-4953-abba-3b1
29d3c64da/pyspark-9329020e-2a8e-411e-88fb-1768fc13ceaf
18/03/08 14:00:47 INFO ShutdownHookManager: Deleting directory /tmp/spark-d1076f8b-abd6-4953-abba-3b1
29d3c64da
giljael@i21b-s2:~/spark$
```

DIY6: WORDCOUNT USING SPARK-SUBMIT

```
$> SPARK_MASTER=$(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
: 36
command:: 6
Master:: 48
app-20180308135820-0000: 5
"-Dspark.driver.port=43217": 1
Unable: 2
kill: 5
ALIVE: 1
ExecutorRunner:: 16
/apps/spark/2.2.0-hadoop2.7: 1
giljael: 16
172.16.206.4:51098: 1
bootstraps): 1
…
```

SPARK JOB HISTORY



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.