

# SPARK ON HIPERGATOR

Ying Zhang  
yingz@ufl.edu

*March 15<sup>th</sup>, 2018*

# AGENDA

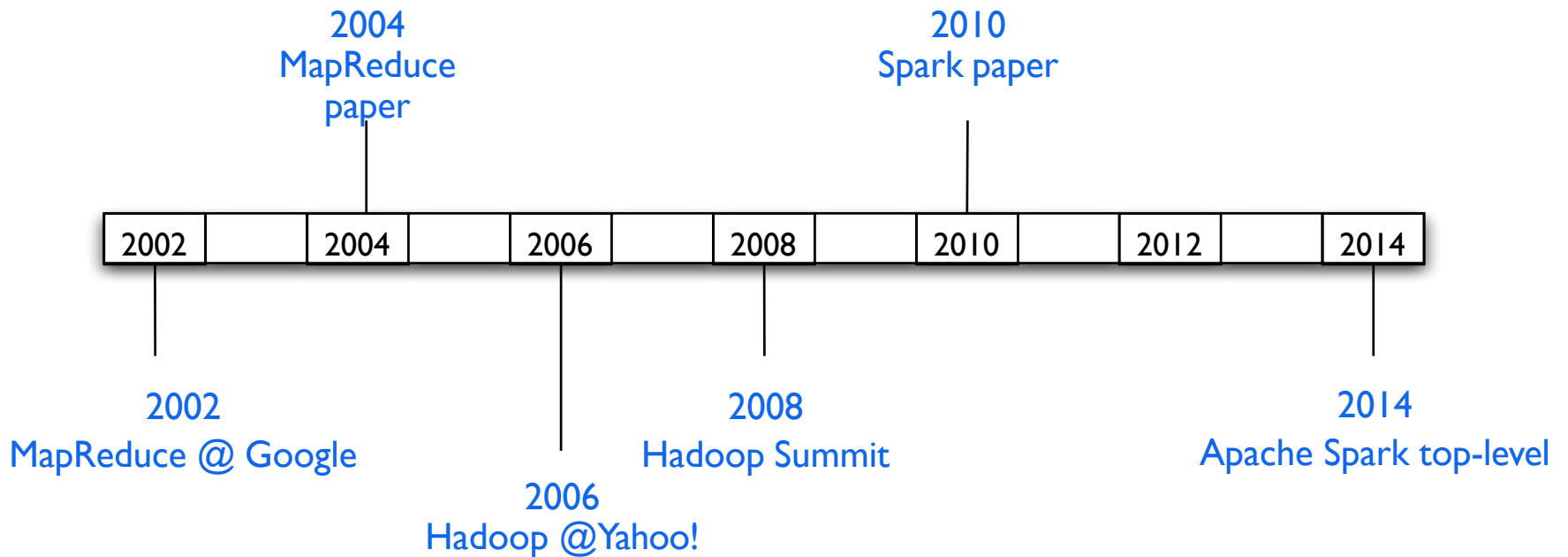
- Introduction
  - Apache Spark
  - Research Computing and HiPerGator
- Spark on HiPerGator
- Hands-on Exercises

# AGENDA

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

# APACHE SPARK

- A brief history



# MAP-REDUCE

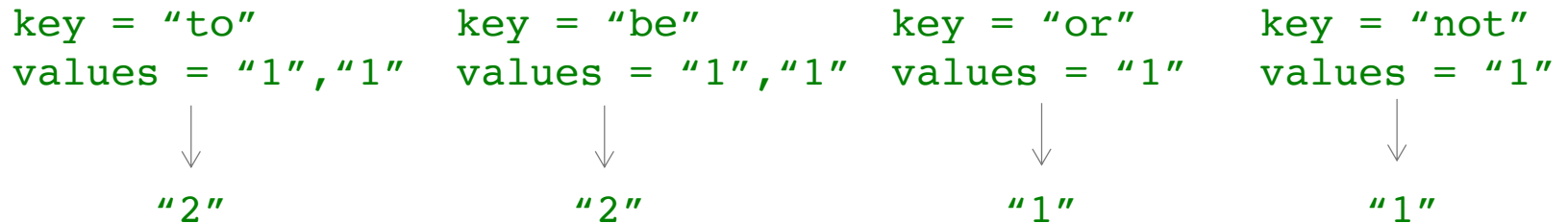
- Data-parallel model
  - One operation, run it on all of the data
- A simple programming model that applies to many large-scale computing problems
- Typical problem
  - Read a lot of data
  - Map: extract desired information from each record
  - Shuffle and sort
  - Reduce: aggregate, summarize, filter,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

# MAP-REDUCE

- Word count example:
  - *Map* function:
    - Input: a key/value pair
      - Key = "file1"
      - Value = "to be or not to be"
    - Output: key/value pairs
      - "to", "1"
      - "be", "1"
      - "or", "1"
      - "not", "1"
      - "to", "1"
      - "be", "1"

# MAP-REDUCE

- Word count example:
  - *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.



# SPARK

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

# 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

# HADOOP VS. SPARK

● apache hadoop  
Search term

● apache spark  
Search term

+ Add comparison

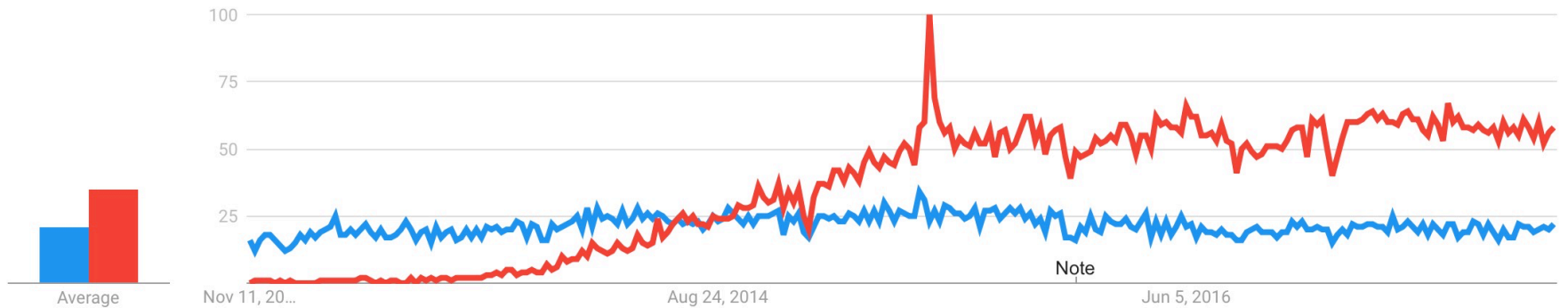
Worldwide ▼

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

# RDD: RESILIENT DISTRIBUTED DATASETS

- Primary abstraction in Spark
- Collection of elements that can be operated on in parallel
  - Transformations
  - Actions
- Fault tolerance: track the series of transformations used to build them (lineage)

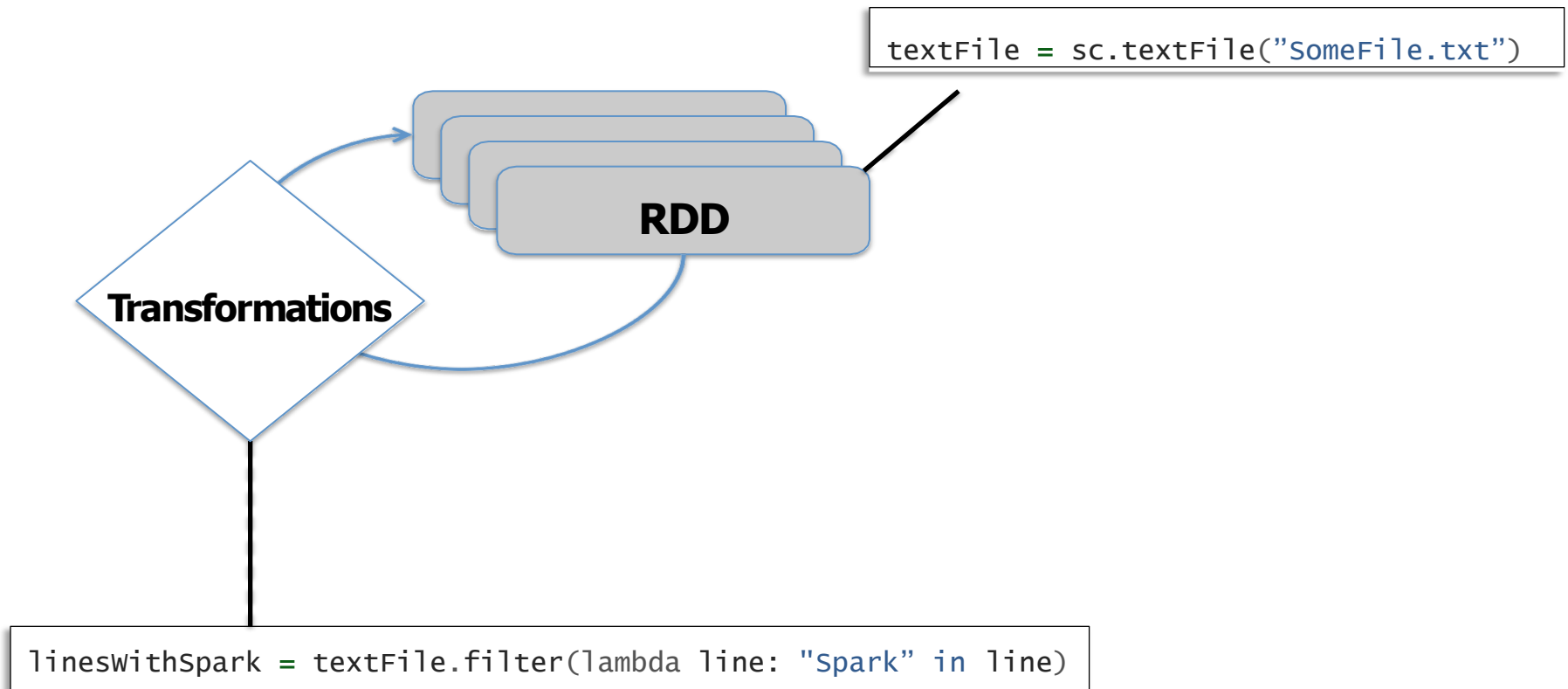
# RDD: HOW DOES IT WORK?

```
textFile = sc.textFile("SomeFile.txt")
```



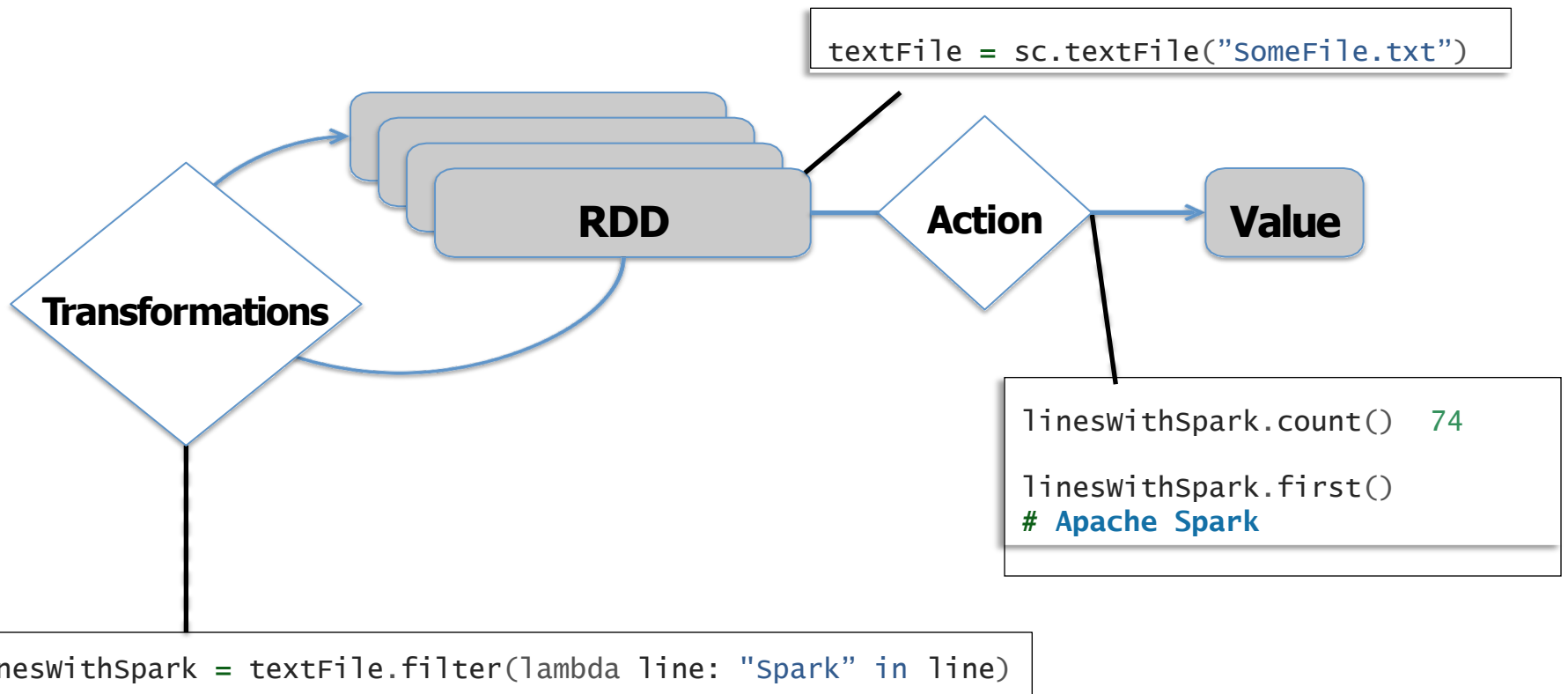
**RDD**

# RDD: HOW DOES IT WORK?

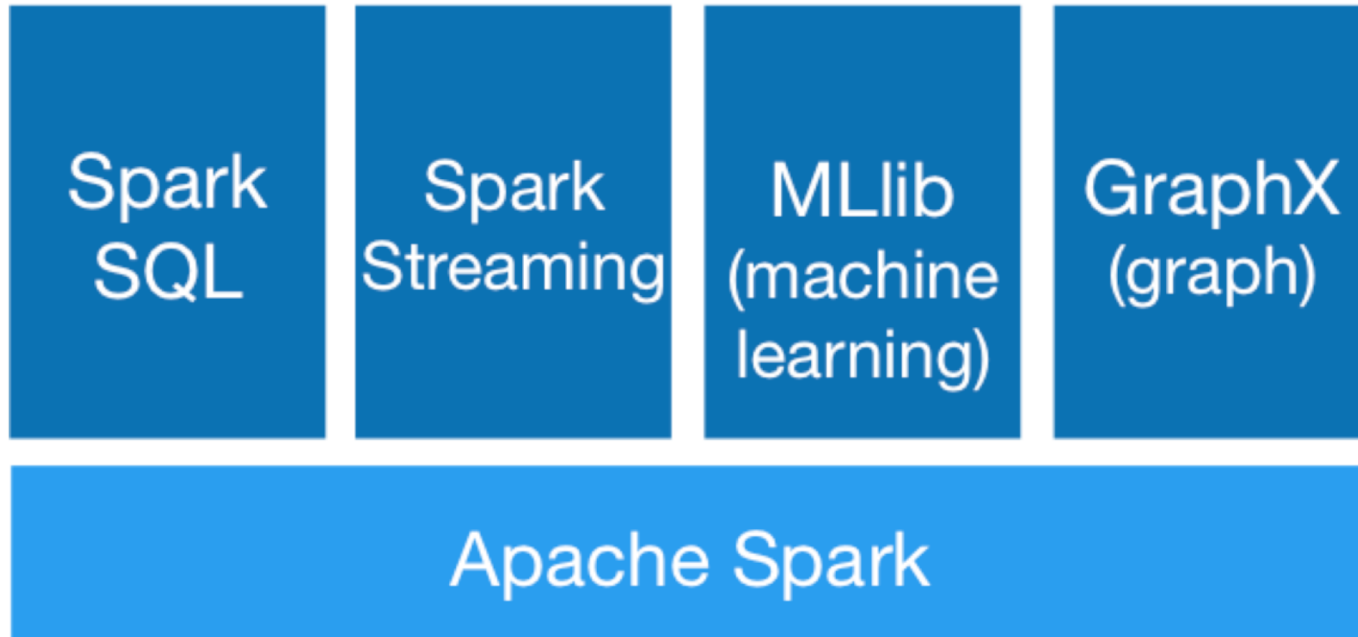




# RDD: HOW DOES IT WORK?



# SPARK ECOSYSTEM

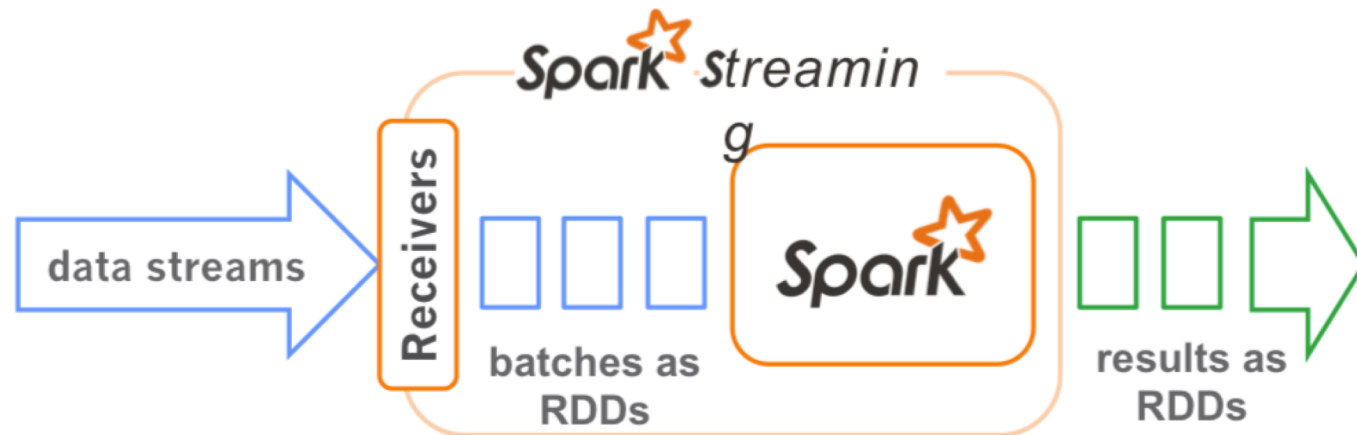


# SPARK SQL AND DATAFRAMES

- 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

# AGENDA

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

# HIPERGATOR



#GATORGOO

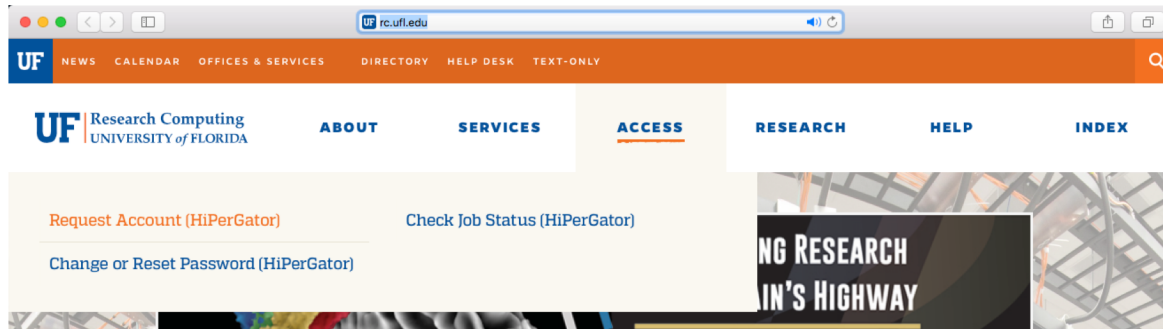


# HIPERGATOR LOGISTICS

- 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 displaying the HiPerGator website. The browser's address bar shows [help.rc.ufl.edu](https://help.rc.ufl.edu). The website has a header with the HiPerGator logo and the text "RESEARCH COMPUTING". Below the header is a navigation menu with links for Docs, Infrastructure, Research, Software, Training, and Navigation. The main content area is titled "UFRC Help and Documentation" and contains a welcome message. Below the welcome message are several sections of help topics, each enclosed in a colored box: "Getting Started" (orange box), "Software and Libraries" (blue box), "Batch System" (orange box), "Connecting and Data Transfer" (blue box), and "Specific Research Areas" (orange box). At the bottom of the page, there is a "Category: Docs" dropdown menu.

Page [Discussion](#) [Read](#) [View source](#) [View history](#)  [Go](#) [Log in](#)

## 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](#)

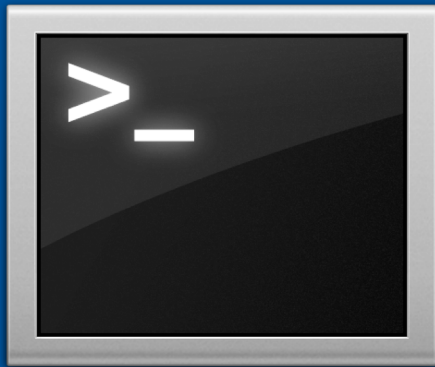
### Specific Research Areas

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

Category: [Docs](#)

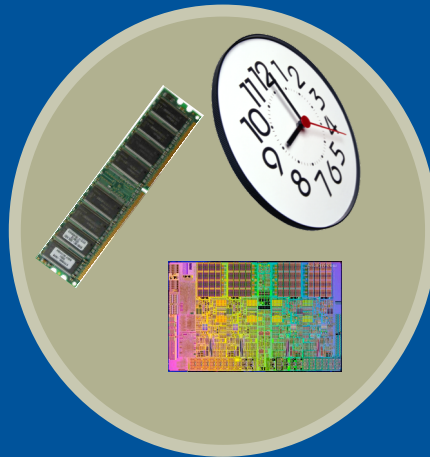
# CLUSTER BASICS

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)
```

# AGENDA

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