https://images.techhive.com/images/article/2016/08/thinkstockphotos-462205183-100677632-large.jpg

Lecture 1: Introduction to Spark and HPC

Haiping Lu

COM6012: Scalable ML

YouTube Playlist: https://www.youtube.com/c/HaipingLu/

Week 1 Contents / Objectives

- The Big Data Problem: Why Spark?
- What is Spark?: The Essentials
- An Example of Spark: Log Mining
- How to Use Spark: PySpark, HPC, Resources

Week 1 Contents / Objectives

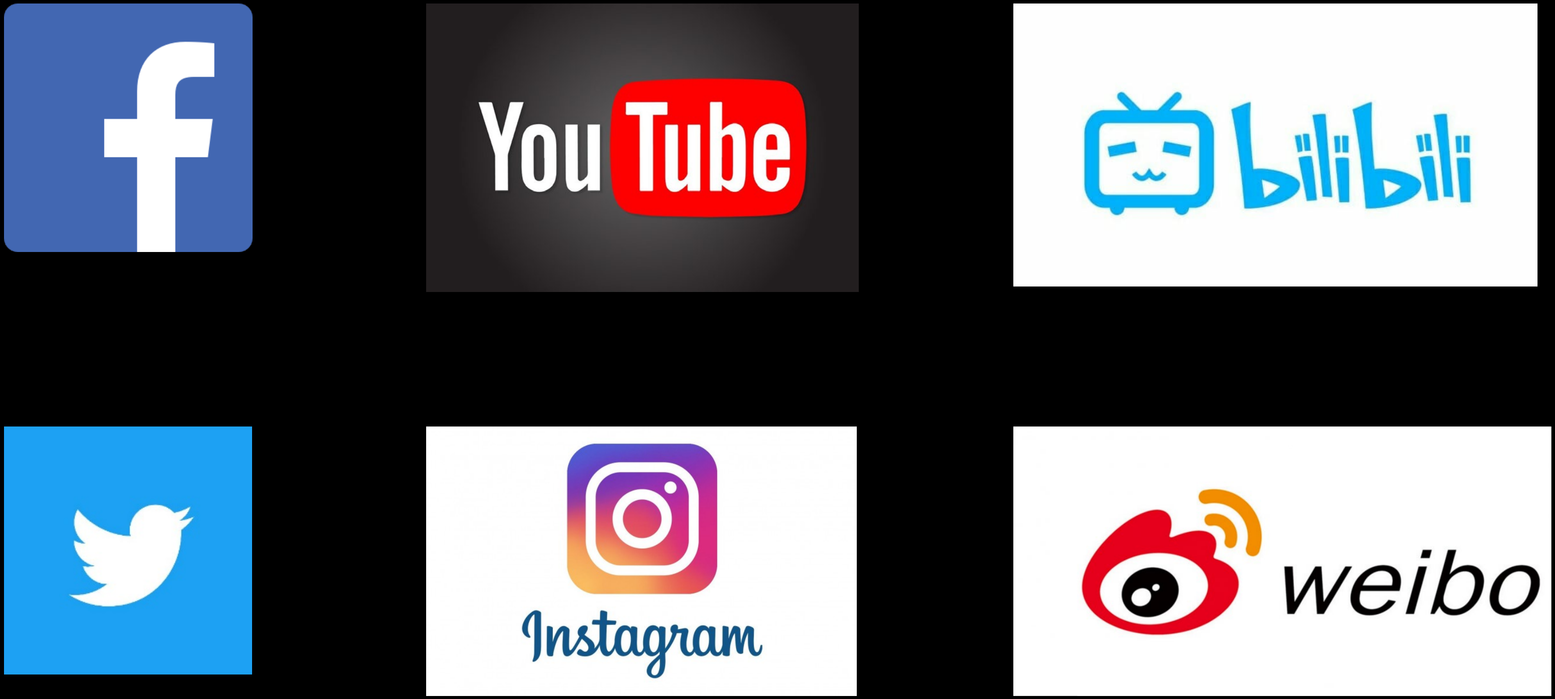
- The Big Data Problem: Why Spark?
- What is Spark?: The Essentials
- An Example of Spark: Log Mining
- How to Use Spark: PySpark, HPC, Resources

Where Does Big Data Come From?

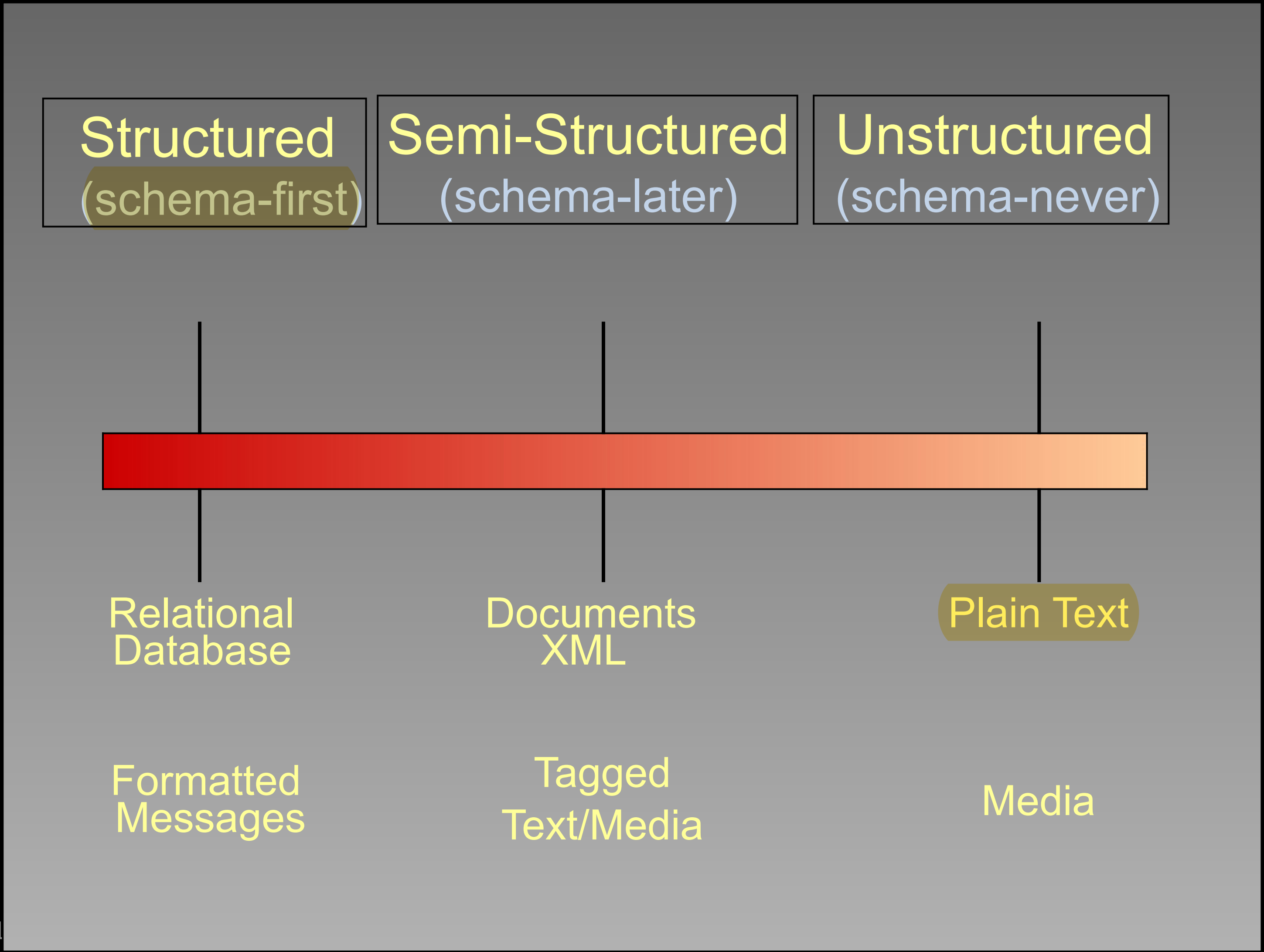
- All happening online, e.g. tracking of:
  - Clicks
  - Billing events
  - Server requests
  - Transactions
  - Network messages
  - Faults
  - …

Where Does Big Data Come From?

- User generated content: web + mobile



Data Structure Spectrum



Structured (schema-first)

Semi-Structured (schema-later)

Unstructured (schema-never)

Relational Database

Documents XML

Plain Text

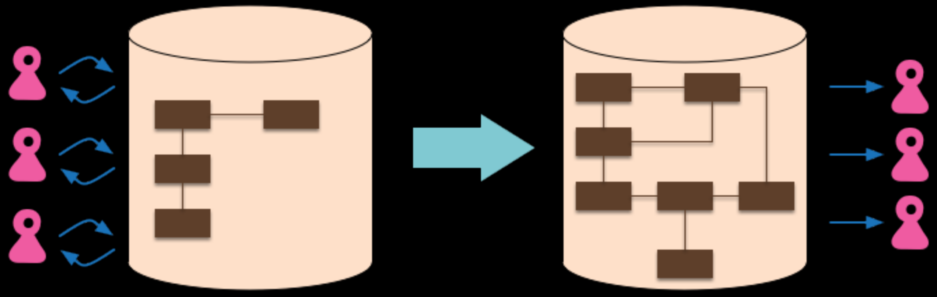
Formatted Messages

Tagged Text/Media

Media

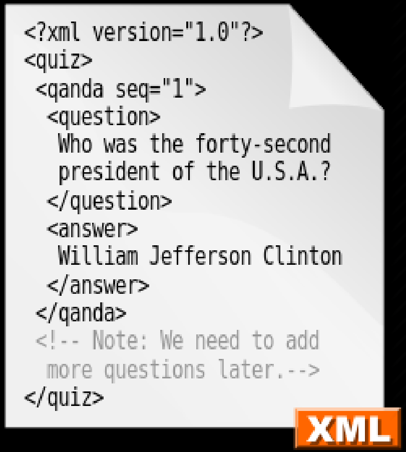
Structured Data

- Database: relational data model → how a database is structured and used (hottest job 20 years ago)
- Schema: the organisation of data as a blueprint of how the database is constructed
  - The programmer must statically specify the schema
  - Decreasing ← consumer/media app, enterprise search
- SQL: Structured Query Language



Semi-Structured Data

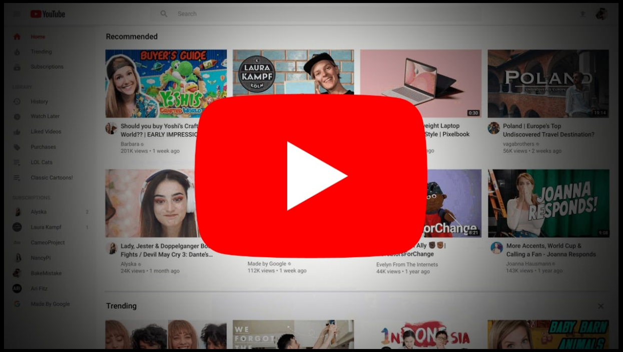
- Self-describing rather than formal structures, tags/markers to separate semantic elements
- The column types → the schema for the data
  - Spark dynamically infers the schema while reading each row
  - Programmer statically specifies the schema
- Examples:

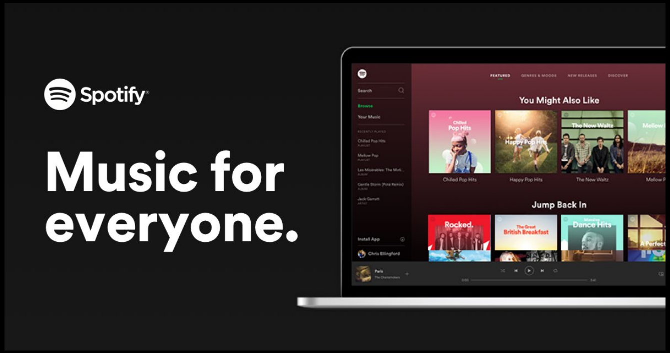


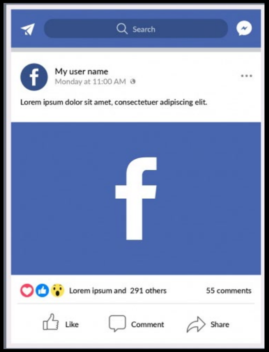
```
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

Unstructured Data

- Only one column with string or binary type
- Examples



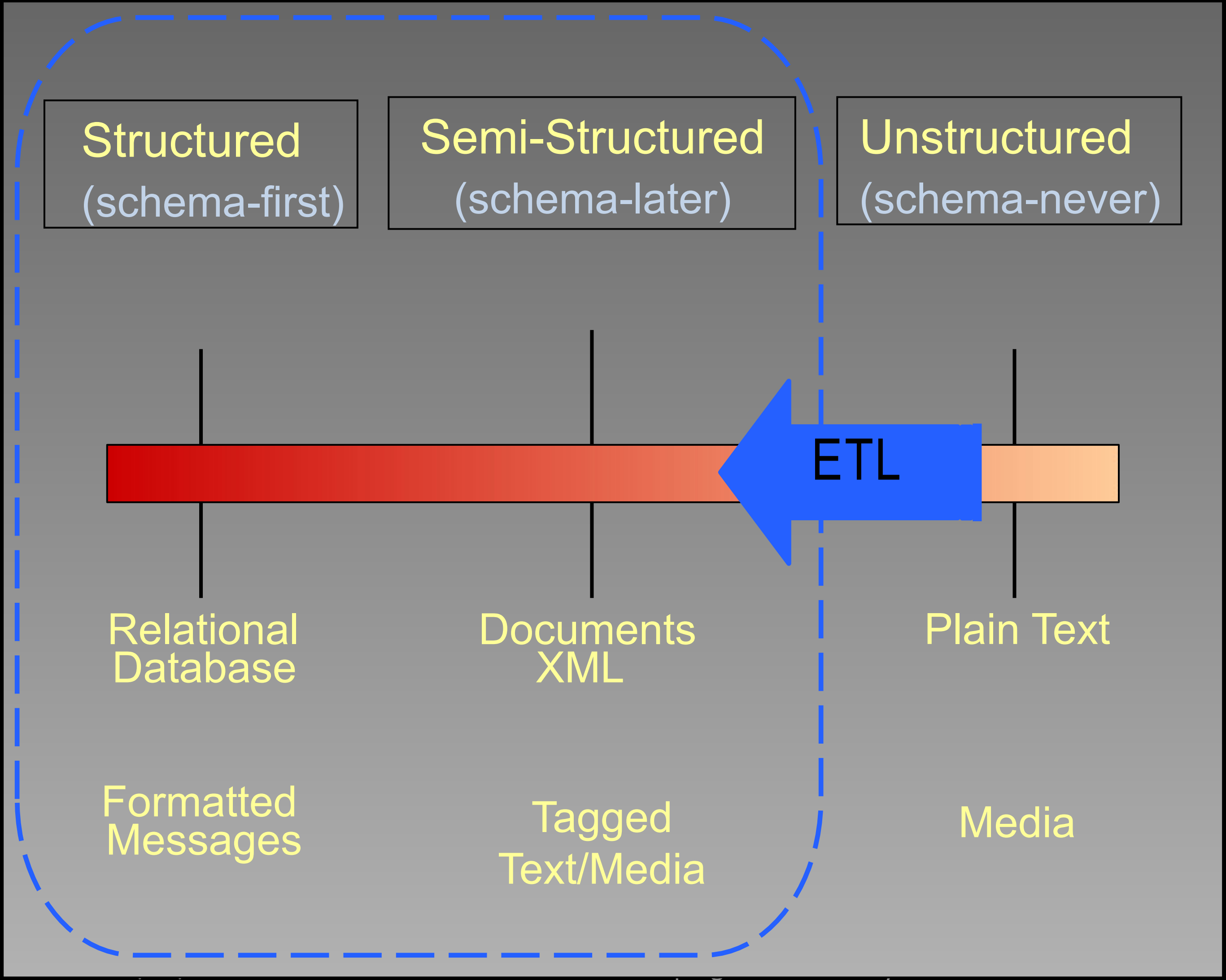




Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

- More than 70%–80% of all data in organisations (Shilakes 1998)

Traverse the Data Structure Spectrum



Structured (schema-first)

Semi-Structured (schema-later)

Unstructured (schema-never)

ETL

Relational Database

Documents XML

Plain Text

Formatted Messages

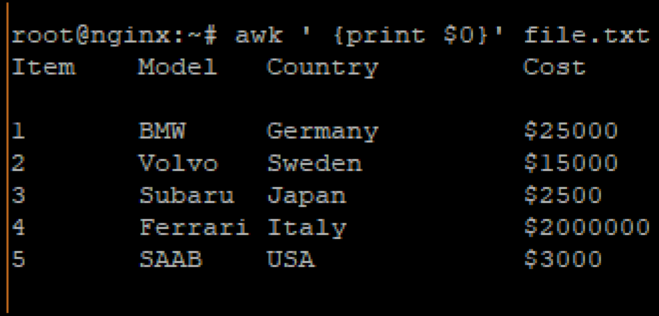
Tagged Text/Media

Media

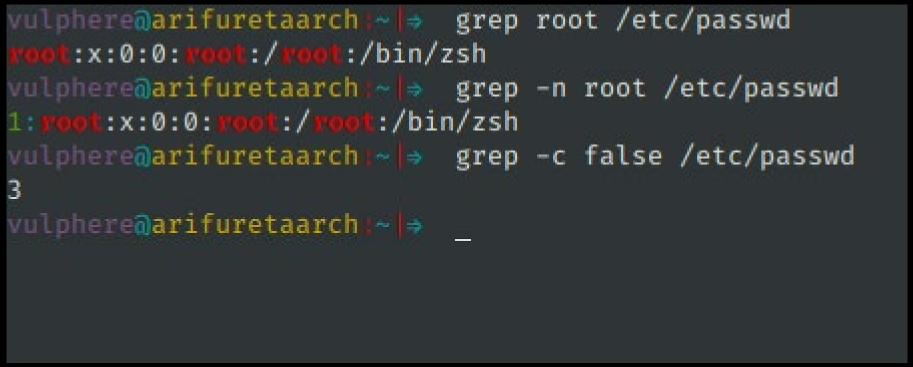
- Impose structure on unstructured data
  - Extract
  - Transform
  - Load

Traditional Analysis Tools

- Unix shell commands (awk, grep, …)



```
root@nginx:~# awk ' {print $0}' file.txt
Item    Model   Country         Cost

1       BMW     Germany         $25000
2       Volvo   Sweden          $15000
3       Subaru  Japan           $2500
4       Ferrari Italy           $2000000
5       SAAB    USA             $3000
```

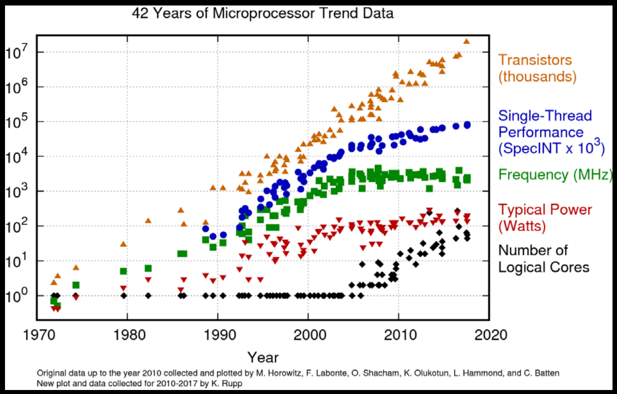


```
vulphere@arifuretaarch:~|⇒  grep root /etc/passwd
root:x:0:0:root:/root:/bin/zsh
vulphere@arifuretaarch:~|⇒  grep -n root /etc/passwd
1:root:x:0:0:root:/root:/bin/zsh
vulphere@arifuretaarch:~|⇒  grep -c false /etc/passwd
3
vulphere@arifuretaarch:~|⇒  _
```

All run on a single machine!

The Big Data Problem

- Data growing faster than computation speeds
- Growing data sources
  - Web, mobile, scientific, …
- Storage getting cheaper
  - Size doubling every 18 months
- But, stalling CPU speeds and storage bottlenecks



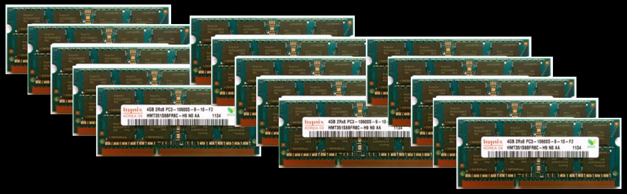
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2017 by K. Rupp

Solution for the Big Data Problem

- One machine can not process or *even store* all the data!
- Solution: distribute data over a cluster of machines

Lots of hard drives

… and CPUs



… and memory!

Week 1 Contents / Objectives

- The Big Data Problem: Why Spark?
- What is Spark?: The Essentials
- An Example of Spark: Log Mining
- How to Use Spark: PySpark, HPC, Resources

Apache Spark

- Fast and general cluster computing system
- Interoperable with hadoop Map Reduce
- Improves efficiency through:
  - In-memory computing primitives
  - General computation graphs

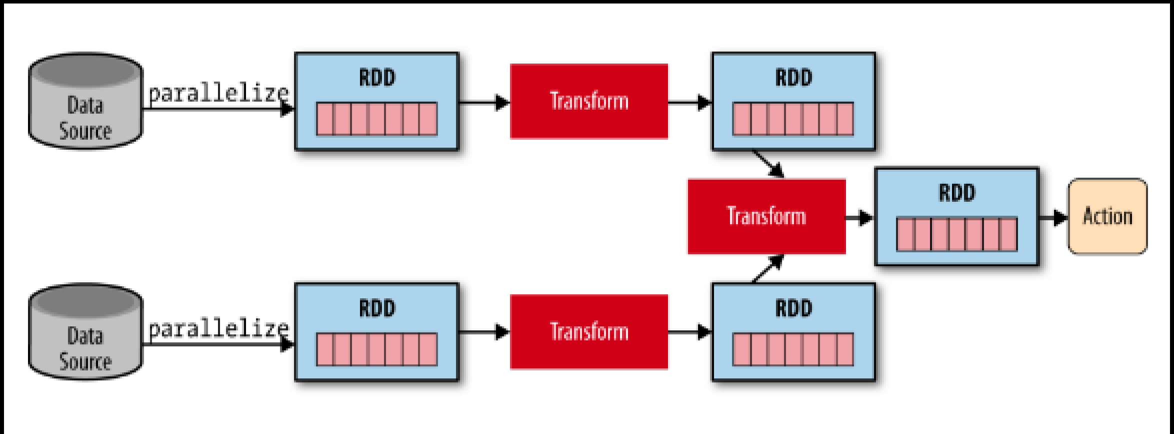
→ Up to 100× faster (2-10× on disk)

- Improves usability through:
  - Rich APIs in Scala, Java, Python
  - Interactive shell

→ 2-5× less code

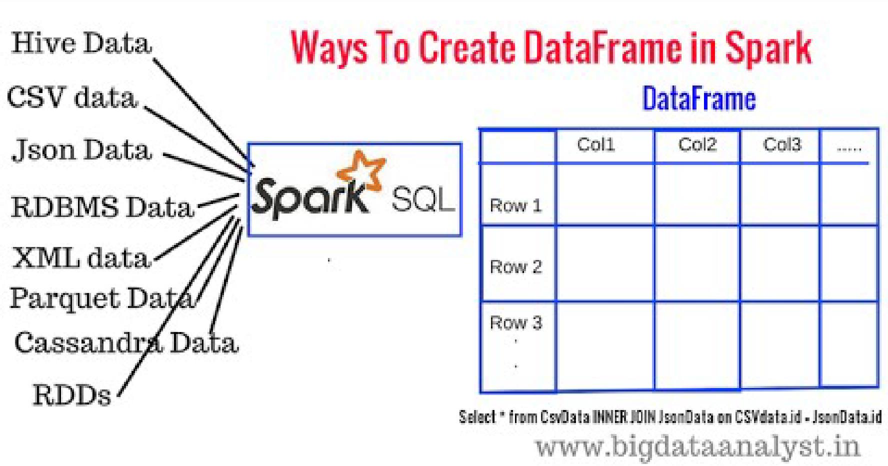
Spark Model

- Write programs in terms of transformations on distributed datasets
- Resilient Distributed Datasets (RDDs)
    - Collections of objects that can be stored in memory or disk across a cluster
    - Parallel functional transformations (map, filter, …)
    - Automatically rebuilt on failure



Spark for Data Science

- DataFrames
  - Structured data (SQL)
  - Familiar API based on R/Python Pandas
  - Distributed, optimised implementation
- Machine learning pipelines
  - Simple construction and tuning of ML workflows



	Col1	Col2	Col3	......
Row 1				
Row 2				
Row 3 . .				

```
Select * from CsvData INNER JOIN JsonData on CSVdata.id = JsonData.id
```

www.bigdataanalyst.in

Spark Computing Framework

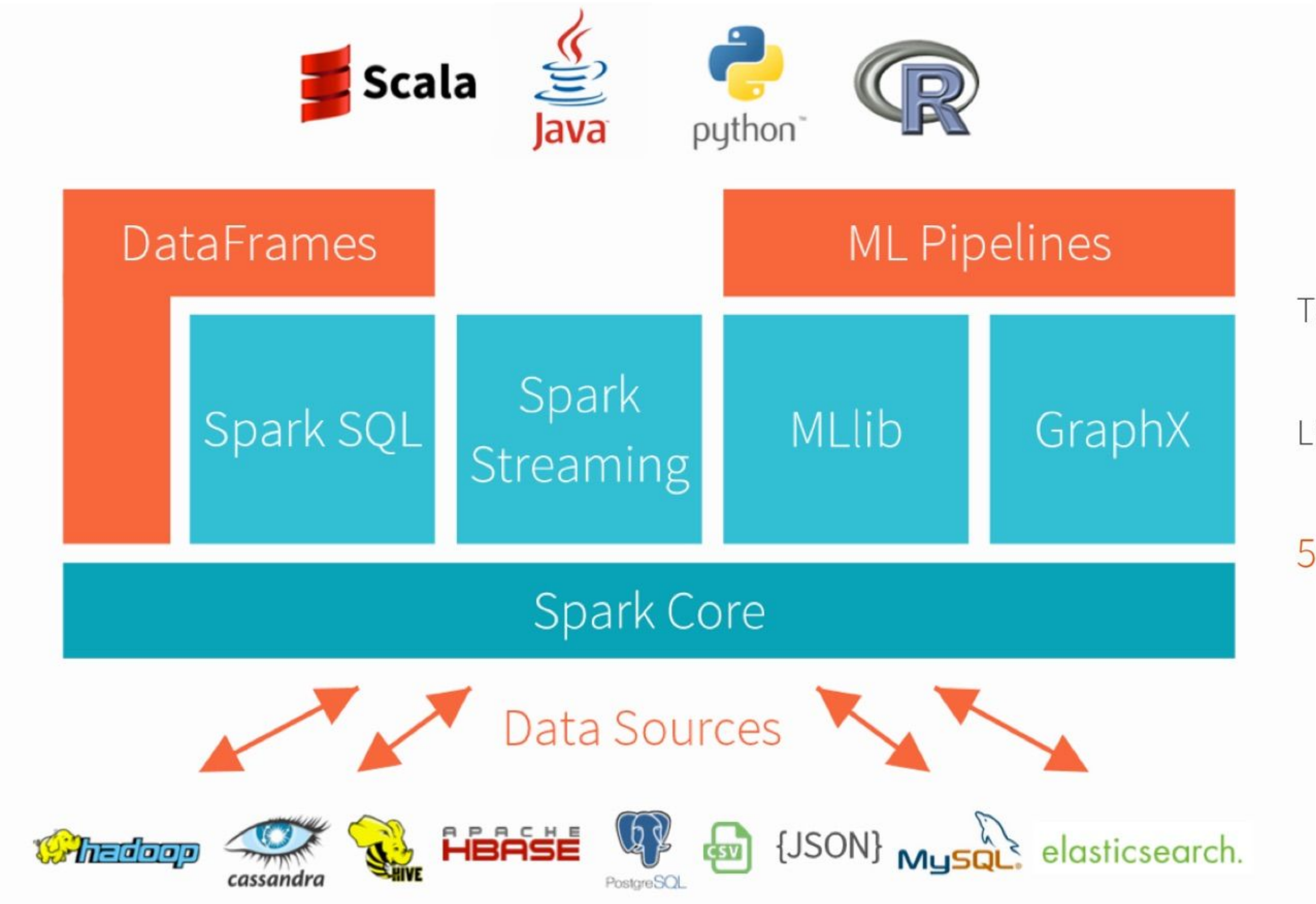
- Programming abstraction and parallel runtime to hide complexities of fault-tolerance and slow machines

"Here's an operation, run it on all of the data"

JUST DO IT.

- I don't care where it runs (you schedule that)
- In fact, feel free to run it twice on different nodes (e.g. when it fails)

Apache Spark Ecosystem



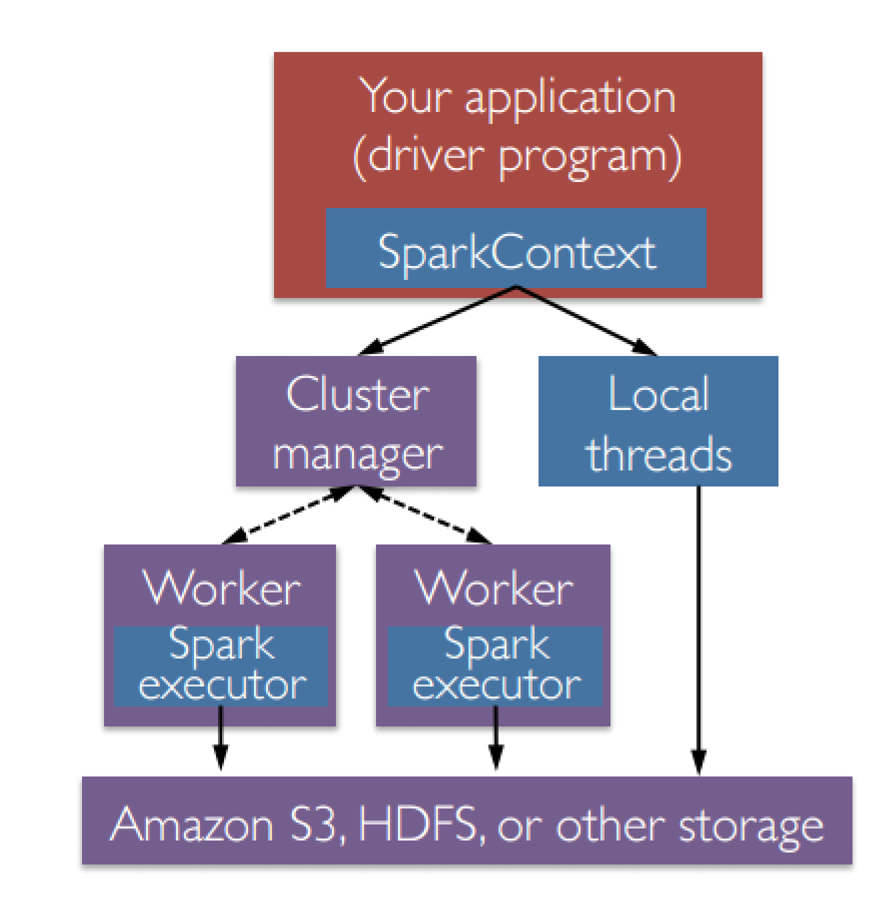
Total contributors: 150 → 500

Lines of code: 190K → 370K

500+ active production deployments

https://i.pinimg.com/originals/e7/f3/2d/e7f32d041846a5938a09e192bdf3885d.jpg

Spark Components



- A Spark program first creates a SparkSession object as the driver (including SparkContext)
  - Tells Spark how/where to access a cluster
  - Connect to cluster managers
- Cluster managers
  - Allocate resources across applications
- Spark executor (worker):
  - Run computations
  - Access data storage

SparkSession and SparkContext

- SparkSession
  - Entry point for DataFrame API, create DataFrames
  - PySpark shell automatically create SparkSession as spark
  - Programs: must create a new SparkSession first (see lab)
- SparkContext
  - Entry point for Spark functionality, create RDDs
  - Connect to a Spark cluster
  - Associated with a SparkSession
  - PySpark shell automatically create SparkContext as sc
  - Programs: sc = spark.sparkContext

The ‘*Master*’ Parameter for a SparkSession

- Determines cluster type and size

Master Parameter	Description
`local`	run Spark locally with one worker thread (no parallelism)
`local[K]` ✓	run Spark locally with K worker threads (ideally set to number of cores)
`spark://HOST:PORT`	connect to a Spark standalone cluster; PORT depends on config (7077 by default)
`mesos://HOST:PORT`	connect to a Mesos cluster; PORT depends on config (5050 by default)

Week 1 Contents / Objectives

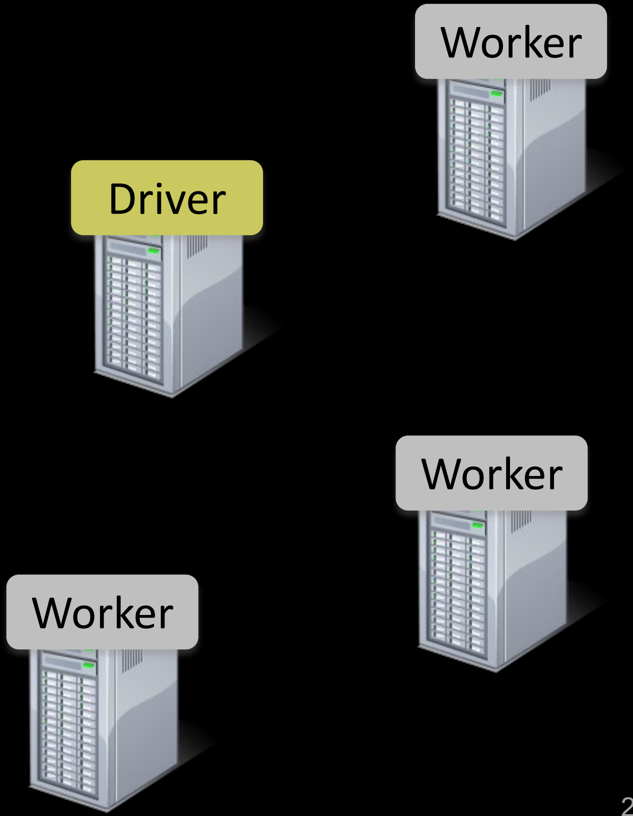
- The Big Data Problem: Why Spark?
- What is Spark?: The Essentials
- An Example of Spark: Log Mining
- How to Use Spark: PySpark, HPC, Resources

Spark Example: Log Mining (w/t RDD)

Load error messages from a log into memory, then interactively search for various patterns

Spark Example: Log Mining

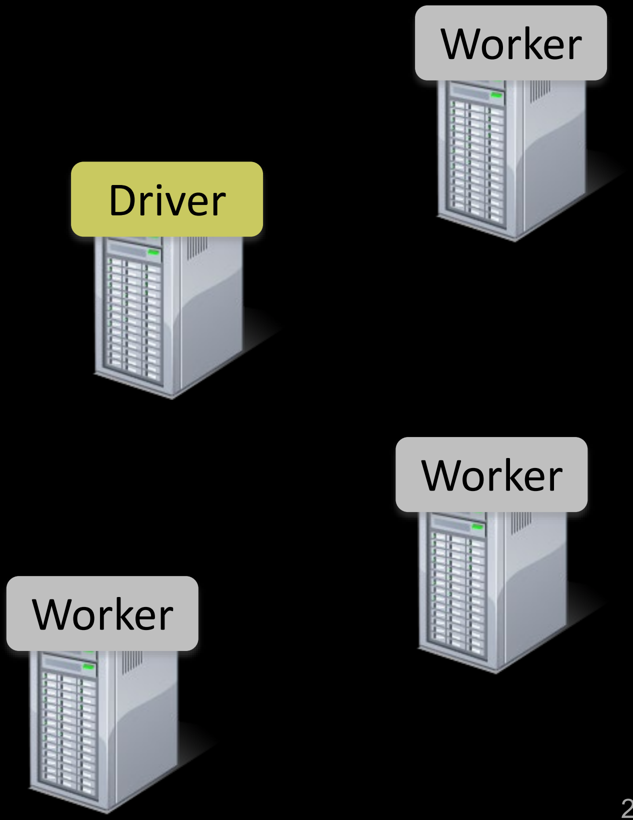
Load error messages from a log into memory, then interactively search for various patterns



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
```

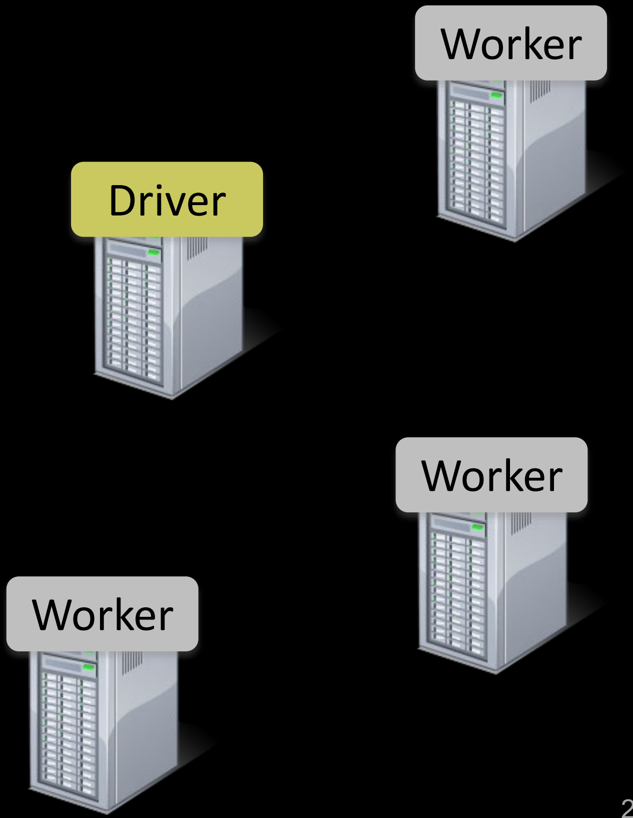


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Base RDD

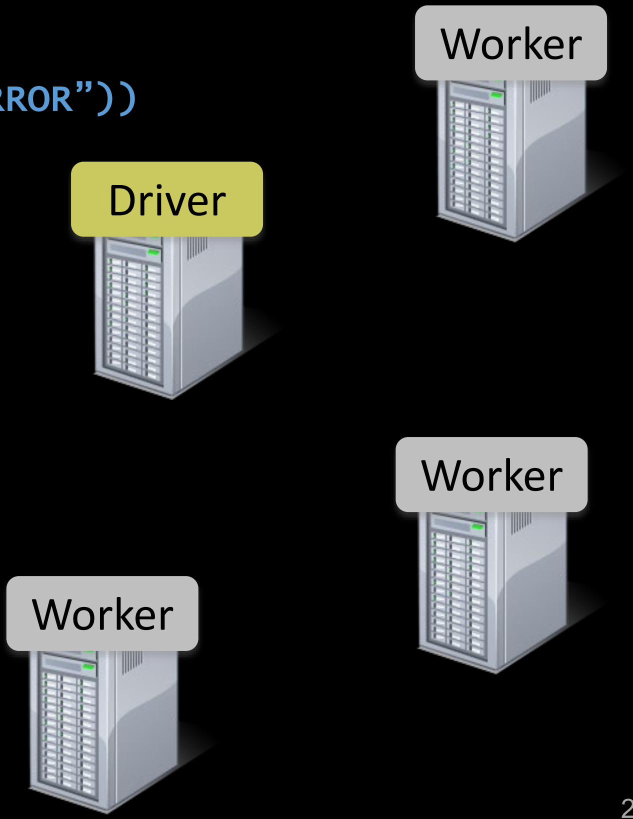
```
lines = spark.textFile(“hdfs://...”)
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
```

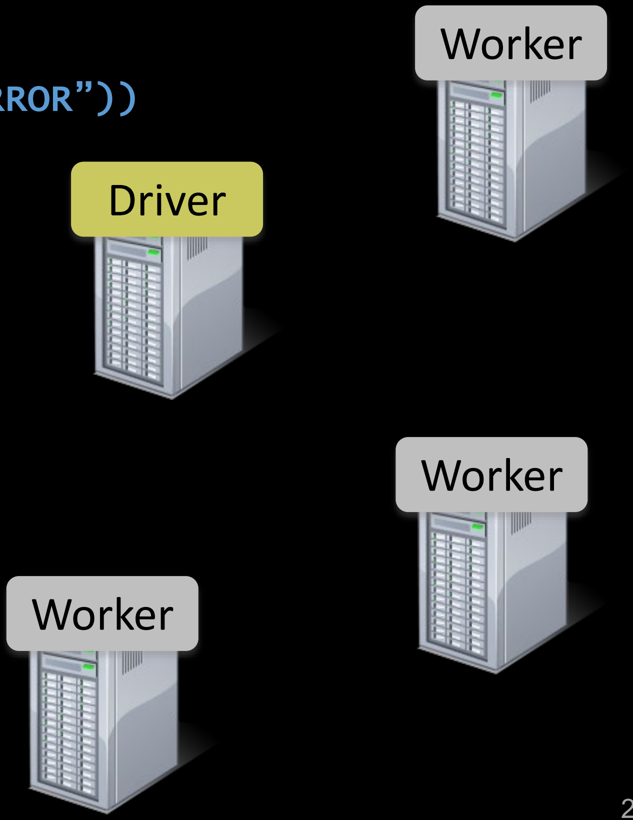


Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

Transformed RDD

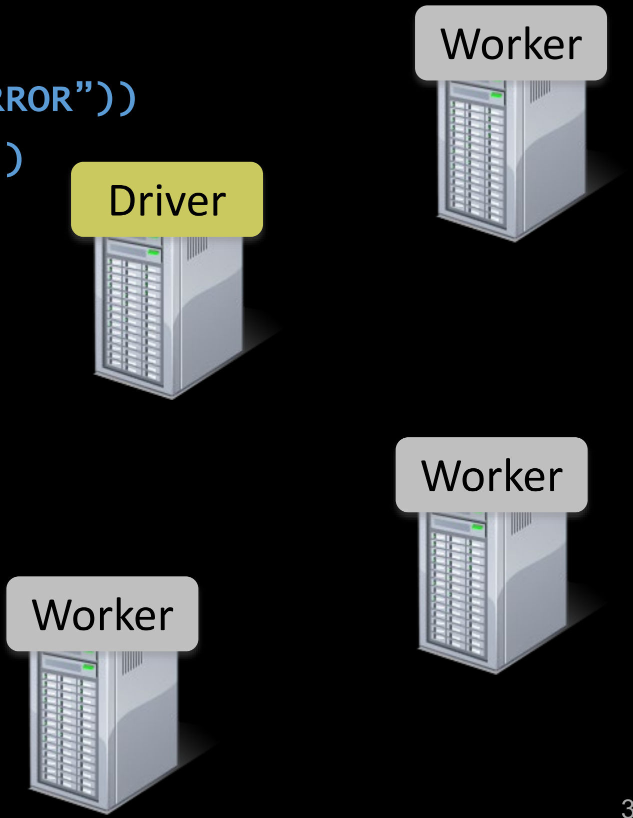
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

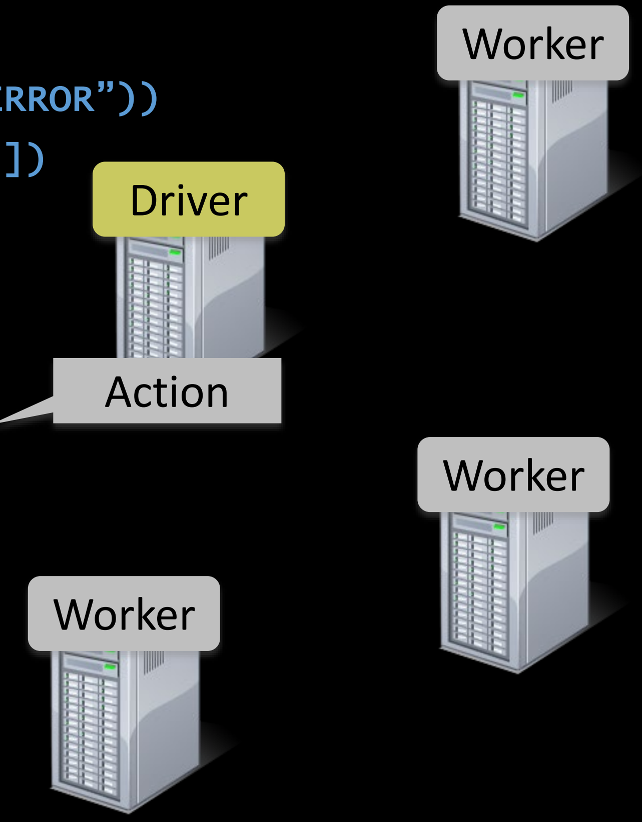
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

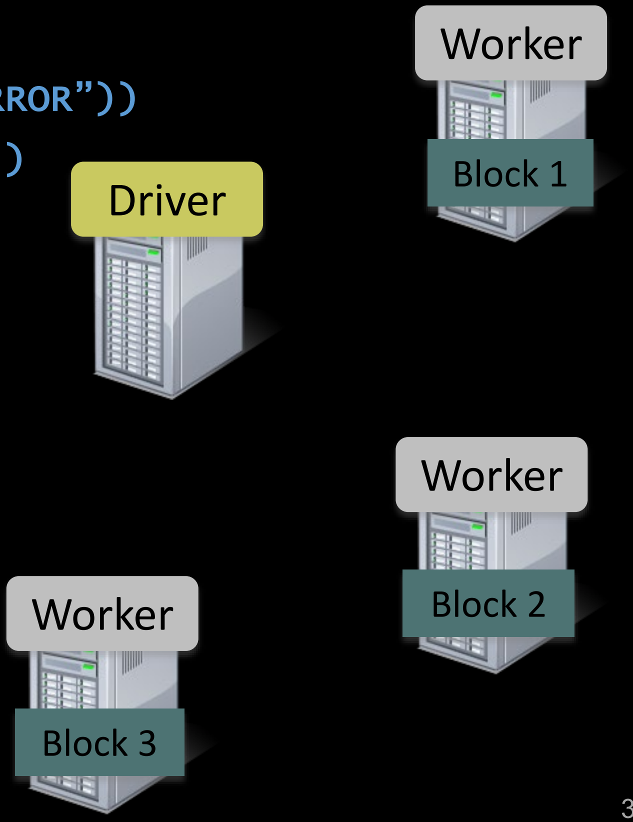
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

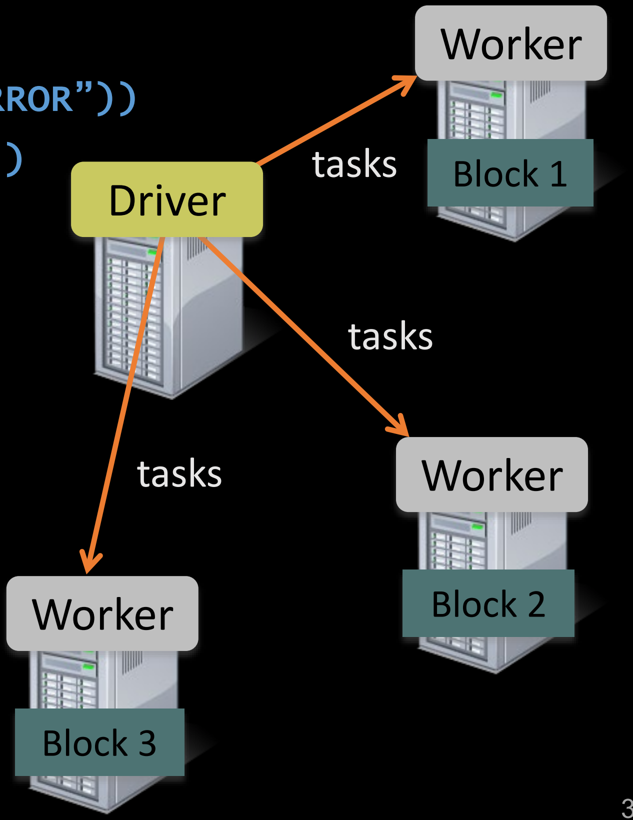
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

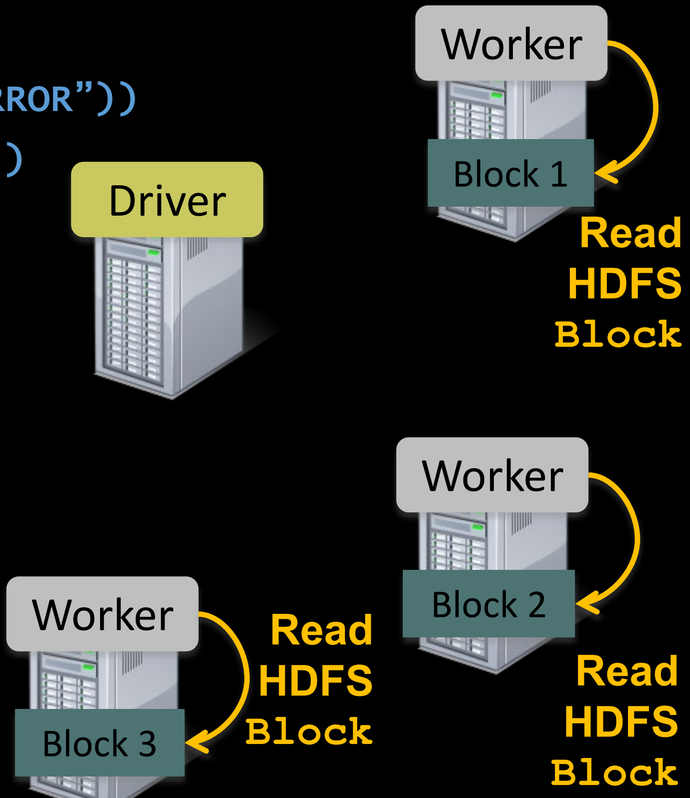
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

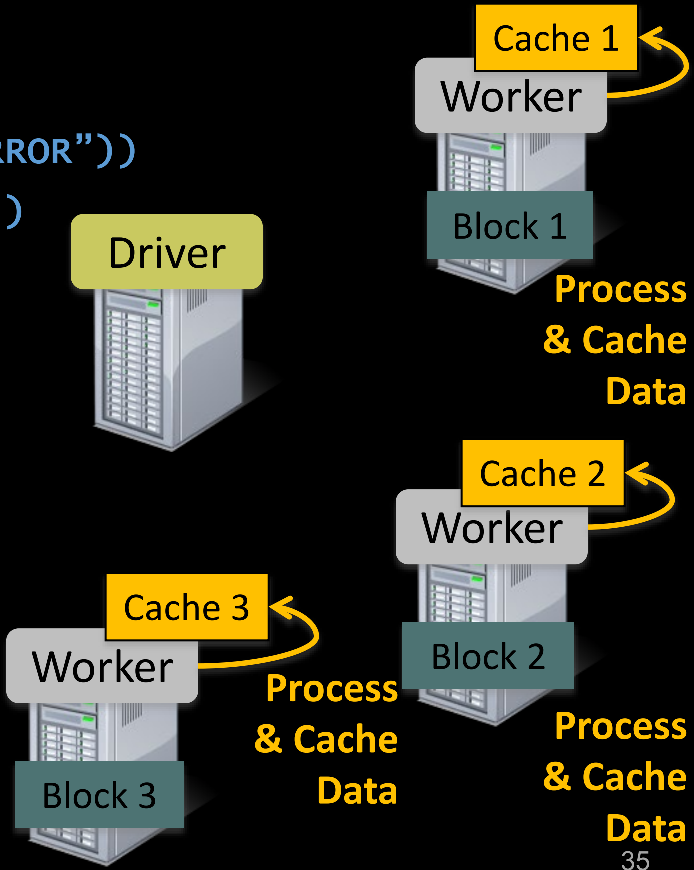
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

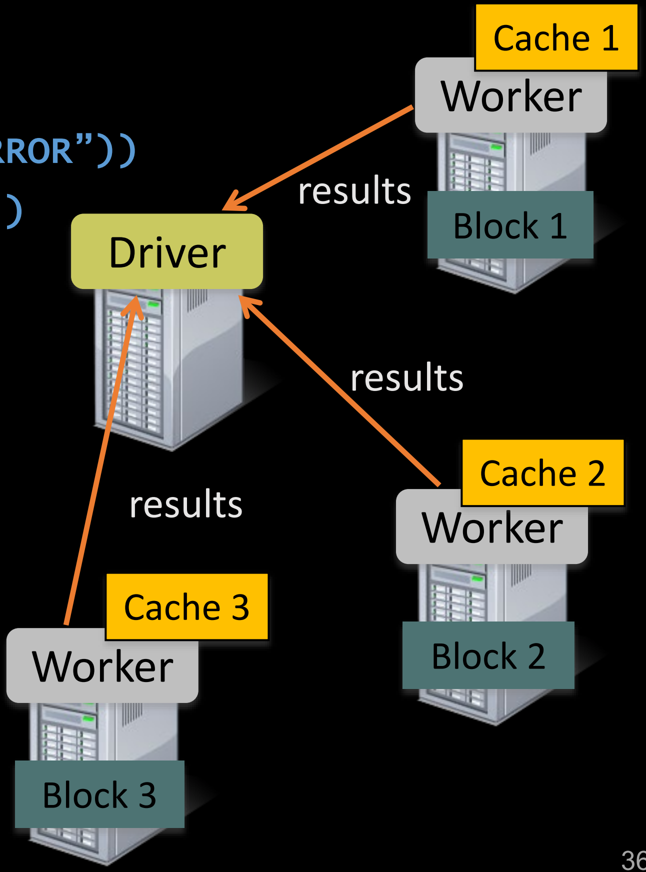
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

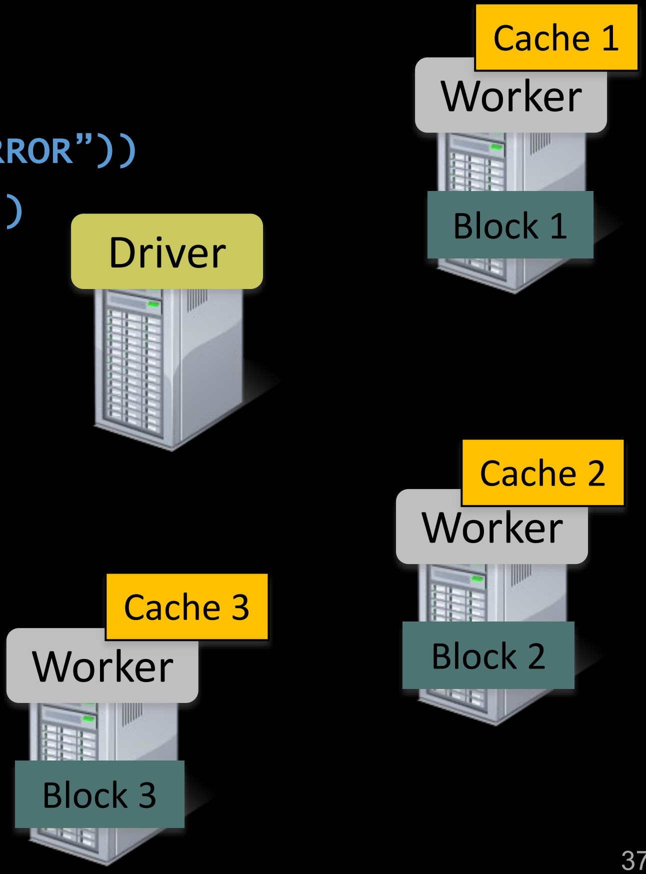
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

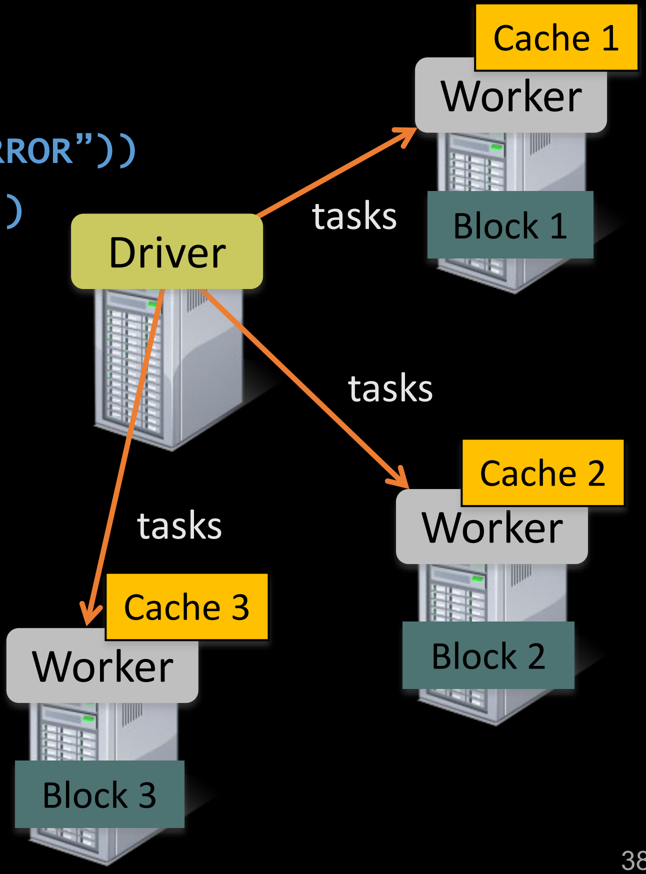
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
messages.filter(lambda s: “php” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

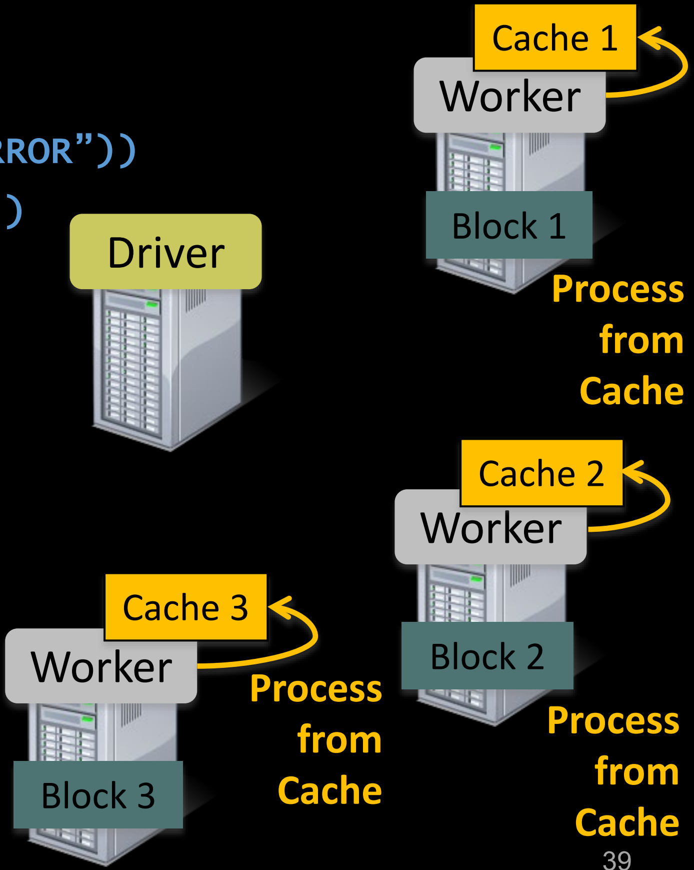
```
lines = spark.textFile(“hdfs://...”)
errors = lines.filter(lambda s: s.startswith(“ERROR”))
messages = errors.map(lambda s: s.split(“\t”)[2])
messages.cache()


messages.filter(lambda s: “mysql” in s).count()
messages.filter(lambda s: “php” in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

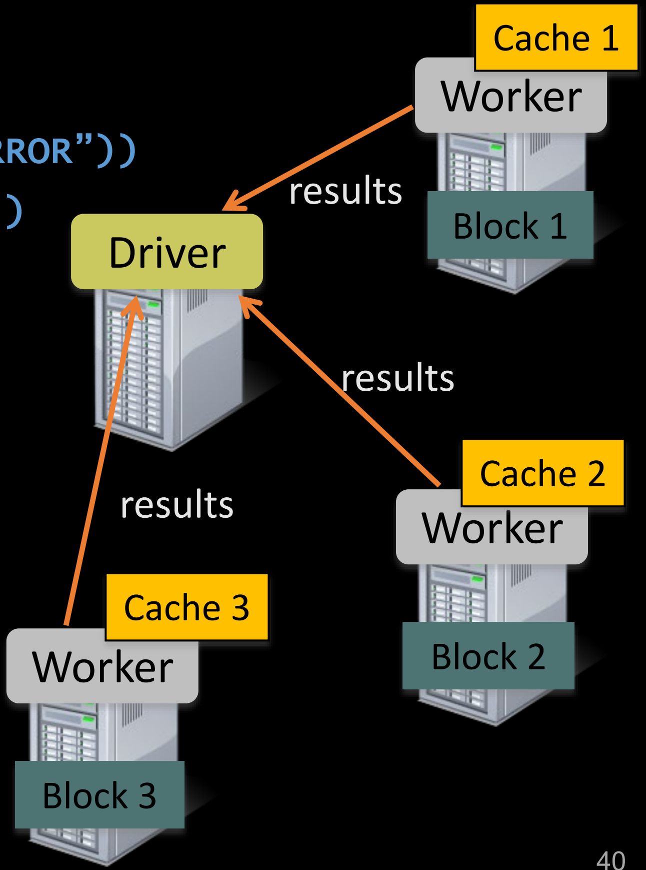
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

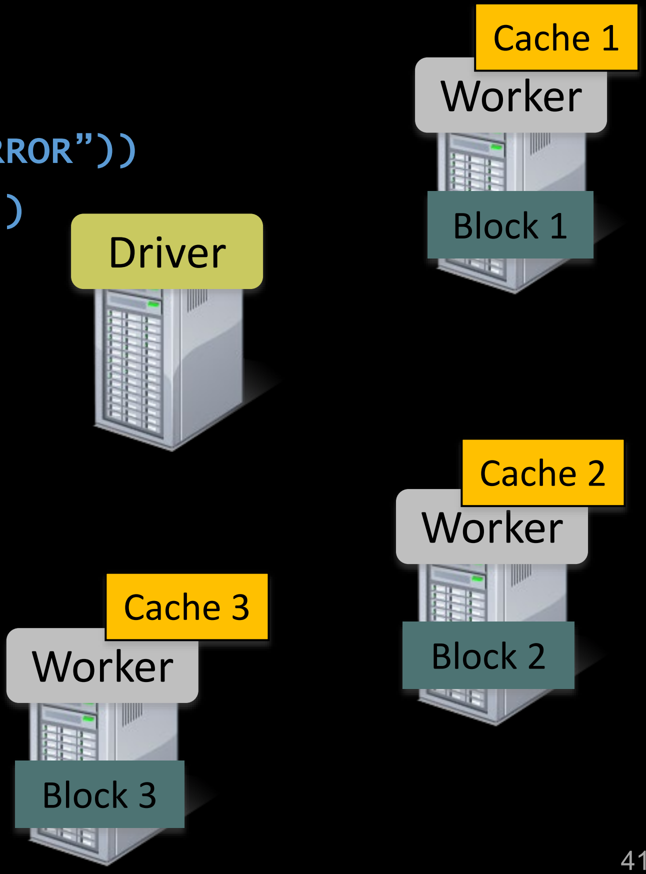
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



Spark Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



Cache your data → Faster results

Full-text search of Wikipedia

- 60GB on 20 EC2 machines
- 0.5 sec from mem vs. 20s for on-disk

Week 1 Contents / Objectives

- The Big Data Problem: Why Spark?
- What is Spark?: The Essentials
- An Example of Spark: Log Mining
- How to Use Spark: PySpark, HPC, Resources

Spark Program Lifecycle

- Create DataFrames from external data or createDataFrame from a collection in a driver program
- Lazily transform them into new DataFrames
- cache( ) some DataFrames for reuse
- Perform actions to execute parallel computation and produce results

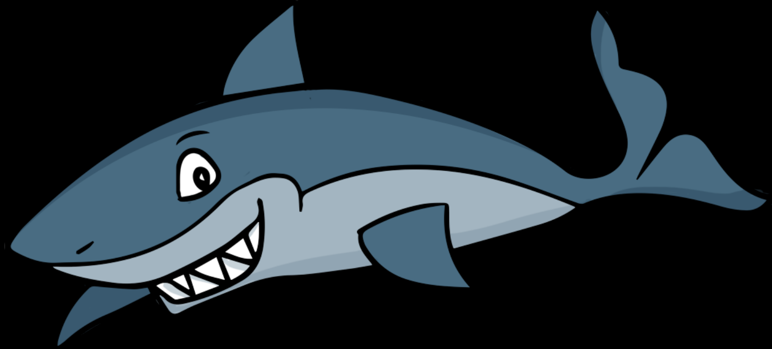
Use Spark Transformations and Actions wherever possible: Search DataFrame reference API

PySpark 3.0.1

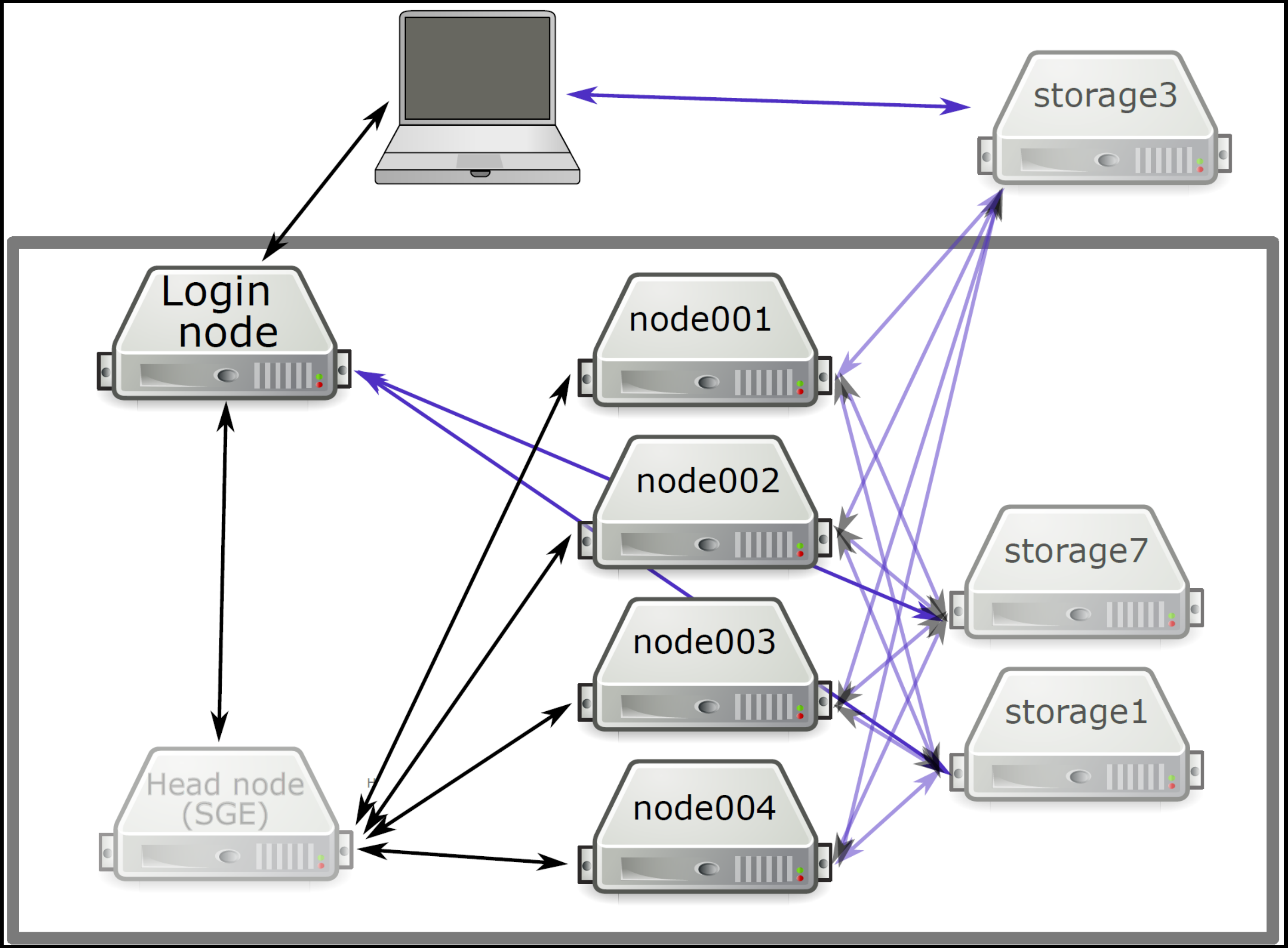
- Need: Java, Python, Spark
- See lab 1 on how to install on HPC
- To install on Windows
  - Lab 1 instructions: Install Java JRE, Python, Spark
  - Or pip install pyspark==3.0.1
- To install on Linux/Mac: see lab references

ShARC HPC @ Sheffield

- ShARC: Sheffield Advanced Research Computer
- VPN: a **MUST** unless you are on a campus network
- SSH access via sharc.sheffield.ac.uk
  - Windows: MobaXTerm
  - Linux/MAC OS: terminal (command line)
- Help: hpc@sheffield.ac.uk



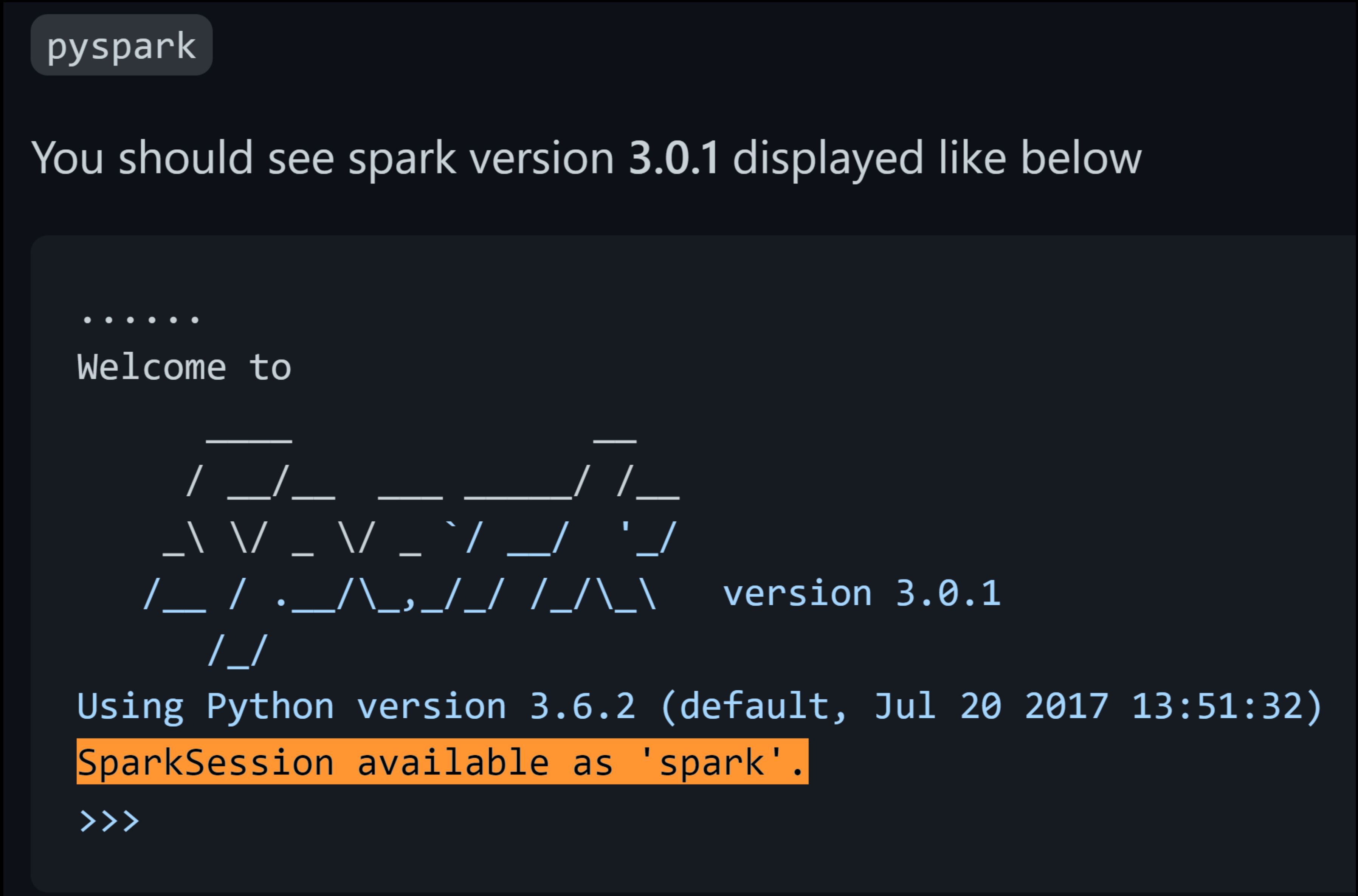
HPC Cluster Structure



Storage

Location	Shared	Quota	Back ups	Speed	Suitable for?
/home/$USER	Y	10GB	Y	>	Personal data
/data/$USER	Y	100GB	Y	>	Personal data
/fastdata/$USER	Y	-	N	>>>	Temporary big files
/scratch	N	-	N	>>>	Temporary small files

Interactive Session

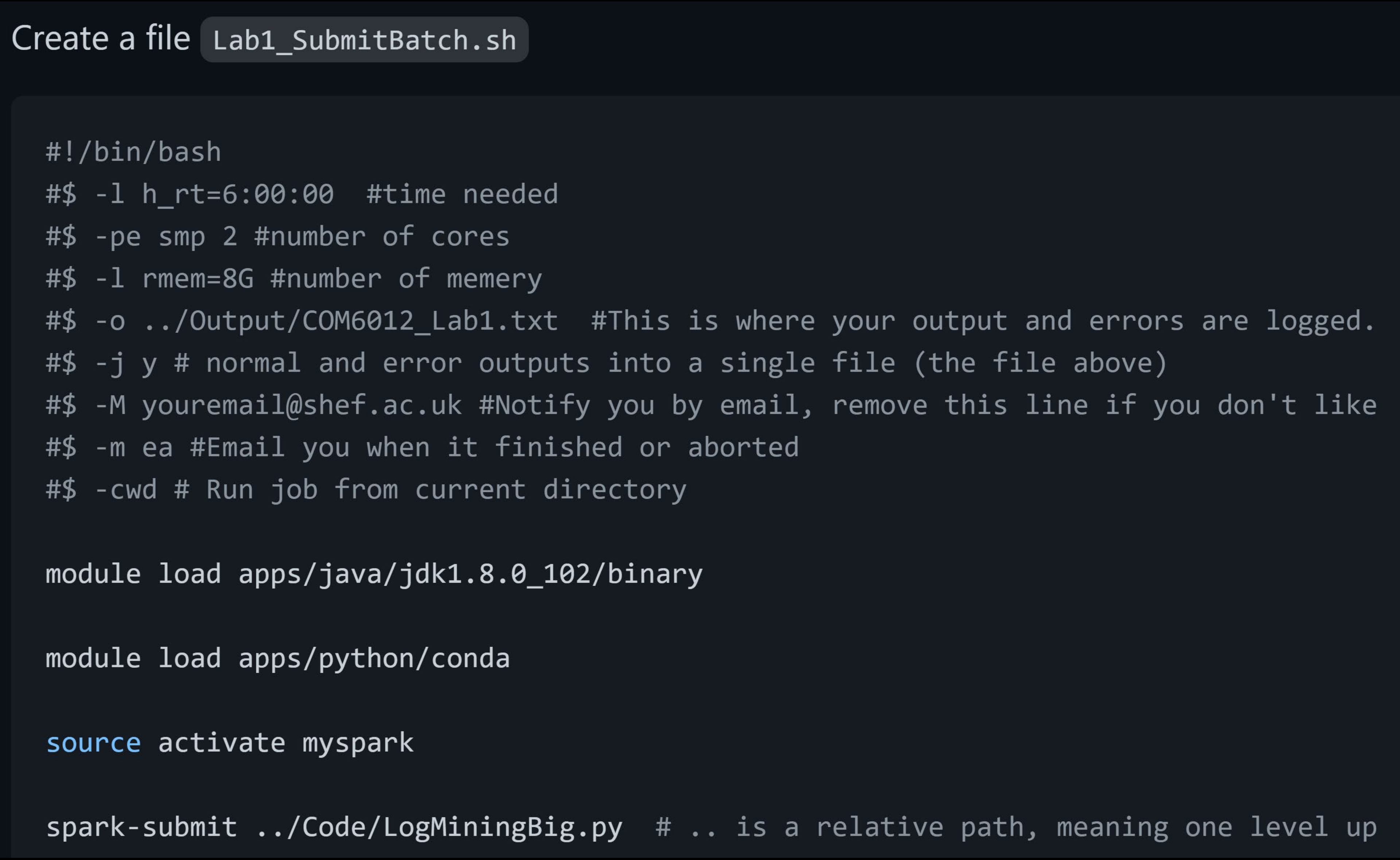


`pyspark`

You should see spark version **3.0.1** displayed like below

```
......
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 3.0.1
      /_/
Using Python version 3.6.2 (default, Jul 20 2017 13:51:32)
SparkSession available as 'spark'.
>>>
```

Batch Session – Shell Script xx.sh



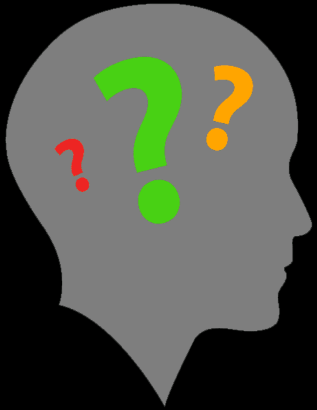
Create a file `Lab1_SubmitBatch.sh`

```
#!/bin/bash
#$ -l h_rt=6:00:00  #time needed
#$ -pe smp 2 #number of cores
#$ -l rmem=8G #number of memery
#$ -o ../Output/COM6012_Lab1.txt  #This is where your output and errors are logged.
#$ -j y # normal and error outputs into a single file (the file above)
#$ -M youremail@shef.ac.uk #Notify you by email, remove this line if you don't like
#$ -m ea #Email you when it finished or aborted
#$ -cwd # Run job from current directory

module load apps/java/jdk1.8.0_102/binary

module load apps/python/conda

source activate myspark

spark-submit ../Code/LogMiningBig.py  # .. is a relative path, meaning one level up
```

Batch Session: Submit & Relax

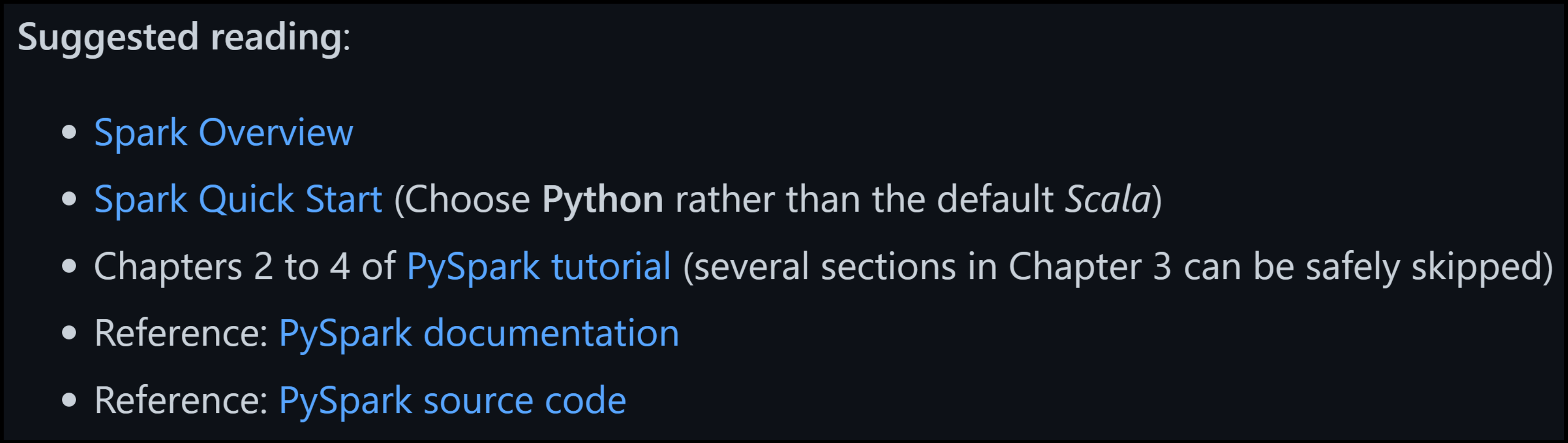
- qsub your job (can run at the login node): see Lab 1
- Then?
  - Close the terminal and leave
  - Wait for pre-set email notification
  - Check status: qstat
  - Cancel/amend job: qdel

- How much resources to request
  1. Run short test jobs
  2. View resource utilisation
  3. Extrapolate
  4. Submit larger jobs



Spark Resources

- Apache Spark Documentation (3.0.1)
- PySpark tutorial (last update Feb 2020)
- Spark videos on YouTube
- Open source code
- Suggested reading in labs



Suggested reading:

- Spark Overview
- Spark Quick Start (Choose **Python** rather than the default *Scala*)
- Chapters 2 to 4 of PySpark tutorial (several sections in Chapter 3 can be safely skipped)
- Reference: PySpark documentation
- Reference: PySpark source code

Acknowledgements

- Some slides (sec. 1) are modified from the "Introduction to Apache Spark" course by Prof. A. D. Joseph, University of California, Berkeley.
- This module benefits from many open resources. See the acknowledgement on our GitHub page.
- There are many other resources that I have consulted but somehow lost track of the origins.

Thank-you-word-cloud.jpg (967×522) (wikimedia.org)