

Lecture 1: Introduction to Spark and HPC

Shuo Zhou

COM6012: Scalable ML

Week 1 Objectives

- Identify key data types across the **data structure spectrum** and describe the **Extract–Transform–Load** (ETL) process.
- Summarise the challenges of the **big data problem**.
- Explain the **architecture, components, and resource management** of a Spark application, and implement simple PySpark programs.
- Explain the structure of the University's **high-performance computing (HPC) cluster**, and request resources to execute computational tasks.

Contents

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

Check-in code:
XX-XX-XX

Contents

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

Check-in code:
XX-XX-XX

Where Does Big Data Come From?

- Online activities:
 - Clicks
 - Billing events
 - Server requests
 - Transactions
 - Network messages
 - Faults
 - ...
- Data sharing platforms: web + mobile

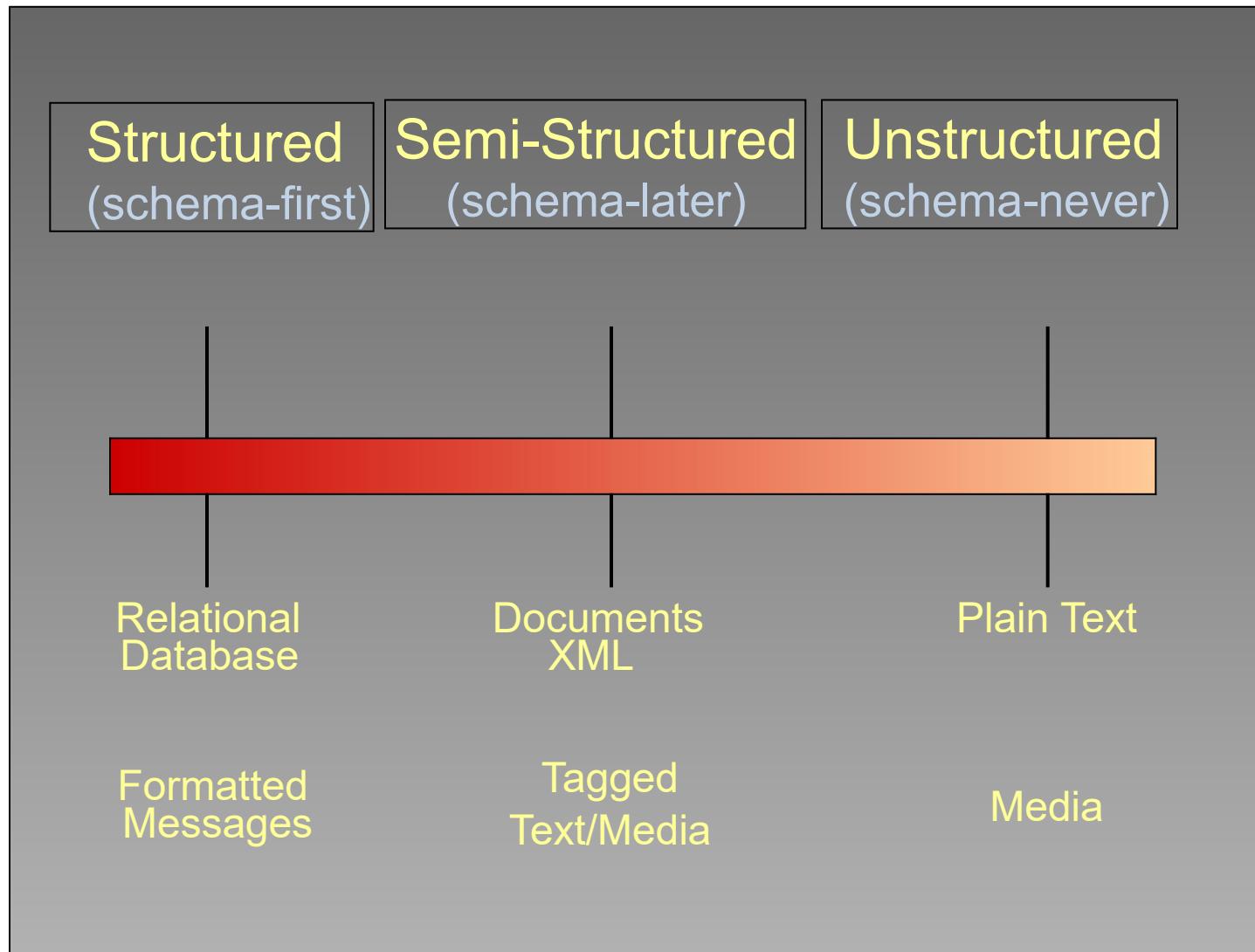


Where Does Big Data Come From?

AI generated content

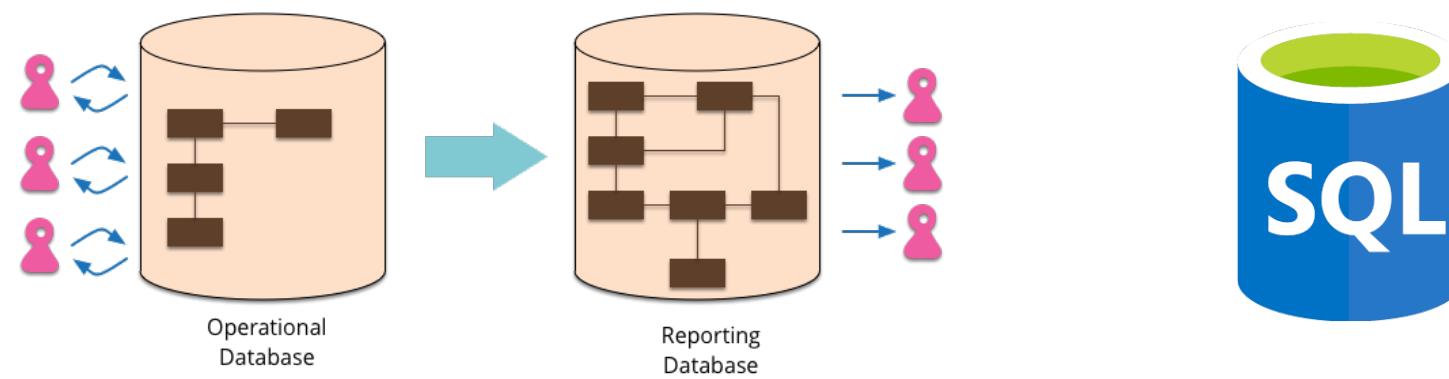


Data Structure Spectrum



Structured Data

- **Database:** relational data model → how a database is structured and used
- **Schema:** the organisation of data as a blueprint of how the database is constructed
 - The programmer **must statically specify** the schema
- **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:

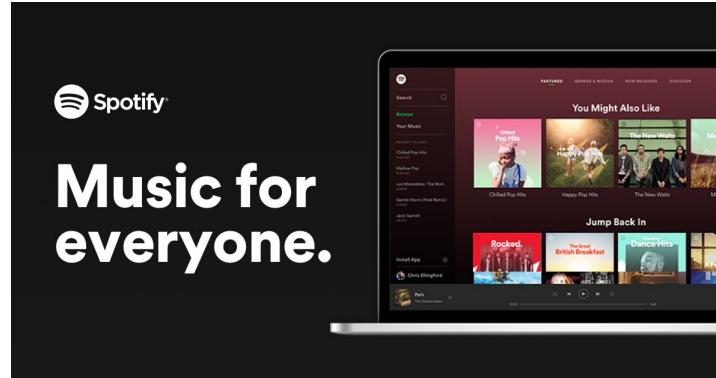
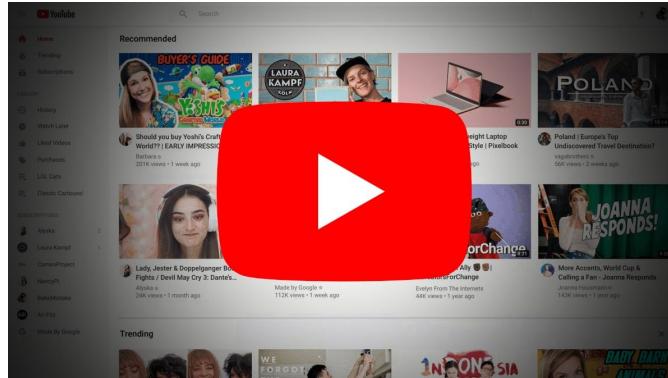


```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
       more questions later...-->
</quiz>
```

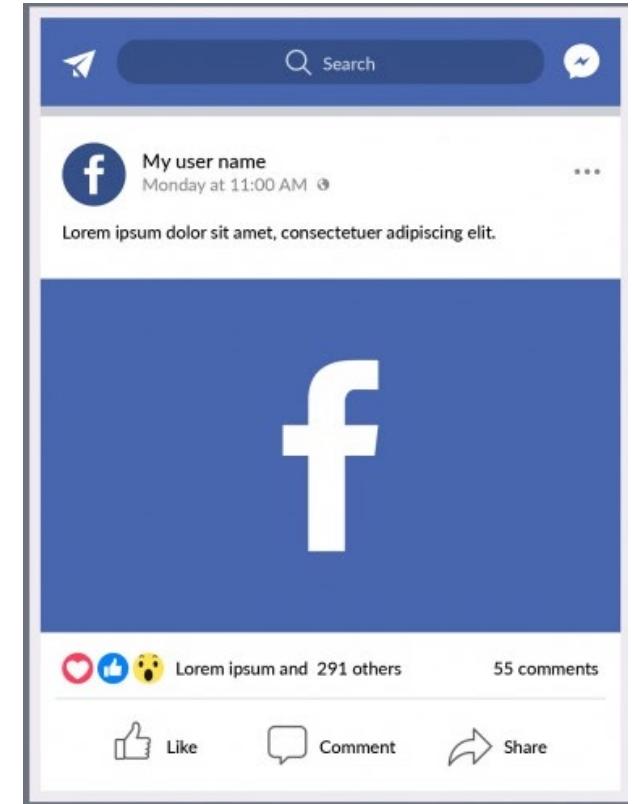


Unstructured Data

- Only one column with string or binary type
- Examples

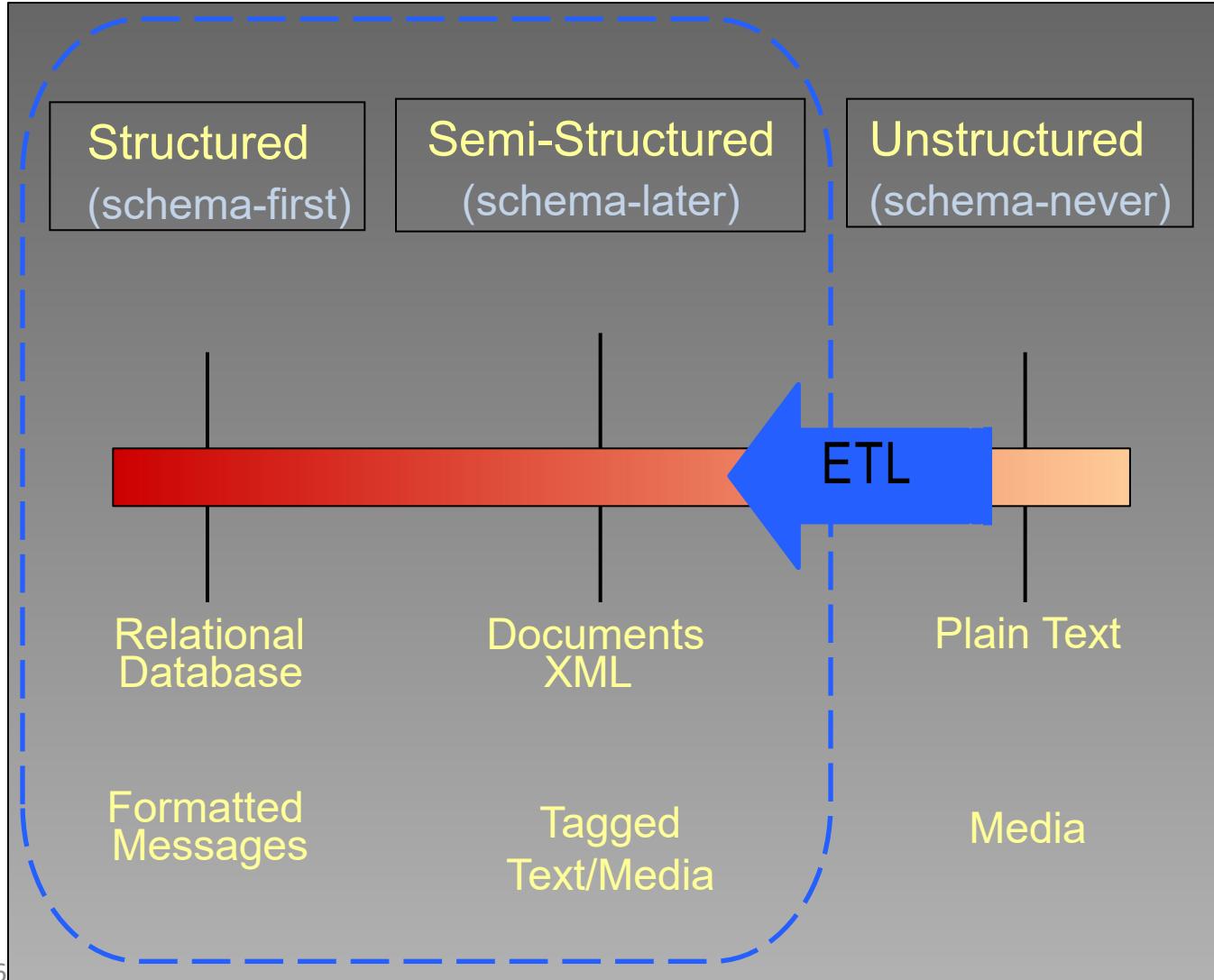


- Note: File formats ≠ data structure



<https://cdn.wccftech.com/wp-content/uploads/2019/11/YouTube-Redesign-2019-768x432.jpg>
<https://th.bing.com/th/id/OIP.MNR8Ck5DWZb32tkqOfLuXAAAAA?rs=1&pid=ImgDetMain>
https://files.codingninjas.in/article_images/preparation-guide-for-facebook-1-1662401488.webp

Traverse the Data Structure Spectrum



- Impose structure on unstructured data
 - [Extract](#)
 - [Transform](#)
 - [Load](#)

Traditional Analysis Tools

- Unix shell commands (awk, grep, ...)

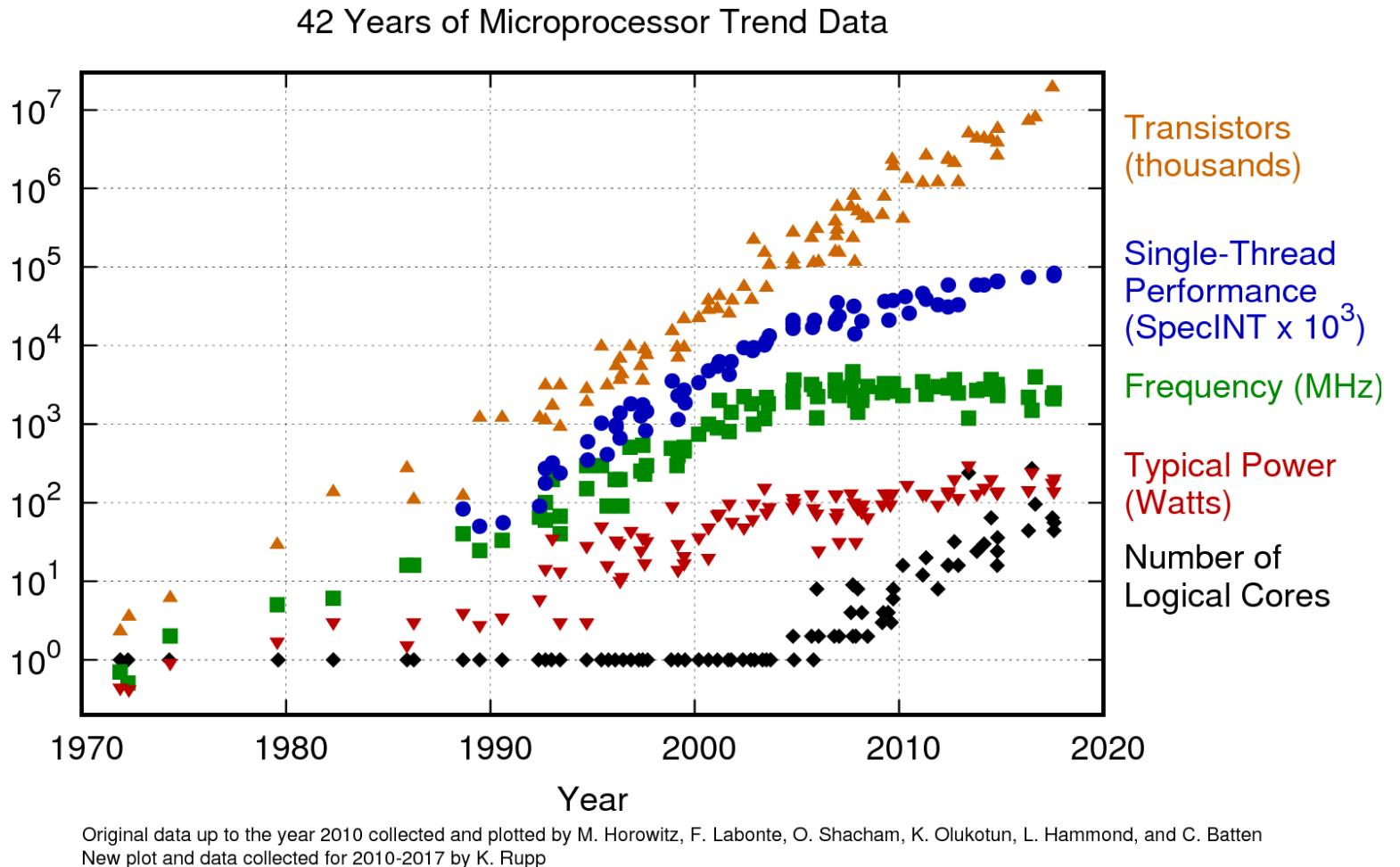
```
root@nginx:~# awk '{print $0}' file.txt
Item      Model      Country          Cost
1        BMW       Germany        $25000
2        Volvo     Sweden        $15000
3        Subaru    Japan         $2500
4        Ferrari  Italy       $2000000
5        SAAB      USA          $3000
```

```
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!

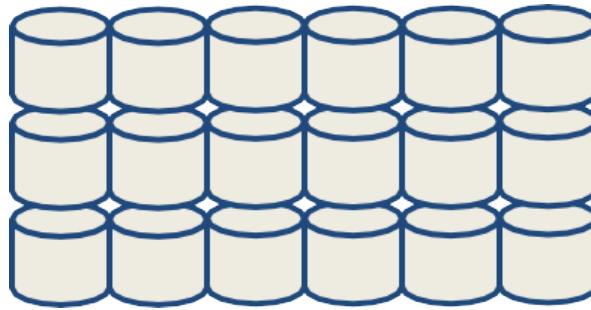
The Big Data Problem

- Data growing faster than computation speeds
- Growing data sources
 - Web, mobile, scientific, ...
- Storage getting cheaper
- But, stalling CPU speeds and storage bottlenecks

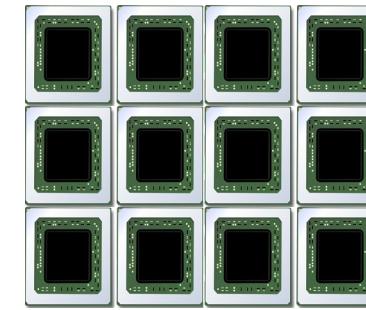


Solution for the Big Data Problem

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



Lots of hard drives



... and CPUs



... and memory!

Solution in this module



“Apache Spark is an industry-leading platform for distributed extract, transform, and load (ETL) workloads on large-scale data.”

– NVIDIA Technical Blog, Jun 12, 2023

Contents

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

Check-in code:
XX-XX-XX

Apache Spark

- Fast and general **cluster** computing system

- Interoperable with



- Improves efficiency through:

- In-memory computing primitives
 - General computation graphs

→ Up to 100× faster
(2-10× on disk)

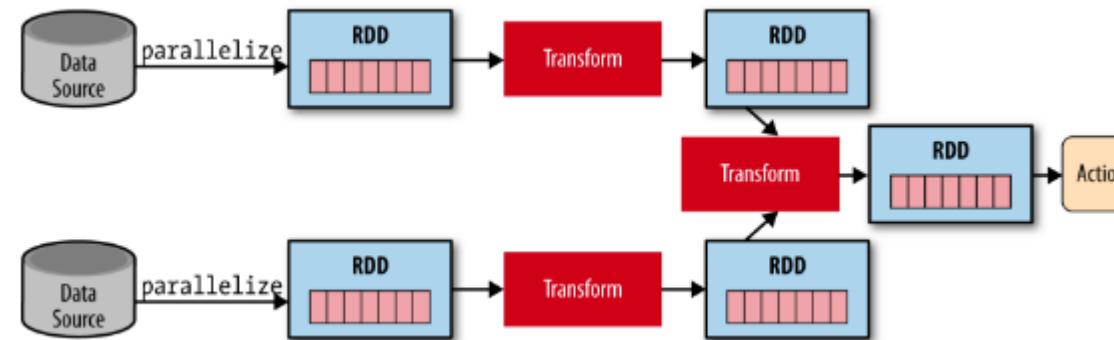
- Improves usability through:

- Rich APIs in Scala, Java, Python
 - Interactive shell

→ 2-5× less code

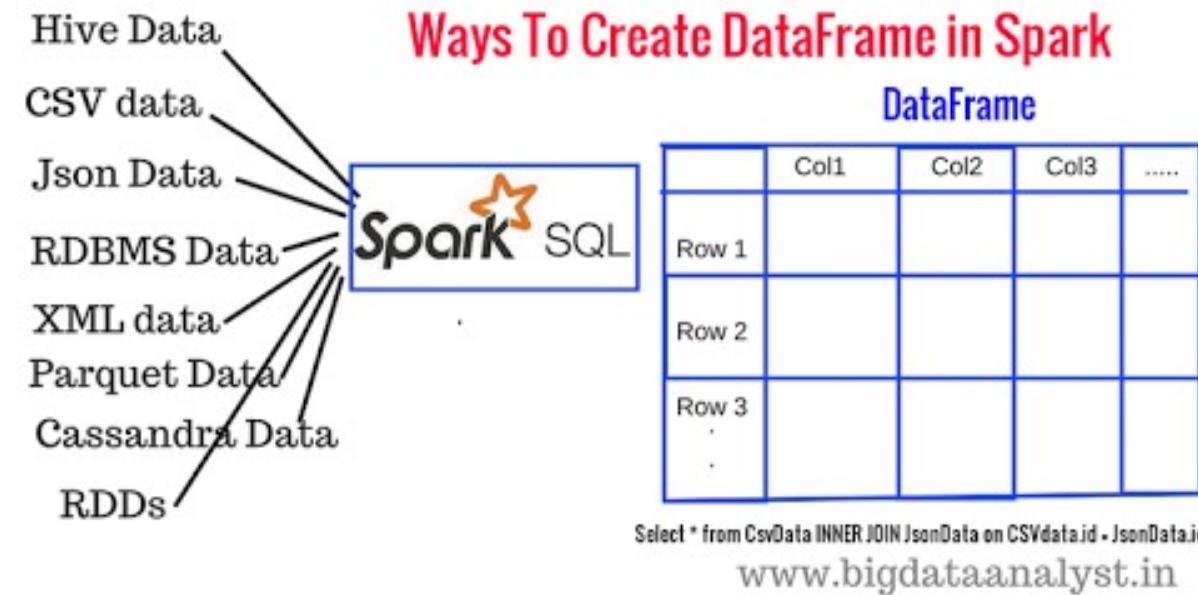
Spark Model

- 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](#)



Spark Computing Framework

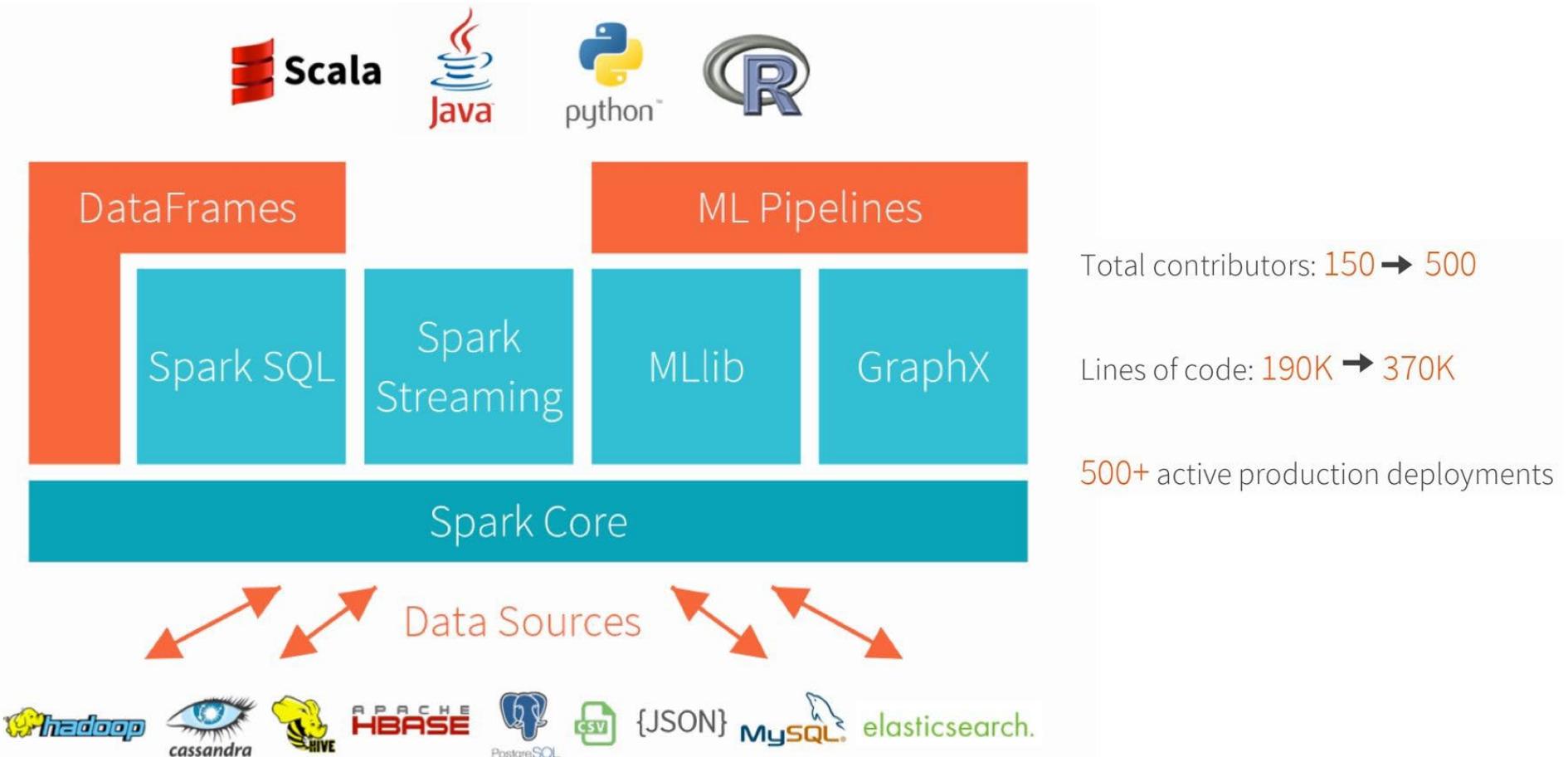
- Programming abstraction and parallel runtime to hide complexities of **fault-tolerance** and **slow** machines

“Here’s an operation, run it on all of the data”

JUST DO IT.

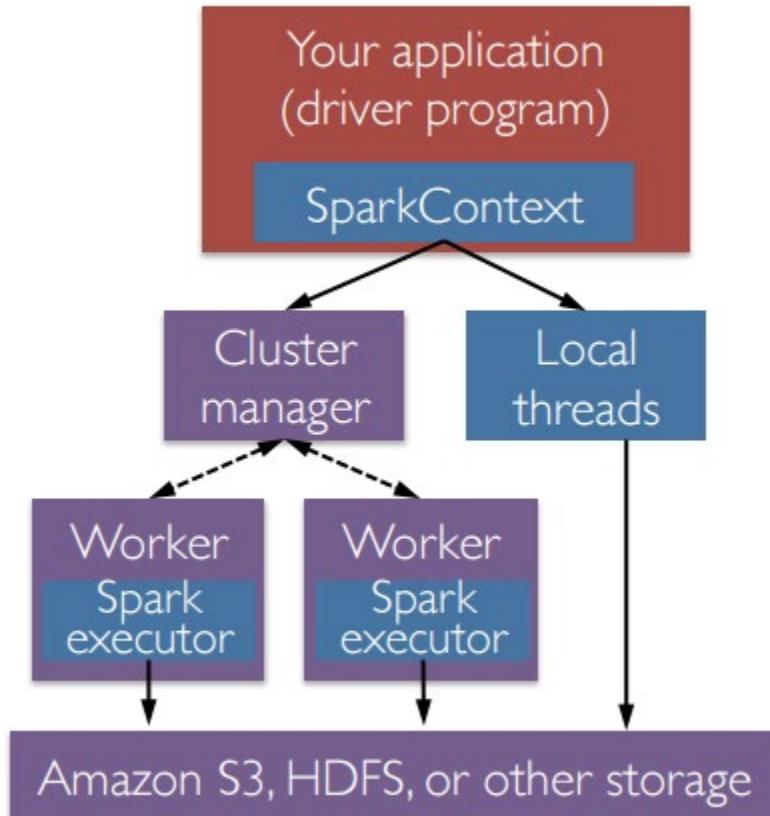
- Don’t care where it runs (you schedule that)
- In fact, feel free to run it twice on different nodes (e.g. when it fails)

Apache Spark Ecosystem



<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):**
 - Access data storage
 - Run computations

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)

Contents

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

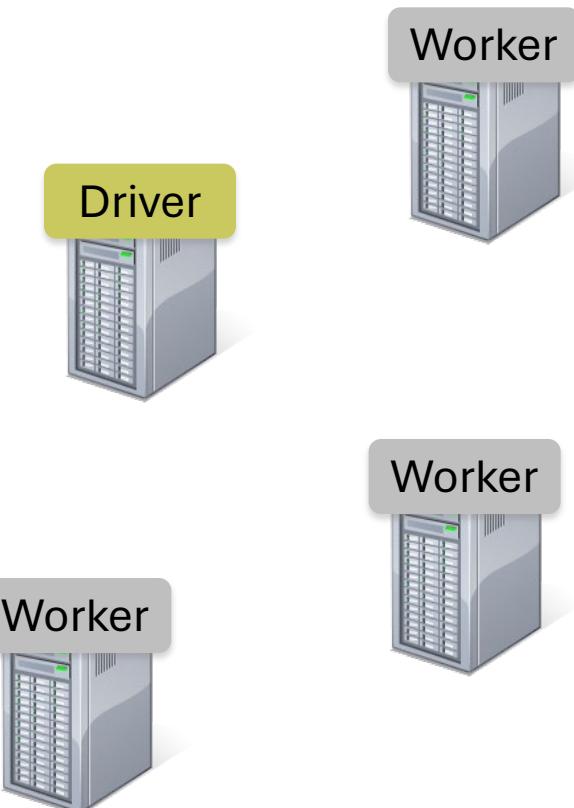
Check-in code:
XX-XX-XX

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

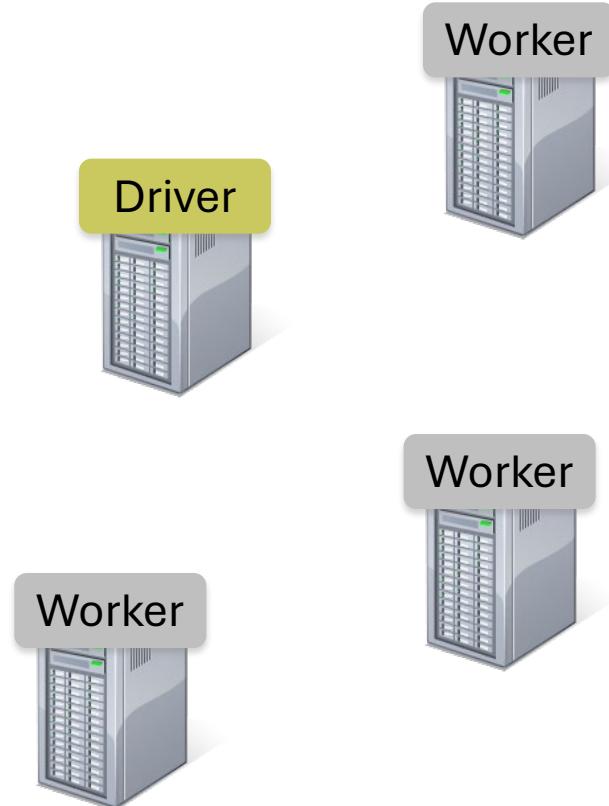
Load error messages from a log into memory, then interactively search for various patterns



Spark Example: Log Mining

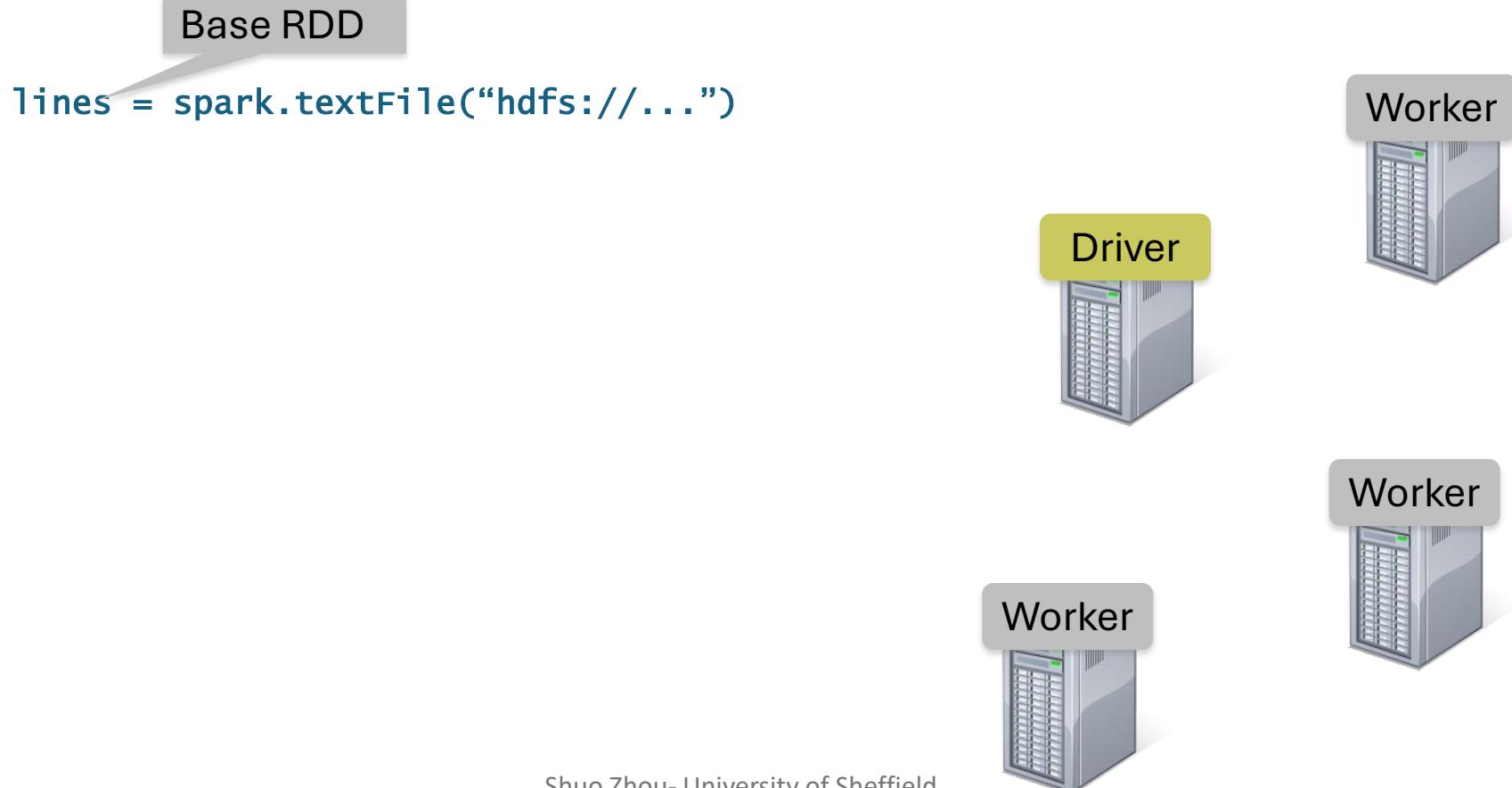
Load error messages from a log into memory, then interactively search for various patterns

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



Spark Example: Log Mining

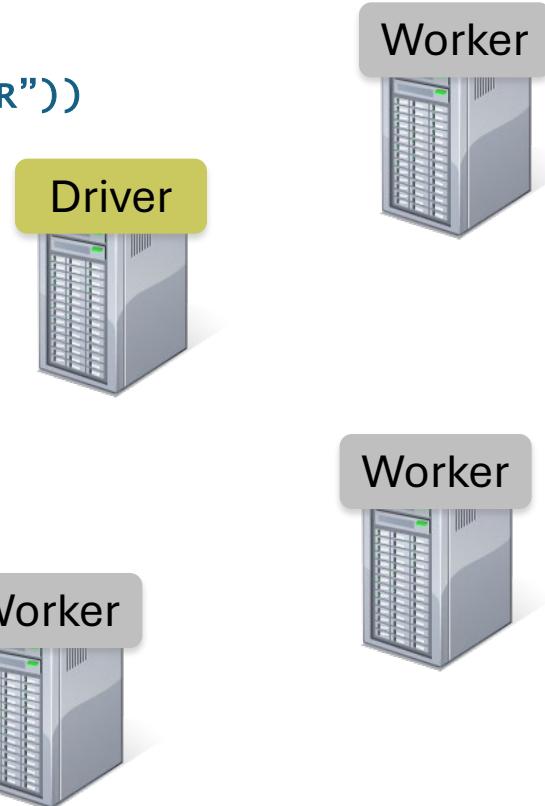
Load error messages from a log into memory, then interactively search for various patterns



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"))
```

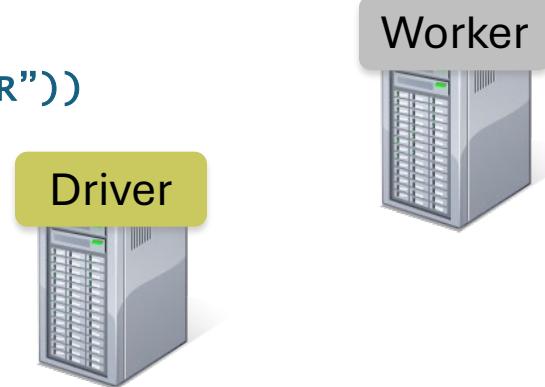


Spark Example: Log Mining

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()
```

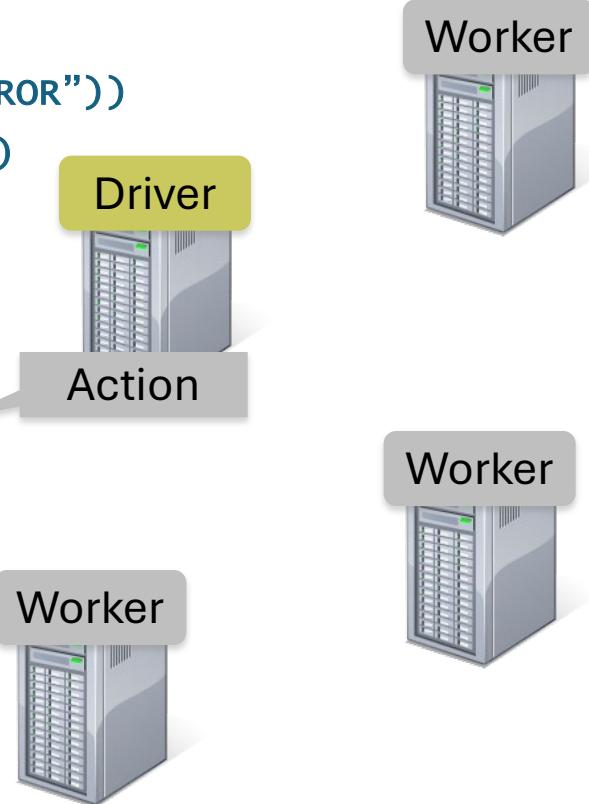
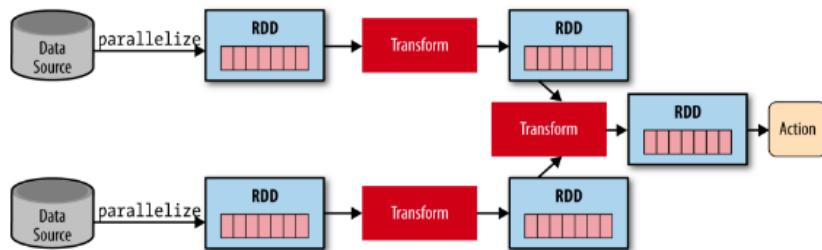


Spark Example: Log Mining

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()
```

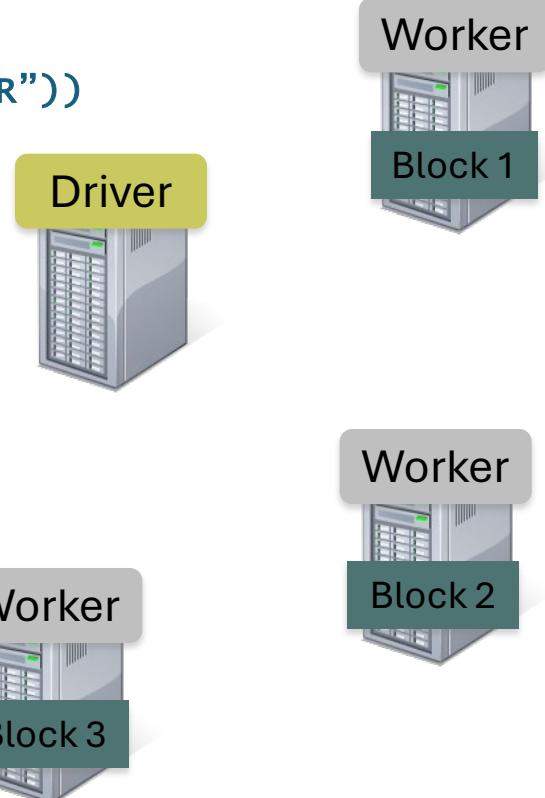


Spark Example: Log Mining

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()
```

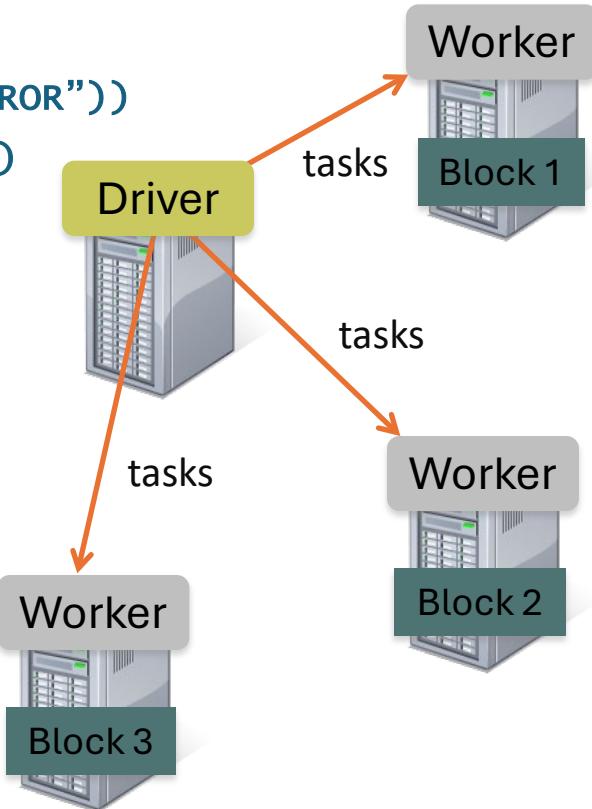


Spark Example: Log Mining

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()
```

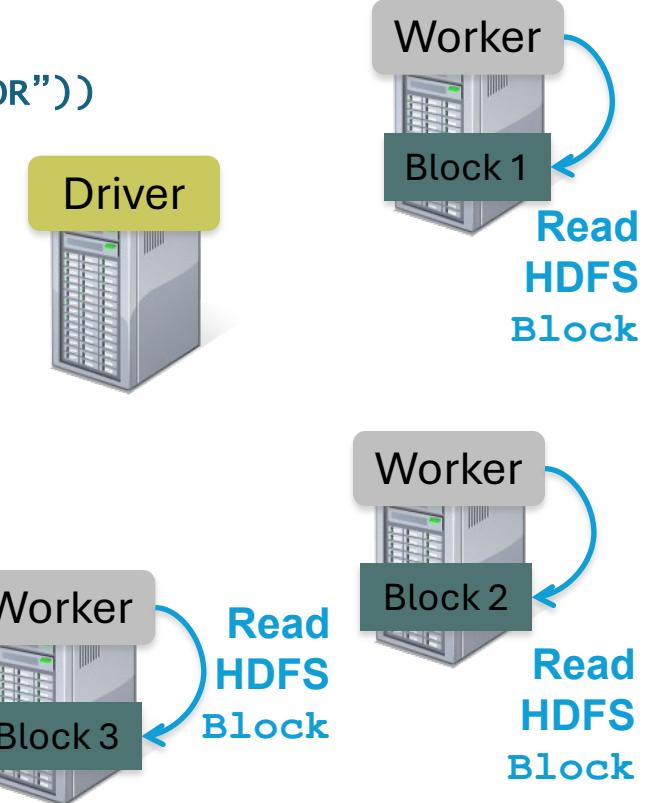


Spark Example: Log Mining

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()
```

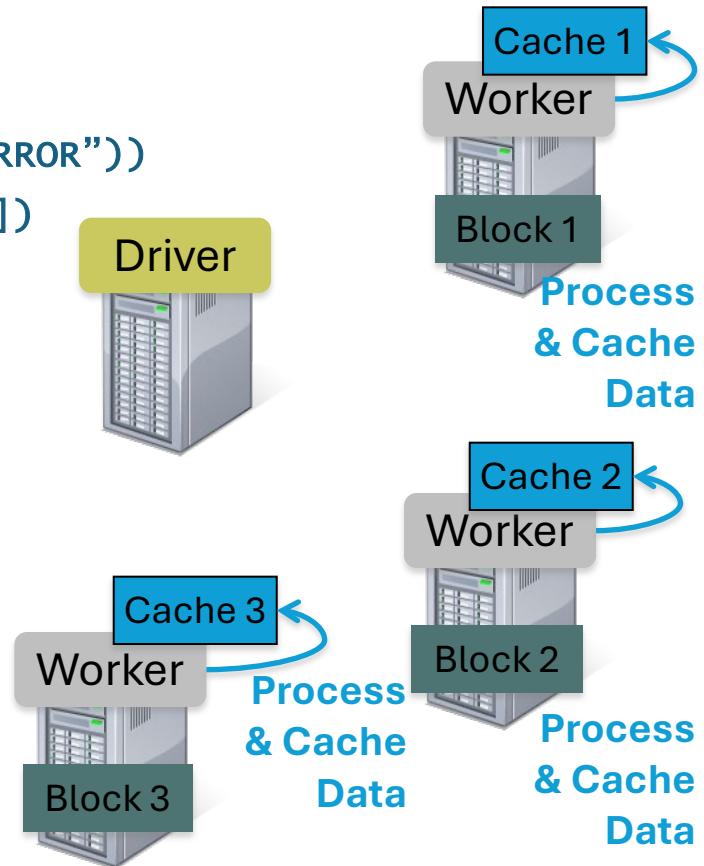


Spark Example: Log Mining

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()
```

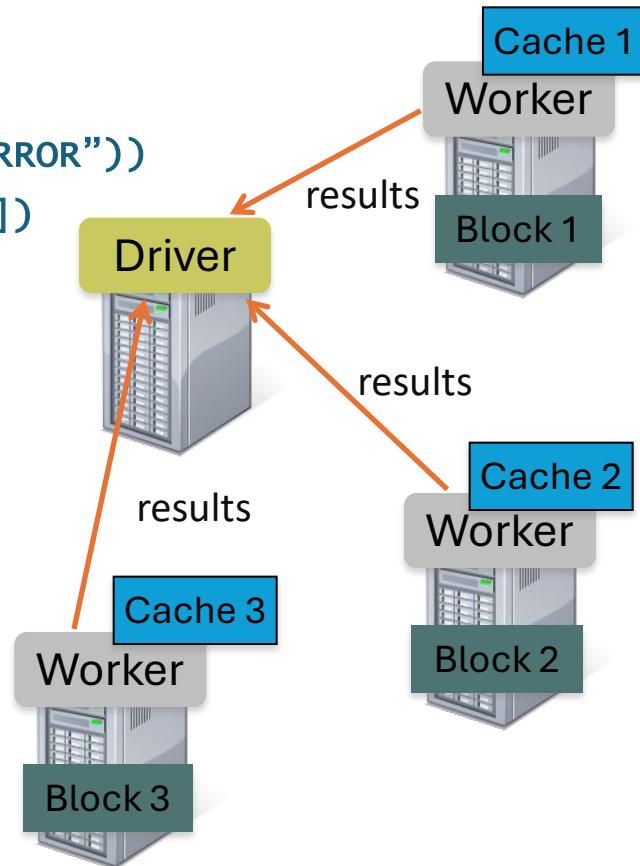


Spark Example: Log Mining

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()
```

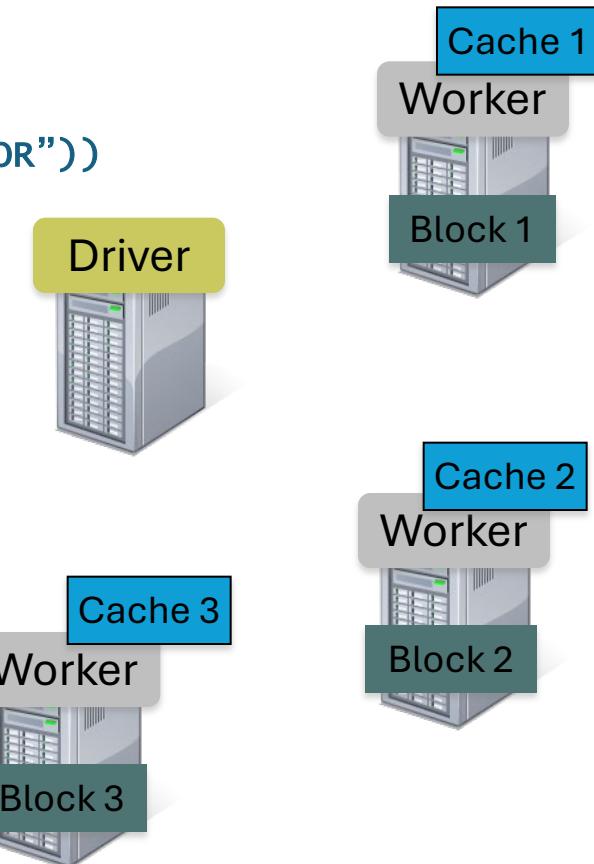


Spark Example: Log Mining

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()
```

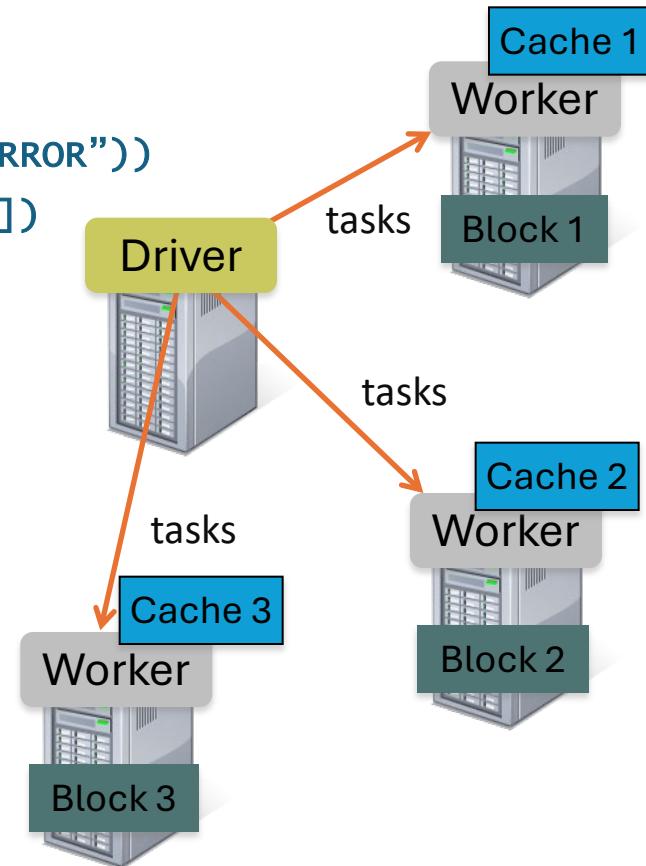


Spark Example: Log Mining

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()
```

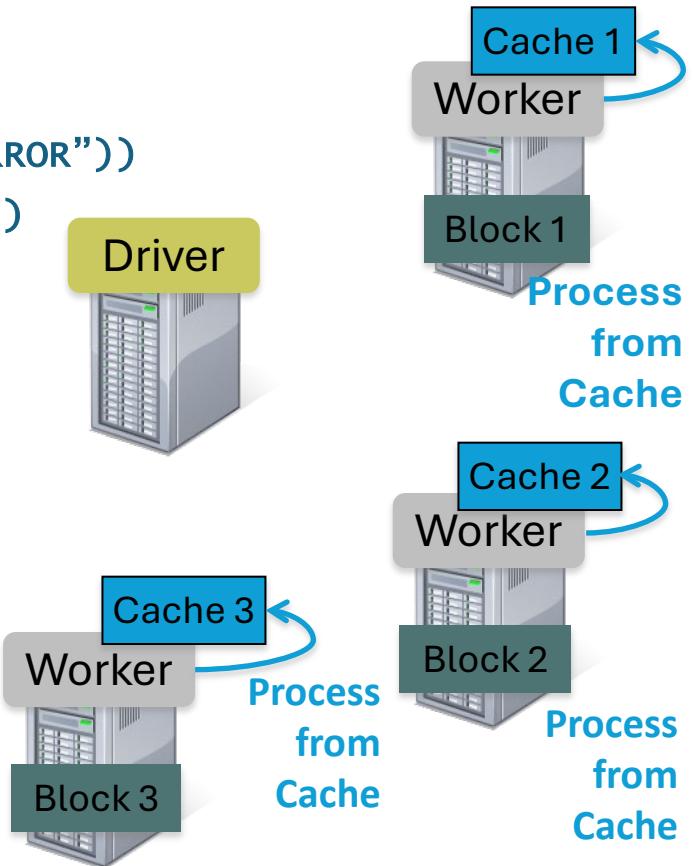


Spark Example: Log Mining

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()
```

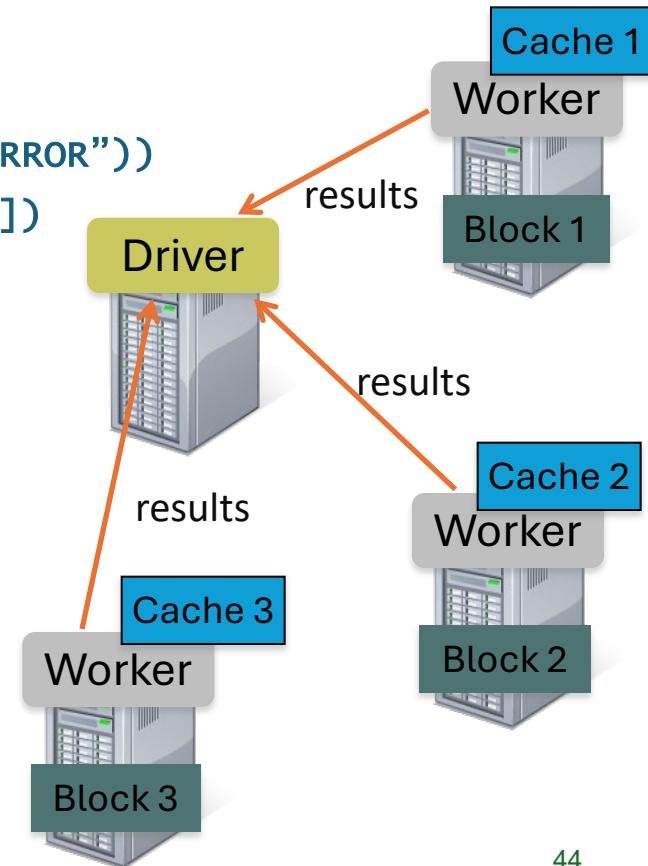


Spark Example: Log Mining

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

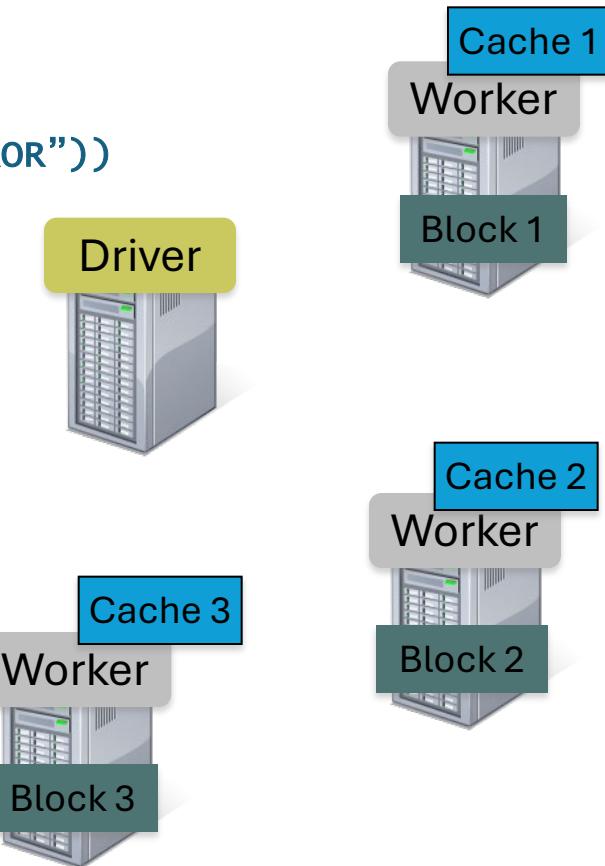
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()
```

```
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



Spark Program Lifecycle

- Create DataFrames from external data or [create DataFrame](#) 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](#)

Contents

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

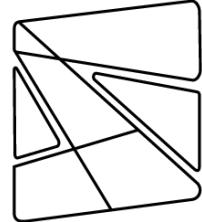
Check-in code:
XX-XX-XX

PySpark 4.1.0

- Need: Java, Python, Spark
- See lab 1 on how to install on HPC
- To install on Windows (optional)
 - [Lab 1 instructions](#): Install Java JRE, Python, Spark
 - Or pip install pyspark==4.1.0
- To install on Linux/Mac (optional): see lab references



University of Sheffield's HPC

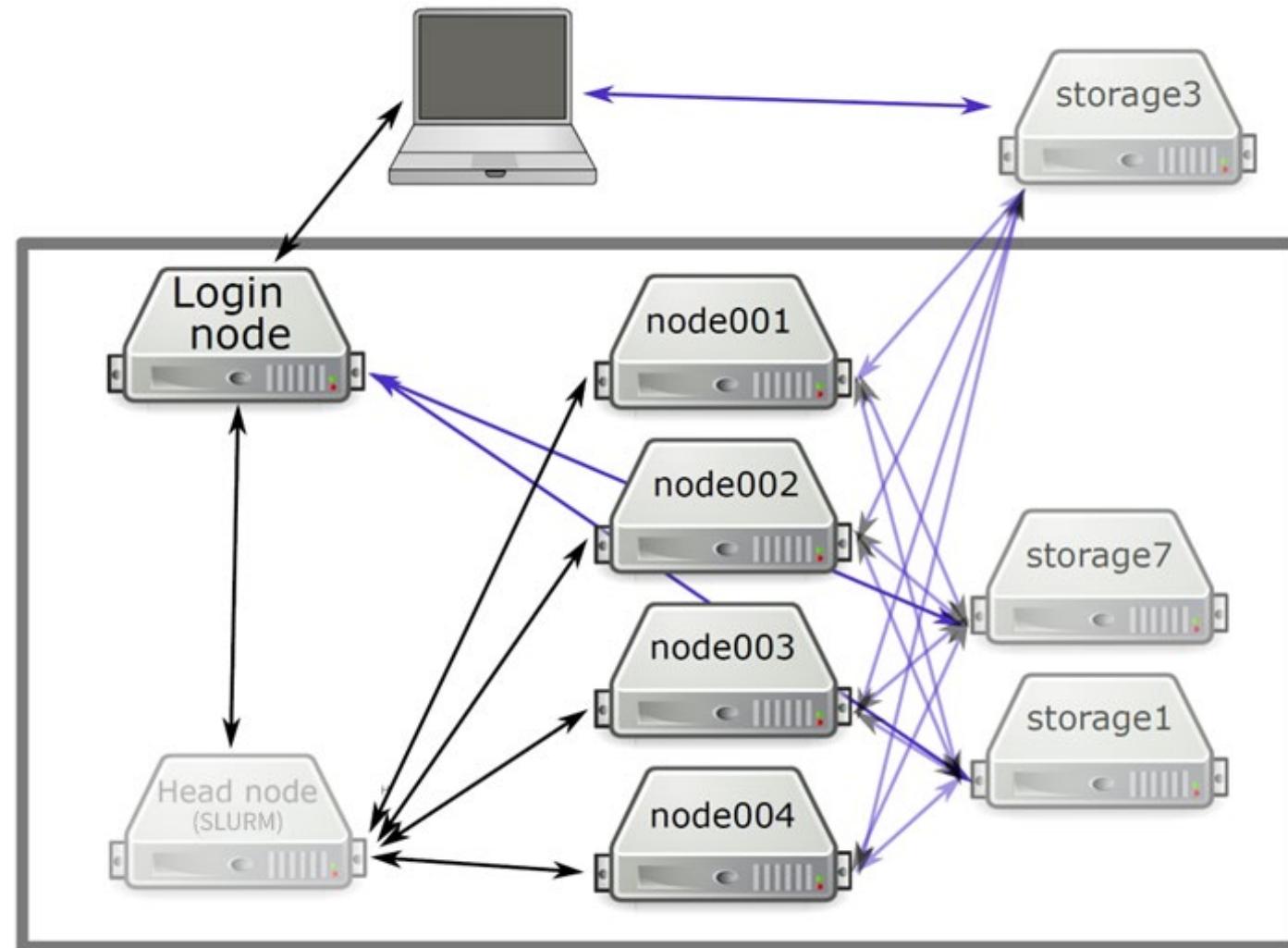


S T A N A G E



- Account created for you already!
- Training (due 13th Feb Thursday-AS0): [HPC Driving License test](#)
- SSH access via stancode.sheffield.ac.uk
 - Windows: MobaXTerm
 - Linux/MAC OS: terminal (command line)
- [VPN](#): a **MUST** for when connecting on campus using Eduroam or off campus

HPC Cluster Structure



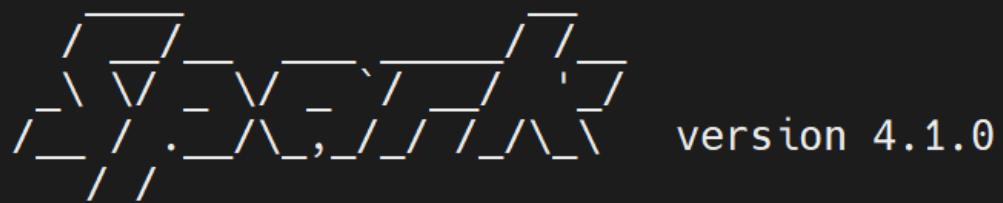
Storage

Location	Quota	Speed	Suitable for?
/users/\$USER	50GB	>	Personal data
/mnt/parscratch/	No limits	>>>	Temporary large files
/tmp	No limits	>>>	Temporary lots of small files

More information available at: <https://docs.hpc.shef.ac.uk/en/latest/hpc/filestore.html>

Interactive Session

```
Python 3.13.5 | packaged by Anaconda, Inc. | (main, Jun 12 2025, 16:09:02) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
WARNING: Using incubator modules: jdk.incubator.vector
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
26/02/03 14:52:48 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... us
ing builtin-java classes where applicable
Welcome to
```



```
Using Python version 3.13.5 (main, Jun 12 2025 16:09:02)
Spark context Web UI available at http://node001.pri.stanage.alces.network:4040
Spark context available as 'sc' (master = local[*], app id = local-1770130369562).
SparkSession available as 'spark'.
>>> |
```

Batch Session – Shell Script xx.sh

Create a file `Lab1_SubmitBatch.sh`

```
#!/bin/bash
#SBATCH --time=00:30:00 # Change this to a longer time if you need more time
#SBATCH --nodes=1 # Specify a number of nodes
#SBATCH --mem=4G # Request 4 gigabytes of real memory (mem)
#SBATCH --output=./Output/COM6012_Lab1.txt # This is where your output and errors are logged
#SBATCH --mail-user=username@sheffield.ac.uk # Request job update email notifications

module load Java/17.0.4
module load Anaconda3/2024.02-1

source activate myspark

spark-submit ./Code/LogMiningBig.py
```

Batch Session: Submit & Relax

- **sbatch** 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: **squeue**
 - Cancel job: **scancel**
- How much resources to request
 1. Run **short** test jobs
 2. View resource utilisation
 3. Extrapolate
 4. Submit larger jobs



<https://assets.entrepreneur.com/content/3x2/2000/20150216153034-laptop-desk-glasses.jpeg?format=pjpeg&auto=webp>
<https://th.bing.com/th/id/OIP.P2L5k-Z44bVifj1bnm6-SwAAAA?w=300&h=300&rs=1&pid=ImgDetMain>

Summary of key concepts



Data structure spectrum

Structured, semi-structured, unstructured data, schema, ETL



Spark's computing model and framework

RDDs, DataFrames, parallel transformation, fault-tolerance, ...



Components and lifecycle of a spark programme

SparkSession and SparkContext
Creating DataFrames, lazy transformations, caching, and actions



Key features of the Stanage cluster

HPC structure, storage, interactive vs. batch session

Spark Resources

- [Apache Spark open-source repository](#)
- [Apache Spark Documentation](#)
- [Learning Apache Spark with Python](#) (Chapters 2 and 4)
- 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 4.1.0 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 may somehow lost track of the origins.



[Thank-you-word-cloud.jpg \(967x522\) \(wikimedia.org\)](#)

Image sources for the logos

- Page 5:
 - https://upload.wikimedia.org/wikipedia/commons/thumb/b/b9/2023_Facebook_icon.svg/1024px-2023_Facebook_icon.svg.png
 - https://upload.wikimedia.org/wikipedia/en/thumb/a/a9/TikTok_logo.svg/1920px-TikTok_logo.svg.png
 - <https://mmoculture.com/wp-content/uploads/2019/12/Bilibili-logo-768x307.png>
 - https://upload.wikimedia.org/wikipedia/commons/thumb/2/20/YouTube_2024.svg/1920px-YouTube_2024.svg.png
 - https://upload.wikimedia.org/wikipedia/commons/thumb/0/06/Amazon_2024.svg/330px-Amazon_2024.svg.png
 - https://upload.wikimedia.org/wikipedia/commons/thumb/c/ce/X_logo_2023.svg/150px-X_logo_2023.svg.png
 - https://upload.wikimedia.org/wikipedia/en/thumb/6/6e/Sina_Weibo.svg/225px-Sina_Weibo.svg.png
 - https://upload.wikimedia.org/wikipedia/commons/thumb/9/95/Instagram_logo_2022.svg/195px-Instagram_logo_2022.svg.png
- Page 6:
 - <https://upload.wikimedia.org/wikipedia/commons/thumb/e/ef/ChatGPT-Logo.svg/180px-ChatGPT-Logo.svg.png>
 - https://upload.wikimedia.org/wikipedia/commons/thumb/e/ec/DeepSeek_logo.svg/420px-DeepSeek_logo.svg.png
 - https://upload.wikimedia.org/wikipedia/commons/thumb/8/8a/Google_Gemini_logo.svg/460px-Google_Gemini_logo.svg.png
- Page 15:
 - https://upload.wikimedia.org/wikipedia/commons/thumb/f/f3/Apache_Spark_logo.svg/375px-Apache_Spark_logo.svg.png
- Page 17:
 - [Page https://www.simplilearn.com/ice9/free_resources_article_thumb/mapreduce_1.JPG](https://www.simplilearn.com/ice9/free_resources_article_thumb/mapreduce_1.JPG)
- Page 46:
 - <https://www.edureka.co/blog/wp-content/uploads/2018/07/PySpark-logo-1.jpeg>