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

Lecture 1: **Intro to Spark and HPC**

Haiping Lu - Scalable ML 2020

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

***Slides credit**: **Prof. A.D. Joseph, UC Berkeley**

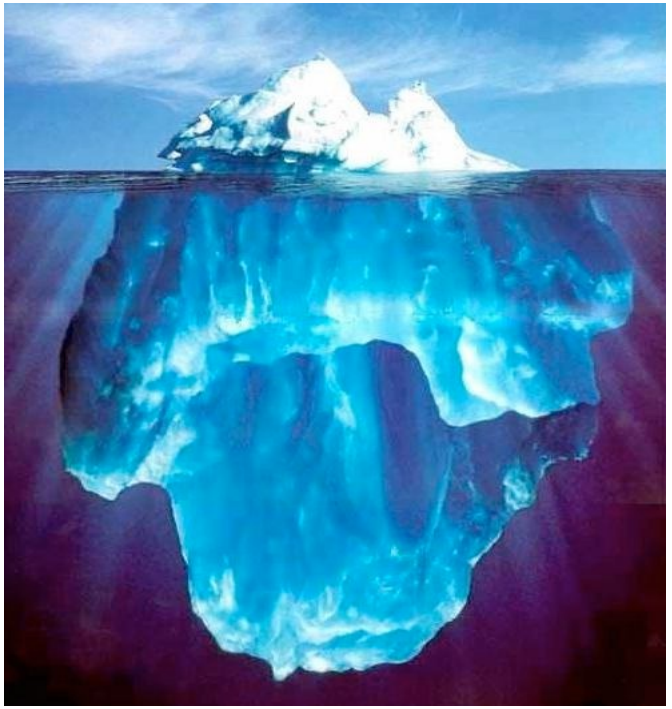
Week 1 Contents / Objectives

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

***Slides credit**: **Prof. A.D. Joseph, UC Berkeley**

Where Does Big Data Come From?

- It's all happening online – could record every:
  - Click
  - Ad impression
  - Billing event
  - Fast Forward, pause,…
  - Server request
  - Transaction
  - Network message
  - Fault
  - …

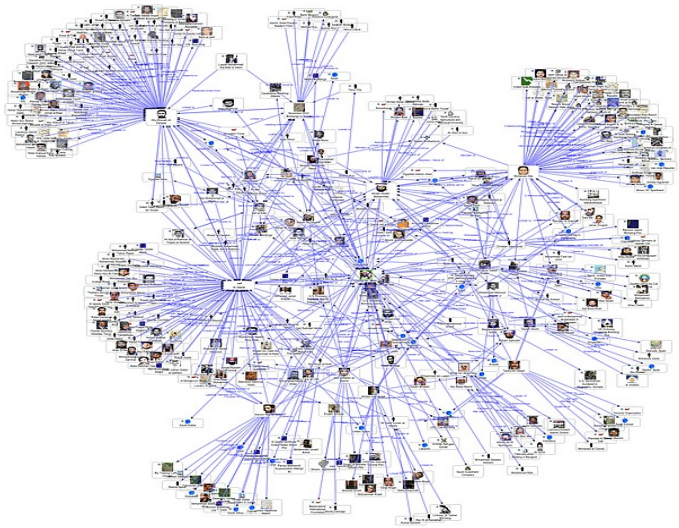


Where Does Big Data Come From?

- User Generated Content (Web & Mobile)
  - Facebook
  - Instagram
  - Yelp
  - TripAdvisor
  - Twitter
  - YouTube
  - …

Graph Data

- Lots of interesting data has a graph structure:
  - Social networks
  - Telecommunication Networks
  - Computer Networks
  - Road networks
  - Collaborations/Relationships
  - …



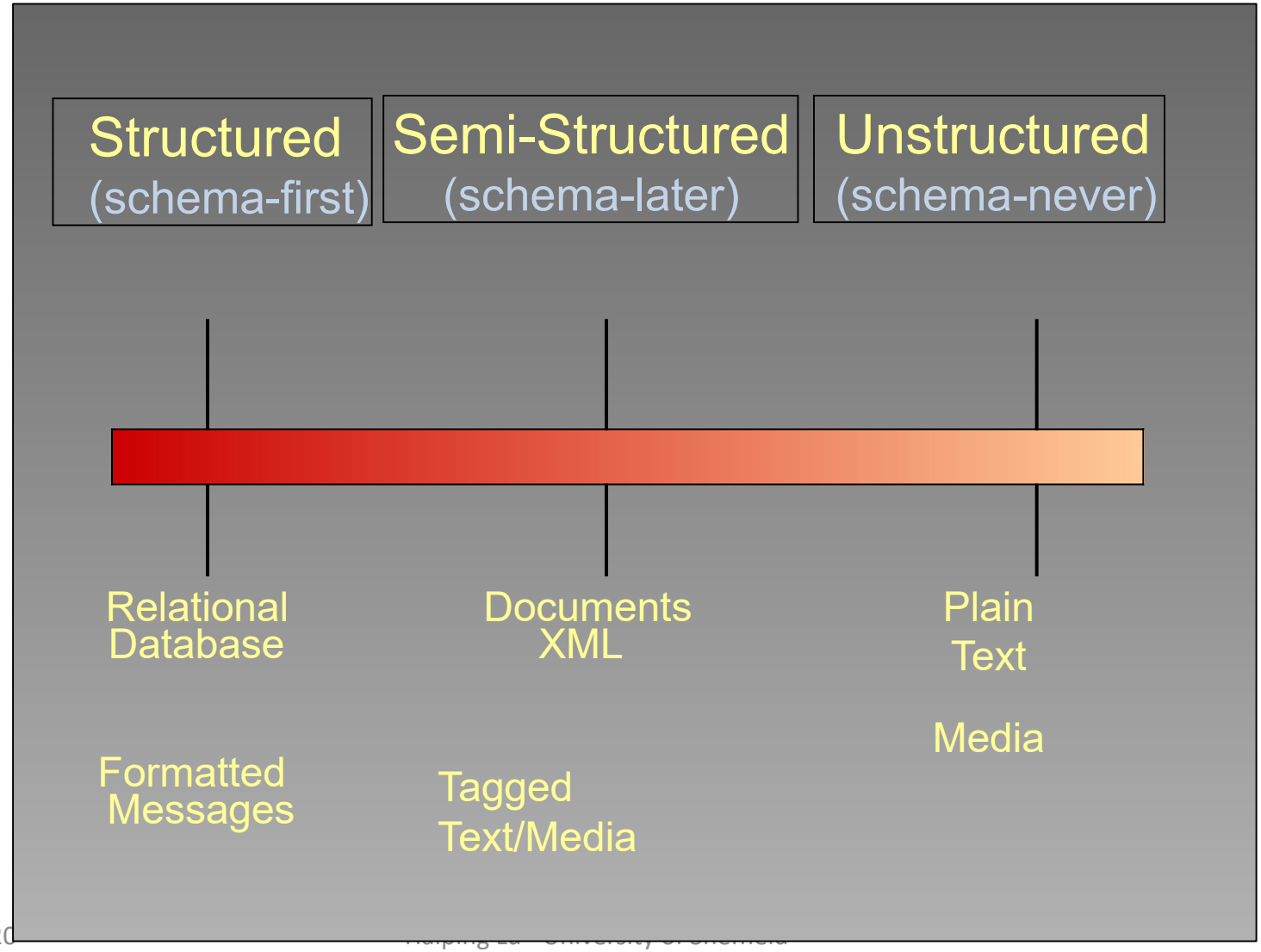
Imaging how big is the Facebook user graph

Key Data Management Concepts

- A **data model** is a collection of concepts for describing data

- A **schema** is a description of a particular collection of data, using a given data model

The Structure Spectrum



Structured Data
(Database: Hottest job 20+ years ago)

- **Database: relational data model** describing how a database is structured and used (from Wiki)
- **Schema:** the organization of data as a blueprint of how the database is constructed (from Wiki)
- The programmer **must statically specify** the schema
- Decreasing ← consumer/media app, enterprise search
- See https://en.wikipedia.org/wiki/Data_model
  https://en.wikipedia.org/wiki/Relational_model
- SQL: Structured Query Language

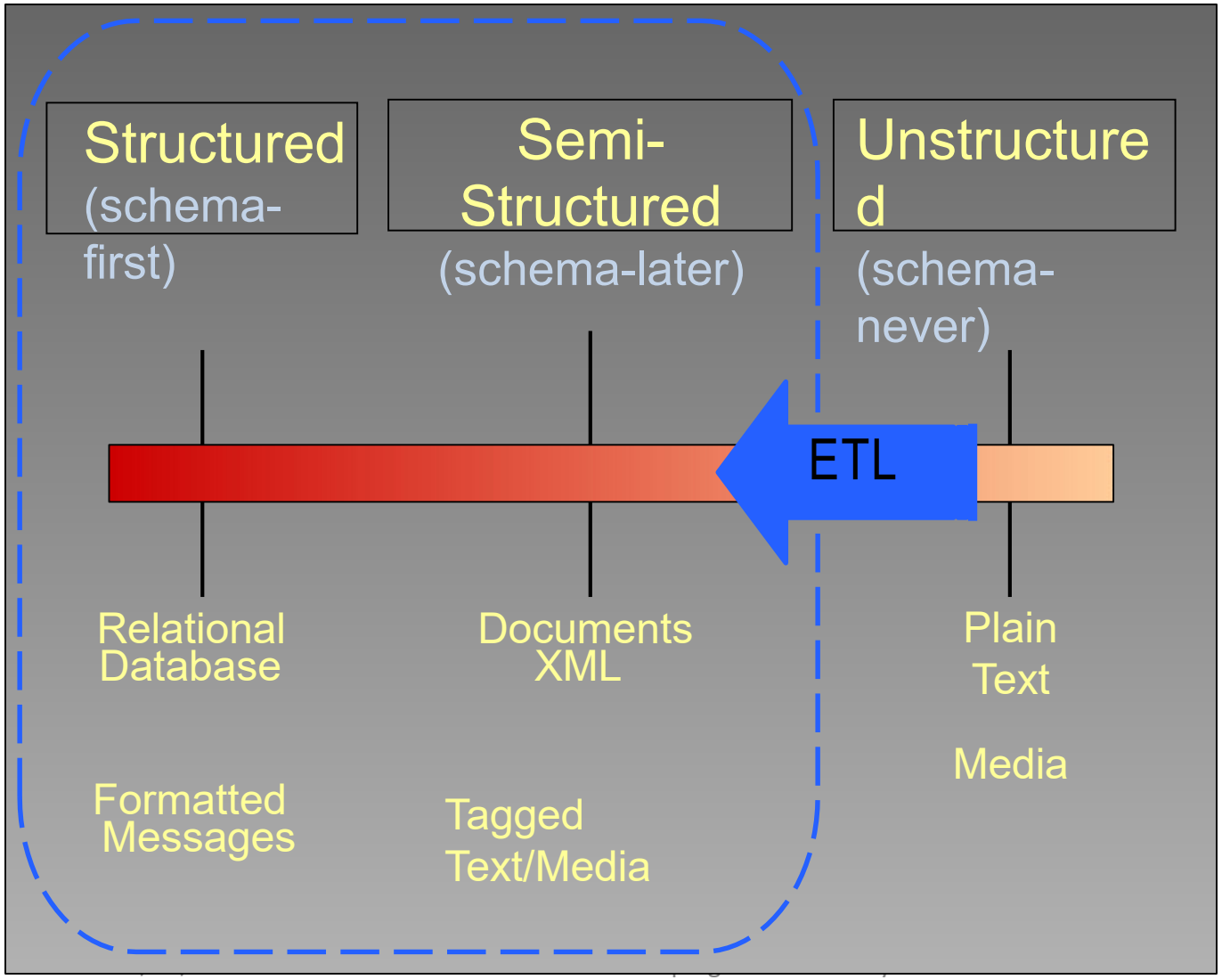
Semi-Structured Data

- **Self-describing** structures rather than formal structures, tags/markers to separate semantic elements (from wiki)
- The column types → the **schema** for the data
  - Spark dynamically infers the schema while reading each row
  - Programmer statically specifies the schema
- Increasingly occurring, XML, JSON

Unstructured Data

- Only one column with string or binary type
  Examples:
  - Facebook post
  - Instagram image
  - Vine video
  - Blog post
  - News article
  - User Generated Content
- More than 70%–80% of all data in organizations (Shilakes 1998)
- https://en.wikipedia.org/wiki/Unstructured_data

The Structure Spectrum



Extract-Transform-Load ➔

- **Impose structure on unstructured data**

Some Traditional Analysis Tools

- Unix shell commands (grep, awk, sed), pandas, R

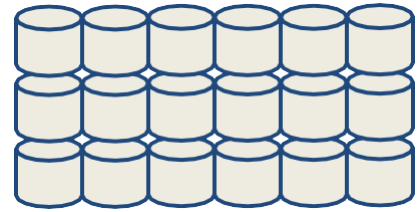
All run on a single machine!

The Big Data Problem

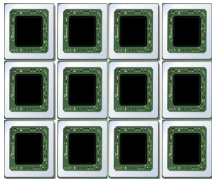
- Data growing faster than computation speeds
- Growing data sources
  - Web, mobile, scientific, …
- Storage getting cheaper
  - Size doubling every 18 months
- But, stalling CPU speeds and storage bottlenecks

The Big Data Problem

- **One machine** can not process or even store all the data!
- Solution is to **distribute** data over 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

***Slides credit**: **Prof. A.D. Joseph, UC Berkeley**

Apache Spark

- Fast and general cluster computing system, interoperable with Hadoop

- Improves efficiency through:
  - In-memory computing primitives
  - General **computation graphs**
- Improves usability through:
  - Rich APIs in Scala, Java, **Python**
  - Interactive shell

→ Up to 100× faster (2-10× on disk)

→ 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
  - Familiar API based on R & Python Pandas
  - Distributed, optimized implementation

- Machine Learning Pipelines
  - Simple construction and tuning of ML workflows

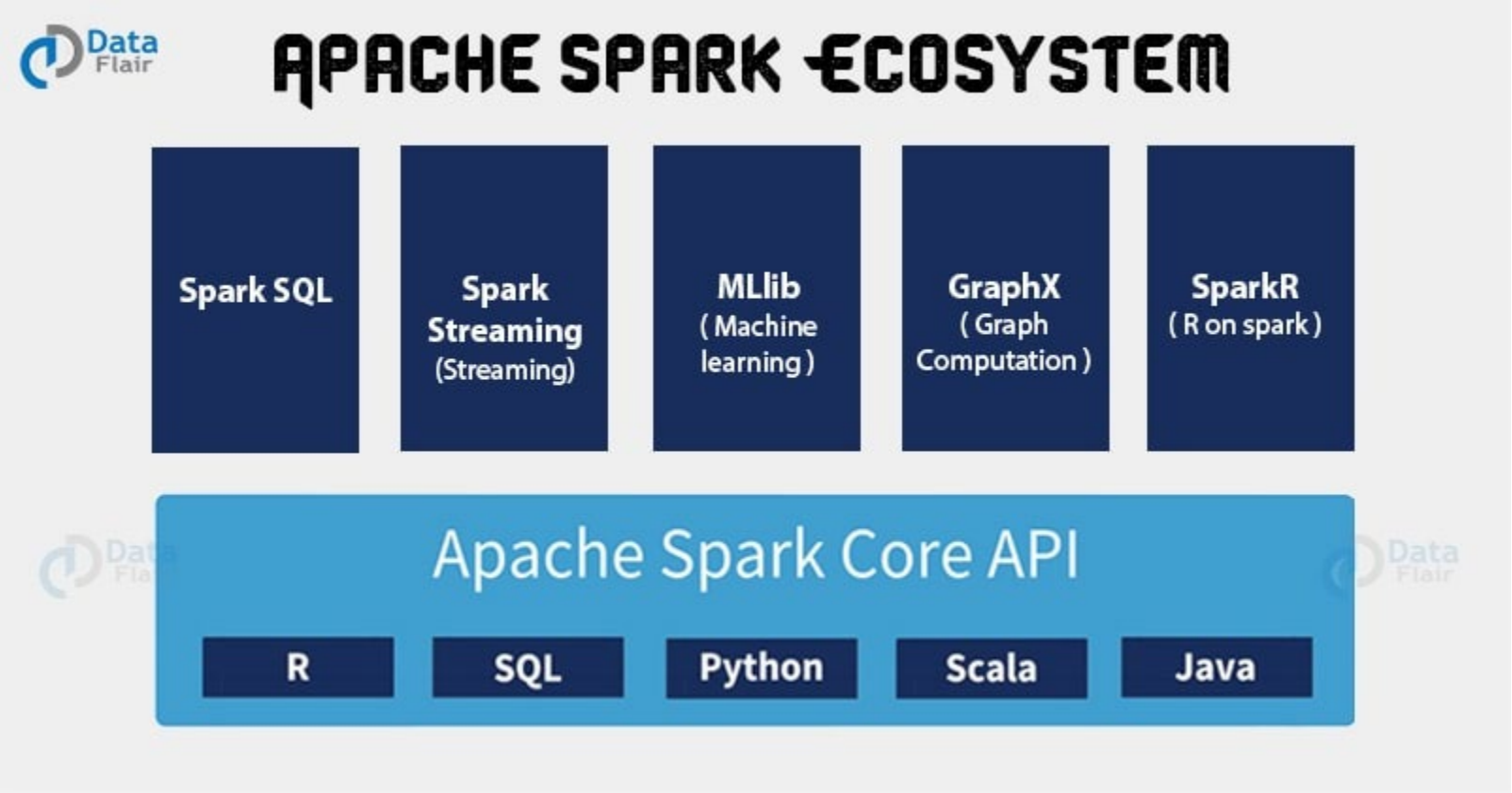
The Spark Computing Framework

- Provides 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"

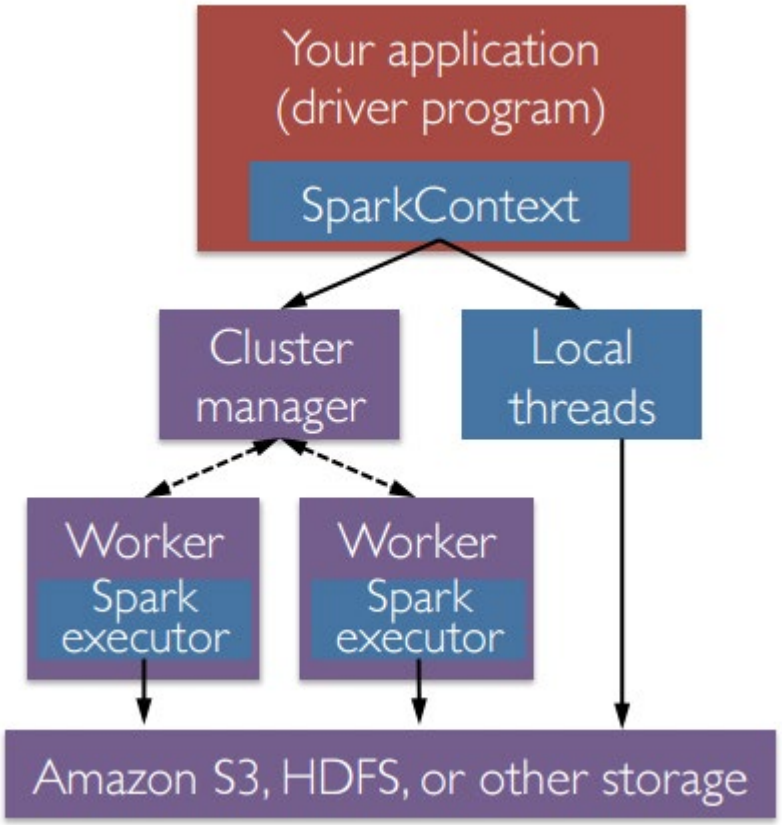
- I don't care where it runs (you schedule that)
- In fact, feel free to run it twice on different nodes (e.g., when it fails)

Apache Spark Ecosystem



***Source:** https://data-flair.training/blogs/apache-spark-ecosystem-components/

Spark Components



- A Spark program first creates a **SparkContext / SparkSession** object (driver)
  - Tells Spark how and where to access a cluster
  - Connect to several types of cluster managers (e.g., YARN or its own manager)
- Cluster manager:
  - Allocate resources across applications
- Spark executor (worker):
  - Run computations
  - Access data storage

Spark and SQL Contexts

- A Spark program is two programs:
  - A driver program and a worker program
  - Worker programs run on cluster nodes or in local threads
- A Spark program first creates a **SparkContext** object
  - tells Spark how and where to access a cluster
  - pySpark shell automatically create **SparkContext**
  - iPython (jupyter notebook) and programs must create a new **SparkContext**
  - 2.0.0+: **SparkSession** as the entry point (RDD→DataFrame)
- The program next creates a sqlContext object
- Use sqlContext to create **DataFrames**

Spark Essentials: Master

- The master parameter for a SparkContext/SparkSesion determines which type and size of cluster to use

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

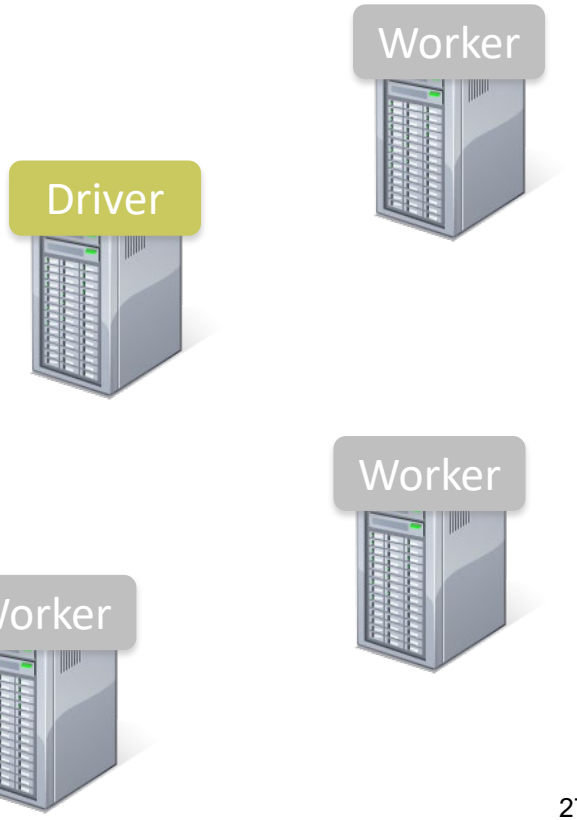
***Slides credit**: **Prof. A.D. Joseph, UC Berkeley**

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

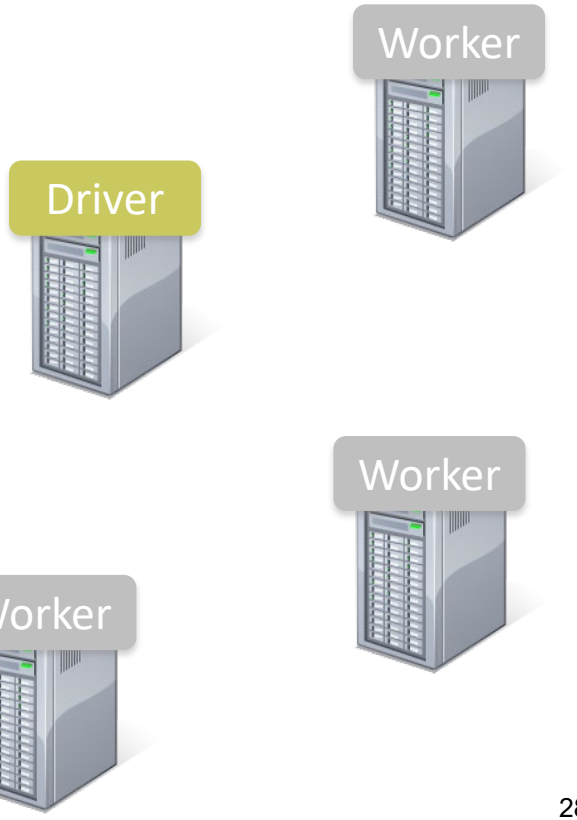
Load error messages from a log into memory, then interactively search for various patterns



Spark Example: Log Mining

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

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

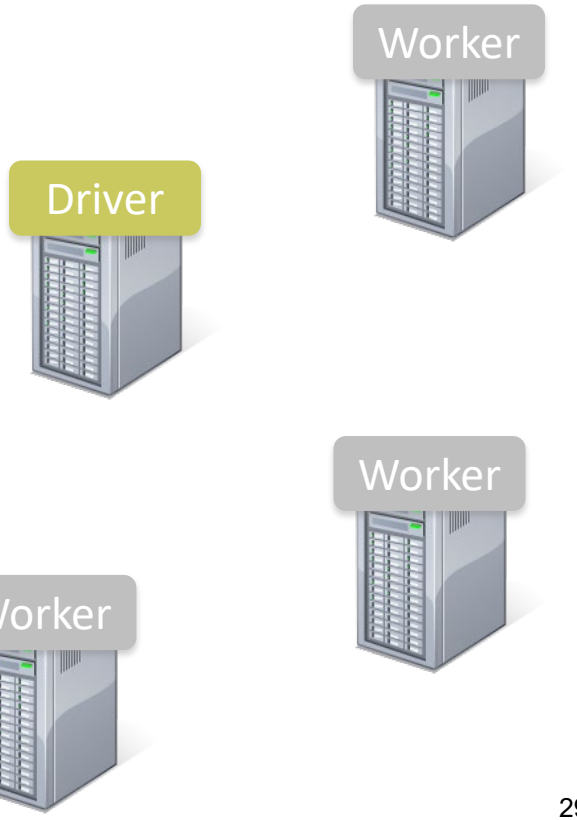


Spark Example: Log Mining

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

Base RDD

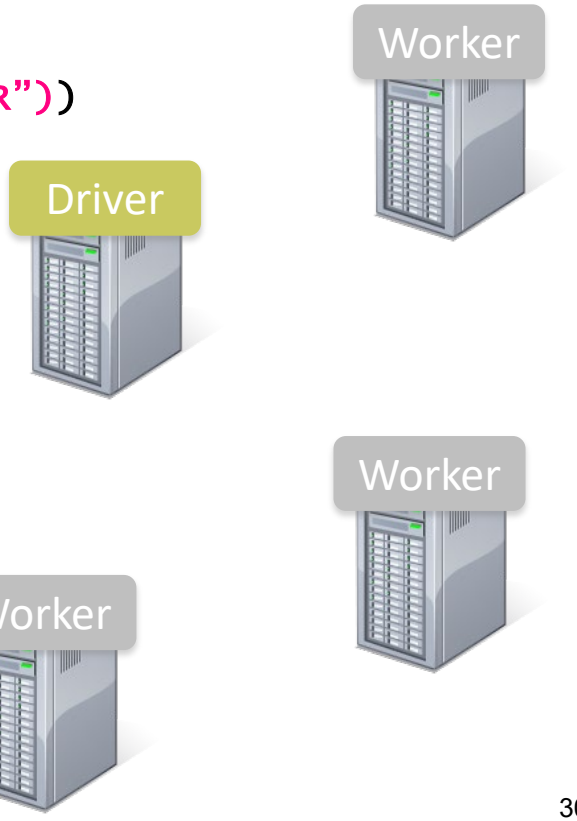
```
lines = spark.textFile("hdfs://...")
```



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

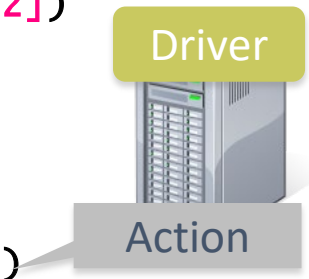
Load error messages from a log into memory, then interactively search for various patterns

```
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

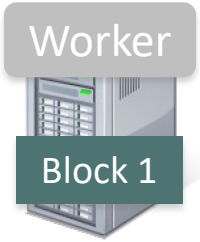
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


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

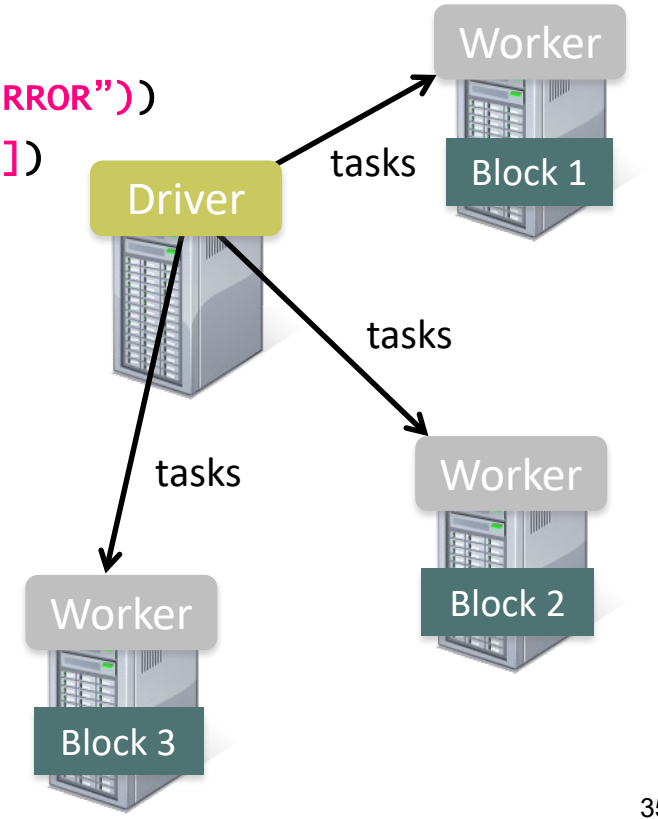
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


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

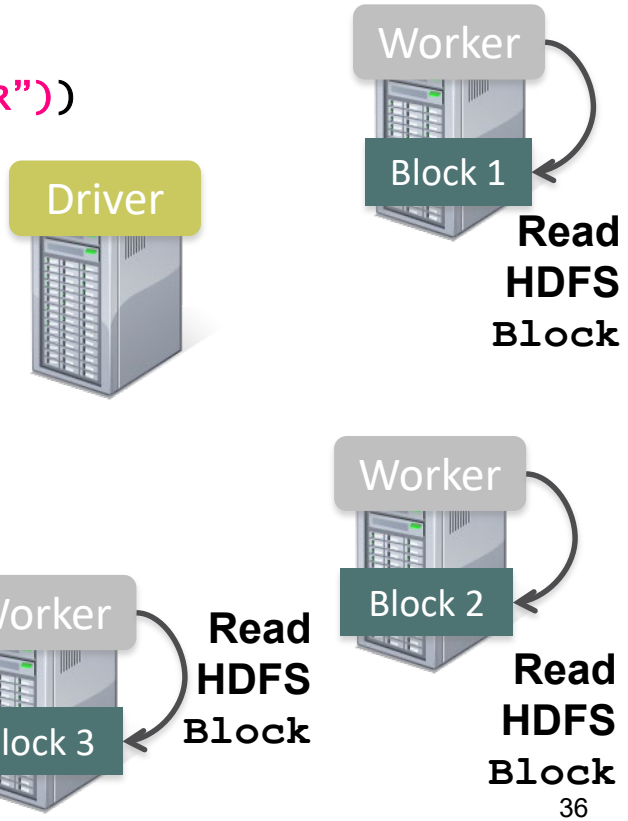
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


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

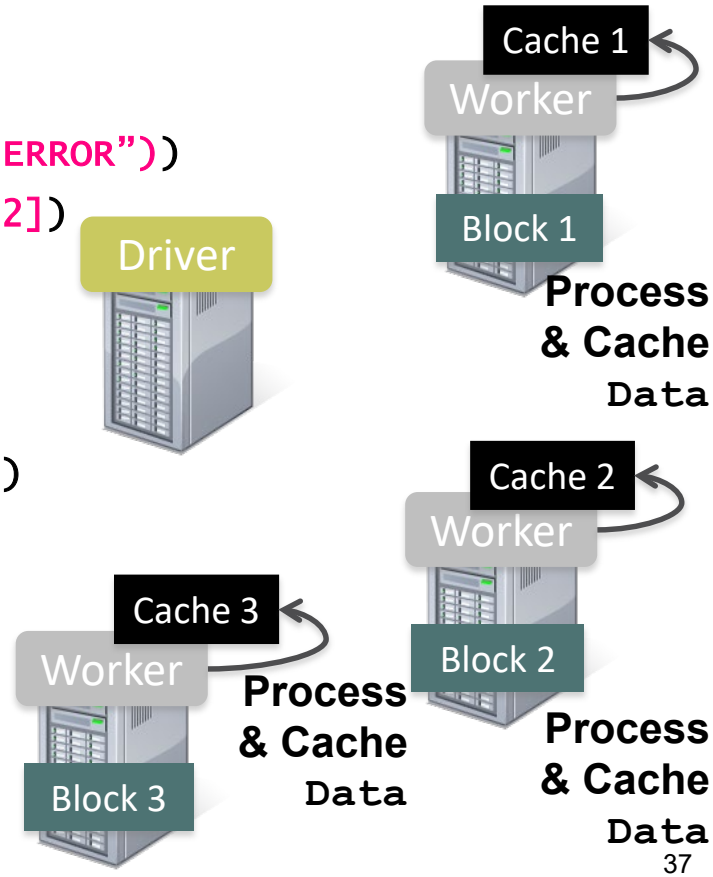
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


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

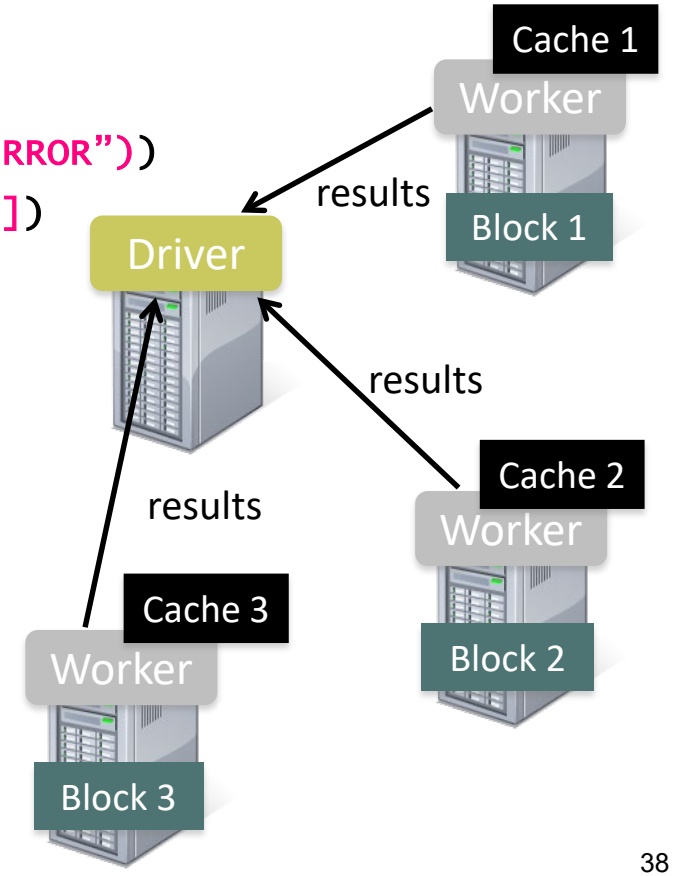
```
lines = spark.textFile("hdfs://...")
errors = lines.filter(lambda s: s.startswith("ERROR"))
messages = errors.map(lambda s: s.split("\t")[2])
messages.cache()


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()
```

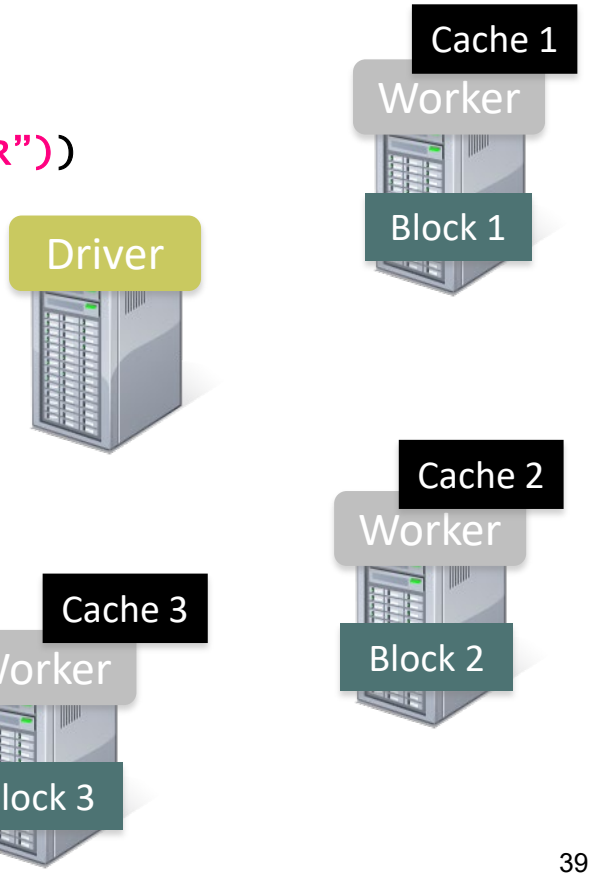


Spark Example: Log Mining

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()
```

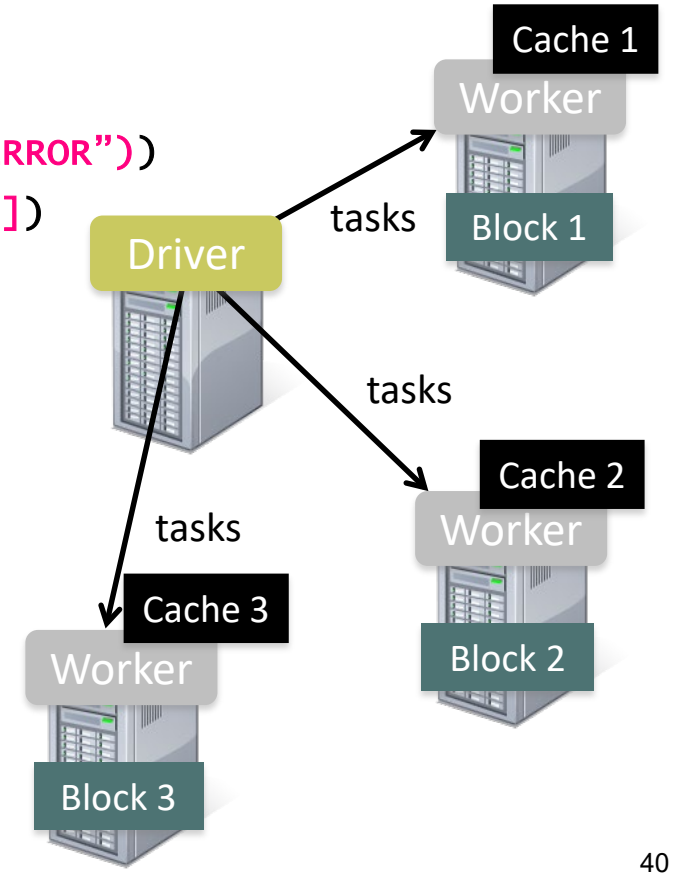


Spark Example: Log Mining

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()
```

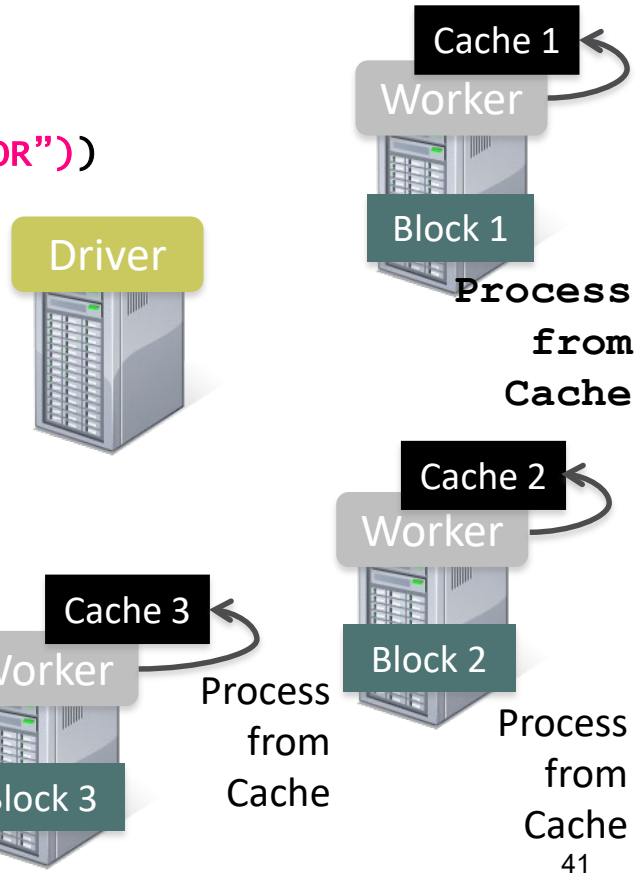
```
messages.filter(lambda s: "mysql" in s).count()
messages.filter(lambda s: "php" in s).count()
```



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()
```

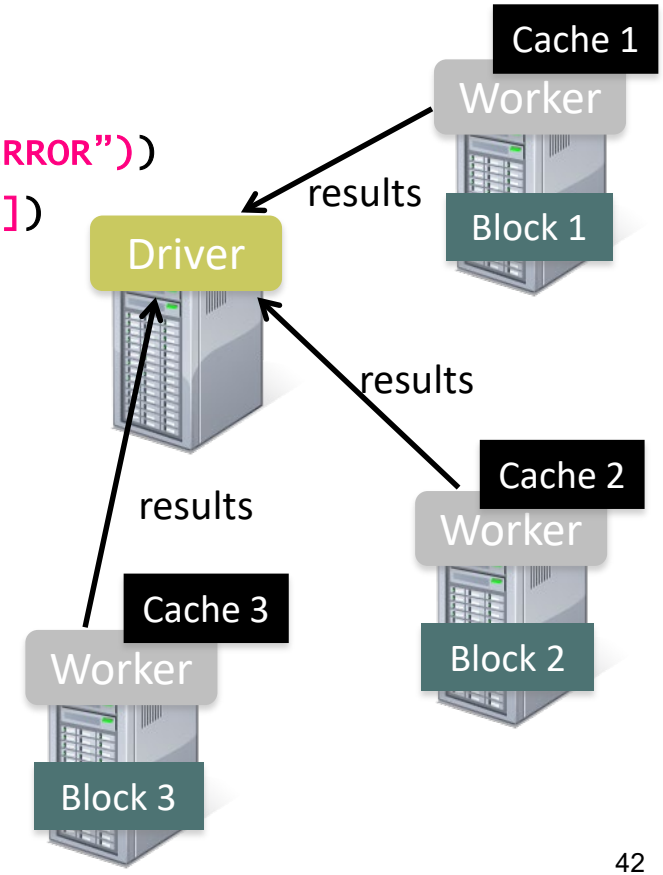


Spark Example: Log Mining

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()
```

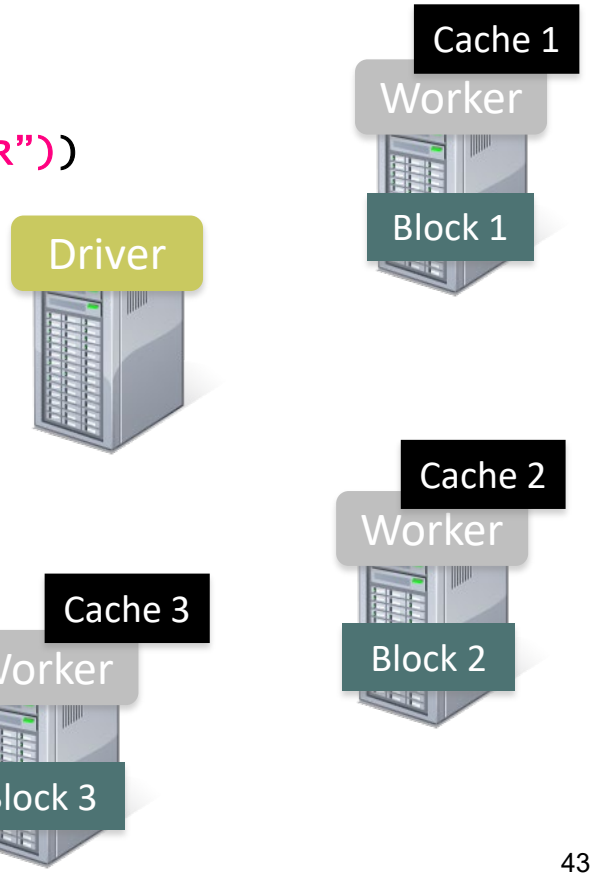


Spark Example: Log Mining

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**

***Slides credit**: **Prof. A.D. Joseph, UC Berkeley**

Spark Program Lifecycle

- Create DataFrames from external data or **createDataFrame** from a collection in 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

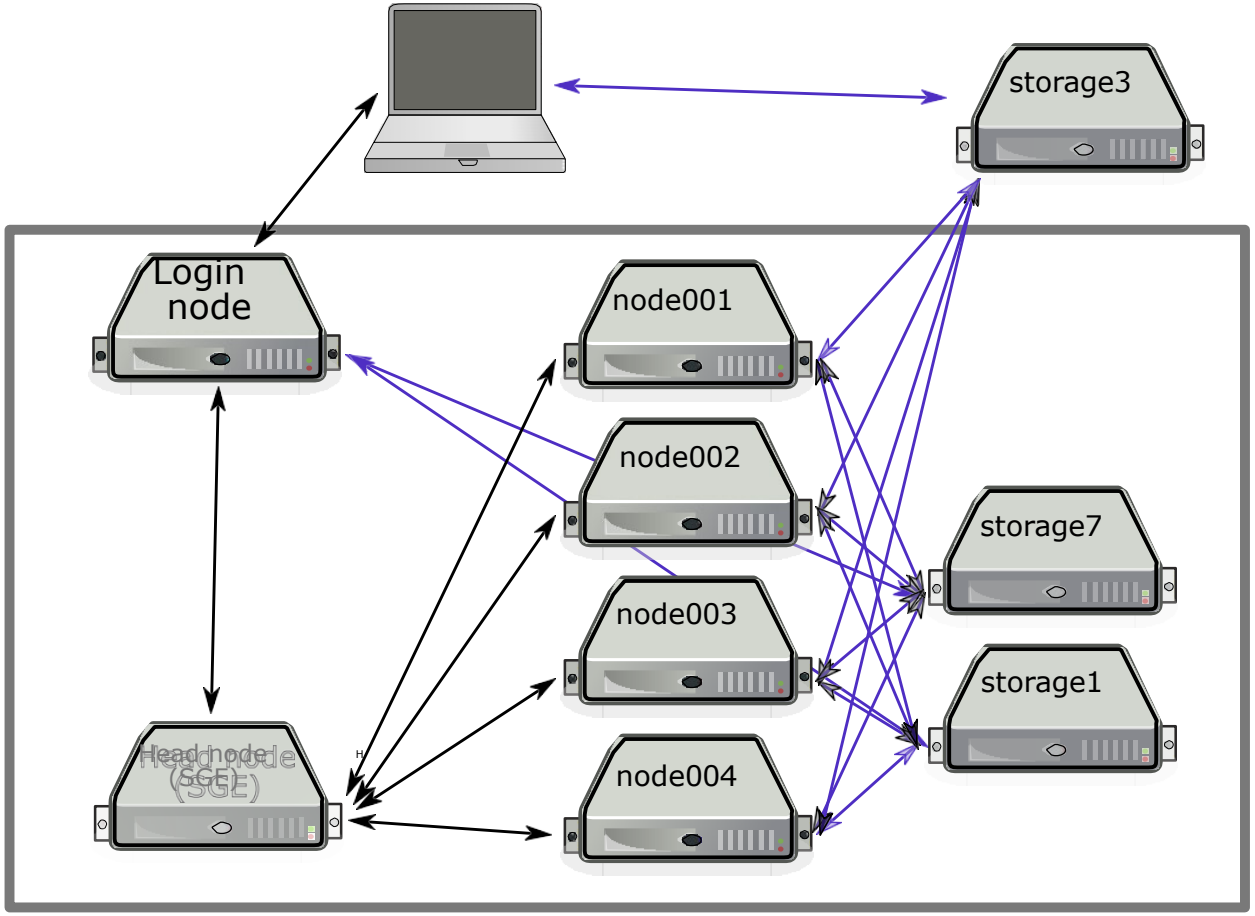
- See lab notebook
- Need: Java; Python (conda)

  conda install -c conda-forge pyspark=2.3.2
- Install on Windows: http://deelesh.github.io/pyspark-windows.html
  - Install Java JRE, Python
  - Install PySpark: pip install pyspark=2.3.2
  - Jupyter: https://changhsinlee.com/install-pyspark-windows-jupyter/
- Install on Linux/Mac
  - See tutorial PDF

HPC@Sheffield (**Click** pls)

- ShARC: **Sh**effield **A**dvanced **R**esearch **C**omputer
  https://www.sheffield.ac.uk/cics/research/hpc/sharc
- Docs:
  http://docs.hpc.shef.ac.uk/en/latest/sharc/index.html
- SSH access: MobaXTerm in Windows/terminal Linux/MAC OS
- Software:
  http://docs.hpc.shef.ac.uk/en/latest/hpc/modules.html
- You need to be on **campus network** to access ShARC
- **Help: hpc@sheffield.ac.uk**; Host: sharc.sheffield.ac.uk

High-Performance Computing Cluster Structure



Storage

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

Interactive Session

```
[me@mylaptop ~]$ ssh te1st@sharc.sheffield.ac.uk
...
[te1st@sharc-login1 ~]$ qrshx -P rse \
                              -pe smp 2 \
                              -l rmem=16G \
                              -m bea \
                              -M myemail@sheffield.ac.uk \
                              -j y
[te1st@sharc-node121 ~]$ ./my_simulation_program --num-cores=2
...
```

Batch Session – Shell Script xx.sh

```
#!/bin/bash
#$ -l h_rt=2:00:00  #time needed
#$ -pe smp 2 #number of cores
#$ -l rmem=4G #number of memery
#$ -o COM6012_Lab1.output #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
```

Batch Session: Submit & Wait

- qsub my-job-script.sh (can run at login node)

```
[me@mylaptop ~]$ ssh telst@sharc.sheffield.ac.uk
[telst@sharc-login1 ~]$ qsub my-job-script.sh
```

- Then?
  - Close the terminal and leave
  - Wait for an email notification (success/abort)
  - Check status (running/queueing): **qstat**
  - Cancel/amend job: **qdel**

https://www.sheffield.ac.uk/cics/research/hpc/sharc/batch

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**

  https://spark.apache.org/docs/2.3.2/ (we'll use **2.3.2**)
- **PySpark tutorial** (keep updating) https://runawayhorse001.github.io/LearningApacheSpark/pyspark.pdf
- Watch: https://www.youtube.com/user/TheApacheSpark/
- Code: https://github.com/apache/spark/
- Book: Karau et al., "Learning Spark: Lightning-Fast Big Data Analysis", O'Reilly Media, **2015 (old)**
- Google: emerging sources (please let me know if you find better ones)

Acknowledgements

- Some slides are extracted from the “Introduction to Apache Spark” course by Prof. Anthony D. Joseph, University of California, Berkeley

  https://www.edx.org/course/introduction-apache-spark-uc-berkeleyx-cs105x

- Many other sources that I have consulted but somehow lost track of the origins.
- Open source software and open knowledge bases benefit us all.