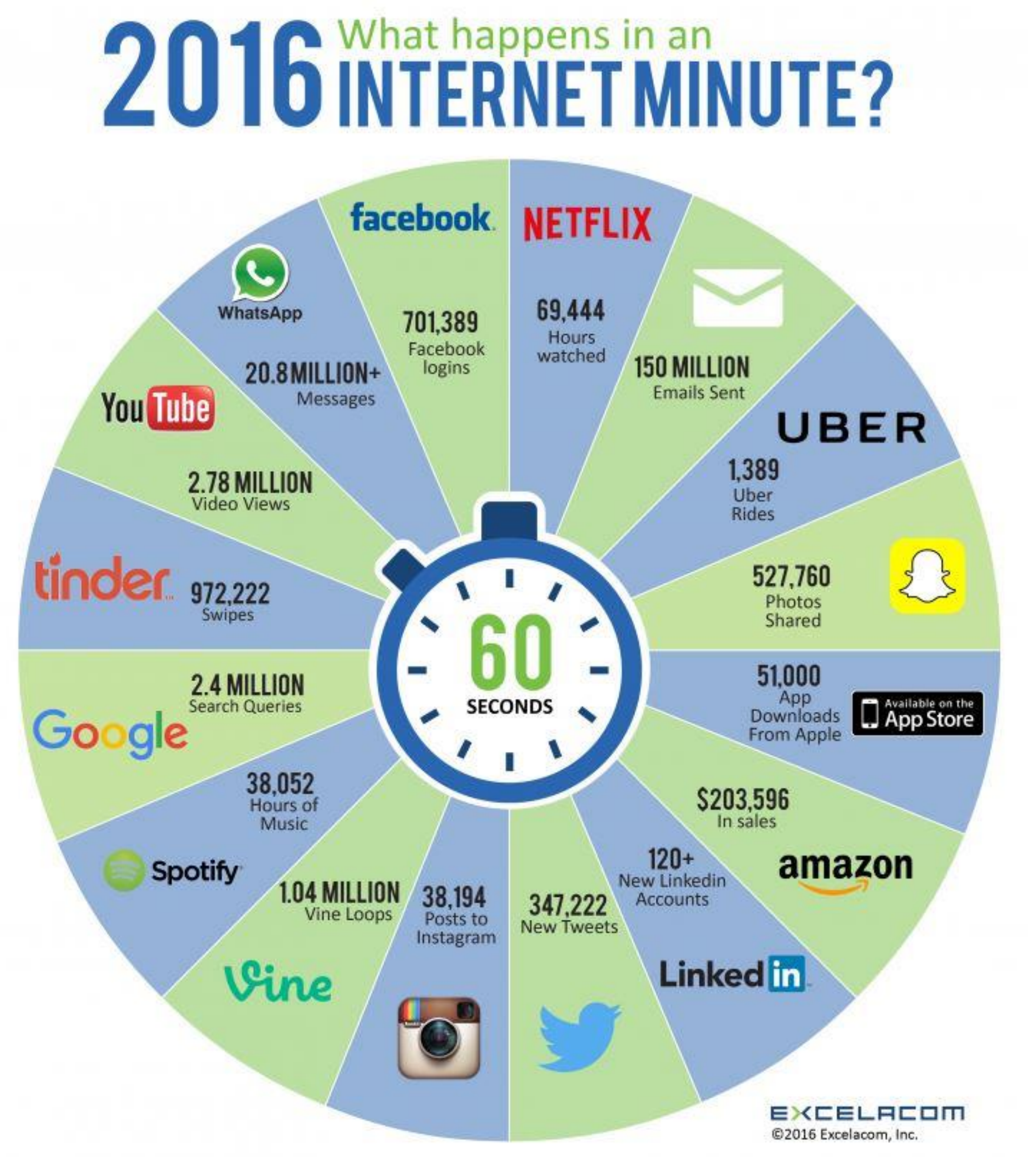
Big Data with Apache Spark

Introduction

“BIG DATA” IS DATA WHOSE SCALE, DIVERSITY, AND COMPLEXITY REQUIRE NEW ARCHITECTURE, TECHNIQUES, ALGORITHMS, AND ANALYTICS TO MANAGE IT AND EXTRACT VALUE AND HIDDEN KNOWLEDGE FROM IT…



The Scale of Big Data

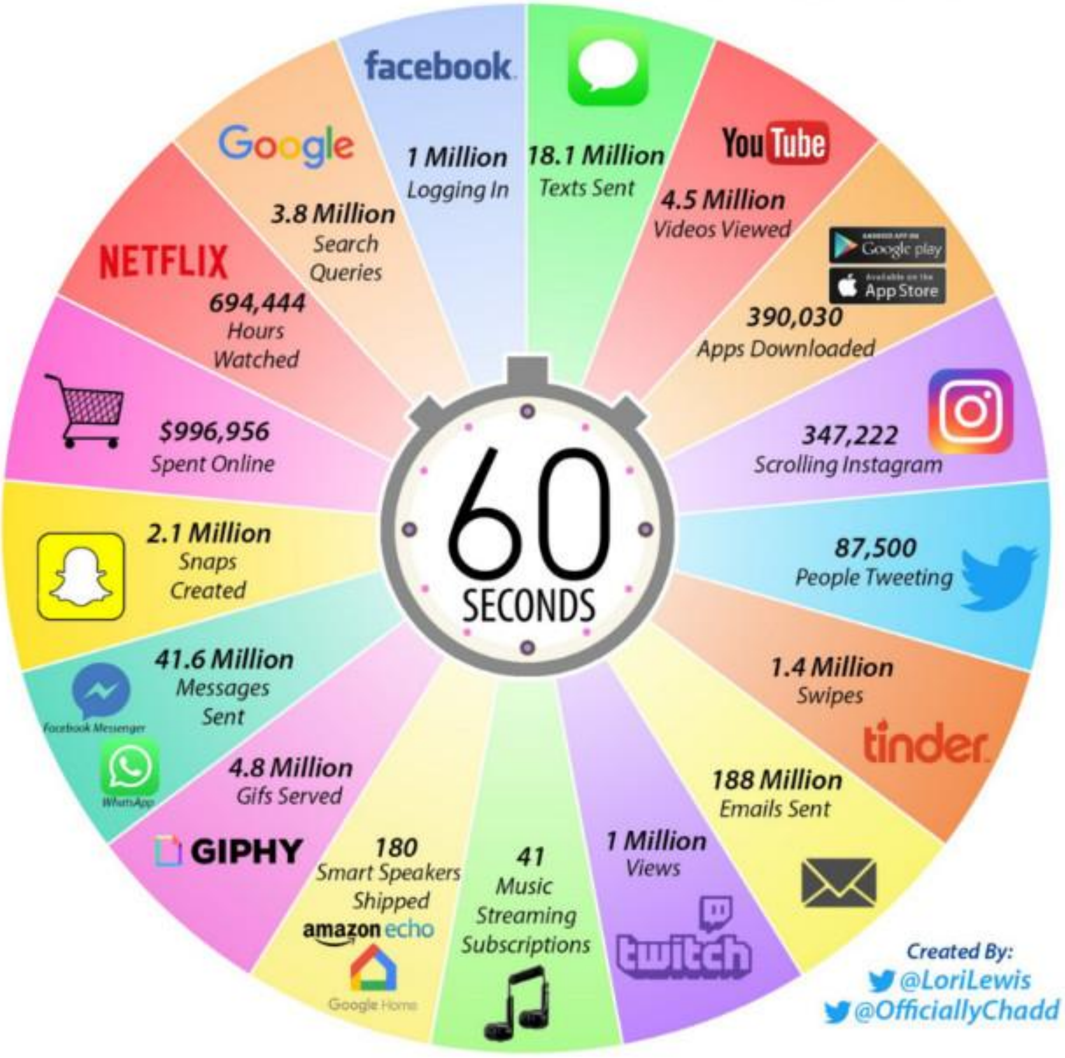
90% | Of today's data has been created in the last two years

Every day we create 2.5 quintillion bytes of data or enough to fill 10 million Blu-ray discs

Most companies in the US have over 100 terabytes (100,000 gigabytes) of data stored

40 zettabytes (40 trillion gigabytes) of data will be created by 2020, an increase of 300 times from 2005, and the equivalent of 5,200 gigabytes of data for every man, woman and child on Earth

2019 This Is What Happens In An Internet Minute



Created By:
@LoriLewis
@OfficiallyChadd

Data Lake

- Repository for analyzing large quantities of disparate sources of data in its native or raw format
- Reduce up-front effort by ingesting data in any format without requiring a schema initially
- Make acquiring new data easy, so it can be available for data science & analysis quickly
- Store large volume of multi-structured data in its native format
- https://www.mongodb.com/databases/data-lake-vs-data-warehouse-vs-database

Data Warehousing

- Data is integrated from multiple systems.
- For example provide a full view of a customer:
  - Sales activity
  - Delinquent invoices
  - Support/help requests
- Focus is on reading the information and creating analysis
- Data modelling and ETL process consume most of the time and effort in setting up a data warehouse

Data Lake

- Agility
- Flexibility
- Rapid Delivery
- Easy exploration
- Data acquisition is easier
- Data retrieval requires more effort

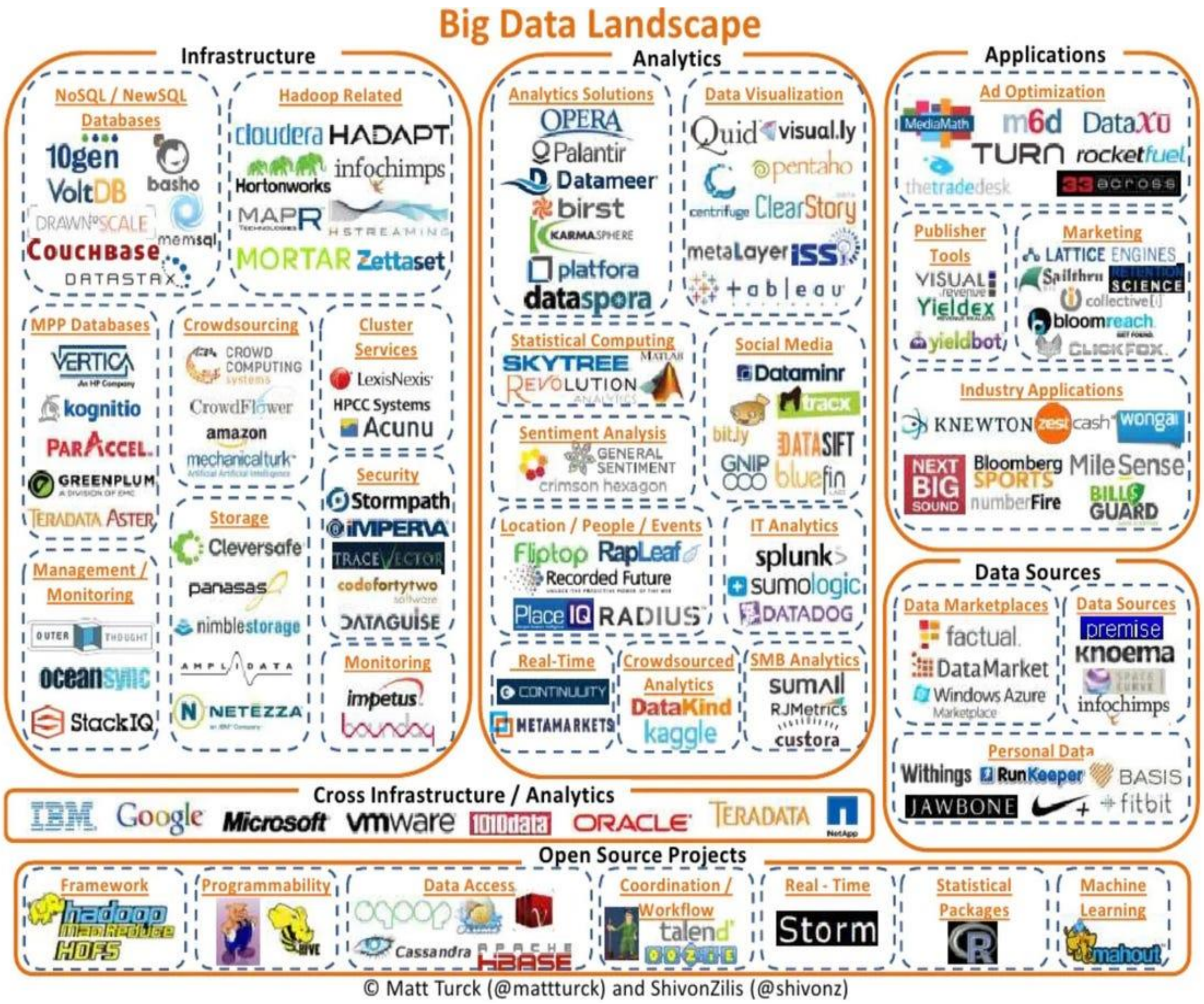
Data Warehouse

- Governance
- Reliability
- Standardization
- Security
- Data acquisition requires more effort
- Data retrieval is easier

ETL – Extract, Transform, Load

- (**ETL**) is the general procedure of copying data from one or more sources into a destination system which represents the data differently from the source(s) or in a different context than the source(s)
- Traditionally, ETL has been used to move data between elements in a data pipeline
  - Online Transaction Processing Database => Data Lake => Data Warehouse
- With changing features of data storage systems we also now use "**ELT"**
- More recently, attempts at providing a "one-stop-shop" may reducing the need for ETL

Technology for Big Data 2022



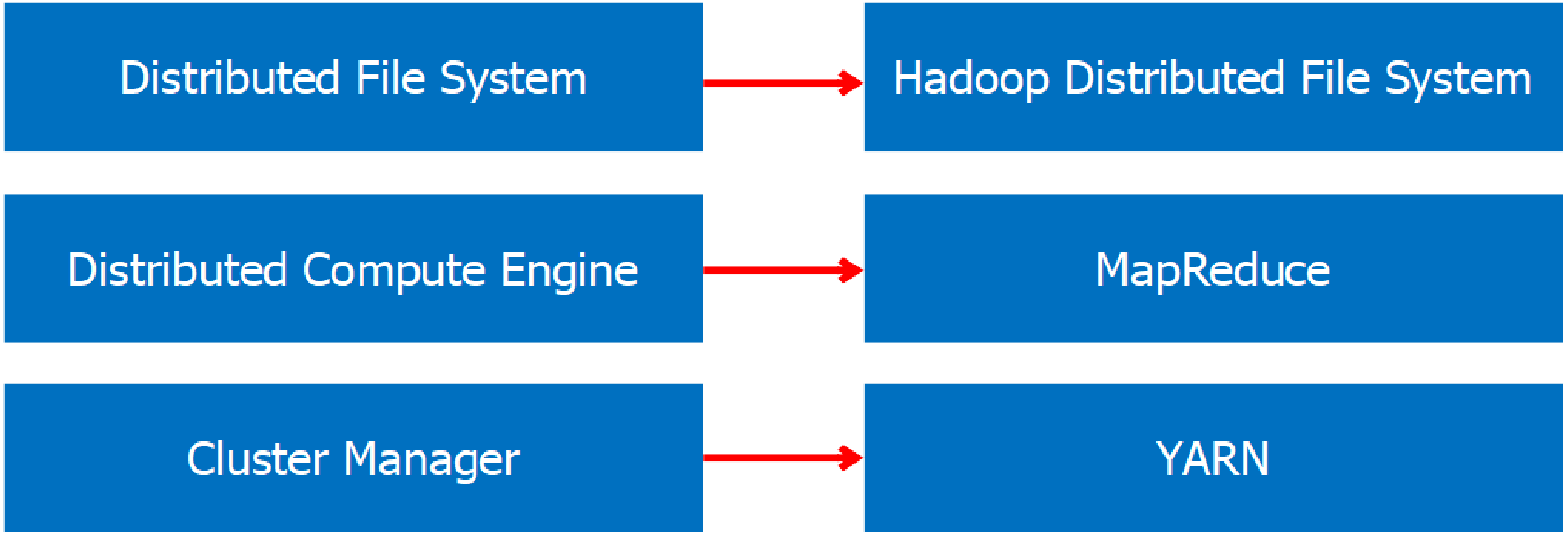
Introduction to Apache Spark

Some History

- Early 2000s - Google needed tools to process very large amounts of data
  - Google File System (GFS)
  - Map Reduce: https://research.google/pubs/pub62/
  - Google BigTable
- On publishing papers about these technologies, development of open-source implementations was taken up by Yahoo, Cloudera, Hortonworks and others
- These projects were eventually donated to the Apache Software Foundation
- This became "Hadoop"!

Hadoop Core Components

- Hadoop isn't really a single thing, it's an eco-system of related projects
  - At its heart are three key elements...



Distributed File System → Hadoop Distributed File System

Distributed Compute Engine → MapReduce

Cluster Manager → YARN

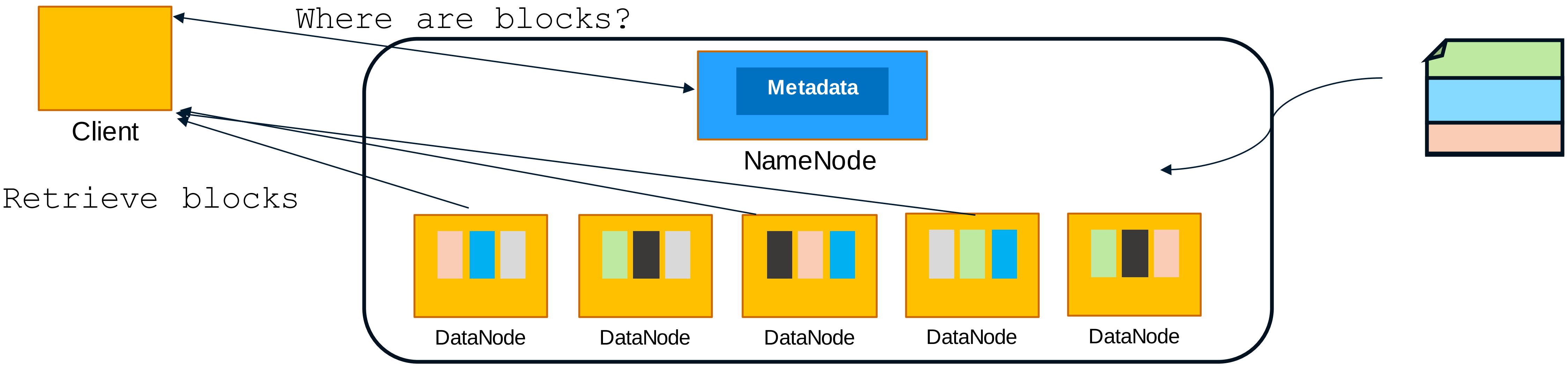
Hadoop conceptual components

Hadoop concrete implementations

Hadoop Distributed Storage

- HDFS is a scalable and fault-tolerant distributed file system
  - Stores a file across a cluster of commodity servers (e.g. 1000s)
  - Aim: to store and allow fast access to big files and large datasets

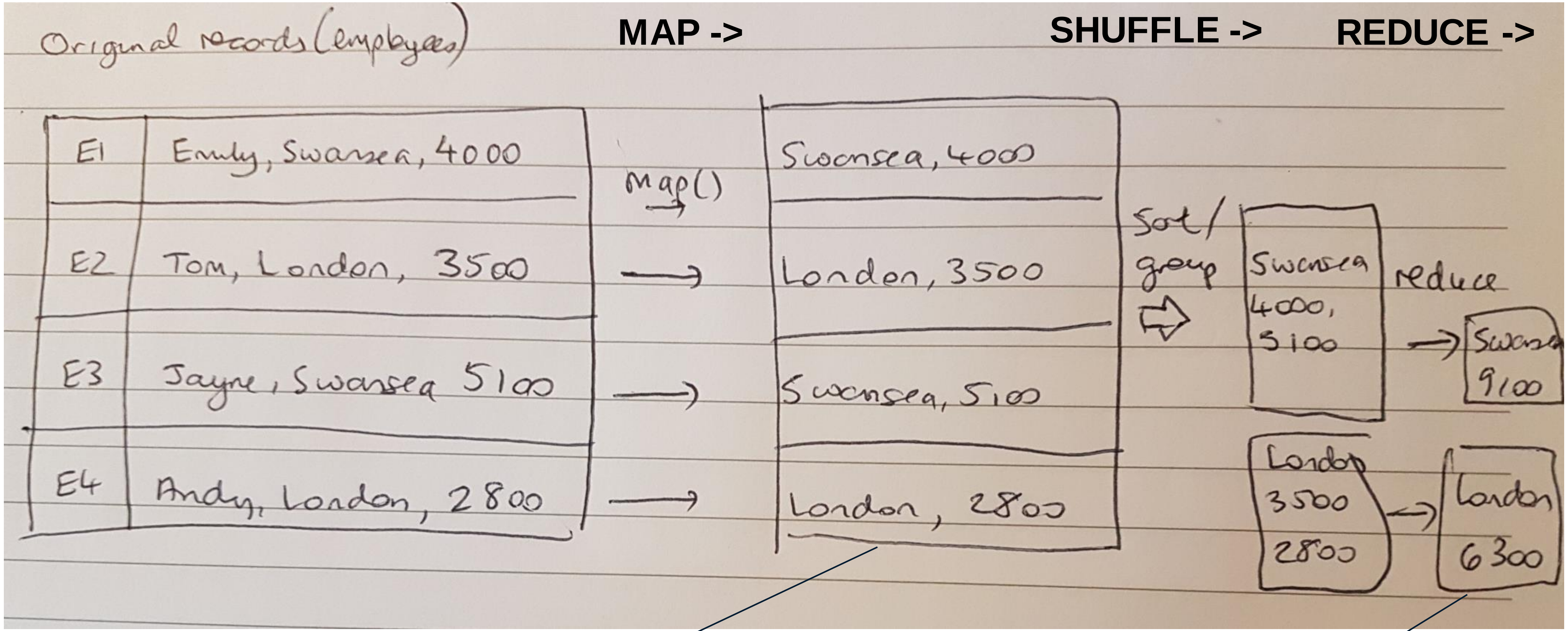
- HDFS spreads file blocks across "worker node" machines
  - Allows file read/write operations to be massively parallelized



How to Process or Analyse Distributed Data?

- Moving **CODE** from one computer to another is much faster and more efficient than moving **LARGE DATASETS**
    - E.g. imagine you have a cluster of 50 computers with 1TB of data on each computer - what are the options for processing this data?

Basic Operations – mapping & reducing



MAP -> **SHUFFLE ->** **REDUCE ->**

Original records (employees)

E1	Emily, Swansea, 4000
E2	Tom, London, 3500
E3	Jayne, Swansea 5100
E4	Andy, London, 2800

Swansea, 4000
London, 3500
Swansea, 5100
London, 2800

MAP: Produces an **altered** version of the original Dataset

REDUCE: Produces an **aggregated** version of the original Dataset

Why NOT Hadoop MR?

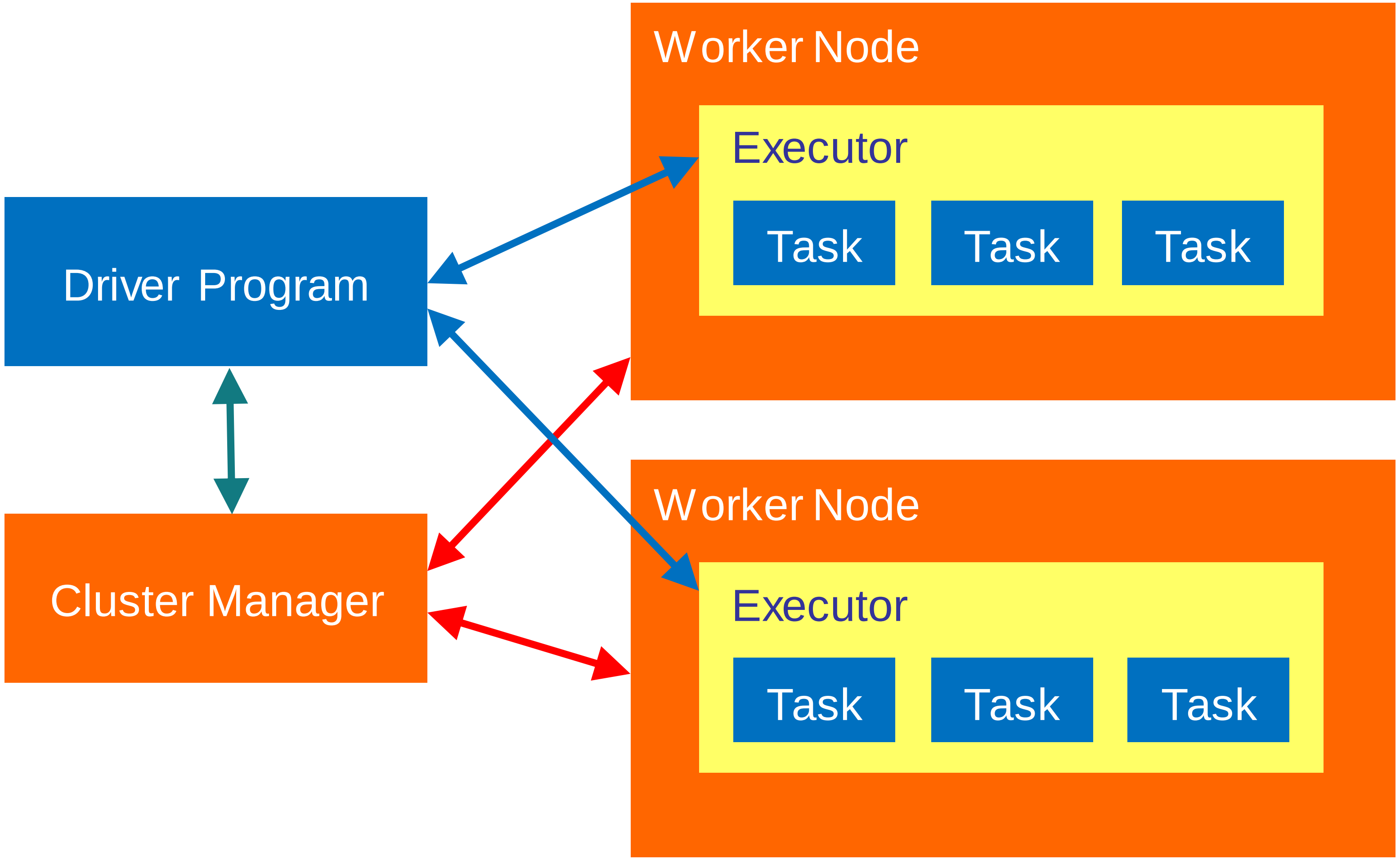
- The Hadoop processing framework "Map-Reduce" was a ground-breaking programming paradigm for parallel computing
- However, the Hadoop implementation had some drawbacks
  - Very verbose code – lots of boilerplate required
  - Complex for developers to write Jobs
  - Not very fault-tolerant
  - Slow
  - Heavily reliant on disk I/O

Why Spark?

- Spark grew out of the need to have a simpler, faster, more robust way to program with parallelism
- Research groups in UC Berkeley began working on this, with some guiding principles
    - Highly Fault Tolerant
    - 100% Parallel
    - In-memory Intermediate results
    - Easy API
    - Program in multiple languages – e.g. Java, Scala, Python, R

Spark Architecture

- Driver loads or requests loading of data onto worker nodes so DATA IS DISTRIBUTED
- Driver Program "gathers" operations to be done to data
- When some "result" is needed the driver will calculate the most efficient sequence of functions to be SENT TO THE DATA
- When necessary, data will be "reorganised" – known as a "SHUFFLE"

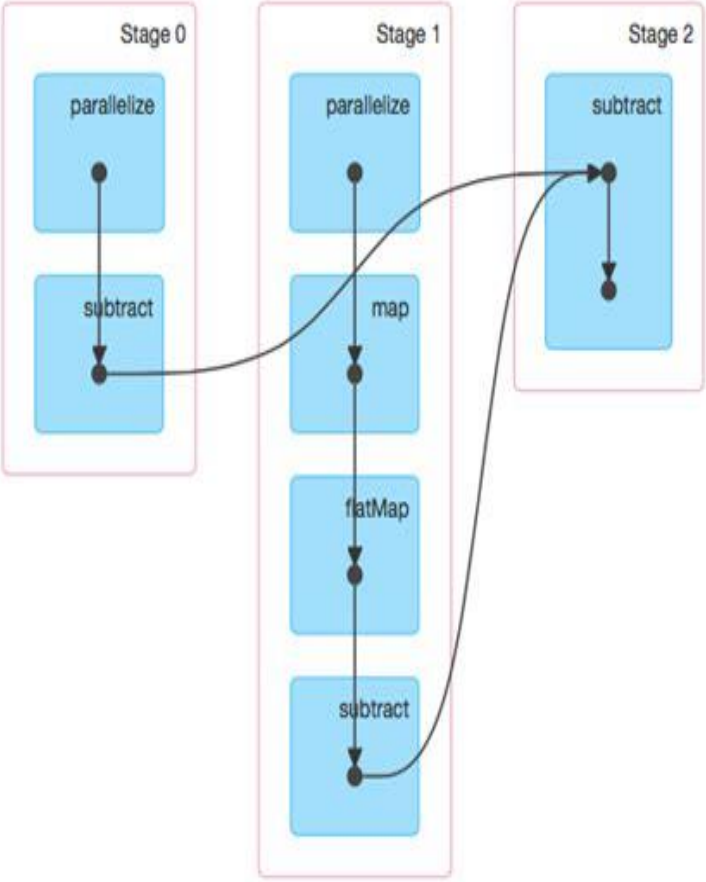


Spark RDD

- All data in spark is based an "RDD"
  - Resilient
  - Distributed
  - Dataset

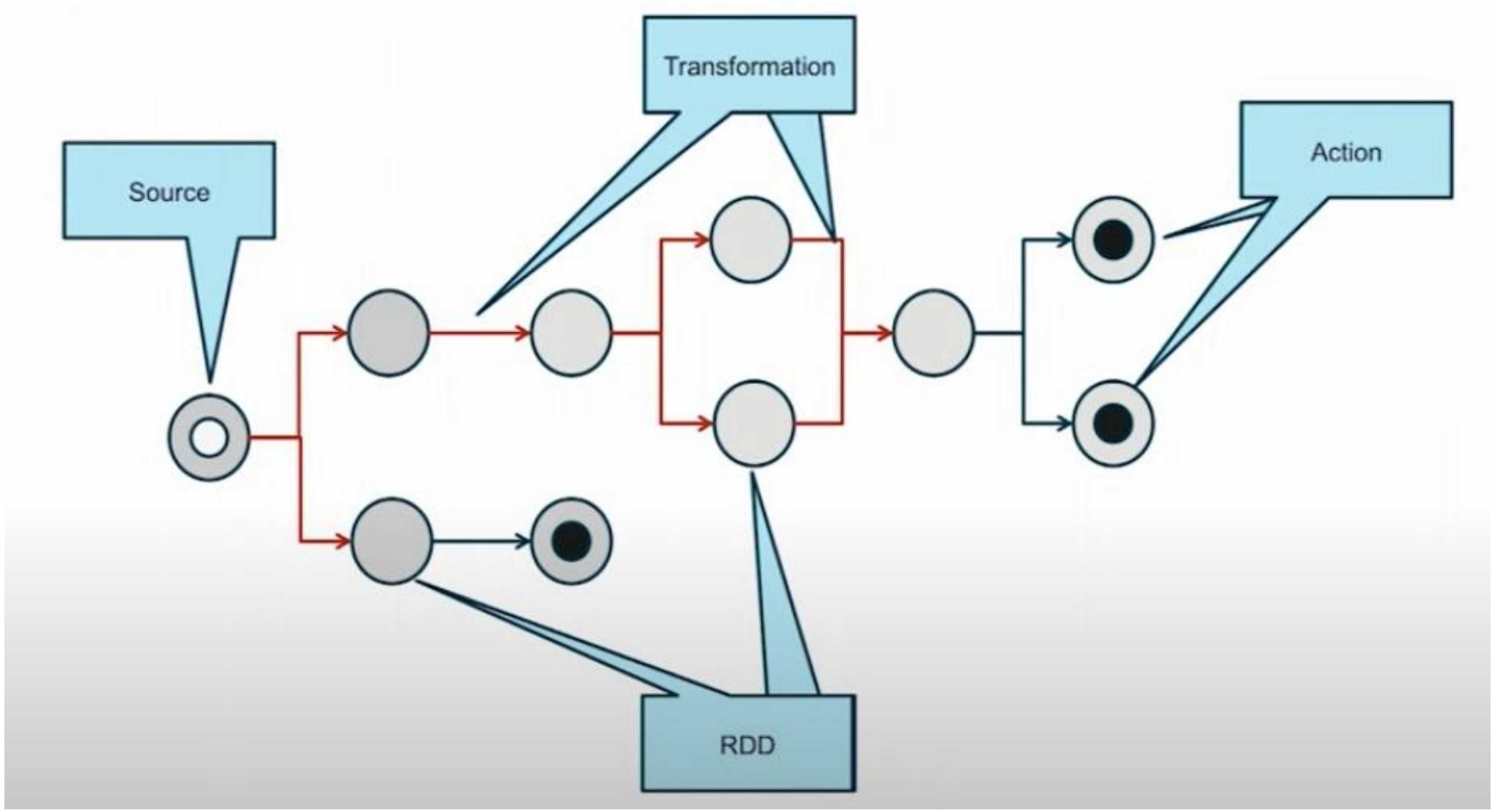
- Spark driver creates a **Directed Acyclic Graph** (DAG) for a job – the most efficient sequence of operations

- We write all of this through high-level APIs



Spark & DAG

- RDDs are **Resilient**
- The DAG contains the instructions to recreate any intermediate RDD



Spark Examples

- Let's take our first steps with Spark in Python

- References: https://github.com/fcallaly/spark-intro-examples

Questions?