SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Established in collaboration with MIT

Big data

PROF. D. HERREMANS

50.038 Computational data science

Big data?

- Incremental increase of data
- IBM estimates that 90% of all data was created in the last two years
- What is this data?
  - Logs from phone companies (e.g. which towers you connect to), websites
  - Medical data such as X-rays
  - Stock transactions
  - Store order data
  - …

  → We need to be able to store and process it

What is big data?

- First mention:

  “…provides an interesting challenge for computer systems: data sets are generally quite large, taxing the capacities of main memory, local disk, and even remote disk. We call this the problem of big data. When data sets do not fit in main memory (in core), or when they do not fit even on local disk, the most common solution is to acquire more resources.”

  In a 1997 paper by NASA scientists Michael Cox and David Ellsworth, when describing a problem they had with visualization

What is big data?

- From around 2008 the term was popularized:

  “Data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges.” – Oxford dictionary

  “The belief that the more data you have the more insights and answers will rise automatically from the pool of ones and zeros.”

  “A new attitude by businesses, non-profits, government agencies, and individuals that combining data from multiple sources could lead to better decisions.”

Source: https://www.forbes.com/sites/gilpress/2014/09/03/12-big-data-definitions-whats-yours/2/#4333b2095b3f

What is big data?

- ***Too big to be processed on a single machine***
- *Can be structured/unstructured/semi-structured*
- *Lately: refers to predictive analytics*

- Challenges: ?
  - Most data is worthless?
  - Data is created fast?
  - Data from different sources in different formats?

What is big data?

- ***Too big to be processed on a single machine***
- *Can be structured/unstructured/semi-structured*
- *Lately: refers to predictive analytics*

- Challenges: ?
  - Most data is worthless? ✗
  - Data is created fast? ✓
  - Data from different sources in different formats? ✓

Applications of big data

- Healthcare & research
- Education
- Telecom
- Media
- IoT
- Banks & Insurance
- Retail
- Manufacturing (automotive production)
- Energy & Utilities
- …

$\rightarrow$ These system all face challenges (3Vs)

The 3 V's of big data

- Challenges of big data is not just about the size:
  - **Volume:** size of the data
  - **Variety:** data comes from many different sources, in many different formats
  - **Velocity:** speed at which it is being generated and needs to be able to be processed

  First defined by Laney, 2001

  These days: fourth V:
  - **veracity**: trustworthiness of the data

1. Volume

- Increasing data amount:
  - Handling Petabytes instead of Megabytes
  - The estimated amount of "**data** in the digital universe” in 2010: 1.2 zettabytes (1.3 trillion gigabytes) – is equal to 75m fully-loaded 16GB iPads
- What produces data?
  - Internet of Things: mobile devices, cameras, microphones, RFID readers,…
- The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s (Hilbert, 2011)

1. Volume

- Cost:
  - 1980: 1 GB → \$100,000 \$
  - 2017: < 0.008 \$

  Note: ***reliable*** storage (e.g. Storage Area Network: SAN) is more expensive

- Most data is useful, so we want to keep as much as possible
- Need a cheap way to ***store, read and process*** data
  - Storing on SAN is easy
  - Streaming this data over network: more difficult

2. Variety

- What do we want to store?
  Transactions, logs, business, user, sensor, medical, social,…
- SQL: extremely structured/predefined
- BUT: Estimated 90% of data is ‘unstructured’ or semi-structured, e.g.
  - Bank may have scans of receipts that look differently
- We want to store the original data:
  - E.g. transcribing a helpdesk conversation to text may be more efficient, but audio will contain tone of voice → different interpretations

    => This is where **Hadoop** comes in (versus a structured database system):
  - Allows you to store raw data + manipulate and reformat later
  - From SQL blob to Hadoop

Types of data

- **Structured:**
  - Represented in strict format, e.g. SQL DB
  - Necessary for operations, e.g. sales transaction
- **Unstructured:**
  - No predefined data model, not organized in pre-defined manner
  - Often interesting for data mining
  - E.g. text, numbers, files,… can be all mixed. -> Hadoop
- **Semi-structured:**
  - May have a certain structure, but not all data has necessarily identical structure
  - No predefined schema
  - Schema information can be mixed with data values, since each data object may have attributes that are not known in advance, e.g. xml, json -> MongoDB

Example of variety

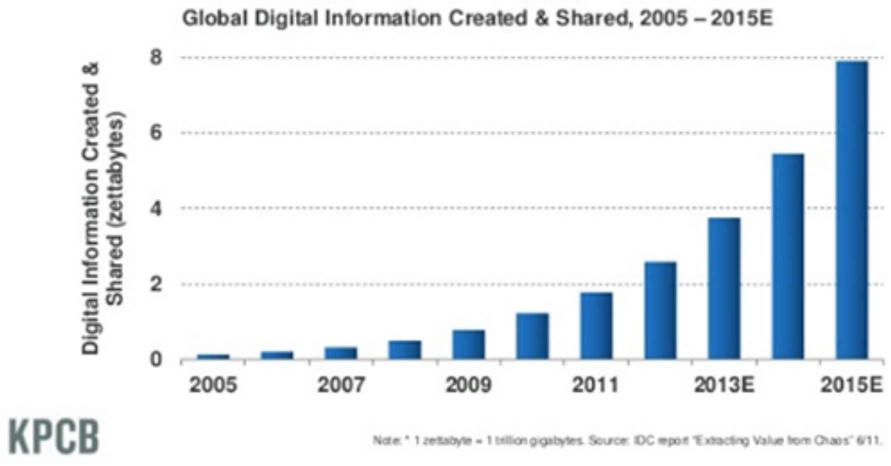
- Delivery company with trucks
- To pick up a new load from a depot, do we just send the closest truck?
  - Current GPS
  - Current map: small roads slower?
  - Traffic data
  - Current load
  - Fuel efficiency
  - …

3. Velocity

- In 2015: 2.5 Quintillion bytes of data created

- We need to be able to store data as it arrives

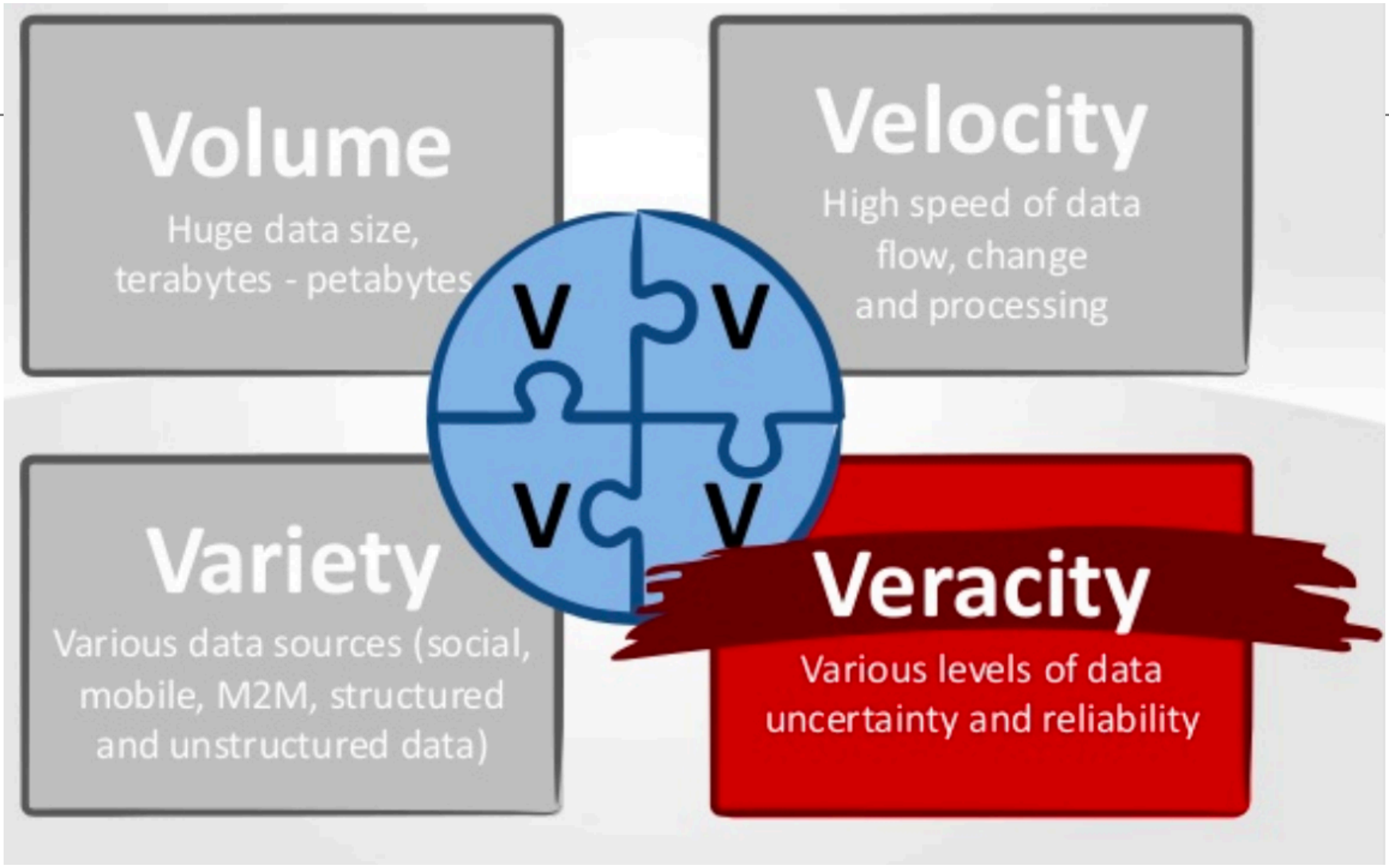
- Why store so much data? E.g. product recommendations



- As of 2012, every day 2.5 exabytes (2.5×10^{18}) of data are generated (IBM, 2012)

4. Veracity

- Various data uncertainty and reliability
- Imprecision of data (especially unstructured data), e.g. text message can have double meaning
- Different level of data quality of structured data (certain and precise) versus unstructured data (fuzzy interpreting of images, free text, speech…)
- Technical data quality issues (various formats, source availability, updates)
- → 5th V: value…


Volume
Huge data size, terabytes - petabytes
Velocity
High speed of data flow, change and processing
V
V
V
V
Variety
Various data sources (social, mobile, M2M, structured and unstructured data)
Veracity
Various levels of data uncertainty and reliability

Source: InfoDiagram

CAP theorem

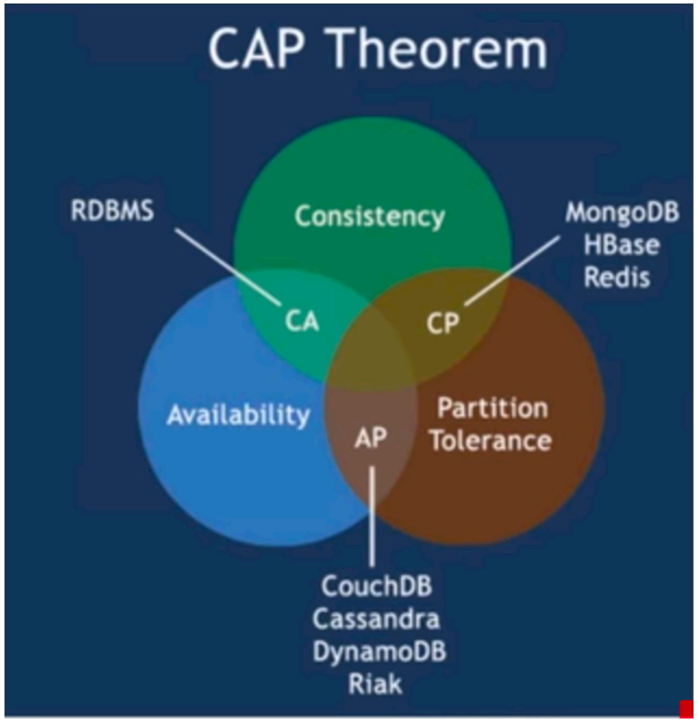
- Eric Brewer:

  "It's impossible for a distributed data store to simultaneously provide more than two out of the following three guarantees":

1. **C**onsistency:
   every read received the most recent write or an error
2. **A**vailability:
   every request receives a non-error response
   (without guarantee that it's the most recent write)
3. **P**artition tolerance:
   The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes. Partition = communications break

CAP theorem

- CA: Single cluster, all nodes are always in contact. When a partition between nodes should be created the data is out of sync until partition is resolved.
- CP: Some data may not be accessible, but the rest is still consistent/accurate
- AP: System is still available under partitioning, but some of the data returned may be inaccurate. Will resync data once the partition is resolved



PACELC theorem

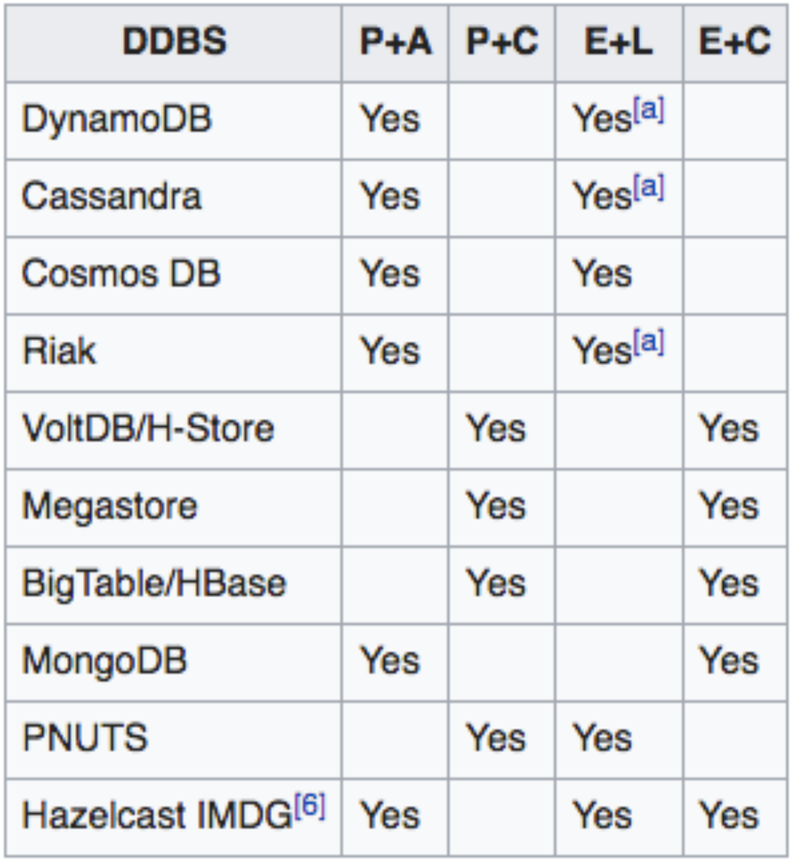
- Daniel Abadi:

  *“if there is a **p**artition (P), how does the system trade off **a**vailability and **c**onsistency (A and C); **e**lse (E), when the system is running normally in the absence of partitions, how does the system trade off **l**atency (L) and **c**onsistency (C)?”.*

  -> Expansion of the earlier defined CAP theorem

  A high availability requirement implies that the system must replicate data. As soon as a distributed system replicates data, a tradeoff between consistency and latency arises.

PACELC



DDBS	P+A	P+C	E+L	E+C
DynamoDB	Yes		Yes[a]	
Cassandra	Yes		Yes[a]	
Cosmos DB	Yes		Yes	
Riak	Yes		Yes[a]	
VoltDB/H-Store		Yes		Yes
Megastore		Yes		Yes
BigTable/HBase		Yes		Yes
MongoDB	Yes			Yes
PNUTS		Yes	Yes	
Hazelcast IMDG[6]	Yes		Yes	Yes

hadoop

& MAP REDUCE

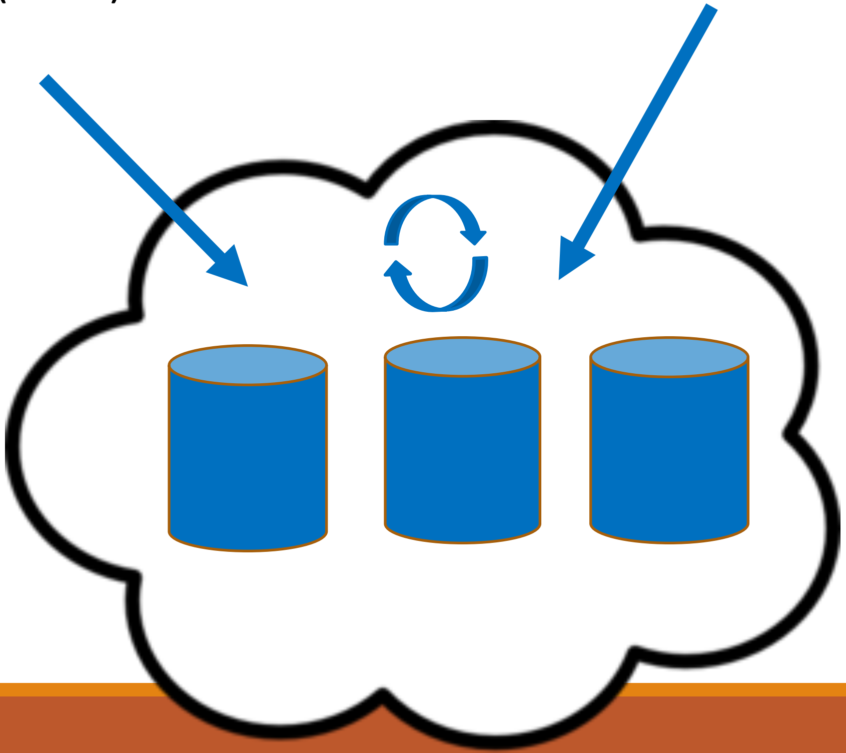
Hadoop

- *”An open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model.”*
- 2 parts:
  - Storage: Hadoop Distributed File System (HDFS)
  - Processing: MapReduce
    → manipulates data where it is stored (data locality principle)
- Inspired by Google File System
- Named after toy elephant of Doug Cutting’s son (co-founder together with Mike Cafarella)
- Coded in java

Core Hadoop

Store
In
Hadoop File
System (HDFS)

Process
with
MapReduce



Hadoop ecosystem

- Easier to use:
- Writing MapReduce code isn't very simple, many people know SQL.

Tools that help other applications integrate Hadoop:

- Hive:
  - Use statements like: "select * from…"
  - Looks like SQL, but Hive transforms this to Map reduce code
- Pig:
  - Analyze data using a simple scripting language
  - This is then transformed into Map reduce code.

Hadoop ecosystem

- Impala:
  - Query data with SQL directly (no MapReduce)
- Sqoop:
  - Takes data from SQL DB and puts it into HDFS as delimited files
- Hue:
  - Graphical interface to the cluster
- Mahout:
  - Machine Learning library
- Hbase:
  - Real time database built on top of HDFS

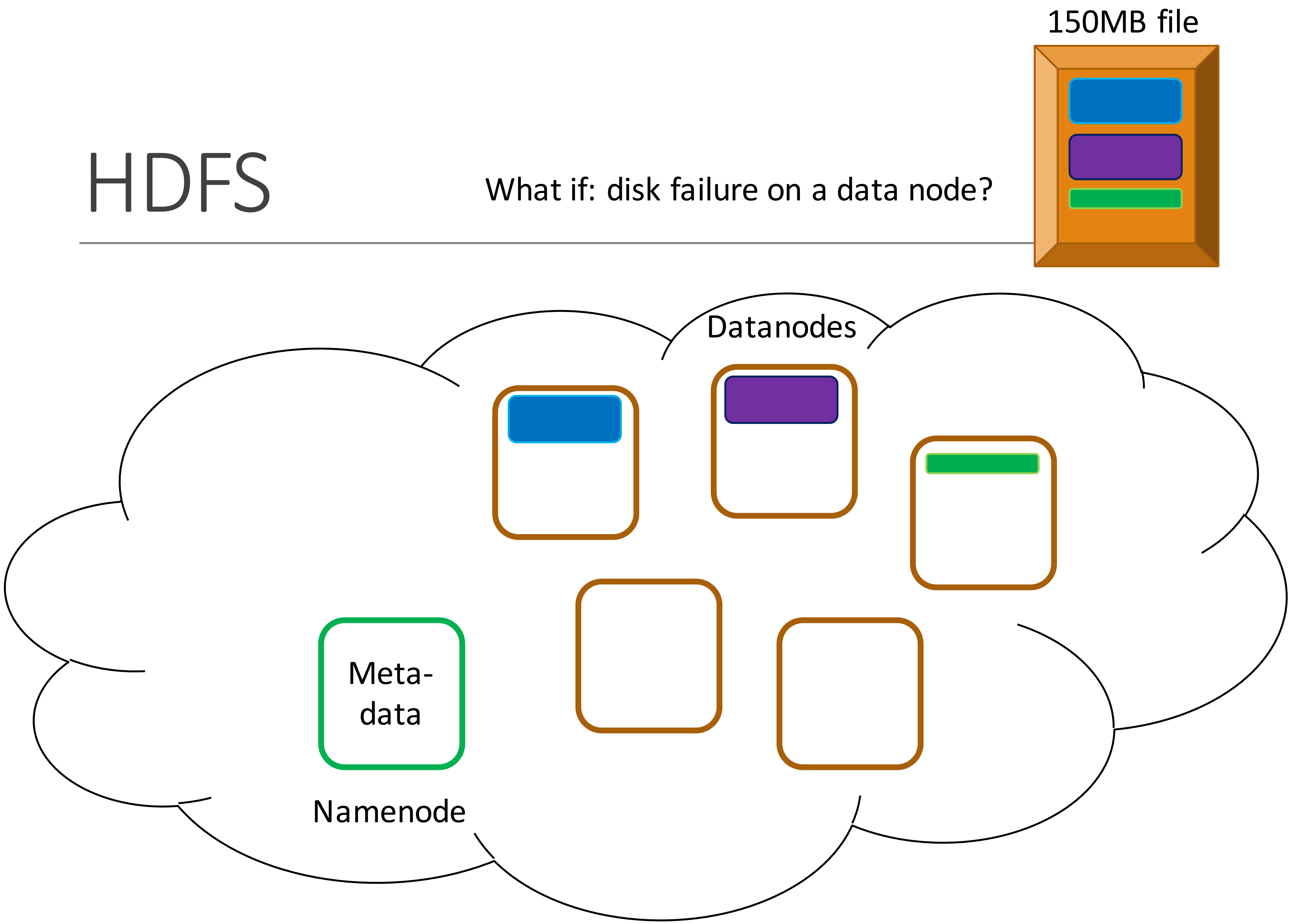
➔ Cloudera Hadoop Distribution (CDH) integrates all these tools

Hadoop Distributed File System (HDFS)

- Looks very much like a normal file system
- Split into blocks (blk_1, blk_2…)
- 64MB default block size
- E.g. 150MB file split in blocks of 64 -64 – 22
- Each block stored in a node
- Daemon runs on each of the machines of cluster:
    - Name node: stores metadata (where each block is stored)
    - Data node: actual data blocks

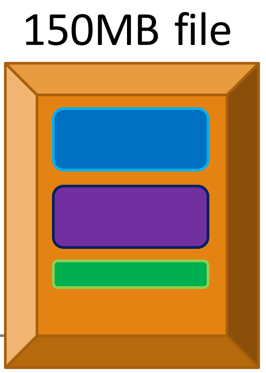
HDFS

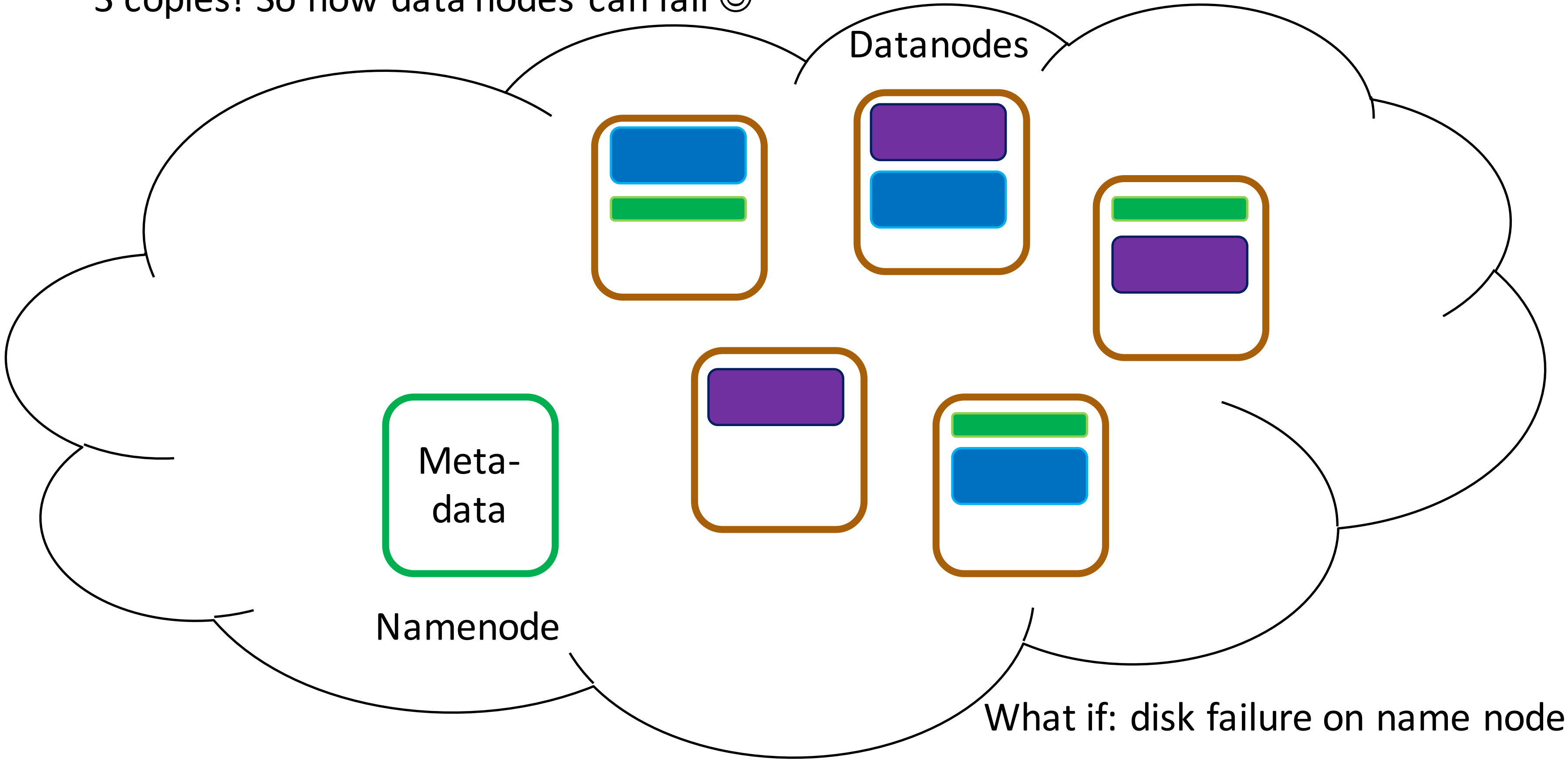
What if: disk failure on a data node?



HDFS: data redundancy!

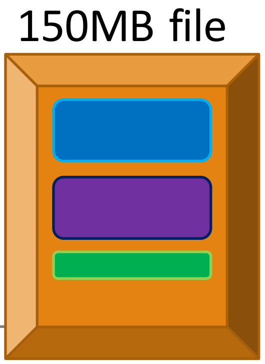
3 copies! So now data nodes can fail ☺



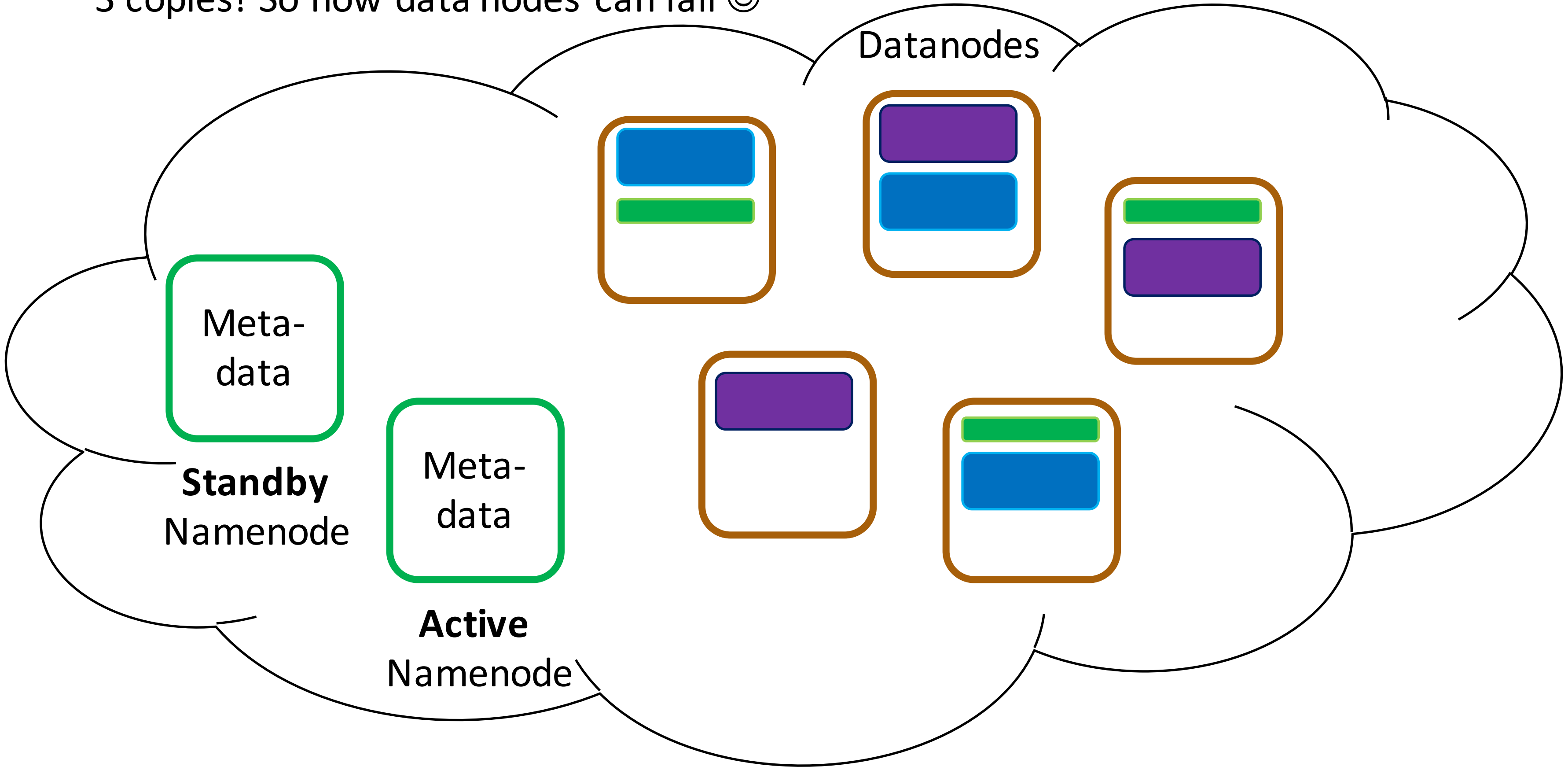


What if: disk failure on name node?

HDFS: standby Name Node



3 copies! So now data nodes can fail ☺



Hadoop filesystem (fs) commands

List files:

```
hadoop fs -ls
```

Store a local file in hdfs:

```
hadoop fs -put myfile.txt
```

Rename file:

```
hadoop fs -mv myfile.txt newname.txt
```

Remove file:

```
hadoop fs -rm filename.txt
```

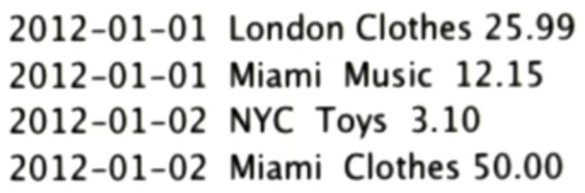
Create directory:

```
Hadoop fs -mkdir newdir
```

=> very similar to traditional unix commands

MapReduce (MR): why?

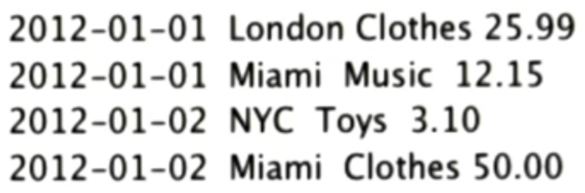
- Processing documents top-bottom: slow
- MR: parallel processing
- Retailer example: calculate total sales per store:
  - Traditional: start from top and increment sales for each store as you go along
  - Store: hash table <key, value>
  - Problems running on 1 T of data?
    - It wont’ work?
    - Out of memory?
    - Long time?
    - Wrong answer?



```
2012-01-01  London Clothes 25.99
2012-01-01  Miami  Music  12.15
2012-01-02  NYC  Toys  3.10
2012-01-02  Miami  Clothes 50.00
```

MapReduce (MR): why?

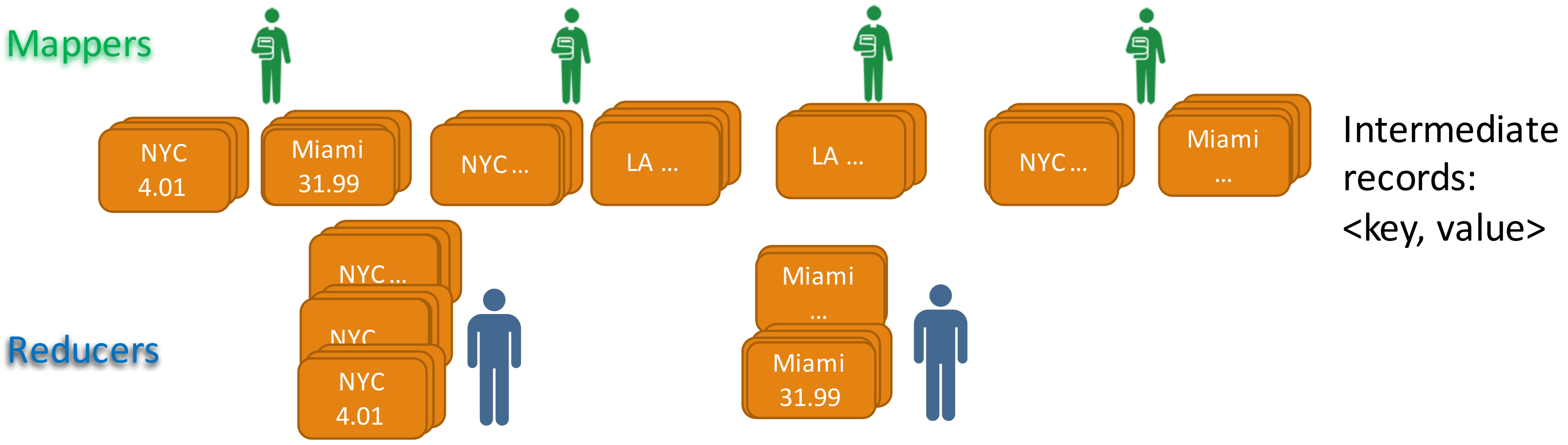
- Processing documents top-bottom: slow
- MR: parallel processing
- Retailer example: calculate total sales per store:
  - Traditional: start from top and increment sales for each store as you go along
  - Store: hash table <key, value>
  - Problems running on 1 T of data?
    - It wont' work? ✗
    - Out of memory? ✓
    - Long time? ✓
    - Wrong answer? ✗



```
2012-01-01  London Clothes 25.99
2012-01-01  Miami  Music  12.15
2012-01-02  NYC  Toys  3.10
2012-01-02  Miami  Clothes 50.00
```

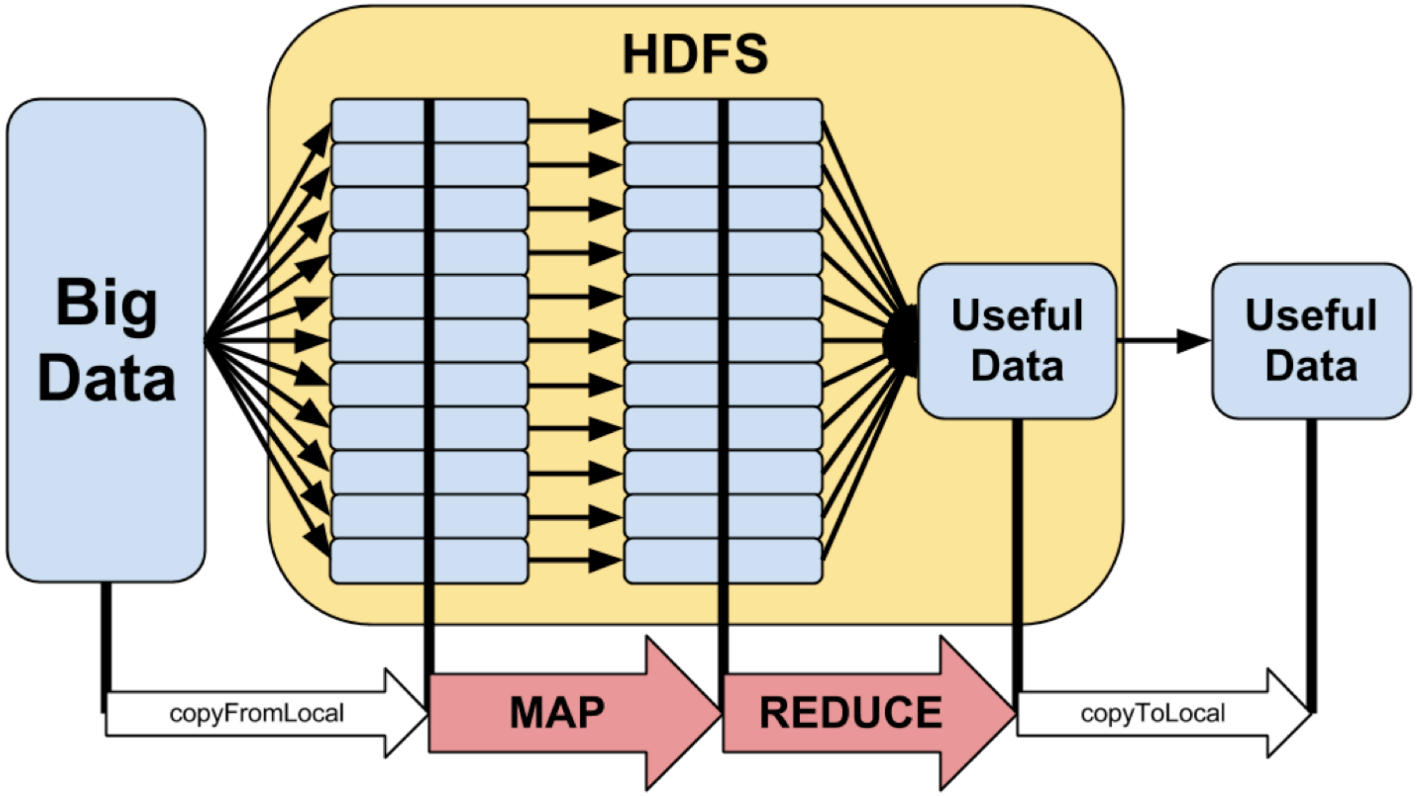
MapReduce

- What if you had more people to help? Mappers and Reducers
- Break ledger into chucks and distribute to mappers

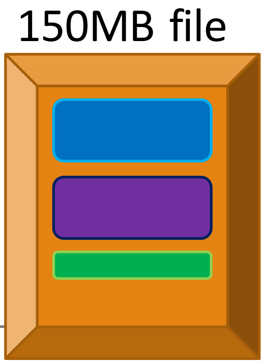


- Reducers get assigned a store
- They ask for the stack of that store at each Mapper: shuffle
- They alphabetically go through their stacks: sort; and process them

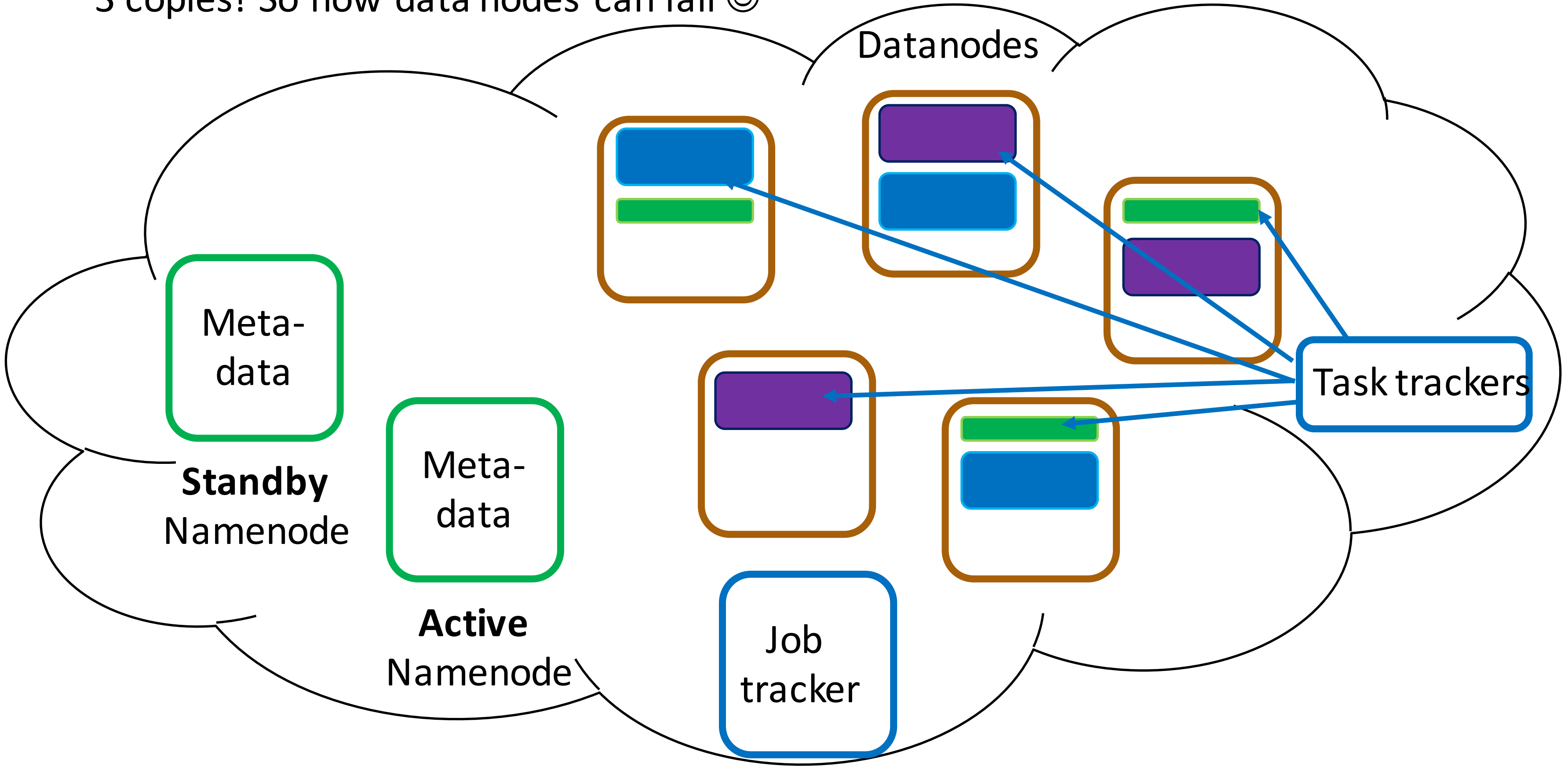
MapReduce


HDFS
Big Data
Useful Data
Useful Data
copyFromLocal
MAP
REDUCE
copyToLocal

Daemons of MR



3 copies! So now data nodes can fail ☺



Daemons of MapReduce

- MapReduce job submitted to Job tracker
- **Job tracker** splits work to mappers and reducers
- **Task tracker**: runs on each data node, executes the actual mapping and reducing
  -> on the same machine as data node, so saves network traffic!
  - Hadoop will try to make sure a task tracker works on data from the same machine (if not already busy)
- **Mappers** perform filtering and sorting and will pass the intermediate data to reducers
- **Reducers** process this (summary operation) and write final output to HDFS

Hadoop & MapReduce

Thursday lab, **please install Hadoop beforehand**, in 1 of 2 ways:

1. (Lazy way) through a virtual machine image ← **preferred**
2. Native install on your Unix machine
    - Installation instructions, see https://goo.gl/c6HDTt

- We will use Hadoop Streaming, which will allow us to use Python for MapReduce scripts