Class 3: What is Big Data?

Realtime and Big Data Analytics

Summer 2017

Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte		

* This column shows magnitude.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}
Terabyte	2^{40}	10^{12}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}
Terabyte	2^{40}	10^{12}
Petabyte	2^{50}	10^{15}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}
Terabyte	2^{40}	10^{12}
Petabyte	2^{50}	10^{15}
Exabyte	2^{60}	10^{18}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}
Terabyte	2^{40}	10^{12}
Petabyte	2^{50}	10^{15}
Exabyte	2^{60}	10^{18}
Zettabyte	2^{70}	10^{21}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Class 3

Name	Number of Bytes (exponential, base 2)	*Number of Bytes (exponential, base 10)
Kilobyte	2^{10}	10^{3}
Megabyte	2^{20}	10^{6}
Gigabyte	2^{30}	10^{9}
Terabyte	2^{40}	10^{12}
Petabyte	2^{50}	10^{15}
Exabyte	2^{60}	10^{18}
Zettabyte	2^{70}	10^{21}
Yottabyte	2^{80}	10^{24}

* This column shows magnitude: $10^3 \sim= 2^{10}$.

How big is BIG?

Name	***Abbr.***	***Number of Bytes (exponential, base 2)***	****Number of Bytes (exponential, base 10)***	****Number of Bytes***
Kilobyte	KB	2^{10}	10^{3}	1,000
Megabyte	MB	2^{20}	10^{6}	1,000,000
Gigabyte	GB	2^{30}	10^{9}	1,000,000,000
Terabyte	TB	2^{40}	10^{12}	1,000,000,000,000
Petabyte	PB	2^{50}	10^{15}	1,000,000,000,000,000
Exabyte	EB	2^{60}	10^{18}	1,000,000,000,000,000,000
Zettabyte	ZB	2^{70}	10^{21}	1,000,000,000,000,000,000,000
Yottabyte	YB	2^{80}	10^{24}	1,000,000,000,000,000,000,000,000

* This column shows magnitude: $10^3 = 1000 \sim= 2^{10}$.

How big is BIG?
Class 3

Name	***Abbr.***	***Bytes (exp., base 2)***	***Number of Bytes***	****Bytes (exp., base 10)***	****Number of Bytes***
Kilobyte	KB	2^{10}	1,**024**	10^3	1,000
Megabyte	MB	2^{20}	1,**048,576**	10^6	1,000,000
Gigabyte	GB	2^{30}	1,**073,741,824**	10^9	1,000,000,000
Terabyte	TB	2^{40}	1,**099,511,627,776**	10^{12}	1,000,000,000,000
Petabyte	PB	2^{50}	1,**125,899,906,842,620**	10^{15}	1,000,000,000,000,000
Exabyte	EB	2^{60}	1,**152,921,504,606,850,000**	10^{18}	1,000,000,000,000,000,000
Zettabyte	ZB	2^{70}	1,**180,591,620,717,410,000,000**	10^{21}	1,000,000,000,000,000,000,000
Yottabyte	YB	2^{80}	1,**208,925,819,614,630,000,000,000**	10^{24}	1,000,000,000,000,000,000,000,000

* This column shows magnitude: $10^3 = 1000 \sim= 2^{10}$, which equals 1024 exactly.

How big is BIG?
Class 3

2006

2009

2013
2014

Name	*Abbr.*	*Bytes*	*Number of Bytes*	**Bytes*
Exabyte	EB	2^{60}	1,152,921,504,606,850,000	10^{18}
Zettabyte	ZB	2^{70}	1,180,591,620,717,410,000,000	10^{21}
Yottabyte	YB	2^{80}	1,208,925,819,614,630,000,000,000	10^{24}

2006 - World's hard drives estimated at: **~160 exabytes (EB)**
2009 - Internet estimated to contain: **~500 exabytes (EB)**
By 2013, entered ZB range

How big is BIG?

Class 3

Storage Sizes – How big is BIG?

Range	*Example*
Kilobyte	
Megabyte	
Gigabyte	
Terabyte	
Petabyte	
Exabyte	
Zettabyte	
Yottabyte	

[1] http://en.wikipedia.org/wiki/Yottabyte

Storage Sizes – How big is BIG?

Range	*Example*
Kilobyte	Text file
Megabyte	Song, mp3 file
Gigabyte	Movie file
Terabyte	External laptop hard drive
Petabyte	Rack of nodes, e.g. Oracle Big Data Appliance (BDA)
Exabyte	Datacenter
Zettabyte	(Internet data in 2009 = 500EB) + (All the world's hard drives in 2006 = 160EB) + (Internet data 2009 to present) + (All the world's hard drives 2006 to present)
Yottabyte	???????

[1] http://en.wikipedia.org/wiki/Yottabyte

What is Big Data?
Class 3

Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

"Big" data is not new

- Oil companies, telecommunications companies, and other data-centric industries have had huge datasets for a long time.

- Datacenter energy example

- GPS ground stations example

Reference: http://radar.oreilly.com/2010/06/what-is-data-science.html

What is Big Data?

Class 3

As storage capacity continues to expand,

today’s “big” is

tomorrow’s “medium” and

next week’s “small.”

Reference: http://radar.oreilly.com/2010/06/what-is-data-science.html

What is Big Data?
Class 3

"Big data" is …
when the size of the data itself
becomes part of the problem.

At some point, traditional techniques for working with data run out of steam.

Reference: http://radar.oreilly.com/2010/06/what-is-data-science.html

What is Big Data?
Class 3

Size of the ‘digital universe’ in…

2006 – 0.18 ZB

2011 – 1.8 ZB ← *10-fold growth in five years!*

2013 – 4.4 ZB ← *More than doubled in 2 years*

Reference: Hadoop: The Definitive Guide, by Tom White

What is Big Data?
Class 3

New York Stock Exchange	Over 4 TB of ***new*** data ***per day***
Facebook	240 billion photos, growing by 7 PB per month
Large Hadron Collider (Geneva)	Produces 30 PB per year

Reference: Hadoop: The Definitive Guide, by Tom White

What is Big Data?

Class 3

No longer is it only corporations who generate mountains of data.

Now, individuals have a large and growing footprint too.

Consider these sources…

Photos	*Spreadsheets*	*Tweets*
Blogs	*Sensor Data*	*YouTube Videos*
PowerPoints	*Word Documents*	*Etc….*

Reference: Hadoop: The Definitive Guide, by Tom White

What is Big Data?
Class 3

Can you think of another contributor?

What is Big Data?

Class 3

Machines! They generate operation logs

- Monitoring agents installed in servers, laptops, and Virtual Machines
  - Monitoring data can include CPU utilization, Network Utilization, Disk IO, Memory Utilization
- Raw monitoring data are collected every second/minute/hour
- Raw monitoring data are summed to higher levels of granularity, e.g. week/ month/year – and stored this way in data warehouses!

Reference: Hadoop: The Definitive Guide, by Tom White

What is Big Data?

Class 3

Machines generate usage logs too

- EZ Pass
- GPS tracking tools
- Retail transactions - think of Amazon, EBay, PayPal - globally!
- Consumer historic data (again, summarized/rolled-up data)
- Computer and network performance for SLAs (Service Level Agreements)
- Computer security logs
- Predictions about consumer behavior today and tomorrow which are inputs to predictions for all tomorrows…

Reference: Hadoop: The Definitive Guide, by Tom White

Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

Why is Big Data a problem?

Class 3

Problems:

Where can I store my company's ever-growing data?
How much is that going to cost?
How am I going to manage all that hardware and software?
Users are asking bigger questions – how can I provide compute power? …

2006

2009

2013
2014

Name	*Bytes*	*Number of Bytes*	*Example*
Terabyte	2^{40}	1,099,511,627,776	External laptop hard drive
Petabyte	2^{50}	1,125,899,906,842,620	Rack of nodes, Oracle Big Data Appliance (BDA)
Exabyte	2^{60}	1,152,921,504,606,850,000	Datacenter
Zettabyte	2^{70}	1,180,591,620,717,410,000,000	All internet data + all the world's hard drives
Yottabyte	2^{80}	1,208,925,819,614,630,000,000,000	…

Problem: Users are asking Bigger Questions
Class 3

- Several thousand dish antennas will augment millions of low frequency antennas
- Will cover one million square meters, spiral layout
- Operational in the mid 2020s
- Antennas will gather 14 EB daily and store about 1 PB

Square Kilometre Array (SKA) – the world's largest telescope

References:
http://www.scribd.com/doc/125147649/Ultimate-Big-Data-Challenge#page=1
https://www.skatelescope.org
Artist's impression - https://www.skatelescope.org/layout/

Problem: Cost of Storing and Processing Big Data

Class 3

- **Square Kilometre Array (SKA)**
  - Fully operational in 2024, €1.5 billion, very expensive supercomputer solution
  - Glimpse back 13 billion years to answer questions about the origins of the universe
  - Will be built in Sub-Saharan states with cores in South Africa and Australia, where the view of the Milky Way Galaxy is best and radio interference least.
  - Will generate **1 EB** of data **each DAY** from 3000 radio telescopes.
    - Rounding, that's about **1 ZB** every two years
  - Requires long-haul links with a capacity greater than the current global Internet
  - **Will survey the sky more than 10,000 times faster than ever before**
  - Construction scheduled to begin in 2016, observations begin by **2019**
  - The headquarters of the project is in Manchester, in the U.K.

Ref: http://spectrum.ieee.org/tech-talk/aerospace/astrophysics/an-exascale-challenge-for-radio-astronomy?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%253A%20IeeeSpectrum%20%2528IEEE%20Spectrum%2529

Remember - I/O Speed is a Problem

Class 3

	1990	…2003, 2004, 2005	2011
Drive Capacity	1.4 GB	* ~1000 =	1 TB
Transfer Speed	4.4 MB/sec	* ~25 =	100 MB/sec
Whole drive read time	5 minutes		2.5 HOURS
Whole drive write time	… Even slower …		

Reference: Hadoop: The Definitive Guide, by Tom White

How do we manage these problems?

Class 3

Hadoop can help with

- Cost of storing and processing Big Data
- Hard drive transfer speed
- Processing power
- Hardware and software management
- Users asking Bigger questions
    - Low-cost platform for formulating Bigger questions that consume Big Data

How can we solve the Big Data problem(s)?

Class 3

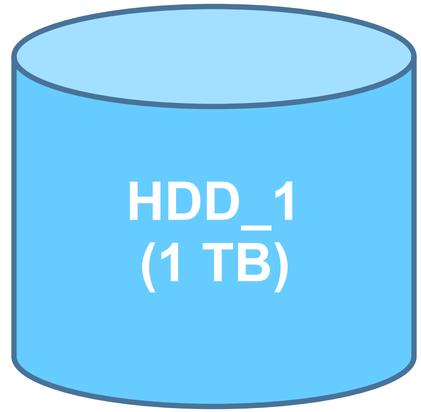
Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

How can we solve the Big Data problem?

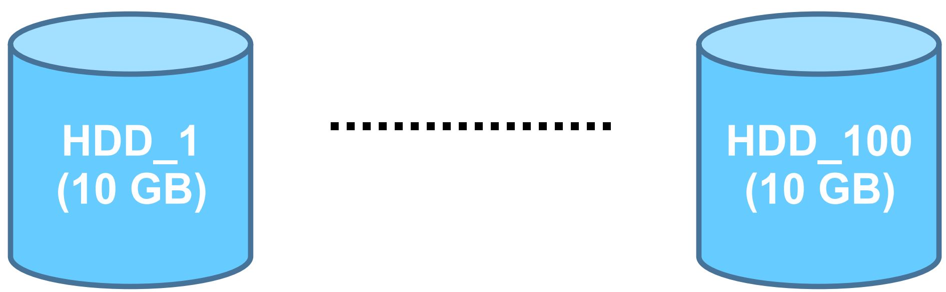
Class 3

1 Terabyte Hard Disk Drive (HDD):



Or

1 Terabyte of Storage with 100 HDDs:

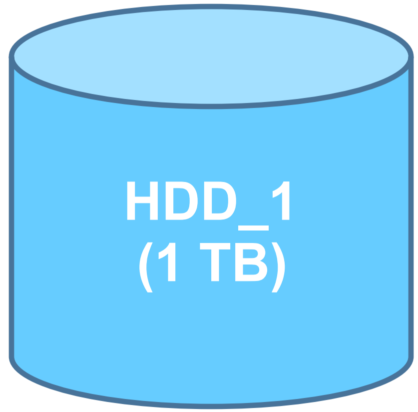


Which is better?

How can we solve the Big Data problem?

Class 3

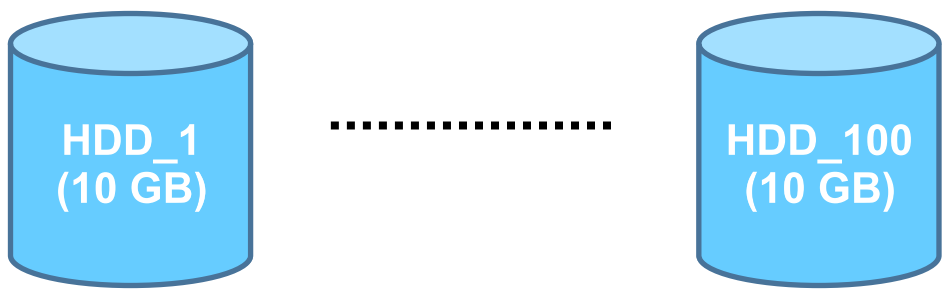
1 Terabyte Hard Disk Drive (HDD):



- One computer reading from one drive is inefficient - 1 r/w head

Or

1 Terabyte of Storage with 100 HDDs:

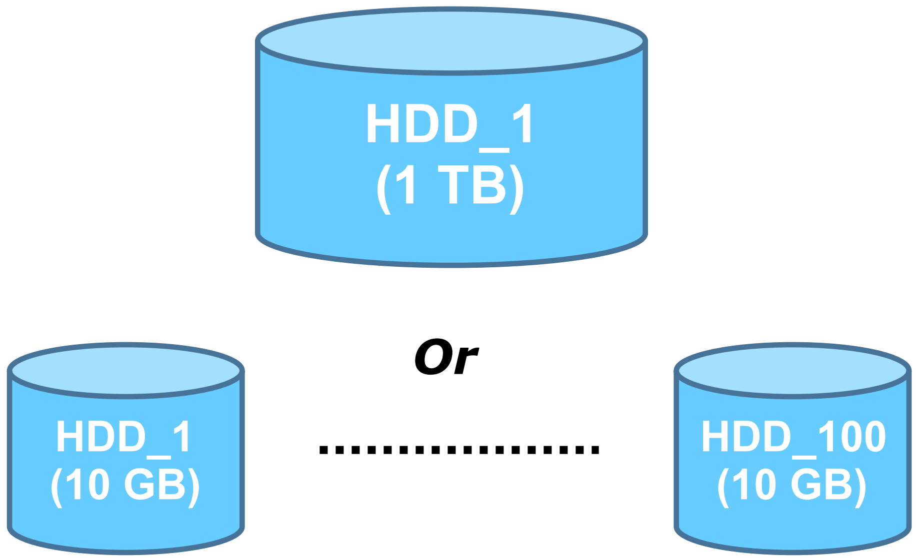


- Prefer multiple drives

Which is better?

How can we solve the Big Data problem?
Class 3

1 Terabyte:



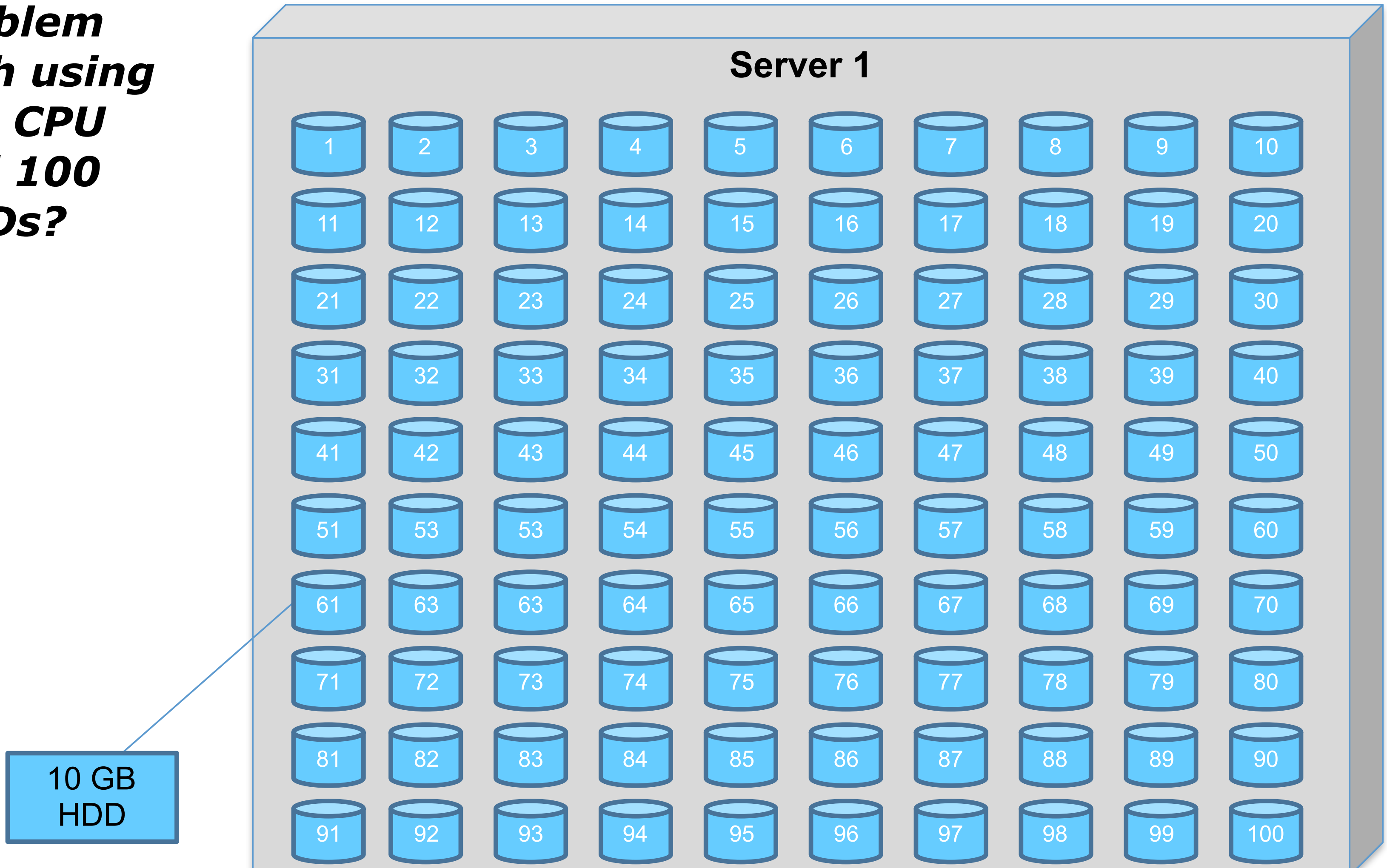
1 TB of data spread across 100 HDDs is better

- Advantage is one read/write head *per* drive

How can we solve the Big Data problem?

Class 3

Any problem with using one CPU and 100 HDDs?

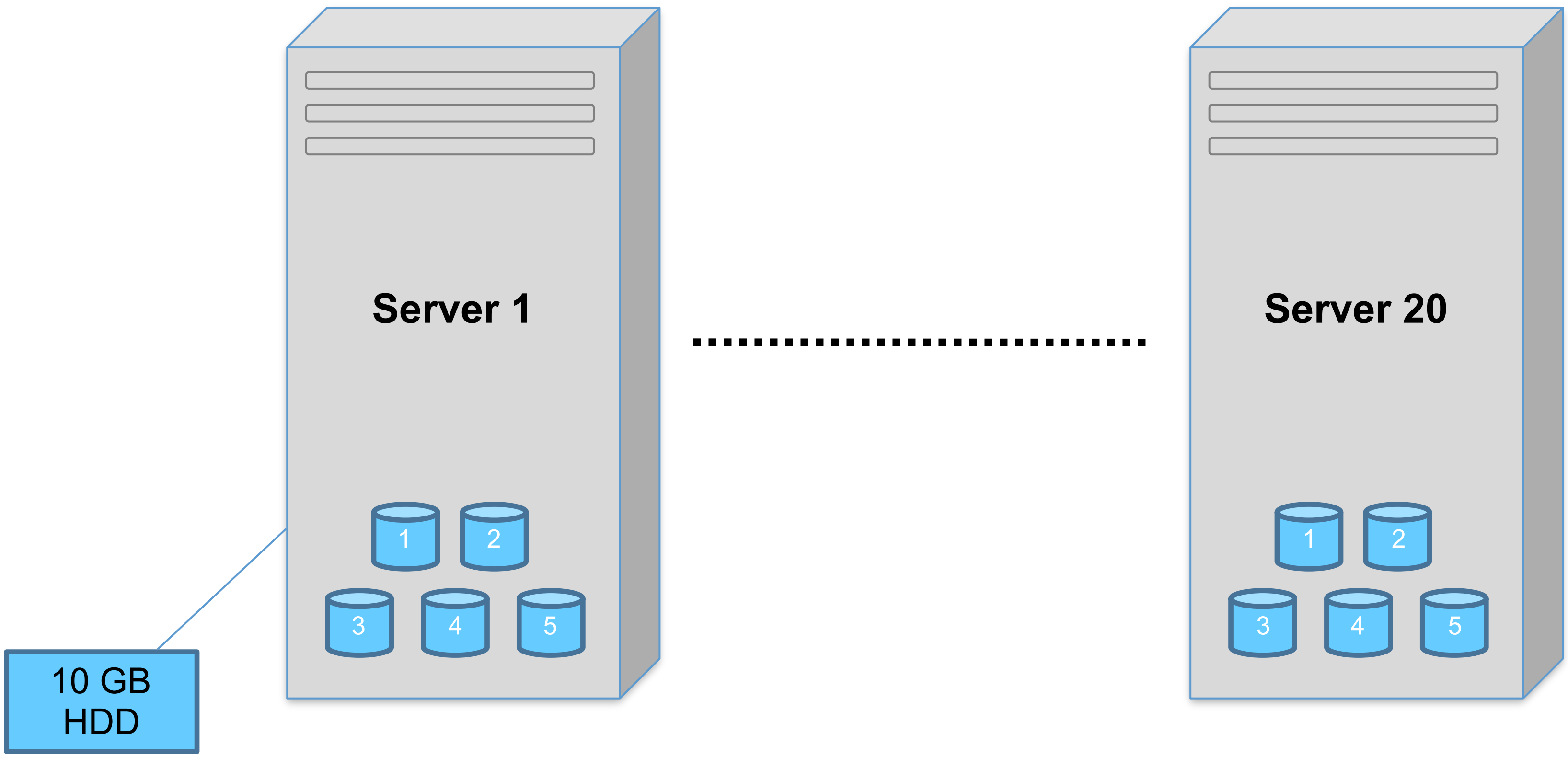


How can we solve the Big Data problem?

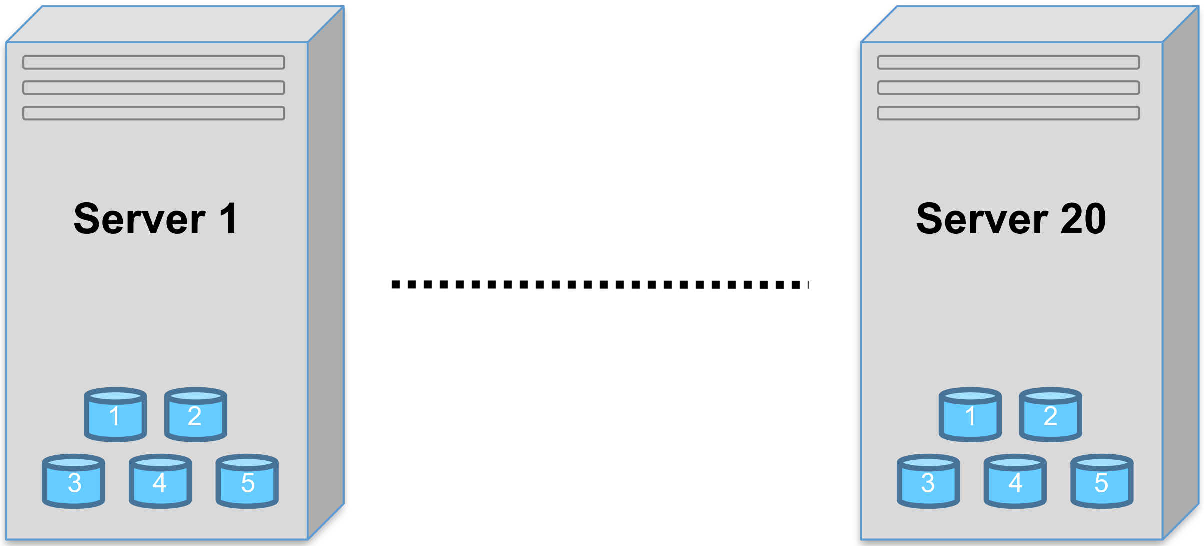
Class 3

What if, instead of one computer with 100 HDDs, we distribute the HDDs across 20 computers?

- ***Each computer has five 10GB HDDs***



How can we solve the Big Data problem?



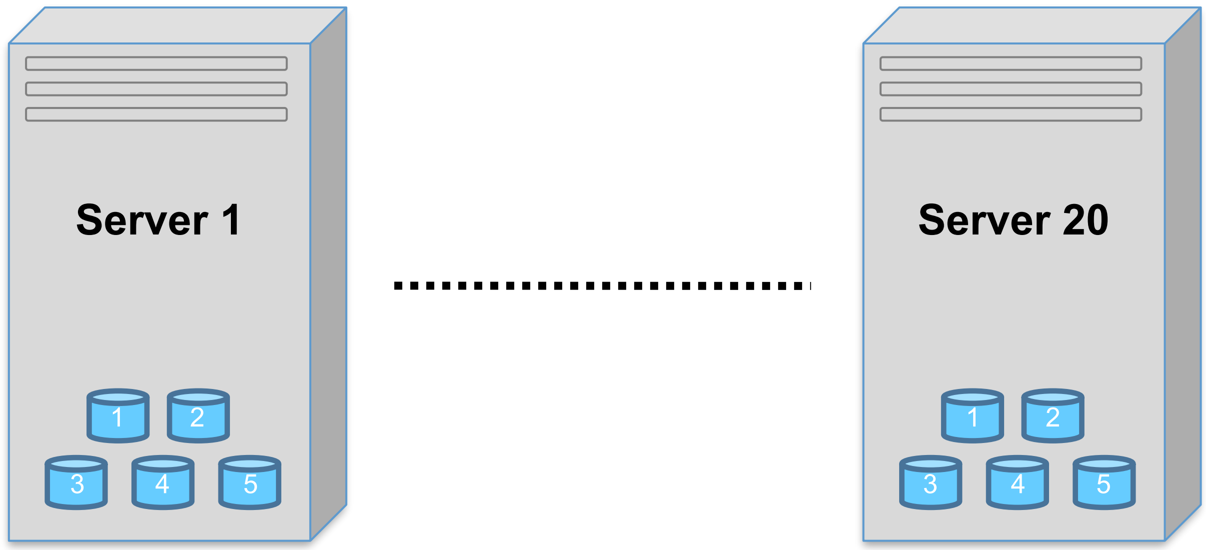
Given: 10 GB per drive = 10,000,000,000 bytes per drive
20 servers * 5 HDDs per server * 10GB per HDD = 1 TB
Read rate is 100 MB/second

The full 1 TB of data can be read in 100 seconds :
10 GB / 100 MB per second = 10,000,000,000 / 100,000,000 =
100 seconds to read one drive.

This is how we can read 1TB in 100 seconds, instead of 2.5 hours.

How can we solve the Big Data problem?

Class 3



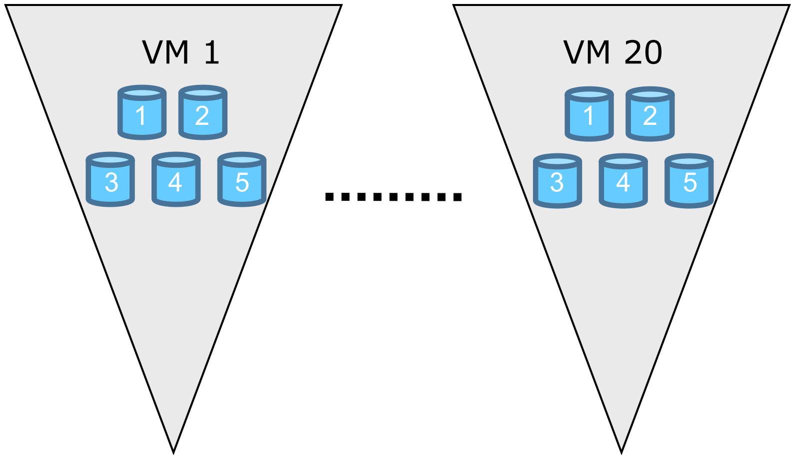
This is the architecture in which Hadoop shines because not only is the data read in parallel, it is processed in parallel as well.

Notes:

- In practice, multiple drives are installed in one server, sometimes as many as twelve or more. Each drive is usually 2-4 TB in size.

**** It is important to match the speed of the drives to the processing power of the server.***

How can we solve the Big Data problem?
Class 3



We can also choose to build our cluster using Virtual Machines (VMs) hosted by your favorite cloud provider:

- Amazon EC2
- IBM BlueMix
- Google Compute Engine
- Digital Ocean

VMs provide us with elastic resources in terms of
- Compute power
  - Number of VMs
  - Number of CPUs per VM
- Storage size
- Memory size

Hadoop as a Solution

Class 3

Hadoop solves

- Cost of storing and processing Big Data
  - Commodity hardware can be used in Hadoop cluster deployment
  - Option to deploy Hadoop cluster in private or public cloud
- Hard drive transfer speed, processing power
  - Multiple hard drives per machine across cluster
  - Processors utilized in parallel to solve problems
  - Scales linearly

Hadoop as a Solution

Class 3

Hadoop solves

- At-scale hardware and software management problems
    - Ambari (free, open source), Cloudera Manager (free, proprietary), etc. for cluster management
- Users asking Bigger questions
    - Low-cost platform
    - Formulate Bigger questions using familiar tools (Java, SQL, Python, C++)
    - Tools for analysts, data scientists, machine learning, prediction, …

Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

HDFS

- Block-structured file system

- Individual files are broken into blocks of a fixed size
  - HDFS block size is 128MB by default in Apache Hadoop
  - HDFS blocks are large compared to disk blocks (512 bytes) or filesystem blocks (4KB)
  - Optimal streaming achieved by reducing the latency that many seeks would cause

- Blocks stored across cluster in one or more machines – **DataNodes**

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

- In HDFS, a file can be made of several blocks, and they are not necessarily stored on the same machine
  - Access to a file may require cooperation of multiple machines
  - Advantage: Support for files whose sizes exceed what one machine can accommodate

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

- HDFS stores files as a set of large blocks across several machines, and these files are not part of the ordinary file system
  - Typing ***ls*** on a machine running a DataNode daemon will display the contents of the ordinary Linux file system being used to host the Hadoop services
    - Files stored inside HDFS are not shown
    - HDFS runs in a separate namespace
  - HDFS comes with its own utilities for file management
  - Blocks that comprise the HDFS files are stored in a directory managed by the DataNode service

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

Distributed File Systems
Class 3

- When the blocks of a file are distributed across the cluster, several machines participate in serving up the file

  - The loss of any one of those machines would make the file unavailable

  - Solution is replication of each block across a number of machines (3 machines, by default)

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

An HDFS cluster is comprised of two types of nodes:

- One NameNode - Master
  - Optionally, can have a Standby NameNode for High Availability (HA)
- Multiple DataNodes (Worker nodes, subservient to NameNode)

In HDFS

- File data is accessed in a write once, read many (WORM) model
- Metadata structures (names of files and directories) can be modified by many clients concurrently
- Metadata remains synchronized by using single machine to manage the metadata – the **NameNode**

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

NameNode

- Master
- Manages filesystem namespace
- Maintains filesystem tree
- Maintains metadata for all files and directories in the tree
  - File names
  - Permissions
  - Locations, i.e. DataNodes, of each block of each file
  - Information is stored in the main memory of NameNode for fast access

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

NameNode Resilience

- Important that NameNodes be resilient to failure
  - Without NameNode, cannot use the Hadoop distributed filesystem

- NameNode marks bad blocks, creates new good replicas – automatically

- For recovery
  - Metadata is persisted in the local filesystem
  - Optionally, persisted to multiple backup filesystems
  - Can add a Standby NameNode for High Availability (HA)

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

NameNode Resilience (continued)

- High Availability with a Secondary NameNode (old approach)
  - Role of Secondary NameNode is different from primary NameNode
  - Manages the edit log by continuously merging the namespace image
  - Lags the primary
  - Can be promoted to primary for recovery – **not automatic**, it's a manual process
- HA with a Standby NameNode (New Approach)
  - The Standby NameNode tracks the state of the cluster very closely, so failovers typically do not result in data loss
  - **Automatic** failover if NameNode dies
  - This is a hot standby

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

DataNodes

- Worker nodes
- Subservient to NameNode of the cluster
- Store and retrieve blocks on demand
  - One large file is split into multiple HDFS blocks
  - Each HDFS block is stored in a DataNode
- Report to NameNode periodically with lists of blocks they are storing
- Compute checksums over blocks
- Report checksum errors to NameNodes

Reference: Hadoop: The Definitive Guide (Chapter 3), by Tom White

- NameNode and DataNode cooperate to access data in HDFS files
    - NameNode provides the list of locations (DataNodes) where blocks that comprise the file are stored, including locations of replicas
    - Data is read from DataNode servers
    - NameNode is not involved in bulk data transfer (if a transfer is needed), keeping its overhead to a minimum

Reference: http://developer.yahoo.com/hadoop/tutorial/module2.html

- If a DataNode fails
  - Data can be retrieved from one of the DataNodes storing replicas of the block
  - Cluster continues to operate

- If the NameNode fails
  - Multiple redundant systems allow the NameNode to protect the file system's metadata in the event of NameNode failure (see earlier slide)
  - NameNode failure is more severe for the cluster than DataNode failure

Reference: http://developer.yahoo.com/hadoop/tutorial/module2.html

Agenda

1. How big is BIG?
2. What is Big Data?
3. Why is Big Data a problem?
4. How can we solve the Big Data problem?
5. Hadoop – HDFS
6. Hadoop MapReduce – Review of the Weather Program

Weather data example – find the max temperature

Dataset (small version):

0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999
0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999
0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999
0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999
0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999

Input to Mapper:

```
Key,   Value          Year                                                                  Temp
0,     0067011990999991950051507004+68750+023550FM-12+038299999V0203301N00671220001CN9999999N9+00001+99999999999
106,   0043011990999991950051512004+68750+023550FM-12+038299999V0203201N00671220001CN9999999N9+00221+99999999999
212,   0043011990999991950051518004+68750+023550FM-12+038299999V0203201N00261220001CN9999999N9-00111+99999999999
318,   0043012650999991949032412004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+01111+99999999999
424,   0043012650999991949032418004+62300+010750FM-12+048599999V0202701N00461220001CN0500001N9+00781+99999999999
```

The key seen by the Mappers is the offset of the start of each record in the file.

MaxTemperatureMapper.java

```
// cc MaxTemperatureMapper Mapper for maximum temperature example
// vv MaxTemperatureMapper
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class MaxTemperatureMapper
  extends Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {

    String line = value.toString();
    String year = line.substring(15, 19);                    //Pickup the year
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);                  //Pickup data that tells us if the data are good
    if (airTemperature != MISSING && quality.matches("[01459]")) {      //Data cleansing step
      context.write(new Text(year), new IntWritable(airTemperature));   //Looks good, write out the intermediate key/value pair (year, temp)
    }
  }
}
// ^^ MaxTemperatureMapper
```

MaxTemperatureMapper.java

```
// cc MaxTemperatureMapper Mapper for maximum temperature example
// vv MaxTemperatureMapper
…
public class MaxTemperatureMapper
  extends Mapper<LongWritable, Text, Text, IntWritable> {

  private static final int MISSING = 9999;

  @Override
  public void map(LongWritable key, Text value, Context context)
      throws IOException, InterruptedException {

    String line = value.toString();
    String year = line.substring(15, 19);                    //Pickup the year
    int airTemperature;
    if (line.charAt(87) == '+') { // parseInt doesn't like leading plus signs
      airTemperature = Integer.parseInt(line.substring(88, 92));
    } else {
      airTemperature = Integer.parseInt(line.substring(87, 92));
    }
    String quality = line.substring(92, 93);                 //Pickup data that tells us if the data are good
    if (airTemperature != MISSING && quality.matches("[01459]")) {   //Data cleansing step
      context.write(new Text(year), new IntWritable(airTemperature));   //Looks good, write out the intermediate key/value pair to local disk
    }
  }
}
// ^^ MaxTemperatureMapper
```

Output from Mapper is ungrouped and unsorted (but will be sorted before it is made available to Reducer).
Before sorting, the Mapper output looks like this:
(1950, 0)
(1950, 22)
(1950, -11)
(1949, 111)
(1949, 78)

MaxTemperatureReducer.java

```
// cc MaxTemperatureReducer Reducer for maximum temperature example
// vv MaxTemperatureReducer
import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class MaxTemperatureReducer
  extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      //Input types must match Mapper output types

      throws IOException, InterruptedException {

    int maxValue = Integer.MIN_VALUE;
    for (IntWritable value : values) {
      maxValue = Math.max(maxValue, value.get());   //Iterate over array of values for each key (year)
                                                     //After Mapper has output intermediate results, the results are grouped and sorted.
                                                     //So Reducer sees as input: (1949, [111, 78]) and (1950, [0, 22, -11])
    }
    context.write(key, new IntWritable(maxValue));   //Write out result
  }
}
// ^^ MaxTemperatureReducer
```

MaxTemperatureReducer.java

```
// cc MaxTemperatureReducer Reducer for maximum temperature example
// vv MaxTemperatureReducer
…
public class MaxTemperatureReducer
  extends Reducer<Text, IntWritable, Text, IntWritable> {

  @Override
  public void reduce(Text key, Iterable<IntWritable> values, Context context)
      //Input types must match Mapper output types

      throws IOException, InterruptedException {

    int maxValue = Integer.MIN_VALUE;
    for (IntWritable value : values) {
      maxValue = Math.max(maxValue, value.get());    //Iterate over array of values for each key (year)
                                                     //After Mapper has output intermediate results, the results are sorted by key
                                                     //and grouped by key. So Reducer sees: (1949, [111, 78]) and (1950, [0, 22, -11])
    }
    context.write(key, new IntWritable(maxValue));   //Write out result
  }
}
// ^^ MaxTemperatureReducer
```

```
Output from Reducer is:
(1949, 111)
(1950, 22)
```

MaxTemperature.java   - This is the job control file

```
// cc MaxTemperature Application to find the maximum temperature in the weather dataset
// vv MaxTemperature
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class MaxTemperature {

  public static void main(String[] args) throws Exception {
    if (args.length != 2) {
      System.err.println("Usage: MaxTemperature <input path> <output path>");
      System.exit(-1);
    }

    Job job = new Job();
    job.setJarByClass(MaxTemperature.class);
    job.setJobName("Max temperature");

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapperClass(MaxTemperatureMapper.class);
    job.setReducerClass(MaxTemperatureReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
// ^^ MaxTemperature
```

Homework
Class 3

Please see homework packet.