CS523 - BDT
Big Data Technologies

Lesson 1

Introduction

(What is Big Data & Hadoop?
And a peek into HDFS…)

What’s Big Data?

Big Data is when data itself becomes part of the problem!

No single definition; here is from Wikipedia:

- Big data is a broad term for data sets so large or complex that traditional data processing applications are inadequate.
- Challenges include capture, store, process, search and analyze using conventional DB systems.
- Uses of Big Data are shaping the world around us, offering more qualitative insights into our everyday lives.

Measuring Data

Bit	1 bit	1/8
Nibble	4 bits	1/2 (rare)
Byte	8 bits	1
Kilobyte	1,024 bytes (2^{10})	1,024
Megabyte	1,024 kilobytes (2^{20})	1,048,576
Gigabyte	1,024 megabytes (2^{30})	1,073,741,824
Terabyte	1,024 gigabytes (2^{40})	1,099,511,627,776
Petabyte	1,024 terabytes (2^{50})	1,125,899,906,842,624
Exabyte	1,024 petabytes (2^{60})	1,152,921,504,606,846,976
Zettabyte	1,024 exabytes (2^{70})	1,180,591,620,717,411,303,424
Yottabyte	1,024 zettabytes (2^{80})	1,208,925,819,614,629,174,706,176

Growth of Data

IDC estimates

- 4.4 zettabytes in 2013 (zettabytes = 10^{21})
- 44 zettabytes in 2020

The world's information gets doubled in every 2 years!

Big Data = BD = Beyond Dimensions

IDC - International Data Corporation, an American market research, analysis and advisory firm, specializes in information technology, telecommunications, and consumer technology, Software Development.

Sources of Big Data

Google :- 3.5 billion searches per day; processed 20 PB a day in 2008.

Facebook :- 2.5 PB of user data, growing 15 TB per day. In 2013, they had 100 PBs of photos and videos! 1 out of every 13 people is a FB user and half of them are logged in on any one day!!

Twitter :- users tweet nearly 300,000 times every min.

Instagram :- users post nearly 220,000 new photos every min.

YouTube :- users upload 72 hours of new video content every min.

Amazon :- generates over $80,000 in online sales every min.

eBay :- 9 PBs of storage and growing by 150 billion new records per day.

NYSE :- generates about 4-5 TBs of data per day.

Pearson OpenClass program is 10 TB of graph data with 6.24B vertices and 121B edges!

Sources of Big Data

Web Logs

66.249.65.107 - - [08/Oct/2007:04:54:20 -0400] "GET /support.html HTTP/1.1" 200 11179 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)"

111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET / HTTP/1.1" 200 10801 "http://www.google.com/search?q=log+analyzer&ie=utf-8&oe=utf-8 &aq=t&rls=org.mozilla:en-US:official&client=firefox-a" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"

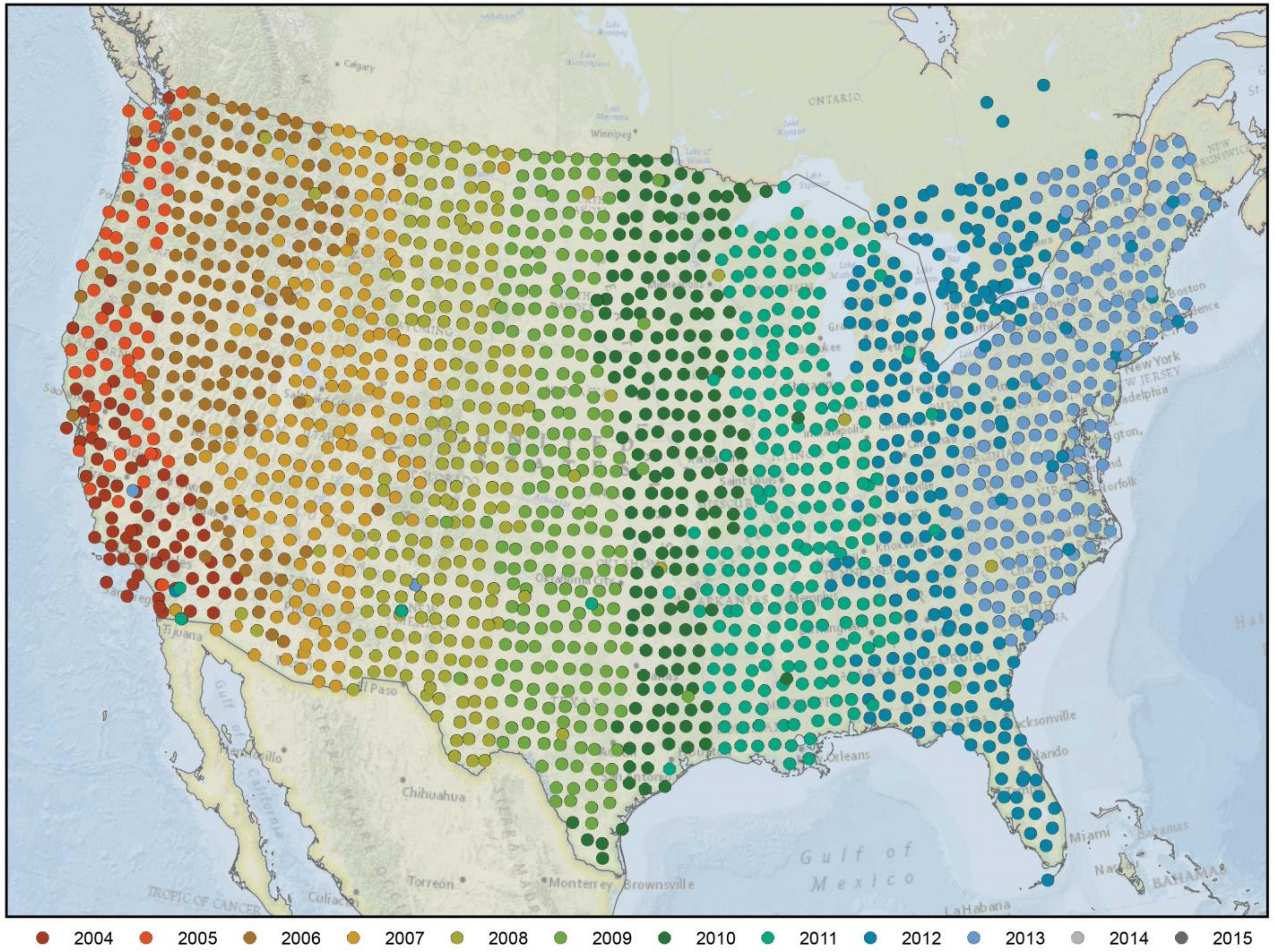
111.111.111.111 - - [08/Oct/2007:11:17:55 -0400] "GET /style.css HTTP/1.1" 200 3225 "http://www.loganalyzer.net/" "Mozilla/5.0 (Windows; U; Windows NT 5.2; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"

Sources of Big Data

Climate/Weather Data Modeling

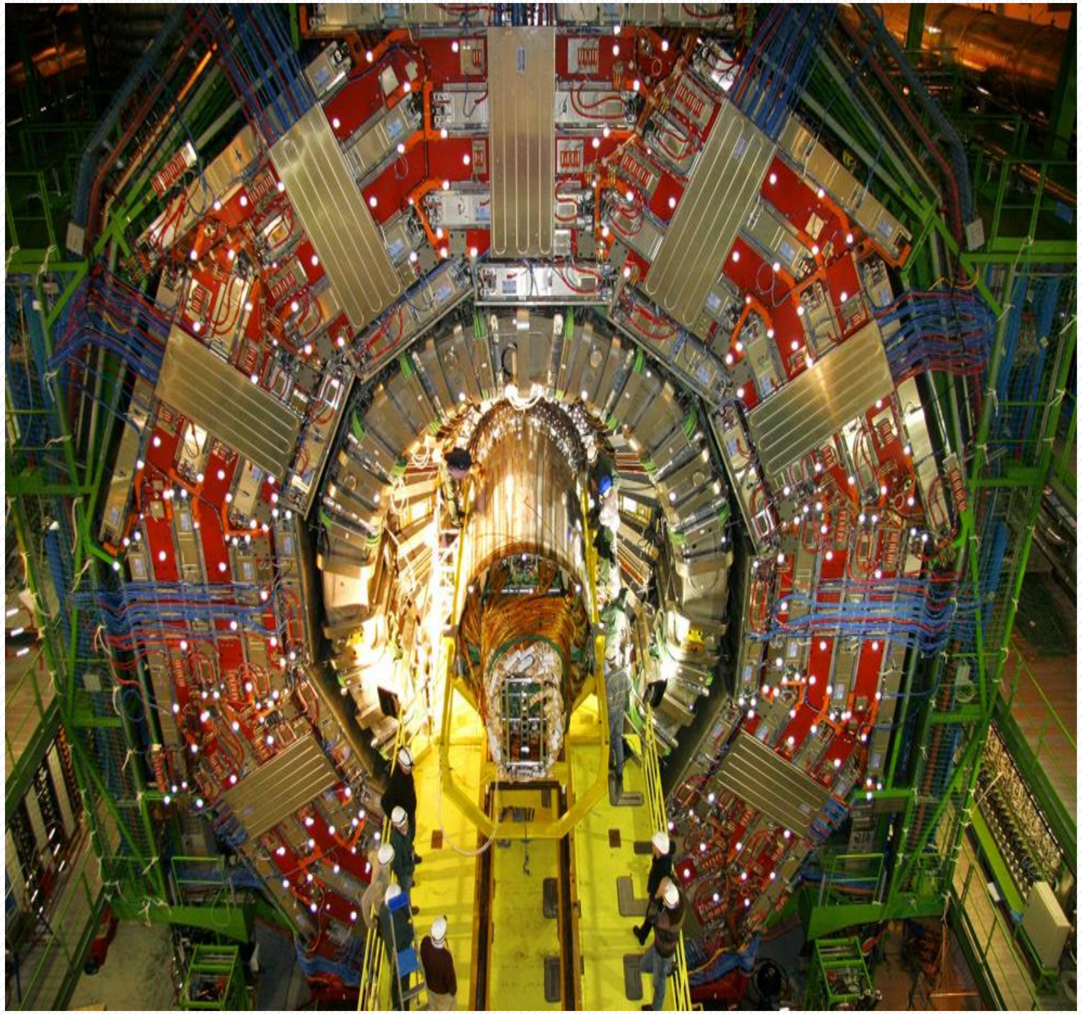
Exploring the Structure and Evolution of the North American Continent

The Earthscope Project



Sources of Big Data

CERN's Large Hadron Collider (LHC) generates 30 PB a year



Sources of Big Data

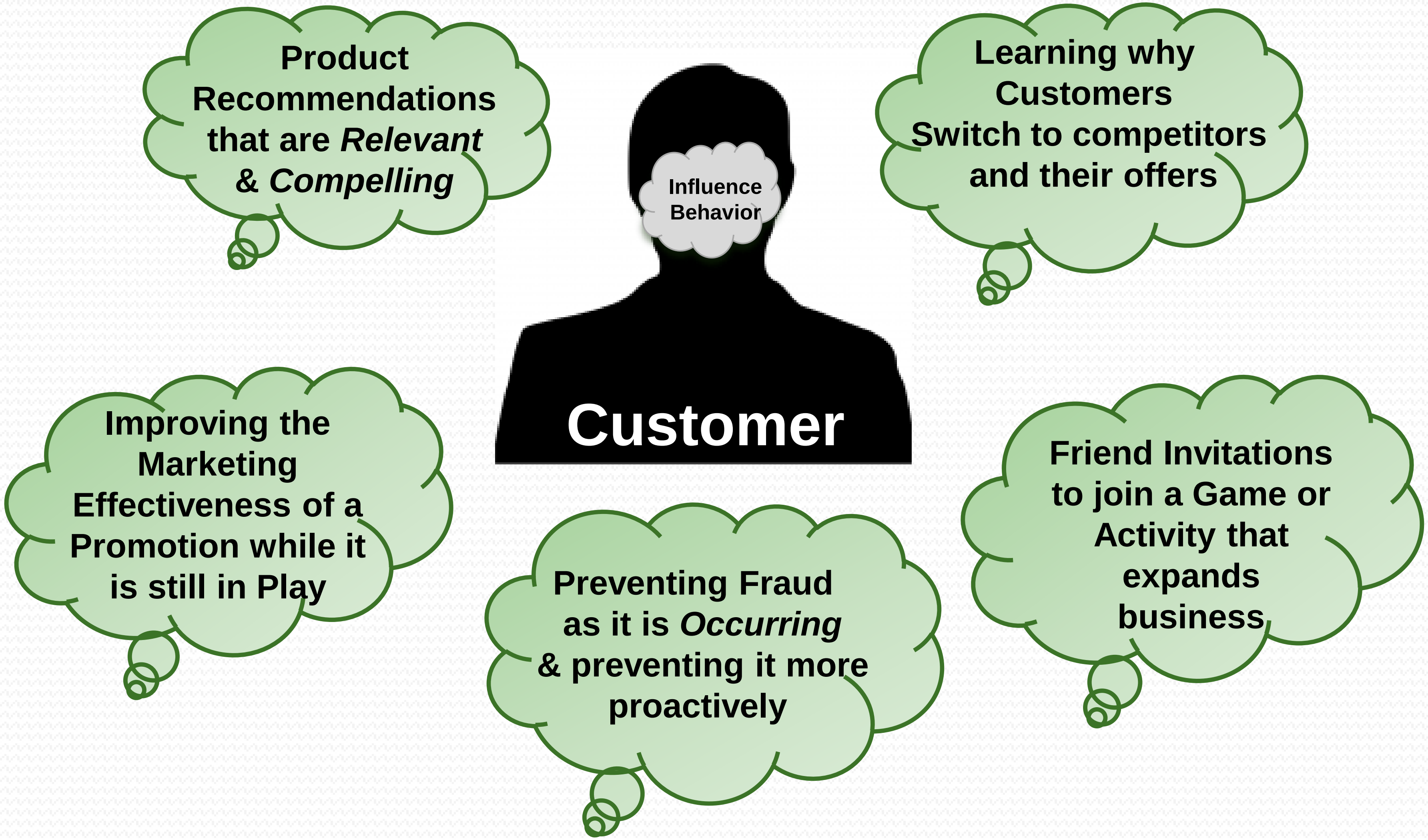
- **Bioinformatics data**:- From about 3.3 billion base pairs in a human genome to huge number of sequences of proteins and the analysis of their behaviors.
- **Financial applications**:- That analyses volumes of data for trends and other deeper knowledge.
- **Health Care**:- Huge amount of patient data, drug and treatment data
- **The universe:-** The Hubble ultra deep telescope shows 100s of galaxies each with billions of stars…

Why to Store & Analyze Big Data?

- Are you living in a world in which more data provides diminishing returns or are you living in a world in which more data truly is better?

- The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data.
- It allows for correlations to be found to "spot business trends, determine quality of research, prevent diseases, combat crime, determine real-time roadway traffic conditions, etc."

Why to Store & Analyze Big Data?



More data usually beats better algorithms!
More data allows the data to speak for itself!

Use Cases of Big Data

- Identify customers who are most important
- Identity the best time to perform maintenance based on the usage patterns
- Analyzing your brands reputation by analyzing social media posts
- Click stream analysis to personalize customer’s buying experience

Types of Data

- **Structured Data** – Data organized into entities that have a defined format.
  - Realm of RDBMS / Transaction data
- **Semi-Structured Data** – There may be a schema, but often ignored; schema is used as a guide to the structure of the data.
  - XML, JSON, Emails
- **Unstructured Data** – Doesn’t have any particular internal structure (raw data).
  - Facebook updates, Tweets on Twitter, Reviews, Web Logs, Videos, Documents, Images, etc.
  - MapReduce works well with semi-structured and unstructured data.
- Data is in digital format - Challenge is to make sense out of it. That is termed as Big Data Analytics.

Characteristics of Big Data

4 V's of Big data

- **Volume** – Scale at which data is generated
  - Cannot be stored using traditional methods
- **Velocity** – Data arrives in a continuous stream
  - Multiple, varied source produce data continuously
  - Peaks and bursts unpredictable
  - "Always on": no down time for maintenance or re-orgs
  - No "known users" – unpredictable, unknown patterns/scale
- **Variety** – Different forms of data
  - Big data is usually not structured; structure not known in advance; structure not controlled by consumer
  - May not always be in text format (more than just binary)
- **Veracity** – Various levels of data uncertainty and reliability
  (managing the reliability and predictability of imprecise data types).

Challenges of Big Data

- PB datasets were rapidly becoming the norm, and the trends were clear: our ability to store and process the data was fast overwhelming!
- Problems caused by massive amounts of data, especially data that contain a mixture of complex, unstructured and structured information, which does not lend itself well to being placed in RDBMS.
- But this type of data has a uniquely different characteristic than the transactional or the "customer order" data : "write once read many (WORM)"
- **The Bottleneck was in technology**
  - New architecture, algorithms, techniques were needed
- **Also in technical skills**
  - Experts in using the new technology and dealing with big data

Traditional Large-Scale Computation

- Traditionally, computation has been processor-bound
  - Relatively small amounts of data
  - Significant amount of complex processing performed on that data
- When the data started to grow, the primary push was to increase the computing power of a single machine – Faster processor, more RAM (this idea was there for decades)
- Is it a good idea?
- If the computation power and storage size doubled, then the cost will increase exponentially.
- **A new distinct approach is needed** that might run counter to traditional models of computing.

- In Pioneer days, people used to use oxen for heavy pulling.

- And when one ox couldn't pull the load, what did they do?

They didn't try to grow a larger ox!!

- We shouldn't be trying for bigger computers, but for more systems of computers.

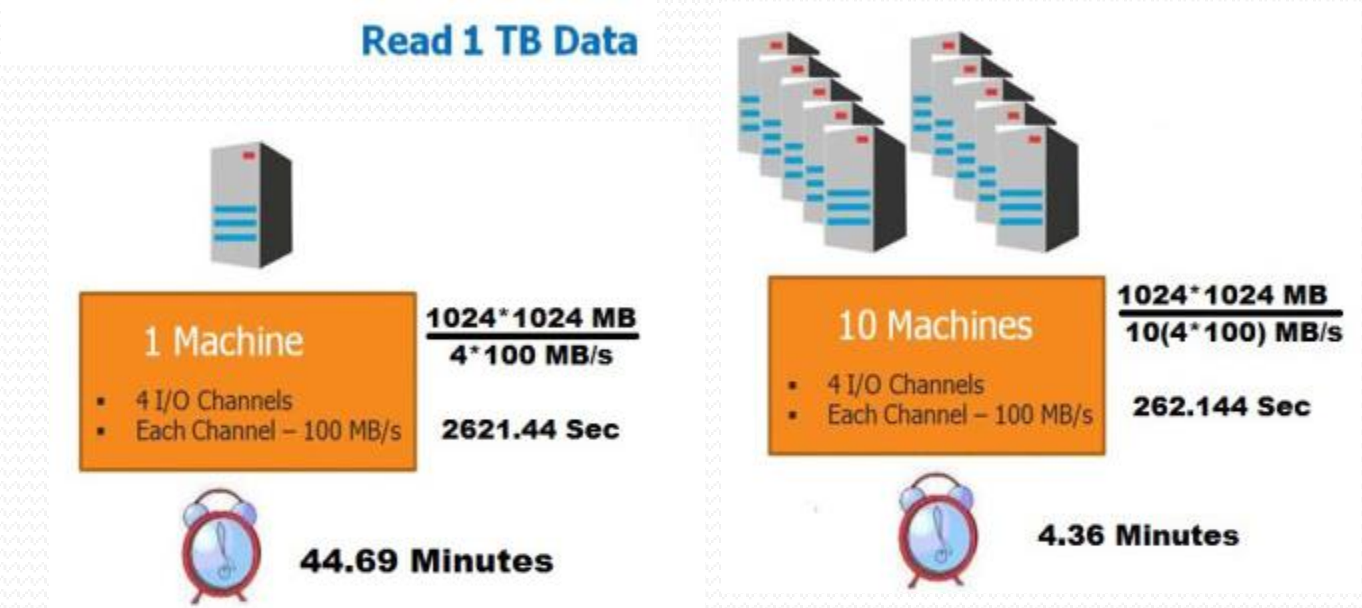
Evolution of Distributed Systems

- A machine with 4 times the power of a standard PC costs a lot more than putting 4 such standard PCs in a cluster.

Problem - Read 1 TB data

(a) 1 machine having 4 I/O channels (or 4 hard drives) such that each can read 100 MB/sec.

(b) 10 machines each having 4 I/O channels (or 4 hard drives) such that each can read 100 MB/sec.



I/O speed is the challenge; not the storage capacity.
So we need to distribute data among many nodes (machines).

Distributed Systems: Data Storage

- Typically, data for a distributed system is stored on a SAN (Storage Area Network)
- At compute time, data is copied to the compute nodes (app servers) from the SAN and then computed answer is written back to the SAN - Fine for relatively limited amounts of data
- Getting the data to the processors becomes the bottleneck
  - Quick calculation
    - Typical disk data transfer rate: 75MB/sec
    - Time taken to transfer 100GB of data to the processor: approx 22 minutes!
      - Assuming sustained reads
      - Actual time will be worse, since most servers have less than 100GB of RAM available.
- So was the traditional distributed system sufficient to handle big data?

Distributed Systems: Problems

- Programming for traditional distributed systems is complex
  - Huge dependency on n/w & huge bandwidth demands
  - Scaling up and down is not a smooth process
  - Partial failures are difficult to handle
  - A lot of processing power is spent on transporting data
  - Data synchronization is required during exchange

- Ken Arnold, CORBA designer:
  - *"Failure is the defining difference between distributed and local programming, so you have to design distributed systems with the expectation of failure"*
    - Developers spend more time designing for failure than they do actually working on the problem itself!

Reality: programmer shoulders the burden of managing concurrency

Key Ideas (Principles)

1. **Scale “out”, not “up”**
   - Large number of commodity low-end servers are preferred over a small number of high-end servers.
2. **Assume failures are common and find remedy**
   - At warehouse scale, failures are not only inevitable, but commonplace.
3. **Move processing to the data**
4. **Hide system-level details from the application developer**
5. **Process data sequentially, avoid random access**
6. **Seamless scalability**

Process Data Sequentially, Avoid Random Access

- Data-intensive processing by definition means that the relevant datasets are too large to fit in memory and must be held on disk.
- Seek times for random disk access are fundamentally limited by the mechanical nature of the devices: read heads can only move so fast and platters can only spin so rapidly.
- As a result, it is desirable to avoid random data access, and instead organize computations so that data is processed sequentially.

Seamless Scalability

- For data-intensive processing, it goes without saying that scalable algorithms are highly desirable.
- Increasing resources should support a proportional increase in load capacity
  - Given twice the amount of data, the same algorithm should take at most twice as long to run, all else being equal
- Given a cluster twice the size, the same algorithm should take no more than half as long to run. Adding load to the system should result in a graceful decline in performance of individual jobs and not the failure of the system!

Big Credits

- Tackling large-data problems requires a distinct approach that sometimes runs counter to traditional models of computing.
- All of these ideas have been discussed in the computer science literature for some time (some for decades), and Hadoop - MapReduce is certainly not the 1st to adopt these ideas.
- Google deserves tremendous credit for pulling these various threads together and demonstrating the power of these ideas on a scale previously unheard of.
- Then Yahoo also deserves a lot of credit for starting the open-source Hadoop project which has made MapReduce accessible to everyone and created the vibrant s/w ecosystem that flourishes today.

Solution – Distributed File System (DFS)

- Filesystems that manage the storage across a network of machines are called distributed file systems.
- Machines are physically located at different places
- Logically, there is only one file system. So we can read data in parallel into multiple machines.

The datacenter *is* the computer!

Google File System (GFS)

- At Google, big data operations are run on a special file system called GFS that is highly optimized for processing Big Data.
- GFS is unique to Google and is not for sale. But it served as a model for file systems for organizations with similar needs.
- Google published a research paper in 2003 about Google File System (GFS).
- MapReduce paper published in 2004.

Hadoop History

- Started as a subproject of Apache Nutch (2002)
  - Nutch's job is to index the web and expose it for searching but the problem was how to scale it to the billions of pages on web! GFS was the answer!
- Doug Cutting (creator of Apache Lucene) and Nutch team implemented Google's frameworks in Nutch and moved it out of Nutch later in 2006 and called it Hadoop.
- In 2006, Yahoo! hired Doug Cutting to work on Hadoop with a dedicated team.
  - Based on GFS, Yahoo! created Hadoop Distributed File System (HDFS).
- In 2008, Hadoop became Apache top level project.

Naming Conventions

- Doug Cutting drew inspiration from his family.
  - **Lucene**: Doug’s wife’s middle name
  - **Nutch**: A word for “meal” that his son used as a toddler
  - **Hadoop**: Yellow stuffed elephant named by his son.

Google Origins

2003

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

2004

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

2006

Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber

{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com

Google, Inc.

What is Hadoop?

Definition

Hadoop is a framework that allows for storage and distributed processing of large data sets across clusters of commodity hardware using a simple computing model called MapReduce to retrieve and analyze data.

- **Scalable fault-tolerant distributed system for Big Data:**
  - Data Storage
  - Data Processing
  - A virtual Big Data machine
  - Borrowed concepts/Ideas from Google; Open source under the Apache license

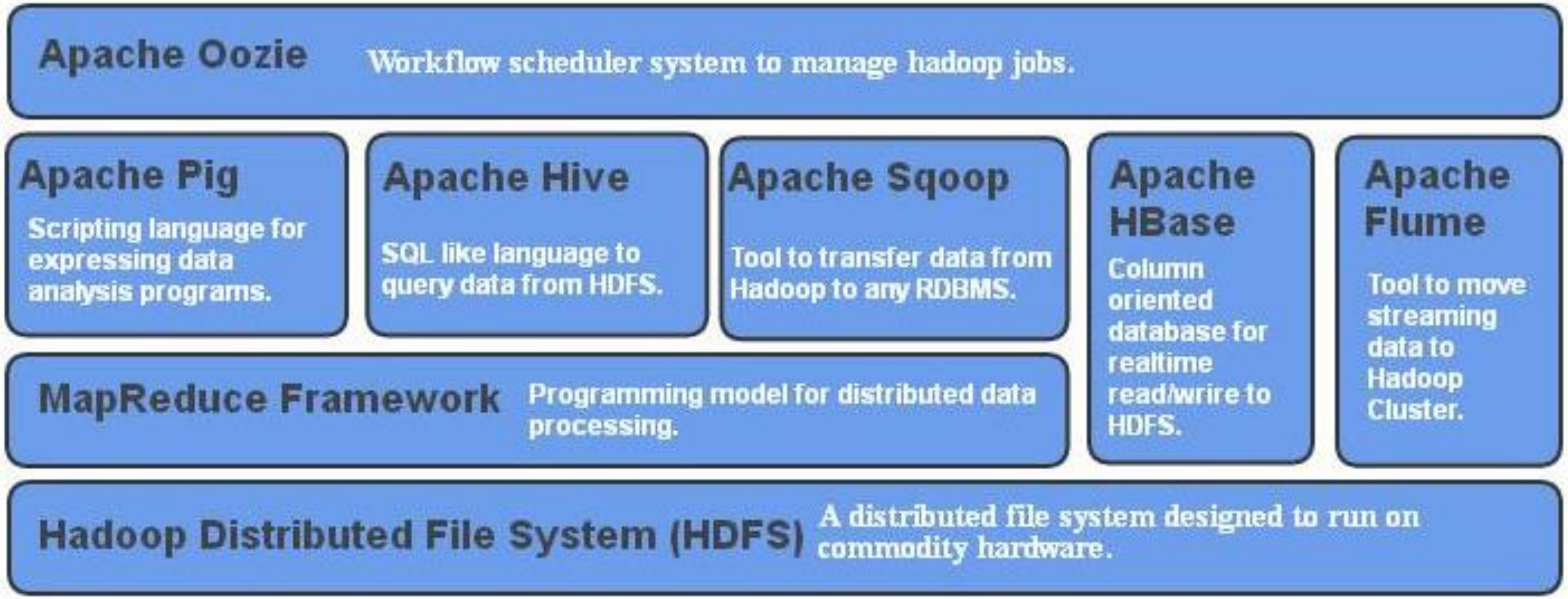
Big Data != Hadoop

Hadoop != Database

Apache Hadoop

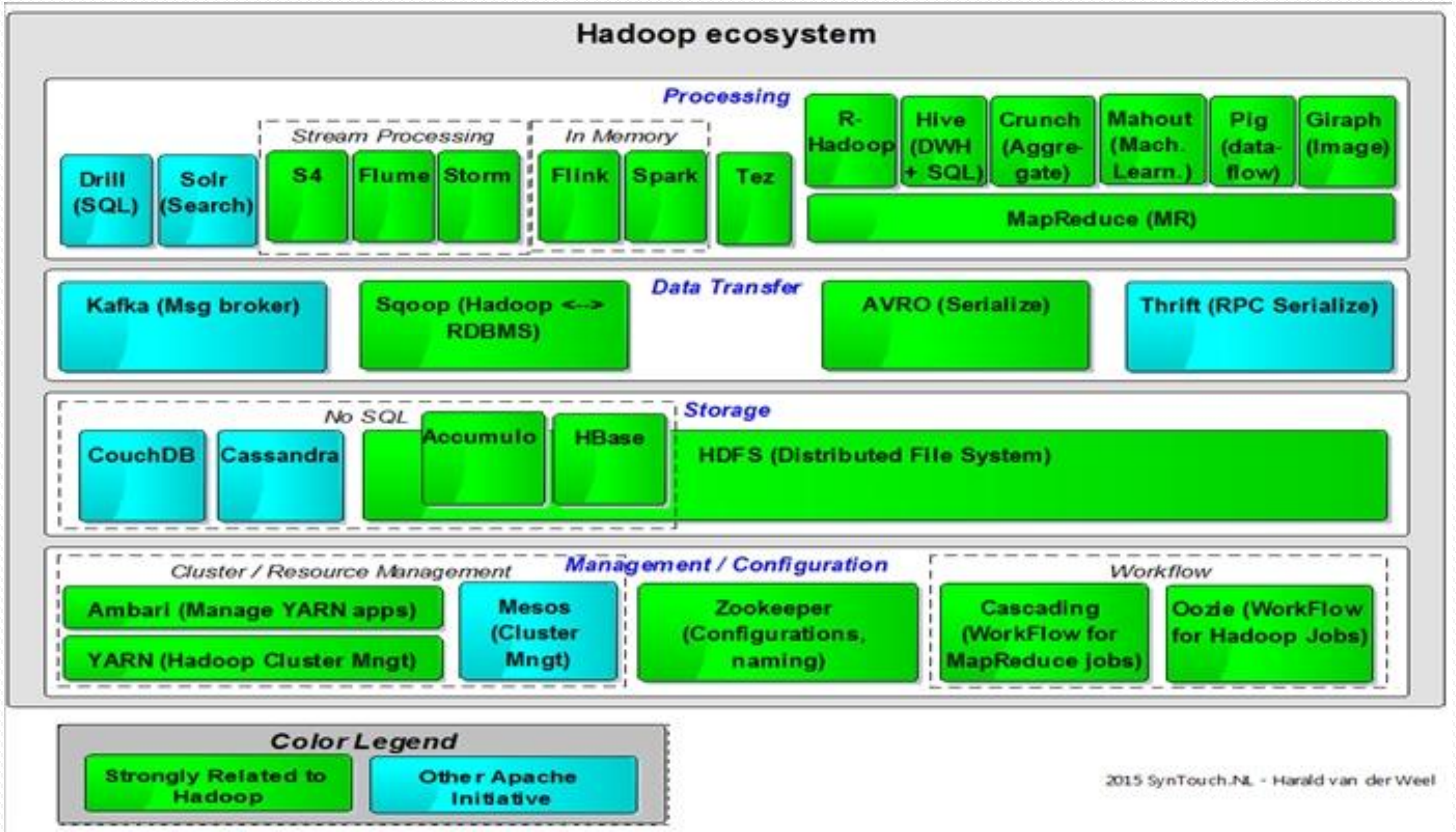
- Hadoop is written in Java and is supported on all major platforms.
- Designed to answer the question: “How to store and process big data with reasonable cost and time?”
- **Hadoop consists of two core components:**
  - The Hadoop Distributed File System (HDFS)
  - MapReduce Software Framework
- There are many other projects based around core Hadoop
  - Often referred to as the ‘Hadoop Ecosystem’
  - Pig, Hive, HBase, Flume, Oozie, Sqoop, Zookeeper etc.

Hadoop Ecosystem



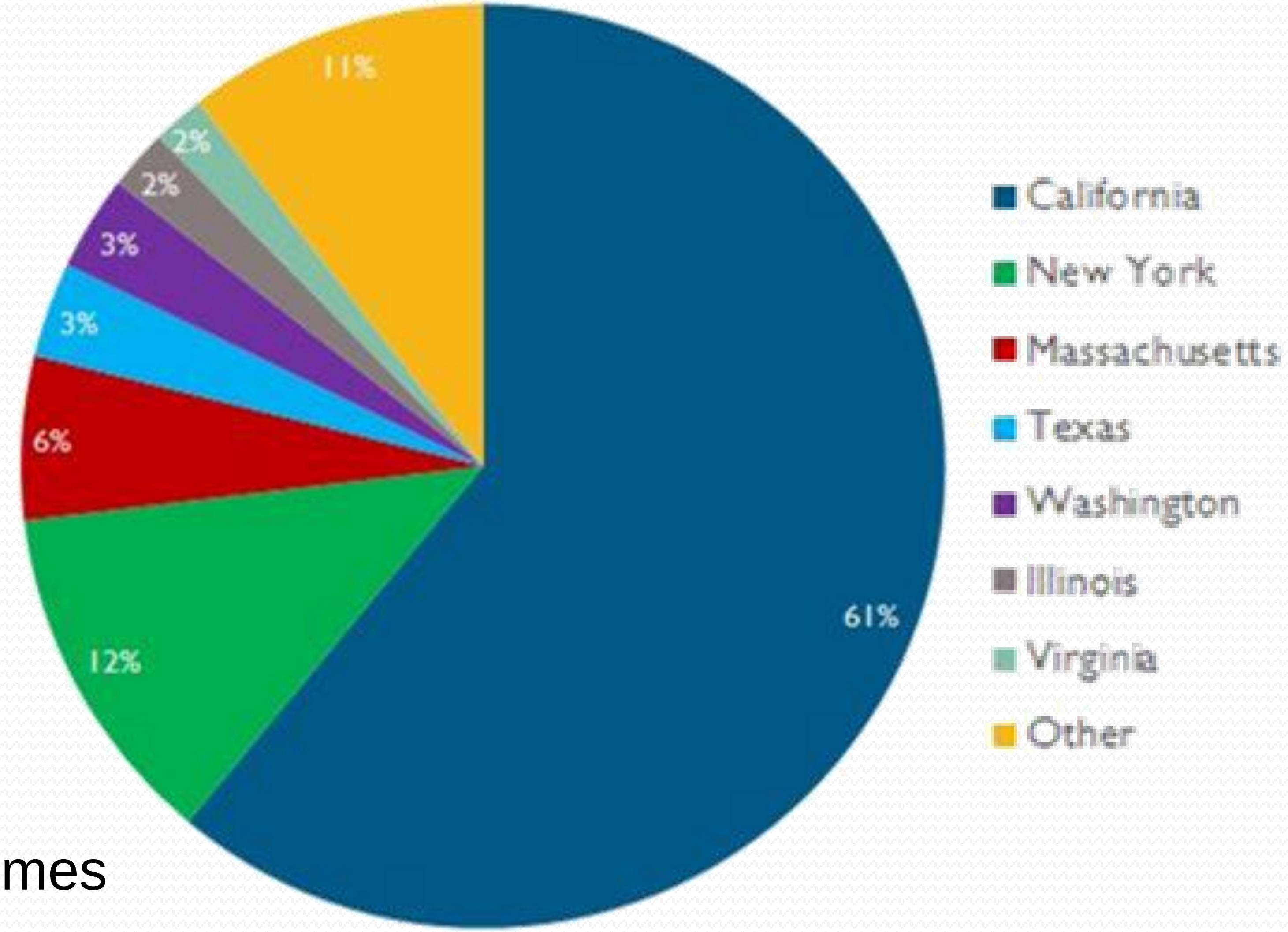
The Ever-Growing Hadoop Ecosystem

Hadoop Framework Tools



Who uses Hadoop?

- Amazon
- Facebook
- Google
- IBM
- AOL
- Last.fm
- PowerSet
- Veoh
- Yahoo!
- LinkedIn
- eBay
- New York Times



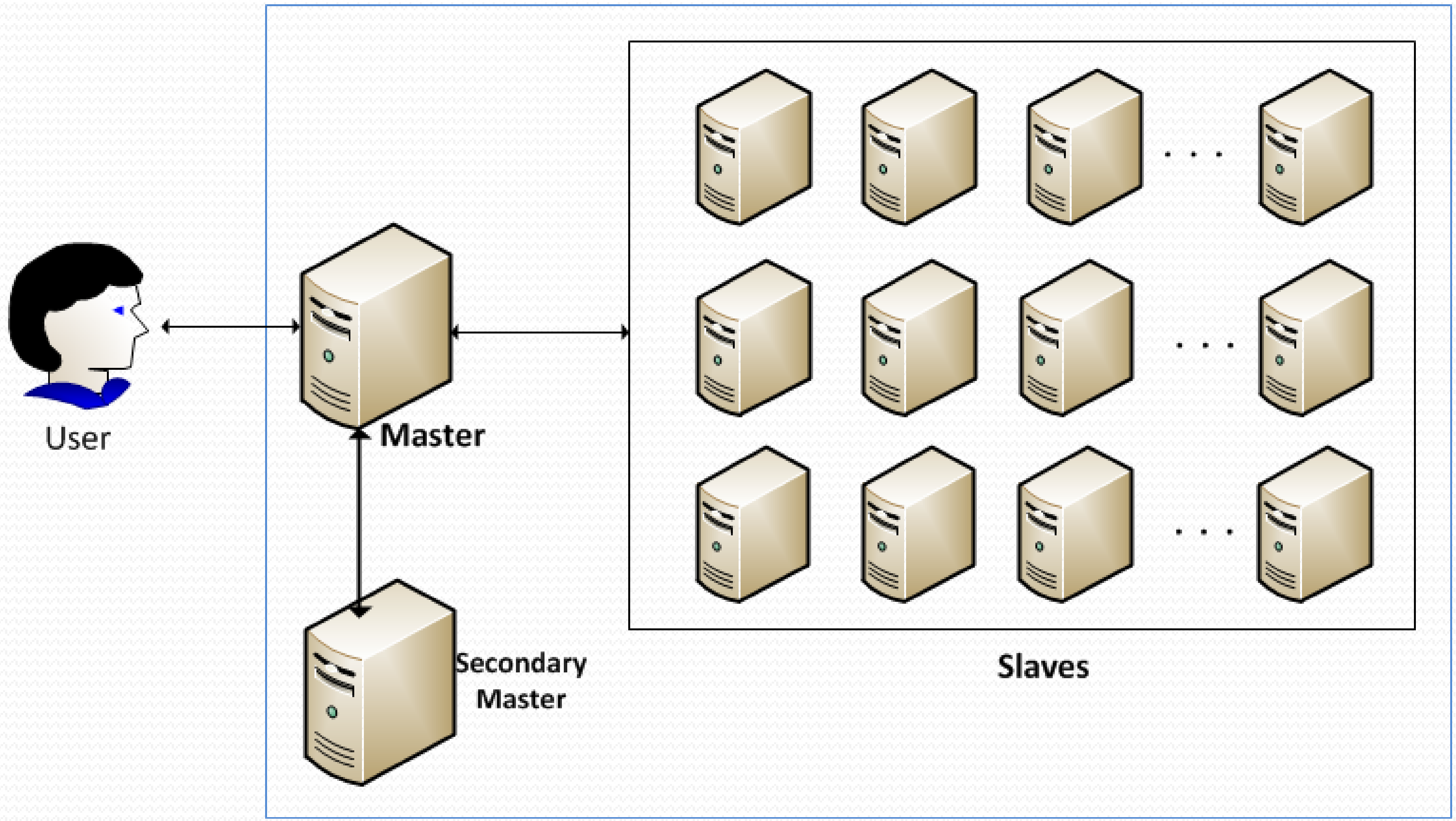
Common Hadoop Distributions

- Hadoop is an Apache open source project, and regular releases of the software are available for download directly from the Apache project's website.
- You can either download and install Hadoop from the website or use a quickstart virtual machine from a commercial distribution, which is usually a great starting point if you're new to Hadoop and want to quickly get it up and running.
  - Apache
  - Cloudera
  - Hortonworks
  - MapR
  - AWS
  - Microsoft (HDInsight)

Hadoop Cluster

- A set of machines running HDFS and MapReduce is known as a Hadoop Cluster.
- Individual machines are known as nodes.
- A cluster can have as few as one node, as many as several thousand nodes.
- More nodes = better performance!
- **Data Locality** - Hadoop tries to co-locate the data with the compute nodes, so data access is fast because it's local.

Hadoop Cluster



Hadoop Component: HDFS

- Hadoop comes with a distributed filesystem called HDFS, which stands for *Hadoop Distributed File System* and it is Hadoop's flagship filesystem.
- HDFS is responsible for storing large data sets on the cluster and it can be built out of commodity hardware.
- Streaming data access patterns
  - Write once and read many times
  - Where getting the entire data faster is more important than getting a specific record
- HDFS provides Java API for applications to use
- HTTP browser can be used to browse the files of HDFS instance

HDFS is a filesystem designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.

Very large files:

- Files that are usually in TBs, PBs and more.

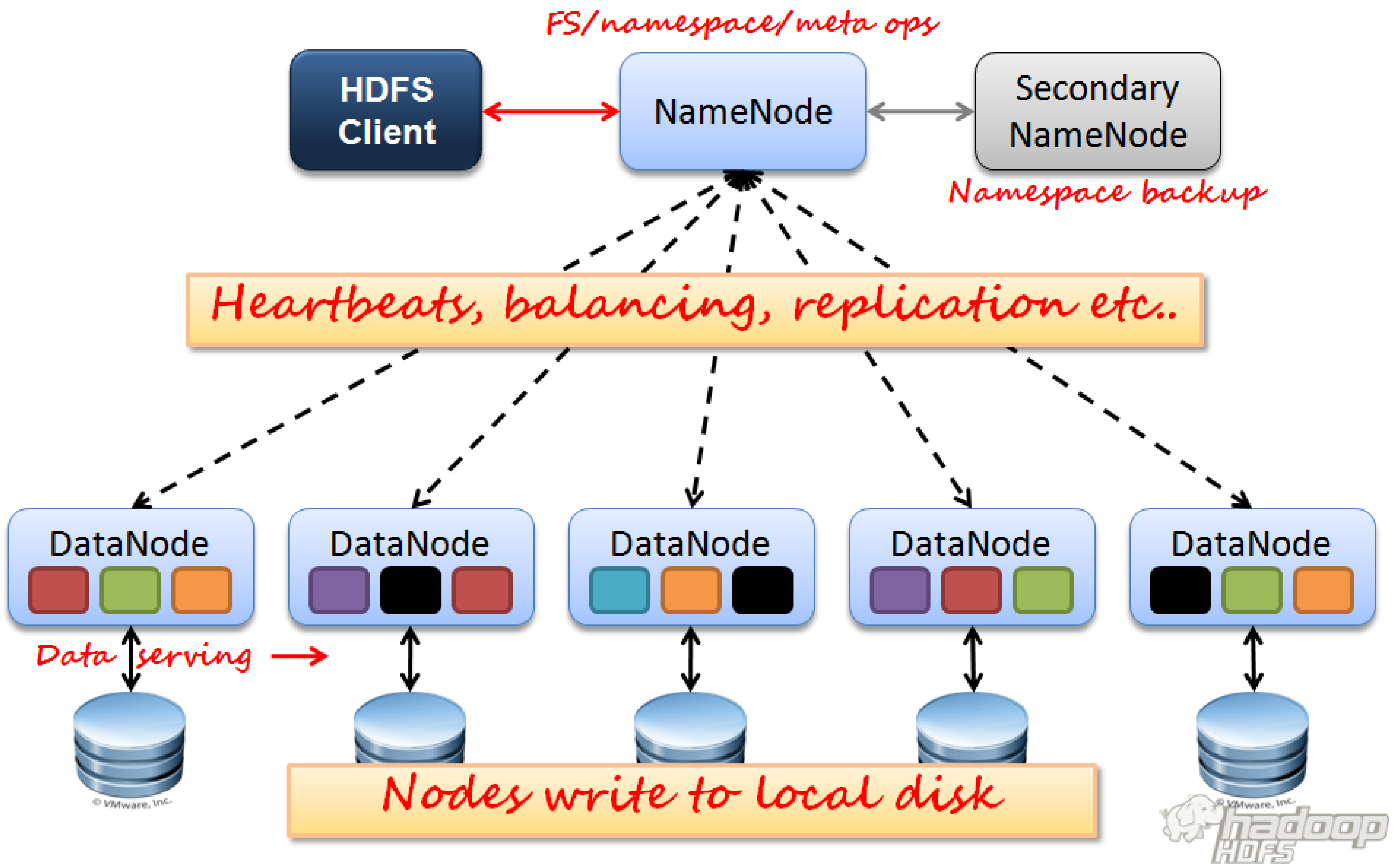
Streaming data access:

- WORM – dataset is generated or copied from source and then various analyses are performed on that dataset over time. Time to read the whole dataset is more important than the latency in reading the first record.

Commodity Hardware:

- Commonly available hardware that can be obtained from multiple vendors which doesn’t have high quality or high availability.
- Chance of node failure across the cluster is high.
- HDFS is designed to carry on working without a noticeable interruption to the user in the face of such failures.

HDFS Architecture



HDFS Architecture

- **Master-Slave** architecture
  - master maintains the file namespace (metadata, directory structure, file to block mapping, location of blocks etc.)
  - slaves manage the actual data blocks
- In GFS, the master is called the GFS master, and the slaves are called GFS chunkservers.
- In Hadoop, the same roles are filled by **NameNode** and **DataNodes**, respectively.

HDFS Concepts - Blocks

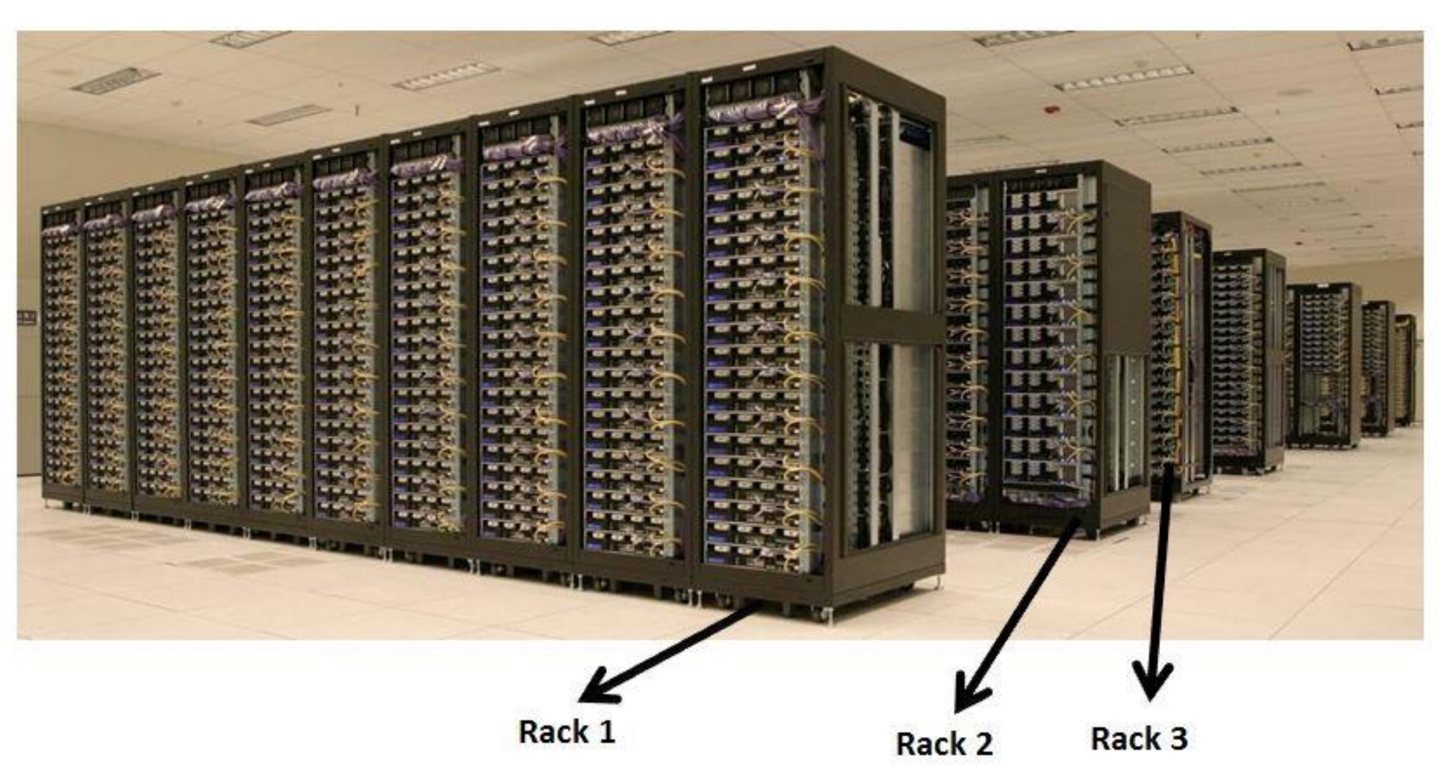
- Like in a filesystem for a single disk, files in HDFS are broken into block-sized chunks, which are stored as independent units.
- **A typical block size is 128 MB** (was 64 MB in Hadoop 1.x).
- Unlike a filesystem for a single disk, a file in HDFS that is smaller than a single block does not occupy a full block's worth of underlying storage.
  - For example, a 1 MB file stored with a block size of 128 MB uses 1 MB of disk space, not 128 MB.
- Blocks are themselves stored on standard single-machine file systems, so HDFS lies on top of the standard OS stack (e.g., Linux).

Why is a block in HDFS so large compared to disk block size?

Fault Tolerance using Data Replication

- In HDFS each file is a sequence of blocks.
- All blocks in the file except the last one are of the same size.
- **Blocks are replicated for fault tolerance.**
- Block size and number of replicas are configurable for files.
- **Replication factor is 3 by default.**
- NameNode tries to place replicas of blocks on multiple racks for improved fault tolerance.

Yahoo's Hadoop Cluster



All server configurations are 512GB RAM, 30TB storage and 16 cores, Ubuntu Linux 14.04LTS server.

Hadoop let's you interact with a cluster, not a bunch of machines!

Rack Awareness

- Usually Hadoop clusters of more than 30-40 nodes are configured in multiple racks.
- Communication between two DataNodes on the same rack is efficient than the same between two nodes on different racks.
- Hadoop lets the cluster administrators decide which rack a node belongs to through configuration scripts.
- Each node runs the script to determine its rack-id.
- A default installation assumes all the nodes belong to the same rack.

Rack Awareness

- In large clusters of Hadoop, in order to improve network traffic while reading/writing HDFS files, NameNode chooses DataNodes which are on the same rack or a near-by rack to read/write request.
- NameNode obtains this rack info by maintaining rack-ids of each DataNode.
- This concept of choosing closer DataNodes based on racks info is called Rack Awareness in Hadoop.

Replica Placement Policy

- While placing new blocks, NameNode considers various parameters before choosing the DataNodes for placing these blocks.

- On multiple rack cluster, block replications are maintained with the following policy: (when the replication factor is 3)
  - Only one replica is placed on one node.
  - And no more than 2 replicas are placed in the same rack.

- Number of racks used for block replication should always be less than total no of block replicas.

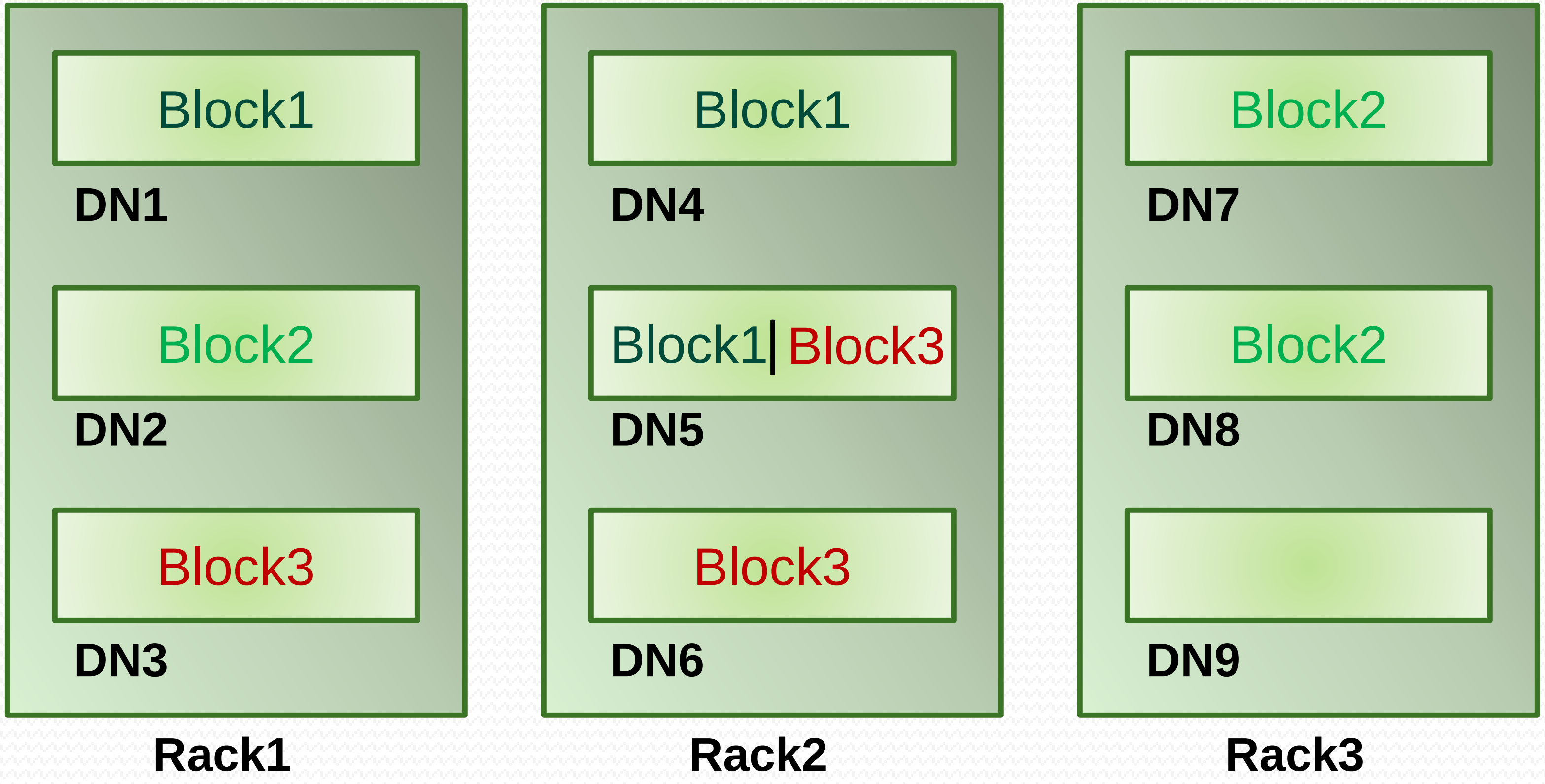
Replica Placement Policy

- Policy to keep one of the replicas of a block on the same node as the node that is writing the block.
- One of the replicas is usually placed on the same rack as the node writing to the file so that cross-rack network I/O is reduced.
- Need to spread different replicas of a block across the racks so that cluster can survive loss of whole rack.
- Spread HDFS data uniformly across the DataNodes in the cluster.

- Note: There is no relationship between replicas of different blocks of the same file as far as their location is concerned. Each block is allocated independently.

Replica Placement Strategy

Three racks, each rack has 3 data nodes, three data blocks (1,2,3)



Necessity For Re-replication

- The necessity for re-replication may arise due to:
  - A DataNode may become unavailable,
  - A replica may become corrupted,
  - A hard disk on a DataNode may fail, or
  - The replication factor on the block may be increased.

Rebalancer Utility

- HDFS data might not always be placed uniformly across the DataNodes. One common reason is addition/deletion of new DataNodes to/from an existing cluster.
- In such unbalanced cluster, data read/write requests become very busy on some DataNodes and some DataNodes are under utilized.
- HDFS provides a tool for administrators that analyzes block placement and rebalances data across the DataNodes if there's any imbalance.
  - `$ hdfs balancer`
- If a Rebalancer is triggered, NameNode will scan entire DataNode list and Spread HDFS data uniformly across the DataNodes in the cluster.

HDFS is Good for...

- **Storing large files**
  - Terabytes, Petabytes, etc...
- **Streaming data**
  - Write once and read-many times patterns
  - Optimized for streaming reads rather than random reads
- **“Cheap” Commodity Hardware**
  - No need for super-computers, use less reliable commodity hardware

HDFS is not so good for…

- **Low-latency data access**

  (*latency*: time interval between data request and data availability)

  - HDFS is optimized for delivering a high throughput of data, and this may be at the expense of latency.
  - Applications that require low-latency access to data, in the tens of milliseconds range, will not work well with HDFS.
    - HBase is a better choice for low-latency access.

- **Multiple writers, arbitrary modifications**
  - Files in HDFS may be written to by a single writer. Writes are always made at the end of the file, in append-only fashion.
  - There is no support for multiple writers or for modifications at arbitrary offsets in the file.

HDFS is not so good for…

- **Lots of small files**
  - HDFS is better for millions of large files instead of billions of small files!
  - Because the NameNode holds filesystem metadata in memory, the limit to the number of files in a filesystem is governed by the amount of memory on the NameNode.
  - As a rule of thumb, each file or directory, and block metadata takes about 150 bytes. So, for example, if you had two million files, each taking one block, you would need at least 600 MB of memory.
  - Although storing millions of files is feasible, billions is beyond the capability of current hardware.

What Hadoop Provides:

- A reliable shared storage and analyses system (HDFS & MapReduce)
- The ability to read and write data in parallel to or from multiple disks.
- Enables applications to work with thousands of nodes and petabytes of data.
- A free license

Hadoop Benchmarks

- *New York Times* used Amazon's EC2 compute cloud to crunch through 4 TBs of scanned archives from the paper, converting them to PDFs for the Web in less than 24 hours using 100 machines with Hadoop.
- In April 2008, Hadoop broke a world record to become the fastest system to sort an entire TB of data. Running on a 910-node cluster, Hadoop sorted 1 TB in 209 seconds (just under 3.5 minutes), beating the previous year's winner of 297 seconds.
- In November of the same year, Google reported that its MapReduce implementation sorted 1 TB in 68 seconds.
- Then, in April 2009, it was announced that a team at Yahoo! had used Hadoop to sort 1 TB in 62 seconds.
- The trend since then has been to sort even larger volumes of data at ever faster rates. In the 2014 competition, a team from Databricks were joint winners of the Gray Sort benchmark. They used a 207-node Spark cluster to sort 100 TBs of data in 1,406 seconds, a rate of 4.27 TBs per minute.

Future Growth

- Hadoop is now more than 10 years old. In that decade, Hadoop has gone from being the hopeful answer to Yahoo's search-engine woes to a general-purpose computing platform that's poised to be the foundation for the next generation of data-based applications.

- The potential for Hadoop is huge. Markets and Markets Research projects that the Hadoop market will grow from $6.7 billion in 2016 to $40.7 billion in 2021, an average growth rate of 43.4%.

- Since Cloudera launched in 2008, Hadoop has spawned dozens of startups and spurred hundreds of millions in venture capital investment since 2008.

Review

- Big Data and it’s fast growth
- Need to store and analyze Big Data
- 4 V’s of Big Data
- 6 key ideas behind Hadoop development
- Hadoop history and it’s core components
- HDFS Architecture (NN and DNs)
- HDFS blocks, block replication and replica placement policy
- Default block size in HDFS
- Pros and Cons of HDFS