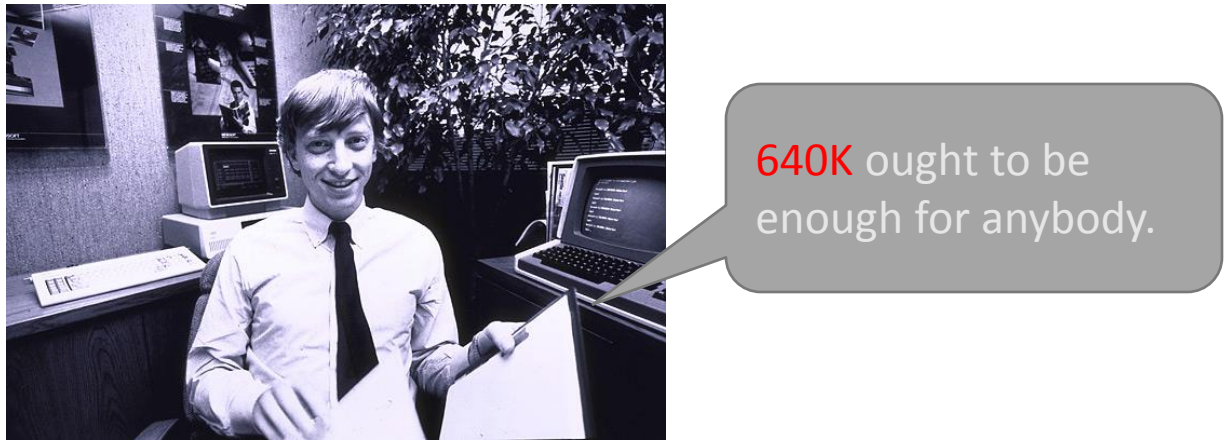
Big Data
What it is and why it matters

Big Data Every Where!

- Lots of data is being collected and warehoused
  - Web data, e-commerce
  - purchases at department/ grocery stores
  - Bank/Credit Card transactions
  - Social Network

How much data?

- Google processes 20 PB a day (2008)
- Wayback Machine has 3 PB + 100 TB/month (3/2009)
- Facebook has 2.5 PB of user data + 15 TB/day (4/2009)
- eBay has 6.5 PB of user data + 50 TB/day (5/2009)
- CERN's Large Hydron Collider (LHC) generates 15 PB a year



Big Data (globally)

facebook

- creates over **30 billion** pieces of content per day
- stores **30 petabytes** of data

twitter

- produces over **90 million** tweets per day

IBM Definition

- Big data is being generated by everything around us at all times. Every digital process and social media exchange produces it. Systems, sensors and mobile devices transmit it.
- Big data is arriving from multiple sources at an alarming velocity, volume and variety. To extract meaningful value from big data, you need optimal processing power, analytics capabilities and skills.

What is Big Data?

IBM Definition

- Every day, we create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone.
- This data comes from everywhere: sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to name a few. This data is big data.

What is Big Data?

Gartner Definition

- Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

What is Big Data?

SAS Definition

- Big data is a term that describes the large volume of data – both structured and unstructured – that inundates a business on a day-to-day basis. But it's not the amount of data that's important. It's what organizations do with the data that matters. Big data can be analyzed for insights that lead to better decisions and strategic business moves.

What is Big Data?

- Big data is changing the way people within organizations work together. It is creating a culture in which business and IT leaders must join forces to realize value from all data. Insights from big data can enable all employees to make better decisions—deepening customer engagement, optimizing operations, preventing threats and fraud, and capitalizing on new sources of revenue.

Competitive advantage

Data is emerging as the world's newest resource for competitive advantage.

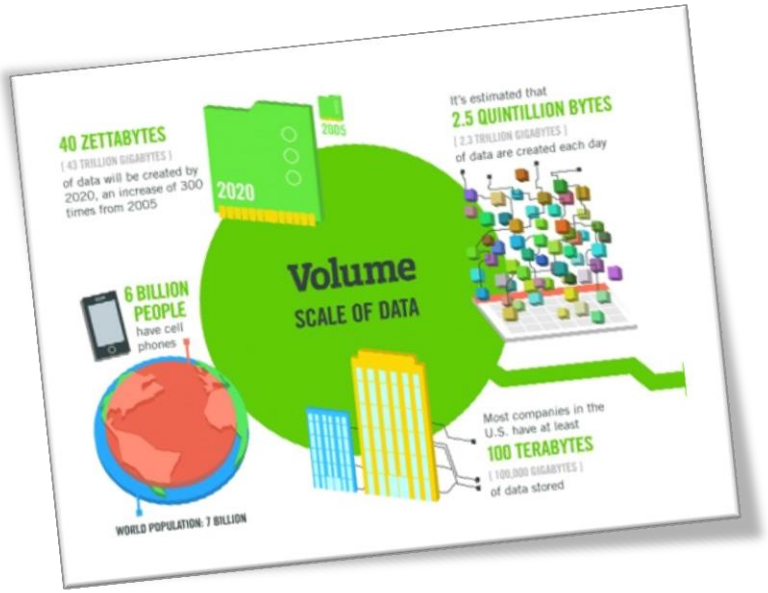
Decision making

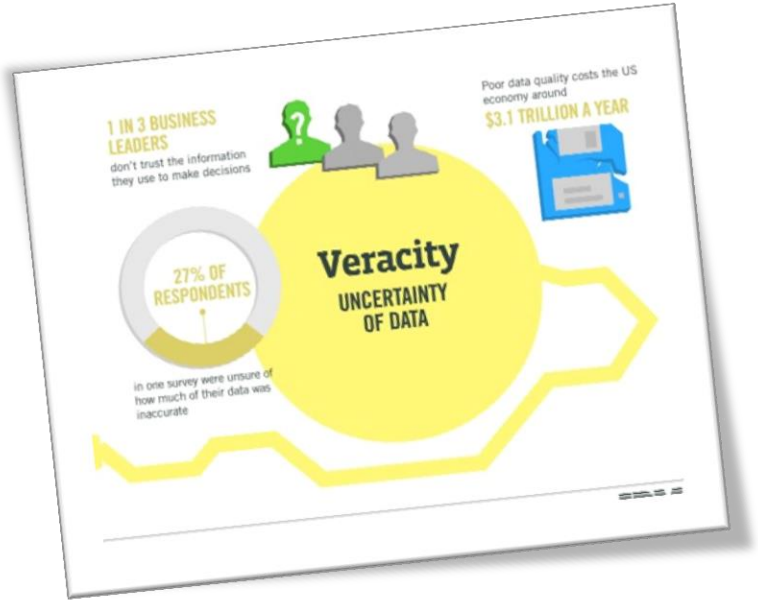
Decision making is moving from the elite few to the empowered many.

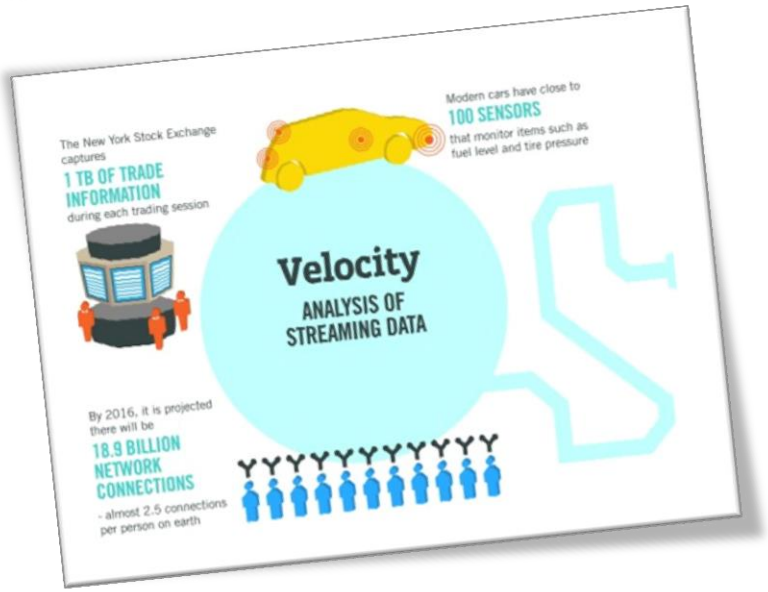
Value of data

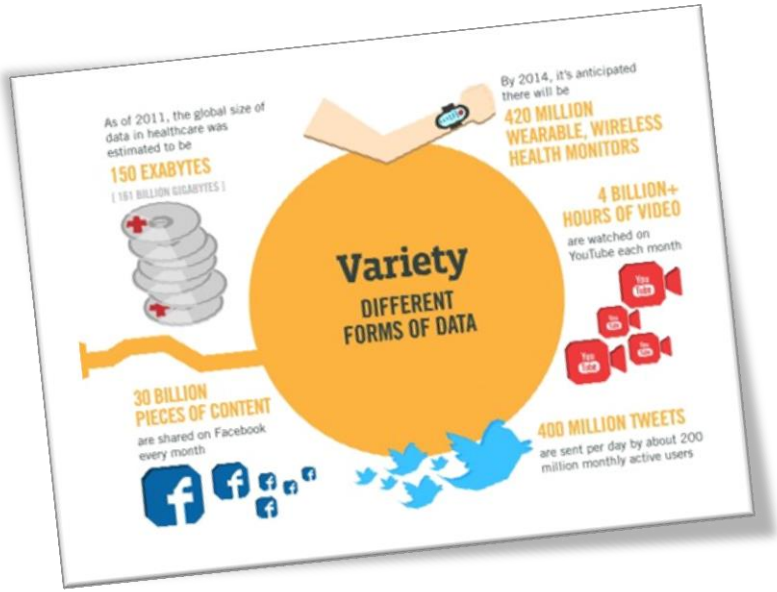
As the value of data continues to grow, current systems won't keep pace.

What is changing in the realm of big data?

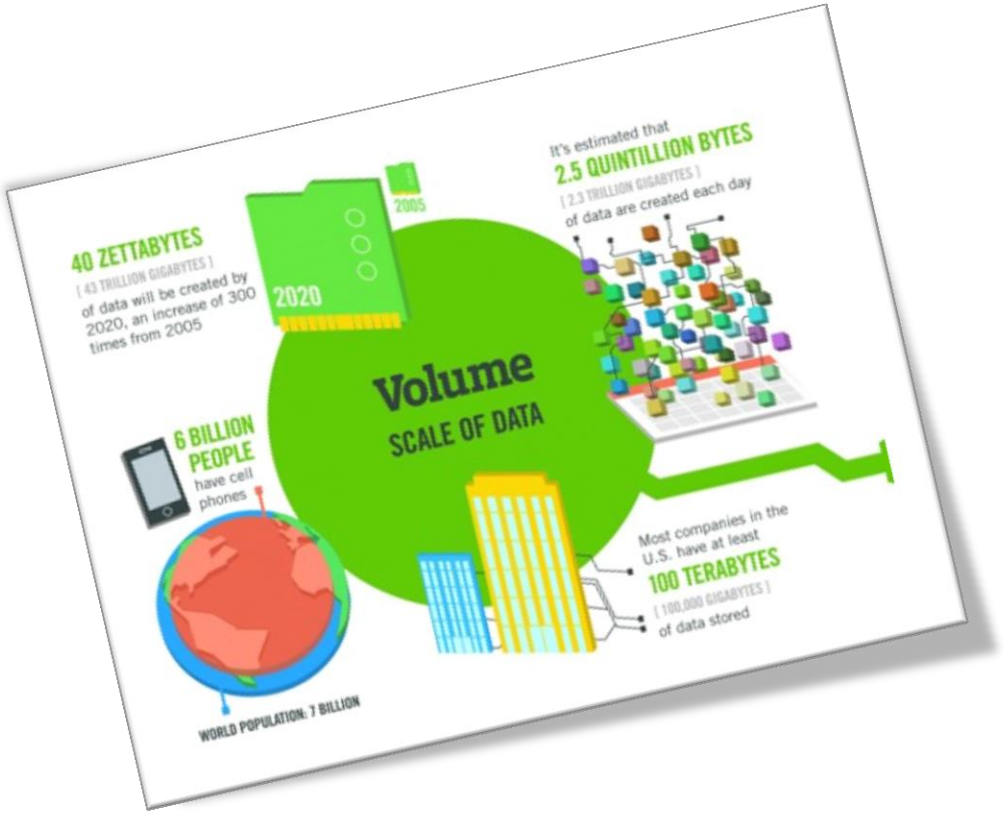








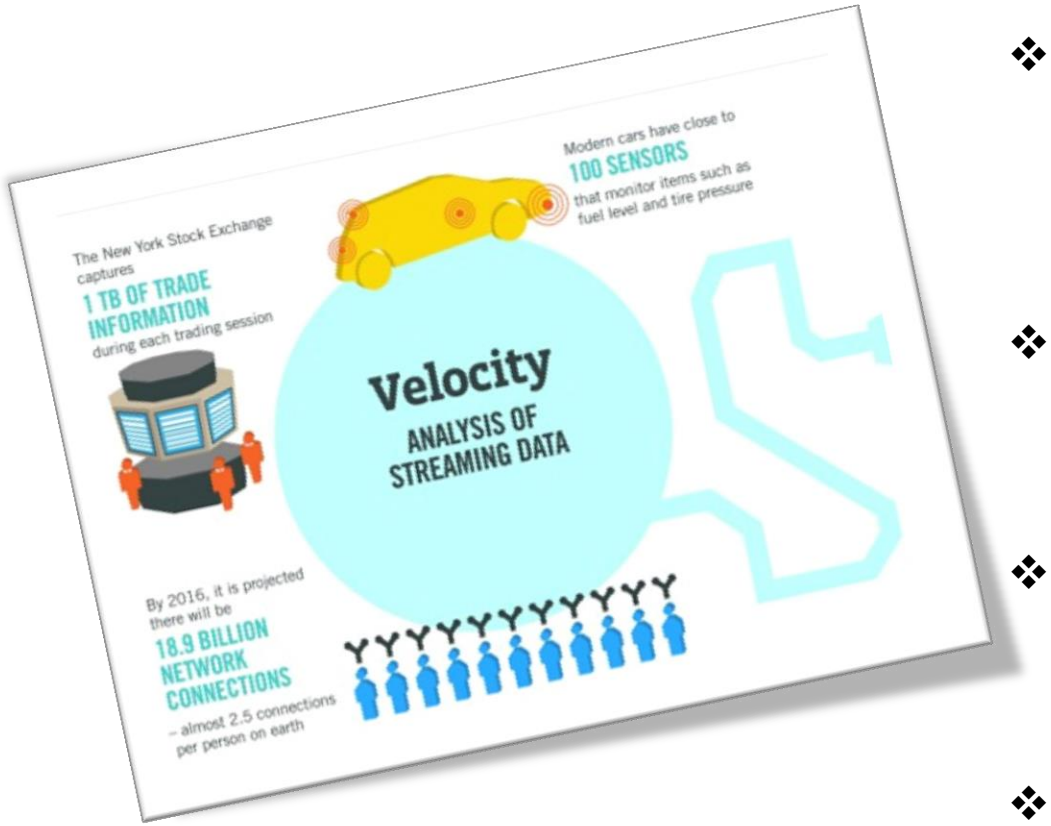
Vs of big data



Volume

- Volume refers to the vast amount of data generated every second.
- Every day, we create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone.
- On Facebook alone we send 10 billion messages per day, click the like button 4.5 billion times and upload 350 million new pictures each and every day.
- Organizations collect data from a variety of sources, including business transactions, social media and information from sensor or machine-to-machine data. In the past, storing it would've been a problem – but new technologies (such as Hadoop) have eased the burden.

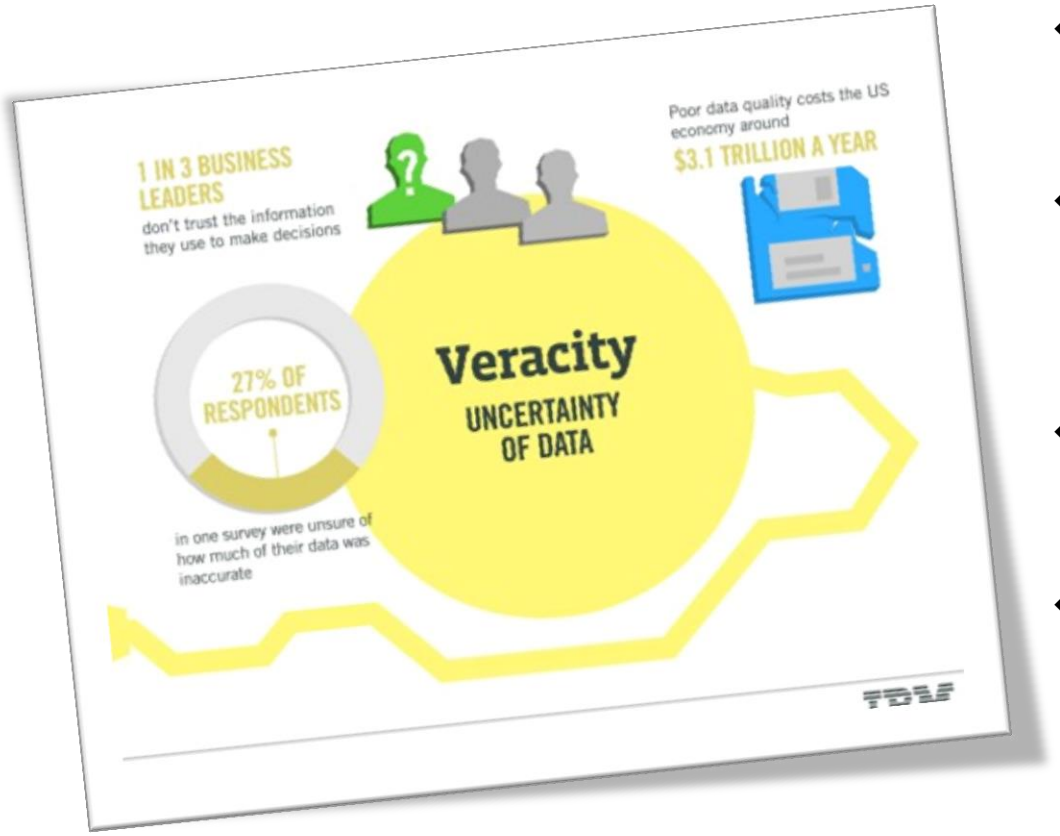
Vs of big data



Vs of big data

Velocity

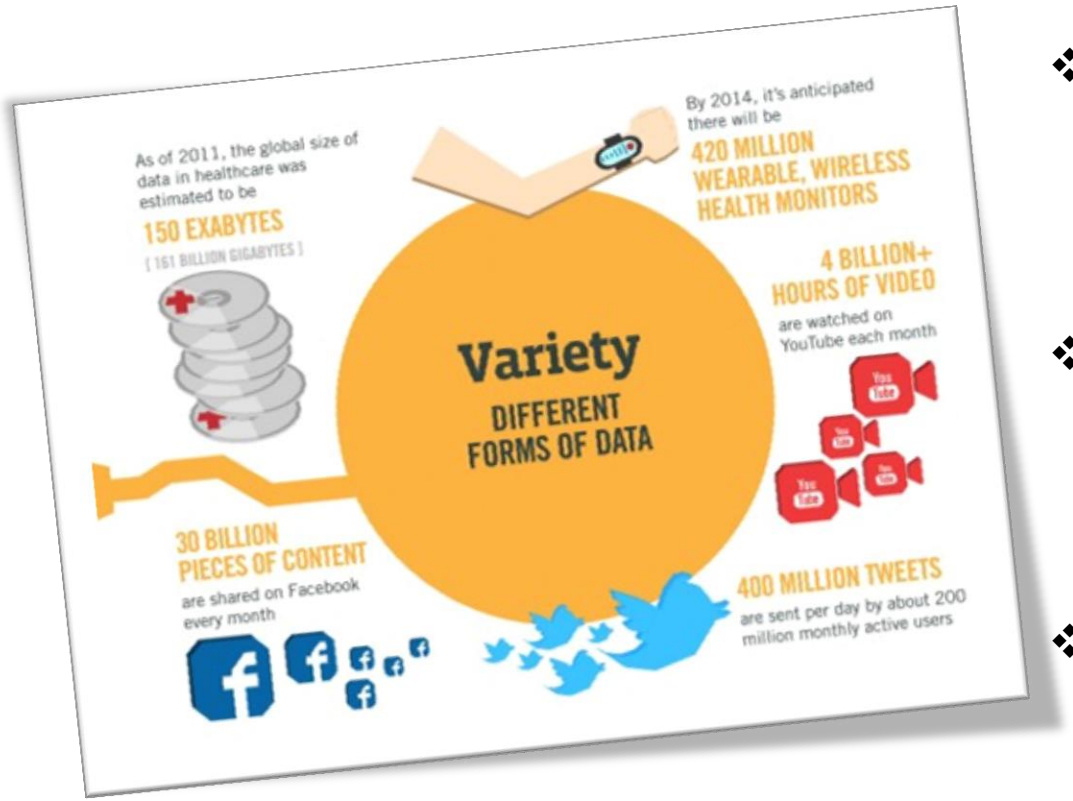
- Data streams in at an unprecedented speed and must be dealt with in a timely manner. RFID tags, sensors and smart metering are driving the need to deal with torrents of data in near-real time.
- Every day, we create 2.5 quintillion bytes of data — so much that 90% of the data in the world today has been created in the last two years alone.
- On Facebook alone we send 10 billion messages per day, click the like button 4.5 billion times and upload 350 million new pictures each and every day.
- Organizations collect data from a variety of sources, including business transactions, social media and information from sensor or machine-to-machine data. In the past, storing it would’ve been a problem – but new technologies (such as Hadoop) have eased the burden.



Veracity

- Veracity refers to the messiness or trustworthiness of the data. The data is virtually worthless if it's not accurate.
- With many forms of big data, quality and accuracy are less controllable, for example Twitter posts with hashtags, abbreviations, typos and colloquial speech.
- Big data and analytics technology now allows us to work with these types of data.
- The volumes often make up for the lack of quality or accuracy.

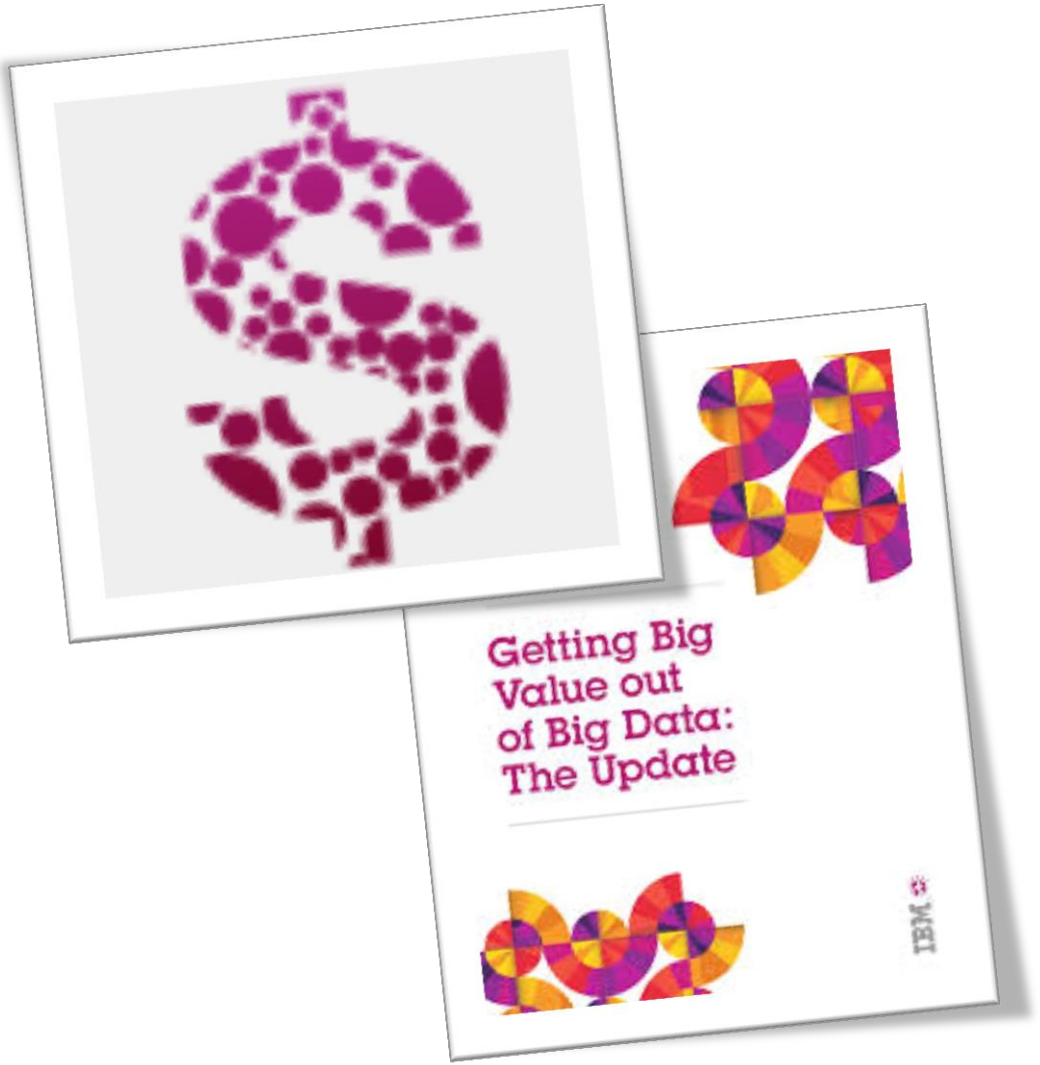
Vs of big data



Variety

- Variety refers to the different types of data we can now use. In the past we focused on structured data that neatly fits into tables or relational databases such as financial data (for example, sales by product or region).
- 80 percent of the world's data is now unstructured and therefore can't easily be put into tables or relational databases—think of photos, video sequences or social media updates.
- With big data technology we can now harness differed types of data including messages, social media conversations, photos, sensor data, video or voice recordings and bring them together with more traditional, structured data.

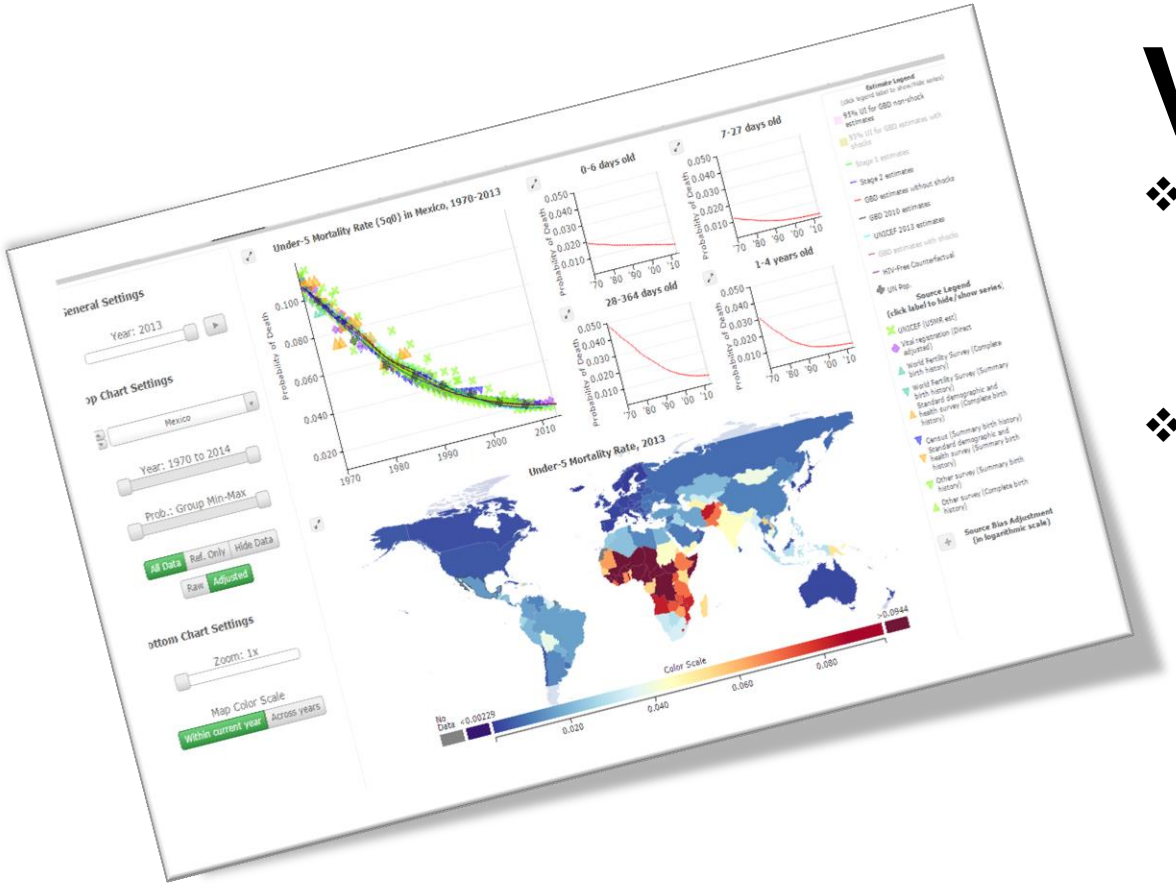
Vs of big data



Value

- It is easy to fall into the buzz trap and embark on big data initiatives without a clear understanding of the business value it will bring.
- Data in itself is not valuable at all. The value is in the analyses done on that data and how the data is turned into information and eventually turning it into knowledge.
- Value refers to our ability turn our data into value. It is important that businesses make a case for any attempt to collect and leverage big data.
- The value is in how organizations will use that data and turn their organization into an information-centric company that relies on insights derived from data analyses for their decision-making.

Vs of big data



Visualization

- Once data has been processed, you need a way of presenting the data in a manner that's readable and accessible- this is where visualization comes in.

- Visualizations can contain dozens of variables and parameters- a far cry from the x and y variables of your standard bar chart- and finding a way to present this information that makes the findings clear is one of the challenges of Big Data.

Vs of big data

- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), …
- Streaming Data
  - You can only scan the data once

Type of Data

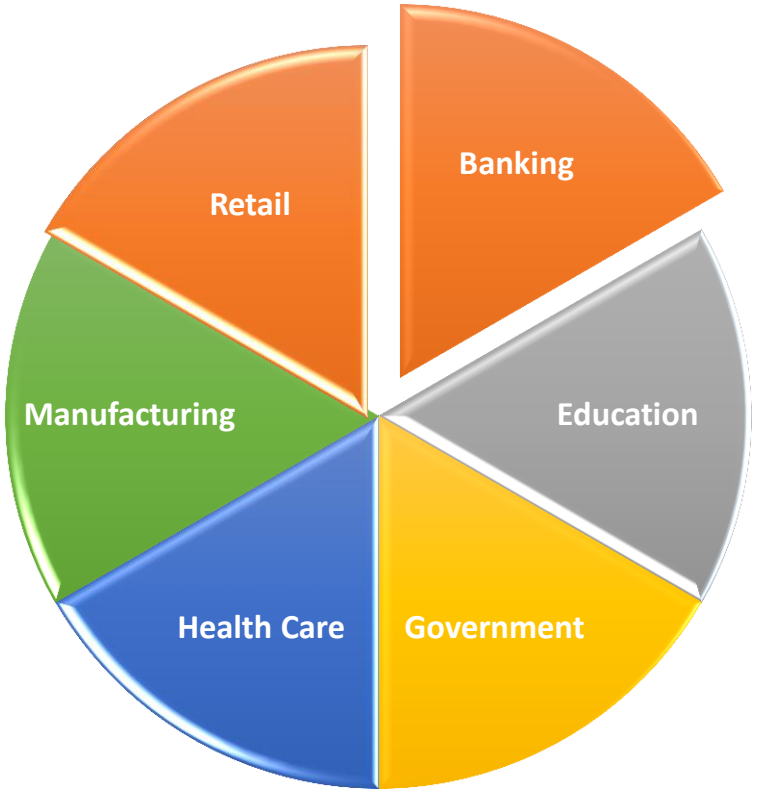
What to do with these data?

- Aggregation and Statistics
  - Data warehouse and OLAP
- Indexing, Searching, and Querying
  - Keyword based search
  - Pattern matching (XML/RDF)
- Knowledge discovery
  - Data Mining
  - Statistical Modeling

Why Is Big Data Important?

- The importance of big data doesn’t revolve around how much data you have, but what you do with it. You can take data from any source and analyze it to find answers that enable
  - cost reductions
  - time reductions
  - new product development and optimized offerings
  - smart decision making.
- When you combine big data with high-powered analytics, you can accomplish business-related tasks such as:
  - Determining root causes of failures, issues and defects in near-real time.
  - Generating coupons at the point of sale based on the customer’s buying habits.
  - Recalculating entire risk portfolios in minutes.
  - Detecting fraudulent behavior before it affects your organization.

Big data affects organizations across practically every industry. See how each industry can benefit from this onslaught of information.



Who uses big data?

- **Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library.**

- **Hadoop has its origins in Apache Nutch, an open source web search engine, itself a part of the Lucene project.**

- **Nutch was started in 2002, and a working crawler and search system quickly emerged.**

A Brief History of Apache Hadoop

- **In April 2008, Hadoop broke a world record to become the fastest system to sort an entire terabyte of data. Running on a 910-node cluster, Hadoop sorted 1 terabyte in 209 seconds (just under 3.5 minutes.**

- **In November of the same year, Google reported that its MapReduce implementation sorted 1 terabyte in 68 seconds.**

- **Then, in April 2009, it was announced that a team at Yahoo! had used Hadoop to sort 1 terabyte in 62 seconds**

Hadoop Benchmarks

Hadoop

What is Hadoop?

- It's a framework for running applications on large **clusters of commodity hardware** which produces **huge data** and to process it
- Apache Software Foundation Project
- Open source
- Amazon’s EC2

Hadoop Includes

- **HDFS** a distributed filesystem
- **Map/Reduce** HDFS implements this programming model. It is an offline computing engine

<u>Concept</u>

Moving computation is more efficient than moving large data

Apache Hadoop Ecosystem

Ambari
Provisioning, Managing and Monitoring Hadoop Clusters

Sqoop
Data Exchange

Flume
Log Collector

Zookeeper
Coordination

Oozie
Workflow

Pig
Scripting

Mahout
Machine Learning

R Connectors
Statistics

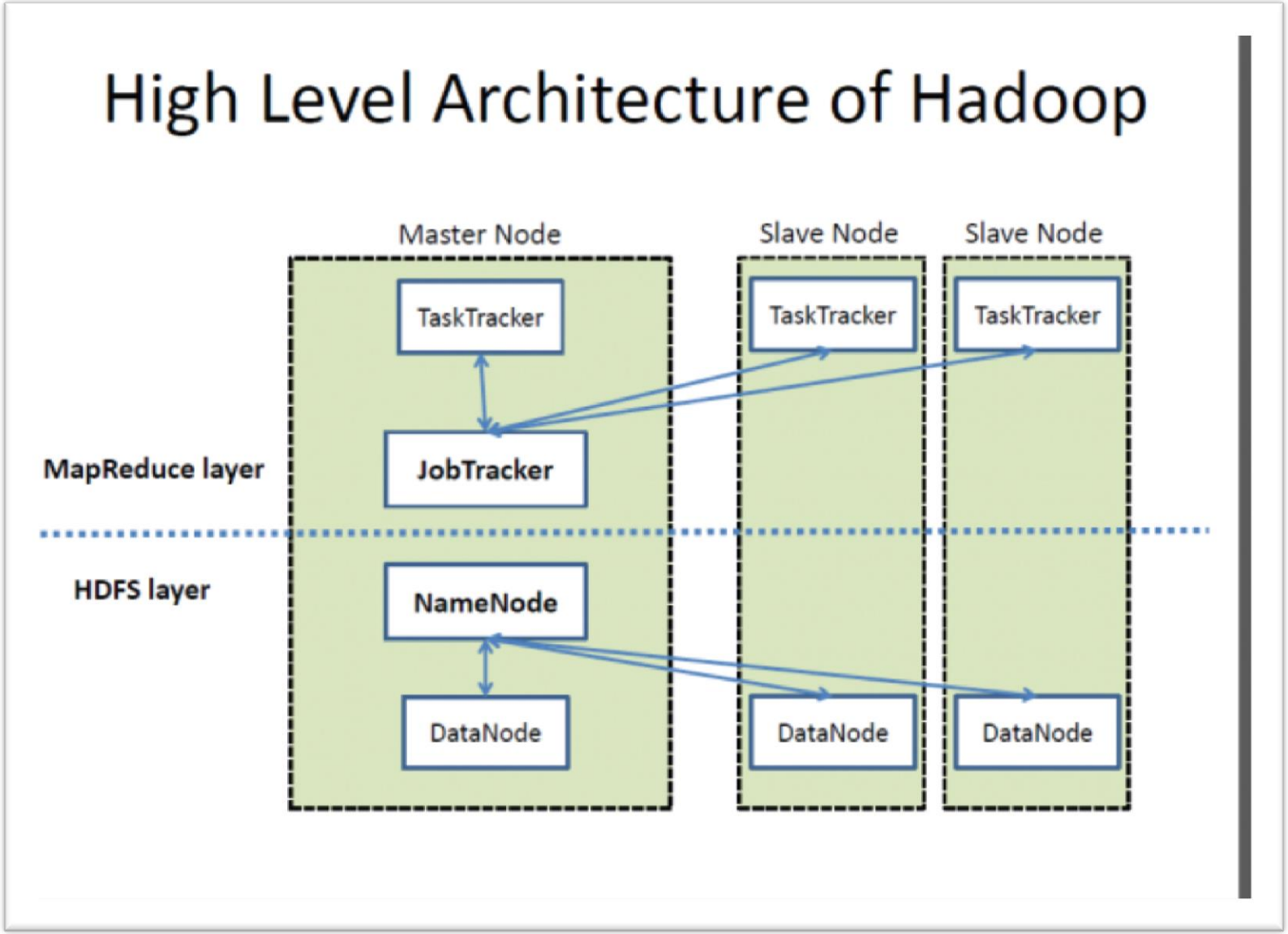
Hive
SQL Query

Hbase
Columnar Store

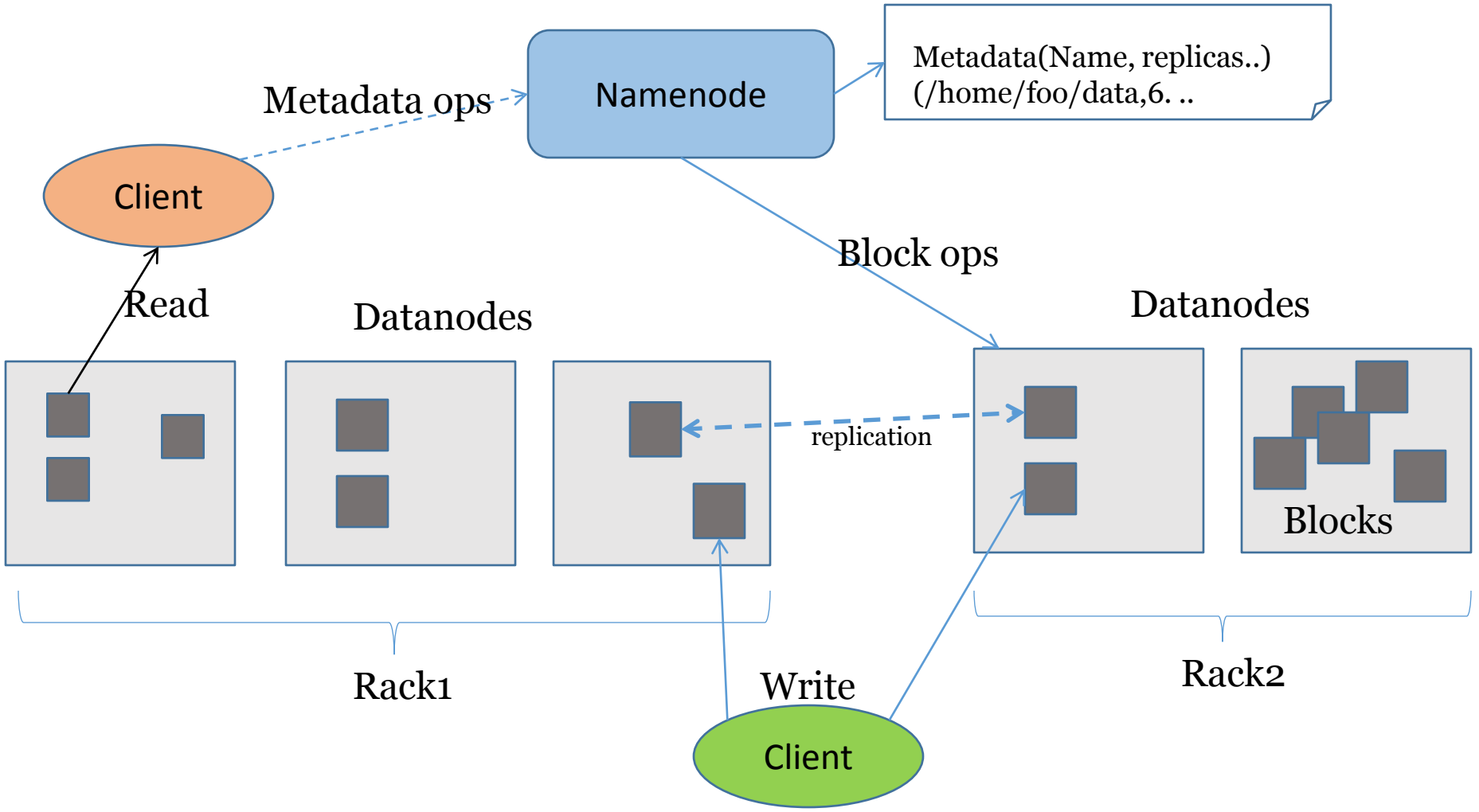
YARN Map Reduce v2
Distributed Processing Framework

HDFS
Hadoop Distributed File System

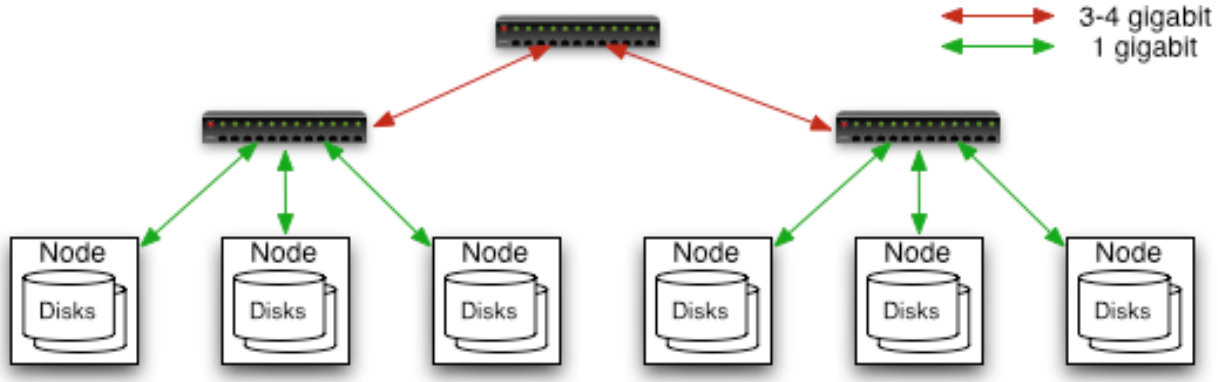
Hadoop Architecture



HDFS Architecture



Commodity Hardware



Typically in 2 level architecture

– Nodes are commodity PCs
– 30-40 nodes/rack
– Uplink from rack is 3-4 gigabit
– Rack-internal is 1 gigabit

Distributed File System

- **Single Namespace for entire cluster**
- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- **Files are broken up into blocks**
  - Typically 128 MB block size
  - Each block replicated on multiple DataNodes
- **Intelligent Client**
  - Client can find location of blocks
  - Client accesses data directly from DataNode

HDFS Why? Seek vs Transfer

- **CPU & transfer speed, RAM & disk size double every 18 - 24 months**
- **Seek time nearly constant (~5%/year)**
- **Time to read entire drive is growing vs transfer rate.**
- **Moral: scalable computing must go at transfer rate**
- **BTree (Relational DBS)**

  – operate at seek rate, log(N) seeks/access

  -- memory / stream based
- **sort/merge flat files (MapReduce)**

  – operate at transfer rate, log(N) transfers/sort

  -- Batch based

Goals of HDFS

- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 PB
- **Assumes Commodity Hardware**
  - Files are replicated to handle hardware failure
  - Detect failures and recovers from them
- **Optimized for Batch Processing**
  - Data locations exposed so that computations can move to where data resides
  - Provides very high aggregate bandwidth
- **User Space, runs on heterogeneous OS**

Basic Features: HDFS

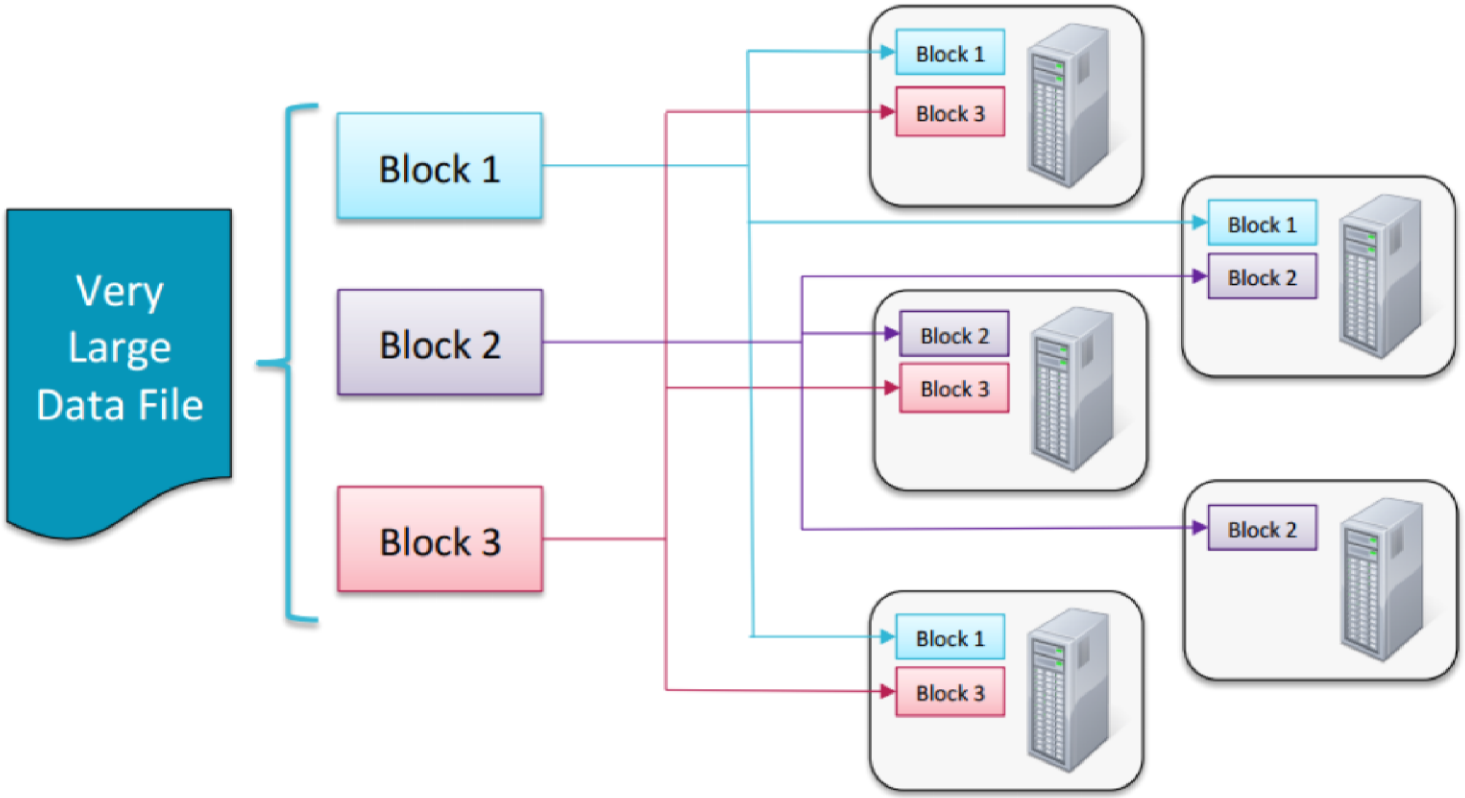
- Highly fault-tolerant
- High throughput
- Suitable for applications with large data sets
- Streaming access to file system data
- Can be built out of commodity hardware

Block

- Generally the user data is stored in the files of HDFS.
- The file in a file system will be divided into one or more segments and/or stored in individual data nodes.
- These file segments are called as blocks.
- In other words, the minimum amount of data that HDFS can read or write is called a Block.
- The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

Block Storage

- **Data files are split into blocks and distributed to data nodes**
- **Each block is replicated on multiple nodes (default 3x)**



Block Placement

- Current Strategy
  - One replica on local node
  - Second replica on a remote rack
  - Third replica on same remote rack
  - Additional replicas are randomly placed
- Clients read from nearest replicas
- Would like to make this policy pluggable

Replication Engine

- NameNode detects DataNode failures
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

- **Default Replication factor is 3**

Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
- Each file is a sequence of blocks.
- All blocks in the file except the last are of the same size.
- Blocks are replicated for fault tolerance.
- Block size and replicas are configurable per file.
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.

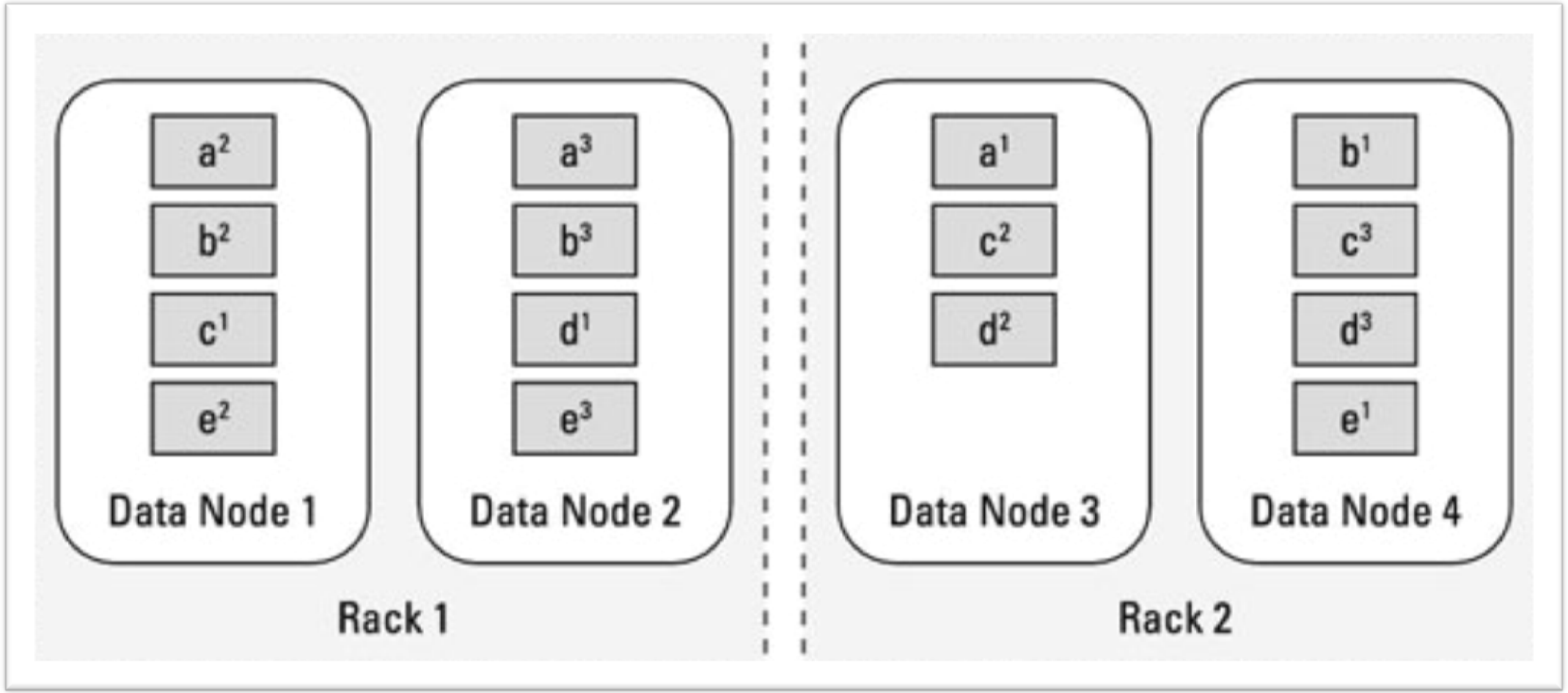
Replication Management

- The NameNode endeavors to ensure that each block always has the intended number of replicas.
- The NameNode detects that a block has become under- or over-replicated when a block report from a DataNode arrives.
- When a block becomes over replicated, the NameNode chooses a replica to remove.
- The NameNode will prefer not to reduce the number of racks that host replicas, and secondly prefer to remove a replica from the DataNode with the least amount of available disk space.
- The goal is to balance storage utilization across DataNodes without reducing the block's availability.

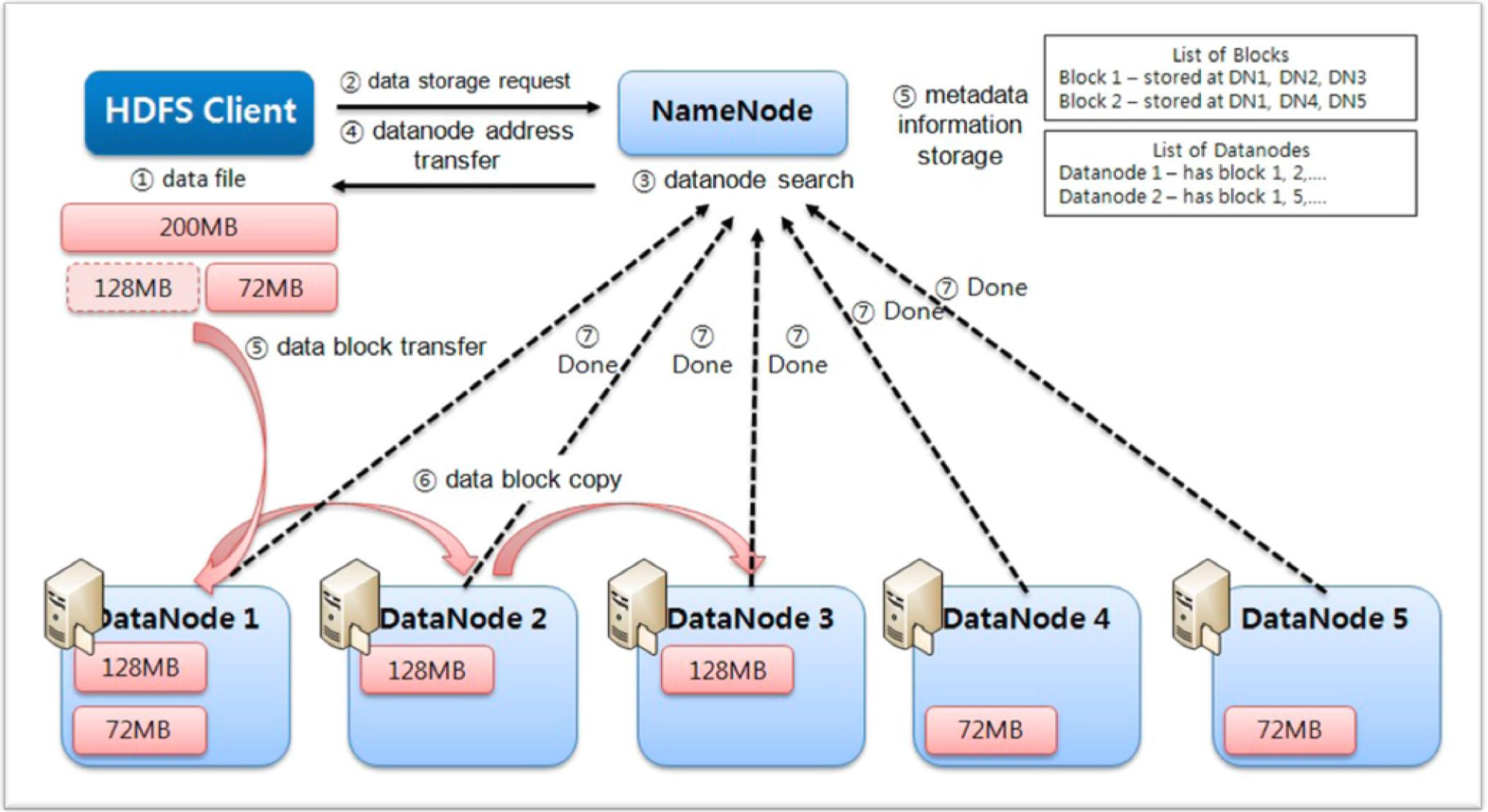
Replication Management Conti…

- A background thread periodically scans the head of the replication queue to decide where to place new replicas.
- When a block becomes under-replicated, it is put in the replication priority queue. A block with only one replica has the highest priority.
- Block replication follows a similar policy as that of new block placement. In case that the block has two existing replicas, if the two existing replicas are on the same rack, the third replica is placed on a different rack.
- The NameNode also makes sure that not all replicas of a block are located on one rack. If the NameNode detects that a block's replicas end up at one rack, the NameNode treats the block as mis-replicated and replicates the block to a different rack using the same block placement policy described above.

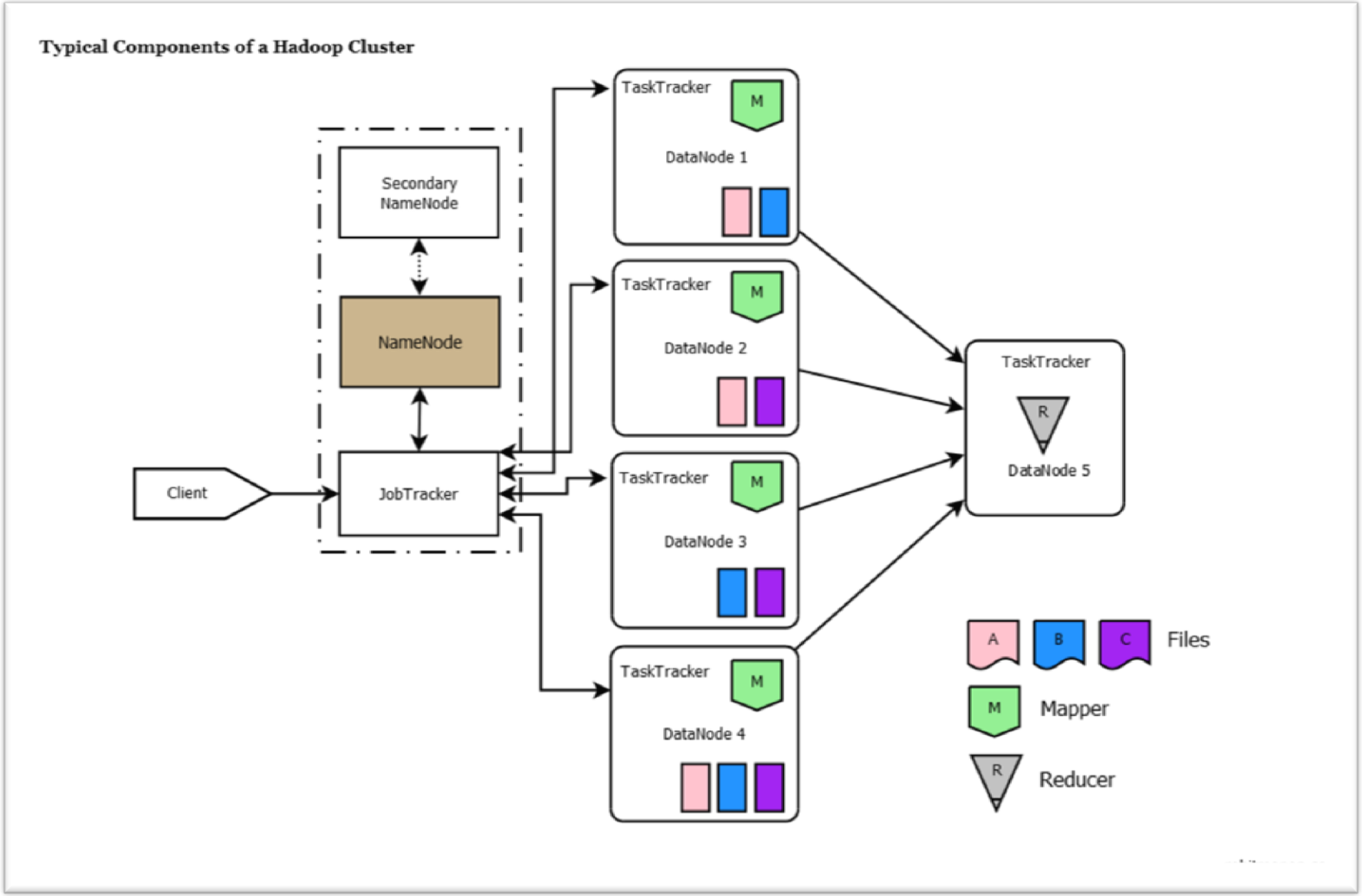
Block Placement And Replication



How Block placement Happens.



Components of Hadoop Cluster



Namenode and Datanodes

- Master/slave architecture
- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.
- There are a number of **DataNodes** usually one per node in a cluster.
- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

Namenode

- Keeps image of entire file system namespace and file Blockmap in memory.
- 4GB of local RAM is sufficient to support the above data structures that represent the huge number of files and directories.
- When the Namenode starts up it gets the FsImage and Editlog from its local file system, update FsImage with EditLog information and then stores a copy of the FsImage on the filesytstem as a checkpoint.
- Periodic checkpointing is done. So that the system can recover back to the last checkpointed state in case of a crash.

Functions of a NameNode

- Manages File System Namespace
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides
- Cluster Configuration Management
- Replication Engine for Blocks

NameNode Metadata

- **Meta-data in Memory**
  - The entire metadata is in main memory
  - No demand paging of meta-data
- **Types of Metadata**
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g creation time, replication factor
- **A Transaction Log**
  - Records file creations, file deletions. etc

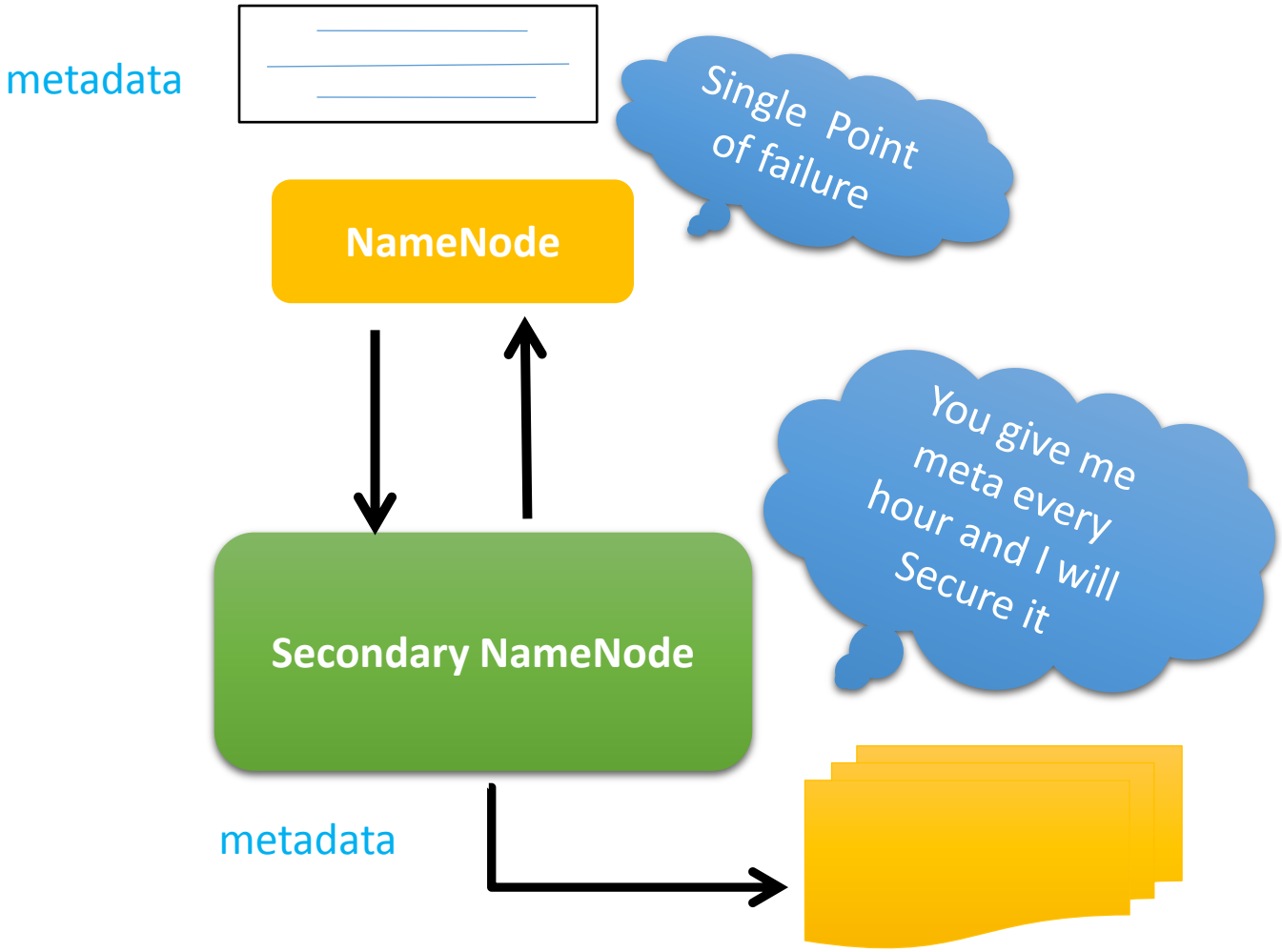
NameNode Metadata

- **Meta-data in Memory**
  - The entire metadata is in main memory
  - No demand paging of meta-data
- **Types of Metadata**
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g creation time, replication factor
- **A Transaction Log**
  - Records file creations, file deletions. etc

NameNode Failure

- A single point of failure
- Transaction Log stored in multiple directories
  - A directory on the local file system
  - A directory on a remote file system (NFS/CIFS)
- Need to develop a real HA solution

Need For Secondary NameNode



Secondary NameNode

- IMPORTANT - The Secondary NameNode is not a failover node for theNameNode.
- In the HDFS Architecture, the name – Secondary NameNode gives an impression that it is a substitute of the NameNode. Alas! It is not!
- The secondary name node is responsible for performing periodic housekeeping functions for the NameNode.
- It only creates checkpoints of the filesystem present in the NameNode.

Need for Secondary NameNode

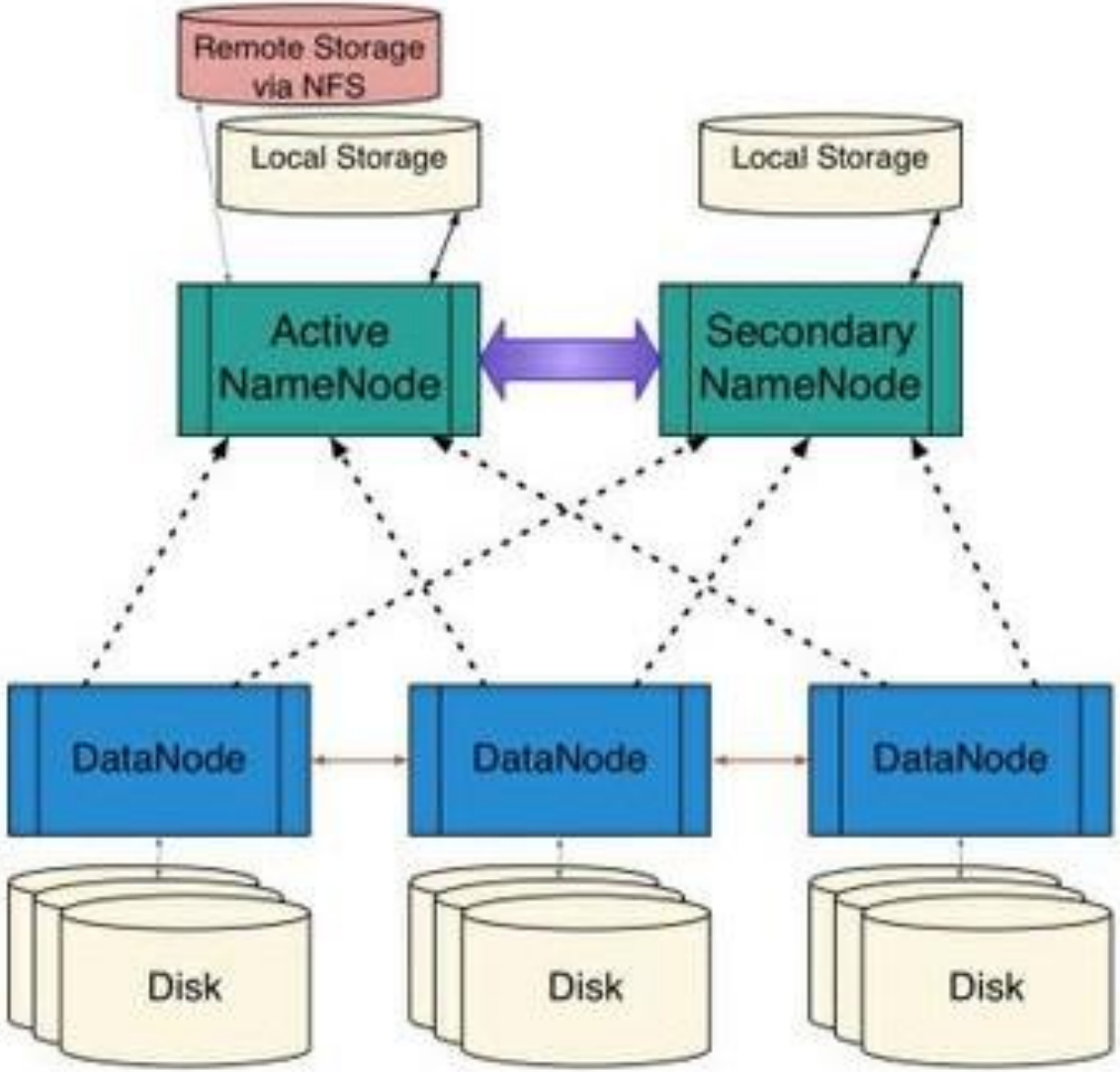
- NameNode stores vital information related to the Metadata of all the blocks stored in HDFS. This data is not only stored in the main memory, but also in the disk.

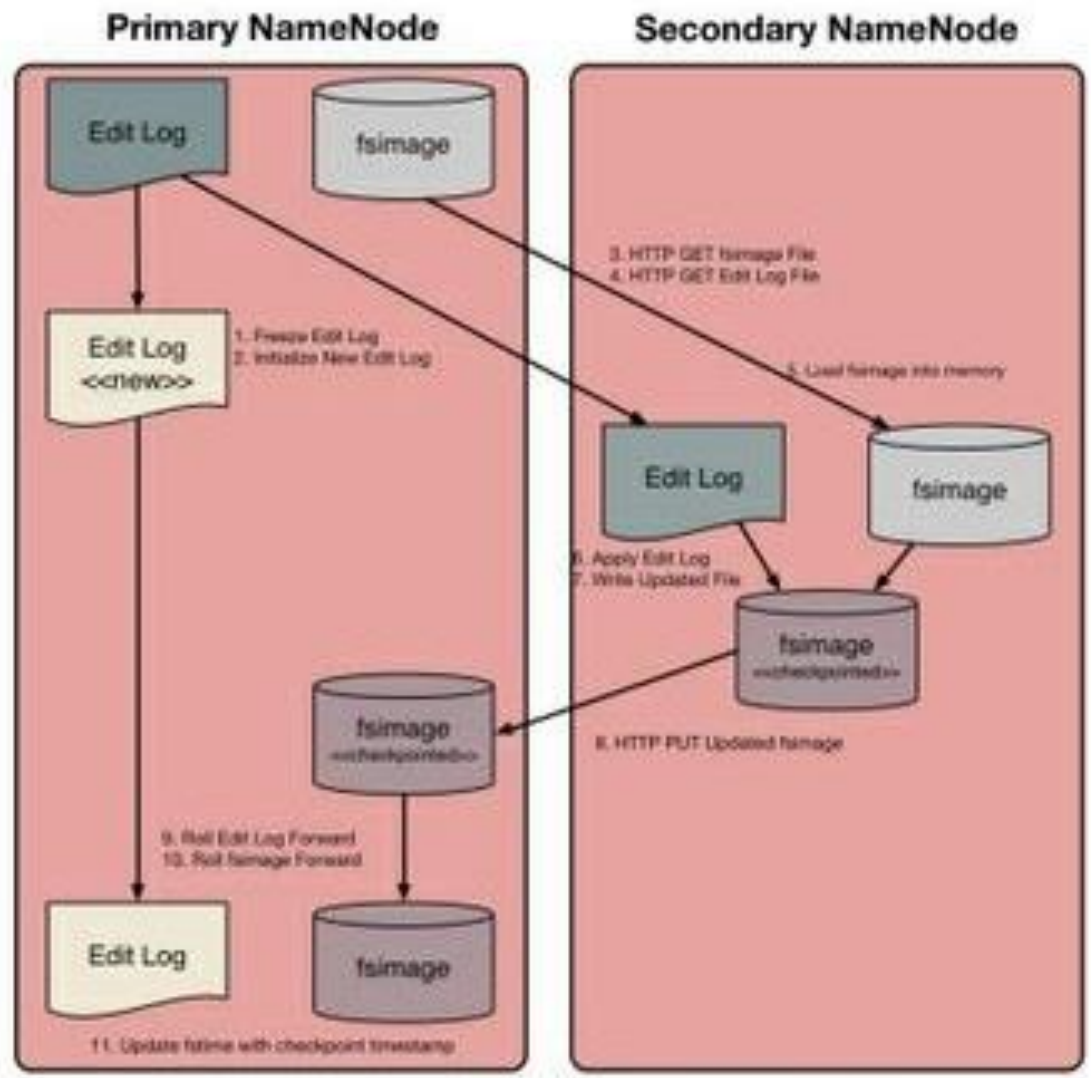
- The two associated files are:
  - Fsimage: An image of the file system on starting the NameNode.
  - EditLogs: A series of modifications done to the file system after starting the NameNode.

- If the NameNode fails, the entire Hadoop HDFS goes down and you will lose the entire RAM present in the RAM.

- It just performs regular checkpoints in HDFS. Just a helper, a checkpoint node!

HDFS-(v2) Secondary NameNode Component Process Flow





Secondary NameNode Functions

- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the editlogs with fsimage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to fsimage.
- The new fsimage is copied back to the NameNode, which is used whenever the Namenode is started the next time.

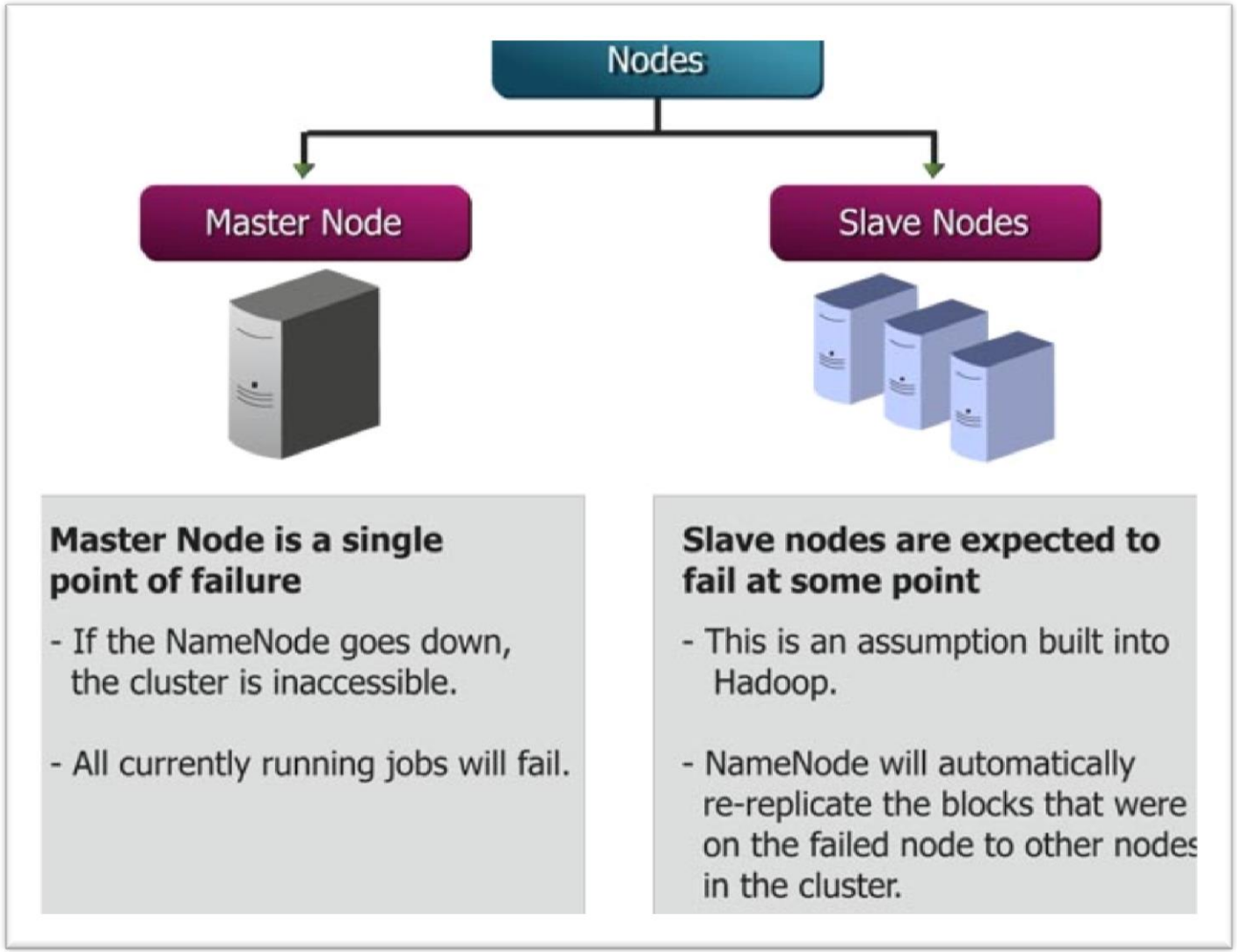
DataNode

- A Block Server
  - Stores data in the local file system (e.g. ext3)
  - Stores metadata of a block (e.g. CRC)
  - Serves data and metadata to Clients
- Block Report
  - Periodically sends a report of all existing blocks to the NameNode
- Facilitates Pipelining of Data
  - Forwards data to other specified DataNodes

Functions of DataNodes:

- Datanodes **perform the low-level read and write requests** from the file system's clients.
- They are also responsible for **creating blocks, deleting blocks and replicating** the same based on the decisions taken by the NameNode.
- They regularly **send a report** on all the blocks present in the cluster to the NameNode.
- Datanodes also enables **pipelining of data**.
- They **forward data** to other specified DataNodes.
- Datanodes **send heartbeats to the NameNode** once every 3 seconds, to report the overall health of HDFS.
- The DataNode **stores each block of HDFS data** in separate files in its local file system.
- When the Datanodes gets started, they **scan through its local file system**, creates a list of all HDFS data blocks that relate to each of these local files and send a Blockreport to the NameNode.

Nodes Failures



DataNode failure and heartbeat

- A network partition can cause a subset of Datanodes to lose connectivity with the Namenode.
- Namenode detects this condition by the absence of a Heartbeat message.
- Namenode marks Datanodes without Hearbeat and does not send any IO requests to them.
- Any data registered to the failed Datanode is not available to the HDFS.
- Also the death of a Datanode may cause replication factor of some of the blocks to fall below their specified value.

Data Correctness

- Use Checksums to validate data
  - Use CRC32
- File Creation
  - Client computes checksum per 512 bytes
  - DataNode stores the checksum
- File access
  - Client retrieves the data and checksum from DataNode
  - If Validation fails, Client tries other replicas

JobTracker

- JobTracker is responsible for taking in requests from a client and assigningTaskTrackers with tasks to be performed.
- The JobTracker tries to assign tasks to the TaskTracker on the DataNode where the data is locally present (Data Locality).
- If that is not possible it will at least try to assign tasks to TaskTrackerswithin the same rack.
- If for some reason the node fails the JobTracker assigns the task to another TaskTracker where the replica of the data exists since the data blocks are replicated across the DataNodes.
- This ensures that the job does not fail even if a node fails within the cluster.

TaskTracker

- TaskTracker is a daemon that accepts tasks (Map,Reduce and Shuffle) from the JobTracker.

- The TaskTracker keeps sending a heart beat message to the JobTracker to notify that it is alive.

- Along with the heartbeat it also sends the free slots available within it to process tasks.

- TaskTracker starts and monitors the Map & Reduce Tasks and sends progress/status information back to theJobTracker.

Fault tolerance

- Failure is the norm rather than exception
- A HDFS instance may consist of thousands of server machines, each storing part of the file system’s data.
- Since we have huge number of components and that each component has non-trivial probability of failure means that there is always some component that is non-functional.
- Detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

Data Pipelining

- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the Client moves on to write the next block in file

Rebalancer

- **Goal: % disk full on DataNodes should be similar**
  - Usually run when new DataNodes are added
  - Cluster is online when Rebalancer is active
  - Rebalancer is throttled to avoid network congestion
  - Command line tool

File system Namespace

- Hierarchical file system with directories and files
- Create, remove, move, rename etc.
- Namenode maintains the file system
- Any meta information changes to the file system recorded by the Namenode.
- An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.

Replica Selection

- Replica selection for READ operation: HDFS tries to minimize the bandwidth consumption and latency.
- If there is a replica on the Reader node then that is preferred.
- HDFS cluster may span multiple data centers: replica in the local data center is preferred over the remote one.

Safemode Startup

- On startup Namenode enters Safemode.
- Replication of data blocks do not occur in Safemode.
- Each DataNode checks in with Heartbeat and BlockReport.
- Namenode verifies that each block has acceptable number of replicas
- After a configurable percentage of safely replicated blocks check in with the Namenode, Namenode exits Safemode.
- It then makes the list of blocks that need to be replicated.
- Namenode then proceeds to replicate these blocks to other Datanodes.

Filesystem Metadata

- The HDFS namespace is stored by Namenode.
- Namenode uses a transaction log called the EditLog to record every change that occurs to the filesystem meta data.
  - For example, creating a new file.
  - Change replication factor of a file
  - EditLog is stored in the Namenode's local filesystem
- Entire filesystem namespace including mapping of blocks to files and file system properties is stored in a file FsImage. Stored in Namenode's local filesystem.

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.

Cluster Rebalancing

- HDFS architecture is compatible with data rebalancing schemes.
- A scheme might move data from one Datanode to another if the free space on a Datanode falls below a certain threshold.
- In the event of a sudden high demand for a particular file, a scheme might dynamically create additional replicas and rebalance other data in the cluster.
- These types of data rebalancing are not yet implemented: research issue.

Data Integrity

- Consider a situation: a block of data fetched from Datanode arrives corrupted.
- This corruption may occur because of faults in a storage device, network faults, or buggy software.
- A HDFS client creates the checksum of every block of its file and stores it in hidden files in the HDFS namespace.
- When a clients retrieves the contents of file, it verifies that the corresponding checksums match.
- If does not match, the client can retrieve the block from a replica.

Metadata Disk Failure

- FsImage and EditLog are central data structures of HDFS.
- A corruption of these files can cause a HDFS instance to be non-functional.
- For this reason, a Namenode can be configured to maintain multiple copies of the FsImage and EditLog.
- Multiple copies of the FsImage and EditLog files are updated synchronously.
- Meta-data is not data-intensive.
- The Namenode could be single point failure: automatic failover is NOT supported! Another research topic.

Data Blocks

- HDFS support write-once-read-many with reads at streaming speeds.
- A typical block size is 64MB (or even 128 MB).
- A file is chopped into 64MB chunks and stored.

Staging

- A client request to create a file does not reach Namenode immediately.
- HDFS client caches the data into a temporary file. When the data reached a HDFS block size the client contacts the Namenode.
- Namenode inserts the filename into its hierarchy and allocates a data block for it.
- The Namenode responds to the client with the identity of the Datanode and the destination of the replicas (Datanodes) for the block.
- Then the client flushes it from its local memory.

Staging (contd.)

- The client sends a message that the file is closed.
- Namenode proceeds to commit the file for creation operation into the persistent store.
- If the Namenode dies before file is closed, the file is lost.
- This client side caching is required to avoid network congestion; also it has precedence is AFS (Andrew file system).

Replication Pipelining

- When the client receives response from Namenode, it flushes its block in small pieces (4K) to the first replica, that in turn copies it to the next replica and so on.
- Thus data is pipelined from Datanode to the next.

Data Pieplining

- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next node in the Pipeline
- When all replicas are written, the Client moves on to write the next block in file

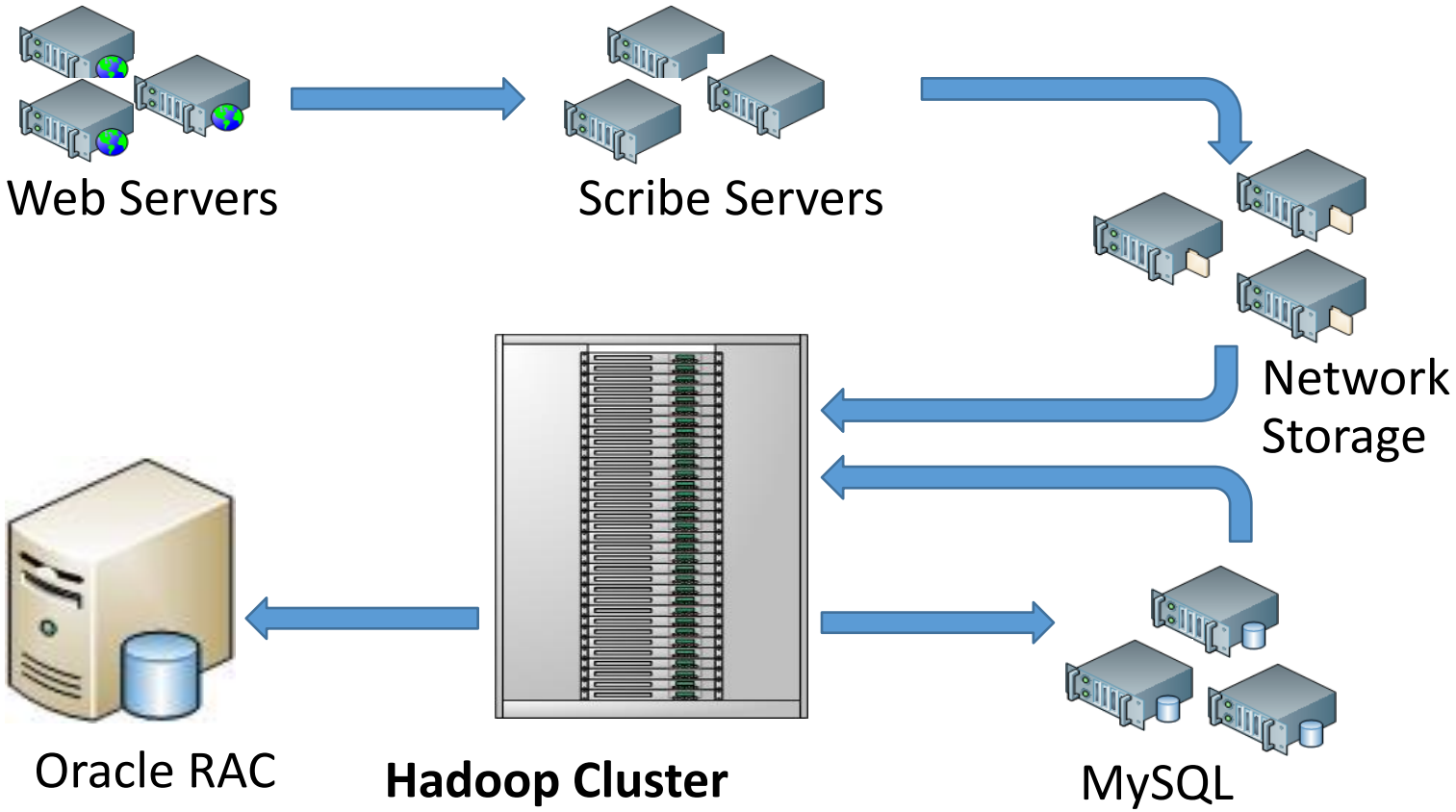
Rebalancer

- Goal: % disk full on DataNodes should be similar
  - Usually run when new DataNodes are added
  - Cluster is online when Rebalancer is active
  - Rebalancer is throttled to avoid network congestion
  - Command line tool

User Interface

- Commads for HDFS User:
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir/myfile.txt
- Commands for HDFS Administrator
  - hadoop dfsadmin -report
  - hadoop dfsadmin -decommision datanodename
- Web Interface
  - http://host:port/dfshealth.jsp

Data Flow


Web Servers
Scribe Servers
Network Storage
Oracle RAC
Hadoop Cluster
MySQL