COMP9313: Big Data Management

Lecturer: Xin Cao

Course web site: http://www.cse.unsw.edu.au/~cs9313/

Chapter 1.2 Introduction to Hadoop, HDFS, and YARN

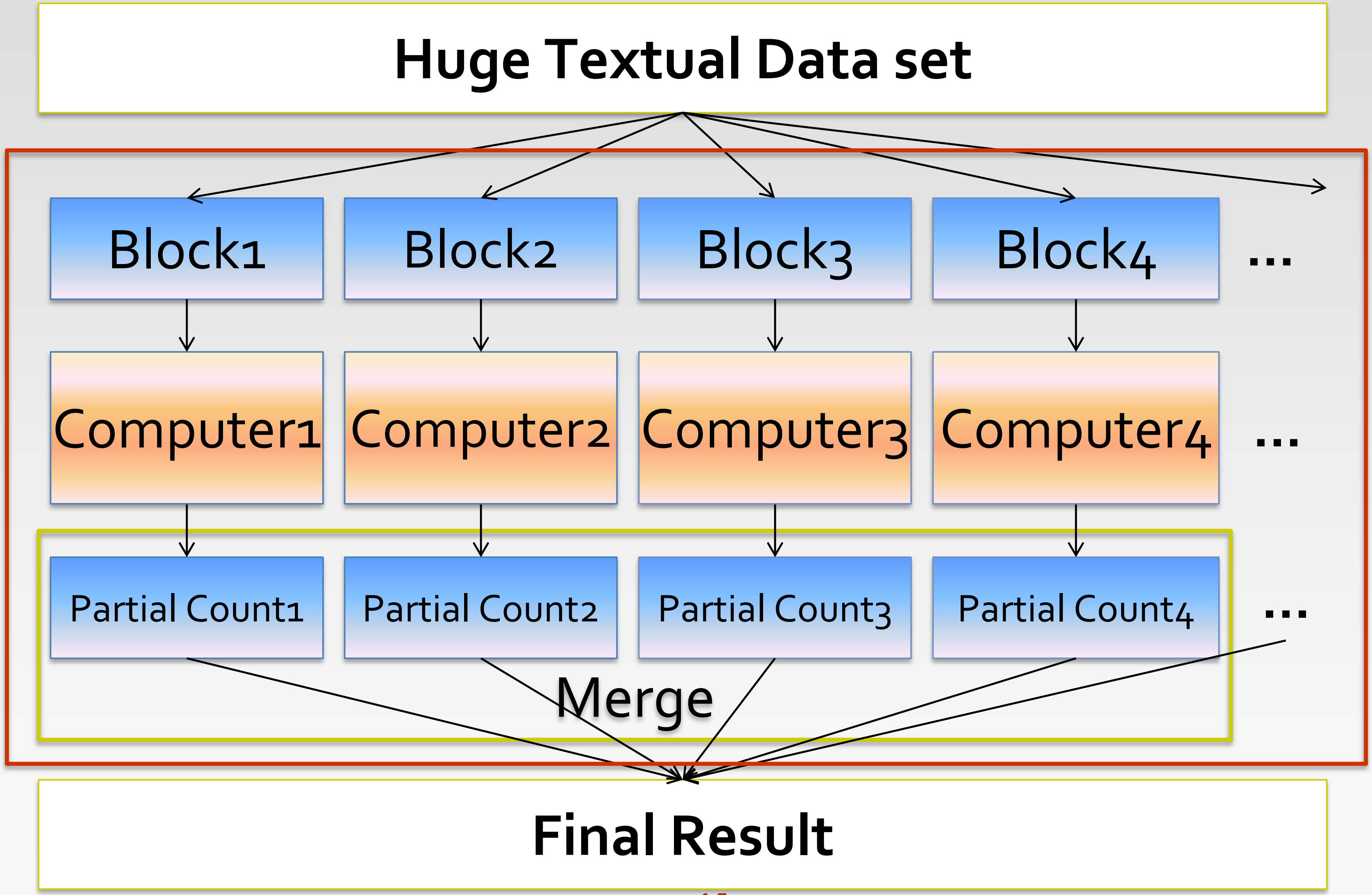
Part 1: Hadoop

Word Counting in Textual Data

- Input: A data set containing several documents
- Task: count the frequency of words appearing in the data set
- A simple solution:
  - Initialize a dictionary (or a map structure) to store the results
  - For each file, use a file reader to get the texts line by line. Tokenize each line into words. For each word, increase its count in the dictionary.

- Problems?
  - The data set cannot be stored on a single machine?

Distributed Word Count



Distributed Word Count

- Challenges?
  - Where to store the huge textual data set?
  - How to split the data set into different blocks?
    - How many blocks?
    - The size of each block?
  - What can we do if one node lost the data it received?
  - What can we do if one node cannot be connected?
  - … …

Distributed Processing is Non-Trivial

- How to assign tasks to different workers in an efficient way?
- What happens if tasks fail?
- How do workers exchange results?
- How to synchronize distributed tasks allocated to different workers?
- … …

Image courtesy of Master isolated images at FreeDigitalPhotos.net

Big Data Storage is Challenging

- Data Volumes are massive (e.g., even a single document cannot be stored on a single machine)
- Reliability of Storing PBs of data is challenging
- All kinds of failures: Disk/Hardware/Network Failures
- Probability of failures simply increase with the number of machines …

What is Hadoop

- Open-source data storage and processing platform
- Before the advent of Hadoop, storage and processing of big data was a big challenge
- Massively scalable, automatically parallelizable
  - Based on work from Google
    - Google: GFS + MapReduce + BigTable (Not open)
    - Hadoop: HDFS + Hadoop MapReduce + HBase（opensource)
- Named by Doug Cutting in 2006 (worked at Yahoo! at that time), after his son's toy elephant.

Hadoop Offers

- Redundant, Fault-tolerant data storage
- Parallel computation framework
- Job coordination

Programmers

No longer need to worry about

Q: Where file is located?

Q: How to handle failures & data lost?

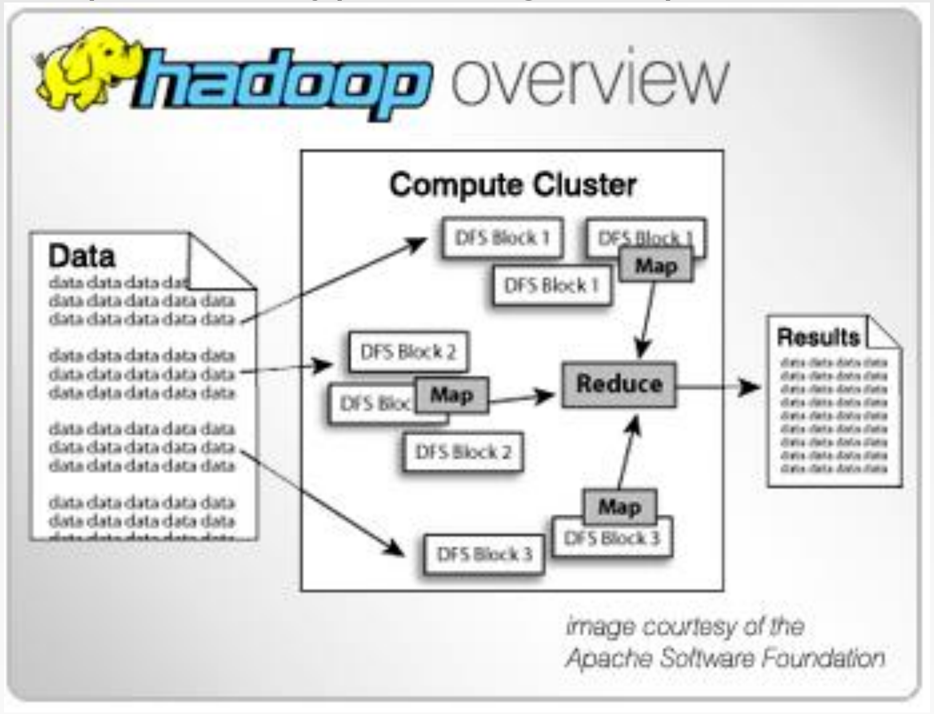
Q: How to divide computation?

Q: How to program for scaling?

… …

Why Use Hadoop?

- Cheaper
  - Scales to Petabytes or more easily
- Faster
  - Parallel data processing
- Better
  - Suited for particular types of big data problems



Companies Using Hadoop

facebook

The New York Times

ebY

hp

IBM

JPMorganChase

eHarmony®

twitter

intel®

NETFLIX

rackspace® HOSTING

amazon.com®

VISA

NING

SAMSUNG

YAHOO!®

Hadoop 1.x

- Data storage (HDFS)
  - Runs on commodity hardware (usually Linux)
  - Horizontally scalable
- Processing (MapReduce)
  - Parallelized (scalable) processing
  - Fault Tolerant
- Other Tools/Frameworks
  - HBase
  - Hive
  - … …

HDFS Storage

Redundant (3 copies)

For large files – large blocks

64 or 128 MB / block

Can scale to 1000s of nodes

MapReduce API

Batch (Job) processing

Distributed and Localized to clusters (Map)

Auto-Parallelizable for huge amounts of data

Fault-tolerant (auto retries)

Adds high availability and more

Other Libraries

Pig

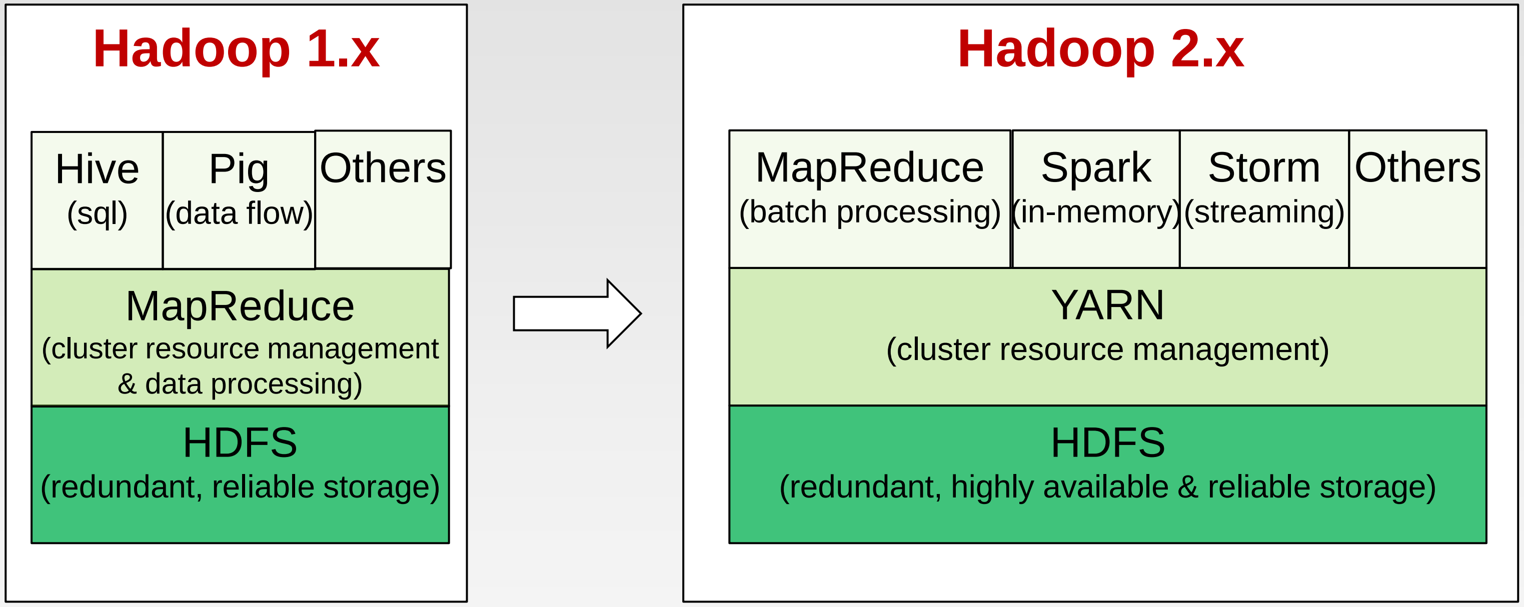
Hive

HBase

Others

Hadoop 2.x

- Single Use System
  - Batch apps
- Multi-Purpose Platform
  - Batch, Interactive, Online, Streaming



Hadoop 1.x
Hive (sql)
Pig (data flow)
Others
MapReduce (cluster resource management & data processing)
HDFS (redundant, reliable storage)
Hadoop 2.x
MapReduce (batch processing)
Spark (in-memory)
Storm (streaming)
Others
YARN (cluster resource management)
HDFS (redundant, highly available & reliable storage)

Hadoop YARN *(Yet Another Resource Negotiator)*: a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications

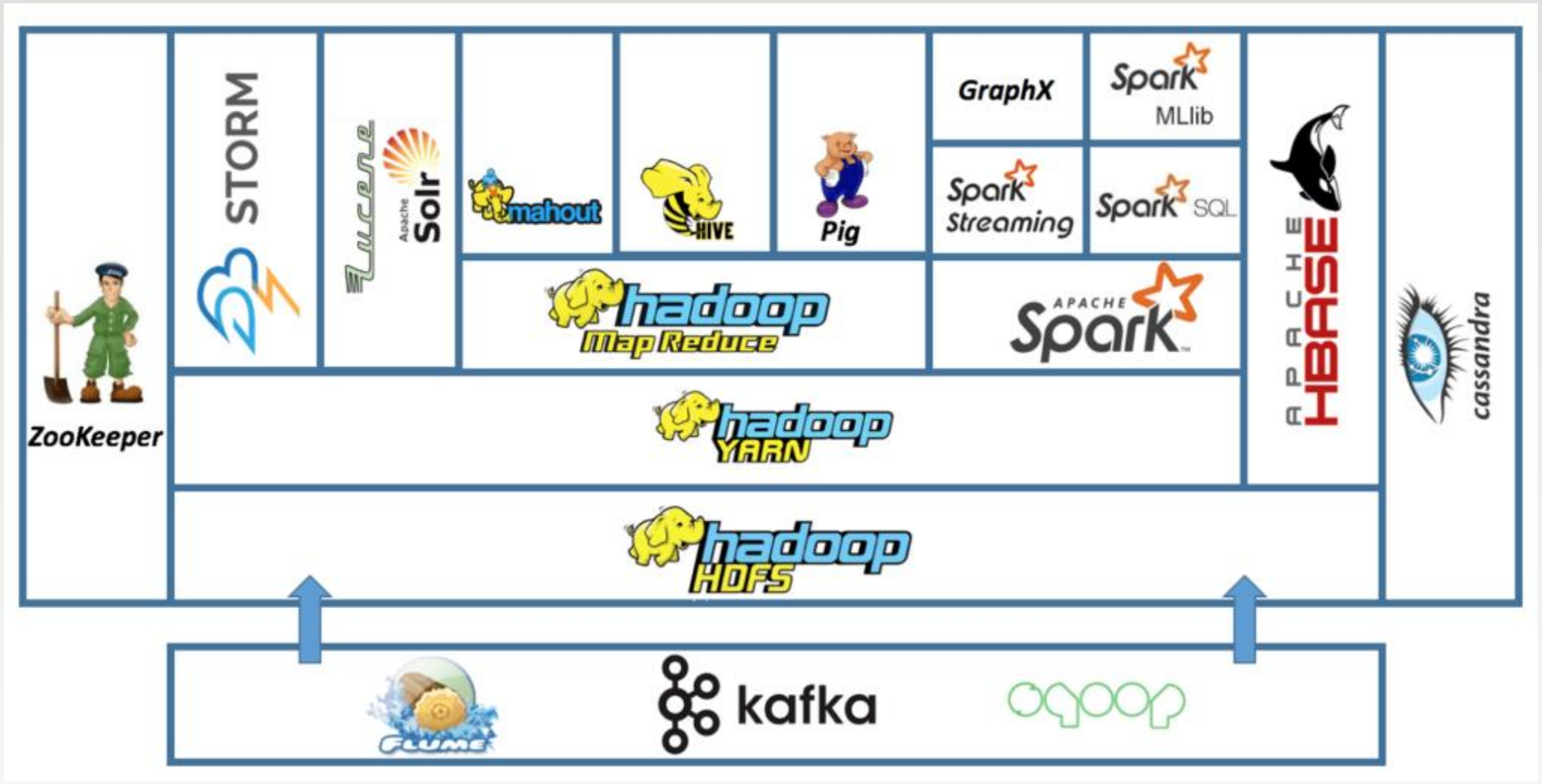
Hadoop 3.x

	Hadoop 2.x	Hadoop 3.x
Minimum supported Java version	JAVA 7	JAVA 8/11
Storage Scheme	3x Replication Scheme	Erasure encoding in HDFS
Fault Tolerance	Replication is the only way to handle fault tolerance which is not space optimized.	Erasure coding is used for handling fault tolerance.
Storage Overhead	200% of HDFS (6 blocks of data will occupy the space of 18 blocks due to replication factor)	50% (6 blocks of data will occupy 9 blocks i.e 6 blocks for actual data and 3 blocks for parity)
Scalability	Limited Scalability, can have upto 10000 nodes in a cluster.	Scalability is improved, can have more then 10000 nodes in a cluster.
NameNodes	A single active NameNode and a single Standby NameNode	allows users to run multiple standby NameNodes to tolerate the failure of more nodes

https://hadoop.apache.org/docs/stable/index.html

Hadoop Ecosystem

A combination of technologies which have proficient advantage in solving business problems.

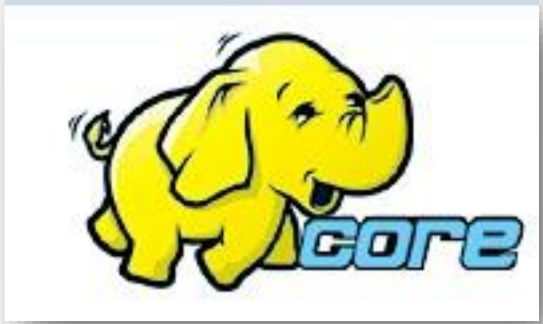


https://www.softwaretestingclass.com/introduction-to-hadoop-architecture-and-components/

Common Hadoop Distributions

- Open Source
  - Apache

- Commercial
  - Cloudera
  - Hortonworks
  - MapR
  - AWS MapReduce
  - Microsoft Azure



Setting up Hadoop Development

Hadoop Binaries

- Local install
  - Linux
  - Windows
- Cloudera's Demo VM
  - Need Virtualization software, i.e. VMware, etc…
- Cloud
  - AWS
  - Microsoft
  - Others

Data Storage

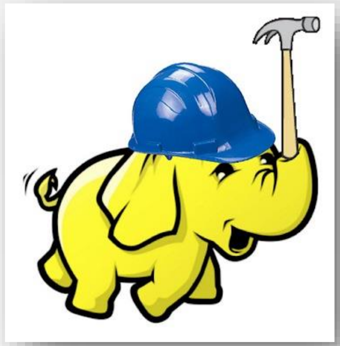
- Local
  - File System
  - HDFS Pseudo-distributed (single-node)
- Cloud
  - AWS
  - Azure
  - Others

MapReduce

- Local
- Cloud

Other Libraries & Tools

- Vendor Tools
- Libraries

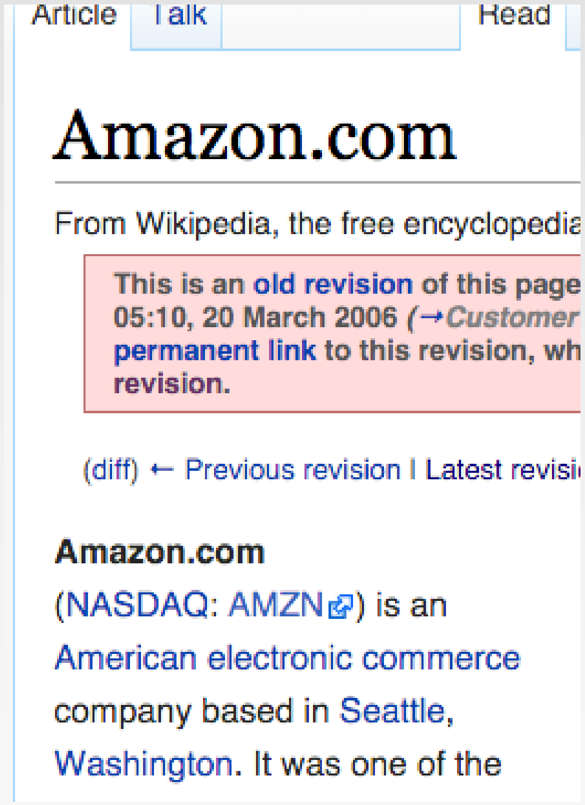


AWS (Amazon Web Services)

- Amazon

From Wikipedia 2006

From Wikipedia 2017



Article Talk Read

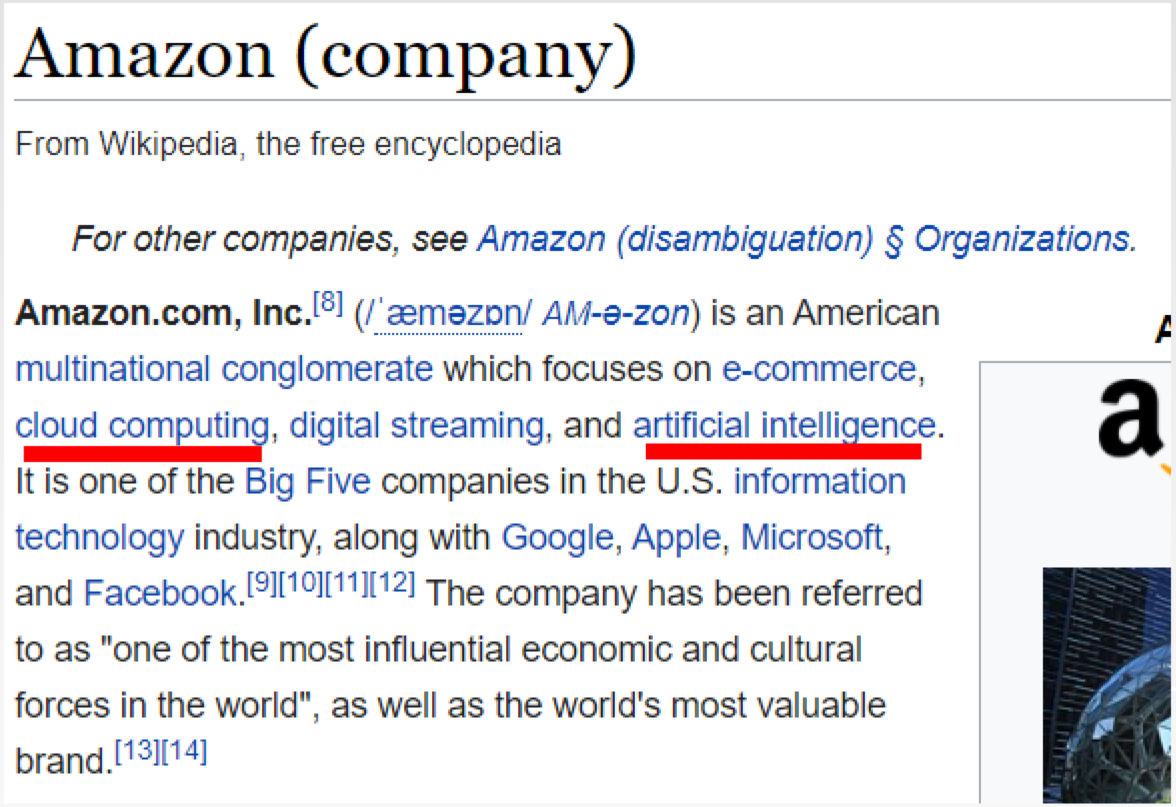
Amazon.com

From Wikipedia, the free encyclopedia

This is an old revision of this page
05:10, 20 March 2006 (→Customer
permanent link to this revision, wh
revision.

(diff) ← Previous revision | Latest revisi

Amazon.com (NASDAQ: AMZN) is an American electronic commerce company based in Seattle, Washington. It was one of the



Amazon (company)

From Wikipedia, the free encyclopedia

For other companies, see Amazon (disambiguation) § Organizations.

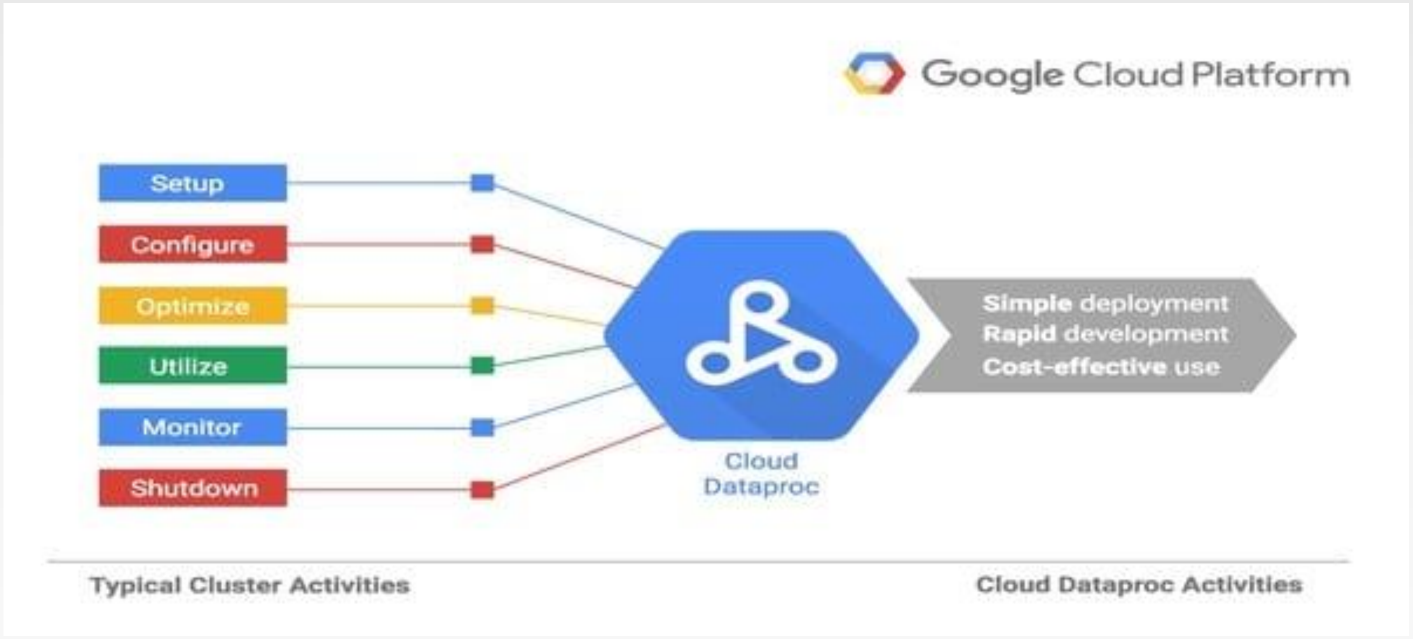
Amazon.com, Inc.[8] (/ˈæməzɒn/ *AM-ə-zon*) is an American multinational conglomerate which focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. It is one of the Big Five companies in the U.S. information technology industry, along with Google, Apple, Microsoft, and Facebook.[9][10][11][12] The company has been referred to as "one of the most influential economic and cultural forces in the world", as well as the world's most valuable brand.[13][14]

AWS (Amazon Web Services)

- AWS is a subsidiary of Amazon.com, which offers a suite of cloud computing services that make up an on-demand computing platform.
- Amazon Web Services (AWS) provides a number of different services, including:
  - Amazon Elastic Compute Cloud (EC2)
    Virtual machines for running custom software
  - Amazon Simple Storage Service (S3)
    Simple key-value store, accessible as a web service
  - Amazon Elastic MapReduce (EMR)
    Scalable MapReduce computation
  - Amazon DynamoDB
    Distributed NoSQL database, one of several in AWS
  - Amazon SimpleDB
    Simple NoSQL database
  - ...

Google Dataproc

- Dataproc is a fully managed and highly scalable service for running Apache Spark, Apache Flink, Apache Hive, and 30+ open source tools and frameworks.
  - Fast & Scalable Data Processing
  - Affordable Pricing
  - Open Source Ecosystem

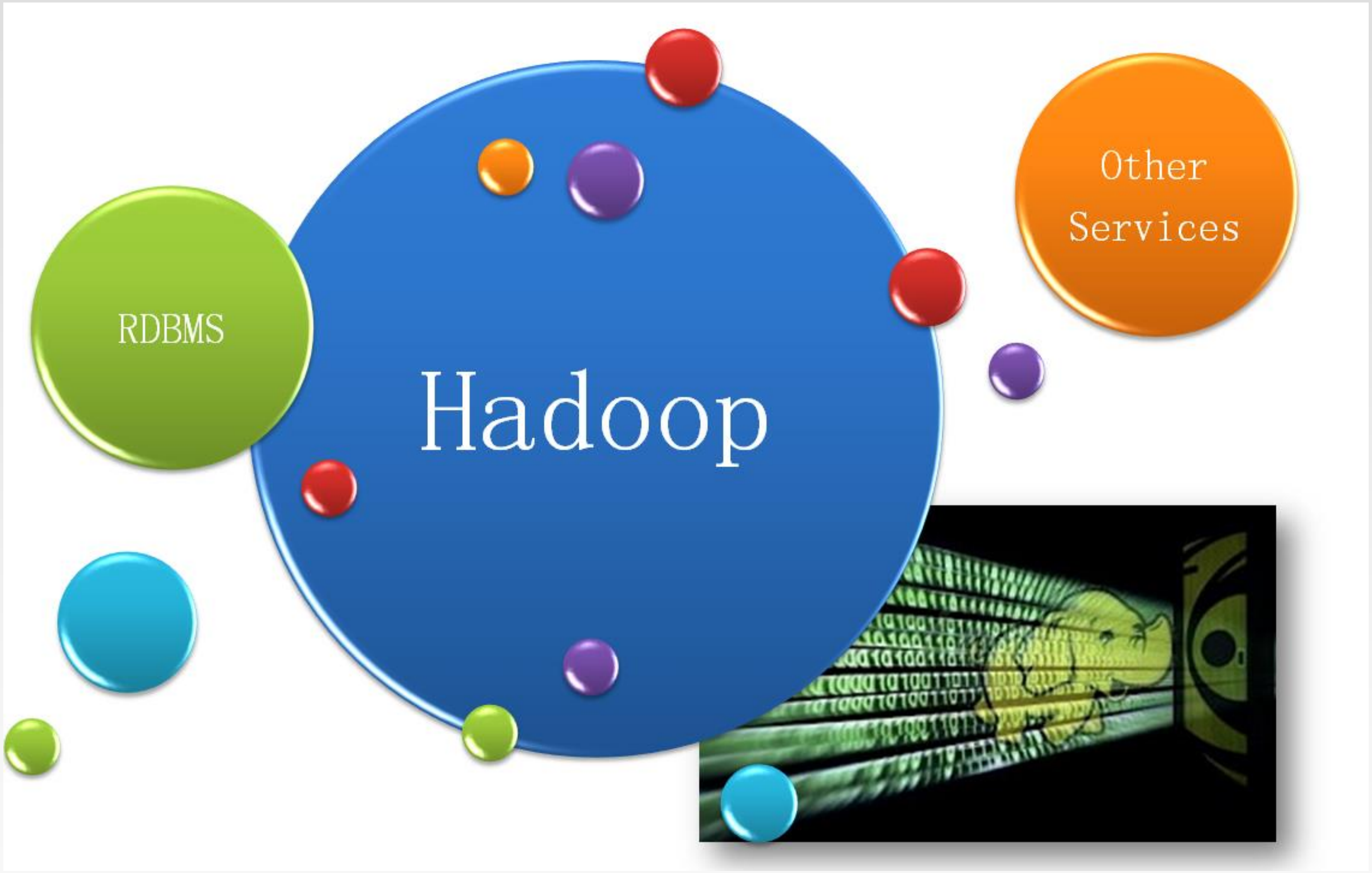


Comparing: RDBMS vs. Hadoop

Feature	RDBMS	Hadoop
Data Variety	Mainly for Structured data.	Used for Structured, Semi-Structured and Unstructured data
Data Storage	Average size data (GBS)	Use for large data set (Tbs and Pbs)
Querying	SQL Language	HQL (Hive Query Language)
Schema	Required on write (static schema)	Required on read (dynamic schema)
Speed	Reads are fast	Both reads and writes are fast
Cost	License	Free
Use Case	OLTP (Online transaction processing)	Analytics (Audio, video, logs etc), Data Discovery
Data Objects	Works on Relational Tables	Works on Key/Value Pair
Throughput	Low	High
Scalability	Vertical	Horizontal
Hardware Profile	High-End Servers	Commodity/Utility Hardware
Integrity	High (ACID)	Low

https://www.educba.com/hadoop-vs-rdbms/

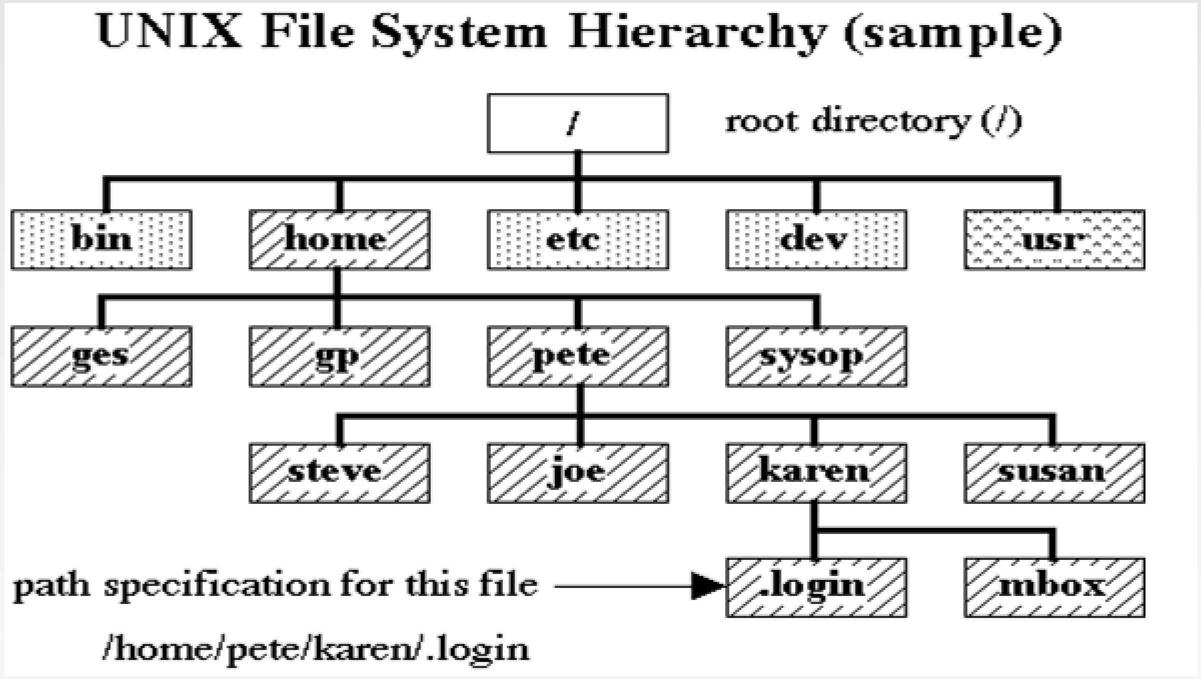
The Changing Data Management Landscape



Part 2: HDFS

File System

- A filesystem is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk.



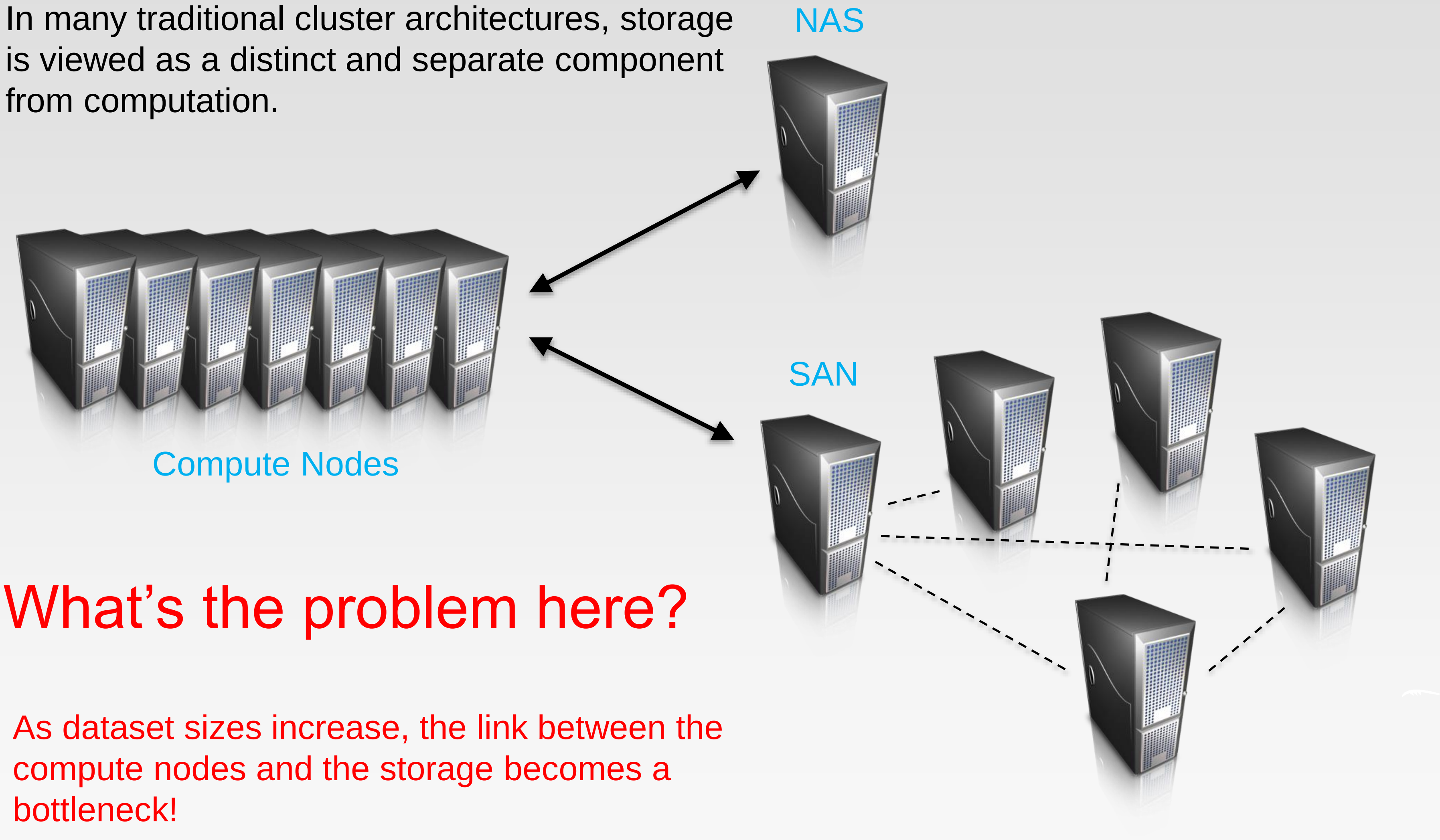
Latency and Throughput

- ***Latency*** is the time required to perform some action or to produce some result.
  - Measured in units of time -- hours, minutes, seconds, nanoseconds or clock periods.
  - I/O latency: the time that it takes to complete a single I/O.

- ***Throughput*** is the number of such actions executed or results produced per unit of time.
  - Measured in units of whatever is being produced (e.g., data) per unit of time.
  - Disk throughput: the maximum rate of sequential data transfer, measured by Mb/sec etc.

How to Move Data to Workers?

In many traditional cluster architectures, storage is viewed as a distinct and separate component from computation.



What's the problem here?

As dataset sizes increase, the link between the compute nodes and the storage becomes a bottleneck!

Distributed File System

- Don’t move data to workers… move workers to the data!
  - Store data on the local disks of nodes in the cluster
  - Start up the workers on the node that has the data local
- Why?
  - Not enough RAM to hold all the data in memory
  - Disk access is slow (low-latency), but disk throughput is reasonable (high throughput)
- A distributed file system is the answer
  - A distributed file system is a client/server-based application that allows clients to access and process data stored on the server as if it were on their own computer
  - GFS (Google File System) for Google’s MapReduce
  - HDFS (Hadoop Distributed File System) for Hadoop

Assumptions and Goals of HDFS

- Very large datasets
  - 10K nodes, 100 million files, 10PB
- Streaming data access
  - Designed more for batch processing rather than interactive use by users
  - The emphasis is on high throughput of data access rather than low latency of data access.
- Simple coherency model
  - Built around the idea that the most efficient data processing pattern is a write-once read-many-times pattern
  - A file once created, written, and closed need not be changed except for appends and truncates
- “Moving computation is cheaper than moving data”
  - Data locations exposed so that computations can move to where data resides

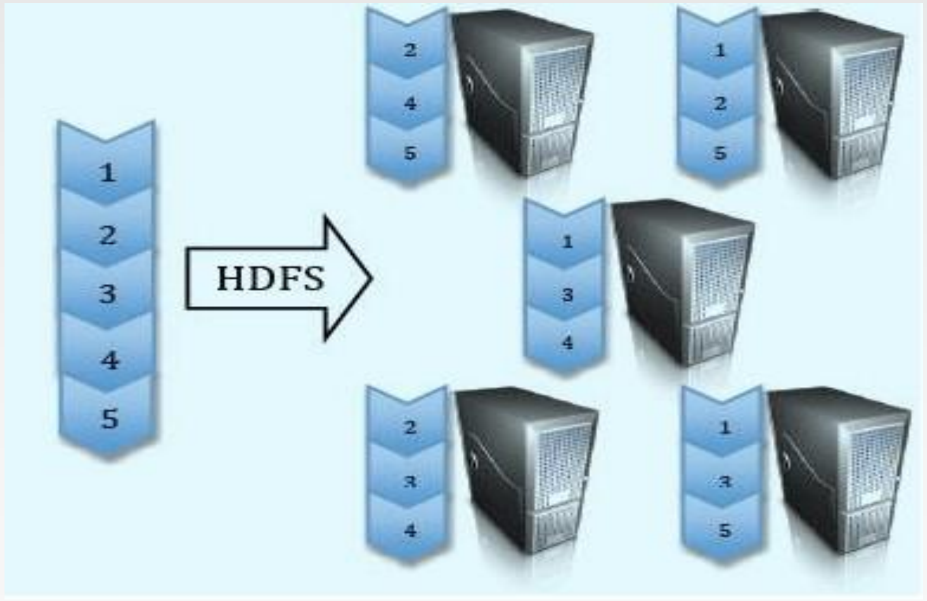
Assumptions and Goals of HDFS (Cont')

- Assumes Commodity Hardware
  - Files are replicated to handle hardware failure
  - Hardware failure is normal rather than exception. Detect failures and recover from them
- Portability across heterogeneous hardware and software platforms
  - designed to be easily portable from one platform to another

- HDFS is not suited for:
  - Low-latency data access (HBase is a better option)
  - Lots of small files (NameNodes hold metadata in memory)

HDFS Architecture

- HDFS is a block-structured file system: Files broken into blocks of 64MB or 128MB
- A file can be made of several blocks, and they are stored across a cluster of one or more machines with data storage capacity.
- Each block of a file is replicated across a number of machines, To prevent loss of data.

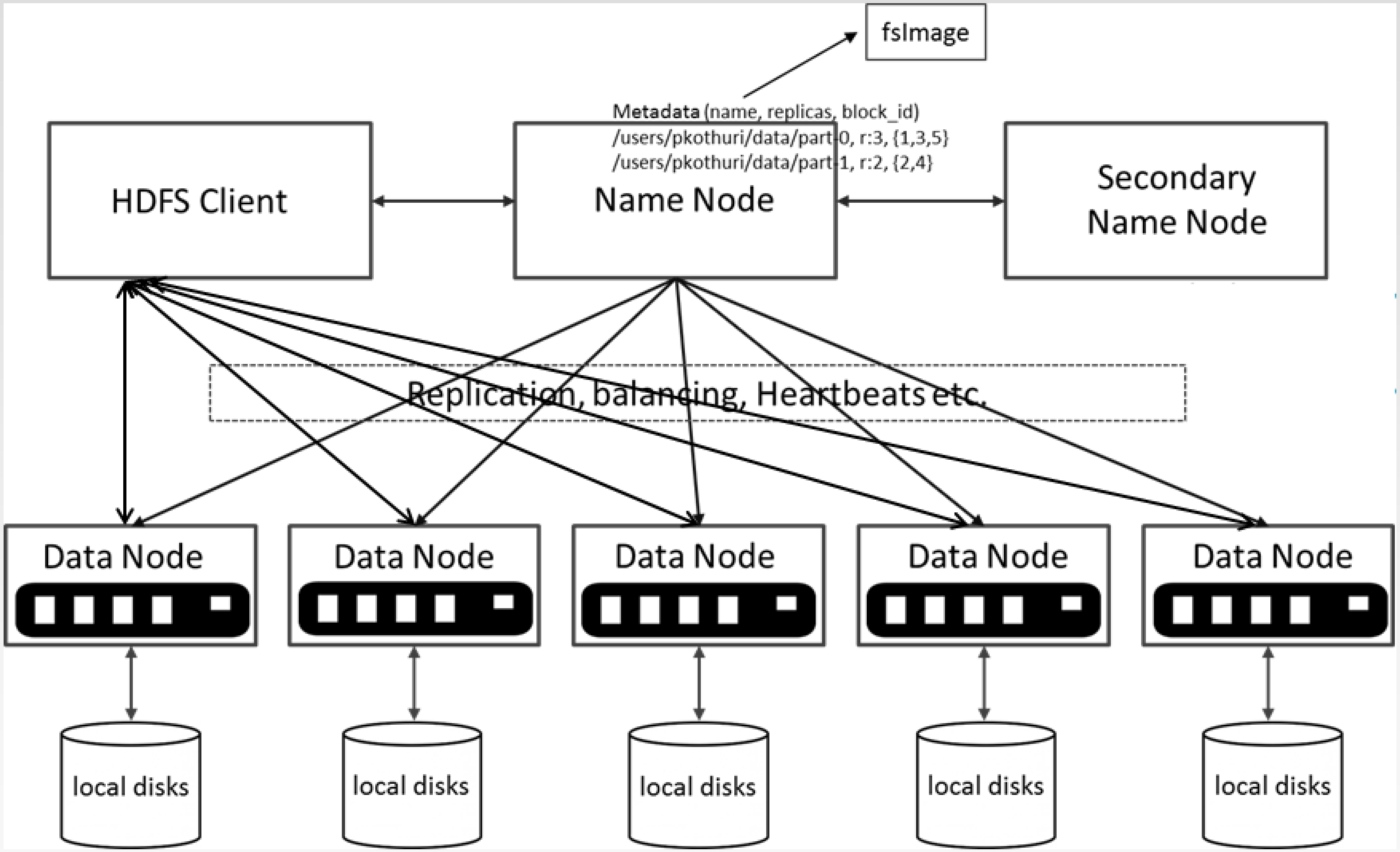


HDFS Architecture

- HDFS has a master/slave architecture.

- There are two types (and a half) of machines in a HDFS cluster
  - NameNode: the heart of an HDFS filesystem, it maintains and manages the file system metadata. E.g., what blocks make up a file, and on which datanodes those blocks are stored.
    - Only one in an HDFS cluster
  - DataNode: where HDFS stores the actual data. Serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode
    - A number of DataNodes usually one per node in a cluster.
    - A file is split into one or more blocks and set of blocks are stored in DataNodes.
  - Secondary NameNode: **NOT** a backup of NameNode!!
    - Checkpoint node. Periodic merge of Transaction log
    - Help NameNode start up faster next time

HDFS Architecture


fsImage
Metadata (name, replicas, block_id)
/users/pkothuri/data/part-0, r:3, {1,3,5}
/users/pkothuri/data/part-1, r:2, {2,4}
HDFS Client
Name Node
Secondary Name Node
Replication, balancing, Heartbeats etc.
Data Node
Data Node
Data Node
Data Node
Data Node
local disks
local disks
local disks
local disks
local disks

Functions of a NameNode

- Managing the file system namespace:
  - Maintain the namespace tree operations like opening, closing, and renaming files and directories.
  - Determine the mapping of file blocks to DataNodes (the physical location of file data).
  - Store file metadata.
- Coordinating file operations:
  - Directs clients to DataNodes for reads and writes
  - No data is moved through the NameNode
- Maintaining overall health:
  - Collect block reports and heartbeats from DataNodes
  - Block re-replication and rebalancing
  - Garbage collection

NameNode Metadata

- HDFS keeps the entire namespace in RAM, allowing fast access to the metadata.
  - 4GB of local RAM is sufficient
- 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 (EditLog)
  - Records file creations, file deletions etc

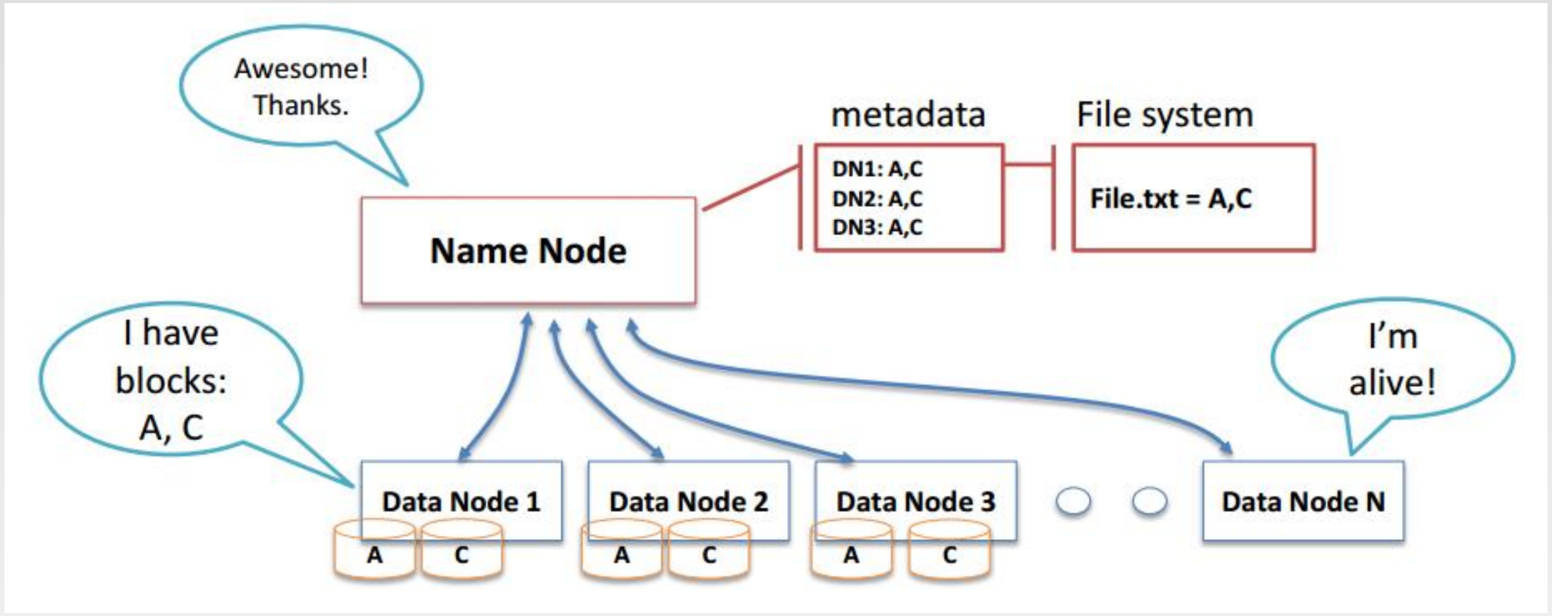
Functions of DataNodes

- Responsible for serving read and write requests from the file system’s clients.
- Perform block creation, deletion, and replication upon instruction from the NameNode.
- Periodically sends a report of all existing blocks to the NameNode (Blockreport)
- Facilitates Pipelining of Data
  - Forwards data to other specified DataNodes

Communication between NameNode and DataDode

- Heartbeats
  - DataNodes send heartbeats to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available.
    - Once every 3 seconds
  - The NameNode marks DataNodes without recent Heartbeats as dead and does not forward any new IO requests to them
- Blockreports
  - A Blockreport contains a list of all blocks on a DataNode
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster periodically

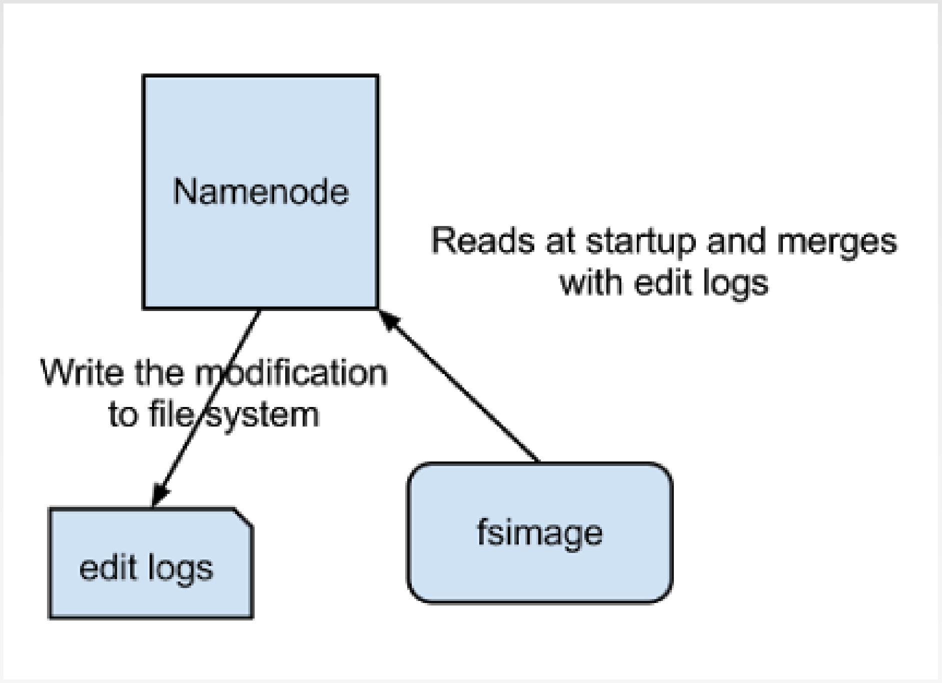
Communication between NameNode and DataDode



- TCP – every 3 seconds a Heartbeat
- Every 10th heartbeat is a Blockreport
- Name Node builds metadata from Blockreports
- If Name Node is down, HDFS is down

Inside NameNode

- FsImage - the snapshot of the filesystem when NameNode started
  - A master copy of the metadata for the file system
- EditLogs - the sequence of changes made to the filesystem after NameNode started


Namenode
Reads at startup and merges with edit logs
Write the modification to file system
edit logs
fsimage

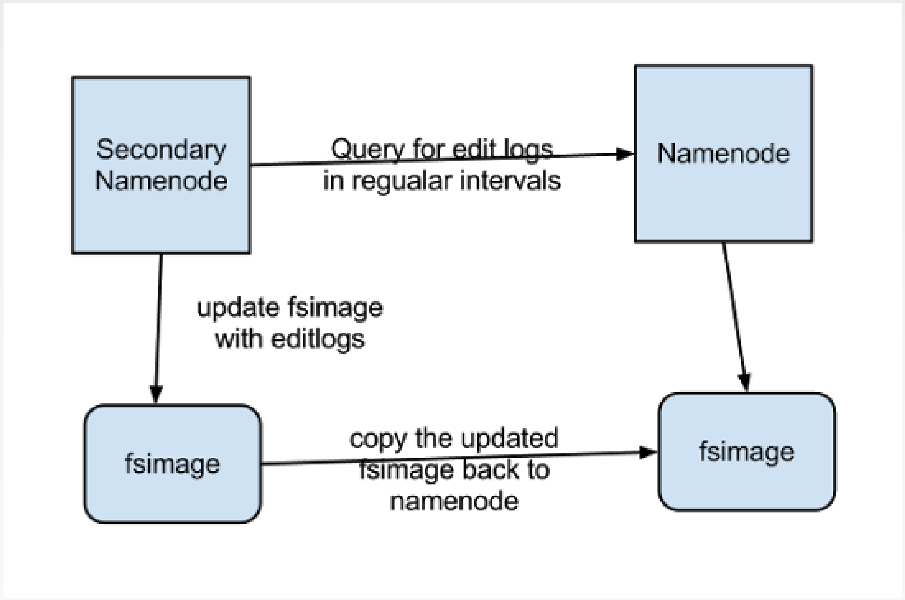
Inside NameNode

- Only in the restart of NameNode, EditLogs are applied to FsImage to get the latest snapshot of the file system.
- But NameNode restart are rare in production clusters which means EditLogs can grow very large for the clusters where NameNode runs for a long period of time.
  - EditLog become very large , which will be challenging to manage it
  - NameNode restart takes long time because lot of changes has to be merged
  - In the case of crash, we will lose huge amount of metadata since FsImage is very old

- How to overcome this issue?

Secondary NameNode

- Secondary NameNode helps to overcome the above issues by taking over responsibility of merging EditLogs with FsImage from the NameNode.
  - It gets the EditLogs from the NameNode periodically and applies to FsImage
  - Once it has new FsImage, it copies back to NameNode
  - NameNode will use this FsImage for the next restart, which will reduce the startup time


Secondary Namenode
Query for edit logs in regualar intervals
Namenode
update fsimage with editlogs
fsimage
copy the updated fsimage back to namenode
fsimage

File System Namespace

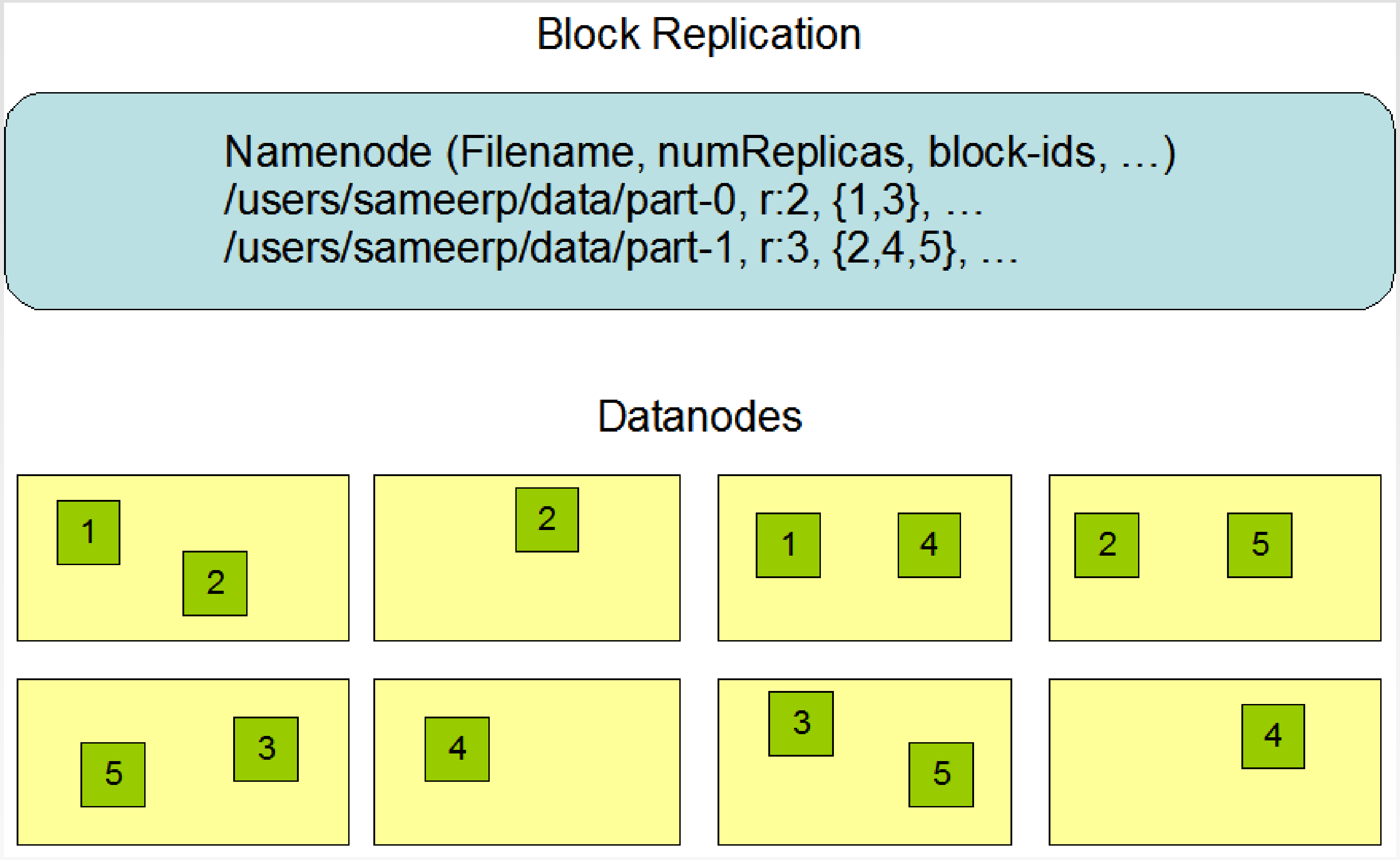
- Hierarchical file system with directories and files
  - /user/comp9313
- Create, remove, move, rename etc.
- NameNode maintains the file system
- Any meta information changes to the file system recorded by the NameNode (EditLog).
- An application can specify the number of replicas of the file needed: replication factor of the file.

HDFS Commands

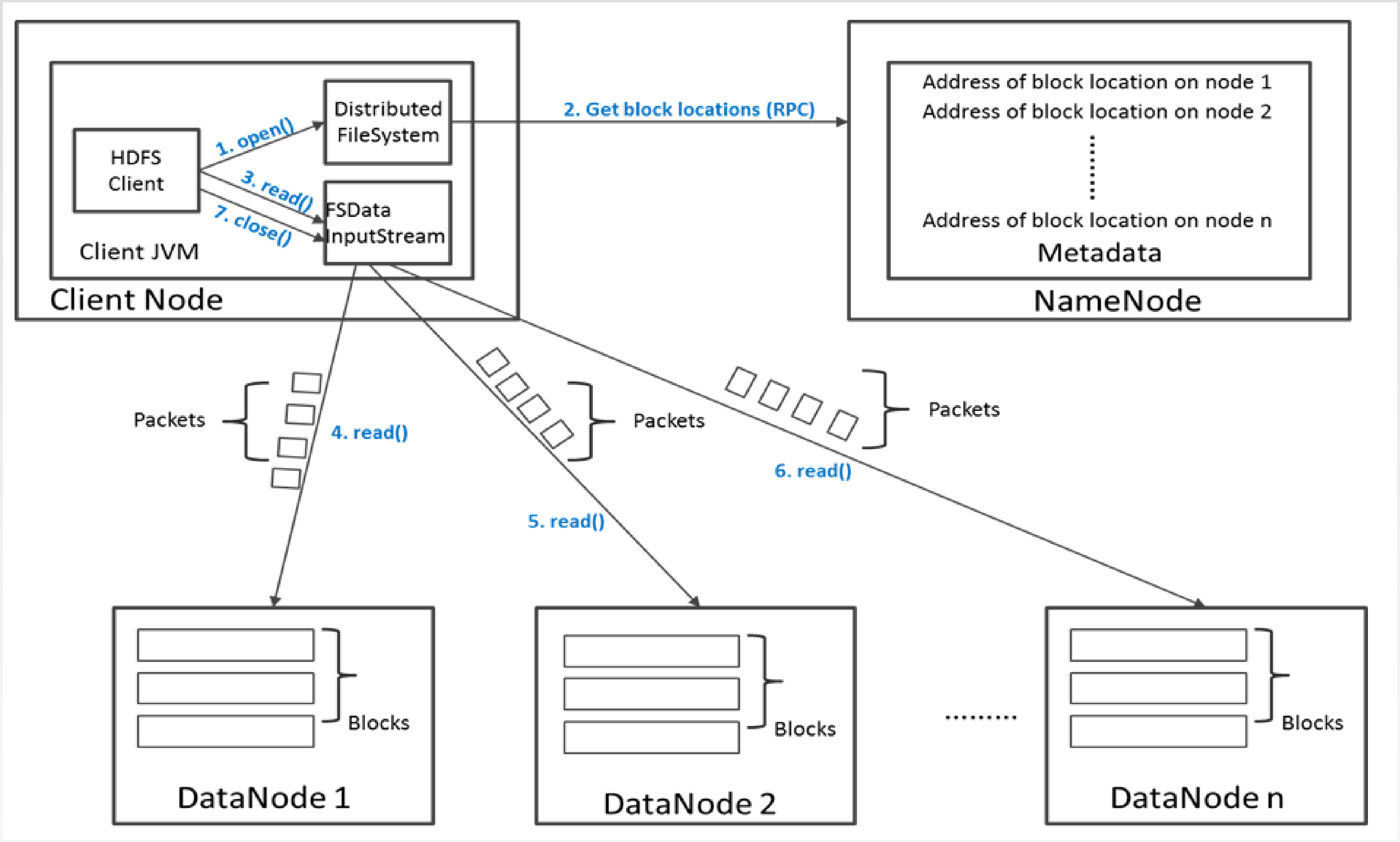
- All HDFS commands are invoked by the bin/hdfs script. Running the hdfs script without any arguments prints the description for all commands.
- Usage: hdfs [SHELL_OPTIONS] COMMAND [GENERIC_OPTIONS] [COMMAND_OPTIONS]
  - hdfs dfs [COMMAND [COMMAND_OPTIONS]]
  - Run a filesystem command on the file system supported in Hadoop. The various COMMAND_OPTIONS can be found at File System Shell Guide.

Data Replication

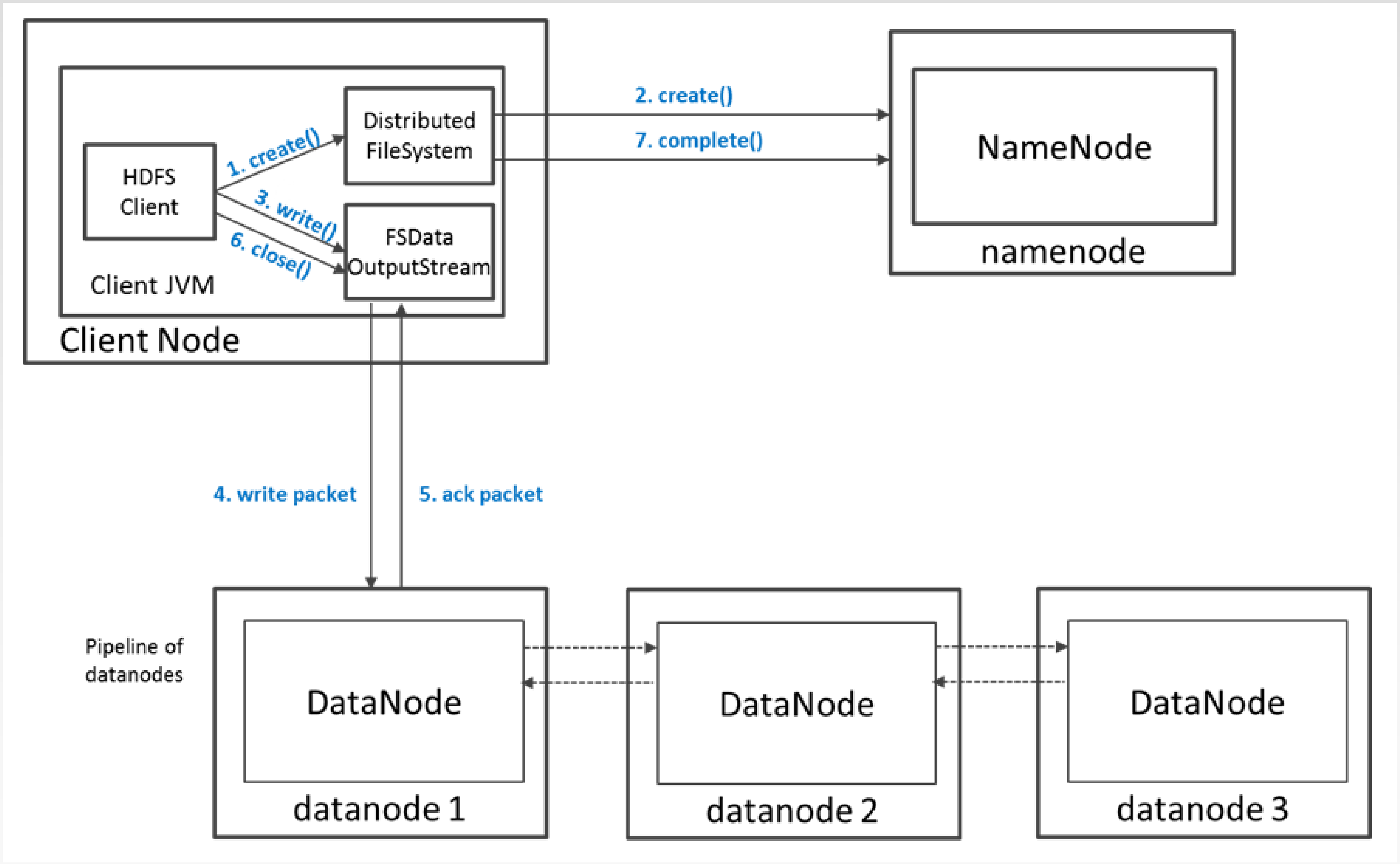
- The NameNode makes all decisions regarding replication of blocks.


Block Replication
Namenode (Filename, numReplicas, block-ids, …)
/users/sameerp/data/part-0, r:2, {1,3}, …
/users/sameerp/data/part-1, r:3, {2,4,5}, …
Datanodes
1
2
2
1
4
2
5
5
3
4
3
5
4

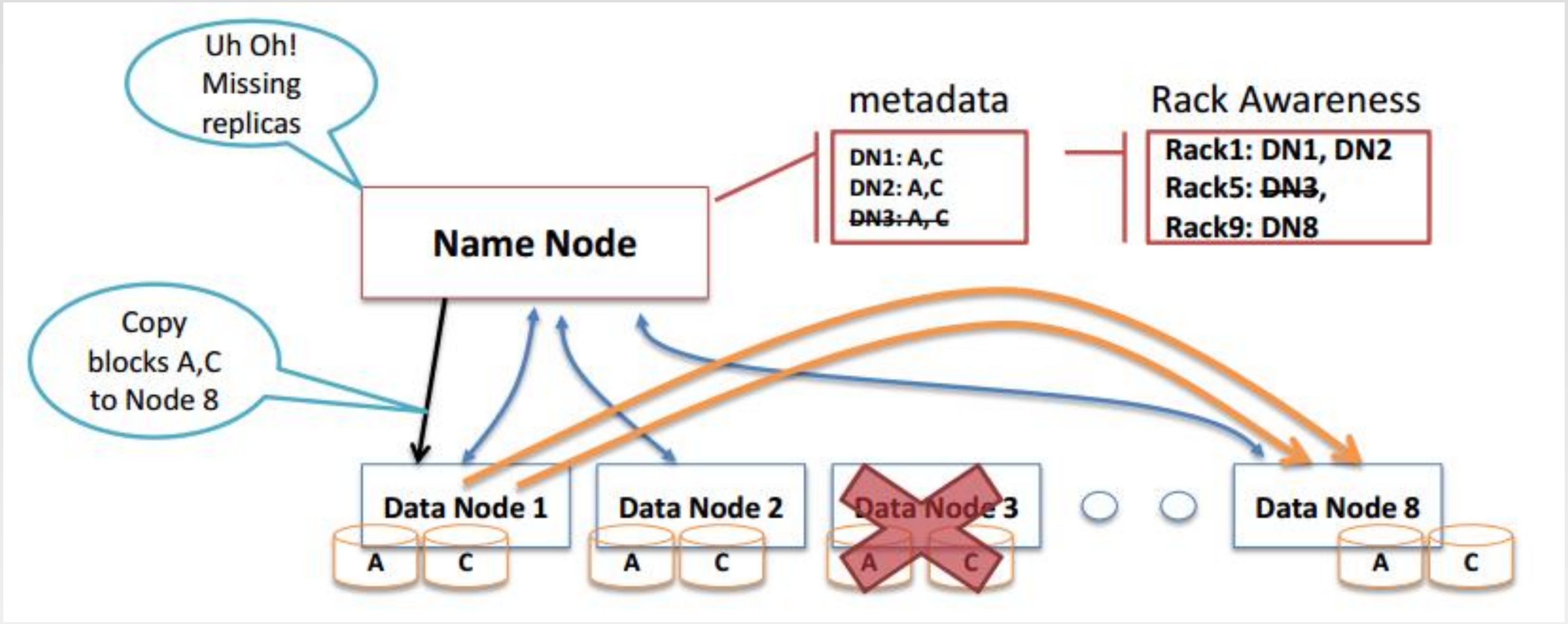
File Read Data Flow in HDFS



File Write Data Flow in HDFS



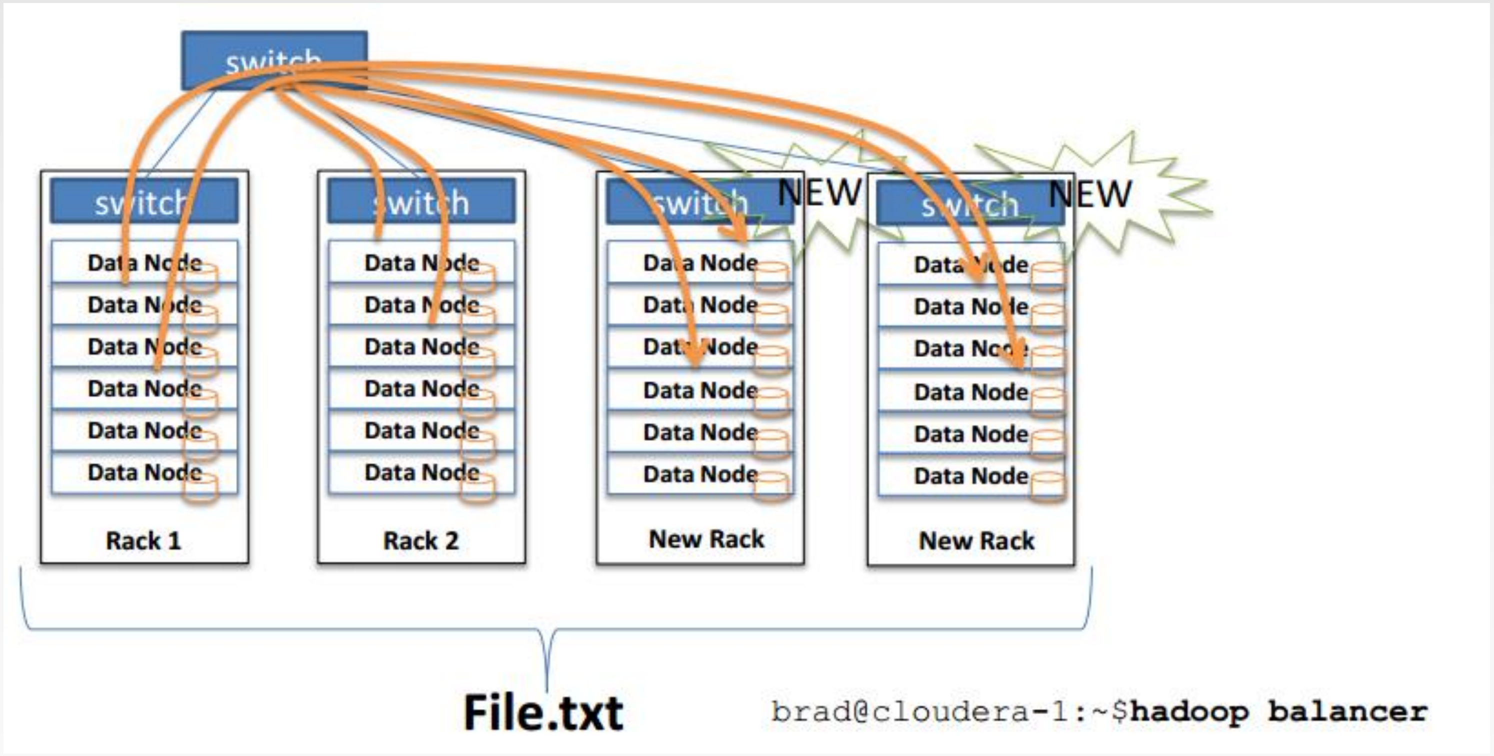
Replication Engine



- NameNode detects DataNode failures
  - Missing Heartbeats signify lost Nodes
  - NameNode consults metadata, finds affected data
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

Cluster Rebalancing

- Goal: % disk full on DataNodes should be similar
  - Usually run when new DataNodes are added
  - Rebalancer is throttled to avoid network congestion
  - Does not interfere with MapReduce or HDFS
  - Command line tool



```
brad@cloudera-1:~$hadoop balancer
```

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.

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.
  - 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

HDFS Erasure Coding

- Replication is expensive – the default 3x replication scheme in HDFS has 200% overhead in storage space and other resources.

- Therefore, a natural improvement is to use Erasure Coding (EC) in place of replication, which provides the same level of fault-tolerance with much less storage space.
  - Erasure Coding transforms a message of *k* symbols into a longer message with *n* symbols such that the original message can be recovered from a subset of the *n* symbols.
  - In typical Erasure Coding (EC) setups, the storage overhead is no more than 50%.

https://en.wikipedia.org/wiki/Erasure_code

Unique features of HDFS

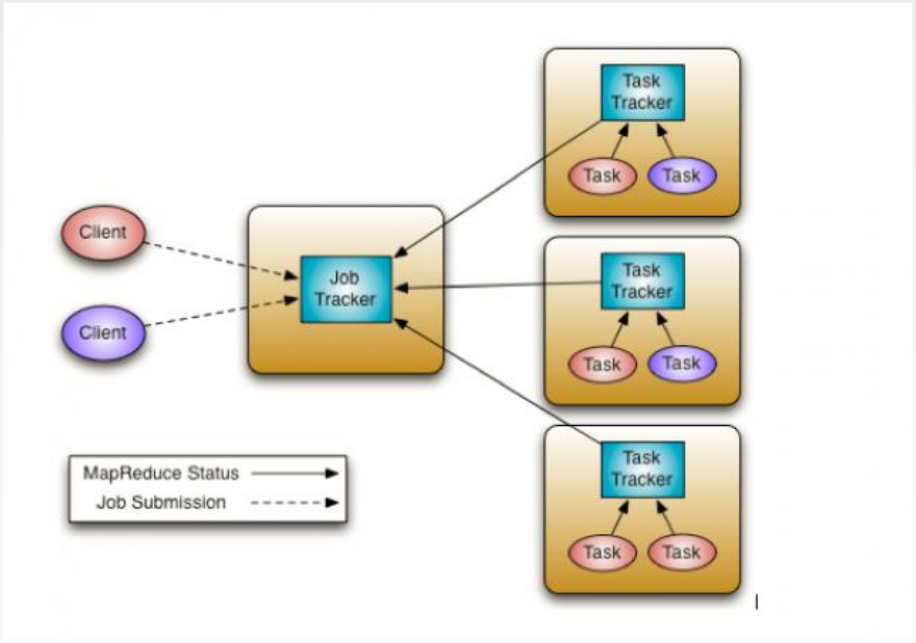
- HDFS has a bunch of unique features that make it ideal for distributed systems:
  - Failure tolerant - data is duplicated across multiple DataNodes to protect against machine failures. The default is a replication factor of 3 (every block is stored on three machines).
  - Scalability - data transfers happen directly with the DataNodes so your read/write capacity scales fairly well with the number of DataNodes
  - Space - need more disk space? Just add more DataNodes and re-balance
  - Industry standard - Other distributed applications are built on top of HDFS (HBase, MapReduce)

- HDFS is designed to process large data sets with write-once-read-many semantics, it is not for low latency access

Part 3: YARN

Why YARN

- In Hadoop version 1, MapReduce performed both processing and resource management functions.
  - It consisted of a Job Tracker which was the single master. The Job Tracker allocated the resources, performed scheduling and monitored the processing jobs.
  - It assigned map and reduce tasks on a number of subordinate processes called the Task Trackers. The Task Trackers periodically reported their progress to the Job Tracker.



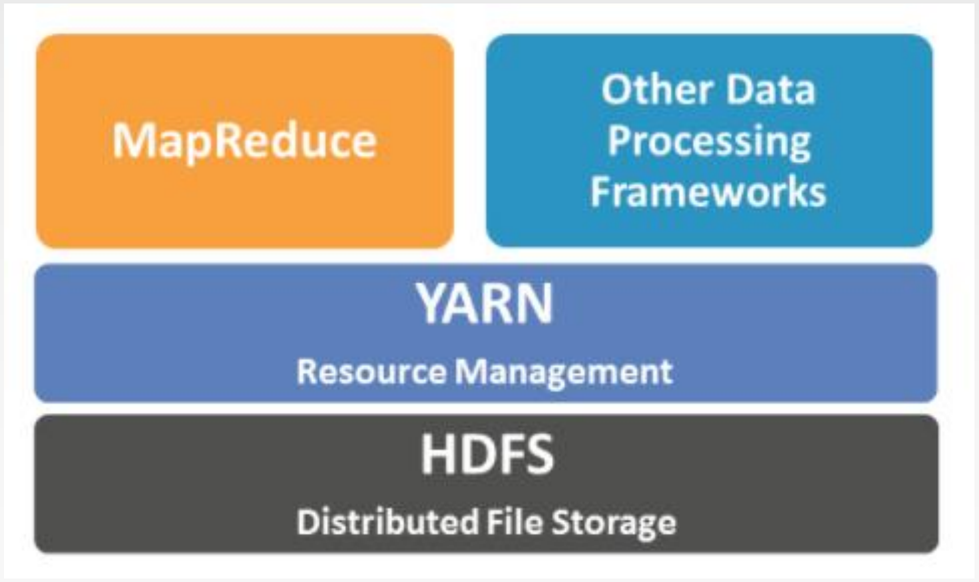
What is YARN

- YARN - “**Y**et **A**nother **R**esource **N**egotiator”
  - The resource management layer of Hadoop, introduced in Hadoop 2.x
  - Monitors and manages workloads, maintains a multi-tenant environment, manages the high availability features of Hadoop, and implements security controls

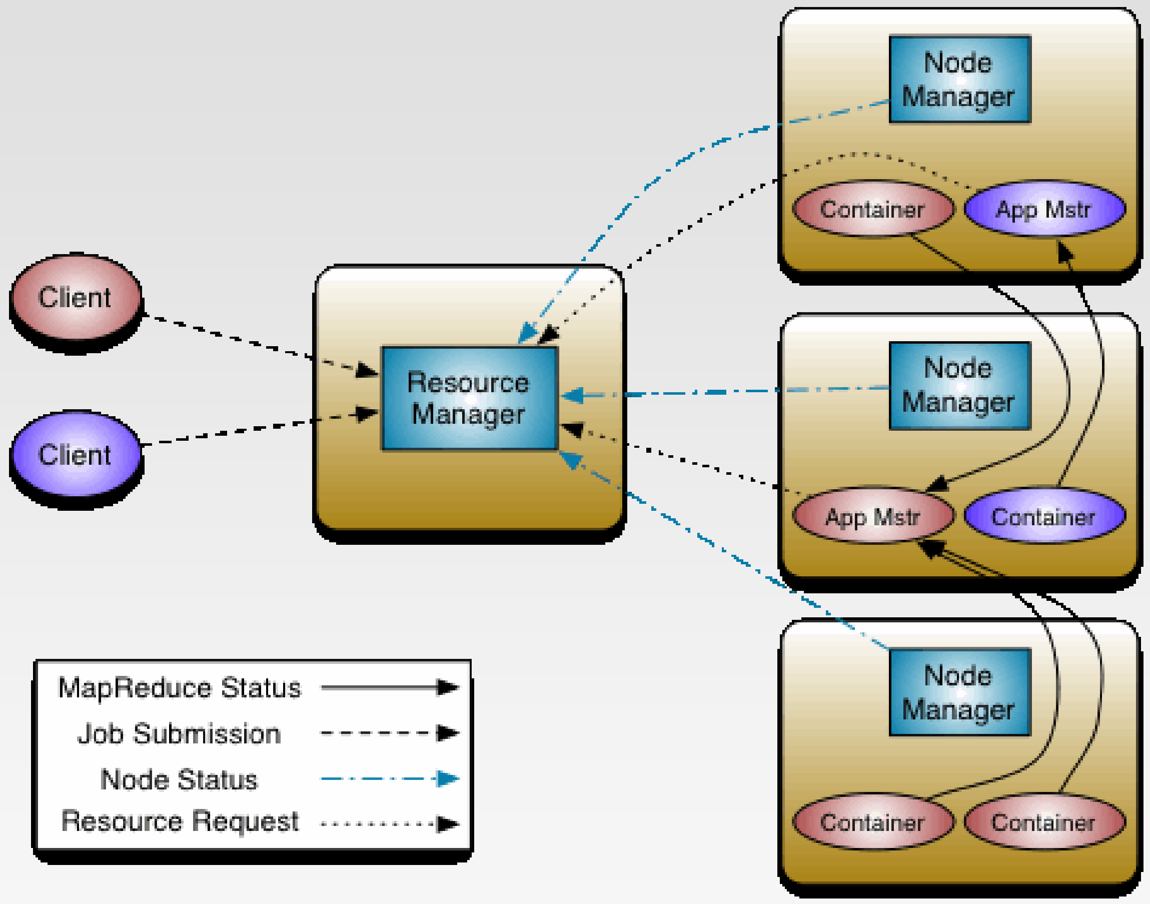
- Motivation:
  - Flexibility - Enabling data processing model more than MapReduce
  - Efficiency - Improving performance and QoS
  - Resource Sharing - Multiple workloads in cluster

What is YARN

- YARN was introduced in Hadoop version 2.0 in the year 2012 by Yahoo and Hortonworks.
- The basic idea behind YARN is to relieve MapReduce by taking over the responsibility of Resource Management and Job Scheduling.
- YARN enabled the users to perform operations as per requirement by using a variety of tools like Spark for real-time processing, Hive for SQL, HBase for NoSQL and others.



YARN Framework



YARN Components

- ResourceManager
  - Arbitrates resources among all the applications in the system
- ApplicationMaster
  - A framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks
- NodeManager
  - The per-machine framework agent who is responsible for containers, monitoring their resource usage (cpu, memory, disk, network) and reporting the same to the ResourceManager
- Container
  - Unit of allocation incorporating resource elements such as memory, cpu, disk, network etc, to execute a specific task of the application

Application Workflow in YARN

- Execution Sequence
  - 1. A client program submits the application
  - 2. ResourceManager allocates a specified container to start the ApplicationMaster
  - 3. ApplicationMaster, on boot-up, registers with ResourceManager
  - 4. ApplicationMaster negotiates with ResourceManager for appropriate resource containers
  - 5. On successful container allocations, ApplicationMaster contacts NodeManager to launch the container
  - 6. Application code is executed within the container, and then ApplicationMaster is responded with the execution status
  - 7. During execution, the client communicates directly with ApplicationMaster or ResourceManager to get status, progress updates etc.
  - 8. Once the application is complete, ApplicationMaster unregisters with ResourceManager and shuts down, allowing its own container process

References

- HDFS Architecture. https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html
- Understanding Hadoop Clusters and the Network. https://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/
- YARN tutorial. https://www.edureka.co/blog/hadoop-yarn-tutorial/

End of Chapter 1.2