Inspire…Educate…Transform.

Applying ML to Big Data using Hadoop and Spark Ecosystem

Day 1: Big Data Overview, Hadoop Ecosystem, HDFS

Thomas K. John
(Slides borrowed from Dr. Manish Gupta)

What is big data?

Volume: How huge is data?

Multiples of bytes		
SI decimal prefixes		Binary usage
Name (Symbol)	Value	
kilobyte (kB)	10^3	2^{10}
megabyte (MB)	10^6	2^{20}
gigabyte (GB)	10^9	2^{30}
terabyte (TB)	10^{12}	2^{40}
petabyte (PB)	10^{15}	2^{50}
exabyte (EB)	10^{18}	2^{60}
zettabyte (ZB)	10^{21}	2^{70}
yottabyte (YB)	10^{24}	2^{80}

YouTube
>Billion users
72 hours of video per minute
~200 – 350K servers

Microsoft Office 365
~10-20 Million users
Online documents/ppts/excel
~100K servers

CERN
Worlds biggest machine
LHC has 27 km circumference
30 PetaBytes of data in 2012
~100K servers

http://home.cern/about/computing
https://www.youtube.com/yt/press/statistics.html
http://windowsitpro.com/blog/office-365-numbers-ever-increasing-trajectory

“Scale” of data in today’s world

- Google processes 3.5 billion requests per day, storing 10 Exabytes of data.
- Amazon hosts as many as 1,400,000 servers.
- Amazon Web Services (AWS) are used by 60,000 companies and field more than 650,000 requests every second.
- Facebook collects 500 terabytes of data daily, including 2.5 billion pieces of content, 2.7 billion likes and 300 million photos.
- 90% of all the data in the world was produced in the last 2 years.
- It is estimated that 40 zettabytes (40,000 Exabytes) of data will be created by 2020. 3.2 zettabytes [in 2014]
- The total amount of data being captured and stored by industry doubles every 1.2 years

Azure case study

2016

>90,000

New Azure customer subscriptions/month

1.4 Million

SQL databases in Azure

>50 Trillion

Storage objects in Azure

425 Million

Azure Active Directory Users

10 April 2014 DATA · INFRA · PRODUCTION ENGINEERING

Scaling the Facebook data warehouse to 300 PB

 Pamela Vagata  Kevin Wilfong

At Facebook, we have unique storage scalability challenges when it comes to our data warehouse. Our warehouse stores upwards of 300 PB of Hive data, with an incoming daily rate of about 600 TB. In the last year, the warehouse has seen a 3x growth in the amount of data stored. Given this growth trajectory, storage efficiency is and will continue to be a focus for our warehouse infrastructure.

What is Big Data ?

- Big does not refer to size or volume alone !

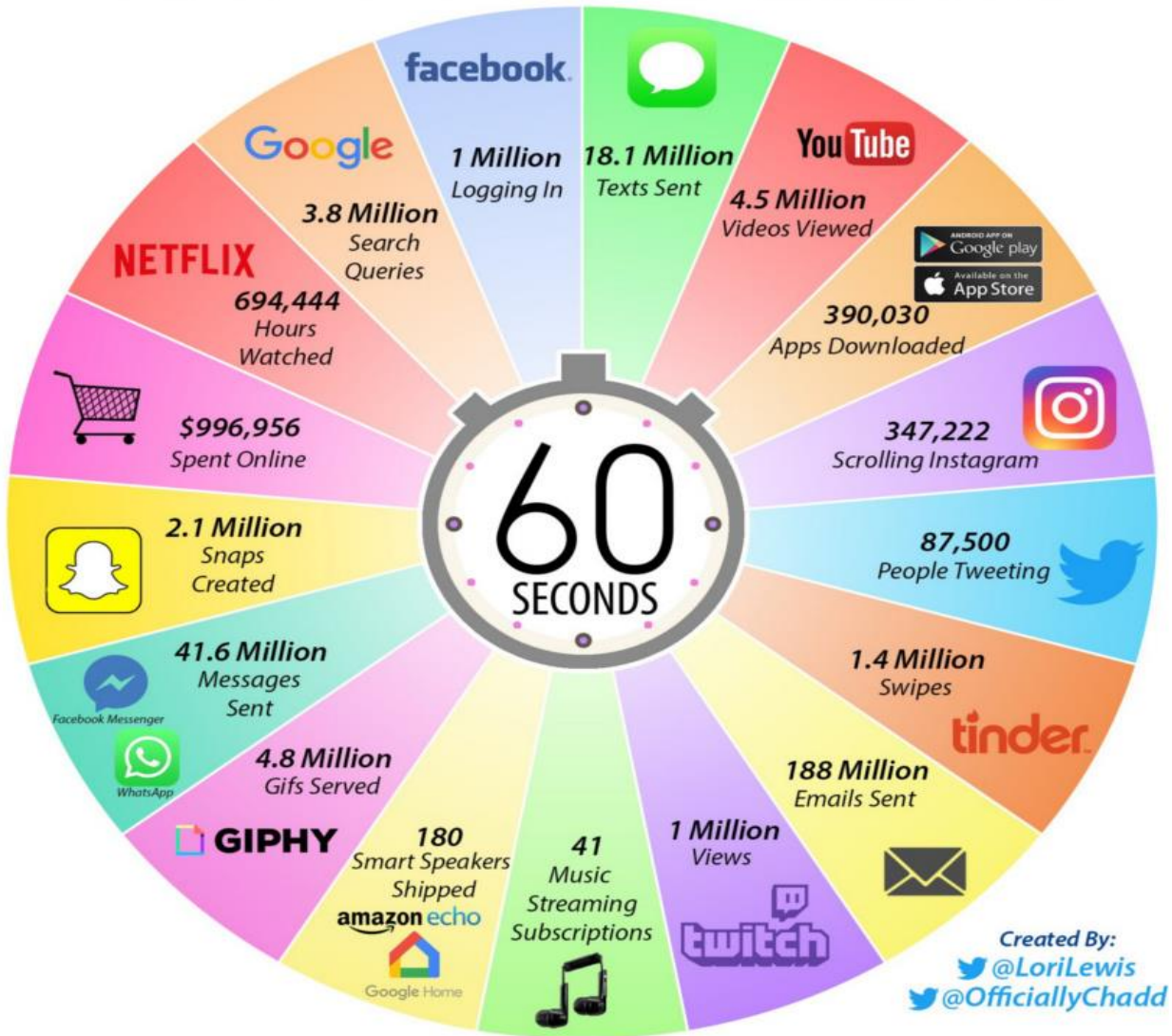
- So what else comes into play?

The V's of Big Data

Velocity

- Speed at which data is generated
- Speed at which it has to be analysed for actionable insights

2019 This Is What Happens In An Internet Minute



2017 This Is What Happens In An Internet Minute

The V's of Big Data

Variety

- Structured
- Semi-structured
- Unstructured

Veracity ?

- Untrusted
- Uncleansed

New Types of Data

Sentiment
Understand how your customers feel about your brand and products – right now

Clickstream
Capture and analyze website visitors' data trails and optimize your website

Sensors
Discover patterns in data streaming automatically from remote sensors and machines

Geographic
Analyze location-based data to manage operations where they occur

Server Logs
Research logs to diagnose process failures and prevent security breaches

Unstructured
Understand patterns in files across millions of web pages, emails, and documents

Gartner, Big Data 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.”

- Volume, Velocity, Variety, Veracity

Why learn about big data technologies?

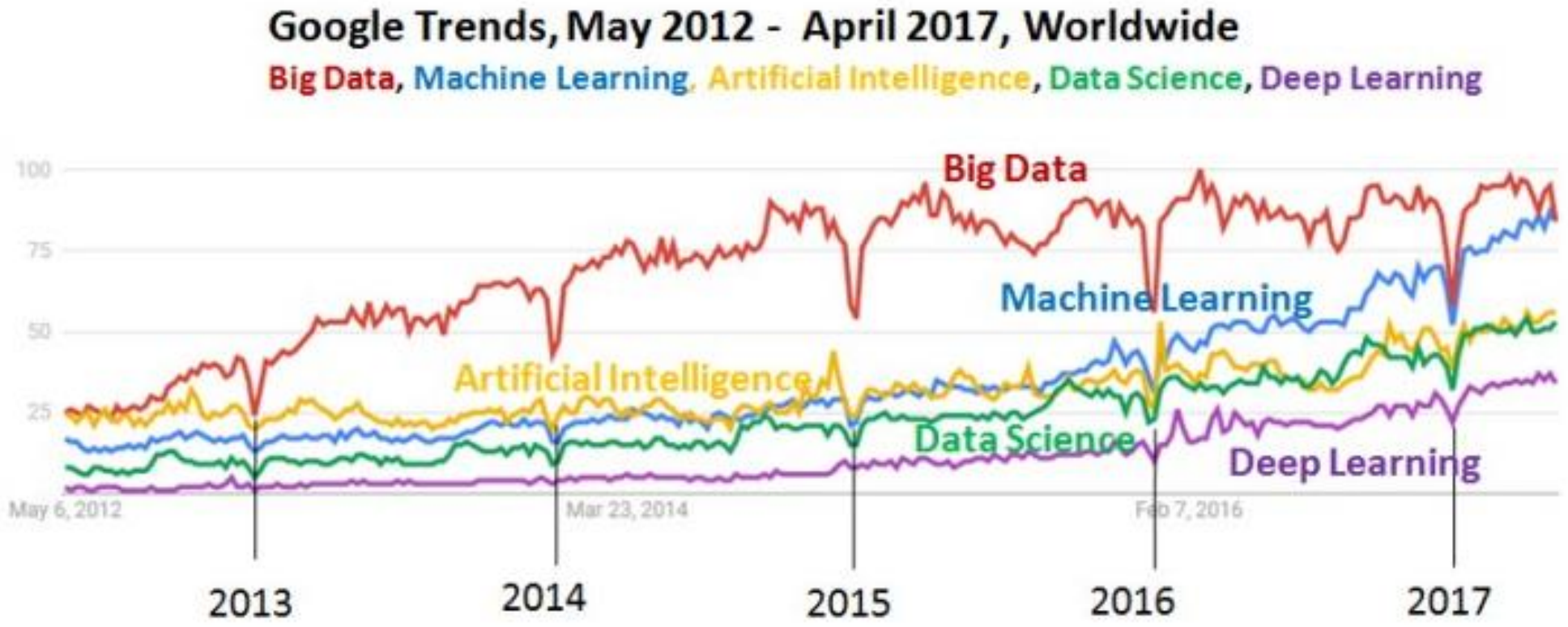
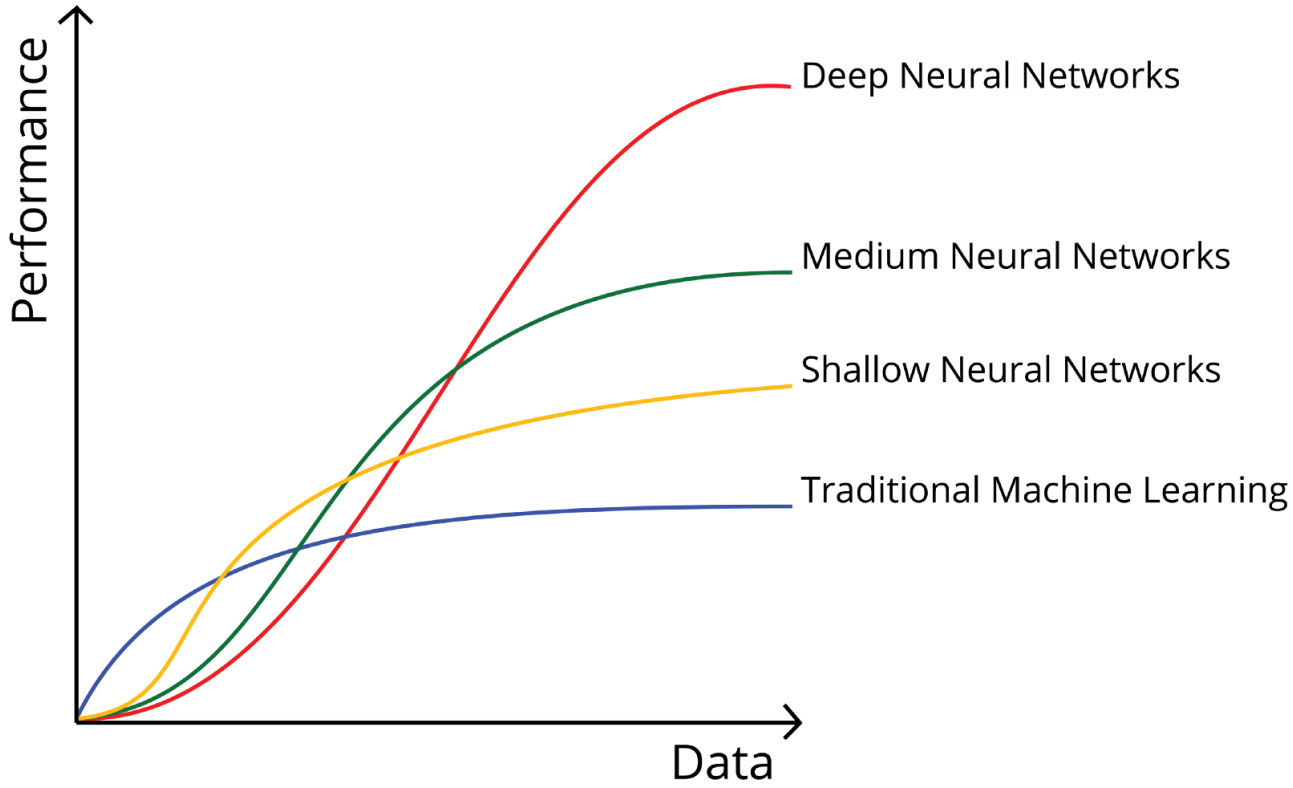


Fig. 3: Google Trends, May 2012 - April 2017, Worldwide
"Big Data" vs "Machine Learning" vs "Artificial Intelligence" vs "Data Science" vs "Deep Learning" search terms.

Accuracy vs Data Size

- The more data you do data science with, the finer (and better) your insights will be.



Big data - model building issues

- Data ought to be in primary storage, or even better, RAM
- Programmers ought to see the storage as monolithic.
  - Resource Management: YARN, Mesos
- “Serially written” programs ought to run in parallel.
  - Map Reduce, MR2, Spark, BSP, Flink, …
- There ought to be faster, more reliable ways of bringing in much more data
  - Data ingestion methods – Sqoop, Flume, Kafka…

Big Data Use-case at CERN

- https://www.youtube.com/watch?v=j-0cUmUyb-Y
- Physicists at CERN have been pondering how to store and share their ever more massive data for decades - stimulating globalization of the internet along the way, whilst 'solving' their big data problem.

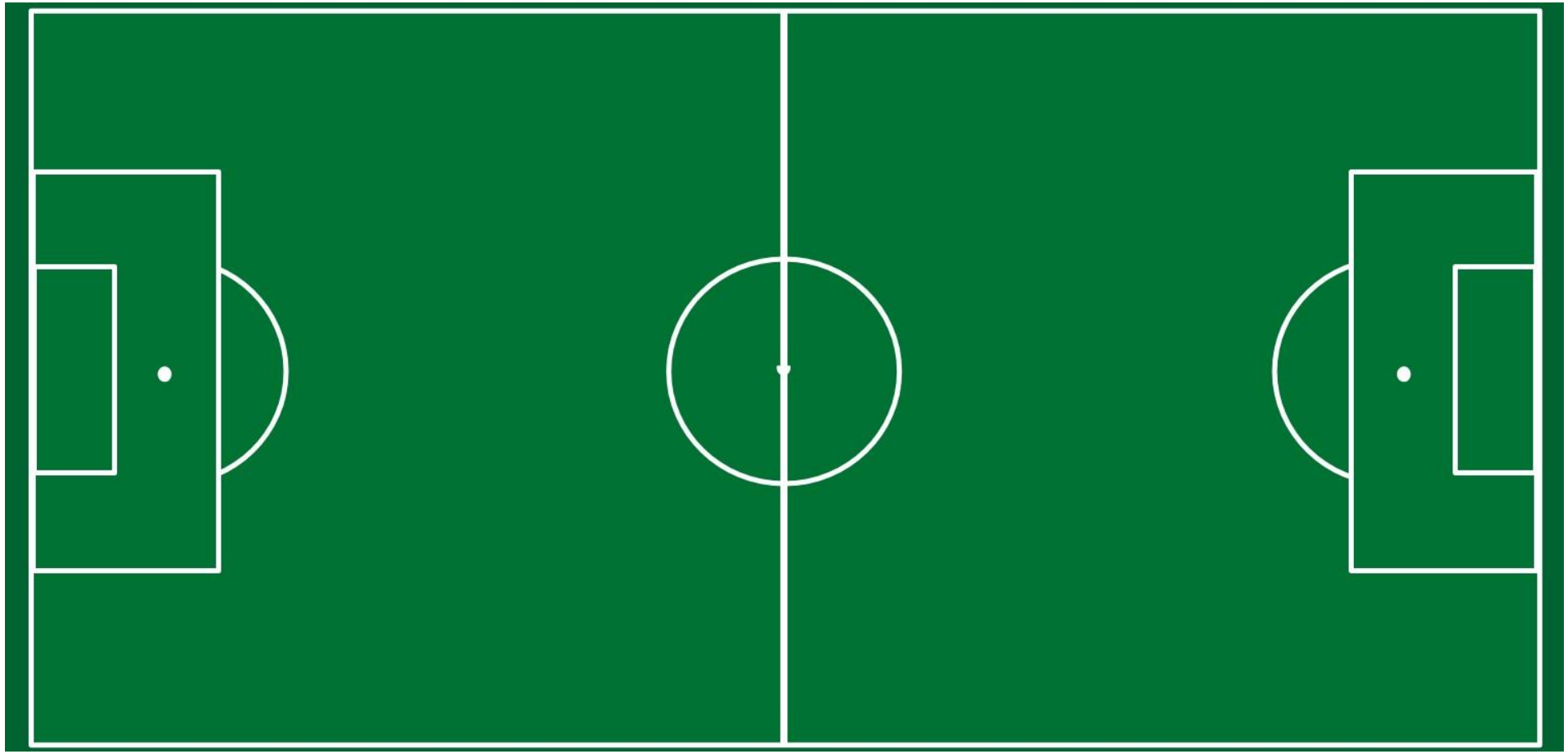
Big data hardware

Ways to Scale ?

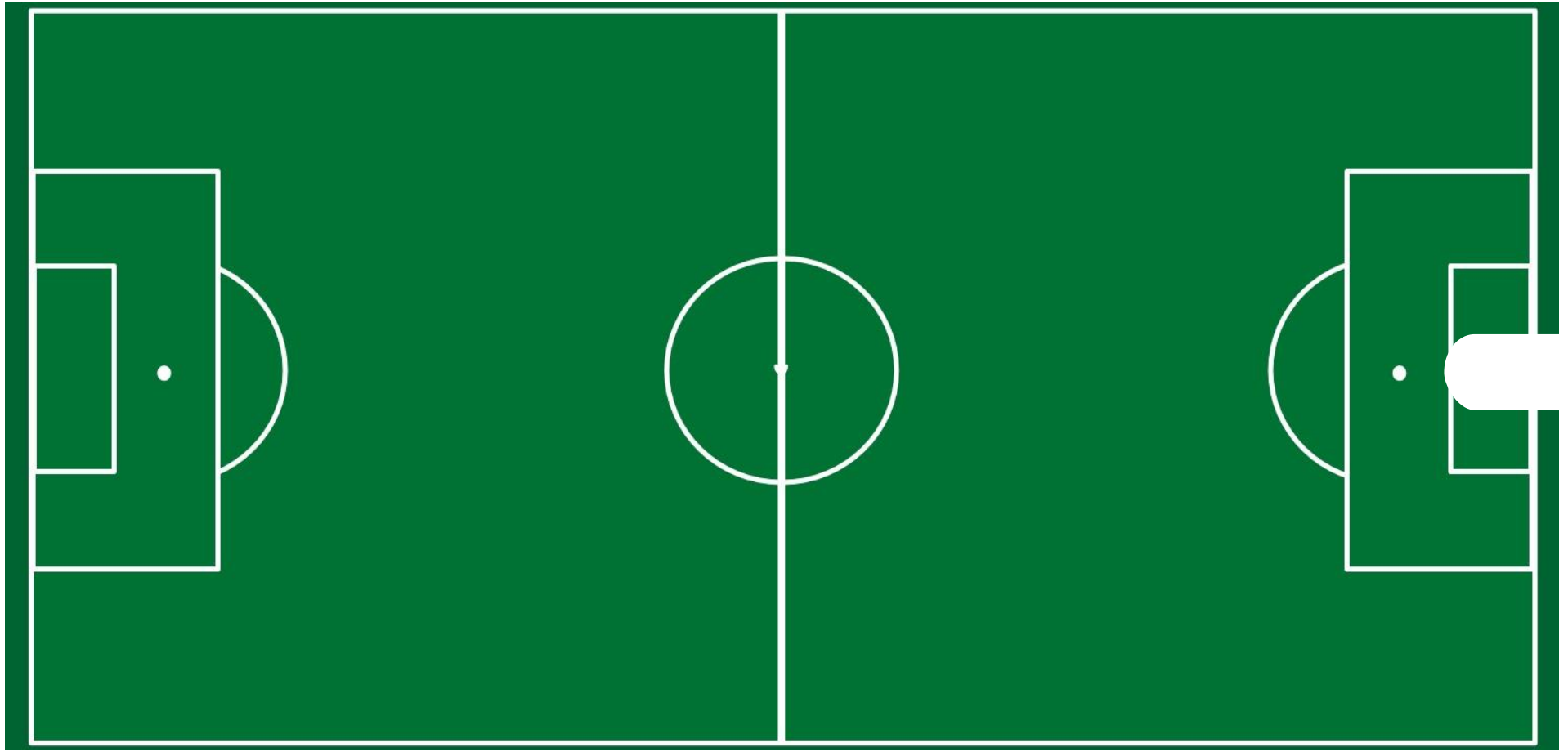
- To **scale horizontally** (or **scale out**) means to add more nodes to a system, such as adding a new computer to a distributed software application.

- To **scale vertically** (or **scale up**) means to add resources to a single node in a system, typically involving the addition of CPUs or memory to a single computer.

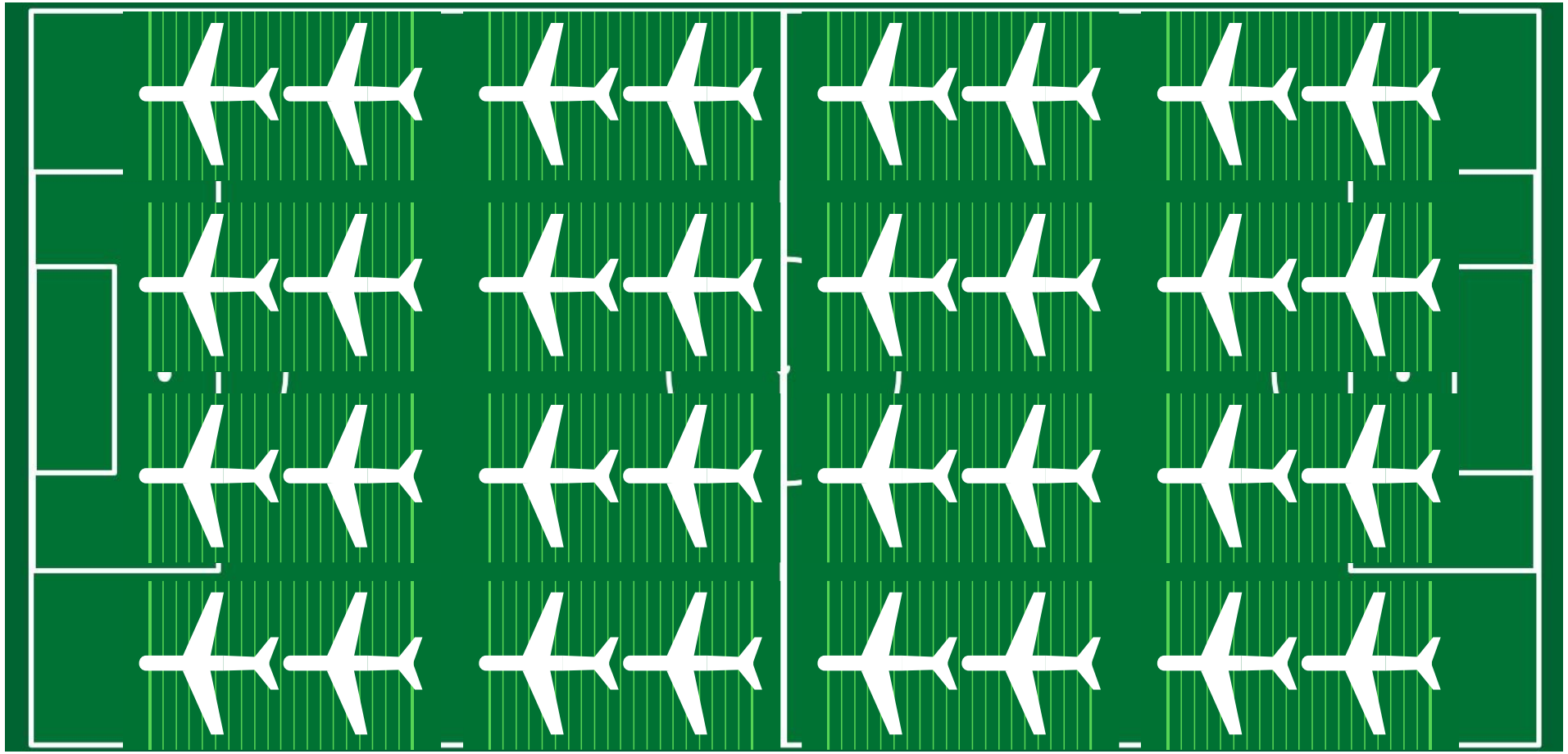
Microsoft Azure data center regions in 2016



Datacenter buildings are about one football field in size



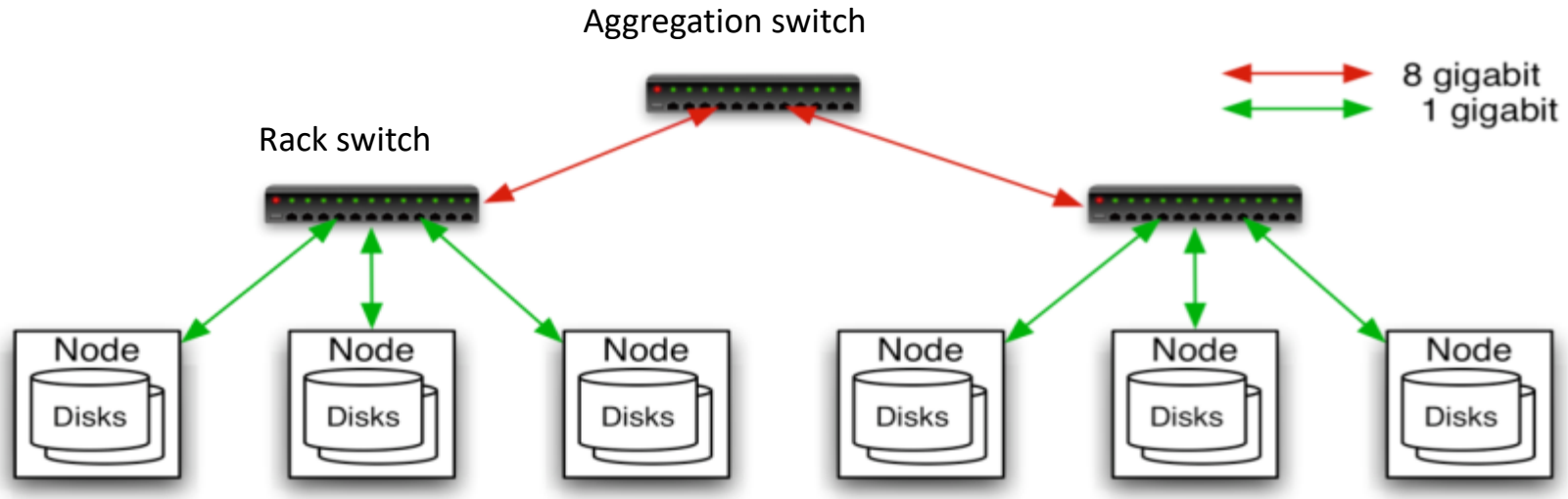
…large enough to hold two jumbo jets



That's up to 600,000 servers in each Azure region

And there are 16 buildings per region…

Hub & Spoke Hardware



- Typically in 2 level architecture
  - Nodes are commodity PCs
  - Typically 30-40 nodes/rack

Rack

The rack contains multiple mounting slots called bays,
each designed to hold a hardware unit secured in place with screws.

A single rack can contain multiple servers stacked one above the other,
consolidating network resources and minimizing the required floor space.

The rack server configuration also simplifies cabling among network components.

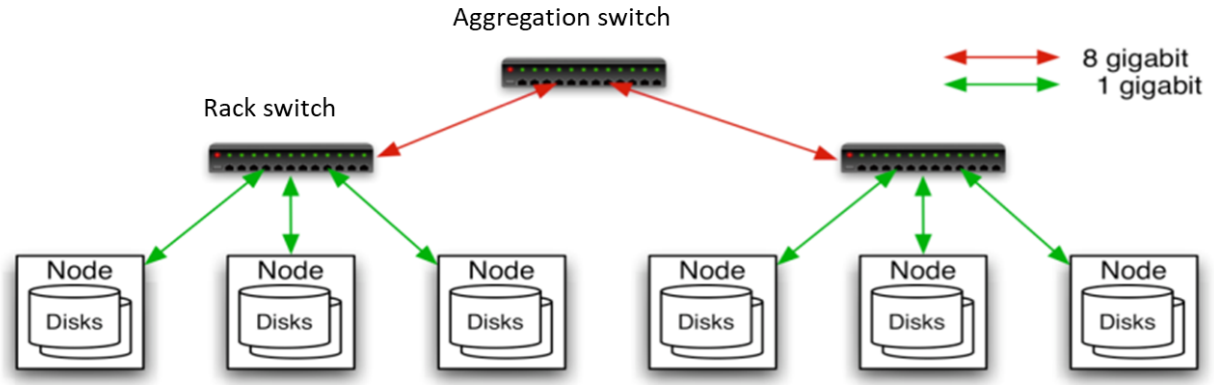
Cooling systems become critical aspects

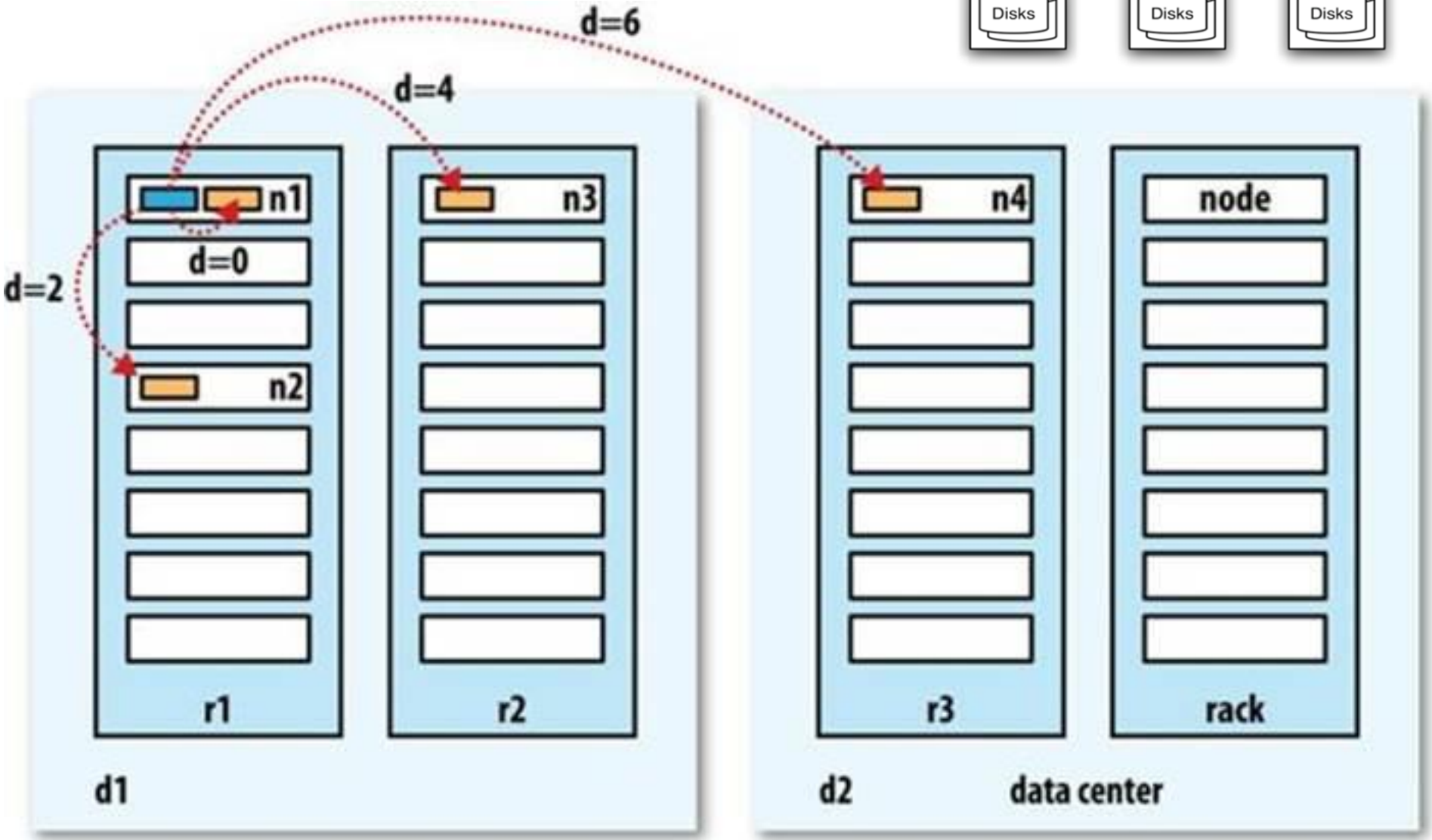
Switch

- A switch, in the context of networking is a high-speed device that receives incoming data packets and redirects them to their destination on a local area network (LAN).

- Essentially, switches are the traffic cops of a simple local area network

- switch is limited to node-to-node communication on the same network.





Notion of Distance [wrt data transfer]

One of the Google data centres powering the cloud

- https://www.youtube.com/watch?v=zDAYZU4A3w0

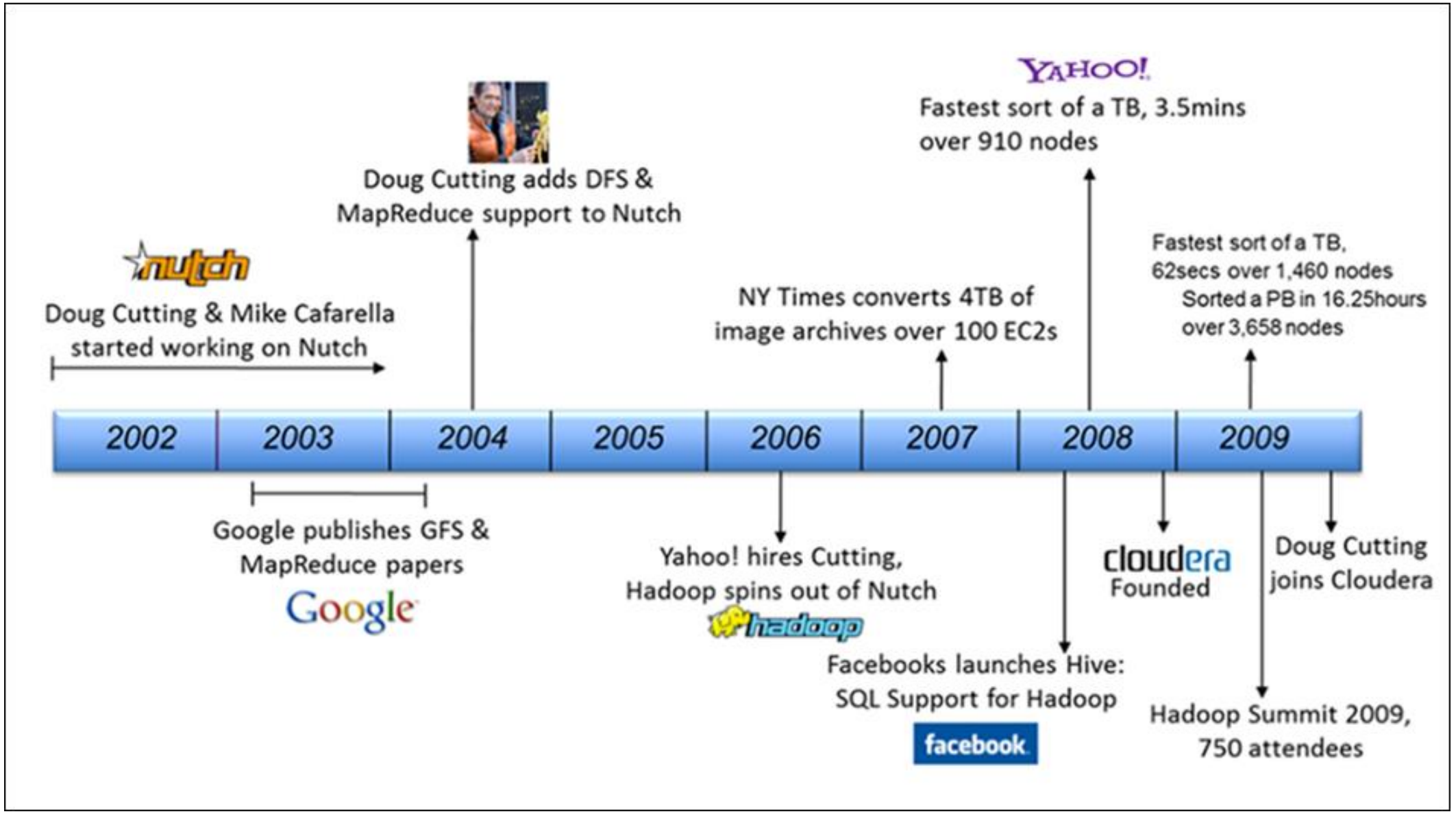
How to use these large clusters nicely?

History of Hadoop

- Hadoop was created by Doug Cutting and Mike Cafarella.

- Originated from an open source web search engine called "Apache Nutch", which is part of another Apache project called "Apache Lucene“

Hadoop Timeline



2011: HW
And
2009: MapR

What is Hadoop?

What is Hadoop?

- Hadoop is an open source framework, from the Apache foundation, capable of processing large amounts of heterogeneous data sets in a distributed fashion across clusters of commodity computers and hardware using a simplified programming model.
- The Hadoop framework is based closely on the following principle:
  - *In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers.*

HDFS and MapReduce

Distributed Storage

HDFS - Reliable Shared Storage

+

MapReduce - Distributed Computation

Parallel processing

=

Hadoop Characteristics:

- Distribute data initially
  - Let processors / nodes work on local data
  - Minimize data transfer over network
  - Replicate data multiple times for increased availability
- Write applications at a high level
  - Programmers should not have to worry about network programming, low level infrastructure, etc
- Minimize talking between nodes (*share-nothing)*

Why Hadoop?

- Scalability: Need to process Multi Petabyte Datasets, quickly.
- Data may not have strict schema
- Expensive to build reliability in each application
- Fault tolerant: Nodes fails everyday
- Need common infrastructure
- Very Large Distributed File System
- Low cost: Assumes Commodity Hardware
- Optimized for Batch Processing
- Runs on heterogeneous OS

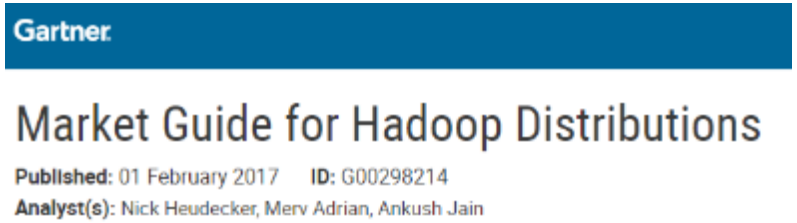
Hadoop eco-system

Key Hadoop Vendors [By Type]

- Pure play Hadoop vendors: Cloudera, Hortonworks, MapR, IBM OpenPlatform, Huawei FusionInsight, Seabox, Transwarp

- Cloud infrastructure as a service (IaaS): Hadoop on AWS, Hadoop on Azure

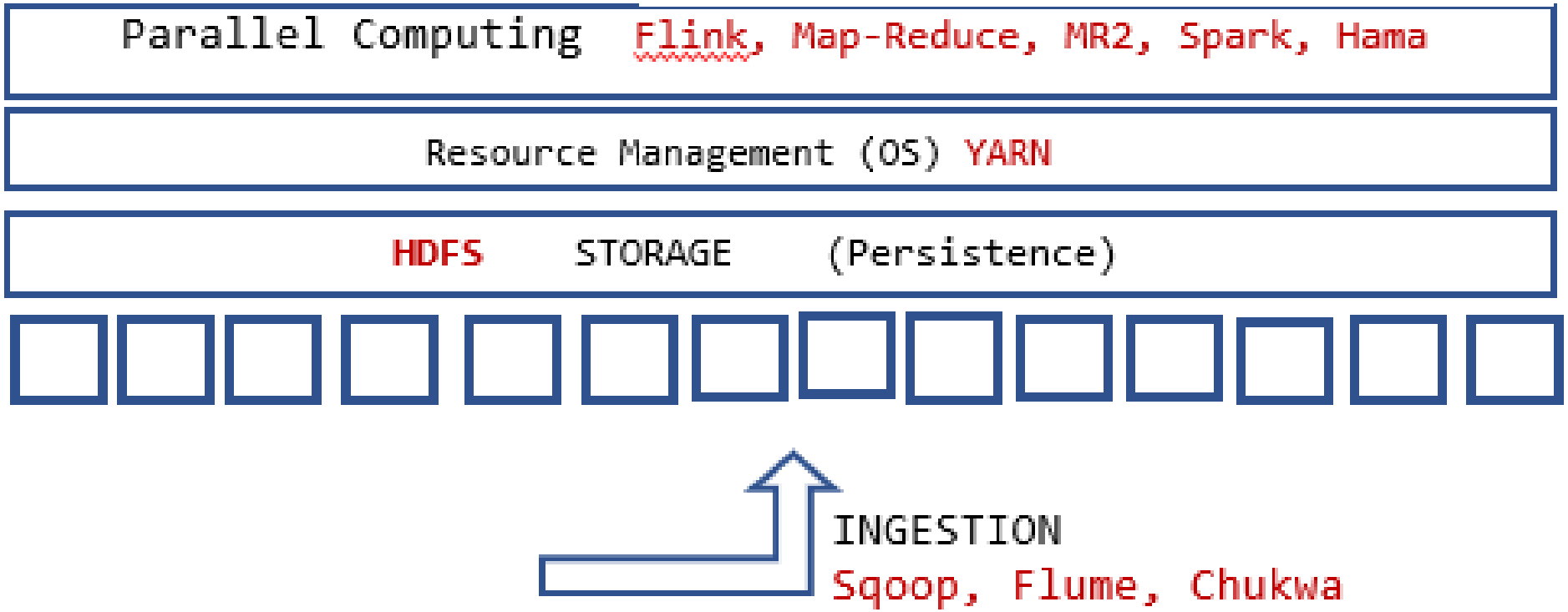
- Platform as a service (PaaS): IBM BigInsights, Microsoft HDInsight, Google Cloud Platform, Amazon EMR, Oracle Big Data Cloud Service, Qbole

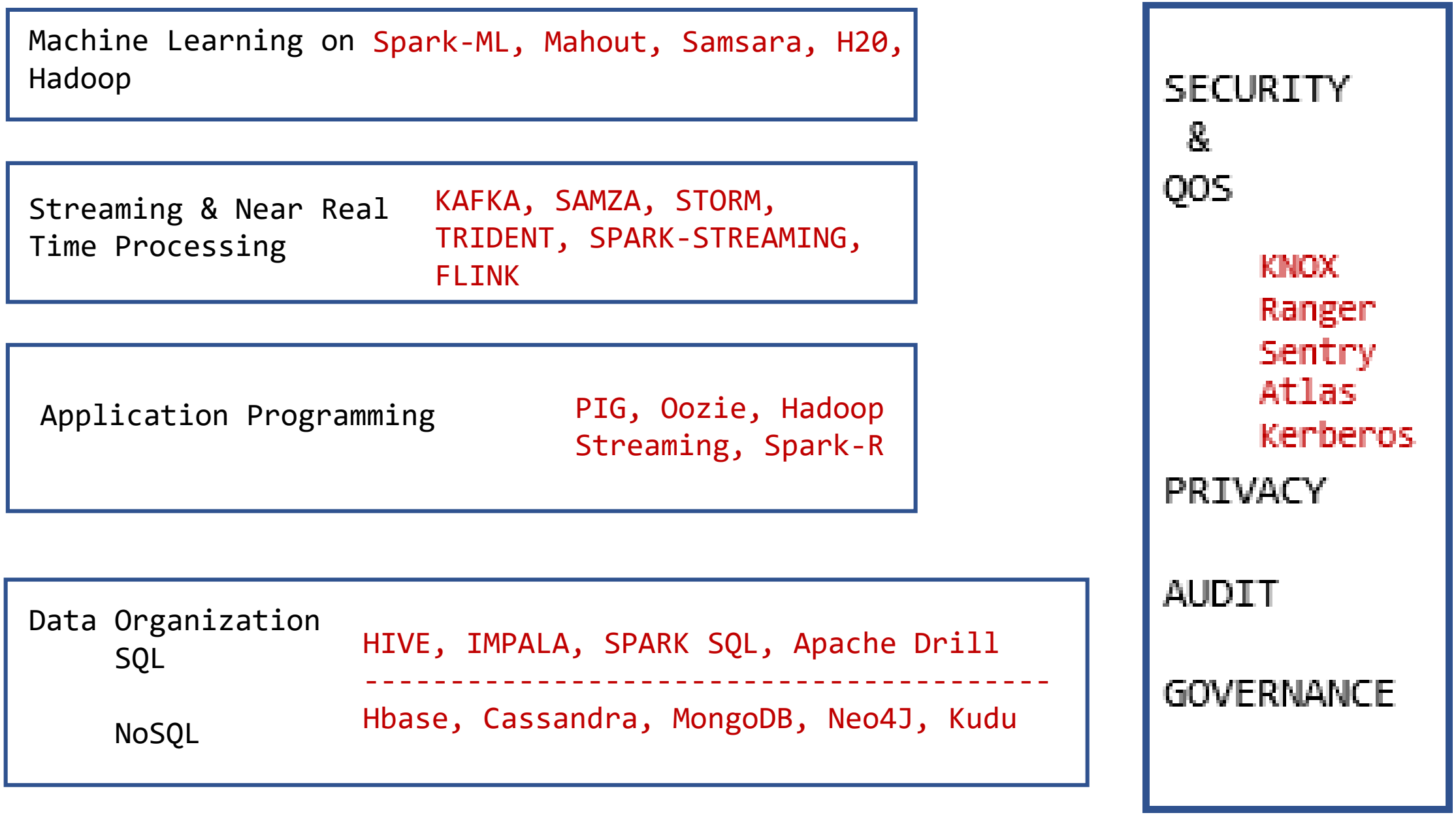


Important Components of Hadoop

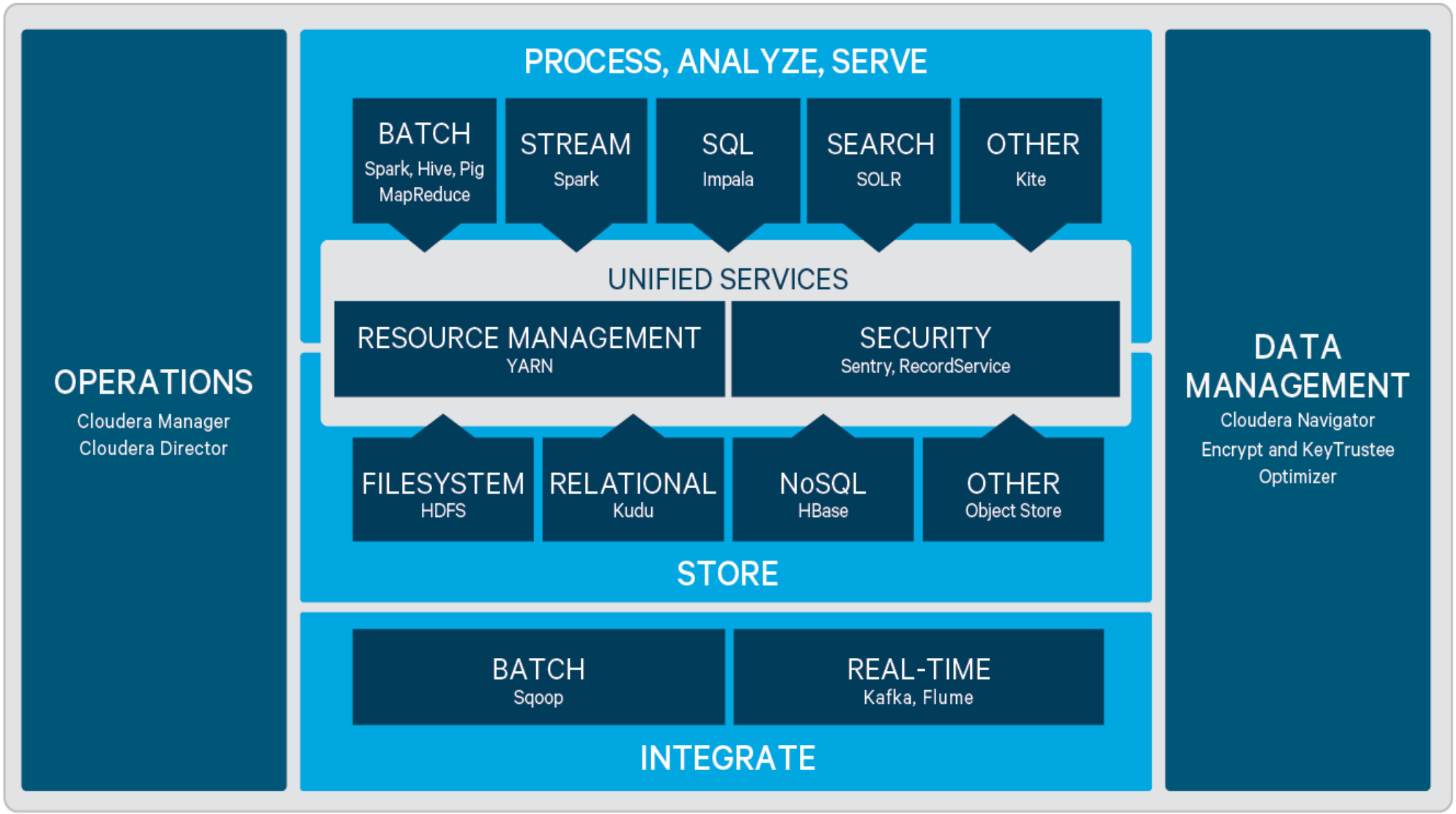
- HDFS - Hadoop Distributed File System
  - Horizontal scalability. Thousands of servers holding petabytes of data.
  - Commodity hardware. HDFS is designed with relatively cheap commodity hardware in mind. HDFS is self-healing and replicating.
  - Fault tolerance. Every component of the Hadoop ecosystem knows how to deal with hardware failures.
- MapReduce
  - MapReduce takes care of distributed computing. It reads the data, usually from its storage, the Hadoop Distributed File System (HDFS), in an optimal way. However, it can read the data from other places too, including mounted local file systems, the web, and databases. It divides the computations between different computers (servers, or nodes). It is also fault-tolerant.
- HBase, the database for Big Data
  - HBase is a database for Big Data, up to millions of columns and billions of rows.
  - It is a key-value database, not a relational database. Key-value databases are considered as more fitting for Big Data. Why? Because they don't store nulls! This gives them the appellation of "sparse."
- ZooKeeper
  - ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
  - ZooKeeper is also fault-tolerant.
- Hive - data warehousing
  - Hive defines a simple SQL-like query language, called QL, that enables users familiar with SQL to query the data.
  - Hive allows you to write custom mappers and reducers to extend the QL capabilities.
- Pig - Big Data manipulation
  - Pig Latin is the language of the stored procedures of Big Data. It allows you to manipulate large volumes of information, analyze them, and create new derivative data sets. Internally it creates a sequence of MapReduce jobs, and thus you can use this simple language to solve pretty sophisticated large-scale problems.

Everyone needs them…..

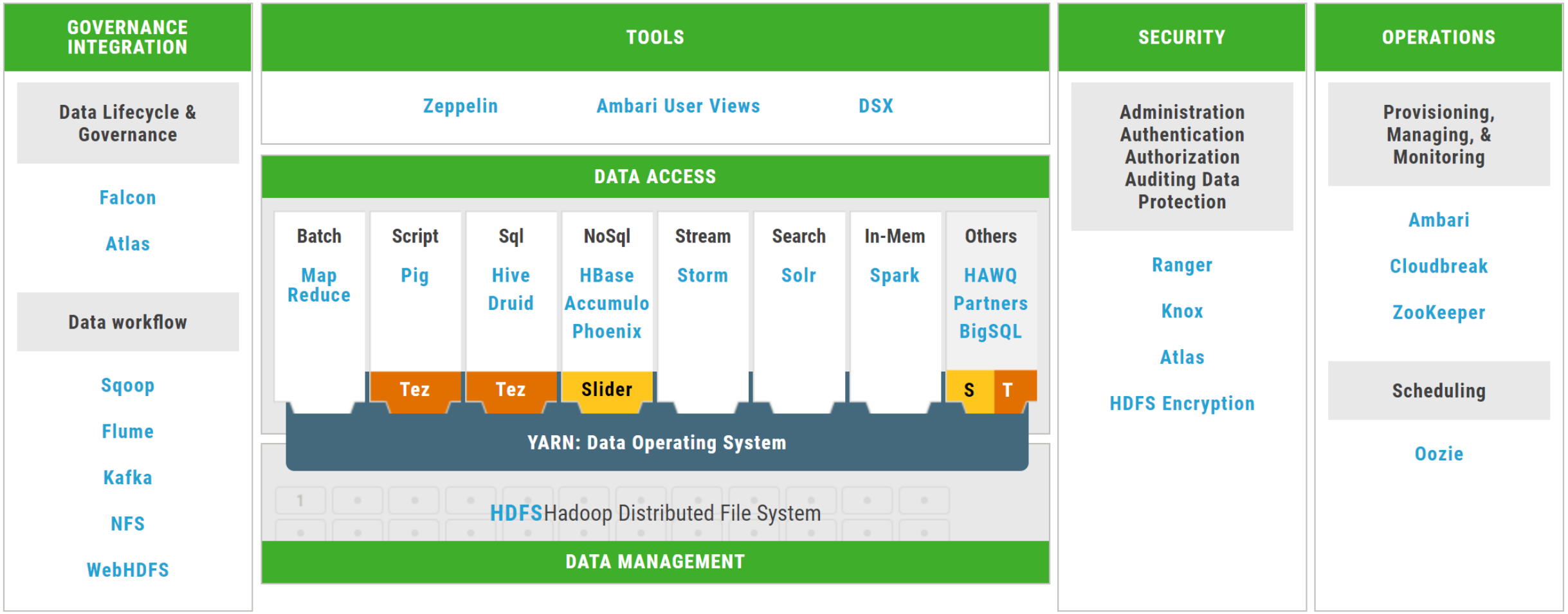



Machine Learning on Spark-ML, Mahout, Samsara, H20, Hadoop
Streaming & Near Real Time Processing
KAFKA, SAMZA, STORM, TRIDENT, SPARK-STREAMING, FLINK
Application Programming
PIG, Oozie, Hadoop Streaming, Spark-R
Data Organization
SQL
HIVE, IMPALA, SPARK SQL, Apache Drill
NoSQL
Hbase, Cassandra, MongoDB, Neo4J, Kudu
SECURITY
&
QOS
KNOX
Ranger
Sentry
Atlas
Kerberos
PRIVACY
AUDIT
GOVERNANCE

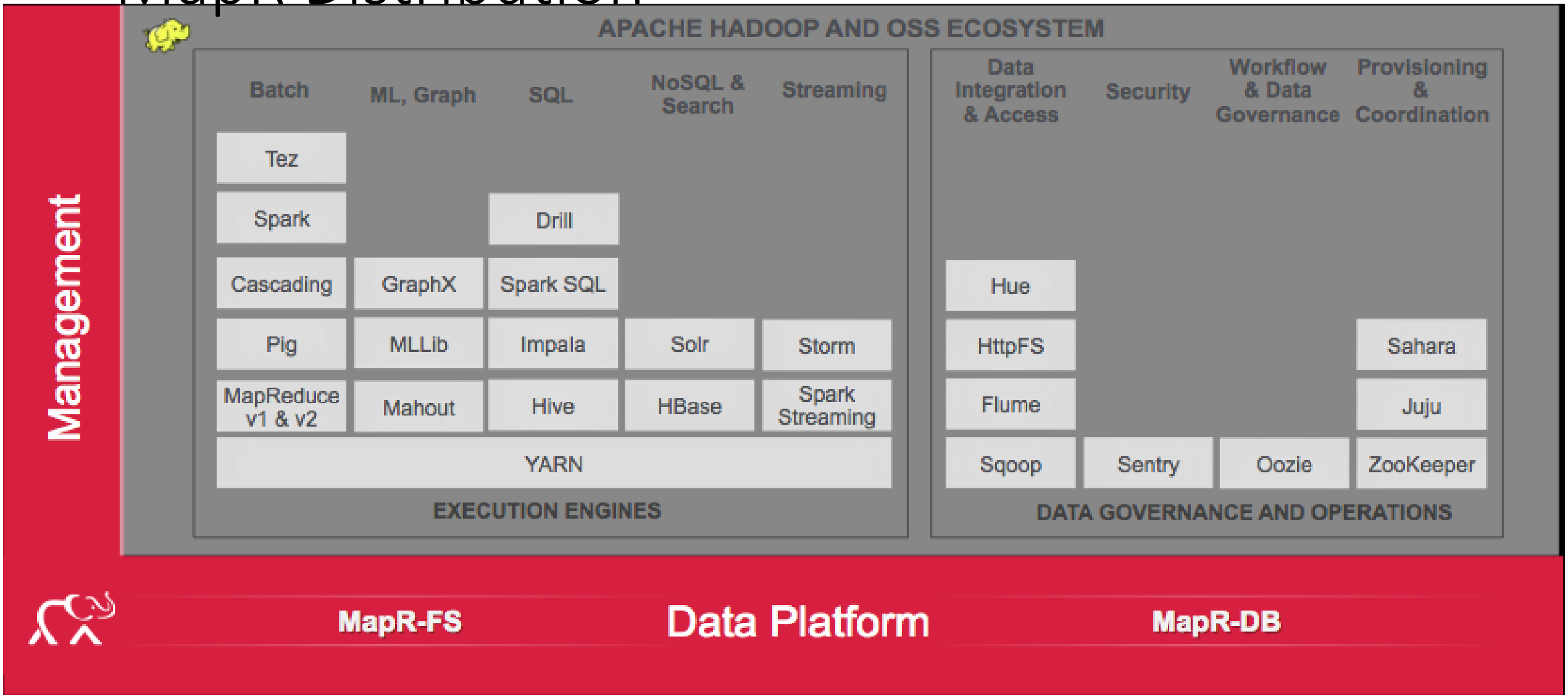
Sample Hadoop Distribution: Cloudera



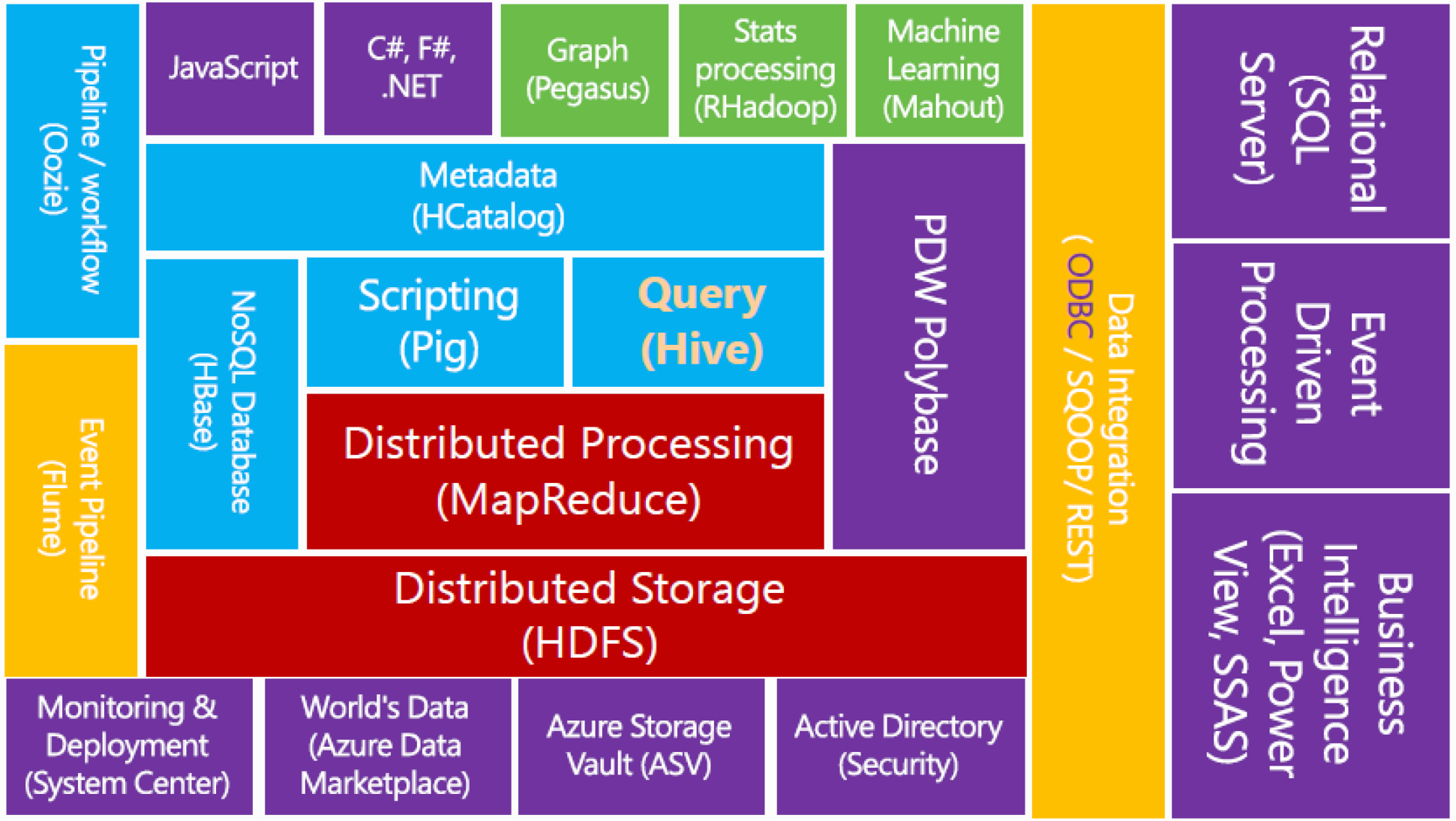
Hortonworks



MapR Distribution



HDINSIGHT / HADOOP Eco-System



Legend

Red = Core Hadoop

Blue = Data processing

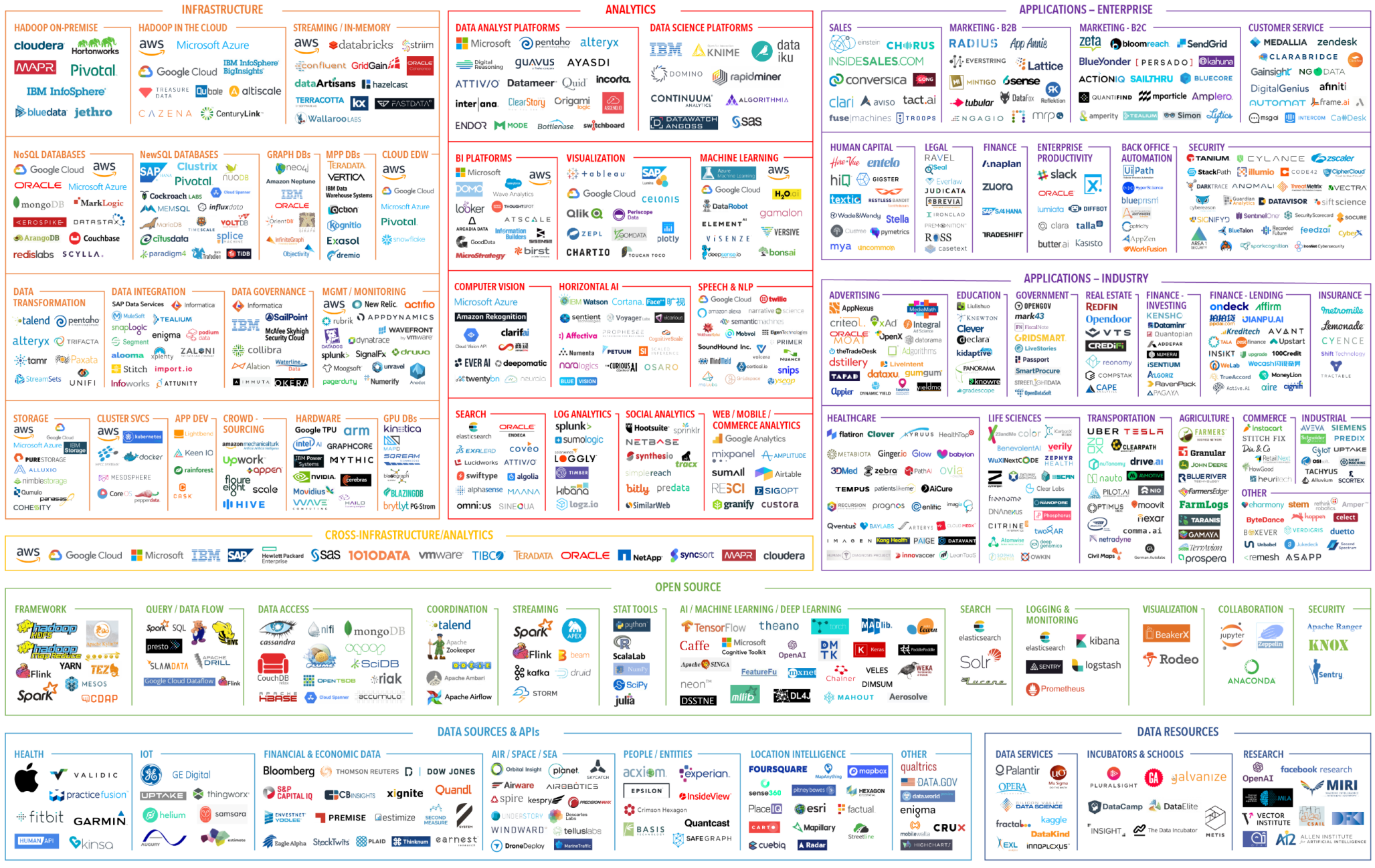
Purple = Microsoft integration points and value adds

Orange = Data Movement

Green = Packages

AWS big data portfolio

BIG DATA & AI LANDSCAPE 2018

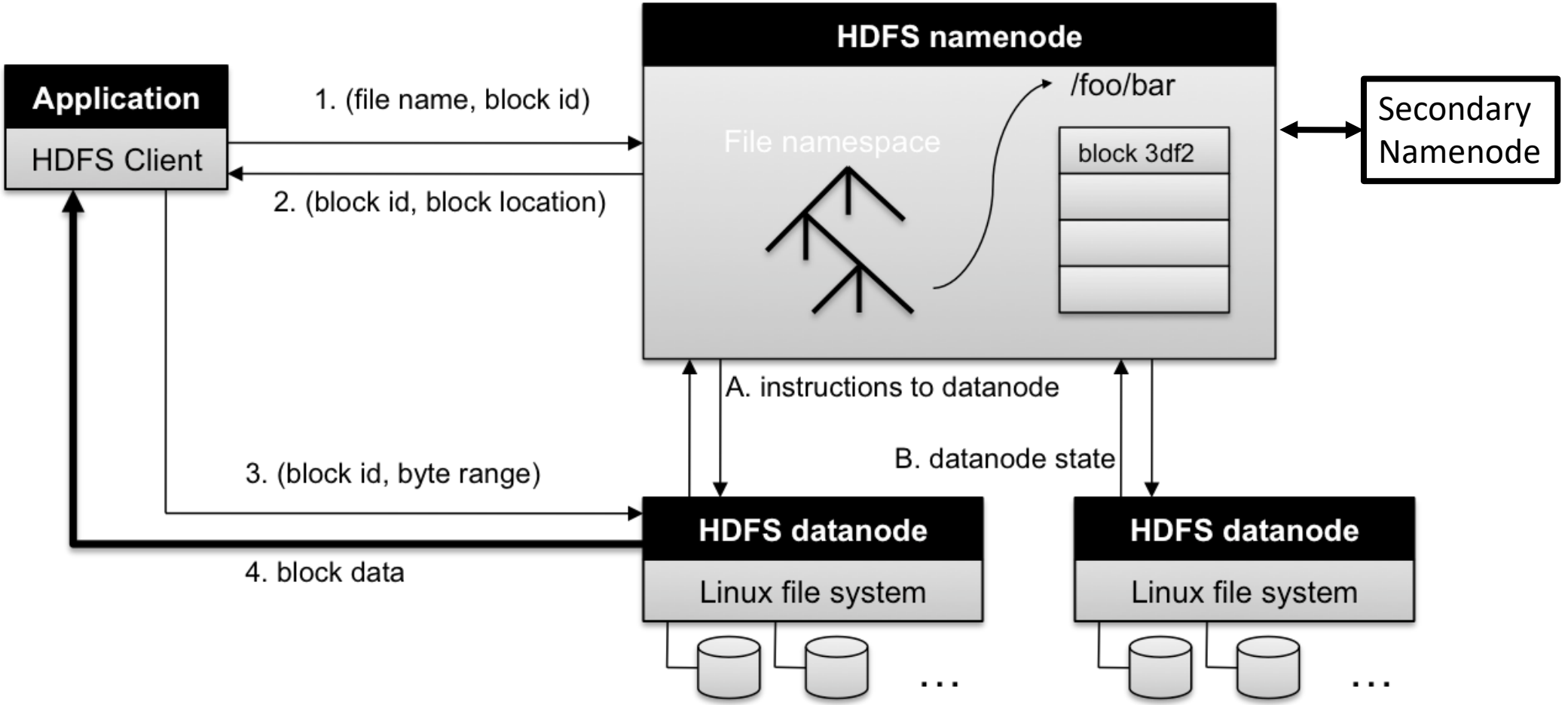


HDFS Architecture

Hadoop Distributed File System (HDFS)

- The file system component of Hadoop
  - Can store very large data sets *reliably*
  - Can stream those data sets at *high bandwidth* to user applications
- R**eplicates file content** on multiple machines (DataNodes)
- **Very Large Distributed File System**
  - 10K nodes, 100 million files, 10 PB
- **Convenient Cluster Management**
  - Load balancing
  - Node failures
  - Cluster expansion
- **Optimized for Batch Processing**
  - Allows to move computation to data
  - Maximizes throughput

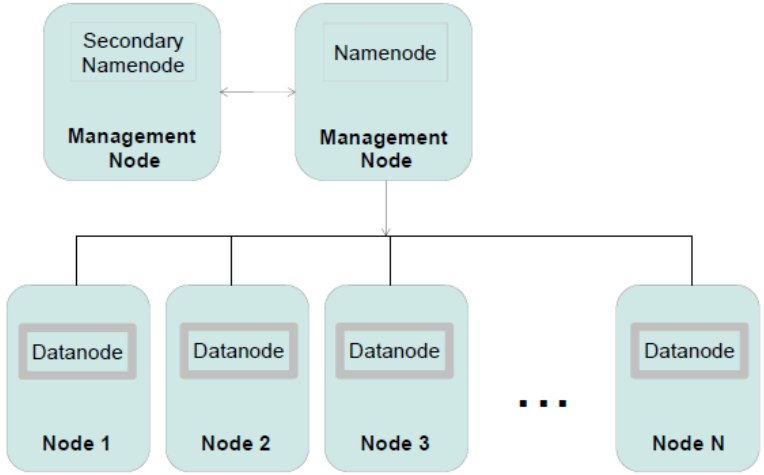
HDFS Architecture



Adapted from (Ghemawat et al., SOSP 2003)

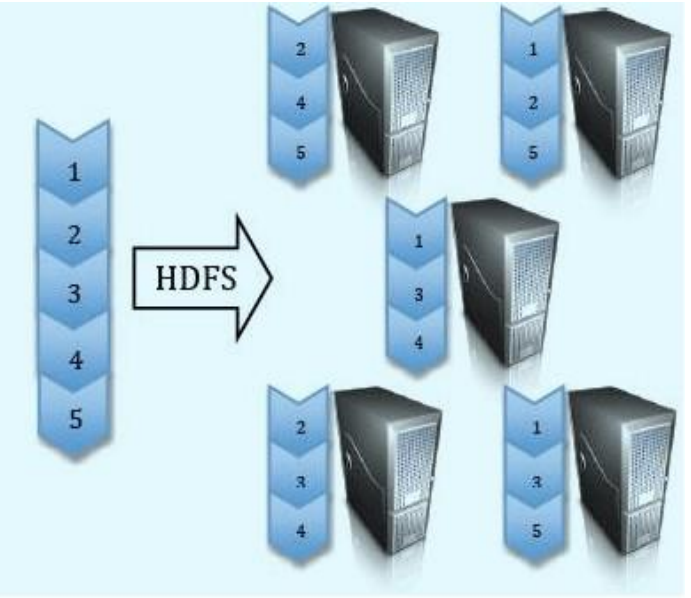
HDFS Components

- *NameNode* manages overall file system metadata
  - Manages the file systems namespace
  - Map from file name to where data is stored, like other file systems
  - Can be a single point of failure in the system
- *DataNodes* (one per machine) store and retrieves actual data
  - DataNodes are easy to add, expanding the file system
  - Each datanode reports to namenode.
  - Acts as a block server: Stores data as blocks and metadata as block checksum. Serves data and metadata to clients.
- Both DataNode and NameNodes include a webserver, so node status can be easily checked
- Secondary namenode: does housekeeping (checkpointing, logging)



Replication and Reliability

- Namenode is “rack aware”: knows how machines are arranged
  - Second replica is on same rack as the first, but different machine
  - Third replica is on a different rack
  - Additional replicas are randomly placed
  - Balances performance (failover time) vs. reliability (independence)
- Namenode does **not** directly read/write data
  - Client gets data location from namenode
  - Namenode ensures that clients read from nearest replicas
  - Client interacts directly with datanode to read/write data
- Namenode keeps all block metadata in (fast) memory
  - Puts constraint on number of files stored: millions of large files



Metadata storage

- HDFS stores file system metadata and application data separately.
- **Metadata** refers to file metadata (attributes such as list of files, permissions, modification, access times, namespace and disk space quotas), list of blocks that belong to the file, list of datanodes for each block.
- Metadata is always stored in main memory.
- HDFS stores metadata on a dedicated server, called the NameNode. (Master) Application data are stored on other servers called DataNodes. (Slaves)
- All servers are fully connected and communicate with each other using TCP-based protocols. (RPC=Remote Procedure Calls)

Responsibilities of a Namenode

- Maintain the namespace tree (a hierarchy of files and directories) operations like opening, closing, and renaming files and directories.
- Determine the mapping of file blocks to DataNodes (the physical location of file data).
- File metadata (i.e. “inode”) .
- Authorization and authentication.
- Collect block reports from Datanodes on block locations.
- Replicate missing blocks.
- HDFS keeps the entire namespace in RAM, allowing fast access to the metadata.

Responsibilities of a DataNode

- The DataNodes are responsible for serving read and write requests from the file system’s clients.
- The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.
- Data nodes periodically send block reports to Namenode.

How do you access HDFS?

HDFS User Interface

- **Java API**
- **Command Line**
  - hadoop dfs -mkdir /foodir
  - hadoop dfs -cat /foodir/myfile.txt
  - hadoop dfs -rm /foodir myfile.txt
  - hadoop dfsadmin -report
  - hadoop dfsadmin -decommission datanodename
  - hadoop dfs –ls
- **Web Interface**
  - http://host:port/dfshealth.jsp

```
hadoop dfs
        [-ls <path>]
        [-du <path>]
        [-cp <src> <dst>]
        [-rm <path>]
        [-put <localsrc> <dst>]
        [-copyFromLocal <localsrc> <dst>]
        [-moveFromLocal <localsrc> <dst>]
        [-get [-crc] <src> <localdst>]
        [-cat <src>]
        [-copyToLocal [-crc] <src> <localdst>]
        [-moveToLocal [-crc] <src> <localdst>]
        [-mkdir <path>]
        [-touchz <path>]
        [-test -[ezd] <path>]
        [-stat [format] <path>]
        [-help [cmd]]
```

Using HDFS file system

- HDFS gives similar control as a traditional file system
  - Paths in the form of directories below a root
  - Can ls (list directory), cat (read file), cd, rm, cp, etc.
  - put: copy file from local file system to HDFS
  - get: copy file from HDFS to local file system
  - File permissions similar to unix/linux
- Some HDFS-specific commands
  - Change file replication level
  - Can rebalance data: ensure datanodes are similarly loaded
  - Java API to read/write HDFS files

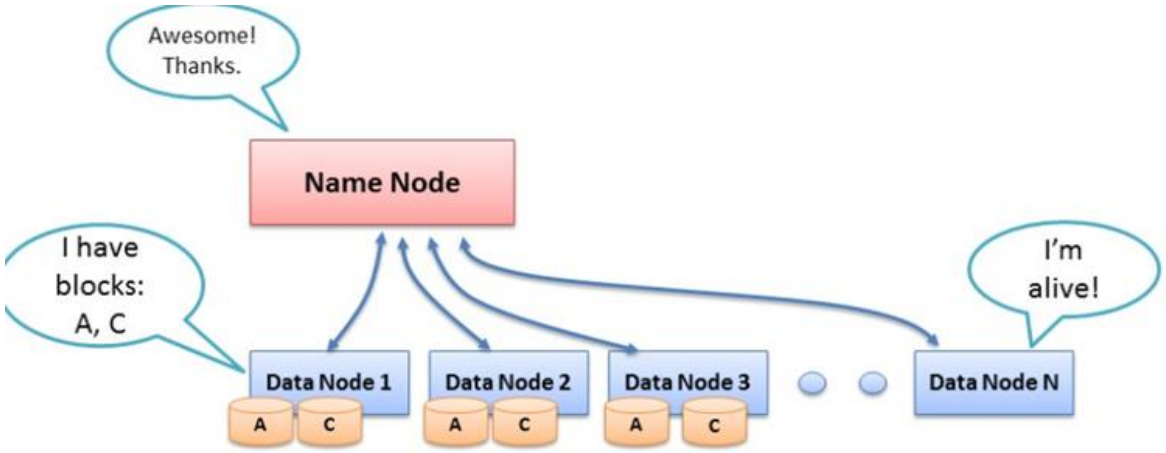
HDFS Data Handling Characteristics

- Write-once-read-many access model
  - Data will be written once, read many times
  - No dynamic updates: append to existing files only
  - Optimize for streaming (sequential) reads, not random access
- Files are broken up into blocks. Typically 128 MB block size. Each block replicated on multiple DataNodes.
- Client: Client can find location of blocks from NameNode. Client accesses data directly from DataNode.

Heartbeats and block reports

Heartbeats and block reports

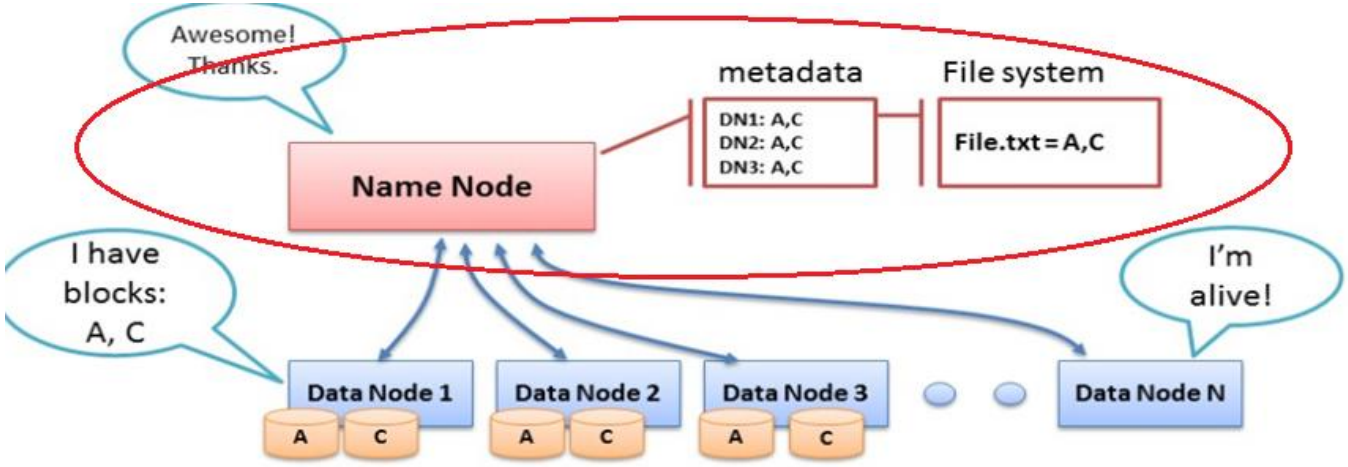
- NameNode and DataNode communication: *Heartbeats.*
- DataNodes send *heartbeats* to the NameNode to confirm that **the DataNode is operating** and **the block replicas it hosts are available.**



- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds

Block Reports

- A DataNode identifies block replicas in its possession to the NameNode by sending a *block report*. A block report contains the ***block id***, the ***generation stamp*** and the **length for each block replica** the server hosts.
- Blockreports provide the NameNode *with an up-to-date view* of where *block replicas* are located on the cluster and nameNode constructs and maintains latest metadata from blockreports.



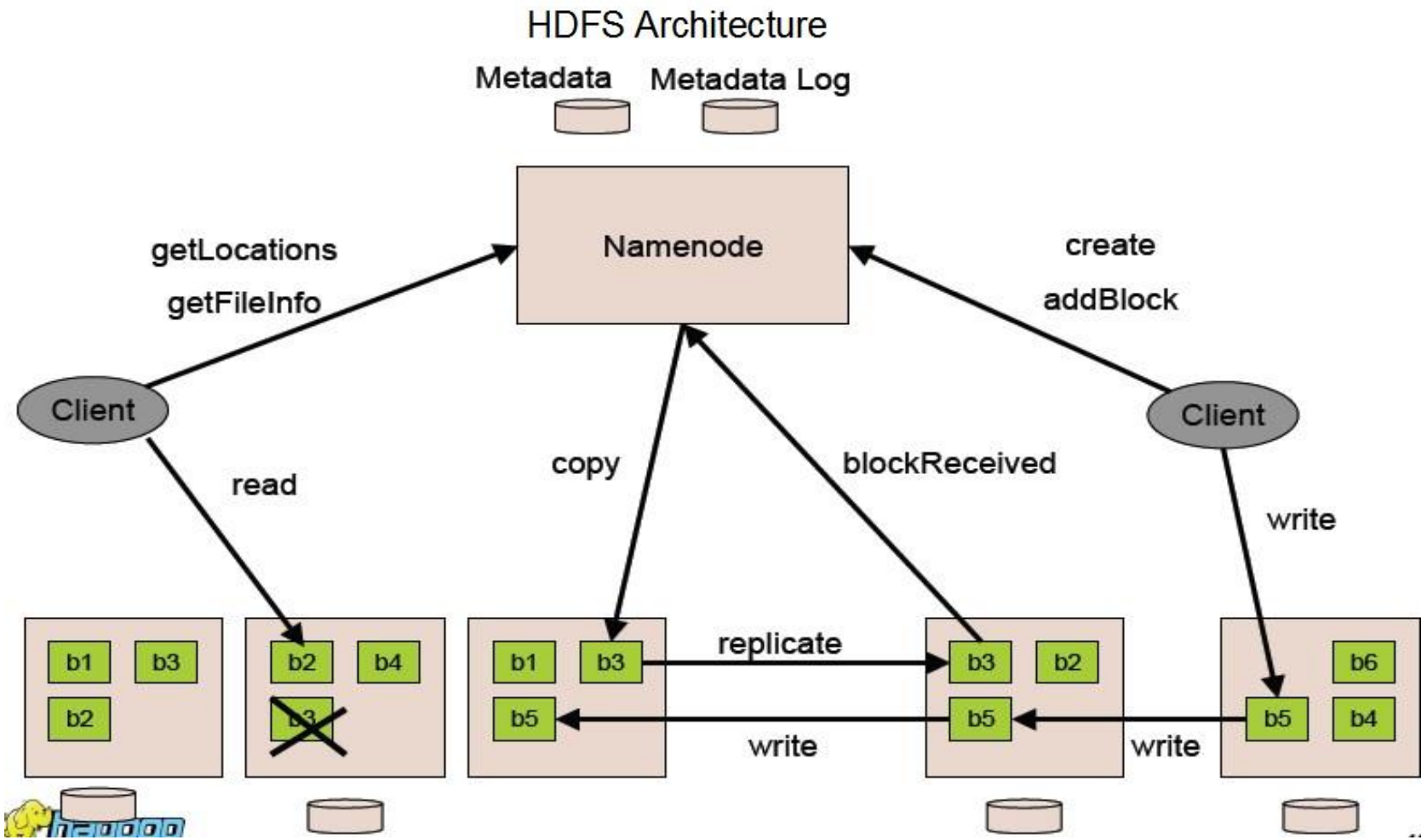
- Data Node sends Heartbeats
- Every 10th heartbeat is a Block report
- Name Node builds metadata from Block reports
- TCP – every 3 seconds

Data Correctness

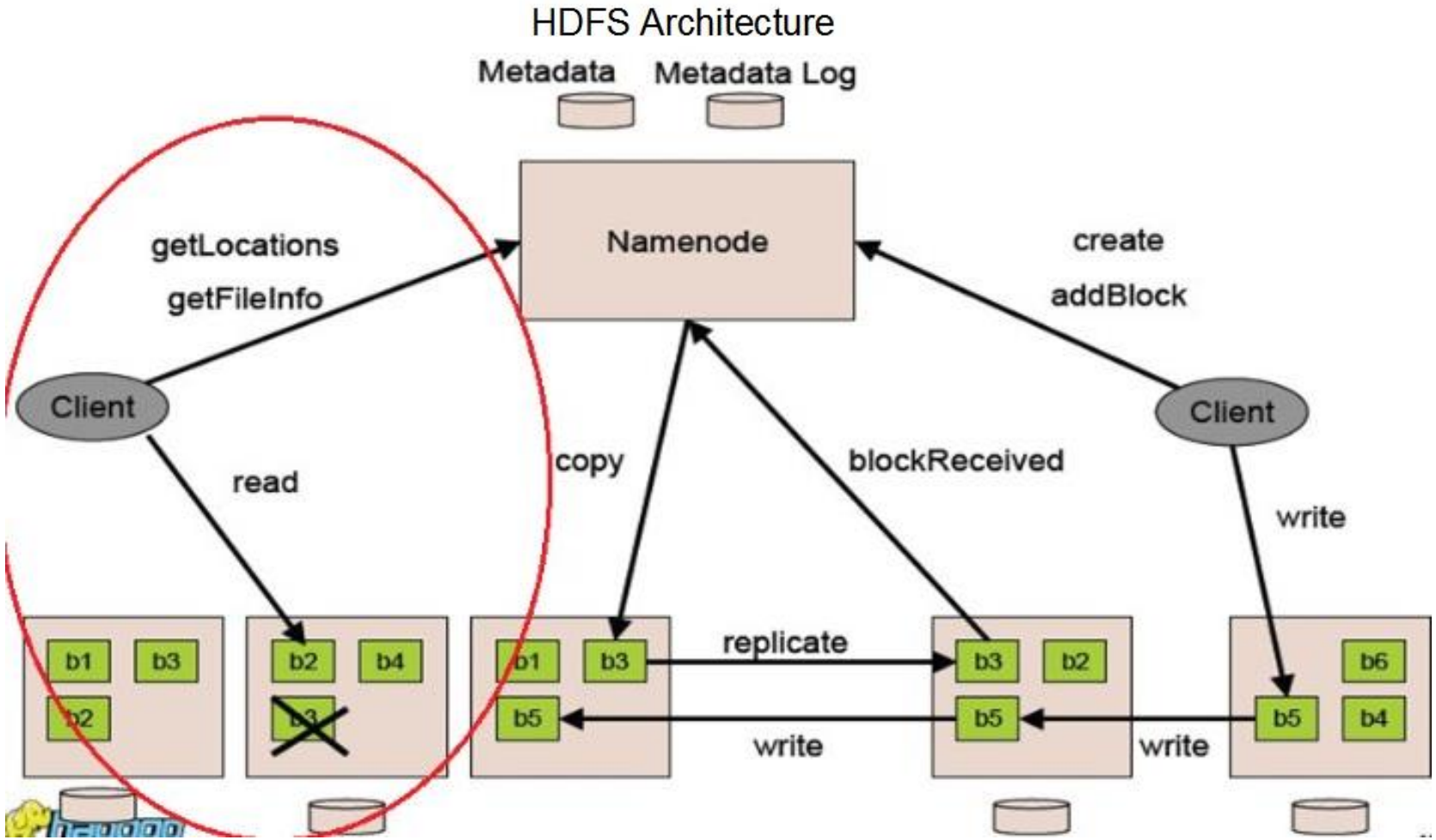
- Use Checksums to validate data – CRC32
- File Creation
  - Client computes checksum per 512 byte
  - DataNode stores the checksum
- File Access
  - Client retrieves the data and checksum from DataNode
  - If validation fails, client tries other replicas

HDFS Read, Write and Replicate

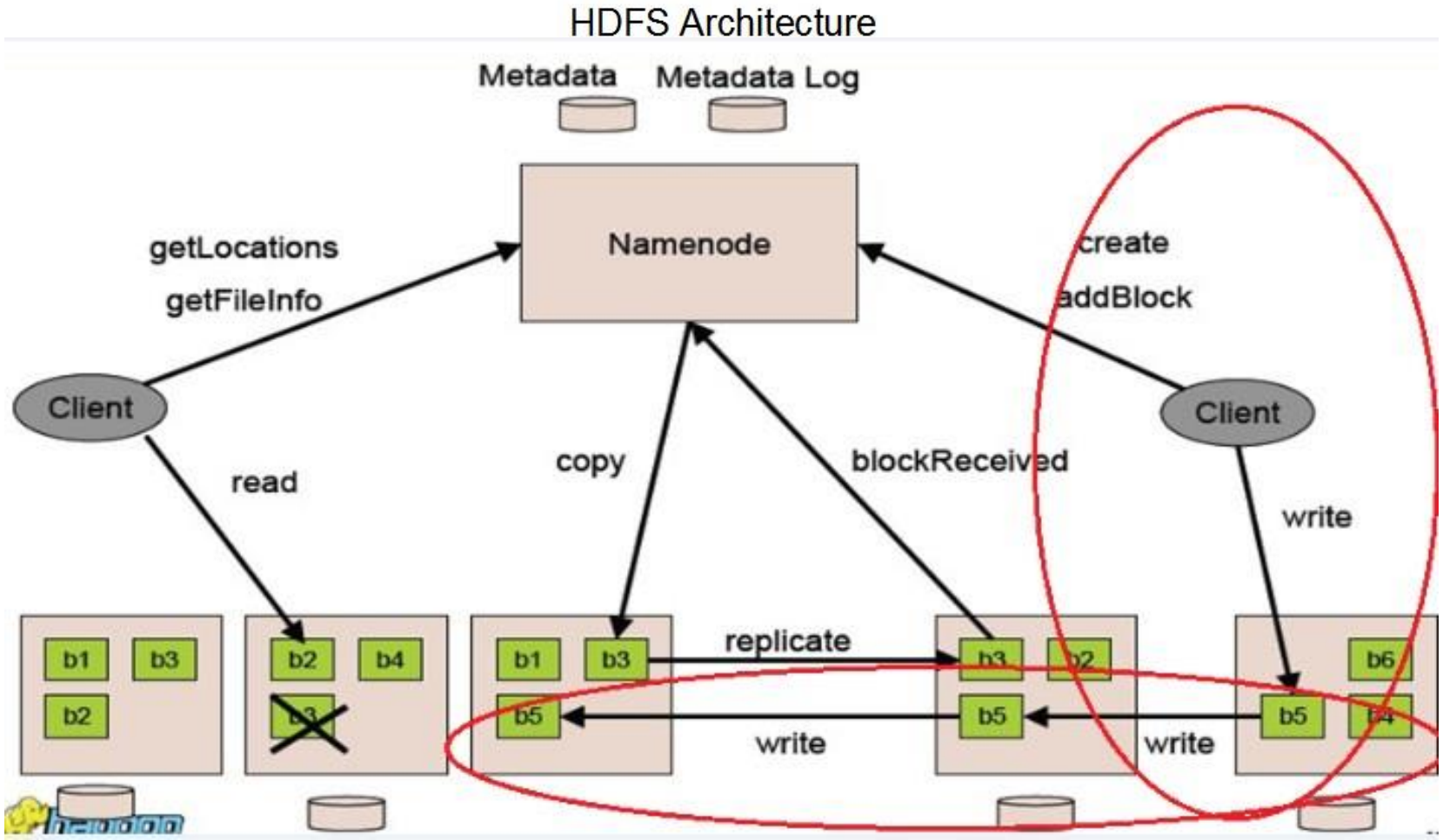
Read, Replicate and Write Operations



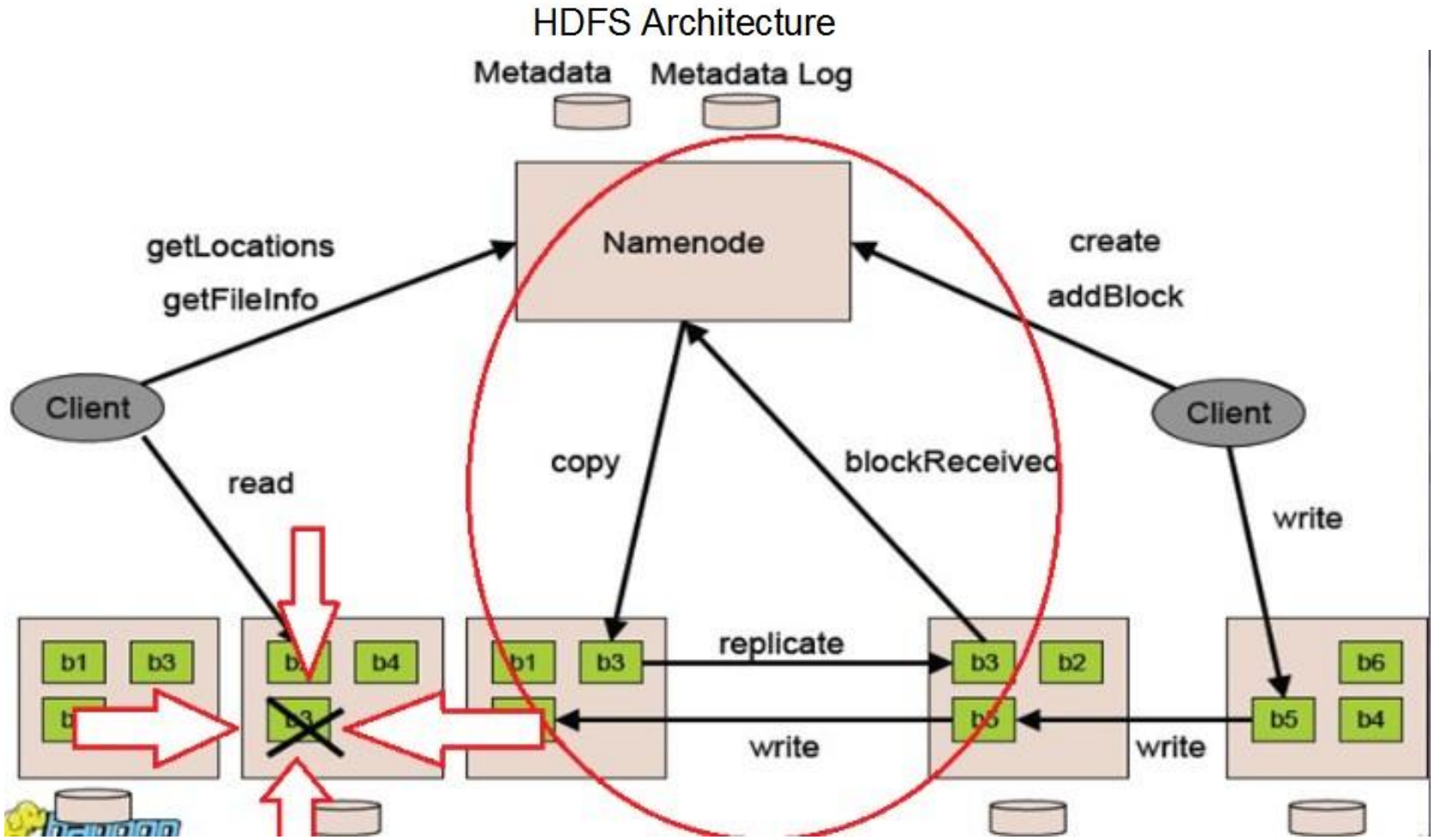
Read, Replicate and Write Operations



Read, Replicate and Write Operations

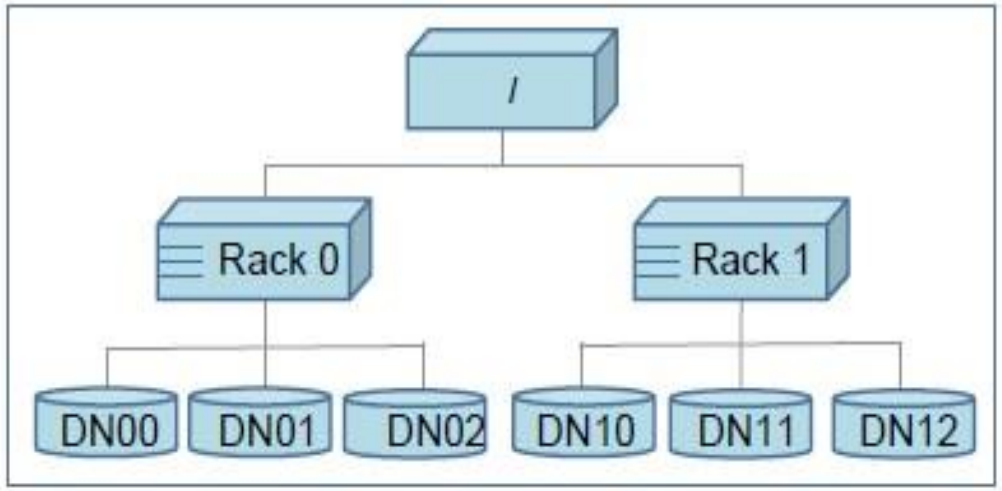


Read, Replicate and Write Operations

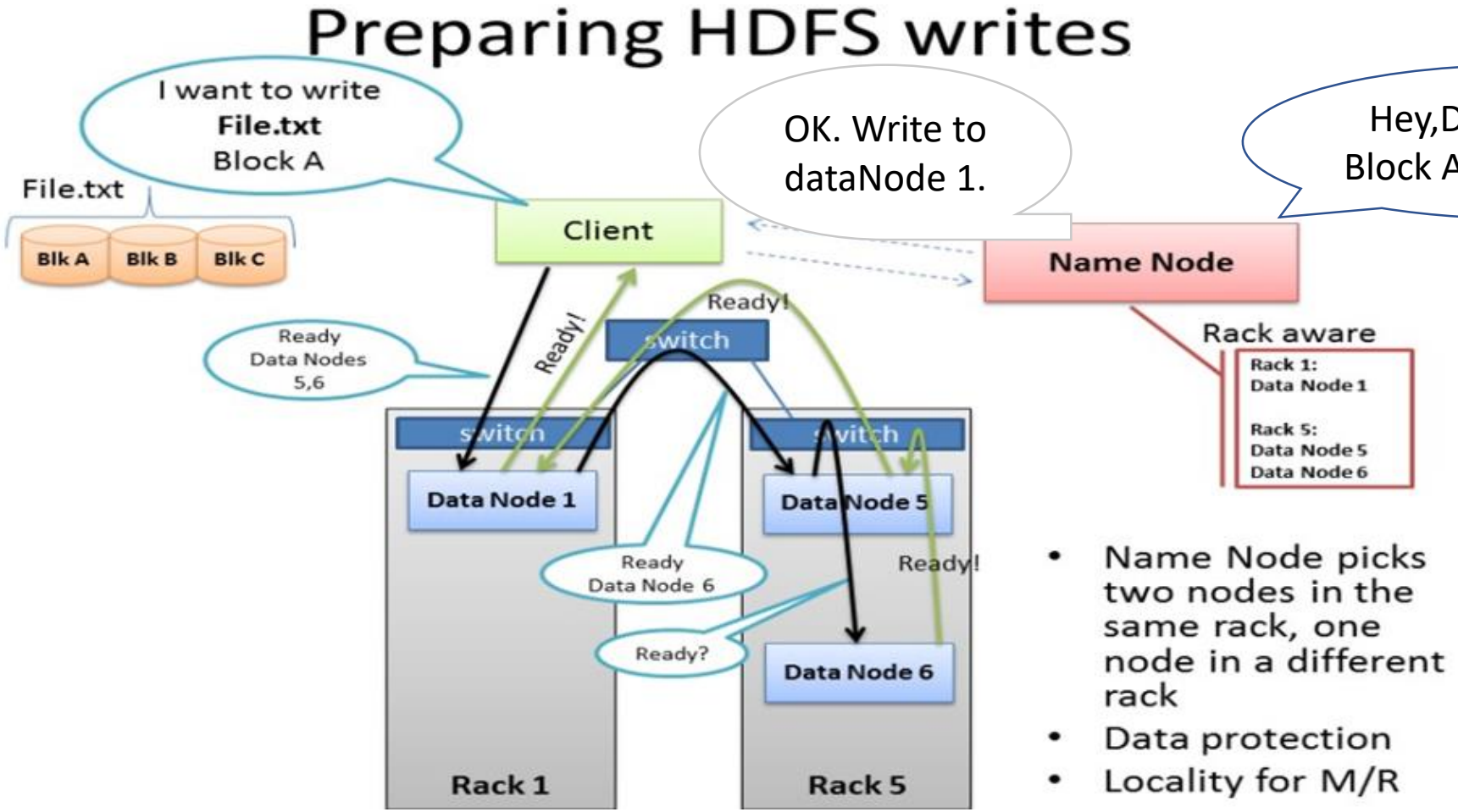


File I/O Operations and Replica Management

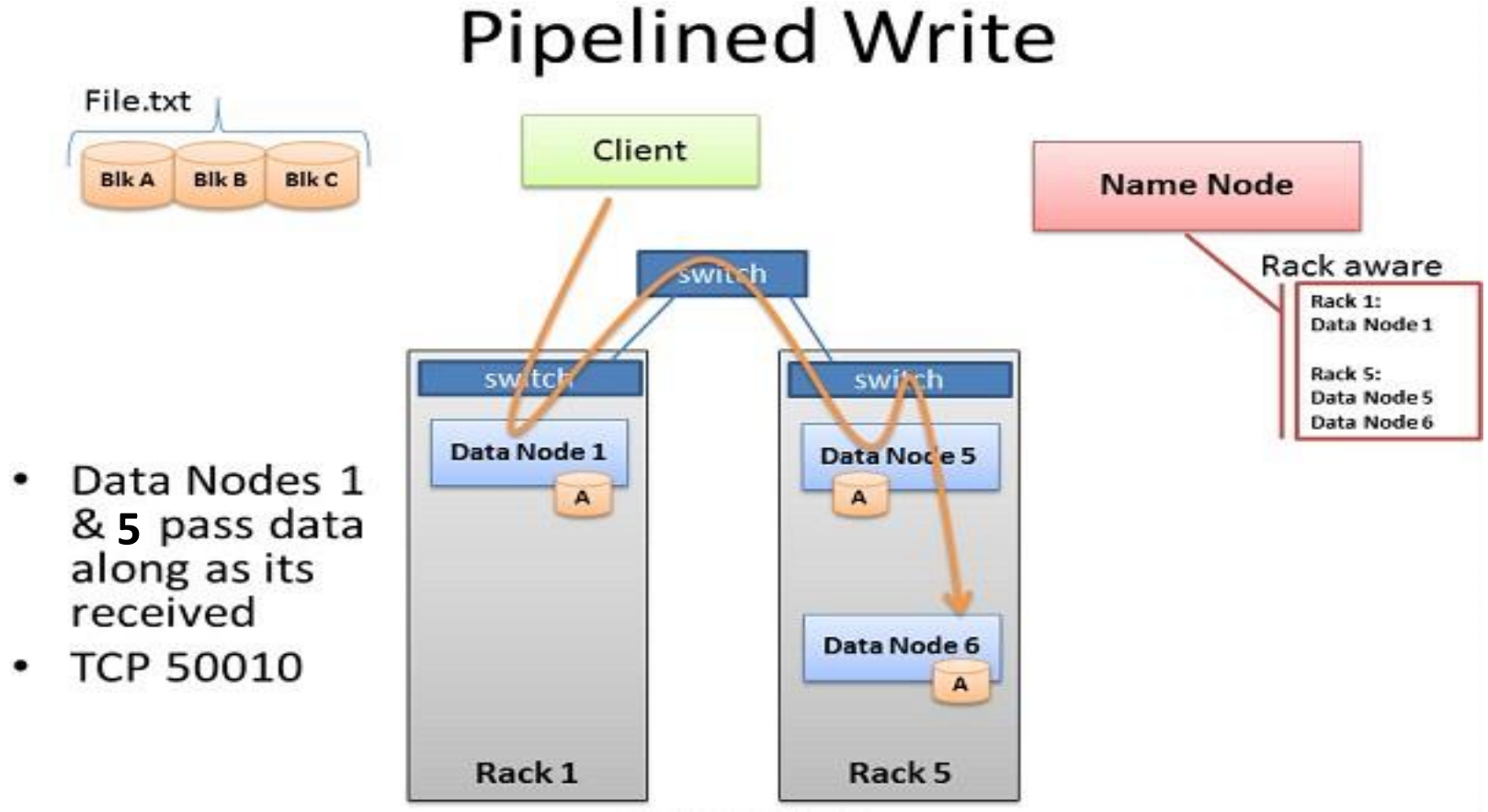
- Hadoop has the concept of “Rack Awareness”.
- The default HDFS replica placement policy can be summarized as follows:
  - No Datanode contains more than one replica of any block.
  - No rack contains more than two replicas of the same block, provided there are sufficient racks on the cluster.



File I/O Operations and Replica Management

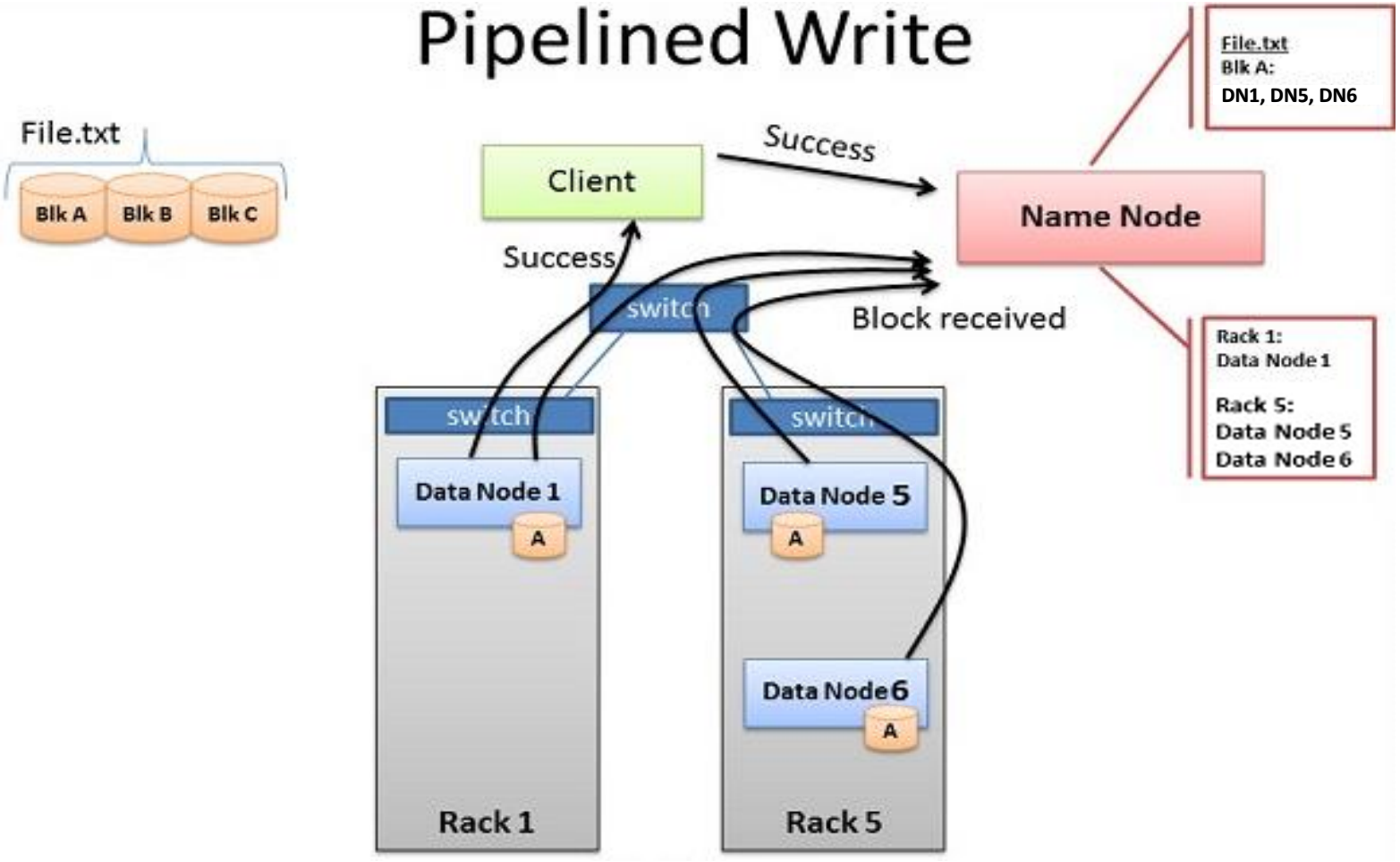


File I/O Operations and Replica Management



Bytes are pushed to the pipeline as a sequence of *packets*. The bytes that an application writes first buffer at the client side. After a packet buffer is filled (typically *64 KB*), the data are pushed to the pipeline.

File I/O Operations and Replica Management

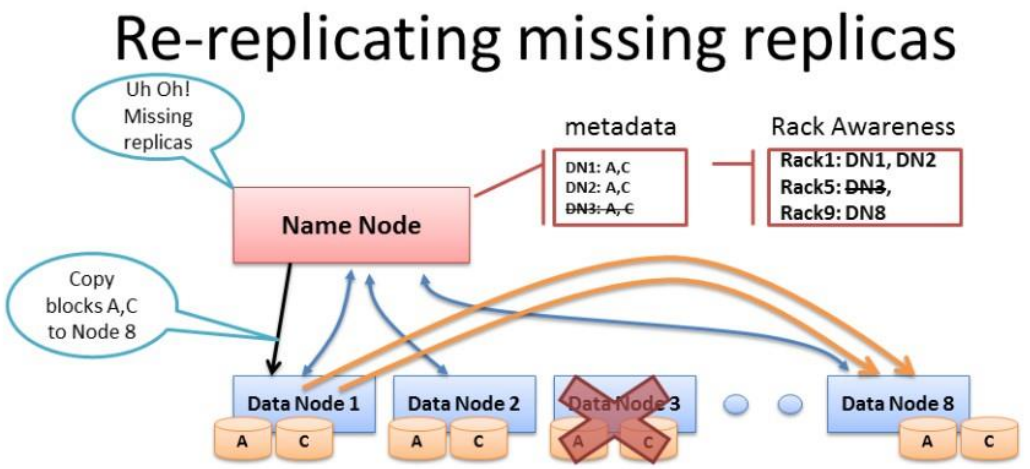


- Client retrieves a list of DataNodes on which to place replicas of a block
- Client writes block to the first DataNode
- The first DataNode forwards the data to the next DataNode in the Pipeline
- When all replicas are written, the client moves on to write the next block in file

Failure recovery in HDFS

Failure Recovery

- The NameNode does not directly call DataNodes. It uses replies to heartbeats to send instructions to the DataNodes. The instructions include commands to:
  - Replicate blocks to other nodes:
    - DataNode died.
    - copy data to local.
  - Remove local block replicas
  - Re-register or to shut down the node
- So when dataNode died, NameNode will notice and instruct other dataNode to replicate data to new dataNode. What if NameNode died?



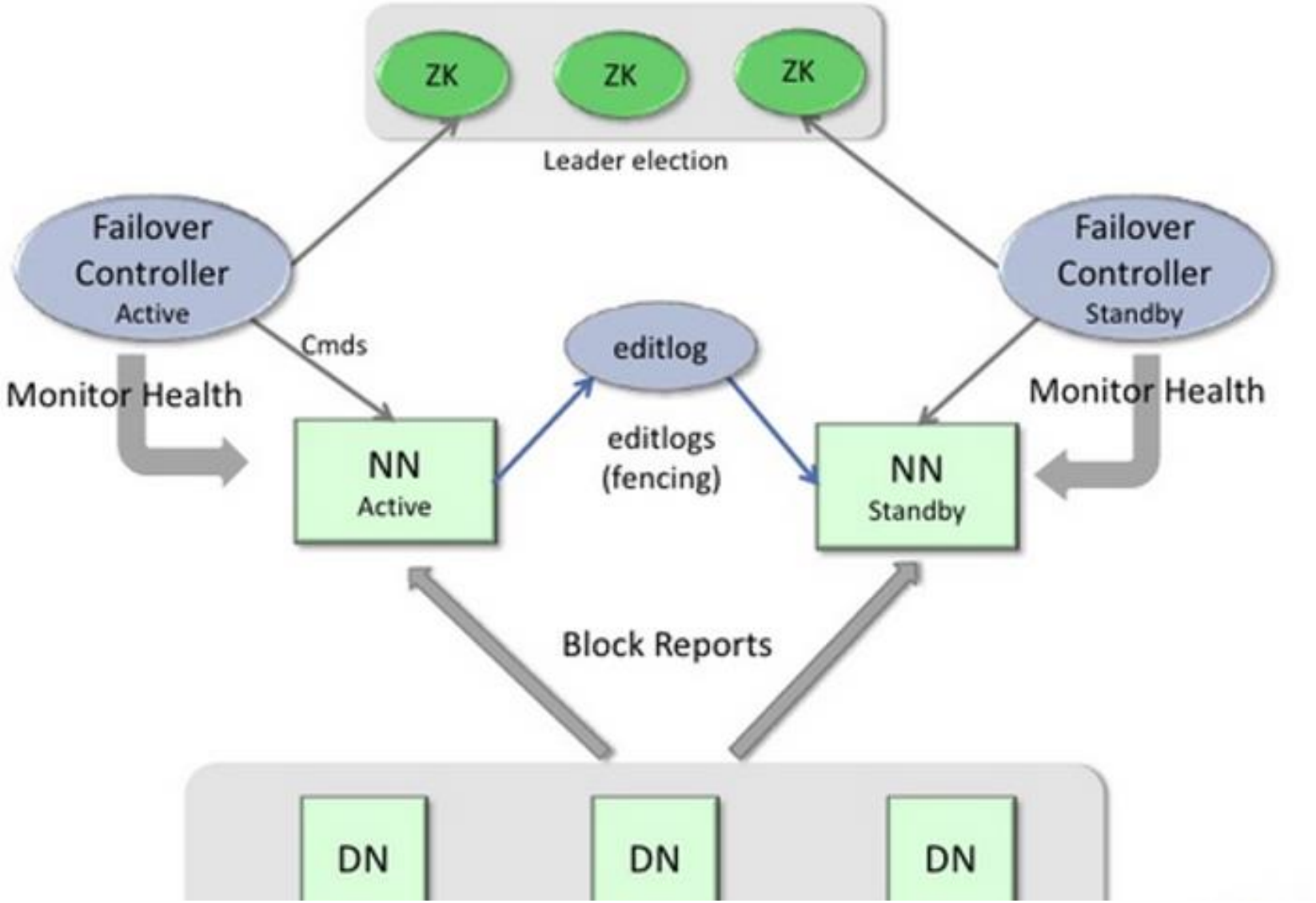
- Missing Heartbeats signify lost Nodes
- Name Node consults metadata, finds affected data
- Name Node consults Rack Awareness script
- Name Node tells a Data Node to re-replicate

Failure Recovery

- Keep journal (the modification log of metadata). Transaction log for file deletes/adds, etc. Does not use transactions for whole blocks or file-streams, only metadata.
- Checkpoint: The persistent record of the metadata stored in the local host's native file system.
- CheckpointNode and BackupNode--two other roles of NameNode
  - CheckpointNode: When journal becomes too long, checkpointNode combines the existing checkpoint and journal to create a new checkpoint and an empty journal.
- During restart, the NameNode initializes the namespace image from the checkpoint, and then replays changes from the journal until the image is up-to-date with the last state of the file system.
- BackupNode: A read-only NameNode
  - It maintains an in-memory, up-to-date image of the file system namespace that is always synchronized with the state of the NameNode.
  - If the NameNode fails, the BackupNode's image in memory and the checkpoint on disk is a record of the latest namespace state.

Avoiding Single NameNode bottleneck

HDFS 2.0: High Availability Elaborated



Reference: Hadoop Summit 2012 | HDFS High Availability talk

In order to provide a fast failover, it is also necessary that the Standby node have up-to-date information regarding the location of blocks in the cluster.

In order to achieve this, the DataNodes are configured with the location of both NameNodes, and send block location information and heartbeats to both.

The ZKFailoverController (ZKFC) is a ZooKeeper client that also monitors and manages the state of the NameNode.

Each of the hosts that run a NameNode also run a ZKFC.

The ZKFC is responsible for **Health monitoring** of Namenode

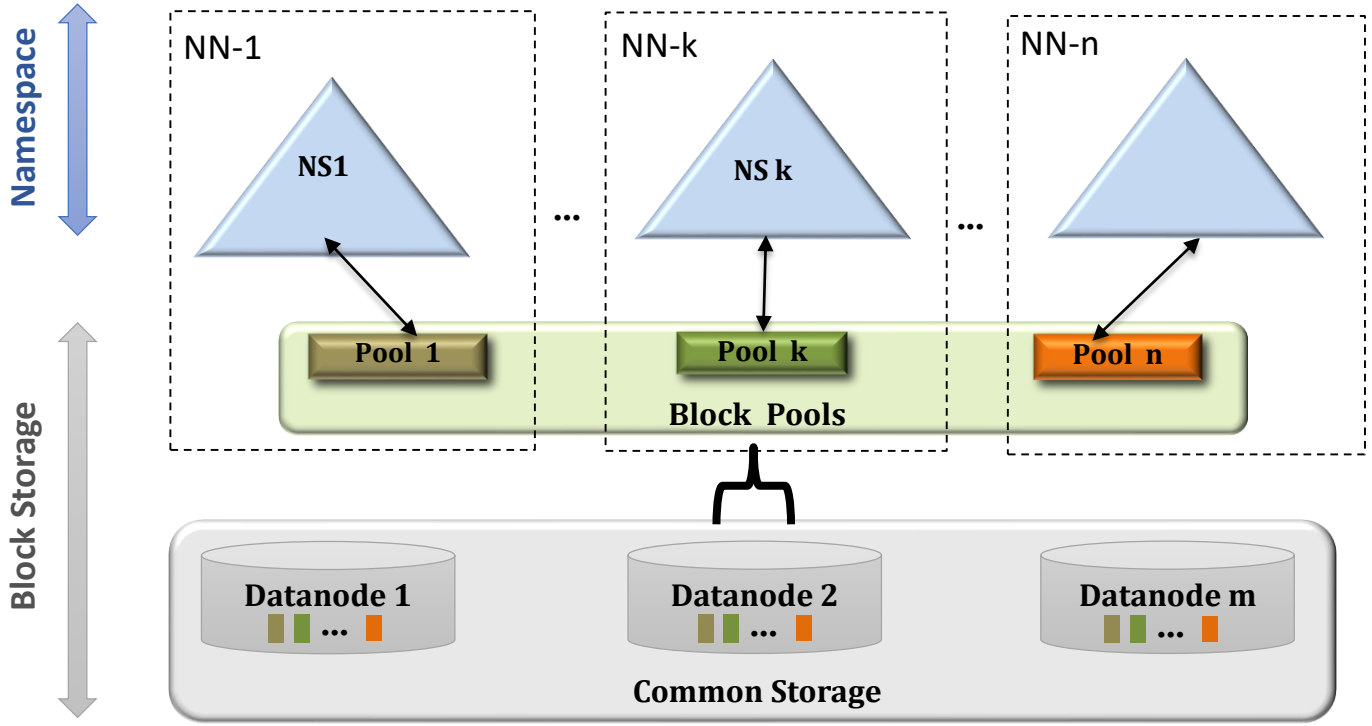
- the ZKFC contacts its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds promptly with a healthy status, the ZKFC considers the NameNode healthy.
- If the NameNode has crashed, frozen, or otherwise entered an unhealthy state, the health monitor marks it as unhealthy.

HDFS federation

Issues with one namenode

- **Tight coupling of Block Storage and Namespace:** This makes alternate implementations of namenodes challenging and limits other services from using the block storage ***directly***.
- **Namespace scalability:** While HDFS cluster storage scales horizontally with the addition of datanodes, the namespace does not. Currently the namespace can only be vertically scaled on a single namenode. The namenode stores the entire file system metadata in memory. This limits the number of blocks, files, and directories supported on the file system to what can be accommodated in the memory of a single namenode.
- **Performance:** File system operations are limited to the throughput of a single namenode, which currently supports 60K tasks.
- **Isolation:** For many deployments, the cluster is used in a multi-tenant environment where many organizations share the cluster. A single namenode offers no isolation in this setup. A separate namespace for a tenant is not possible. An experimental application that overloads the namenode can slow down the other production applications. A single namenode also does not allow segregating different categories of applications (such as HBase) to separate namenodes.

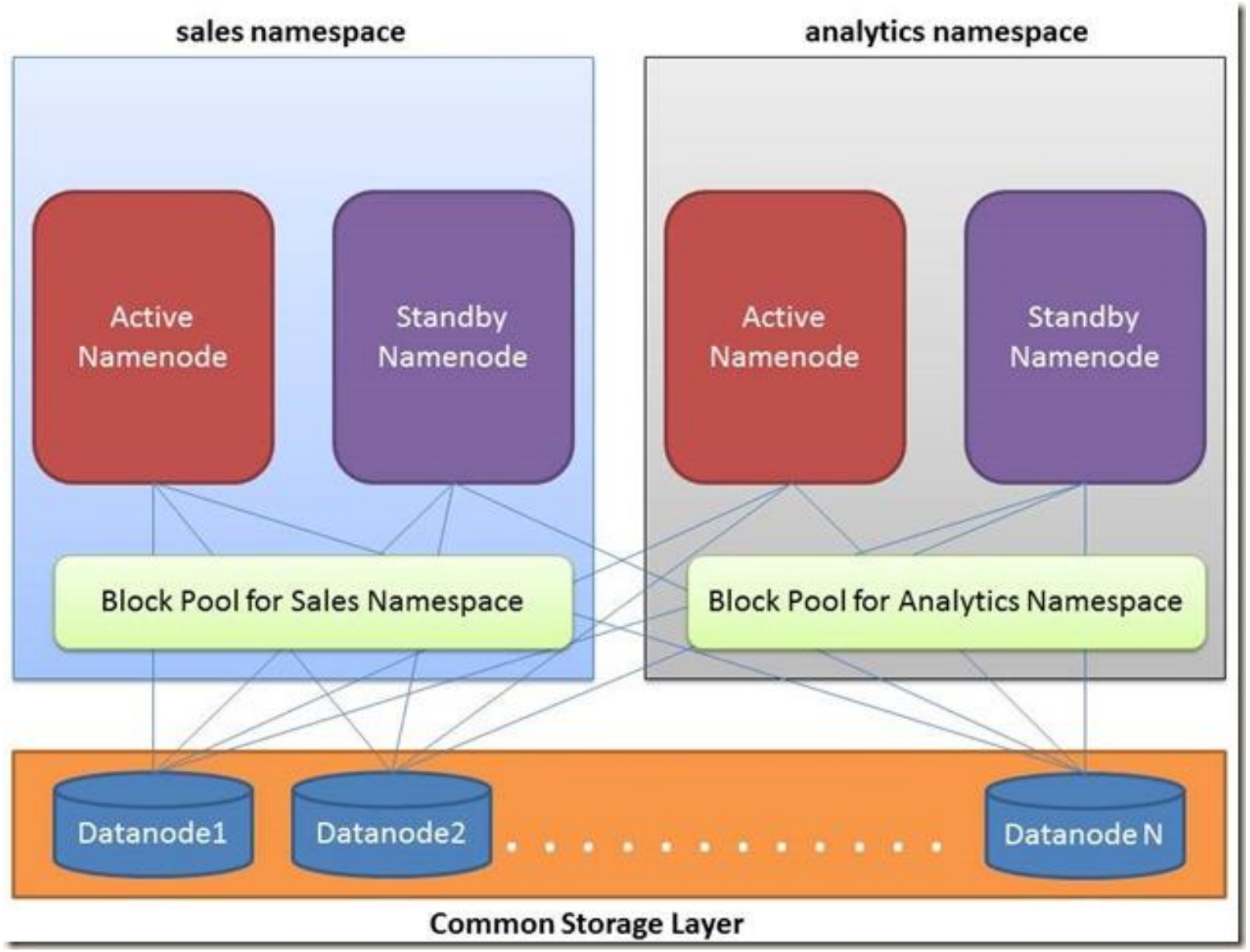
HDFS 2.0: Name Node Federation Elaborated



- Multiple **independent** Namenodes and Namespace Volumes in a cluster
  - Namespace Volume = Namespace + Block Pool
- Block Storage as generic storage service
  - Set of blocks for a Namespace Volume is called a ***Block Pool***
  - DNs store blocks for all the Namespace Volumes – no partitioning

- **Namespace** manages directories, files and blocks. It supports file system operations such as creation, modification, deletion and listing of files and directories.
- **Block Storage** has two parts:
  - **Block Management** maintains the membership of datanodes in the cluster. It supports block-related operations such as creation, deletion, modification and getting location of the blocks. It also takes care of replica placement and replication.
  - **Physical Storage** stores the blocks and provides read/write access to it.

HDFS 2.0: High Availability, Federated



http://pe-kay.blogspot.in/2016/02/configuring-federated-hdfs-cluster-with.html

Storage (HDFS 2.0) Summary

- Specialized master (Name Node), Commodity workers (Data Node),
- Uniform block storage, format-free
- Immutable writes, Streaming reads
- Focus on throughput, more than Latency
- Horizontal Scalability (Federated Name Nodes)
- Self-Managing - Replication, re-replication, rebalancing, garbage collection
- Optimizing – block placement strategy, data pipelining, compression
- Avoids single points of failure – Name node never touches data; secondary name node for backup, high availability (active 0 standby) name node
- HDFS Client, WebHDFS
- Variants & Alternatives – MapR-FS, Amazon S3

Thanks