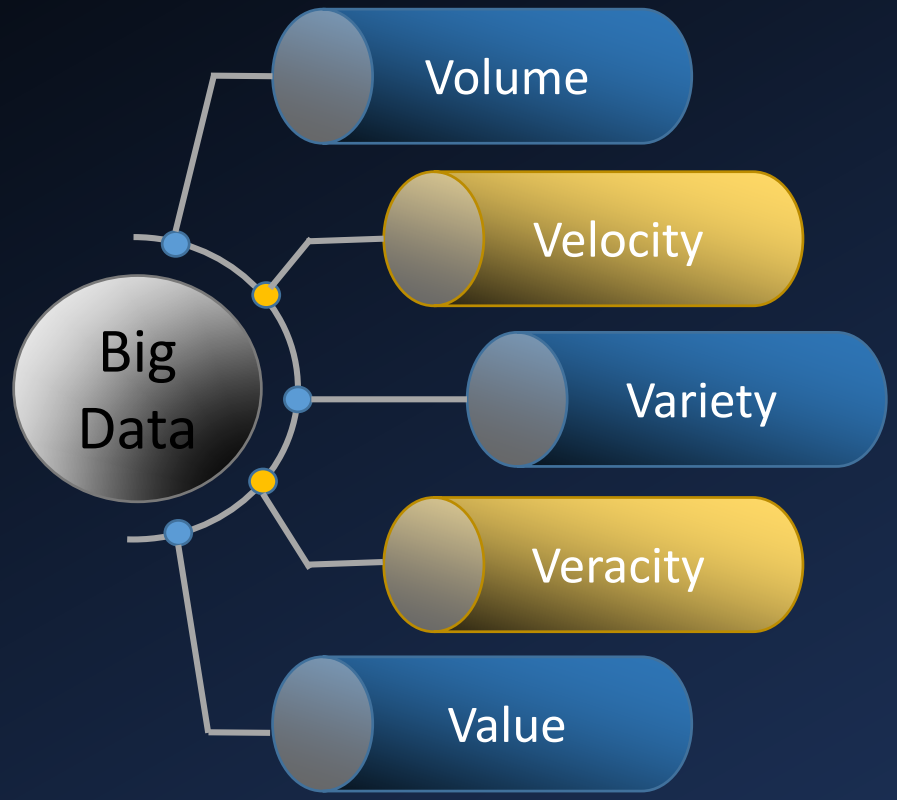
Big Data (MHI222956/MHI225101)

2.2 Distributed File Systems

Big Data Solutions



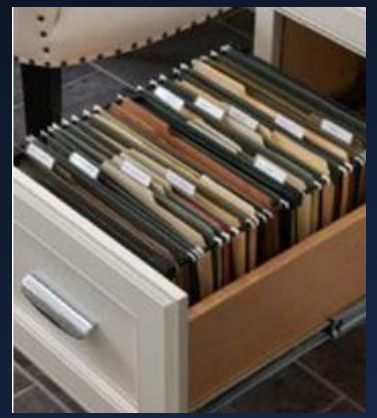
- Scalability in Big Data solutions is typically achieved by distributing storage and/or processing over many low powered computers (scaling out) rather than increasing the power of a single computer (scaling up)

Big Data Challenges

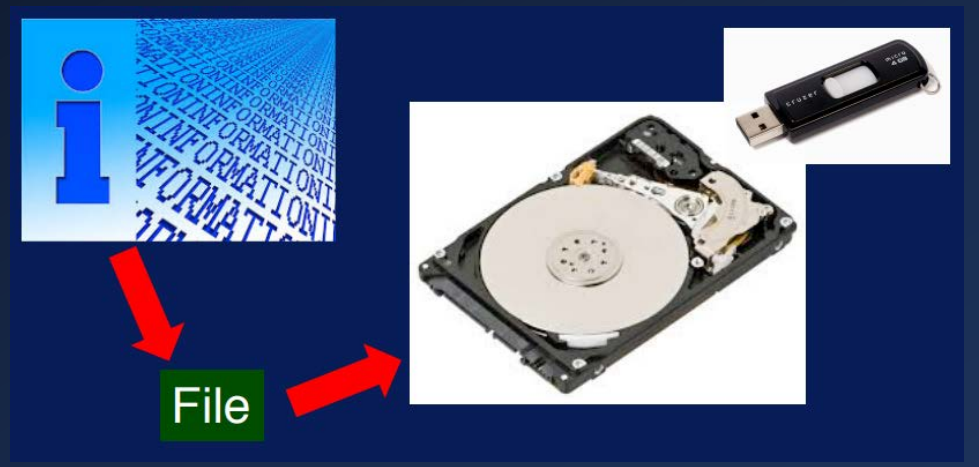
Distributed data store

- A distributed data store is a computer network where information is stored on more than one node.
- Usually specifically refers to a distributed database, but can also be a distributed file system or a peer-to-peer network
- In a Big Data platform, systems tend to operate over many servers operating in a cluster, managing tens or hundreds of terabytes of data across billions of records

File Systems



File cabinets



File systems

- Store information in the long-term
- Multiple processes can access the same information if needed
- File system - how the operating system manages files

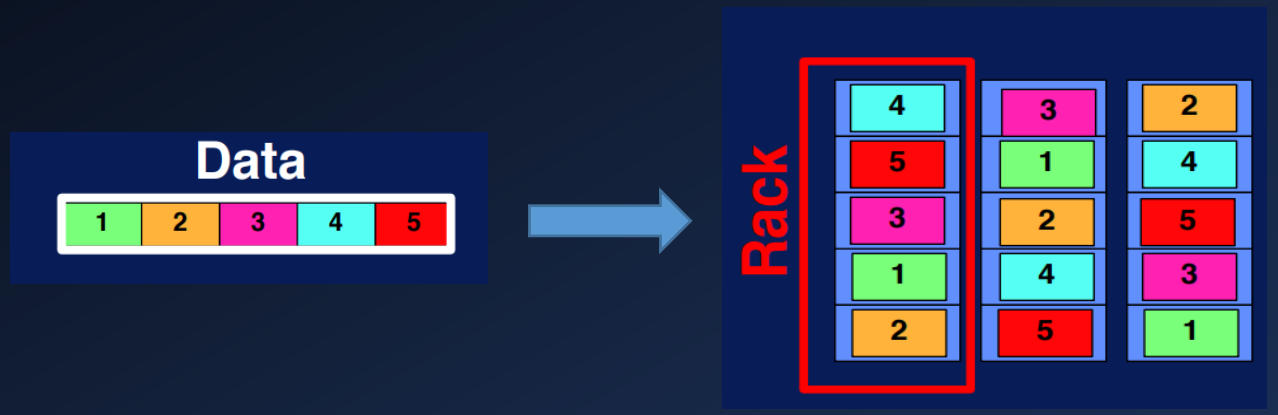
File Systems

- What if you have more data:
  - Buy a bigger disk?
  - Copy data to an external hard drive?
  - etc.

File Systems

- Then, how about having thousands of computers to store your data with big volumes and variety?
  - You need to know where to find the files you need, depending on what you're doing.
  - Distributed file systems can handle the data access and do this for you.

Distributed File Systems

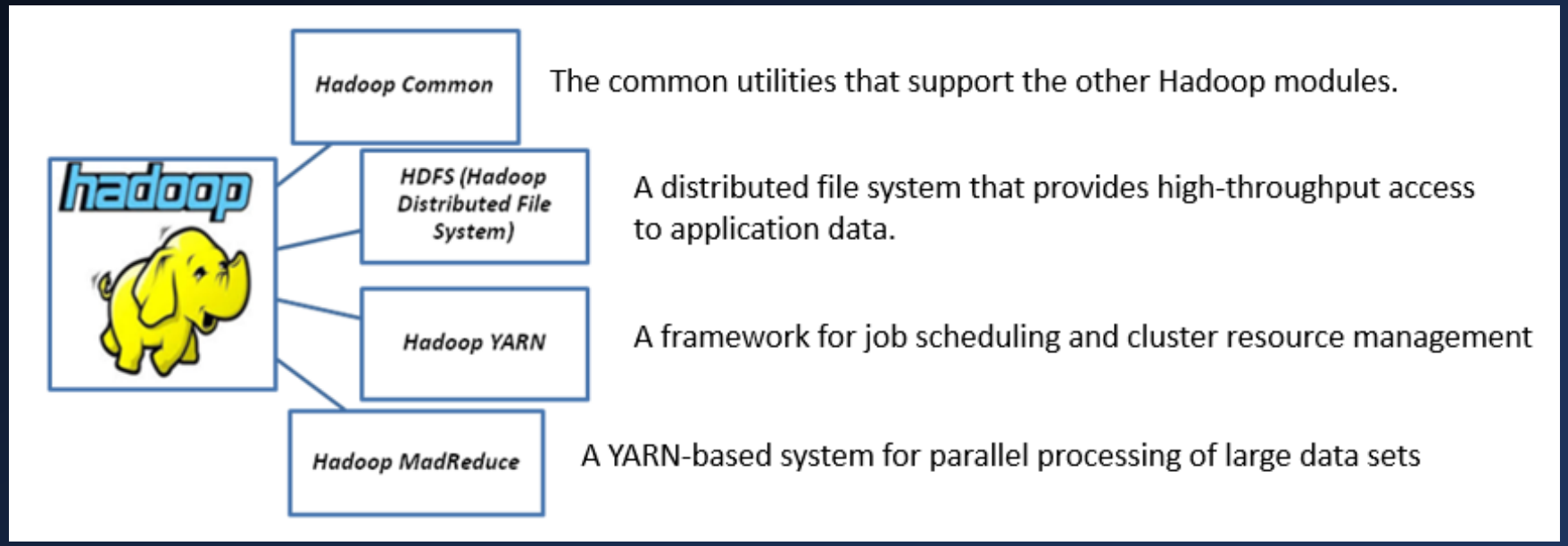


- A DISTRIBUTED FILE SYSTEM a physically distributed implementation of the traditional file system, allowing users to manipulate, organize, and share data seamlessly, regardless of its actual location on the network.
- Data sets, or parts of a data set, can be replicated across the nodes of a distributed file system
  - Analysis of parts of the data is needed in a data parallel fashion, computation can be moved to these nodes
  - Data replication makes the system more fault tolerant.
  - Data replication also helps with scaling the access to this data by many users

Distributed File Systems

- Distributed file systems provide data scalability, fault tolerance, and high concurrency through partitioning and replication of data on many nodes.
- However, it is hard to make changes to data over time
- In most big data systems, the data is written once and the updates to data is maintained as additional data sets over time.

Apache Hadoop



- Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
- Designed to scale up from single servers to thousands of machines, each offering local computation and storage.

HDFS (Hadoop Distributed File System)

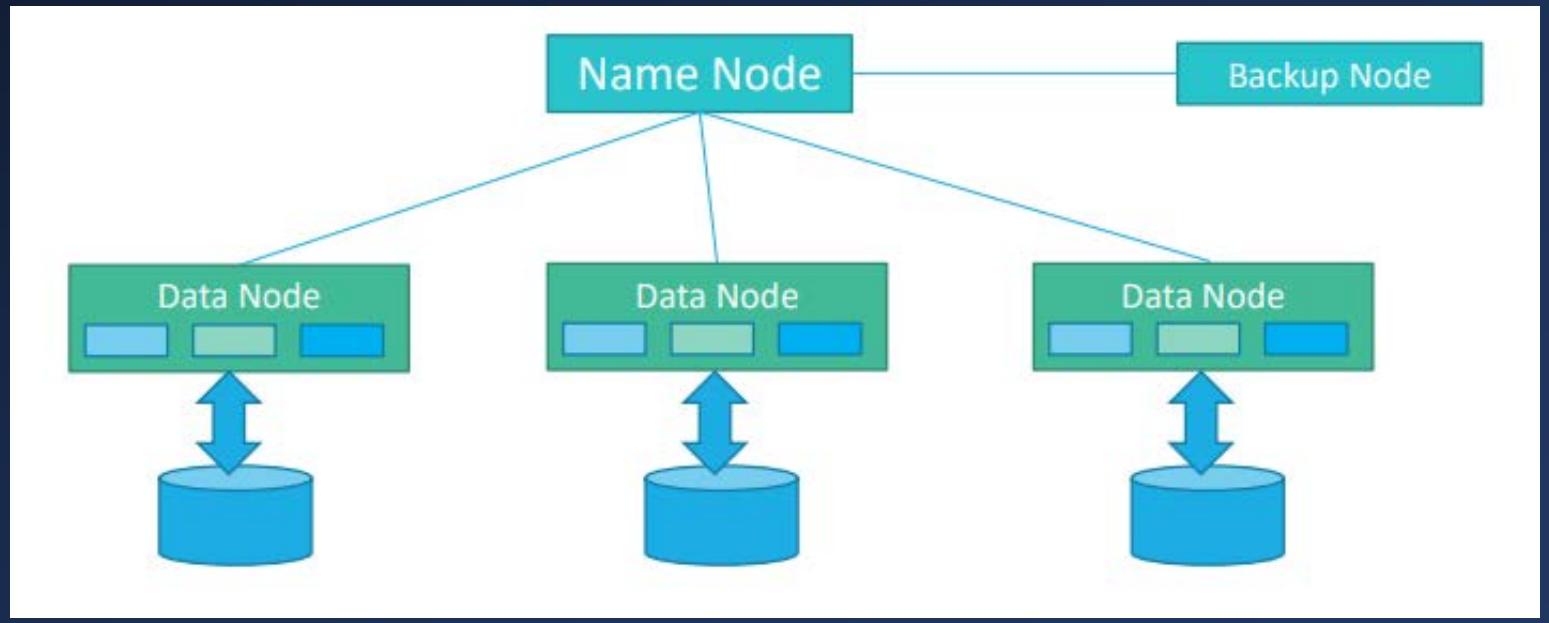
- The primary distributed storage used by Hadoop applications
- Structured similarly to a regular Unix file system except that data storage is distributed across several machines (using commodity hardware) in an HDFS cluster
- It is not intended as a replacement to a regular file system, but rather as a file system-like layer for large distributed systems
- Data can be accessed using either the *Java API* (Hadoop is written in Java), or the *Hadoop command line client*. Many operations are similar to their Unix counterparts

HDFS (Hadoop Distributed File System)

- Files cannot be modified once written, and the latency of reads/writes is poor
- However, throughput scales almost linearly with the number of datanodes in a cluster, so it can handle workloads no single machine would ever be able to
- HDFS has unique features that make it ideal for distributed systems:
  - Failure tolerant
  - Scalable - read/write capacity scales fairly well with the number of datanodes
  - Industry standard - Lots of other distributed applications build on top of HDFS (HBase, Map-Reduce, Spark)
  - Pairs well with MapReduce

HDFS cluster

- HDFS follows the master-slave architecture
  - Single NameNode
  - A number of DataNodes



HDFS cluster

- HDFS exposes a file system namespace and allows user data to be stored in files.
- Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes

- NameNode
  - Master server that manages the file system namespace and regulates client's access to files.
  - Executes file system namespace operations like opening, closing, and renaming files and directories and determines the mapping of blocks to DataNodes
- DataNodes
  - Usually for every node in a cluster, there is a Datanode, which manage storage attached to the nodes that they run on
  - Serve read and write requests from the file system's clients and perform block creation, deletion, and replication upon instruction from the NameNode

HDFS Operations

- Starting HDFS

  Initially, format the configured HDFS file system, open NameNode (HDFS server), and execute the following command.

  ```
  $ hadoop namenode -format
  ```

  Then, using the following command to start the NameNode as well as the data nodes as cluster.

  ```
  $ start-dfs.sh
  ```

- List Files in HDFS

  After loading the information in the server, we can find the list of files in a directory, status of a file, using ‘ls’.

  ```
  $ hadoop fs -ls <args>
  ```

HDFS Operations

- Inserting Data into HDFS
  - Create an input directory, e.g. /user/input

    $ hadoop fs -mkdir /user/input

  - Transfer and store a data file from local systems (e.g., /home/file.txt) to the Hadoop file system using the put command.

    $ hadoop fs -put /home/file.txt /user/input

  - You can verify the file using ls command.

    $ hadoop fs -ls /user/input

HDFS Operations

- Retrieving Data from HDFS
  - View the data from HDFS using cat command.

    $ hadoop fs -cat /user/output/outfile
  - Get the file from HDFS to the local file system using get command.

    $ hadoop fs -get /user/output/ /home/hadoop_tp/
- Shutting Down the HDFS

  $ stop-dfs.sh

Hadoop modes

- By default Hadoop works in a standalone non-distributed mode, as a single process which reads from and writes to the local filesystem
  - Useful for developing/debugging
- Hadoop can also be run on a single-node in a pseudo-distributed mode using multiple Java processes and working with HDFS
  - Again useful for developing/debugging
  - Uses some of the Hadoop configuration files that are used in fully distributed mode
  - Requires SSH to be installed on node
- In fully distributed mode Hadoop clusters can be configured ranging from a few nodes to extremely large clusters with thousands of nodes.

Other distributed file system

- Amazon S3
- Google Cloud Storage

File storage on HDFS

- Theoretically, the format of the files you can store on HDFS is entirely up to you.
- However, unlike a regular file system, HDFS is best used in conjunction with a data processing toolchain like MapReduce or Spark.
- Need to choose file type and format carefully when using HDFS or similar fs to take advantage of the distributed nature of the file system and analytic processing that runs on top of it.

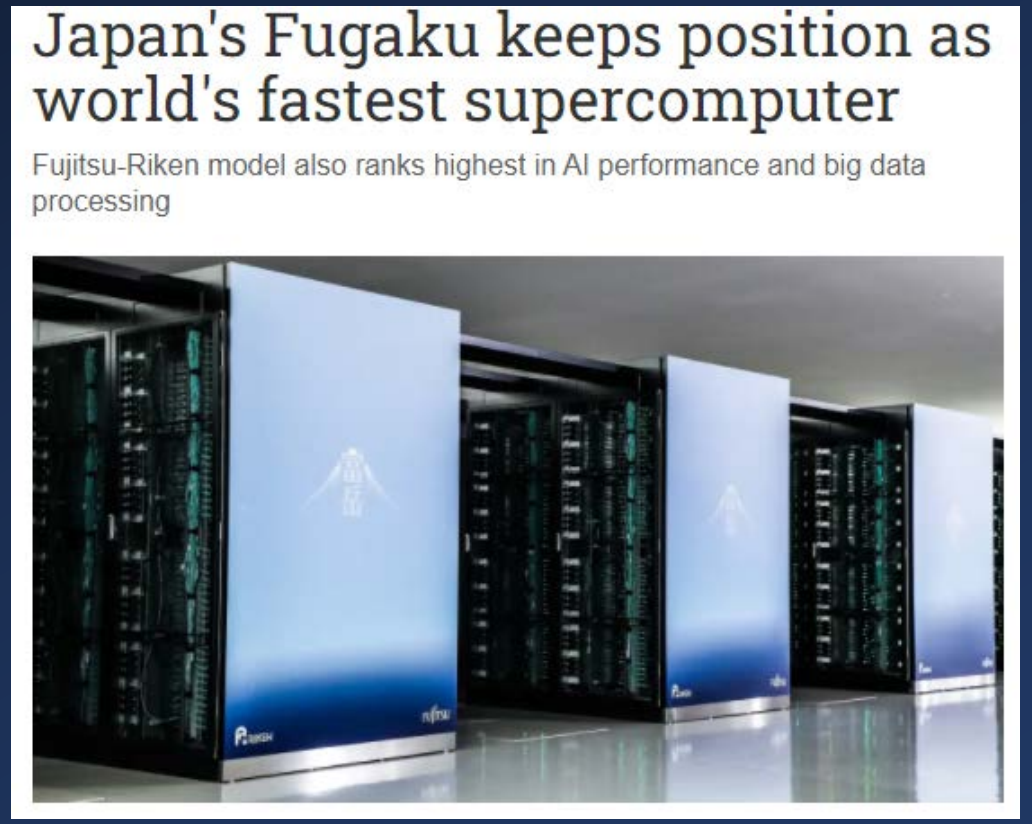
File storage on HDFS

- Splittability
  - Better to use a file format that is splittable
  - HDFS stores large files in blocks (and files in Big Data usually are large), want to be able to start reading at any point in the file to support parallel processing
- Compression
  - Compressed files are smaller so better for I/O throughput and storage, but have processing overhead
  - Better to use a file format that supports block compression, otherwise decompressor will have to start at the beginning of a compressed file to access any block
- Schema evolution
  - May want to be able to add a field and still read historical data
  - Better to used a file format that includes metadata with the data

	Splittability	Compression	Schema evolution
XML file	Not splittable		
JSON file	Not splittable		
CSV file	To support splittability, do not include header or footer lines	do not support block compression	No metadata
JSON records	Splittable, as each line is its own JSON object	do not support block compression	fully enabling schema evolution
Sequence files (Hadoop-specific format)	Blocks are marked with sync markers for splittability	Can be compressed at different levels - record or block	Do not store metadata
Apache Avro	Splittable	support block compression	Very good schema evolution support

Scalable Computing

- Computing on a single compute node
- Parallel computing
  - Computation need more than one node or parallel processing
- Supercomputer



Scalable Computing

- Commodity clusters
  - affordable parallel computers with an average number of computing nodes
  - not as powerful as traditional parallel computers and are often built out of less specialized nodes
  - These type of systems have a higher potential for partial failures.

- It is this type of distributed computing that pushed for a change towards cost effective reliable and Fault-tolerant systems for management and analysis of big data.

Summary

- Distributed File Systems
- HDFS
- File storage
- Scalable Computing