Introducing Apache Hadoop

What is Apache Hadoop?

- Apache Hadoop is a software framework that enables distributed processing on large clusters with thousands of nodes and petabytes of data.
- Apache Hadoop clusters can be built using commodity hardware where failure rates are generally high.
- Users will be able to develop parallel applications quickly, focusing on business logic rather than doing the heavy lifting of distributing data, distributing code for parallel processing, and handling failures.

Apache Hadoop mainly projects

- Apache Hadoop has mainly four projects:
  - Hadoop Common,
  - Hadoop Distributed File System (HDFS),
    - A framework for reliable distributed data storage
    - used to store data
  - Yet Another Resource Negotiator (YARN),
    - A framework for cluster resource management
    - used to manage the resources (CPU and memory) of the cluster and common utilities that support Hadoop
  - MapReduce.
    - used to process data

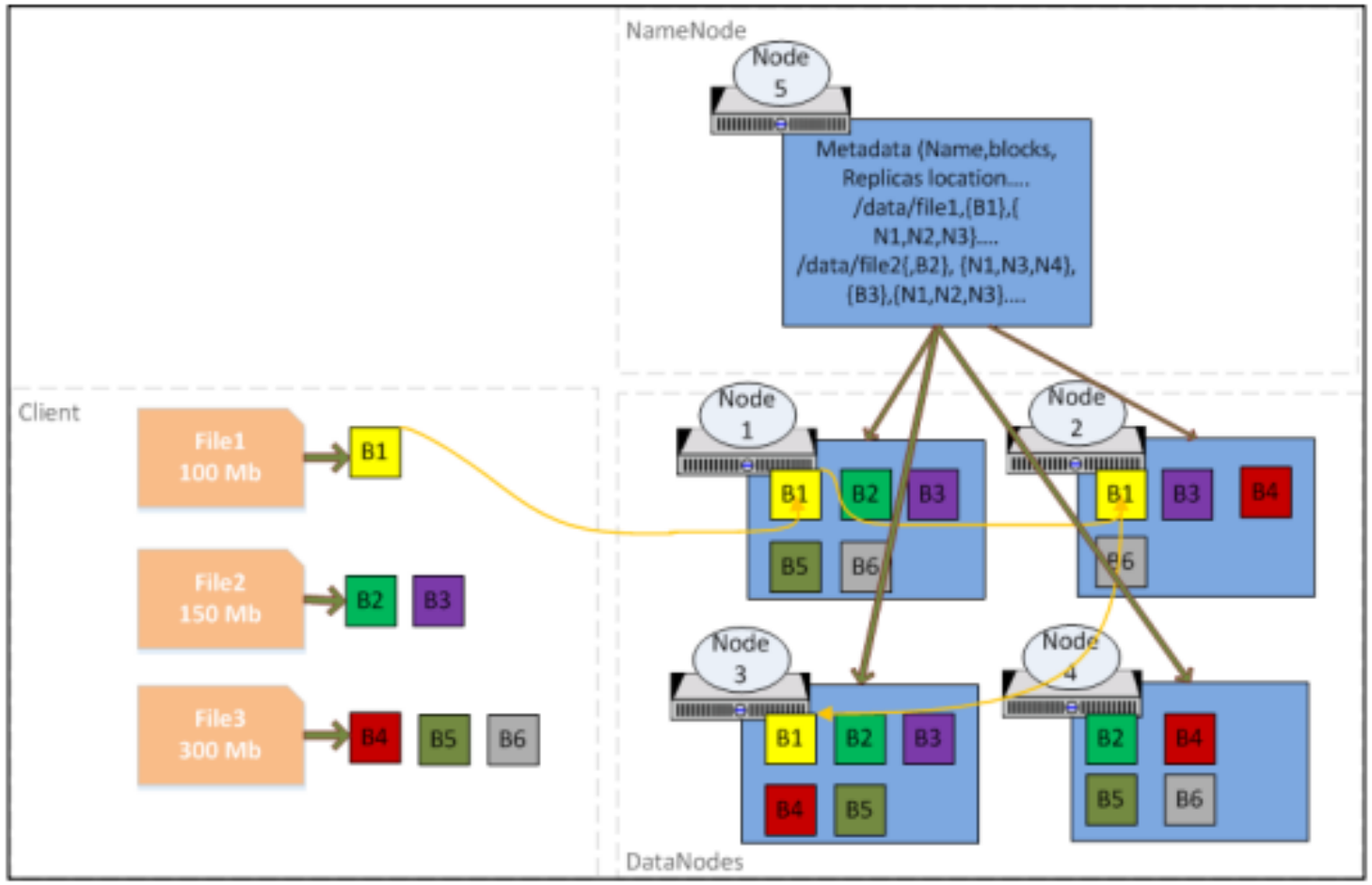
Three Components in Hadoop processing

- A framework for reliable distributed data storage: **HDFS**
- Multiple frameworks for parallel processing of data: **MapReduce, Crunch, Cascading, Hive, Tez, Impala, Pig, Mahout, Spark,** and **Giraph**
- A framework for cluster resource management: **YARN** and **Slider**

Hadoop Distributed File System (HDFS)

- HDFS is becoming a standard enterprise Big Data storage system because of the unlimited scalability and yet provides most features needed for enterprise-grade Big Data applications
- HDFS is a distributed filesystem that provides high scalability and reliability on large clusters of commodity hardware.
- HDFS files are divided into large blocks that are typically 128 MB in size
- then distributed across the cluster.
- Each block is replicated (typically three times) to handle hardware failures
- block placement exposed by NameNode so that computation can be moved to data with the MapReduce framework

HDFS Architecture



HDFS cont.

- HDFS has a bigger block size to reduce the number of disk seeks needed to read the complete file.
- blocks is stored on DataNodes and metadata is stored in NameNode
- If we lose the NameNode for any reason, blocks stored on DataNodes become useless as there is no way to identify the blocks belonging to the file names
- So, creating NameNode high availability and metadata backups is very important in any Hadoop cluster

MapReduce

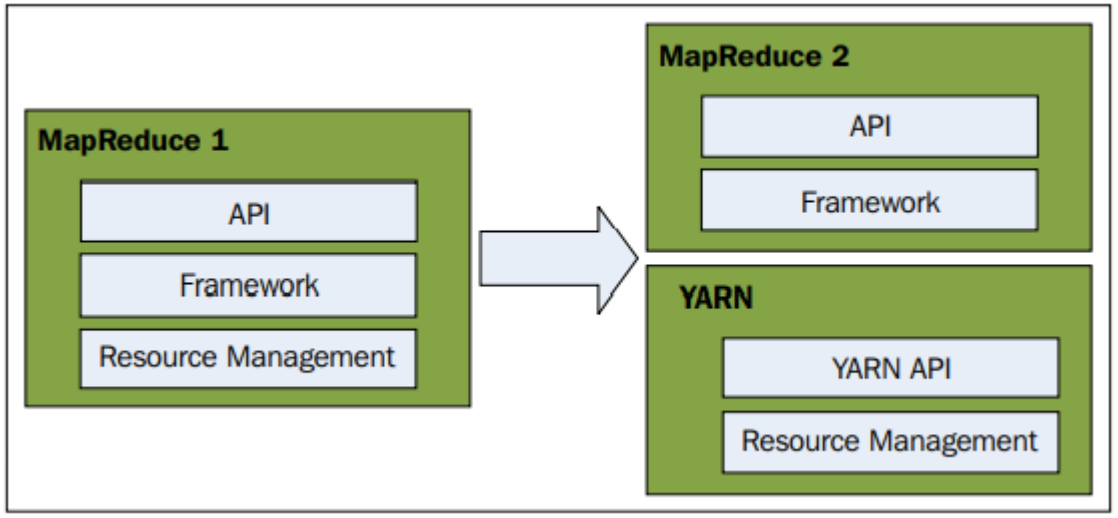
- MapReduce (MR) is a framework to write analytical applications in batch mode on terabytes or petabytes of data stored on HDFS.
- An MR job usually processes each block (excluding replicas) of input file(s) in HDFS with the mapper tasks in a parallel manner.
- The MR framework sorts and shuffles the outputs of the mappers to the reduce tasks in order to produce the output.
- The framework takes care of computing the number of tasks needed, scheduling tasks, monitoring them, and re-executing them if they fail.
- The developer needs to focus only on writing the business logic, and all the heavy lifting is done by the HDFS and MR frameworks
- However, the number of reducer tasks can be increased to provide parallelism at the reducer level.

MapReduce cont.

- For example, in Figure above, if an MR job is submitted for File1, one map task will be created and run on any Node 1, 2, or 3 to achieve data locality.
- In the case of File2, two map tasks will be created with map task 1 running on Node 1, 3, or 4, and map task 2 running on Node 1, 2, or 3, depending on resource availability. The output of the mappers will be sorted and shuffled to reducer tasks. By default, the number of reducers is one.

MapReduce v1 versus MapReduce v2

- MapReduce v1, which is also called Classic MapReduce, has three main components
  - An API to develop MR-based applications
  - A framework to execute mappers, shuffle the data, and execute reducers
  - A resource management framework to schedule and monitor resources
- MapReduce v2 s resource management t



MapReduce v1 challenges

- Inflexible CPU slots configured on a cluster for Map and Reduce led to the underutilization of the cluster
- Resources could not be shared with non-MR applications (for example, Impala or Spark)
- Limited scalability, only up to 4,000 nodes

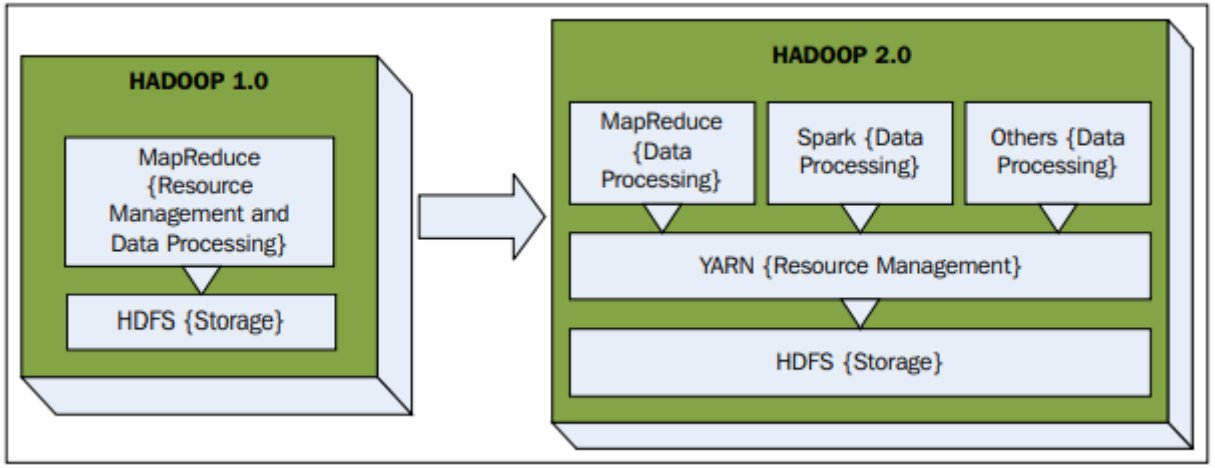
MapReduce v1 versus MapReduce v2. cont.

	MR v1	MR v2
Components used	Job tracker as master and task tracker as slave	Resource manager as master and node manager as slave
Resource allocation	DataNodes are configured to run a fixed number of map tasks and reduce tasks	Containers are allocated as needed for any type of task
Resource management	One job tracker per cluster, which supports up to 4,000 nodes	One resource manager per cluster, which supports up to tens of thousands of nodes
Types of jobs	MR jobs only	Supports MR and other frameworks such as Spark, Impala, and Giraph

YARN

- YARN is the resource management framework that enables an enterprise to process data in multiple ways simultaneously for batch processing, interactive analytics, or real-time analytics on shared datasets.
- While HDFS provides scalable, fault-tolerant, and cost-efficient storage for Big Data, YARN provides resource management to clusters.
- YARN is like an operating system for Hadoop, which manages the cluster resources (CPU and Memory) efficiently.
- Applications such as MapReduce, Spark, and others request YARN to allocate resources for their tasks.
- YARN allocates containers on nodes with the requested amount of RAM and virtual CPU from the total available on that node

Hadoop 1.0 and 2.0 frameworks



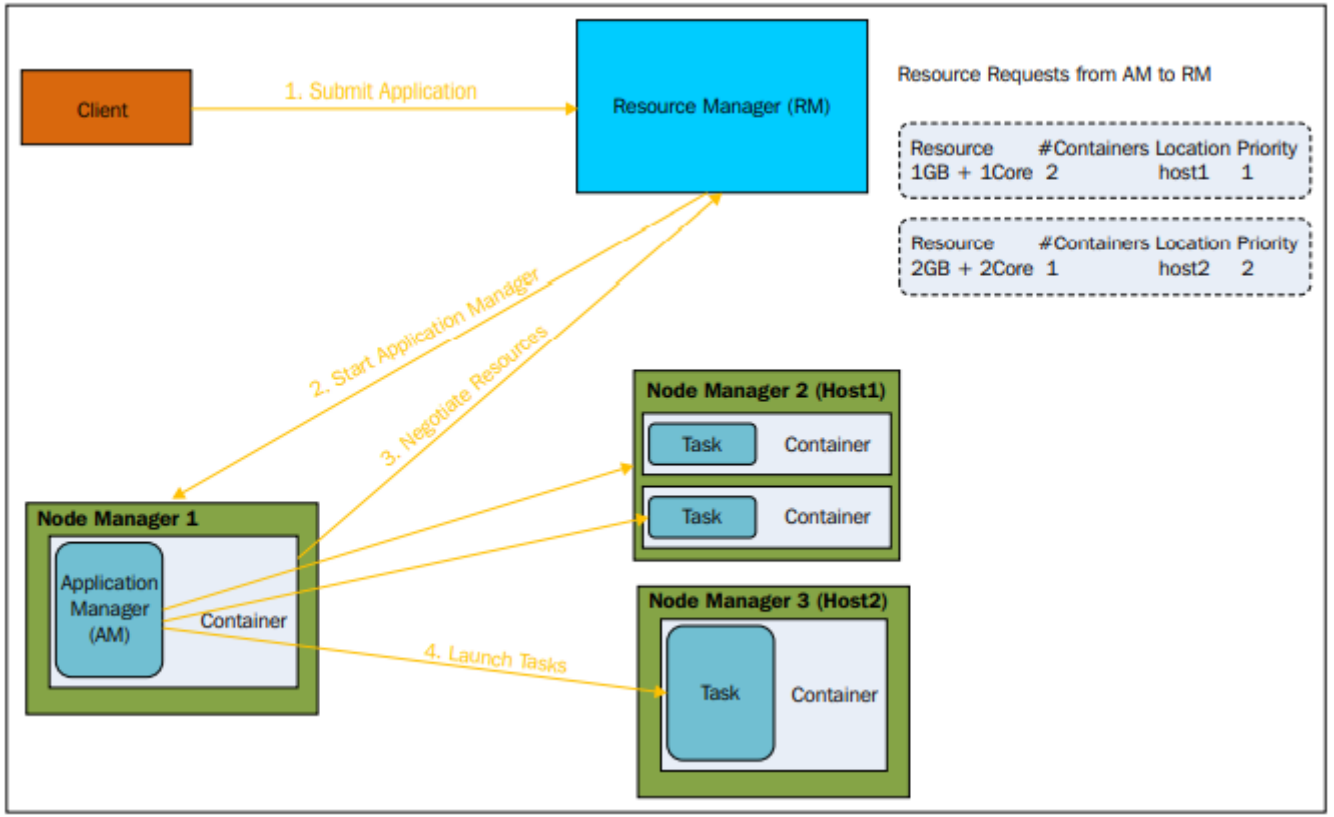
YARN Cont.1

- YARN's original purpose was to split up the two major responsibilities of the JobTracker/TaskTracker (which are part of MapReduce v1) into separate entities:
  - ResourceManager
  - A per-application ApplicationMaster
  - A per-node slave NodeManager
  - A per-application container running on NodeManage
- ResourceManager keeps track of the resource availability of the entire cluster and provides resources to applications when requested by ApplicationMaster
- ApplicationMaster negotiates the resources needed by the application to run their tasks.
  - ApplicationMaster also tracks and monitors the progress of the application.
  - Note that this monitoring functionality was handled by TaskTrackers and JobTrackers in MR v1, which led to overloading the JobTracker. .

YARN Cont.2

- NodeManager is responsible for launching containers provided by ResourceManager, monitoring the resource usage on the slave nodes, and reporting to ResourceManager.
- The application container is responsible for running the tasks of the application.

The YARN application life cycle



Storage options on Hadoop

- When storing data and building applications on Hadoop, some fundamental questions arises:
  - What storage format is useful for my application?
  - What compression codec is optimum for my application?
- Choosing the right file format and compression codec provides optimum performance for the use case that you are working on
- Data can be stored in its raw form using the standard file format or the special Hadoop container file format that offers benefits in specific use case scenarios, which can be split even when data is compressed. Broadly,

File formats

- Standard file formats:
  - Structured text data: CSV, TSV, XML, and JSON files
  - Unstructured text data: Log files and documents
  - Unstructured binary data: Images, videos, and audio files
- Hadoop file formats: Provides splittable compression
  - File-based structures: Sequence file
  - Serialization format: Thrift, Protocol buffers, Avro
  - Columnar formats: RCFile, ORCFile, Parquet

File formats cont.

- Sequence file
  - Sequence files store data as binary key-value pairs. It supports the Java language only and does not support schema evolution. It supports the splitting of files even when the data is compressed
- Protocol buffers and thrift
  - Protocol buffers were developed by Google and open sourced in 2008.
  - Thrift was developed at Facebook and offers more features and language support than protocol buffers.
  - Both of these are serialization frameworks that offer high performance while sending over the network.
- Avro
  - is a row-based data serialization system used for storage and sends data over the network efficiently
  - Rich data structures
  - Compact and fast binary data format
  - Simple integration with any language
  - Support for evolving schemas
  - Great interoperability between Hive, Tez, Impala, Pig, and Spark

File formats cont.

- Parquet
  - Parquet is a columnar format that skips I/O and decompression (if applicable) on columns that are not part of the query.
  - It is generally very efficient in terms of compression on columns because column data is similar within the same column than it is in a block of rows.
- RCFile and ORCFile
  - Record Columnar File (RCFile) was the first columnar format for Hive that provided efficient query processing. Optimized Row Columnar (ORC) format was introduced in Hive 0.11 and offered better compressions and efficiency than the RCFile format. ORCFile has lightweight indexing that enables the skipping of irrelevant columns

Compression formats

- Compressed data can speed up I/O operations
- Compressed data saves storage space
- Compressed data speeds up data transfer over the network
- But, Compression and decompression increases CPU time.

Compression format	Tool	Algorithm	File extension	Splittable?
gzip	Gzip	DEFLATE	`.gz`	No
bzip2	bizp2	bzip2	`.bz2`	Yes
LZO	Lzop	LZO	`.lzo`	Yes, if indexed
Snappy	N/A	Snappy	`.snappy`	No

Assignment: How to install Apache hadoop

- https://www.youtube.com/watch?v=g7Qpnmi0Q-s&t=142s