جامعة الأمير سلطان
PRINCE SULTAN UNIVERSITY

CS435 Distributed systems

HADOOP

Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

TOPICS

- Why Hadoop?
- HDFS
- YARN
- MapReduce
- Summary

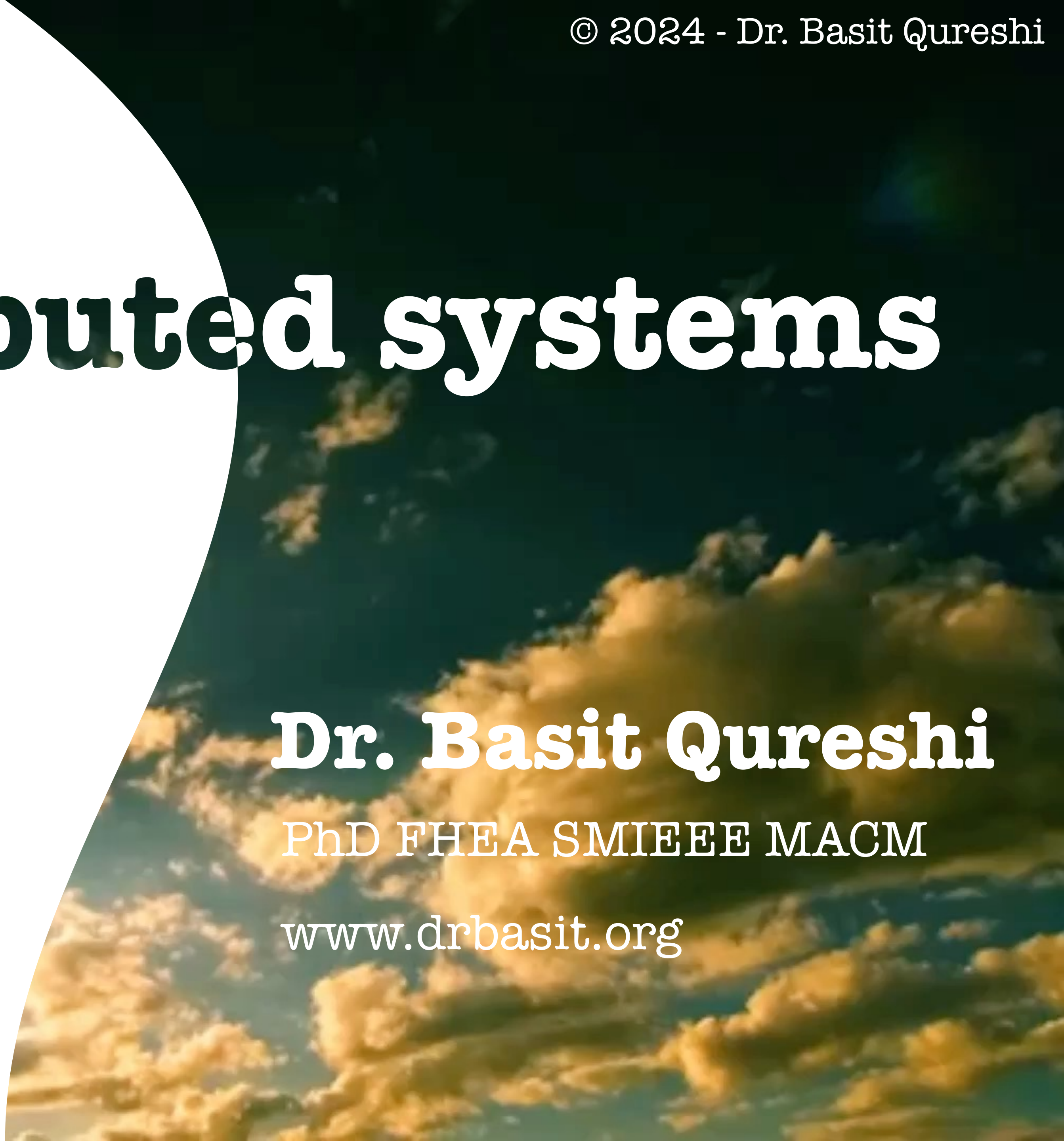
CS435 Distributed systems

WHY HADOOP?

Dr. Basit Qureshi

PhD FHEA SMIEEE MACM

www.drbasit.org

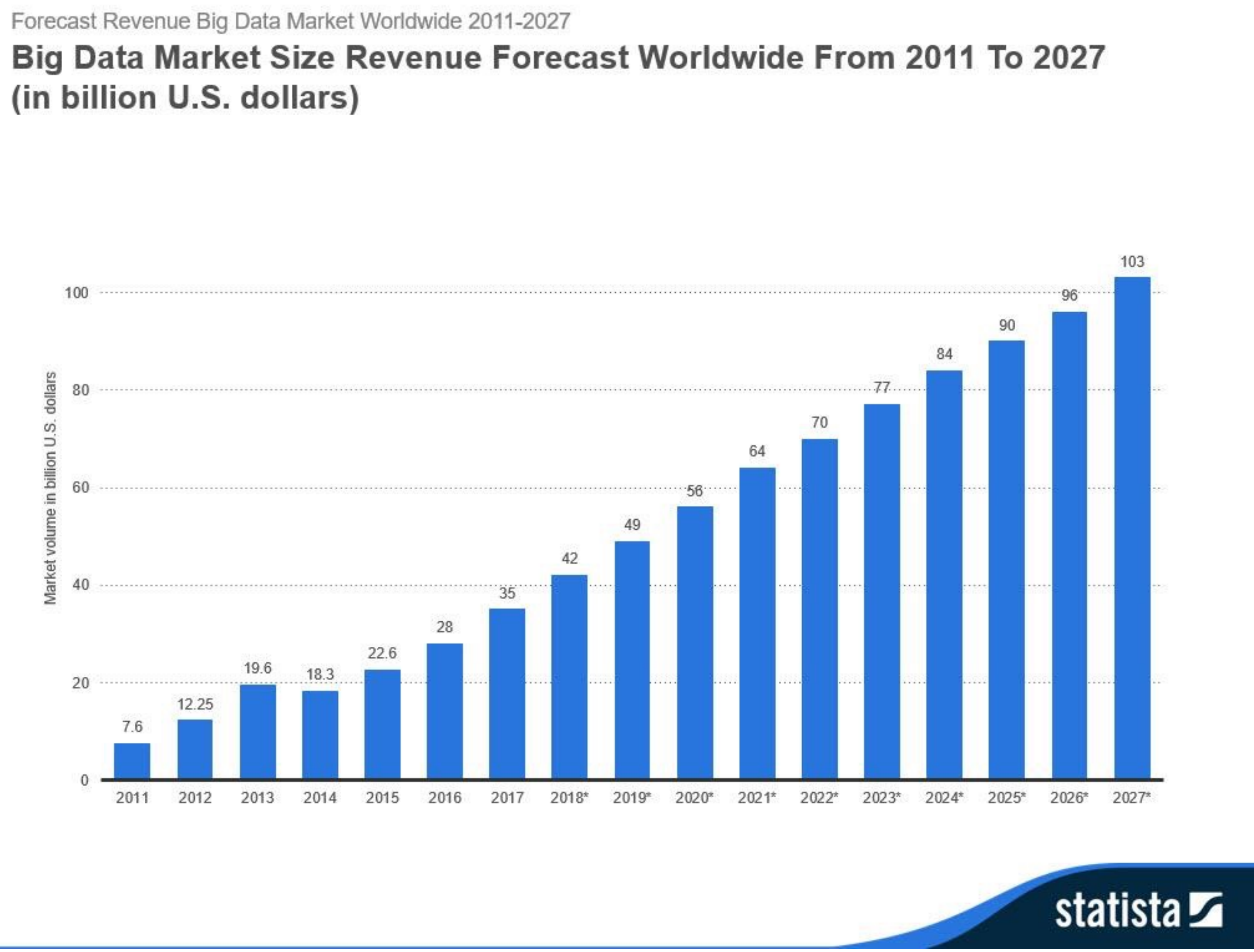


WHY HADOOP?

Big Data: The amount of data in the world is expanding and increasing without limit, driven by artificial intelligence, IoT, cloud computing and other technologies.

Global market for Big Data revenue is expected to reach 100 Billion USD in 2027.

- The global Data volume reached 41 ZettaBytes in 2019
- Expected to reach 163 ZB by 2025
- 1ZB = 1000 exabytes
- 1exabytes = 1 million terabytes

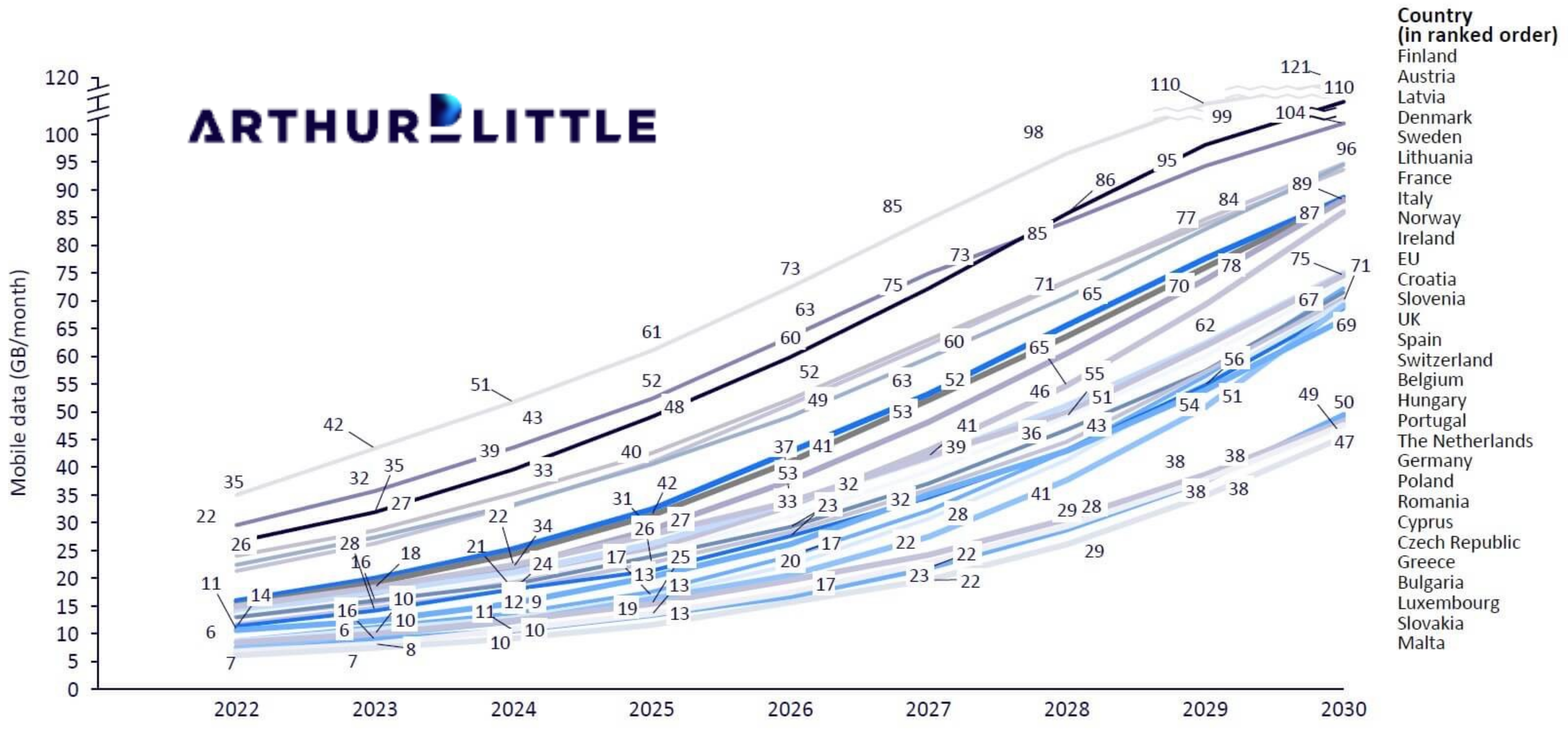


Source: Statista

WHY HADOOP?

Big Data Growth: Driven by video and social networks, in Europe alone, it is anticipated that the average mobile data consumption to continue growing rapidly from approximately 15 GB/month in 2022 to 75-80 GB/month by 2030.

This creates an overall annual growth rate of 25%.



Source: Arthur D. Little

https://www.adlittle.com/en/insights/report/evolution-data-growth-europe

WHY HADOOP?

Open Source: Hadoop has a large community that continuously improves the framework

Big Data: Hadoop is designed to handle vast amounts of data

Distributed Storage: Distributed nature allows it to store data efficiently

Distributed Computing: Hadoop's MapReduce programming model allows for parallel processing

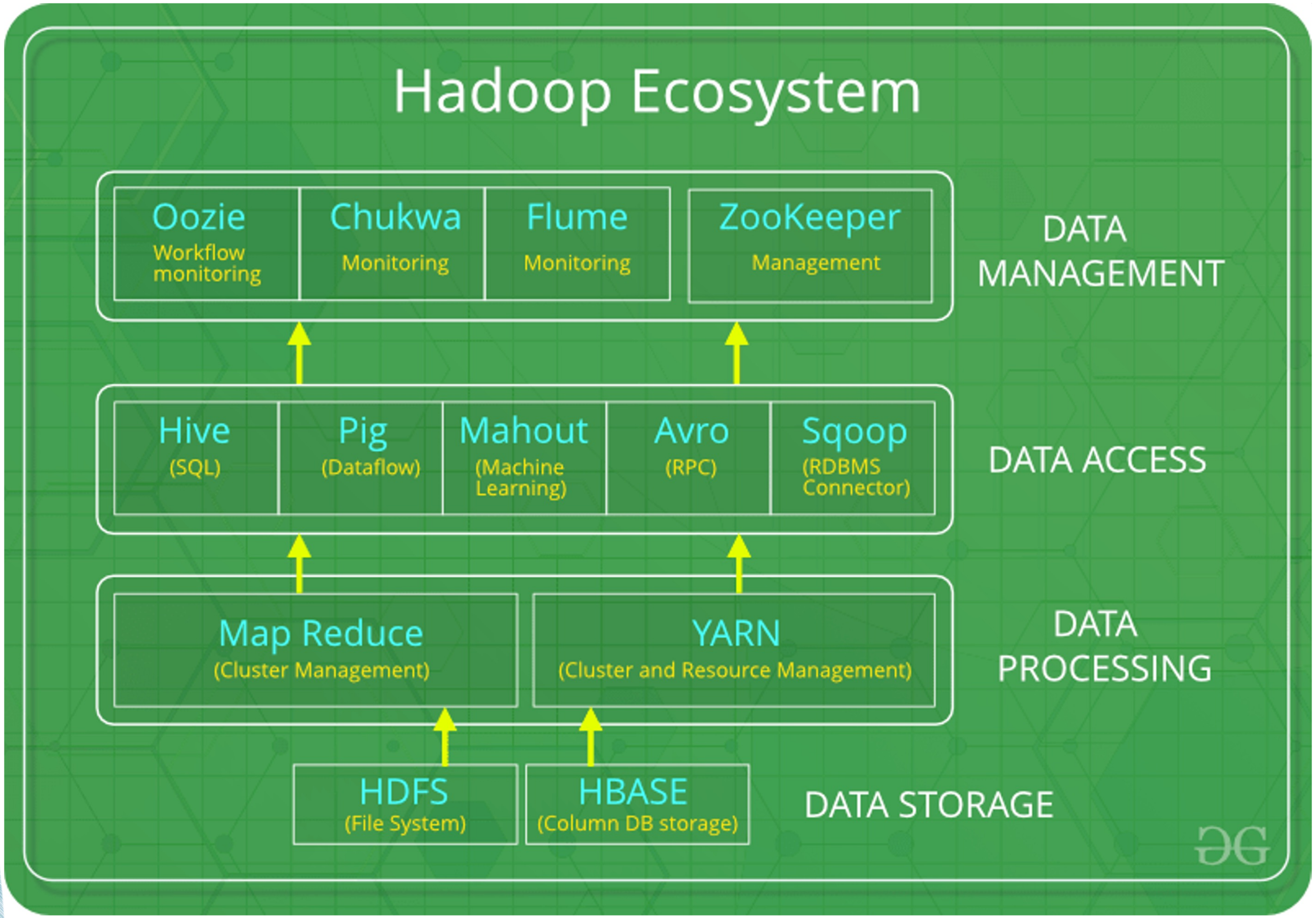
Scalable: Scales horizontally by adding more nodes (systems) to the cluster

Fault Tolerance: If one node fails, the data can still be accessed

Cost effective: Inexpensive Commodity servers

Eco-System: Hadoop is part of a larger ecosystem that includes tools like Hive, Pig, HBase, and Spark,

WHY HADOOP?

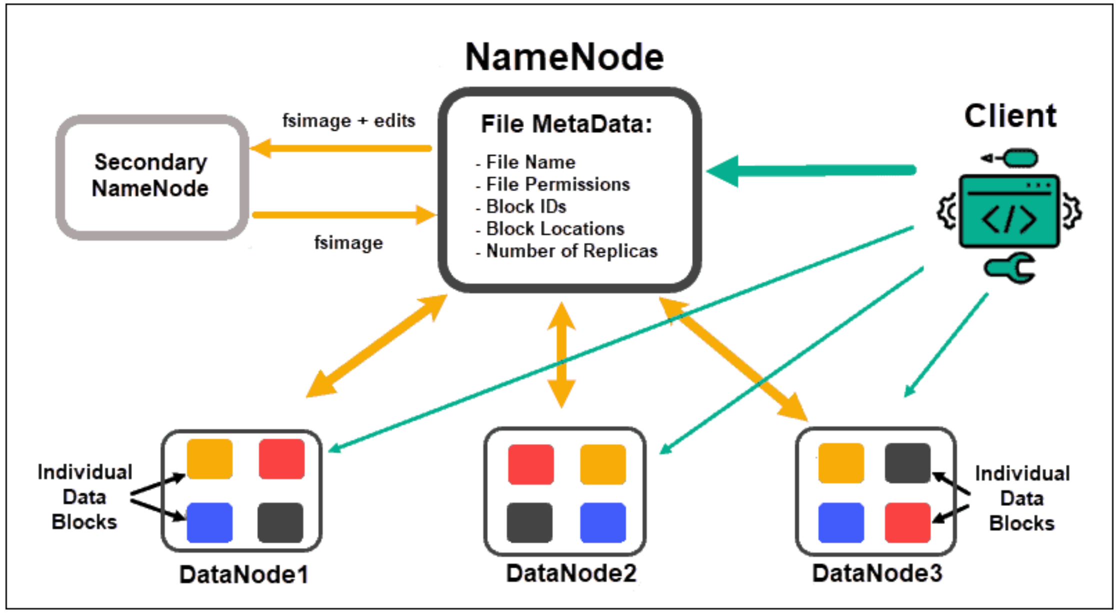


Hadoop Ecosystem
Oozie
Workflow monitoring
Chukwa
Monitoring
Flume
Monitoring
ZooKeeper
Management
DATA MANAGEMENT
Hive
(SQL)
Pig
(Dataflow)
Mahout
(Machine Learning)
Avro
(RPC)
Sqoop
(RDBMS Connector)
DATA ACCESS
Map Reduce
(Cluster Management)
YARN
(Cluster and Resource Management)
DATA PROCESSING
HDFS
(File System)
HBASE
(Column DB storage)
DATA STORAGE

WHY HADOOP?

Hadoop Distributed File System (HDFS)

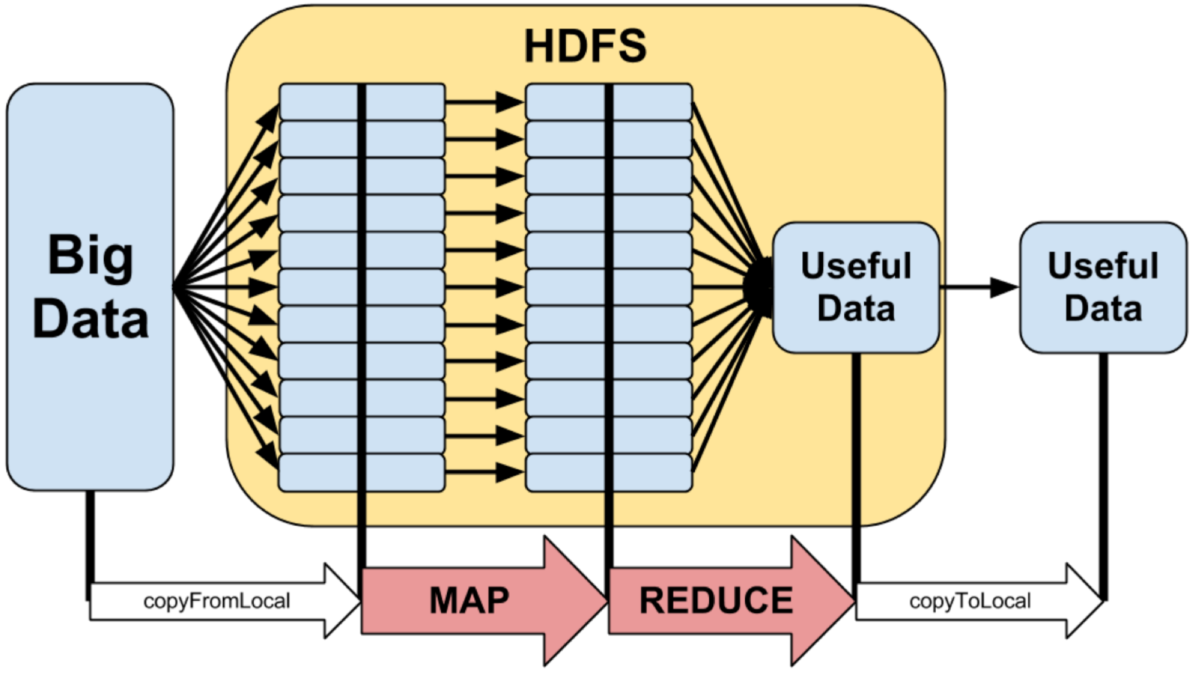
- HDFS is the primary component of Hadoop ecosystem and is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
- HDFS consists of two core components i.e.
  - **NameNode**: Manages the filesystem namespace and controls access to files by clients. It keeps track of which blocks are stored on which DataNodes.
  - **DataNodes**: Store the actual data blocks. They handle read and write requests from clients and periodically report back to the NameNode with the status of the blocks they store.



WHY HADOOP?

MapReduce

- A programming model for distributed computation on large datasets across a cluster of computers. It simplifies data processing tasks by breaking them down into smaller, manageable chunks that can be processed in parallel, making it highly effective for big data analysis.
- **Map Function**: This function takes input data and processes it into key-value pairs. It performs data filtering and transformation. The output of the map function is then passed to the reduce function
- **Reduce Function**: This function takes the output from the map function, aggregates it, and produces the final result.

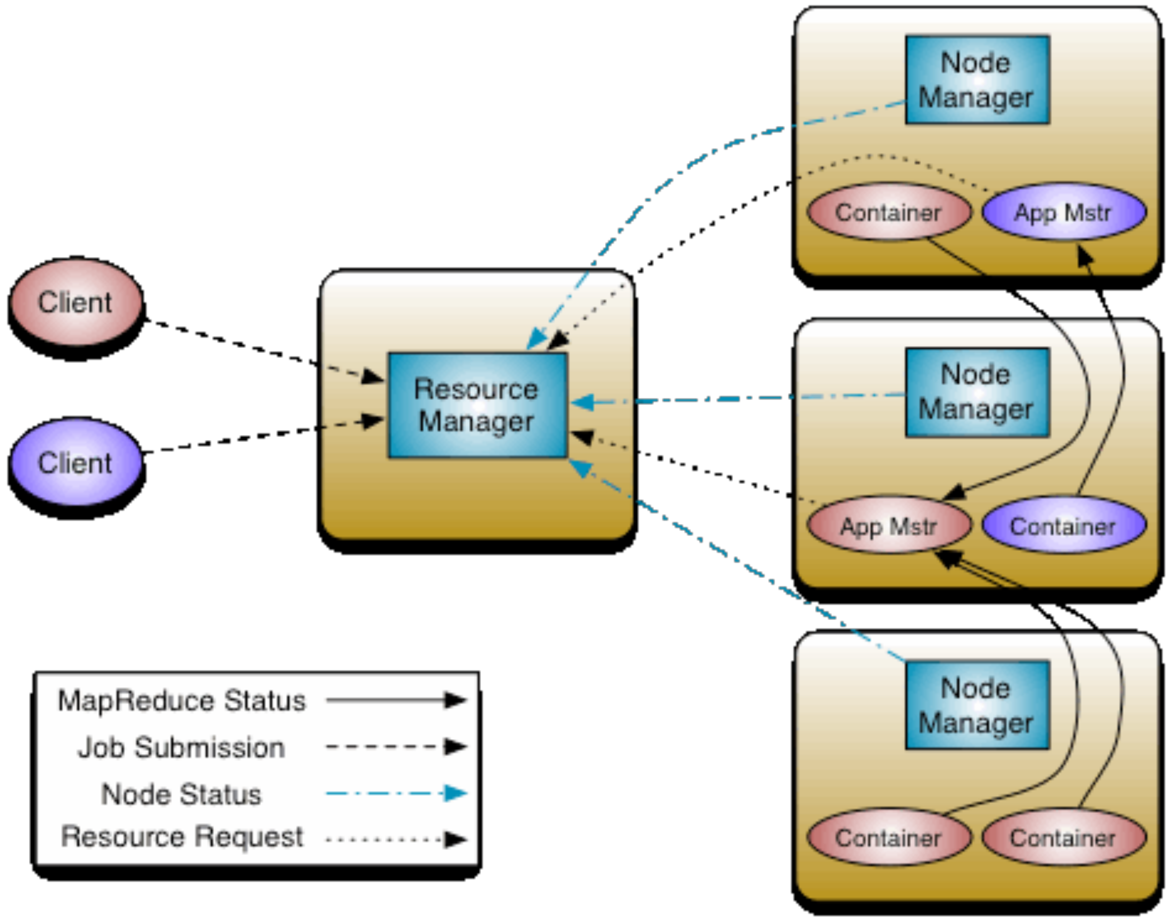


https://www.glennklockwood.com/data-intensive/hadoop/overview.html

WHY HADOOP?

Yet Another Resource Negotiator (YARN)

- YARN performs scheduling and resource allocation for the Hadoop System
- Consists of three major components i.e.
    1. **Resource Manager:** Allocates resources for the applications in a system
    2. **Nodes Manager:** Allocates resources such as CPU, memory, bandwidth on a machine (node)
    3. **Application Manager/Master:** Interface between the resource manager and node manager and performs negotiations



WHY HADOOP?

Other components

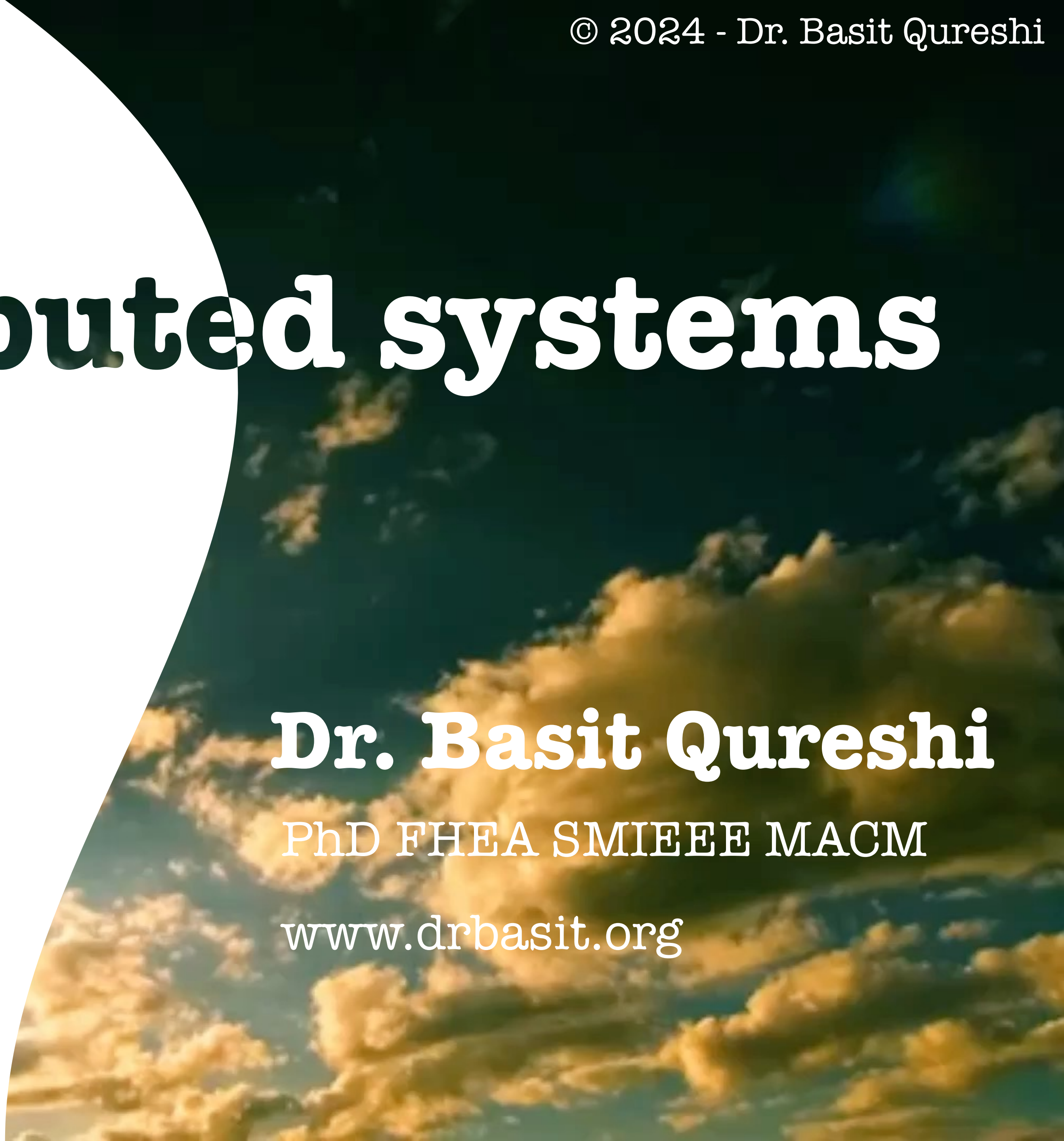
- HIVE: Similar to SQL but designed for large datasets
- PIG: Similar to SQL designed for Yahoo
- Mahout: Supports Machine Learning
- Spark: Designed for real-time big data processing
- Hbase: NoSQL database enables Google Bigtable
- Solr, Lucene: searching and indexing services
- Zookeeper: Data coordination and synchronization
- Oozie: Task scheduler and workflow management

WHY HADOOP?

- **Companies using Hadoop**
- Yahoo
- Microsoft
- Facebook
- Amazon
- Netflix
- LinkedIn
- eBay
- Twitter
- Airbnb
- Spotify
- Adobe
- Bank of America
- Walmart

CS435 Distributed systems

HDFS



Dr. Basit Qureshi
PhD FHEA SMIEEE MACM
www.drbasit.org

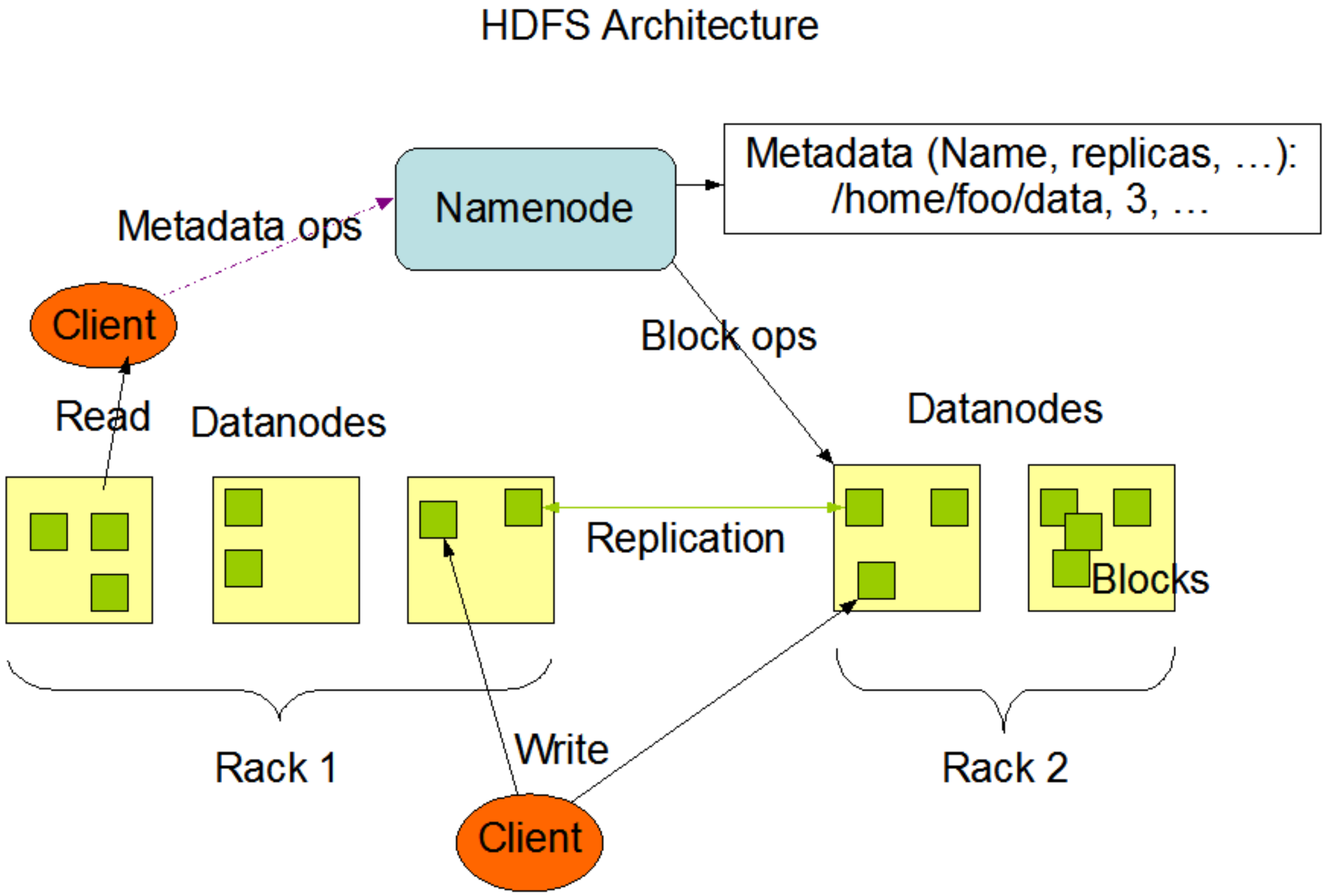
GOALS OF HDFS

- **Very Large Distributed File System**
  - 10,000 nodes, 100 million files, 10 Pbytes per namenode
  - Can be expanded with federation with multiple namenodes
- **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
- **Data Coherency**
  - Write-once-read-many access model
  - Client can only append to existing files
- Files are broken up into blocks
  - Typically **64MB-128MB** block size
  - Each block replicated on multiple DataNodes

HDFS ARCHITECTURE

Major components of HDFS

1. **NameNode**: Manages metadata and file system namespace.
2. **DataNode**: Stores actual data blocks.
3. **Client**: Interacts with HDFS on behalf of users.
4. **Blocks**: Basic unit of data storage in HDFS.



NAMENODE

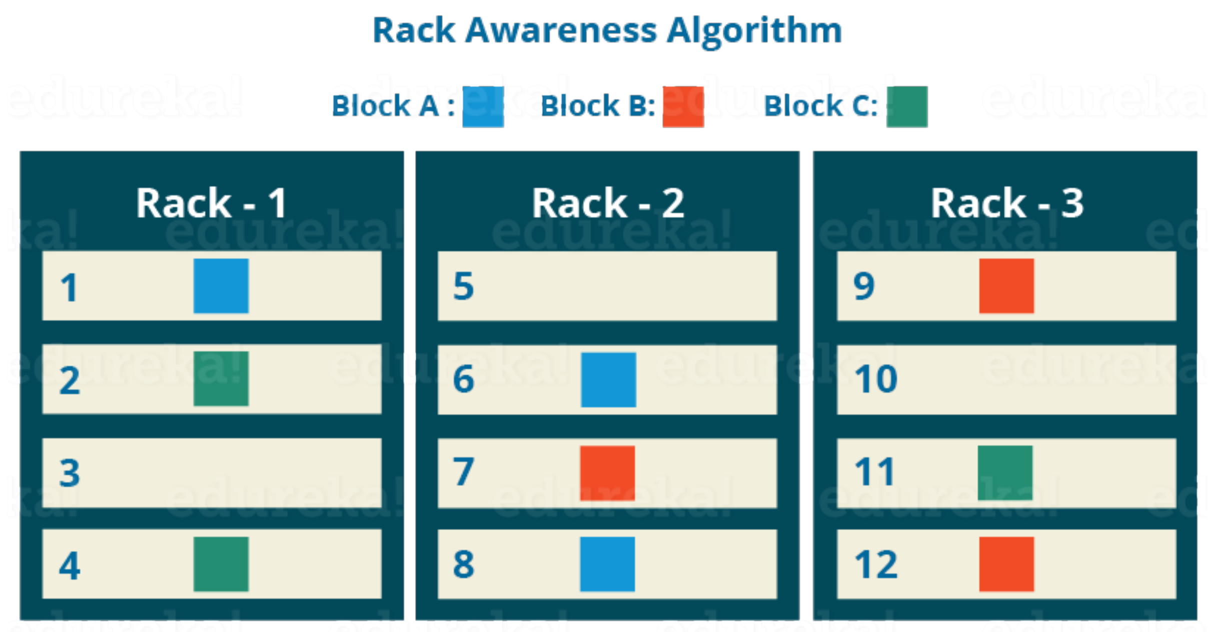
Namenode: The Manager of the Cluster

- Manages File System Namespace
  - Directory Structure
  - Handles create, delete, rename, move files/blocks
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides
- Metadata storage
  - File-to-block mapping (which blocks belong to which files).
  - Block locations (on which DataNodes the blocks reside).
- Block management
  - Allocates blocks for new files. Determines what DataNodes should store these blocks
  - Manages replication factor for each block
- Coordination with Datanodes using heart-beats
- Detects and manages failures
- Interacts with Clients

Transparency

DATANODE

- A Datanode is the worker node
- A Block Server
    - Stores actual Blocks in HDFS
    - Read/Write data in the local file system Stores metadata of a block
    - 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



Replication for Failure management

FAILURE HANDLING

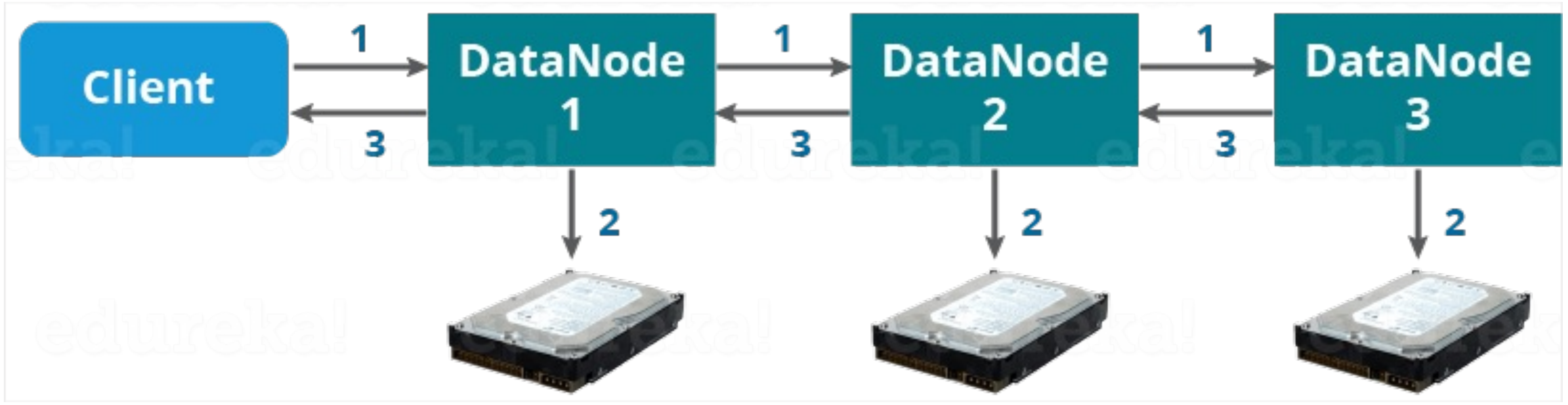
- HEART BEATS: DataNodes send heartbeat to the NameNode periodically
  - Once every 3 seconds
- NameNode uses heartbeats to detect DataNode failure
  - If a DataNode hasn't replied in time;
    - Chooses new DataNodes
    - Distributes replicas to the new DataNodes
  - Balances disk usage
  - Balances communication traffic to DataNodes

DATA PIPELINING (I)

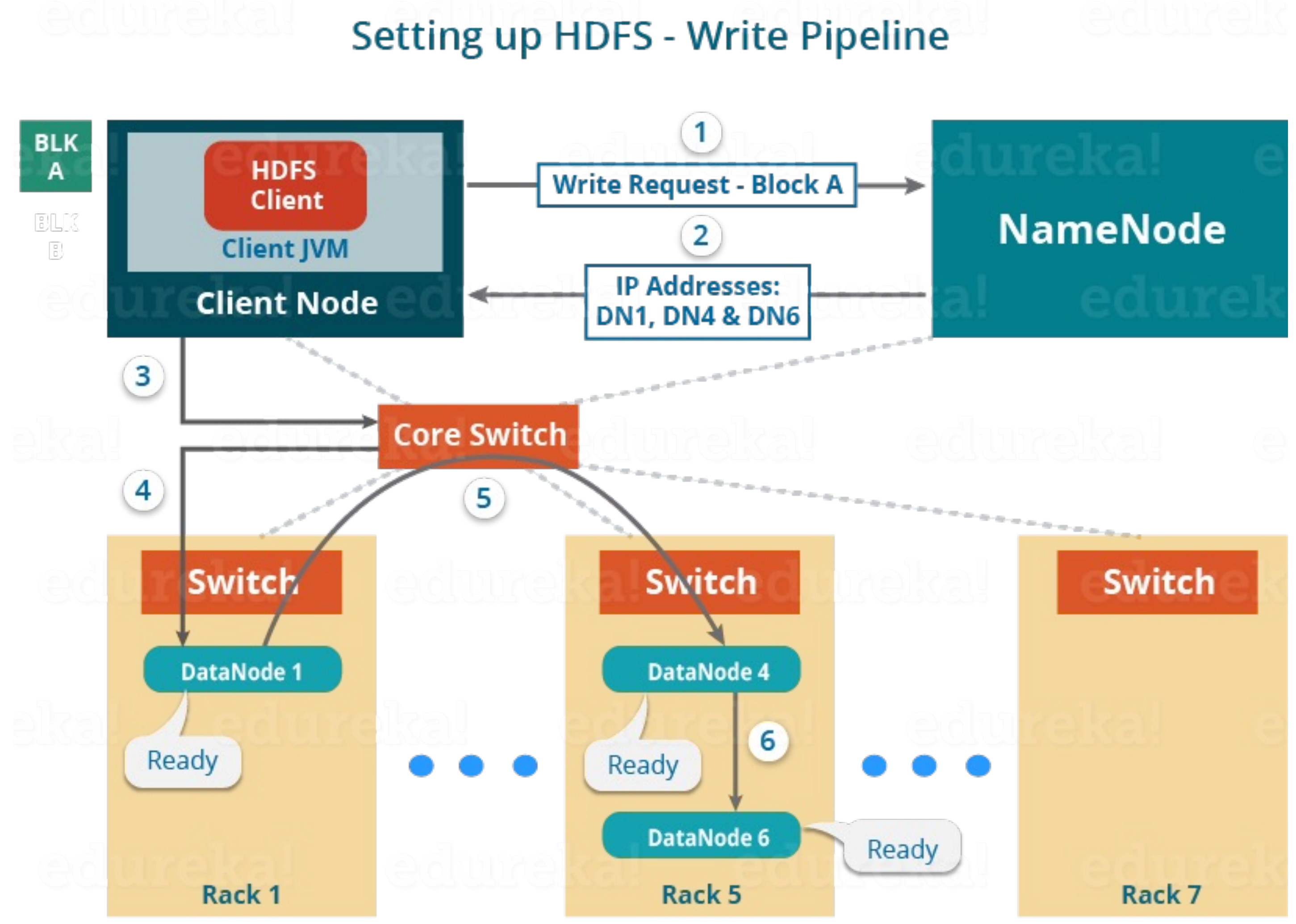
Consistency

Need to update all replicas to ensure Consistency & Data integrity

- 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



DATA PIPELINING (II)



USER INTERFACE

- Commands 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`

CS435 Distributed systems

YARN

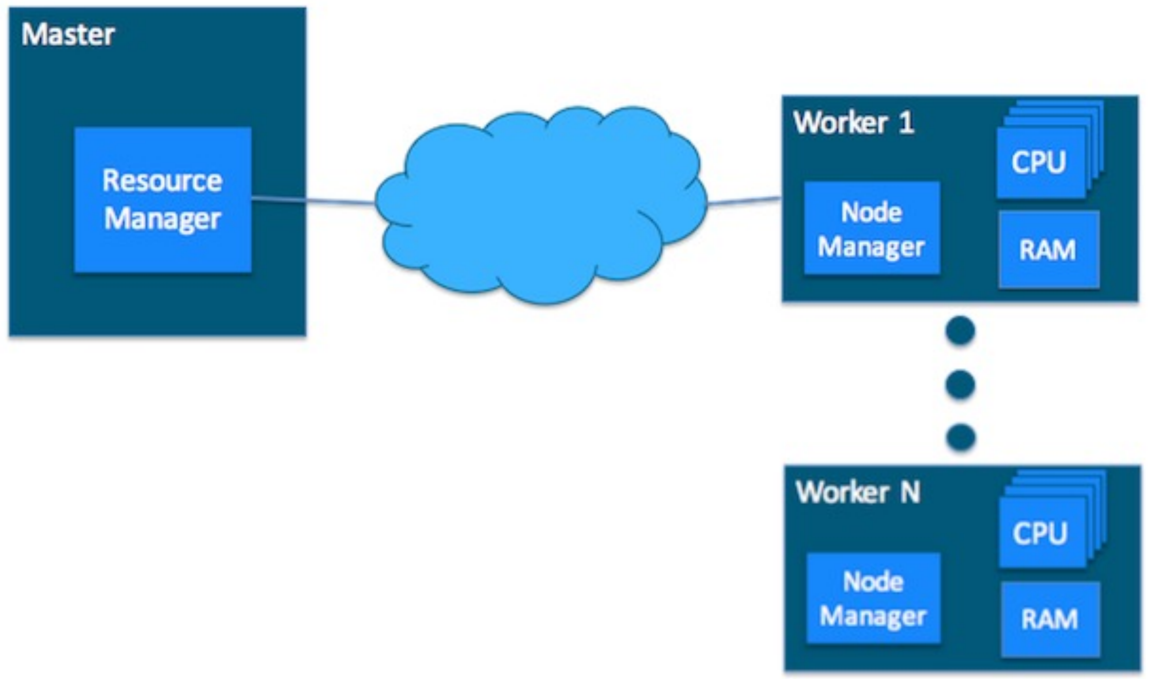
Dr. Basit Qureshi

PhD FHEA SMIEEE MACM

www.drbasit.org

YARN CLUSTER BASICS

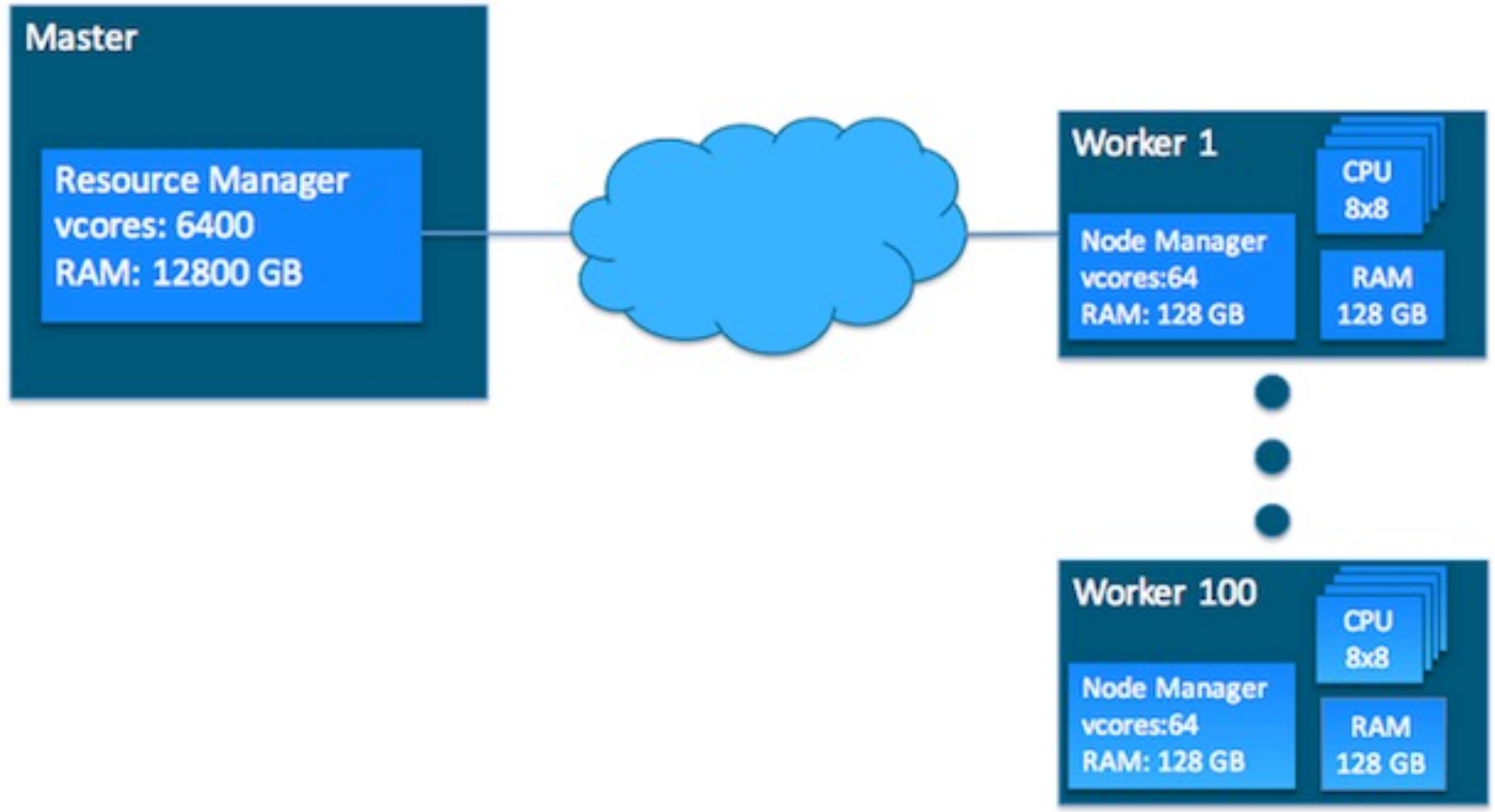
- In a YARN cluster, there are two types of hosts:
  - The *ResourceManager* is the master daemon that communicates with the client, tracks resources on the cluster, and orchestrates work by assigning tasks to *NodeManagers*.
  - A *NodeManager* is a worker daemon that launches and tracks processes spawned on worker hosts.



YARN RESOURCE MONITORING (I)

- YARN currently defines two resources:
  - *v-cores*
  - *memory*.
- Each **NodeManager** tracks
  - its own local resources and
  - communicates its resource configuration to the ResourceManager
- The **ResourceManager** keeps
  - a running total of the cluster's available resources.

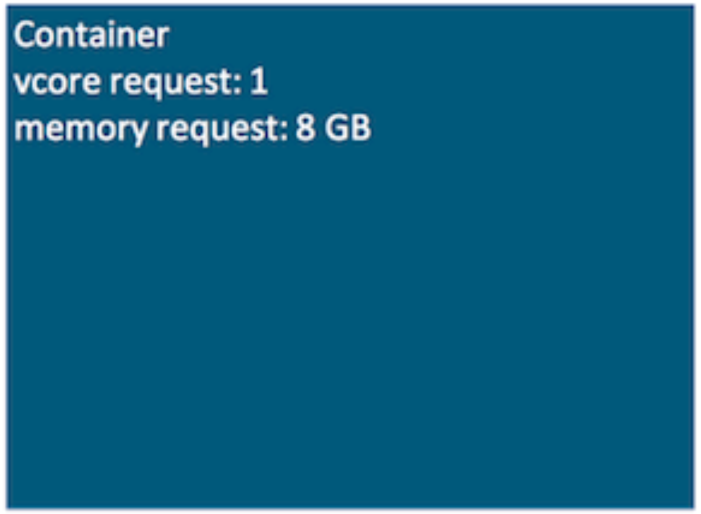
YARN RESOURCE MONITORING (II)



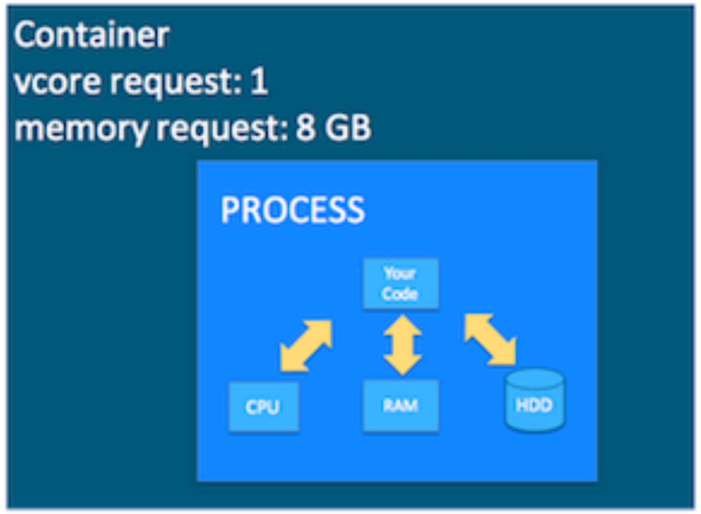
100 workers of same resources

YARN CONTAINER

- Containers
  - a request to hold resources on the YARN cluster.
  - a container hold request consists of vcore and memory



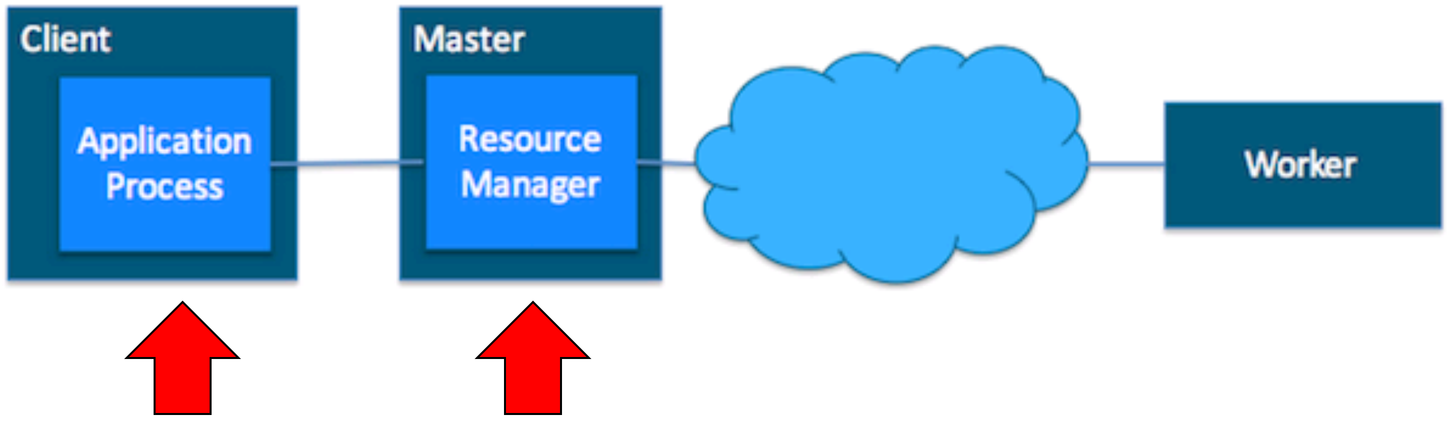
Container as a hold



The task running as a process inside a container

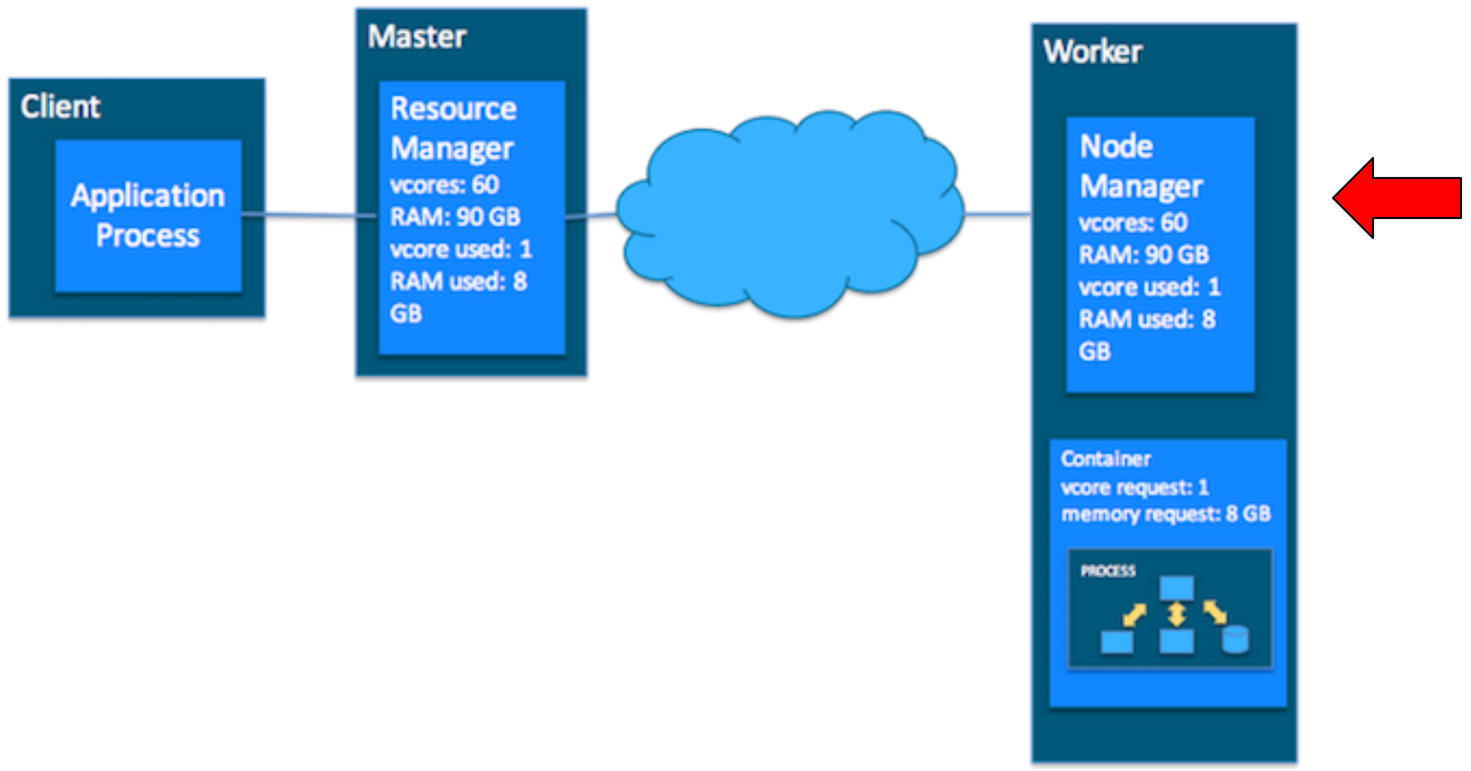
INTERACTIONS AMONG YARN COMPONENTS (I)

1. The application starts and talks to the ResourceManager for the cluster



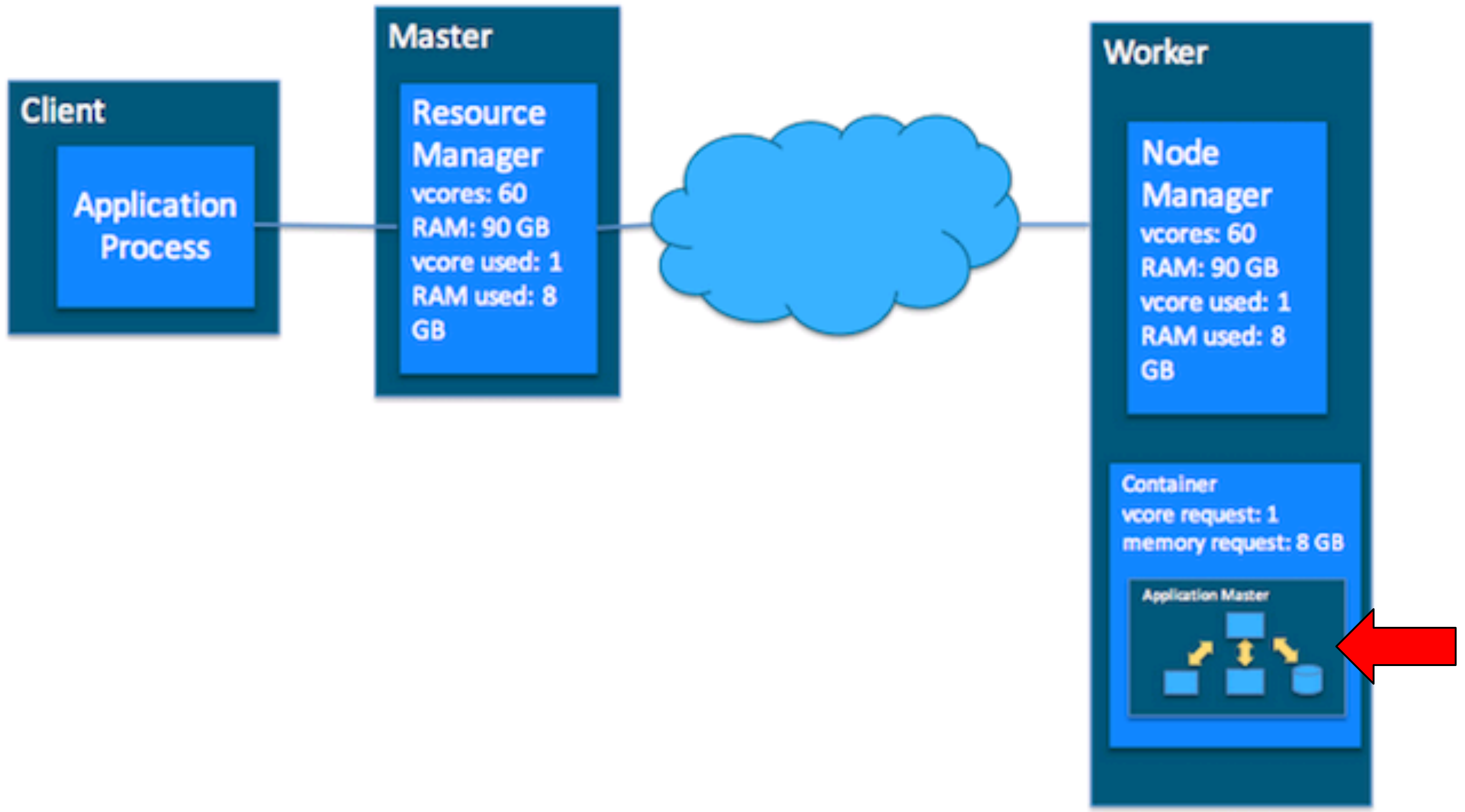
INTERACTIONS AMONG YARN COMPONENTS (II)

2. The ResourceManager makes a single container request on behalf of the application



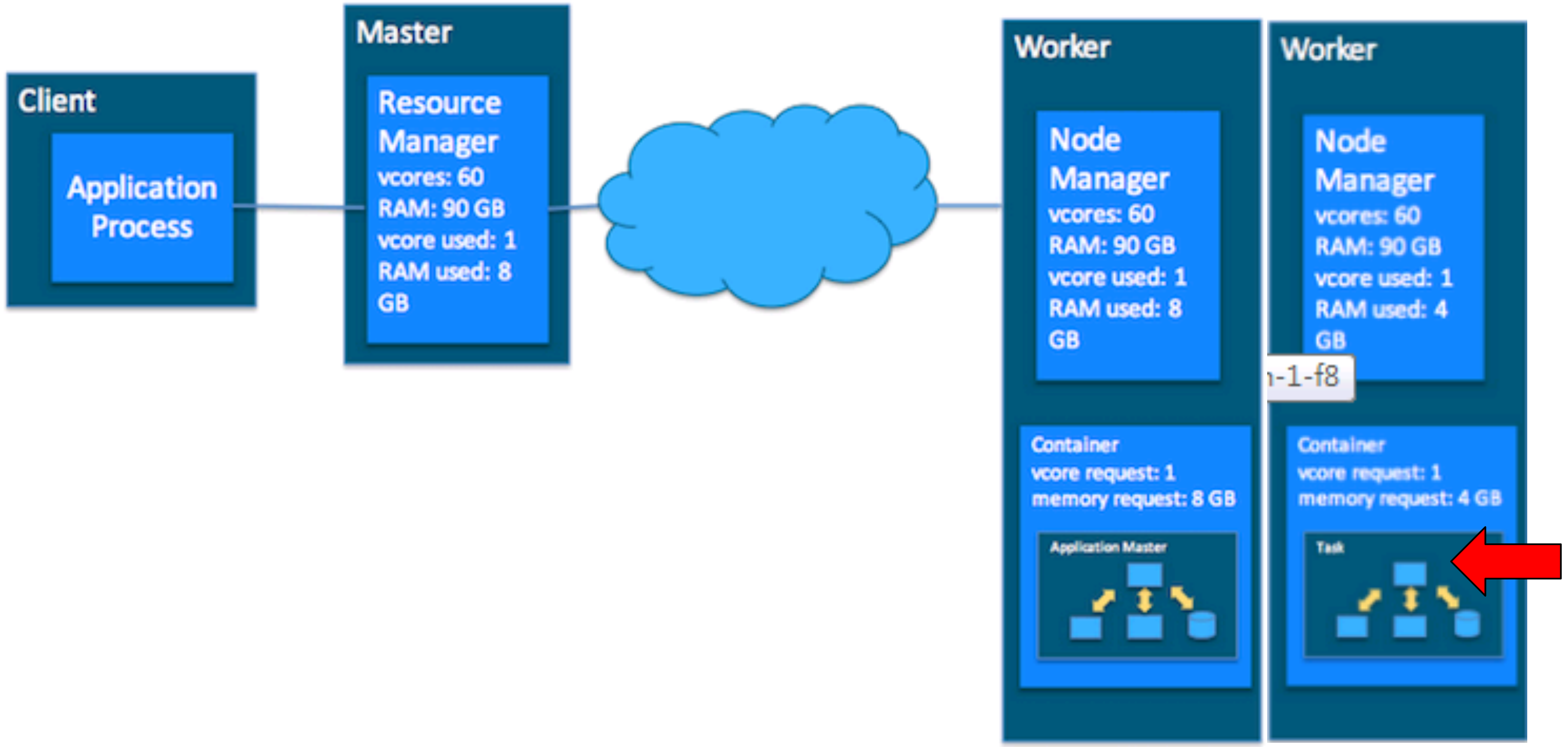
INTERACTIONS AMONG YARN COMPONENTS (III)

3. The ApplicationMaster starts running within that container



INTERACTIONS AMONG YARN COMPONENTS (IV)

4. The ApplicationMaster requests subsequent containers from the ResourceManager that are allocated to run tasks for the application. Those tasks do most of the status communication with the ApplicationMaster allocated in Step 3

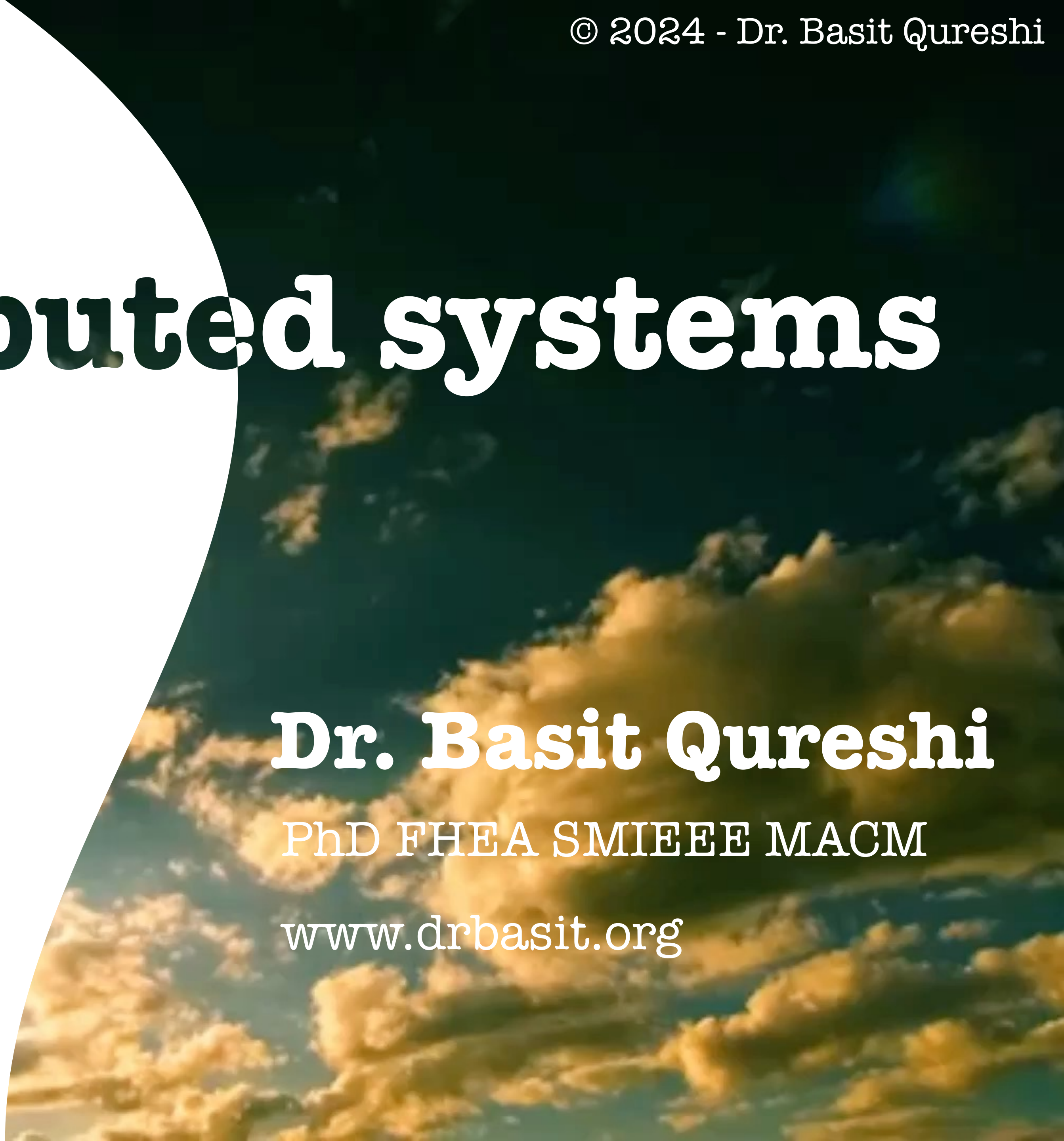


INTERACTIONS AMONG YARN COMPONENTS (V)

5. Once all tasks are finished, the ApplicationMaster exits. The last container is de-allocated from the cluster.

6. The application client exits. (The ApplicationMaster launched in a container is more specifically called a managed AM. Unmanaged ApplicationMasters run outside of YARN’s control.)

CS435 Distributed systems



MAPREDUCE

Dr. Basit Qureshi

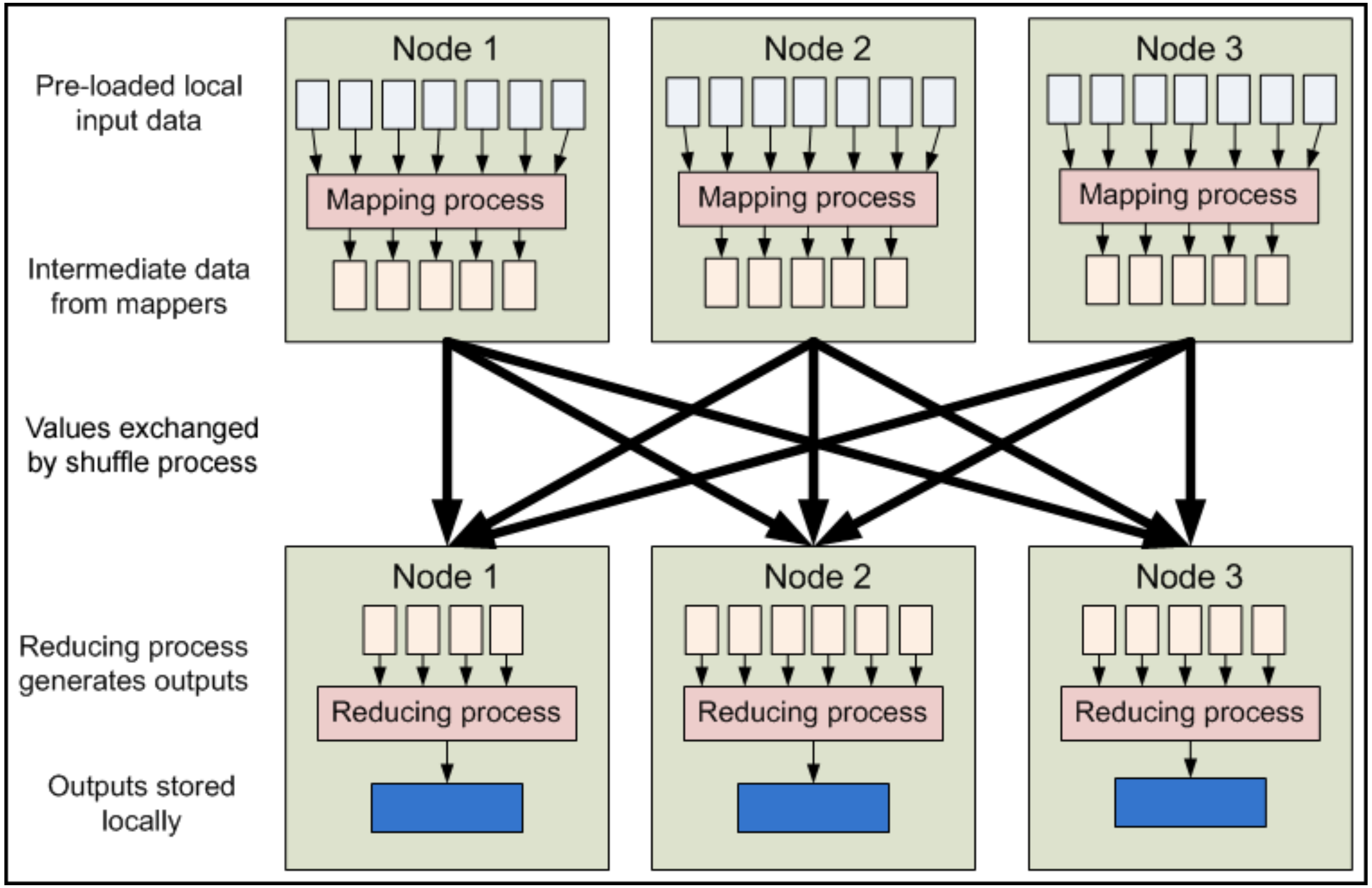
PhD FHEA SMIEEE MACM

www.drbasit.org

MAPREDUCE - WHAT?

- MapReduce is a programming model for efficient distributed computing
- It works like a Unix pipeline
  - cat input | grep | sort | uniq -c | cat > output
  - **Input** **| Map |** Shuffle & Sort **| Reduce** **| Output**
- Efficiency from
  - Streaming through data, reducing seeks
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building

MAPREDUCE - DATAFLOW



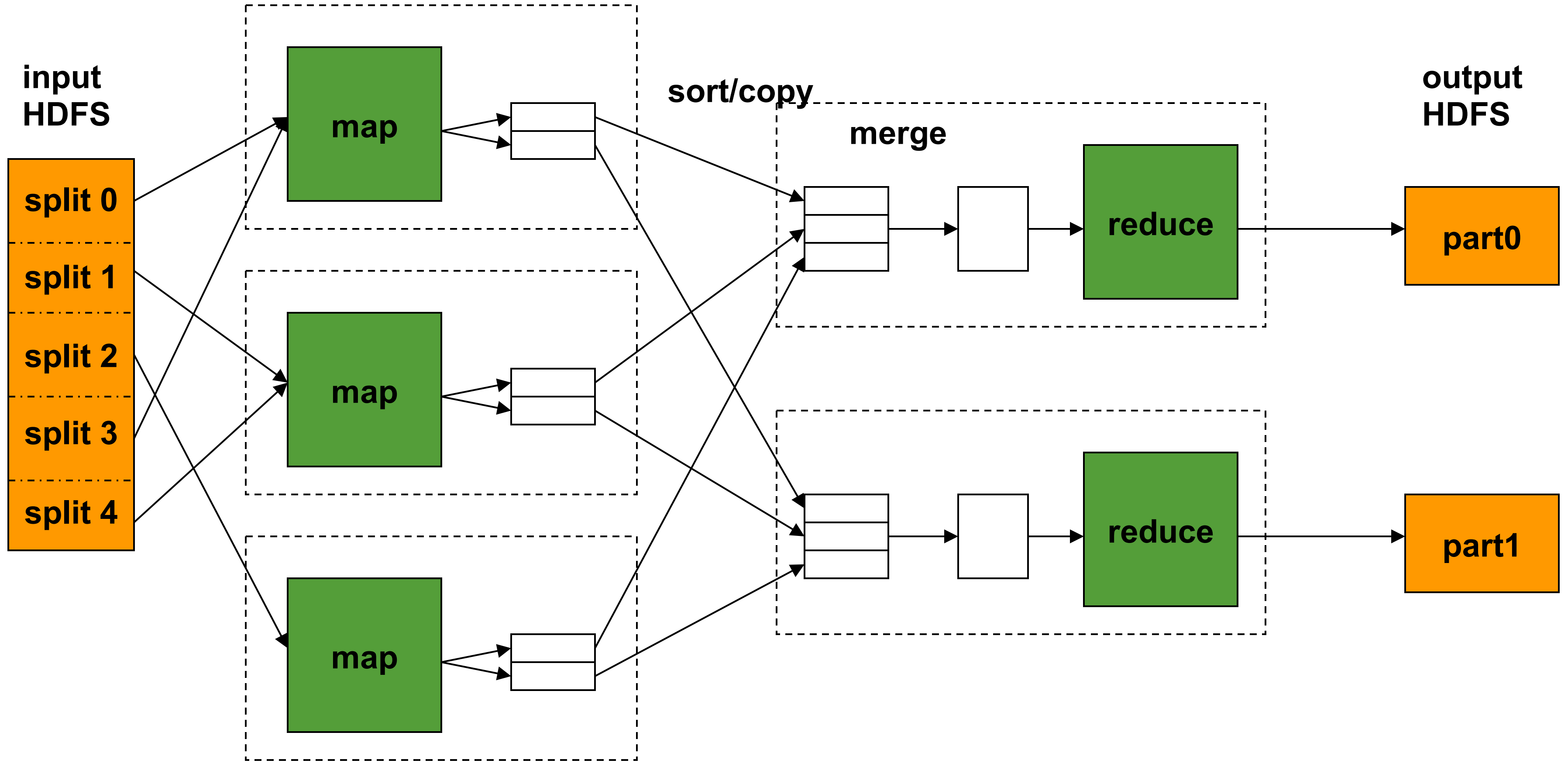
MAPREDUCE - FEATURES

- Fine grained Map and Reduce tasks
  - Improved load balancing
  - Faster recovery from failed tasks
- Automatic re-execution on failure
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks
- Locality optimizations
  - With large data, bandwidth to data is a problem
  - Map-Reduce + HDFS is a very effective solution
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible

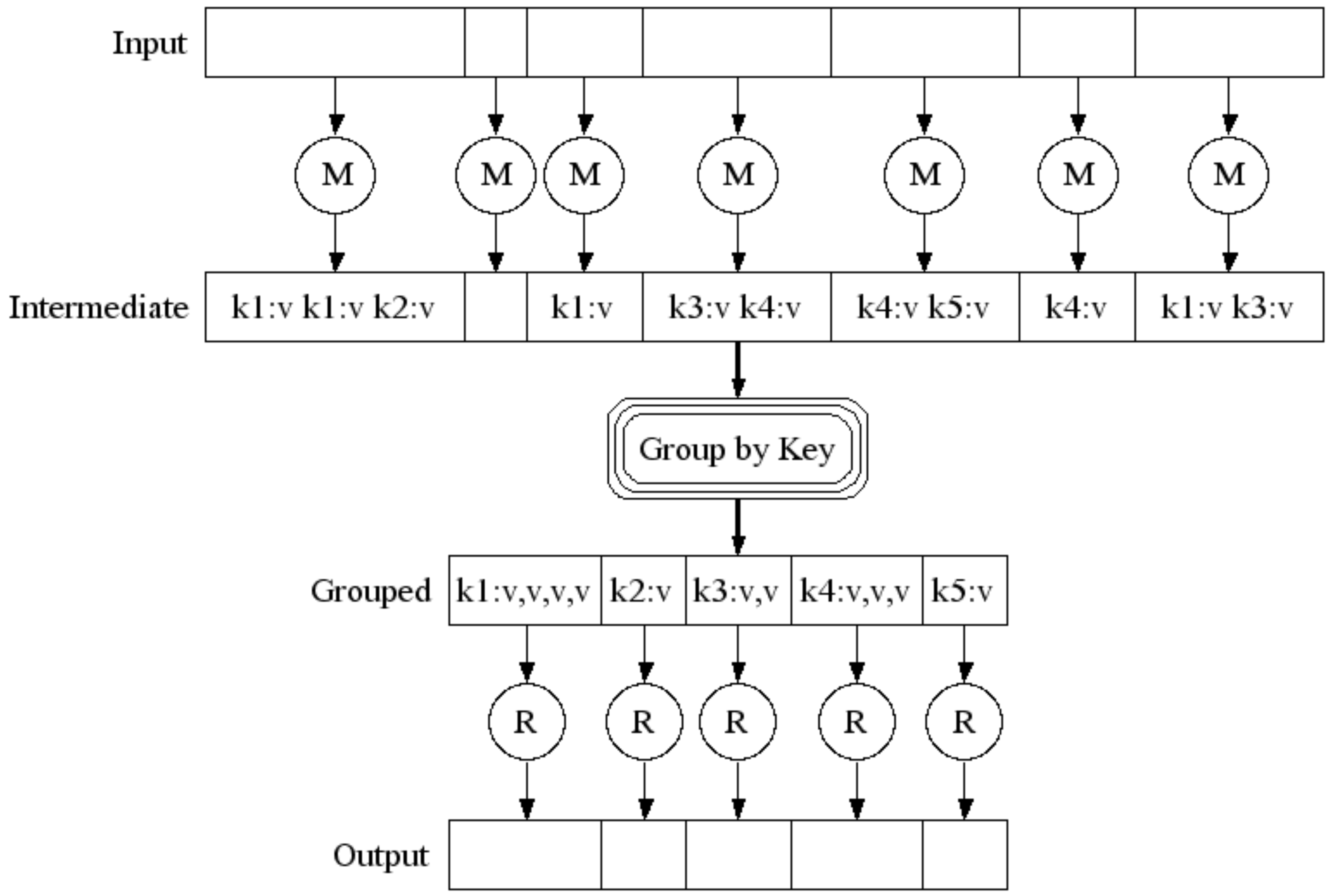
WORD COUNT EXAMPLE

- Mapper
  - Input: value: lines of text of input
  - Output: key: word, value: 1
- Reducer
  - Input: key: word, value: set of counts
  - Output: key: word, value: sum
- Launching program
  - Defines this job
  - Submits job to cluster

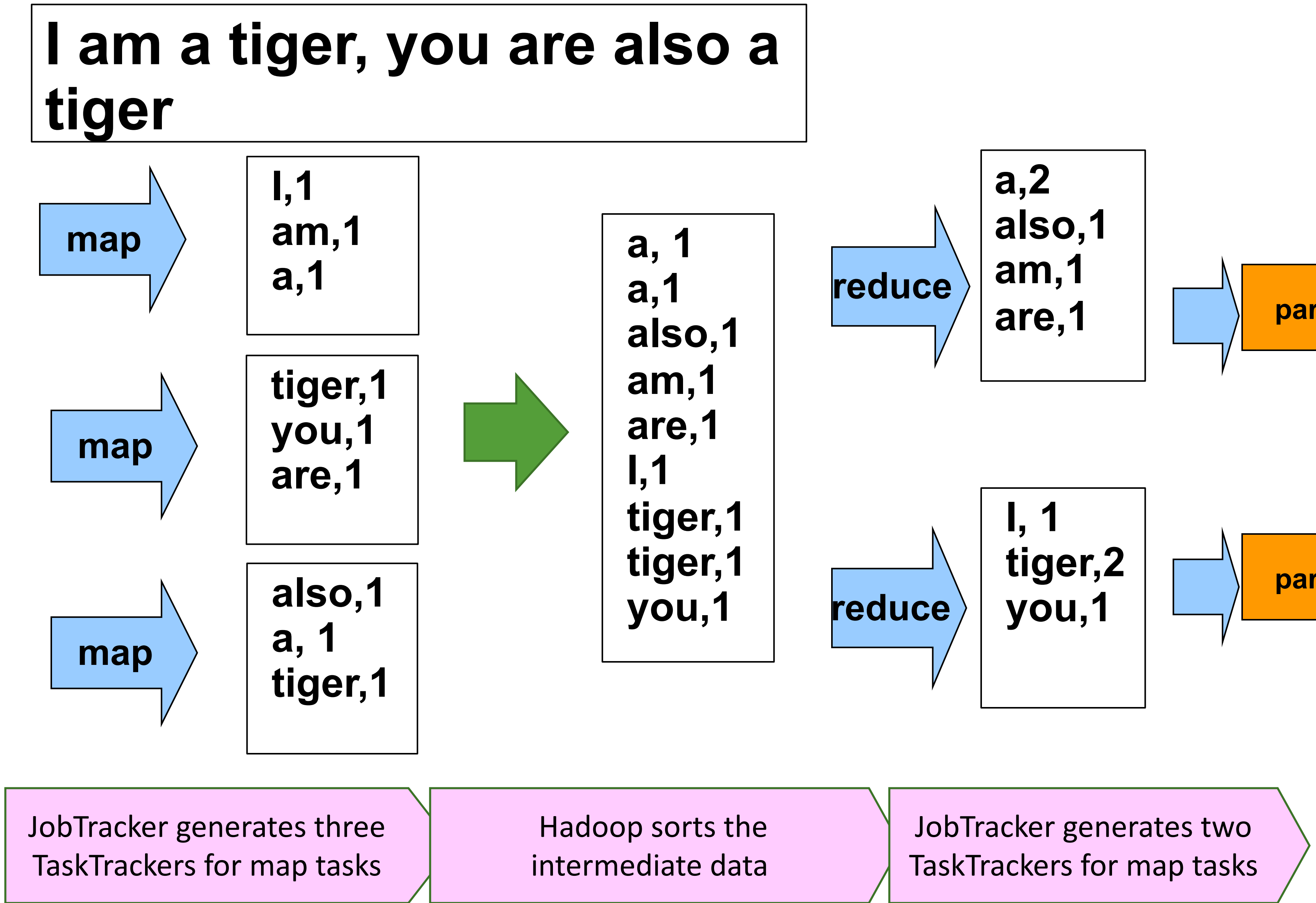
HADOOP-MAPREDUCE WORKFLOW



MAPREDUCE DATAFLOW



EXAMPLE


I am a tiger, you are also a tiger
map
map
map
I,1
am,1
a,1
tiger,1
you,1
are,1
also,1
a, 1
tiger,1
a, 1
a,1
also,1
am,1
are,1
I,1
tiger,1
tiger,1
you,1
reduce
reduce
a,2
also,1
am,1
are,1
I, 1
tiger,2
you,1
part0
part1
JobTracker generates three TaskTrackers for map tasks
Hadoop sorts the intermediate data
JobTracker generates two TaskTrackers for map tasks

INPUT AND OUTPUT FORMATS

- A Map/Reduce may specify how it’s input is to be read by specifying an *InputFormat* to be used
- A Map/Reduce may specify how it’s output is to be written by specifying an *OutputFormat* to be used
- These default to *TextInputFormat* and *TextOutputFormat*, which process line-based text data
- Another common choice is *SequenceFileInputFormat* and *SequenceFileOutputFormat* for binary data
- These are file-based, but they are not required to be

HOW MANY MAPS AND REDUCES

- Maps
  - Usually as many as the number of HDFS blocks being processed, this is the default
  - Else the number of maps can be specified as a hint
  - The number of maps can also be controlled by specifying the *minimum split size*
  - The actual sizes of the map inputs are computed by:
    - *max(min(block_size,data/#maps), min_split_size)*
- Reduces
  - Unless the amount of data being processed is small
    - *0.95*num_nodes*mapred.tasktracker.tasks.maximum*

CS435 Distributed systems

SUMMARY

Dr. Basit Qureshi

PhD FHEA SMIEEE MACM

www.drbasit.org

SUMMARY

- Advantages
  - Scalability, Cost-Effective, Fault Tolerance, Flexibility, High Throughput
  - Open source and widely used in the industry for small and large clusters
- Disadvantages
  - Suitable for batch processing; Not suitable for real-time processing
  - Not designed for transaction processing

SUMMARY

- Alternate eco-systems
  - **Apache Spark** provides in-memory data processing which can significantly speed up data processing tasks compared to Hadoop's disk-based MapReduce
  - **Apache Flink** is designed for stream processing and supports both batch and real-time data processing
  - **Apache Storm** is another real-time computation system that is designed to process unbounded streams of data with low latency.
  - **Google Cloud Dataflow** is a fully managed service for stream and batch data processing
  - **Microsoft Azure Synapse Analytics** combines big data and data warehousing capabilities in Microsoft Azure cloud platform

APACHE STORM™

Apache Flink

Google Dataflow

Azure Synapse Analytics