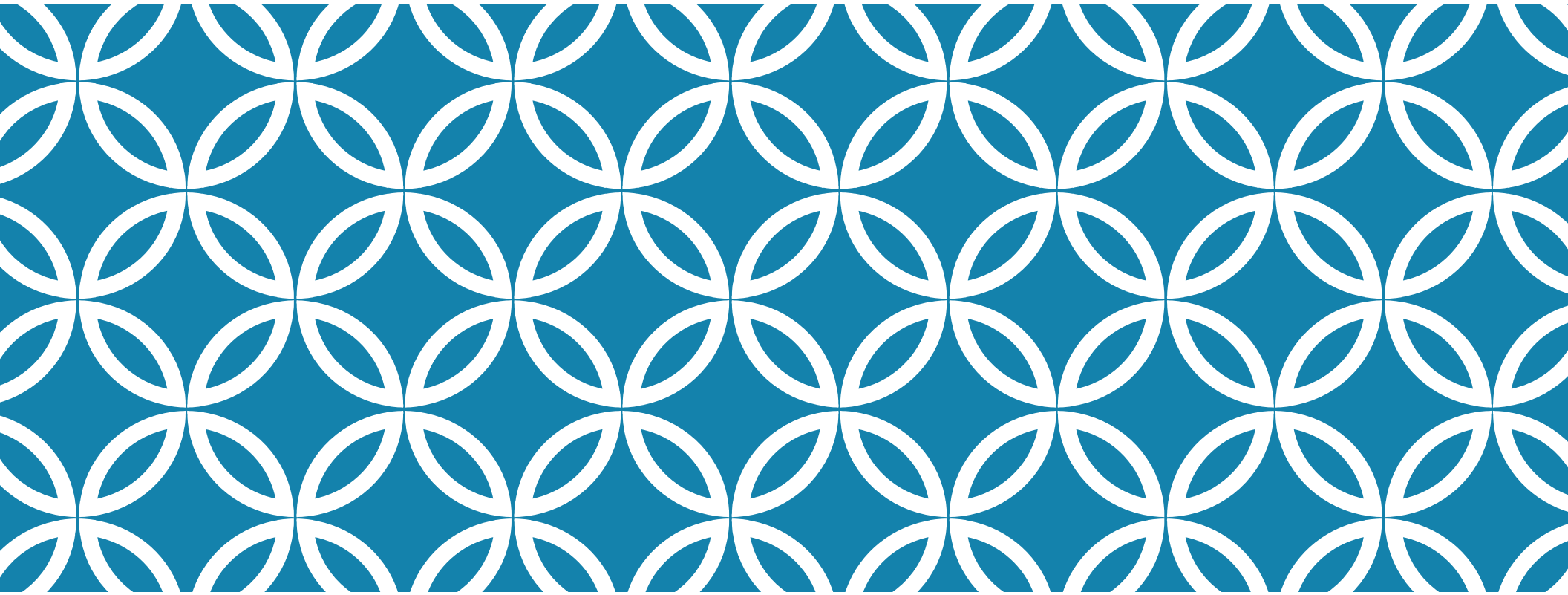
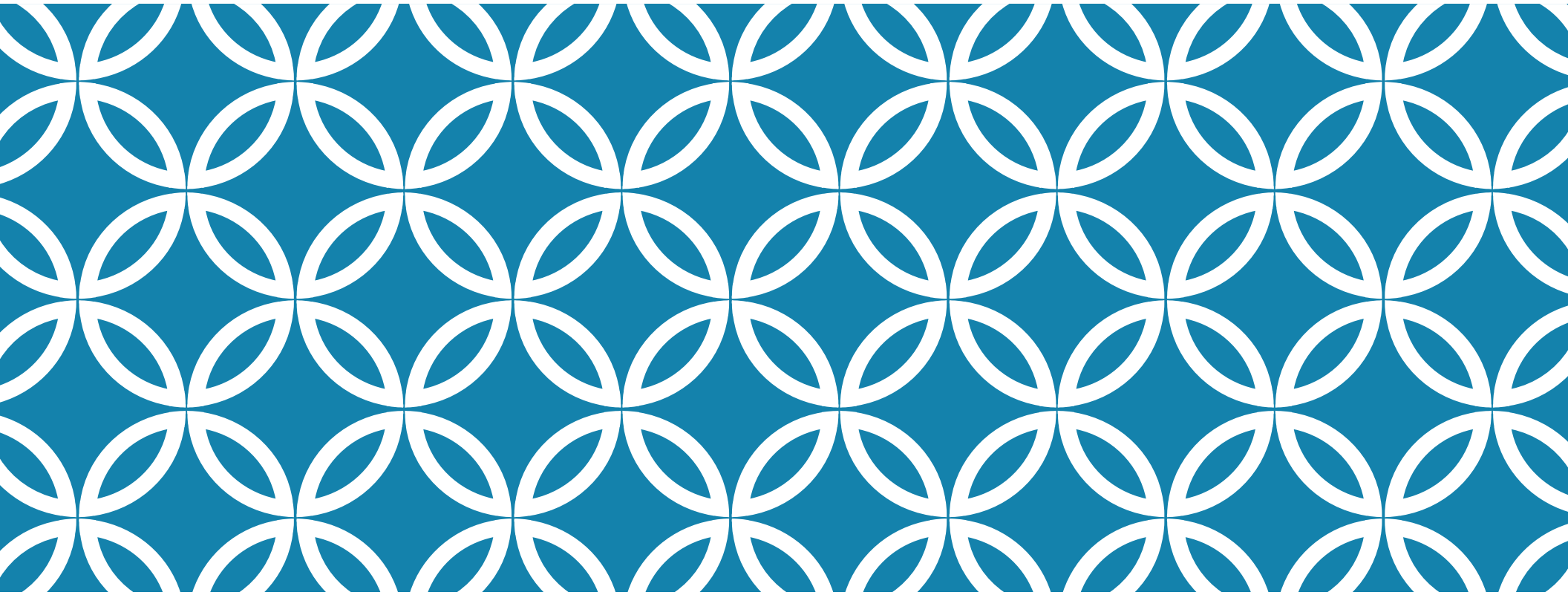


DATA ANALYTICS IN BIG DATA

DATA ANALYTIC'S STACK



GOOGLES'STACK

SEARCH ENGINE PROVIDERS

Mission = Store and process indexes of all the documents in the Internet

Google and YAHOO!

PROBLEM CHARACTERIZATION

Volume

- Giant files

Processing:

- Simple computations

Workload

- Lots of sequential reads
- File changes by appending

VISION

Giant Files + Simple Computations = Distributed System

- Data Distribution
- Parallelism
- Failures
- Performance ...

GOOGLE'S STACK

DATA STORAGE

1. Google File System(GFS)

- Provides a scalable, fault-tolerant and distributed file system across a cluster of commodity servers

Bigtable

- Provides a distributed storage system for managing structured data built over GSF

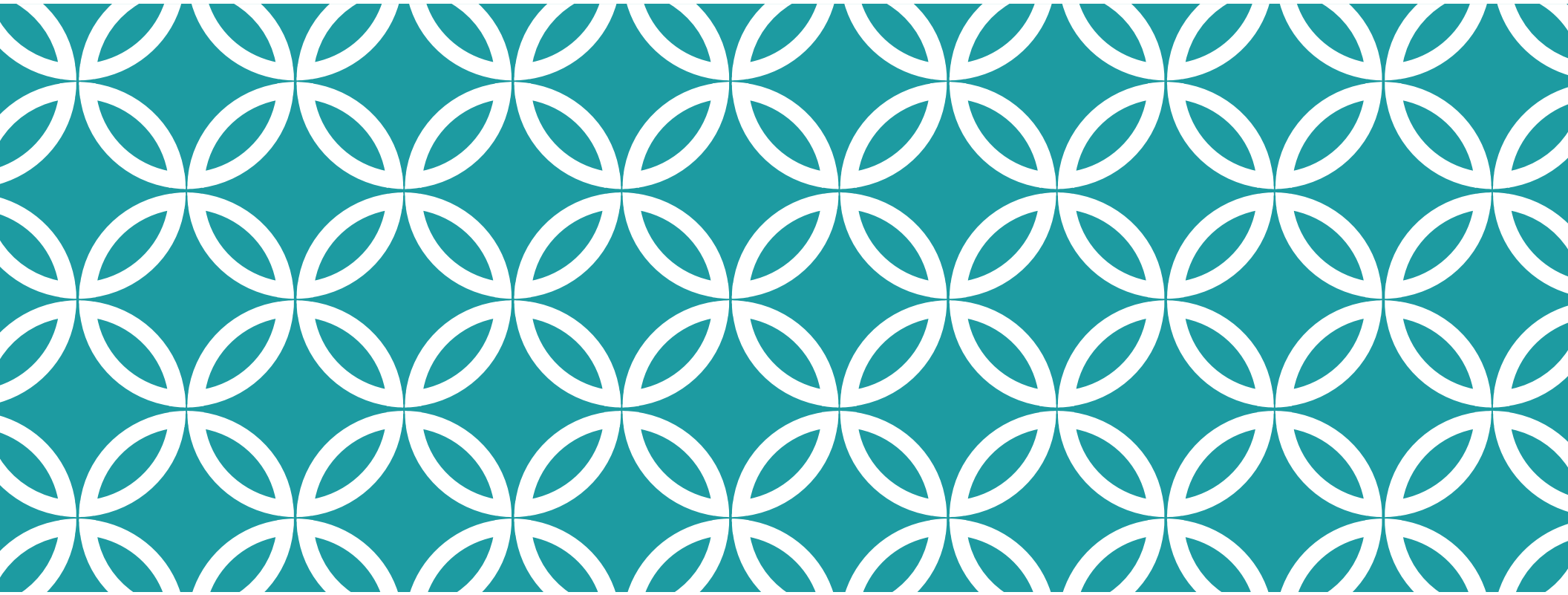
DATA PROCESSING

MapReduce

- Introduces a parallel programming model (based on functional programming) for processing and generating large data sets distributed over Bigtable/GFS

References:

1. The Google File System
2. Bigtable: A Distributed Storage System for Structured Data
3. MapReduce: Simplified Data Processing on Clusters



GOOGLE FILE SYSTEM (GFS)

GOOGLE FILE SYSTEM (GFS)

Component failures are norm, not exception

- Monitoring, error detection, falt tolerance, automatic recover are mandatory

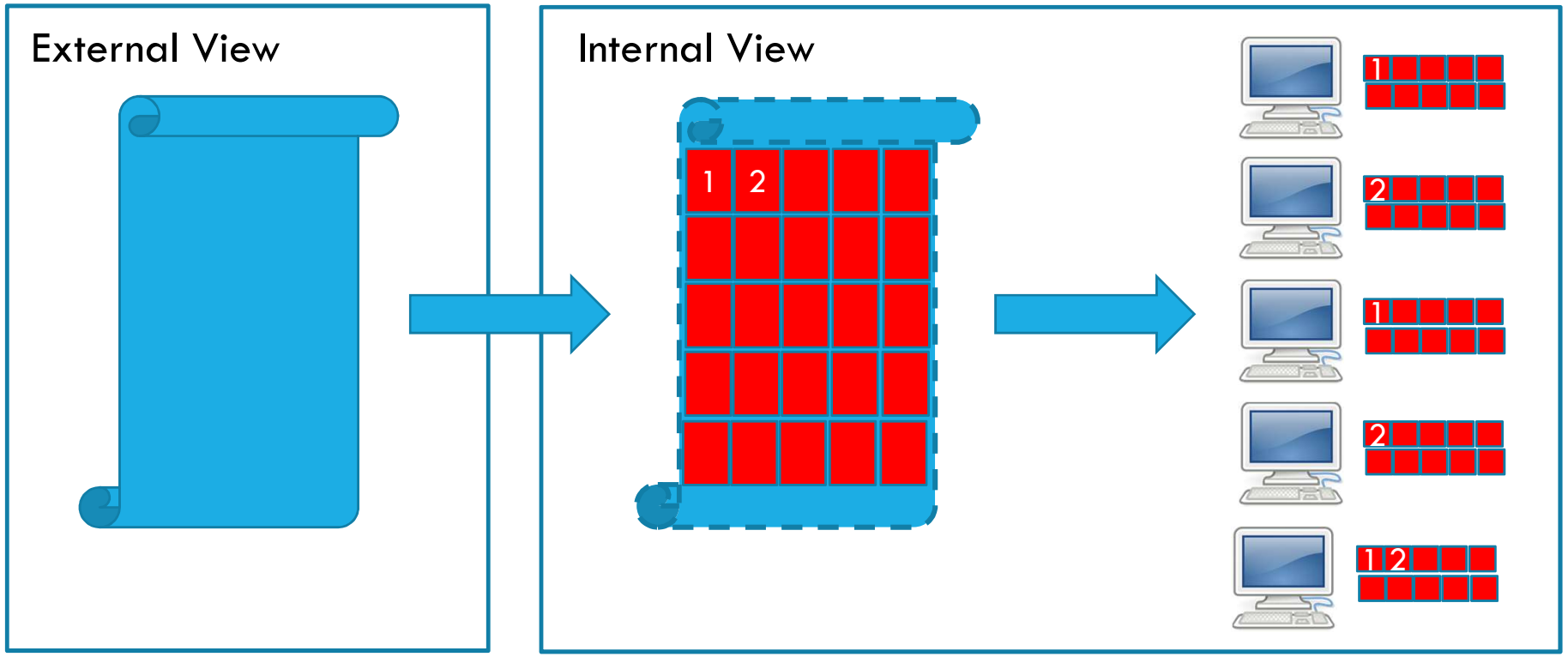
Modest number of usually large files (multi-GB)

- Reconsider block size

Multiple reads, writings are majory appends

- Minimal synchronization overhead

GOOGLE FILE SYSTEM (GFS)



GFS ARCHITECTURE

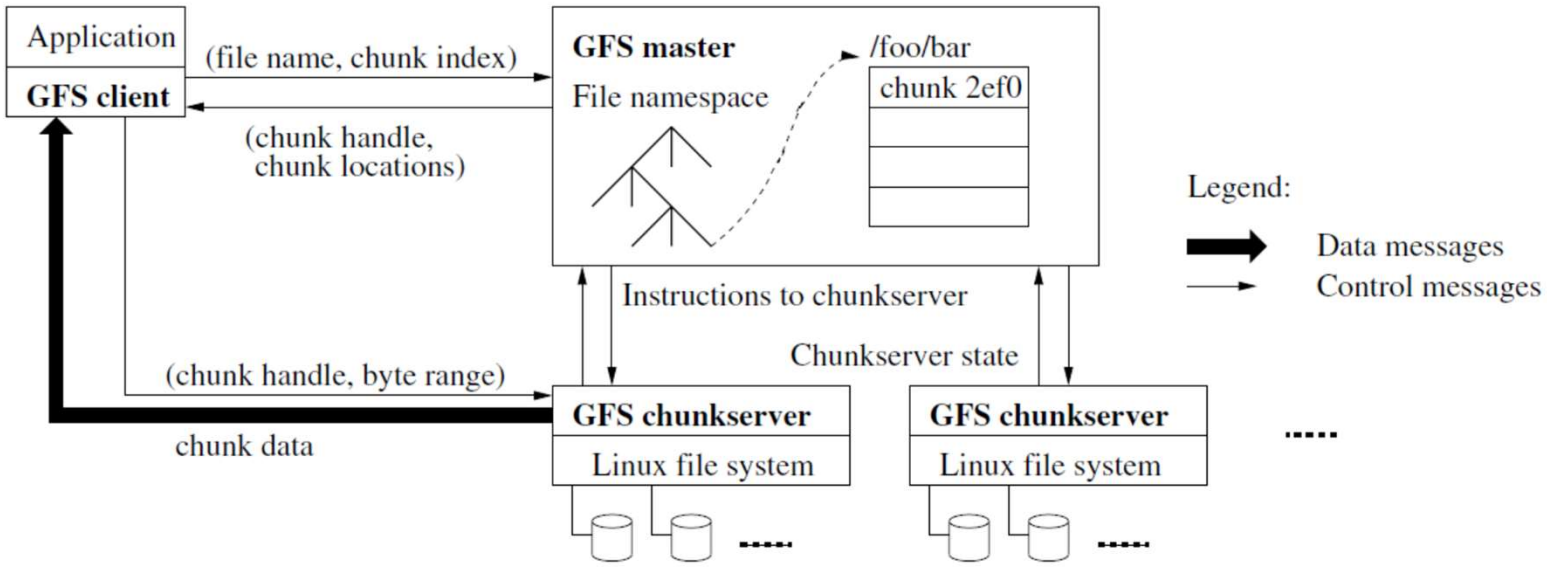
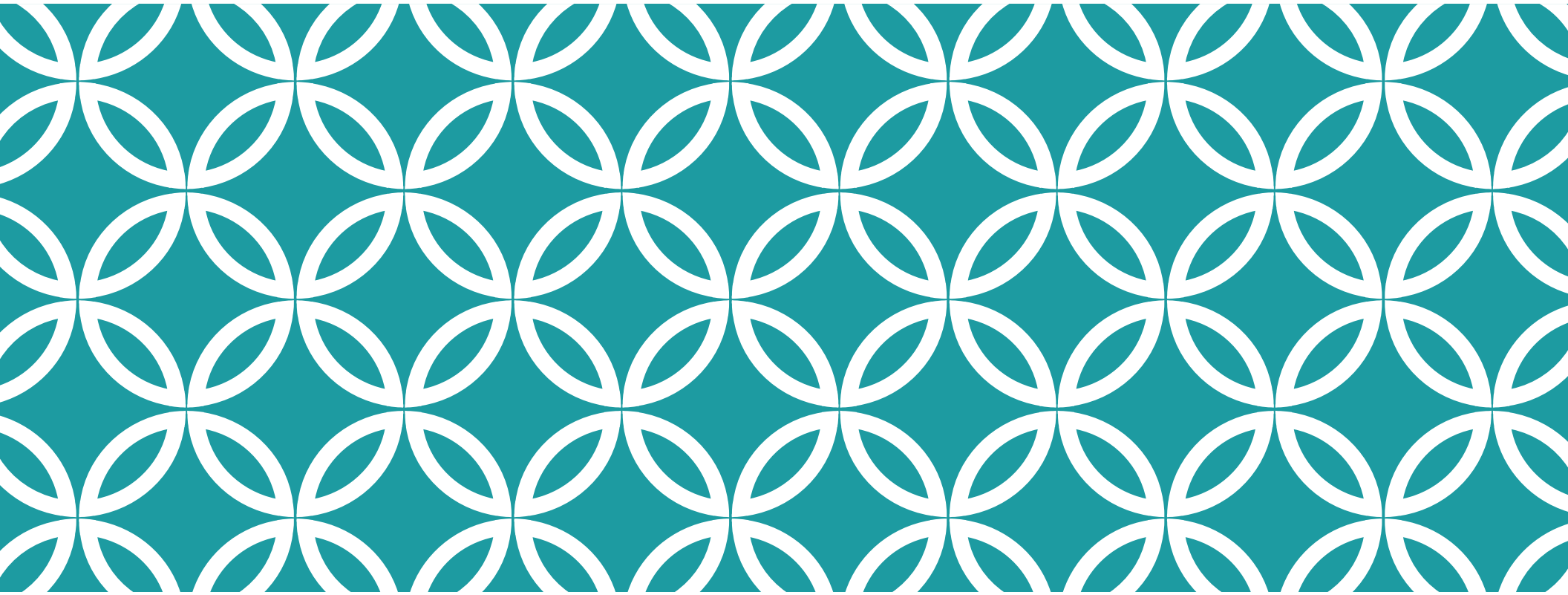


Figure 1: GFS Architecture

https://bit.ly/2qnwqez



BIGTABLE

BIGTABLE

Distributed storage system for managing structured data

Built atop of GFS

Does NOT support a full relational model

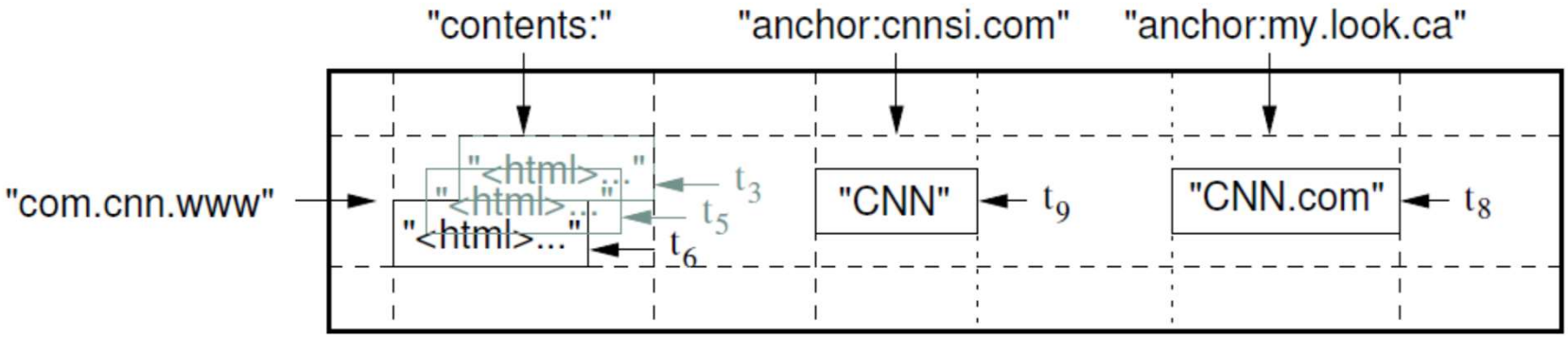
Used by

- Web indexing
- Google Earth
- Google Finance
- Orkut

BIGTABLE'S DATA MODEL

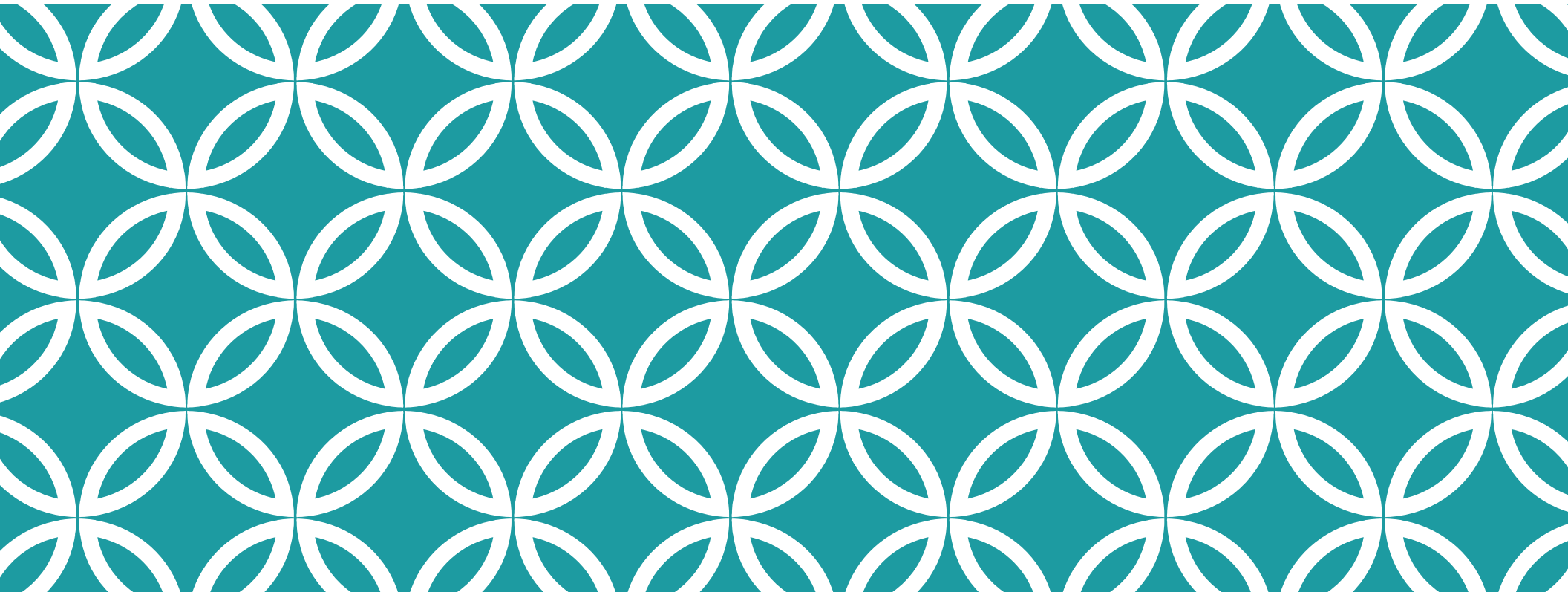
Multi-dimensional sorted map

- (row:string, column:string, time:int64) → string



Ex: A Webtable

https://bit.ly/2r5aKEp



MAPREDUCE

MAPREDUCE: PROBLEM CHARACTERIZATION

Simple computations processing a large amount of data

Distribution can be a solution, but can introduce complexity

- Parallelize operations
- Distributed data
- Failures
- Load balancing

Introduce a new abstraction!

MAPREDUCE

Functional-based **programming model** and **implementation** for parallel processing large data sets

Map

- Receives a input key/value pair
- Produces intermediary key/value pairs

Reduce

- Receives an intermediate key and a set of related values
- Merges all values related to the same intermediary key

MapReduce library

- Groups intermediate values associated with the same key
- Passes them to the Reduce function

USER DEFINED CODE: FREQUENCY OF WORDS

Map Function

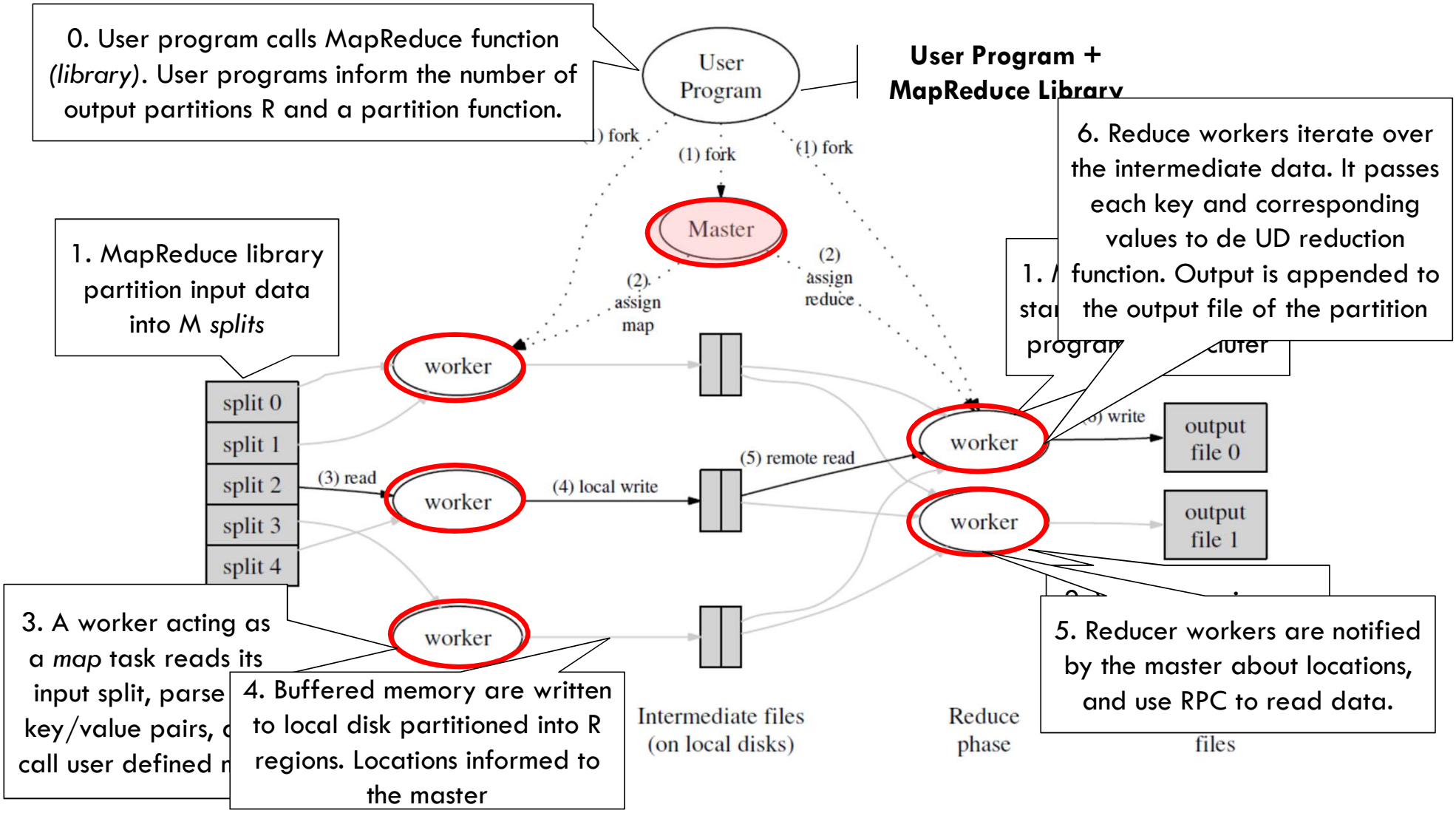
```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
    EmitIntermediate(w, "1");
```

Reduce Function

```
reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
    result += ParseInt(v);
  Emit(AsString(result));
```

```
REGISTER_MAPPER(WordCounter);
```

```
REGISTER_REDUCER(Adder);
```

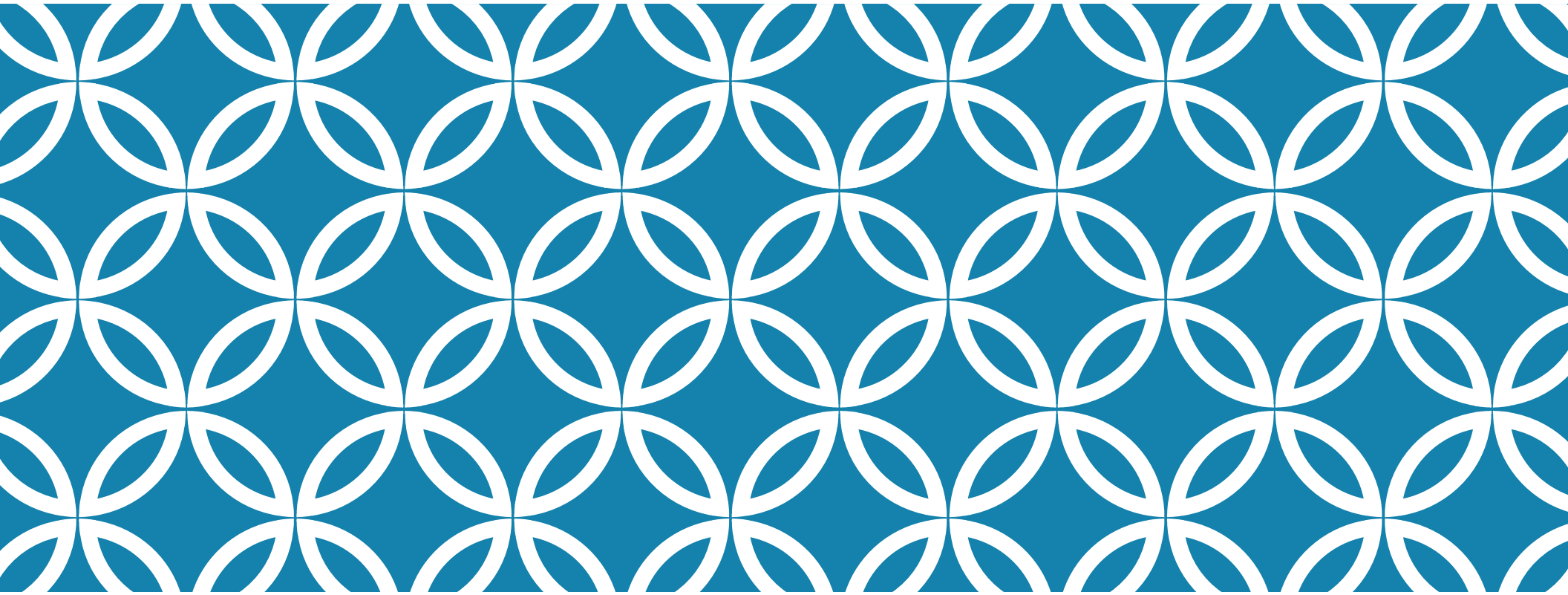

0. User program calls MapReduce function (*library*). User programs inform the number of output partitions R and a partition function.
User Program
User Program + MapReduce Library
(1) fork
(1) fork
(1) fork
Master
(2) assign map
(2) assign reduce
1. MapReduce library partition input data into M *splits*
split 0
split 1
split 2
split 3
split 4
worker
worker
worker
worker
worker
(3) read
(4) local write
(5) remote read
(6) write
output file 0
output file 1
6. Reduce workers iterate over the intermediate data. It passes each key and corresponding values to de UD reduction function. Output is appended to the output file of the partition
3. A worker acting as a *map* task reads its input split, parse key/value pairs, call user defined
4. Buffered memory are written to local disk partitioned into R regions. Locations informed to the master
5. Reducer workers are notified by the master about locations, and use RPC to read data.
Intermediate files (on local disks)
Reduce phase
files

MAPREDUCE ADVANTAGES

Simplifies computation parallelization

Reduces bandwidth moving computation to the data

Fault tolerance implemented based on re-execution



HADOOP'S STACK

WHAT IS HADOOP?

*An open source software platform for **distributed storage** and **distributed processing** of very large data sets on computer clusters built from commodity hardware*

- Hortonworks

BRIEF HISTORY

Developed in 2005 by Doug Cutting and Mike Cafarella as part of the *Nutch search engine*

- Hadoop Distributed File System (HDFS)
- A MapReduce implementation

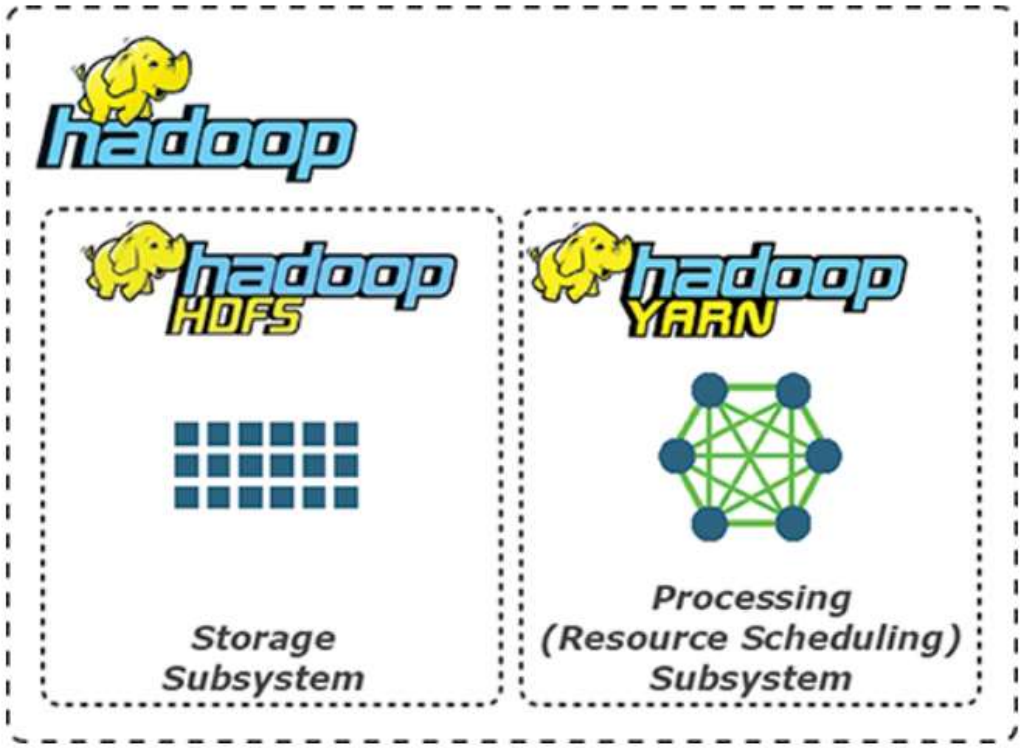
BRIEF HISTORY

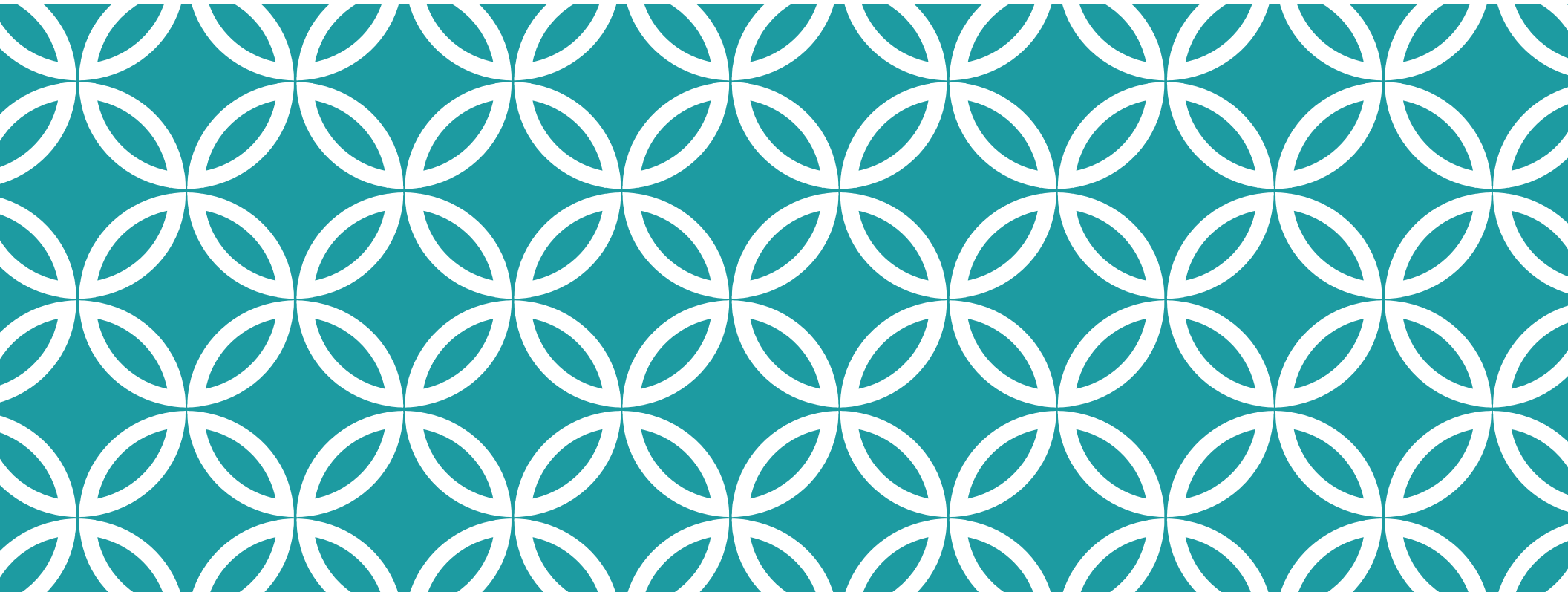
Adopted by YAHOO!

Donated to Apache Software Foundation in 2006

Enriched by lots of new modules that compose the Hadoop Ecosystem

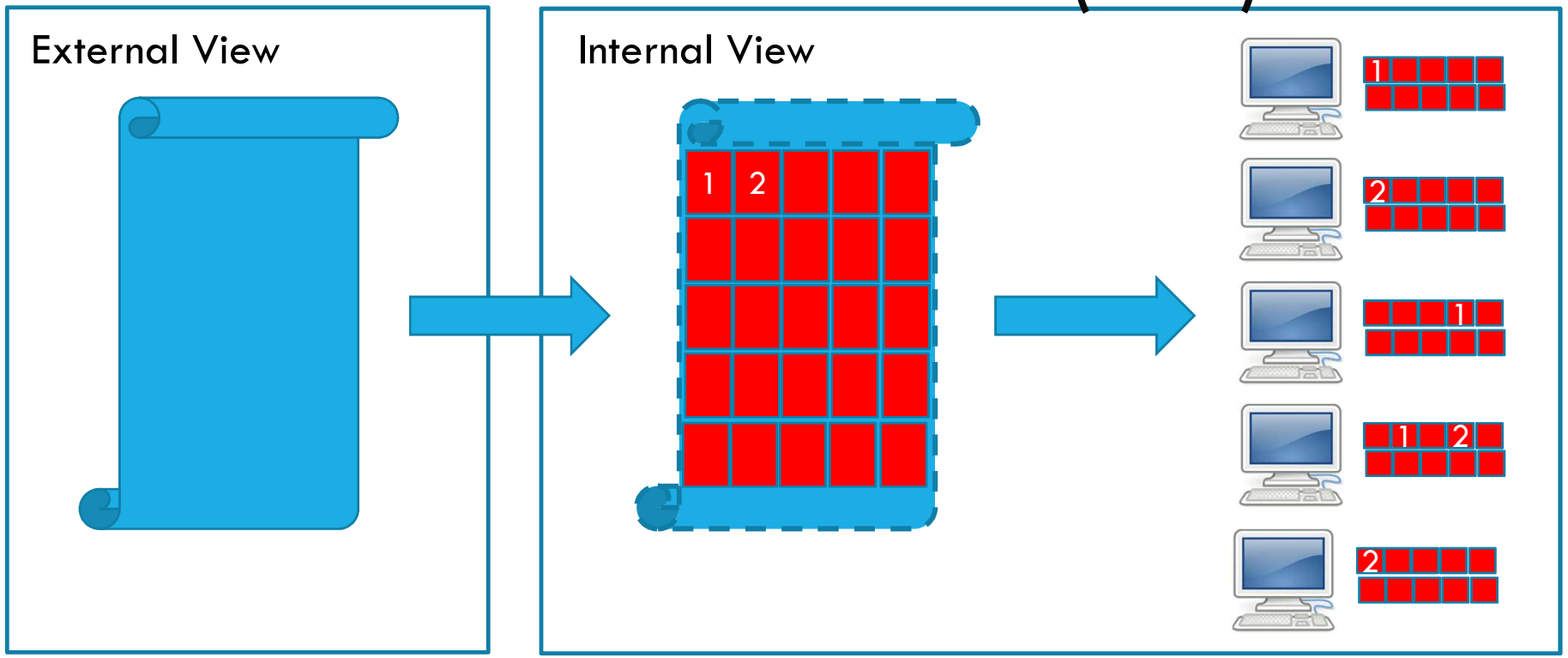
HADOOP CORE ELEMENTS





HDFS

HADOOP DISTRIBUTED FILE SYSTEM (HDFS)



Distributed, scalable, and portable filesystem written in Java for the Hadoop framework

ASSUMPTIONS AND GOALS

Failures

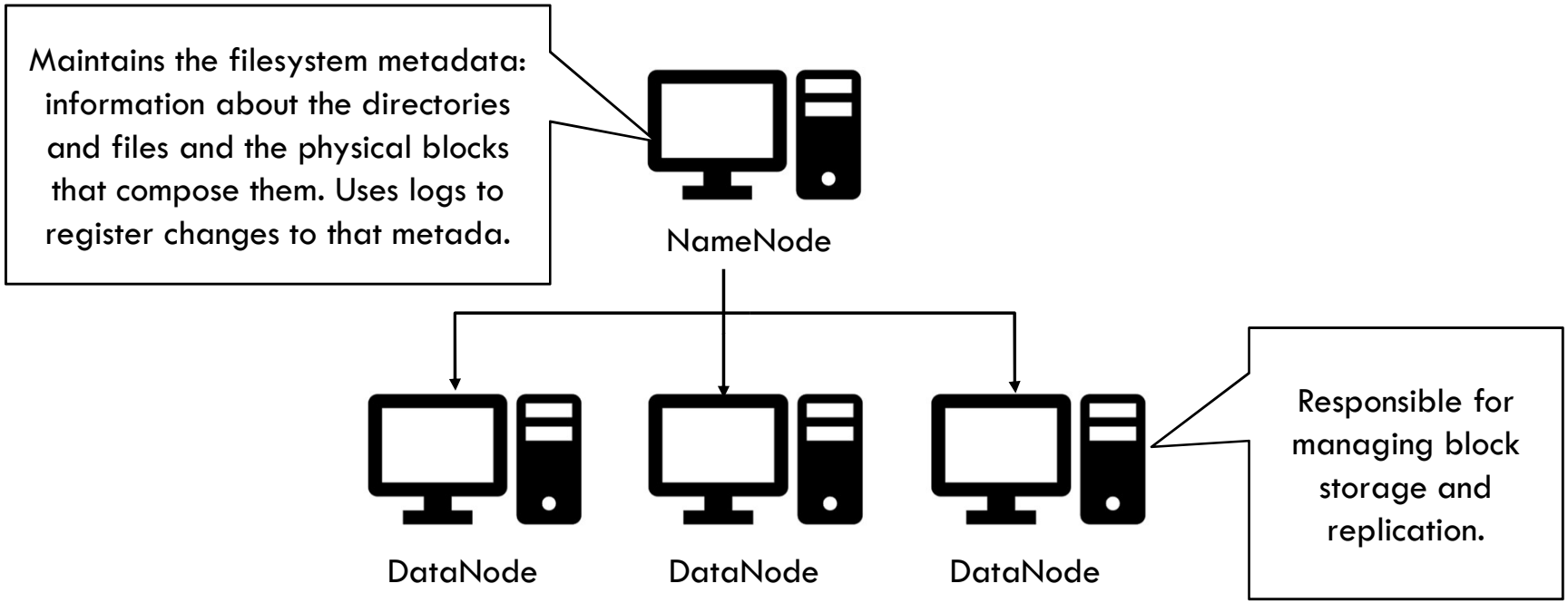
- Highly fault-tolerant and designed to be deployed on low-cost hardware
- Monitoring and automatic recovery

Workloads

- Applications process large data sets
- Batch processing rather than interactive
- *Write once (except for appends and truncates) but they read it a lot more times*

Portability

HDFS CLUSTER (MASTER/SLAVE ARCHITECTURE)



NAMENODE

Manages file system namespace and regulates access to files

Maintains the file system tree and the metadata for all the files and directories present in the system

Executes file system namespace operations like opening, closing, and renaming files and directories.

Determines the mapping of blocks to DataNodes

DATANODES

Manage storage attached to the nodes that they run on

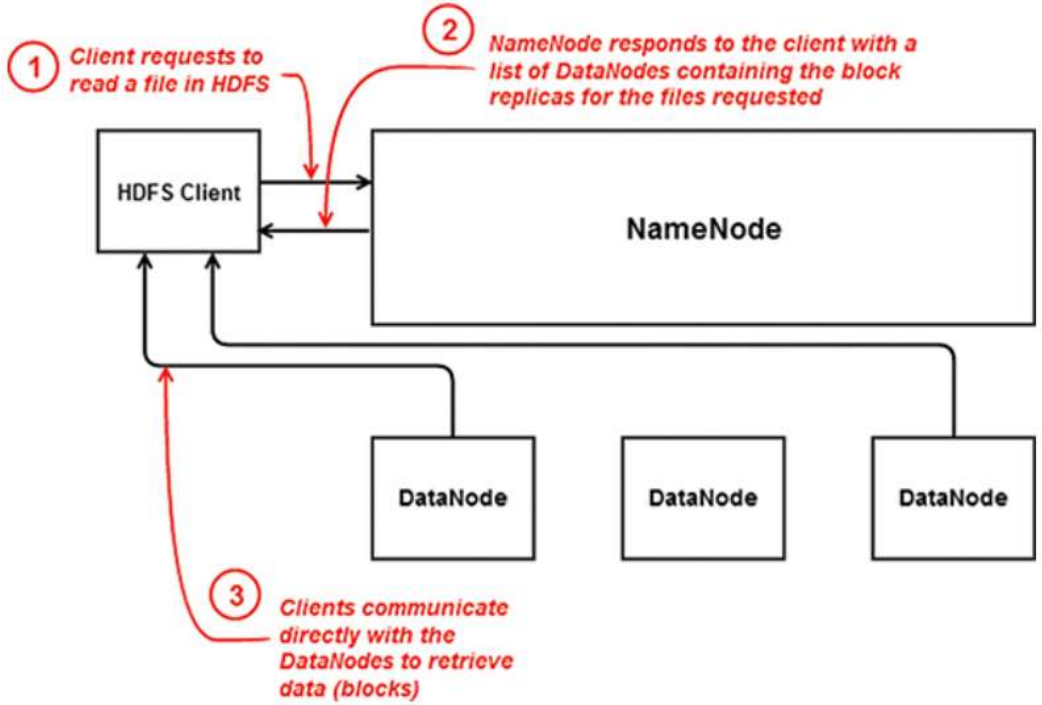
Serve read and write requests from the file system's clients

Perform block creation, deletion, and replication upon instruction from the NameNode

Store each block of HDFS data in a separate file in its local file system

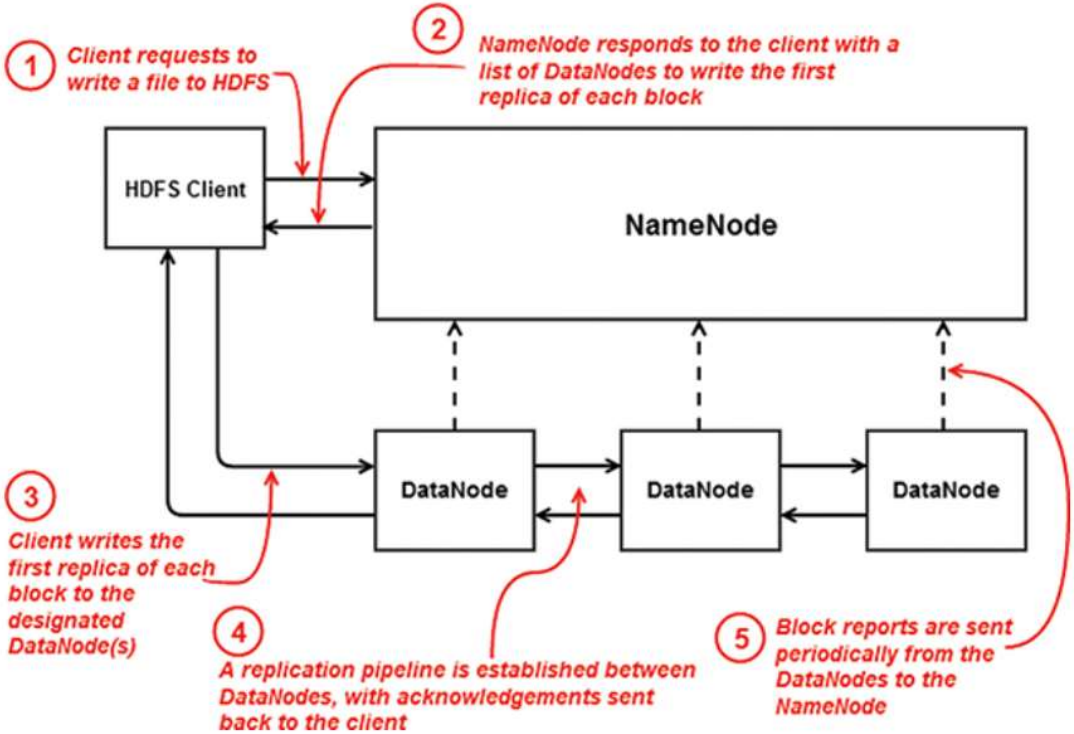
On starts up, it scans its local file system, generates a list of HDFS data blocks that correspond to each of these local files, and sends this report to the NameNode. The report is called the *Blockreport*.

READ OPERATION



Source: Data Analytics with Spark Using Python

WRITE OPERATION



Source: Data Analytics with Spark Using Python

USING HDFS

Natively, HDFS provides a FileSystem Java API for applications to use

A C language wrapper for this Java API and REST API is also available

An HTTP browser and can also be used to browse the files of an HDFS instance.

By using NFS gateway, HDFS can be mounted as part of the client's local file system.

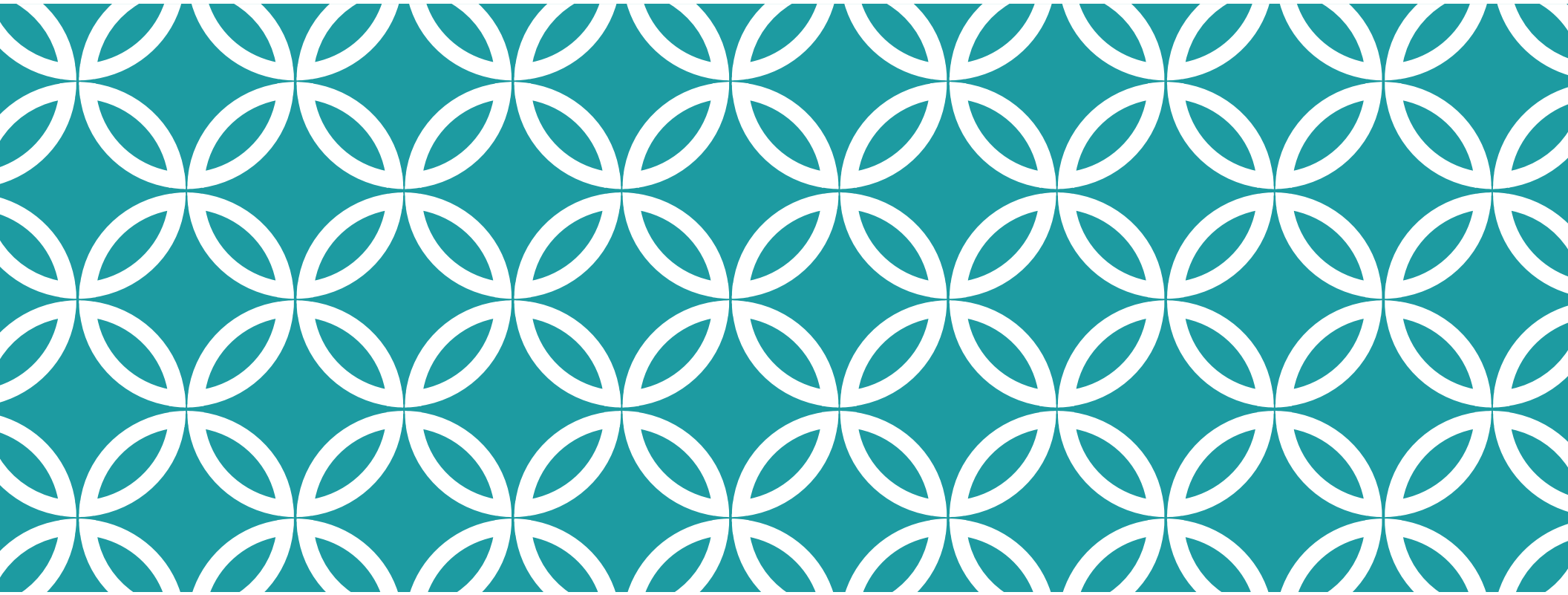
A commandline interface called FS shell that lets a user interact with the data in HDFS.

The DFSAdmin command set is used for administering an HDFS cluster.

REFERENCES

Book: Data Analytics with Spark Using Python

https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html



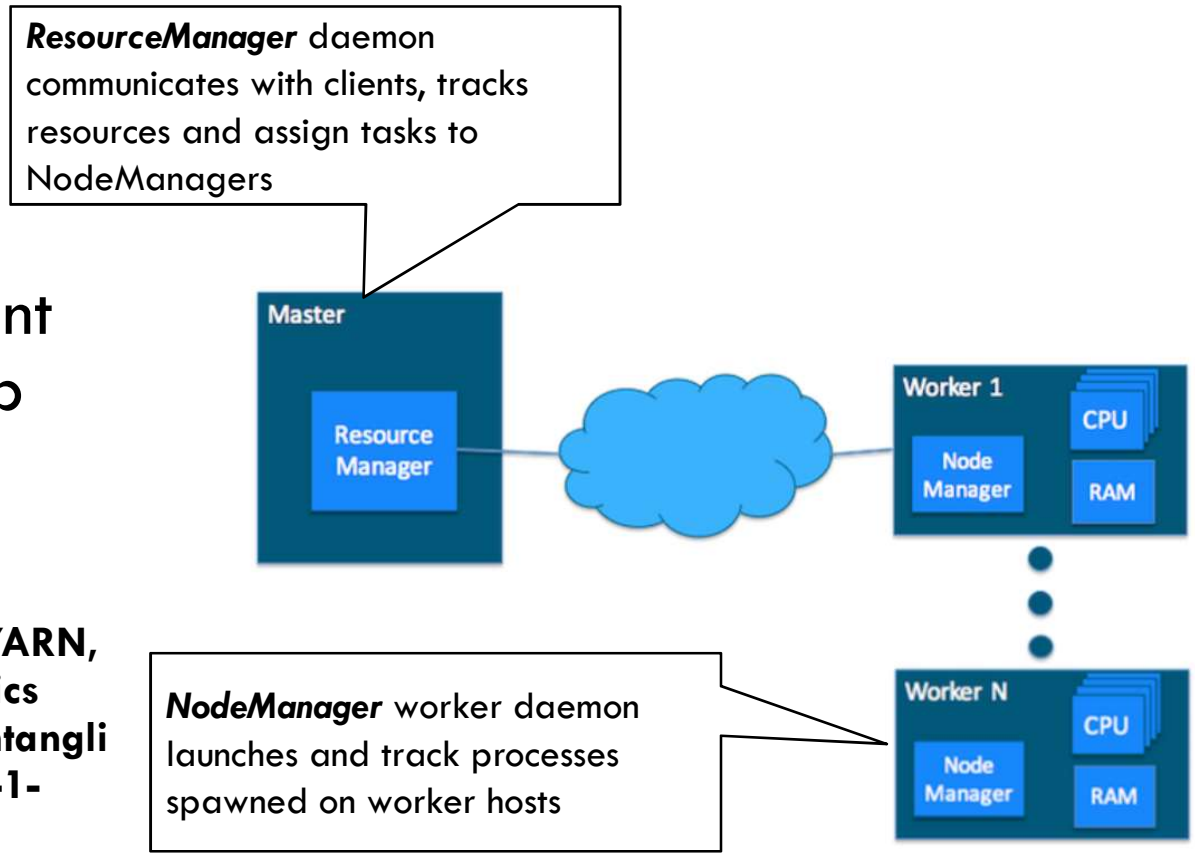
YARN

YARN

Resource management layer for the Hadoop ecosystem

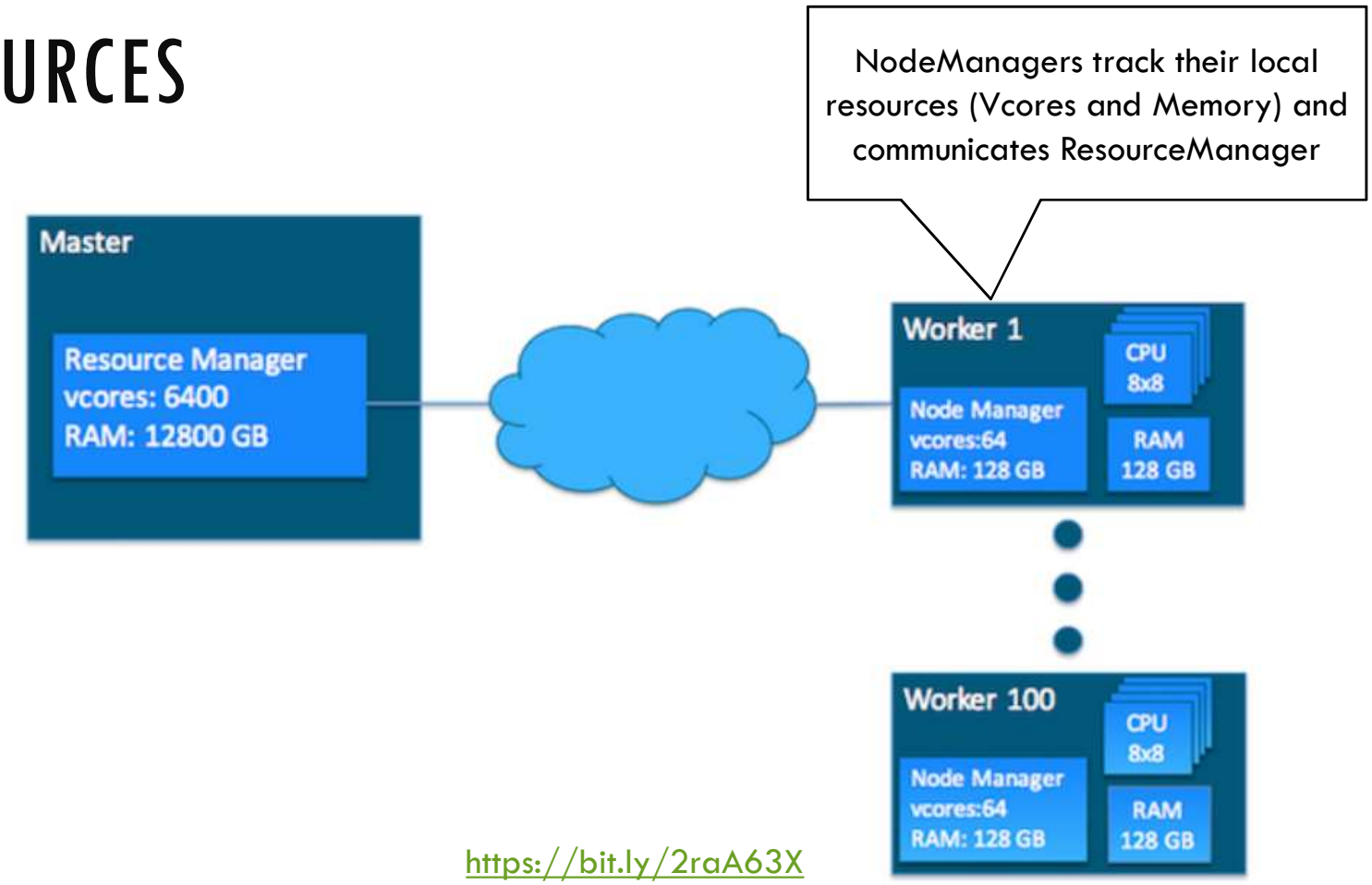
Based on:

- **Untangling Apache Hadoop YARN, Part 1: Cluster and YARN Basics (https://blog.cloudera.com/untangling-apache-hadoop-yarn-part-1-cluster-and-yarn-basics/)**



https://bit.ly/2raA63X

RESOURCES



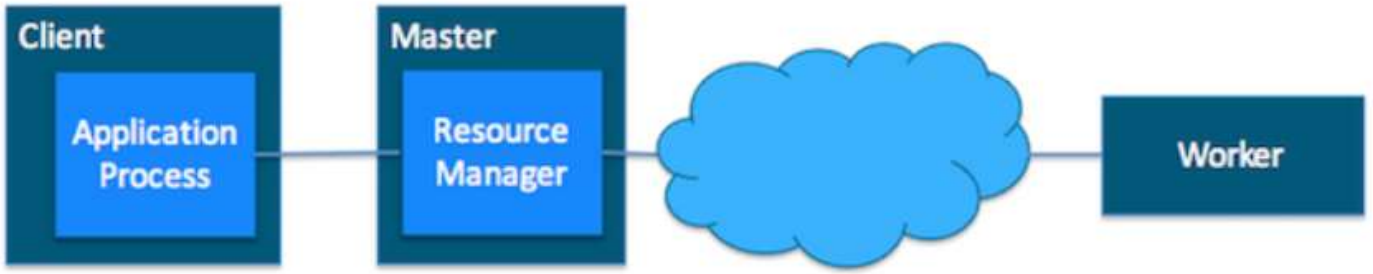
https://bit.ly/2raA63X

CONTAINERS

https://bit.ly/2raA63X

DYNAMICS

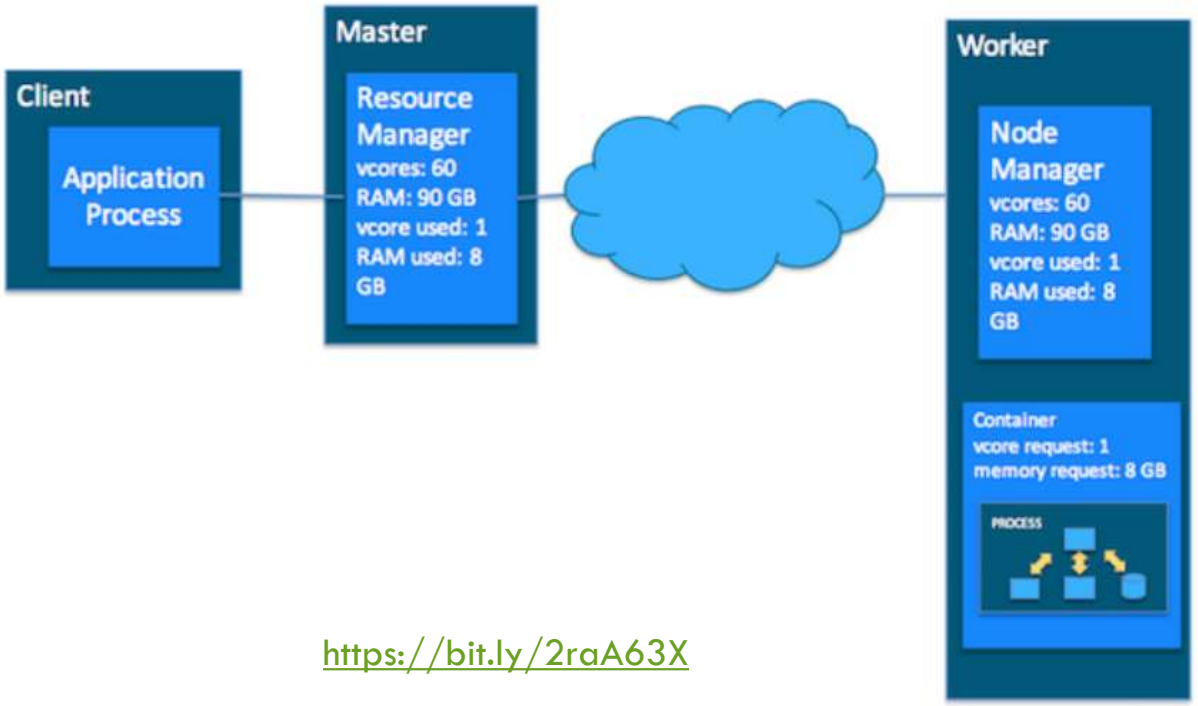
1. ***Application client*** asks the resource manager for a container for the ***Application Master***



https://bit.ly/2raA63X

DYNAMICS

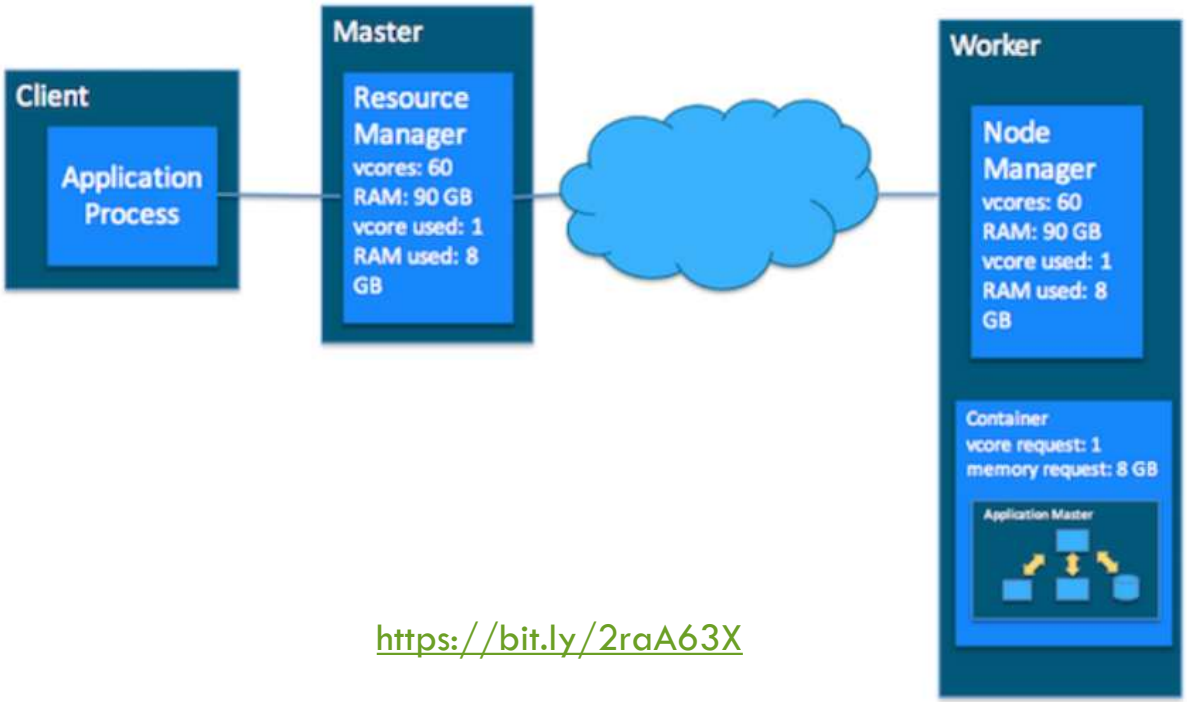
2. *ResourceManager* allocates the container on a *NodeManager*



https://bit.ly/2raA63X

DYNAMICS

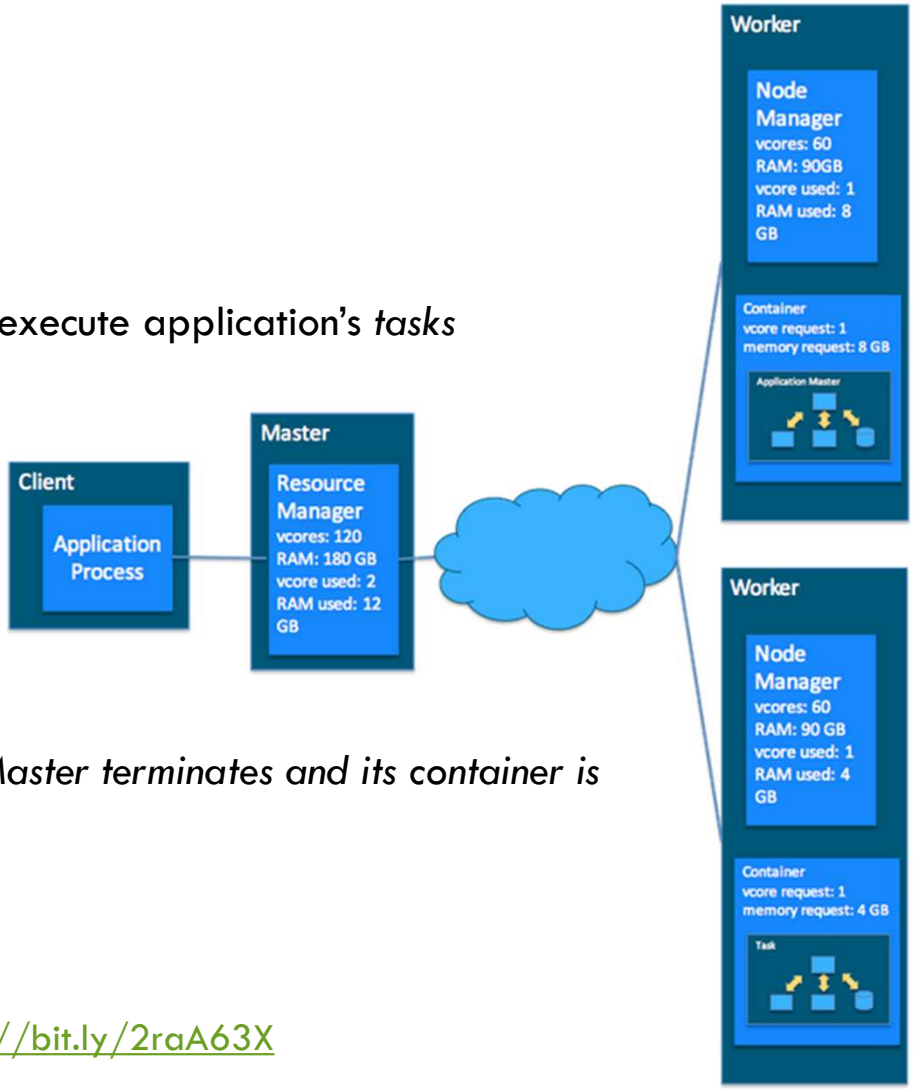
3. *ApplicationMaster* starts running on the container



https://bit.ly/2raA63X

DYNAMICS

4. *Application Master* requests containers to execute application's *tasks*

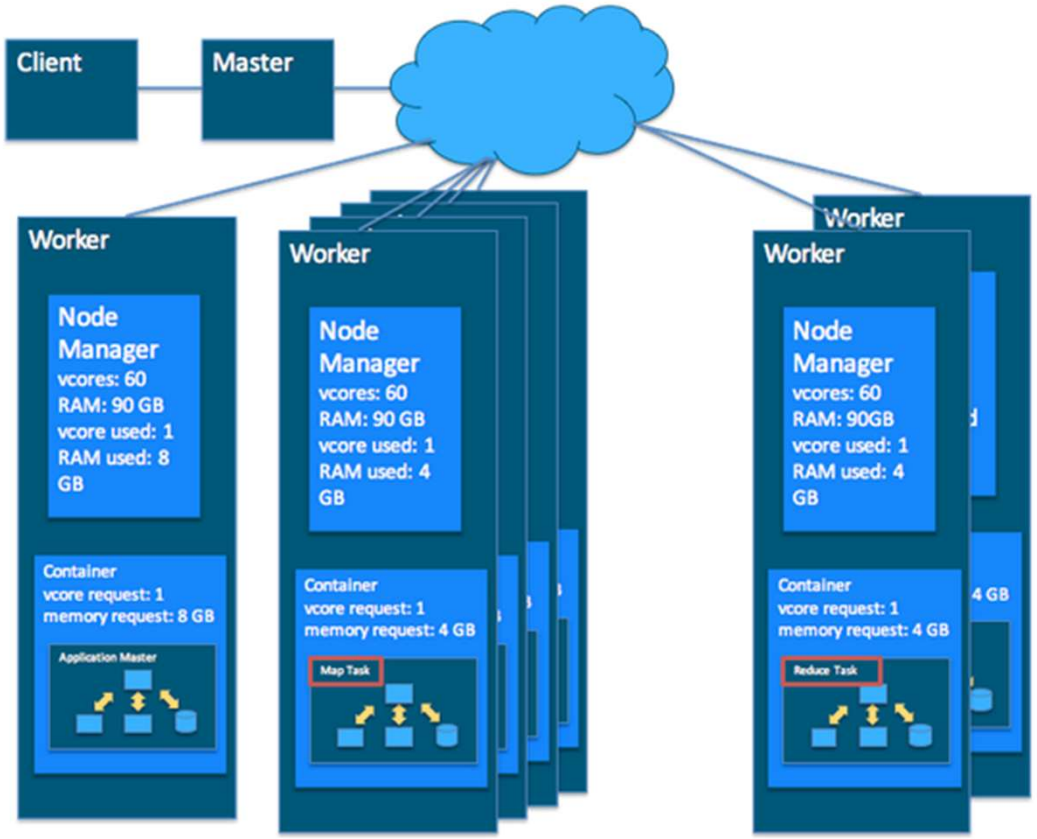


5. Once *tasks* are finished, the *Application Master terminates and its container is deallocated*

6. *Application client terminates*

https://bit.ly/2raA63X

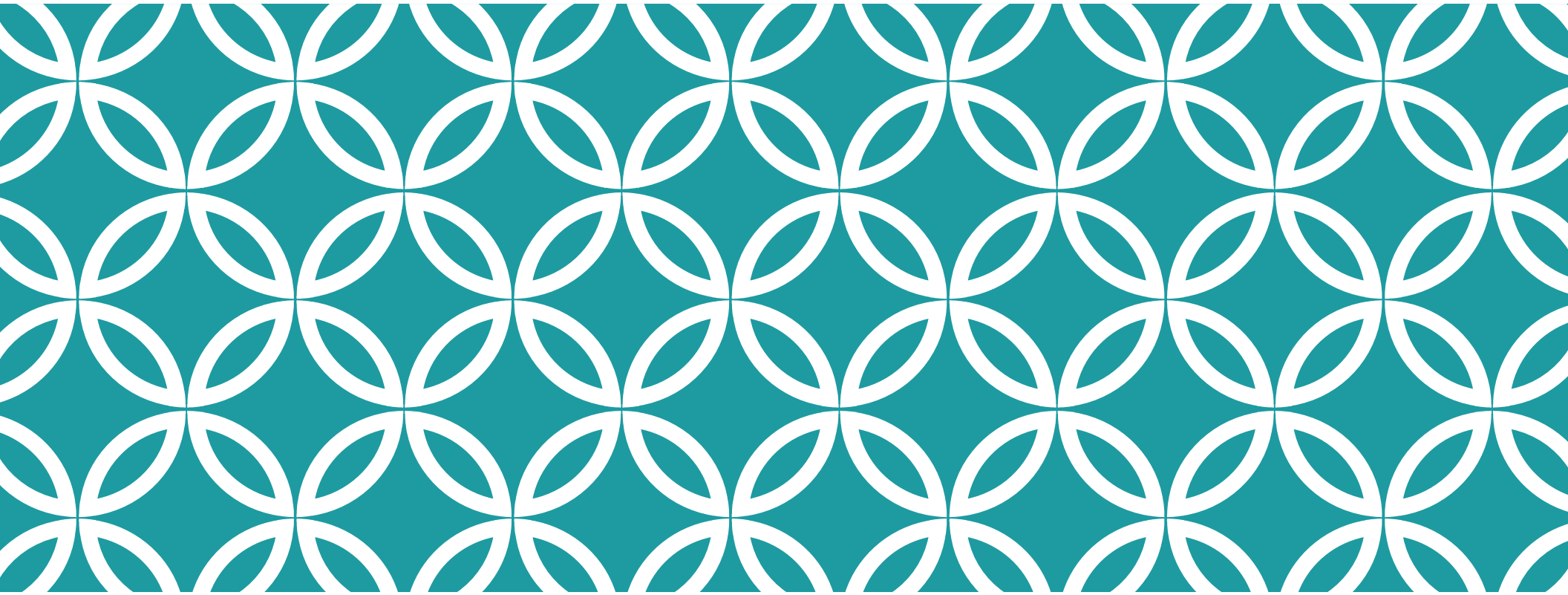
MAPREDUCE AND YARN



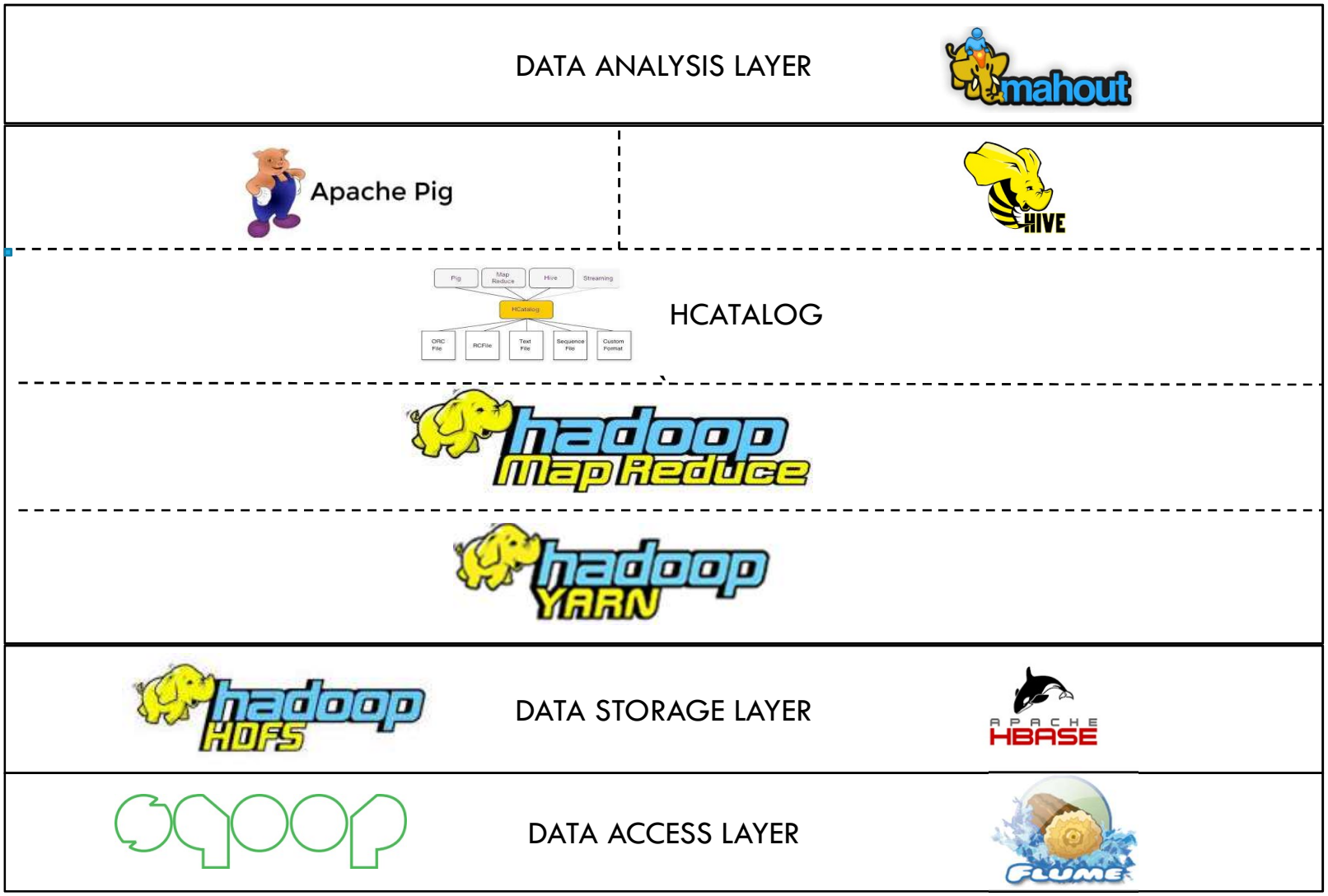
https://bit.ly/2raA63X

REFERENCES

https://blog.cloudera.com/untangling-apache-hadoop-yarn-part-1-cluster-and-yarn-basics/



HADOOP ECOSYSTEM



Data Processing Layer

HBASE

Distributed non relational database built on top of HDFS

Like BitTable, it is based on column-oriented key/value data model

Enables fast record lookups for large tables

Accessed through shell commands, client Java APIs, Rest

APACHE PIG

Implements a scripting language called Pig Latin that is used to specify high level workflows

A compiler translates Pig Latin to MapReduce

Allows iterations with external programs

Has its own data model (Map Data) built on top of HDFS

Originally developed by YAHOO!

APACHE HIVE

A data warehouse system built to enable manipulation of data residing in distributed storage using a SQL like language (HiveQL)

Converts queries into MapReduce jobs

Structure can be projected onto data already in storage

A command line tool and JDBC driver are provided to connect users to Hive.

Originally developed by Facebook

APACHE SQOOP

"Tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases."

Import process

- Reads data from a database table and generates files in HDFS (e.g. delimited text files or Avro)

Export process

- Reads HDFS files, parse then into records, and insert as new rows in a target table

APACHE FLUME

Designed to collect, aggregate and transfer data from external machines to HDFS

Flexible architecture based on streaming data flows

Provides a query processing engine that allows incoming data transformation

“Flume, users can stream data from various and high volume sources (like Avro RPC source and syslog) into sinks (such as HDFS and HBase) for real-time analysis”

HCATALOG

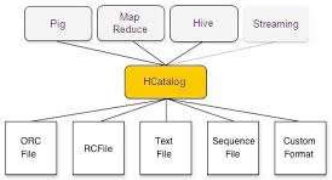


Table and storage management service for Hadoop

Enables interoperability across data processing tools (e.g. Pig, Hive, and MapReduce)

APACHE MAHOUT

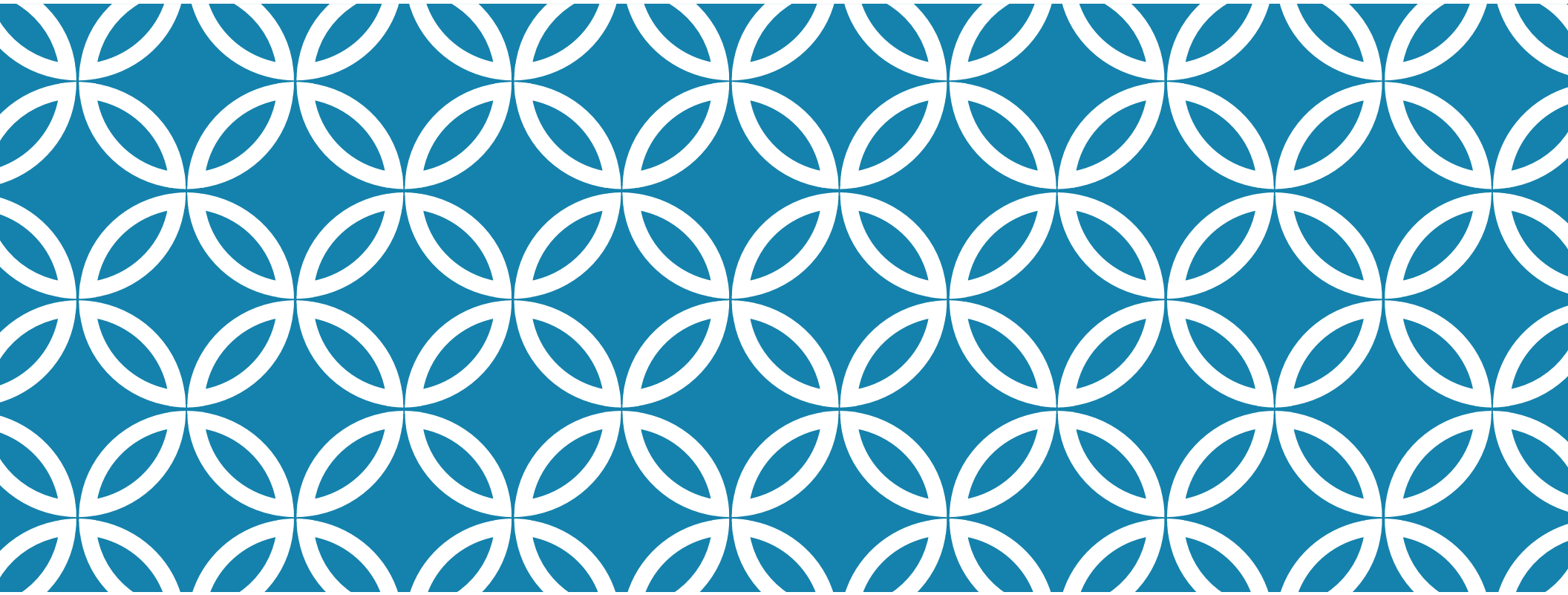
Machine learning software library

When executed on top of Hadoop, translates machine learning tasks into MapReduce jobs

REFERENCES

Big Data technologies: A survey

- Ahmed Oussous, Fatima Zahra Benjelloun, Ayoub Ait Lahcen, Samir Belfkih



QUESTIONS???