University of CINCINNATI

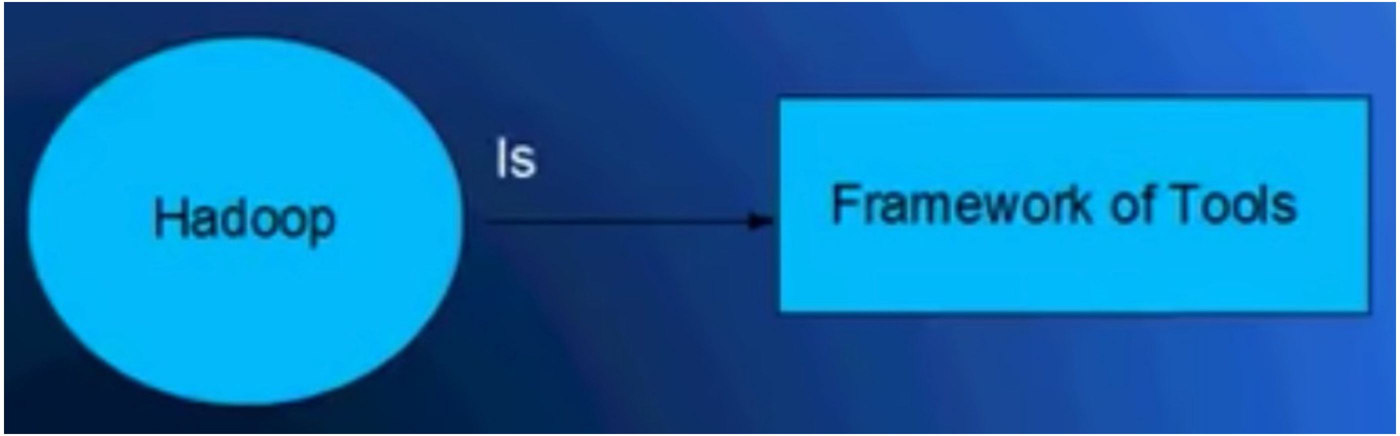
APACHE hadoop

Distributed Computing With Hadoop

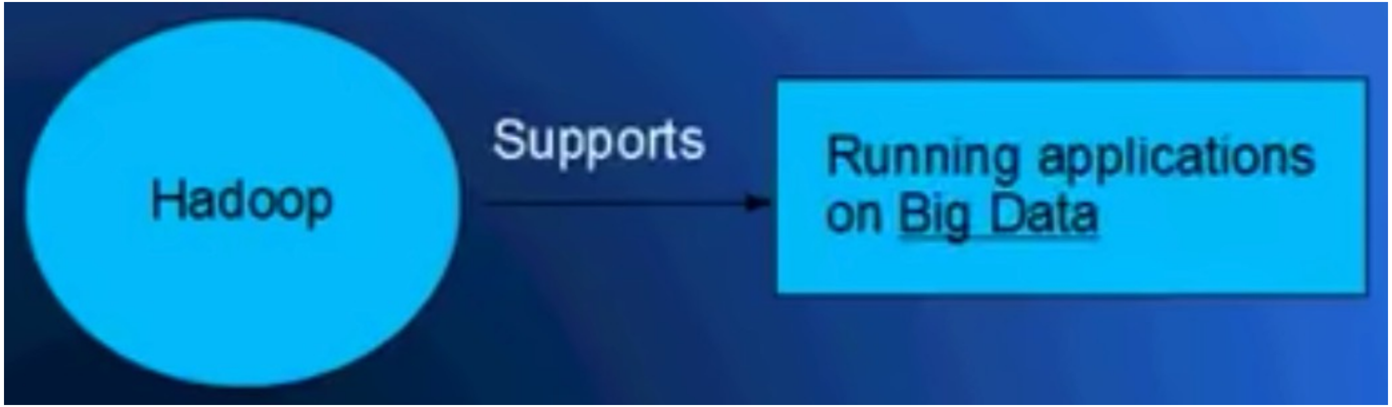
(Introduction)

Rashmi Kansakar

What is Hadoop?



Objective

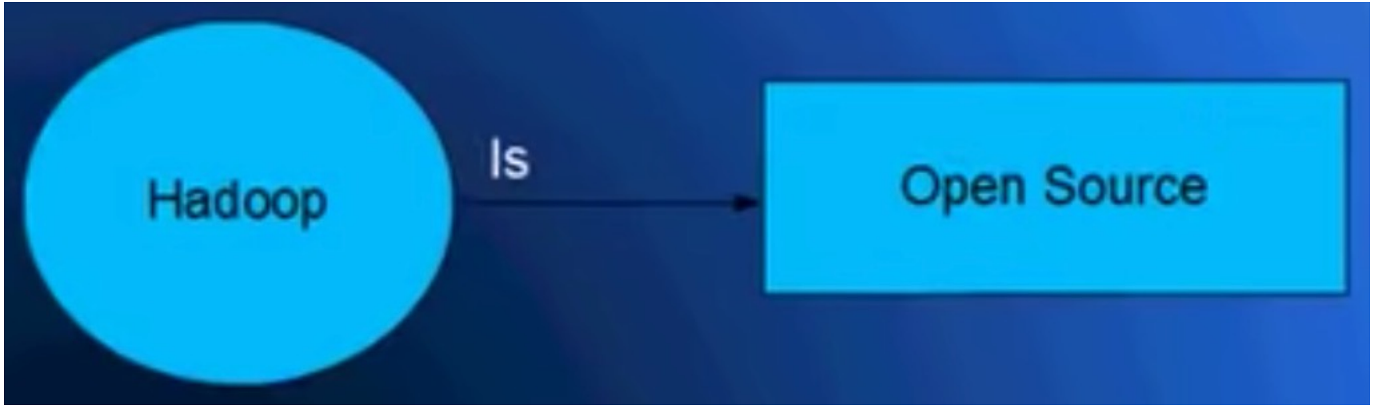


Hadoop

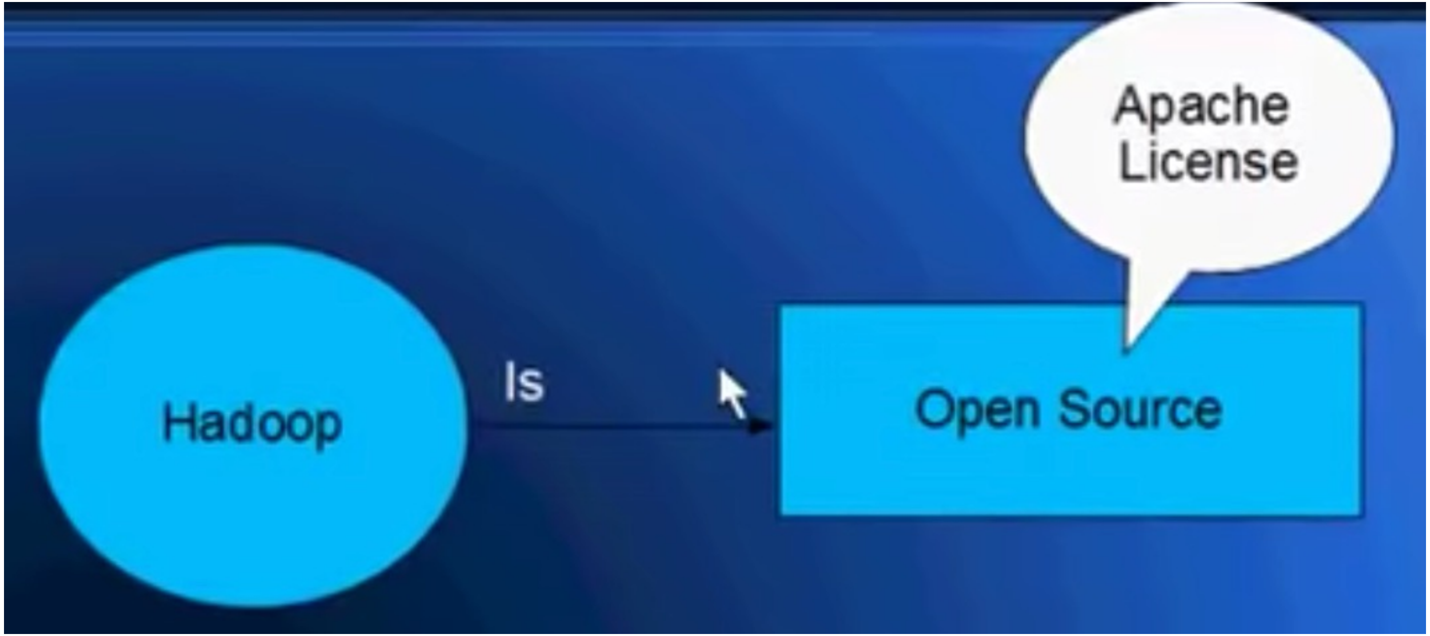
Supports

Running applications on Big Data

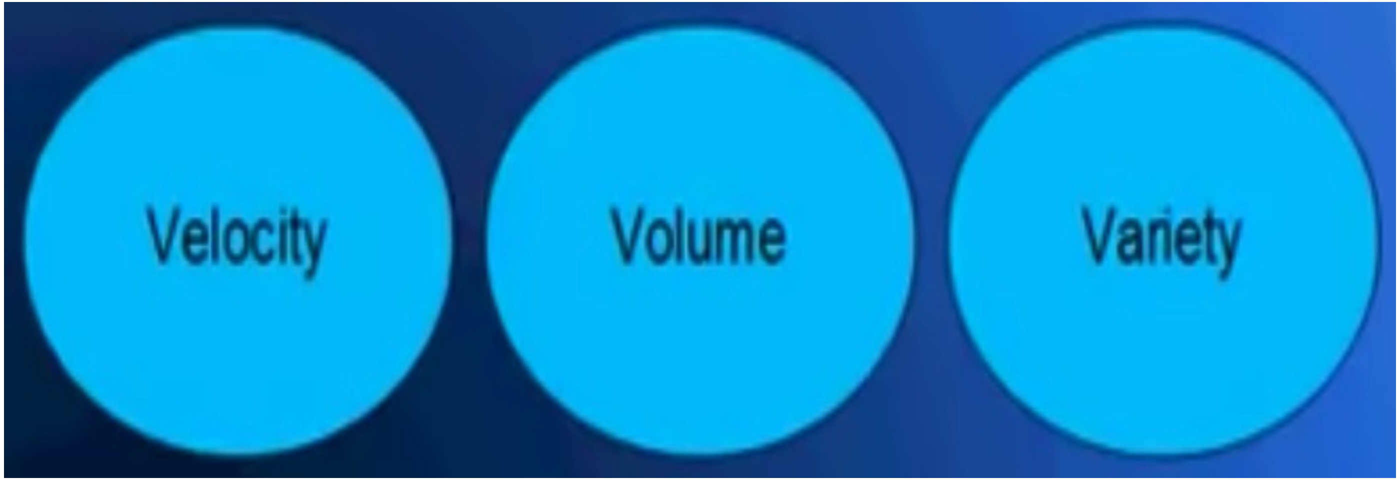
Open Source



Apache



Big Data Challenge Points



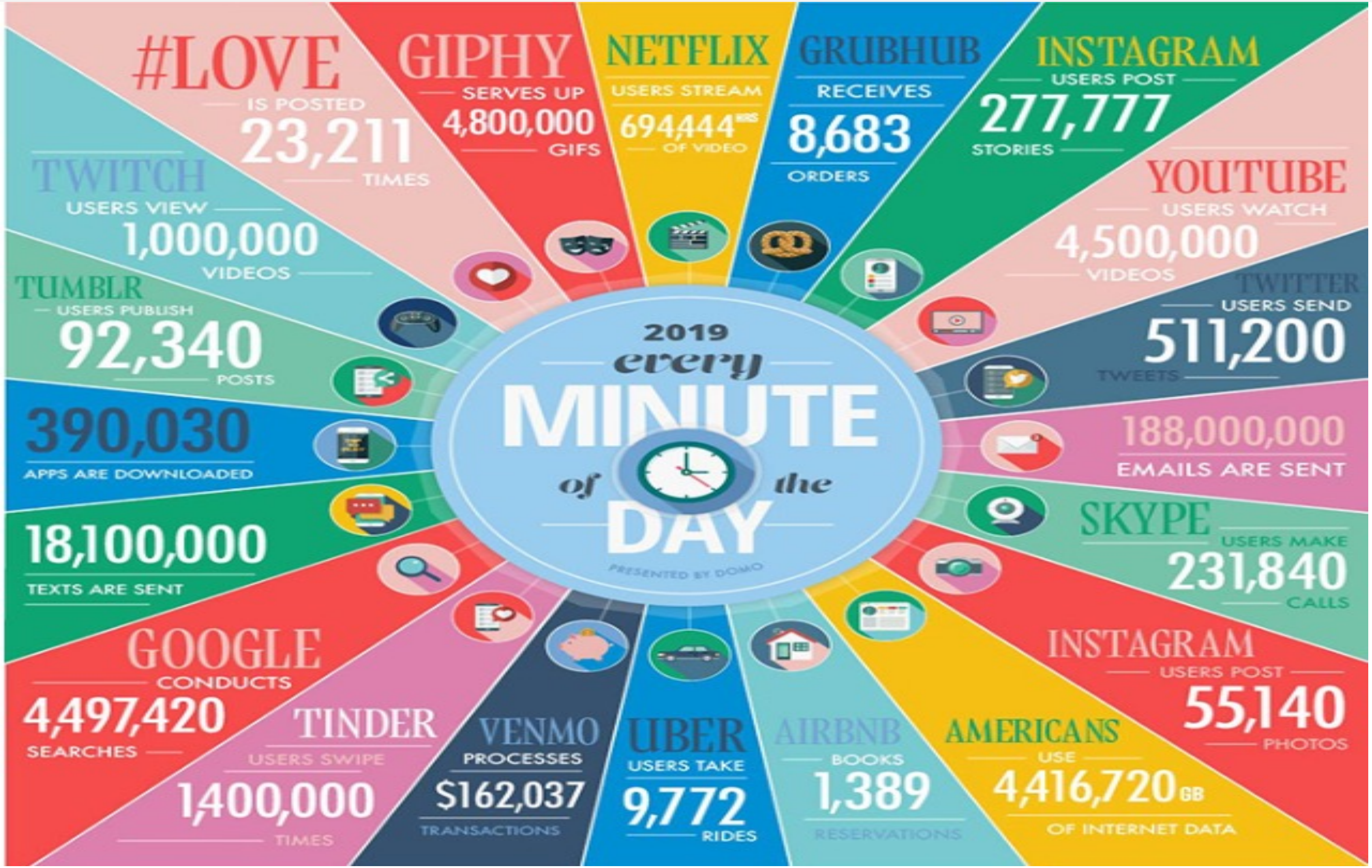
Velocity

Volume

Variety

Data Never Sleeps 7.0

Pass it on…
Knowledge is power

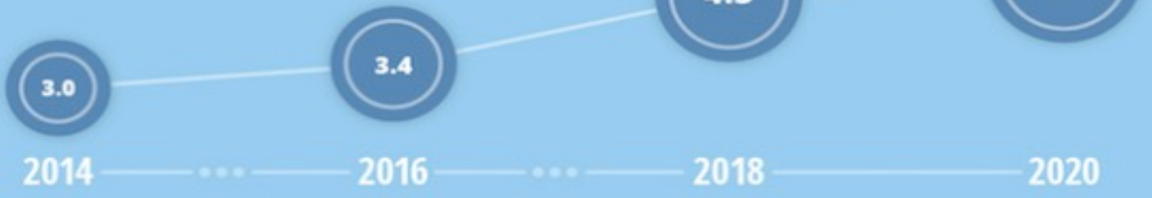


DATA NEVER SLEEPS 8.0

How much data is generated *every minute?*

In 2020, the world changed fundamentally—and so did the data that makes the world go round. As COVID-19 swept the globe, nearly every aspect of life—from work to working out—moved online, and people depended more and more on apps and the Internet to socialize, educate and entertain ourselves. Before quarantine, just 15% of Americans worked from home. Now over half do. And that's not the only big shift. In our 8th edition of Data Never Sleeps, we bring you the latest stats on how much data is being created in every digital minute—a trend that shows no sign of stopping.

The world's internet population is growing significantly year over year. As of April 2020, the internet reaches 59% of the world's population and now represents 4.57 billion people — a 6% increase from January 2019.

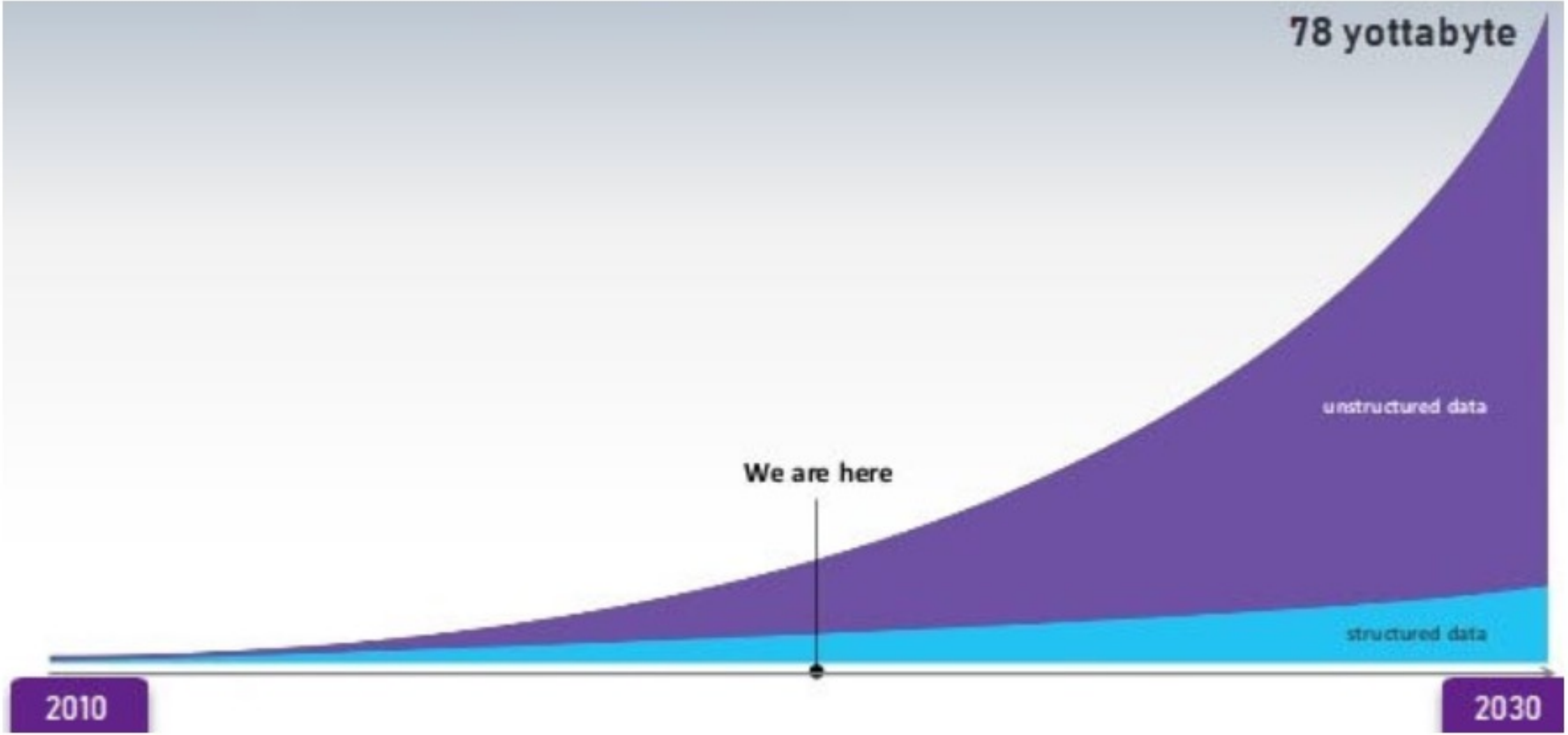


GLOBAL INTERNET POPULATION GROWTH 2014–2020
(IN BILLIONS)

As the world changes, businesses need to change with the times—and that requires data. Every click, swipe, share or like tells you something about your customers and what they want, and Domo is here to help your business make sense of all of it. Domo gives you the power to make data-driven decisions at any moment, on any device, so you can make smart choices in a rapidly changing world.

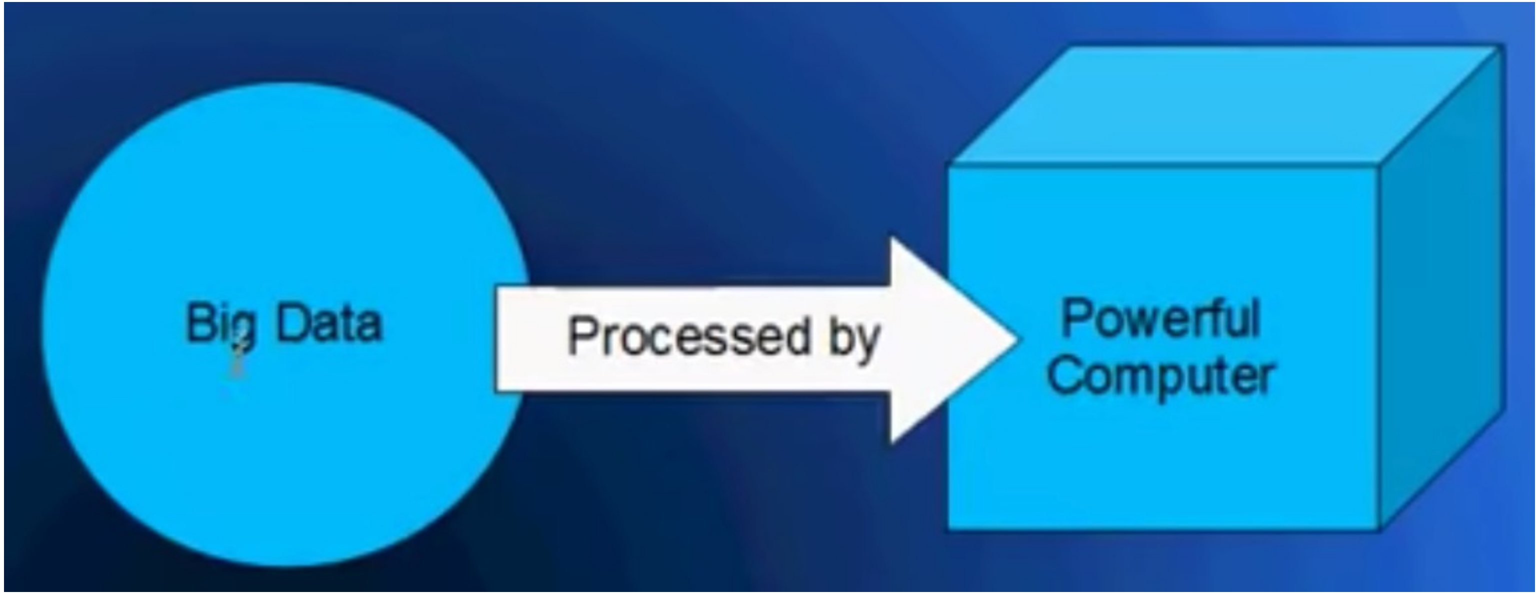
Learn more at domo.com

Data Growth over the years



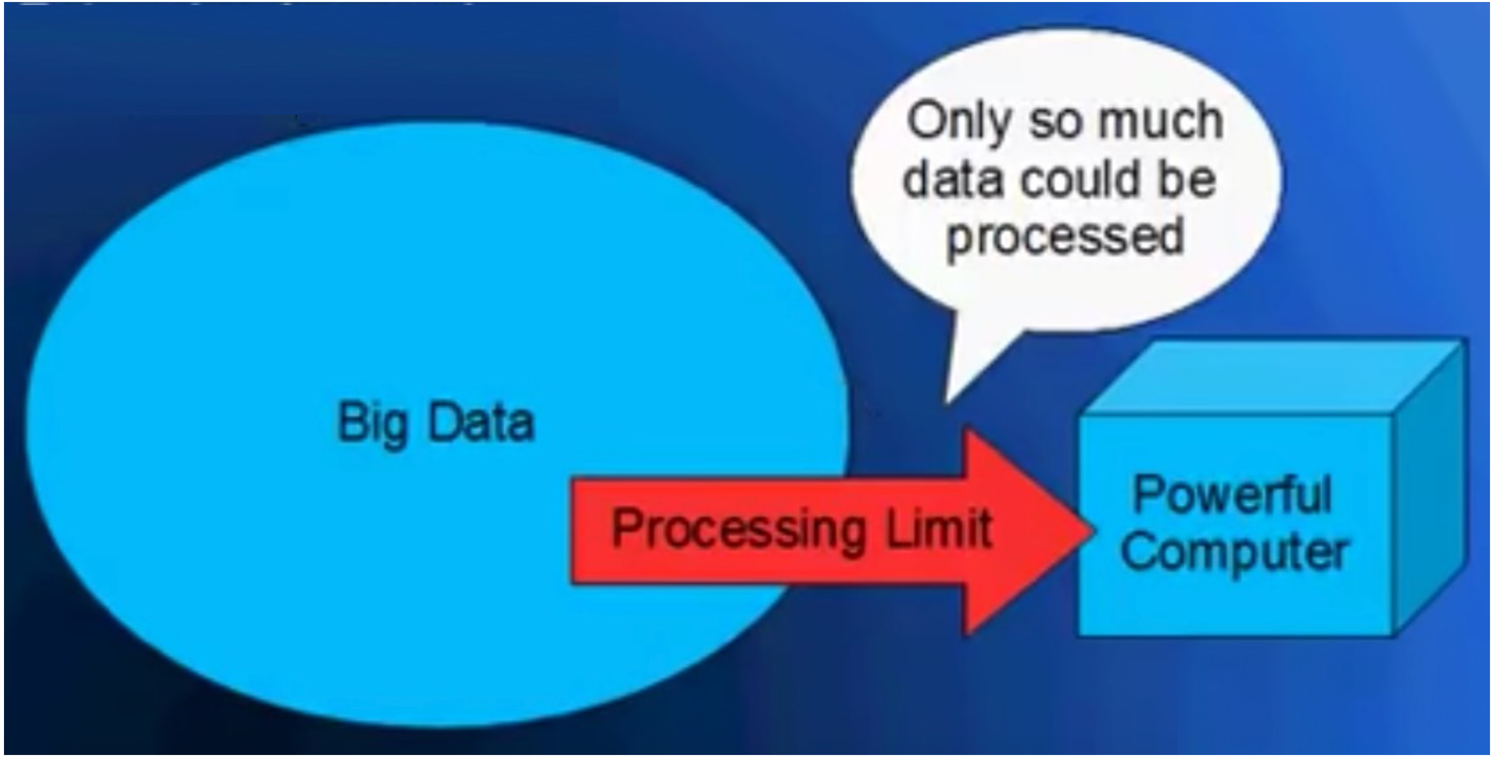
Traditional Approach

Monolithic Approach

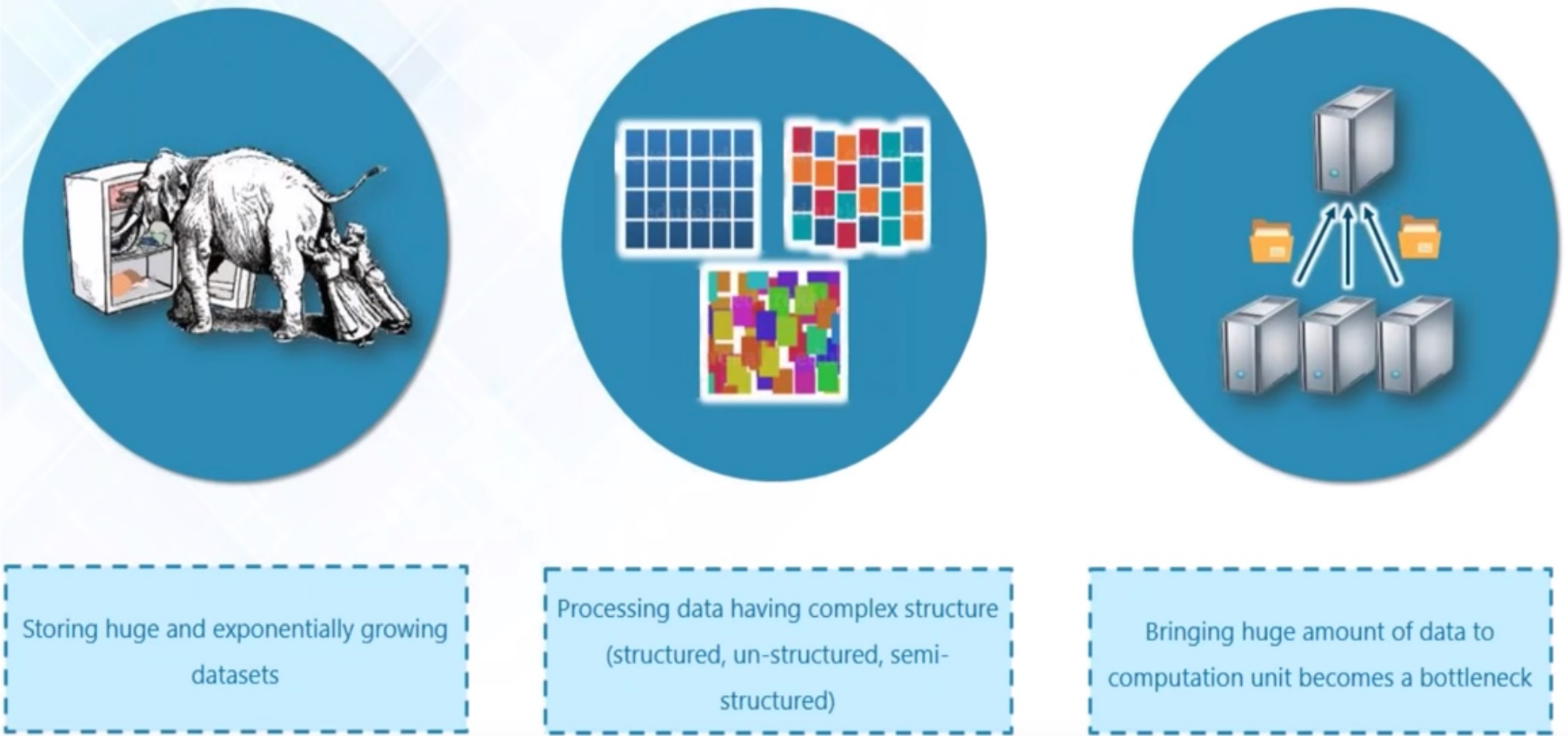


Traditional Approach

Monolithic Approach



Problems with Big Data

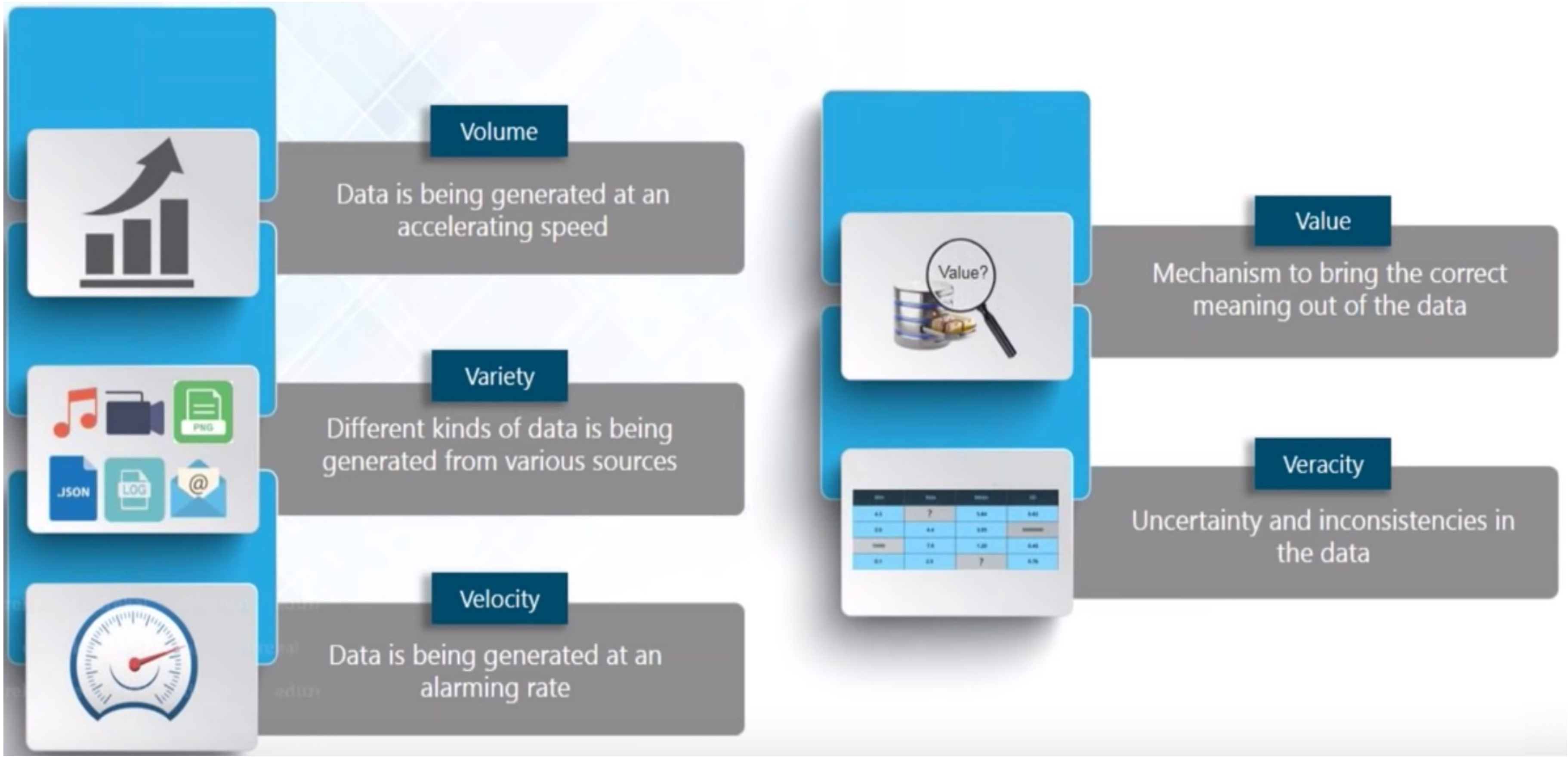


Storing huge and exponentially growing datasets

Processing data having complex structure (structured, un-structured, semi-structured)

Bringing huge amount of data to computation unit becomes a bottleneck

5 V’s of Big Data?



Volume

Data is being generated at an accelerating speed

Variety

Different kinds of data is being generated from various sources

Velocity

Data is being generated at an alarming rate

Value

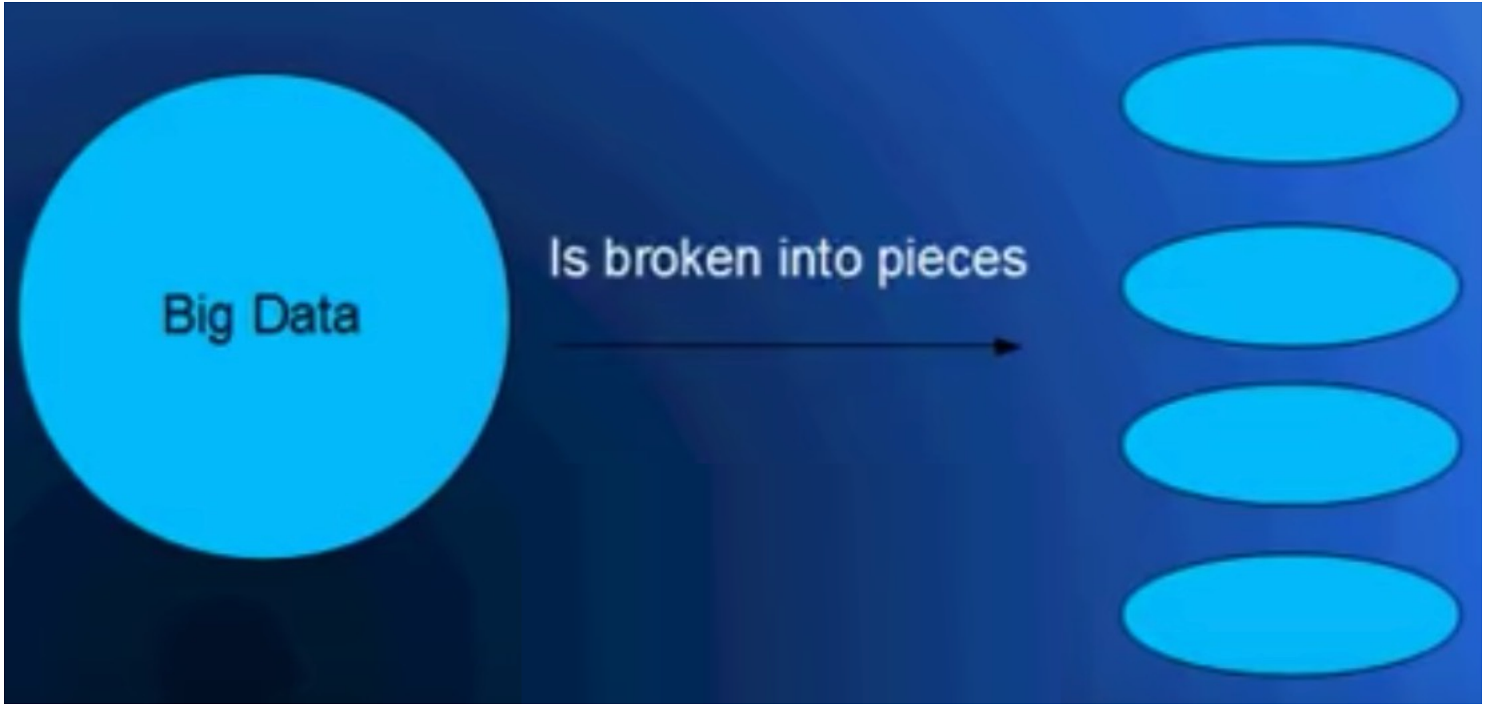
Mechanism to bring the correct meaning out of the data

Veracity

Uncertainty and inconsistencies in the data

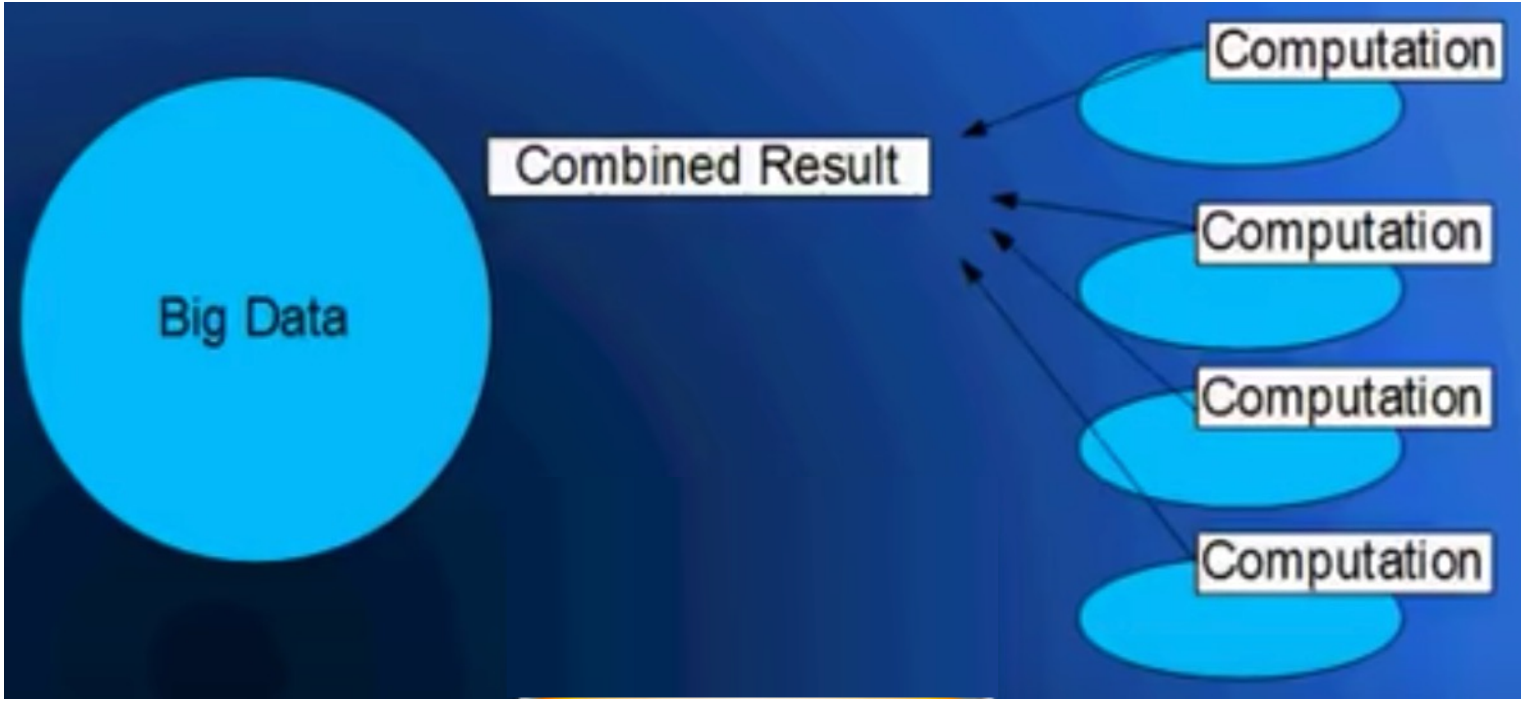
Breaking the Data

Hadoop’s Approach

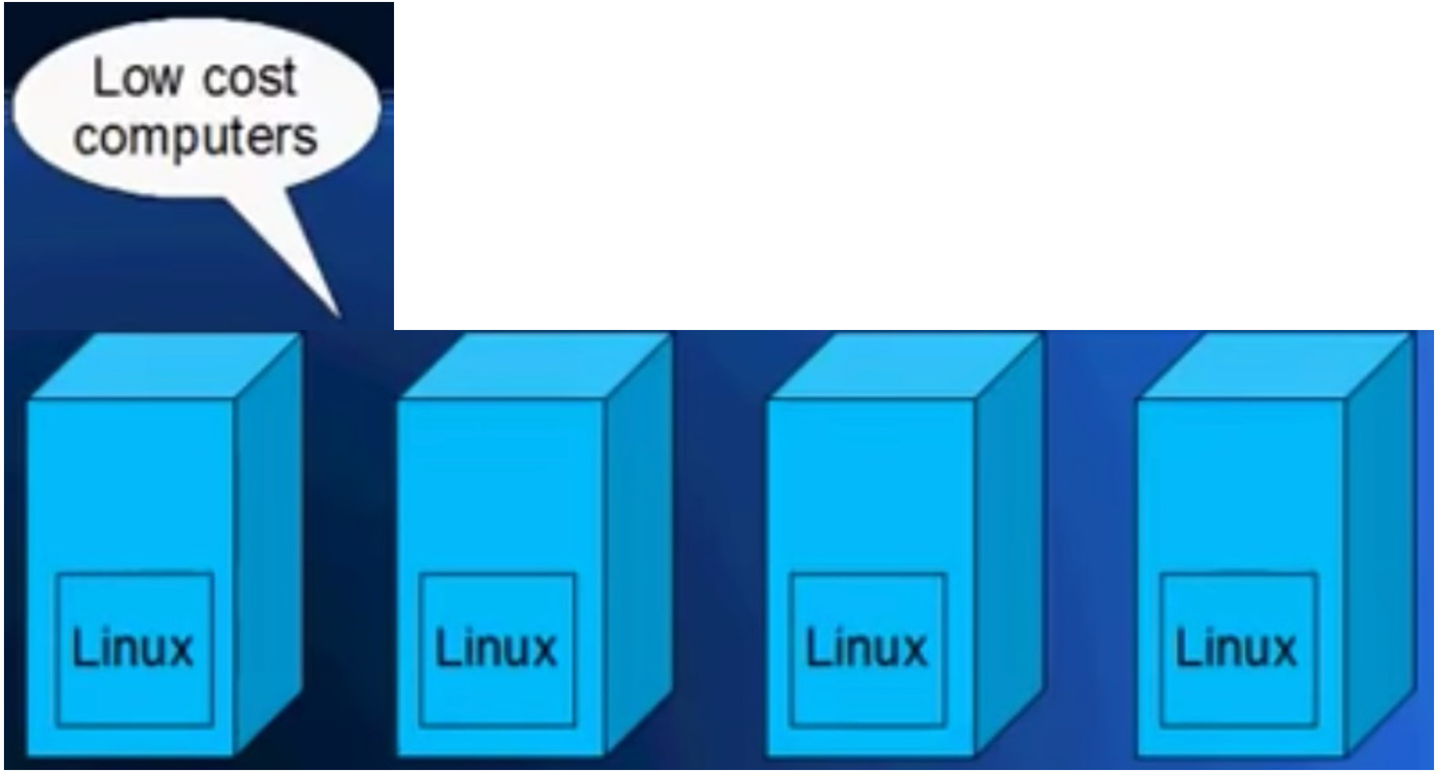


Breaking the Data

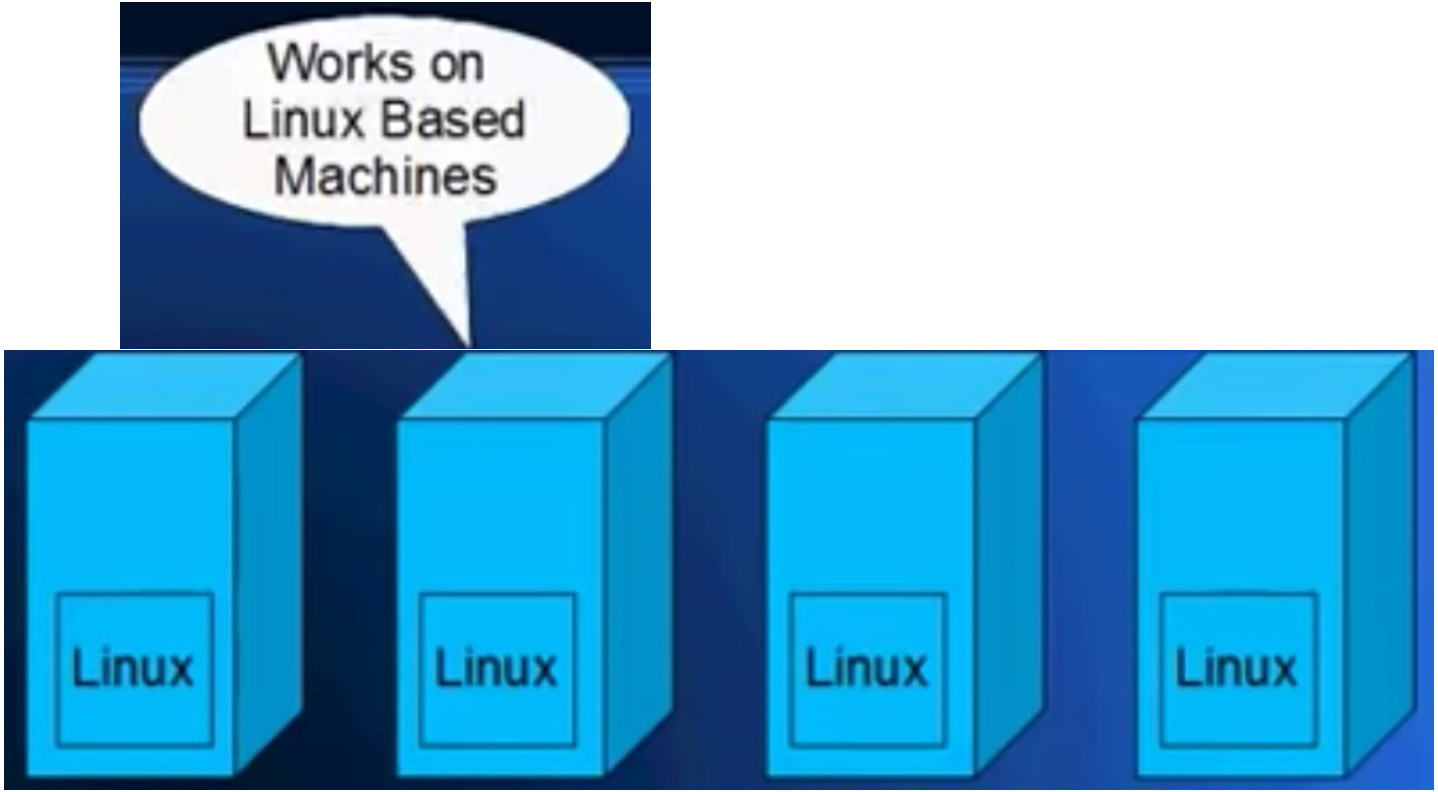
Hadoop’s Approach



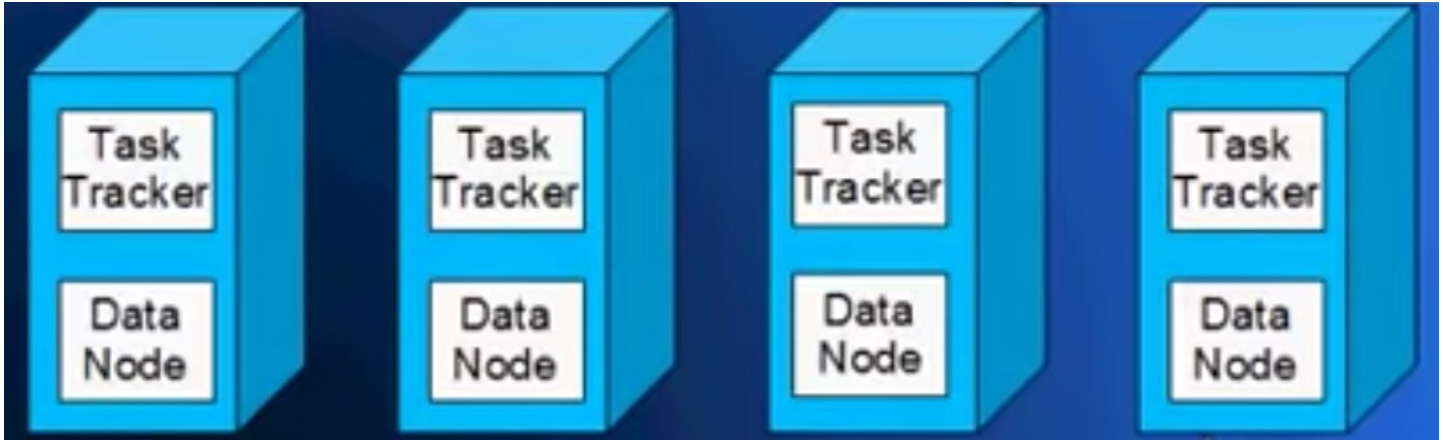
Distributed Model



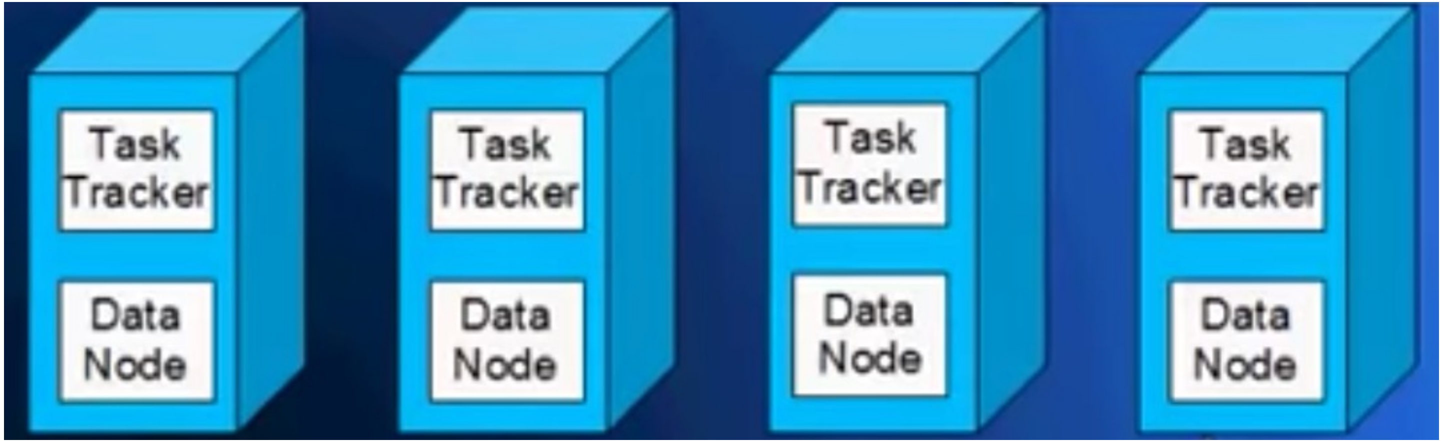
Linux Based



Task Trackers and Data Nodes

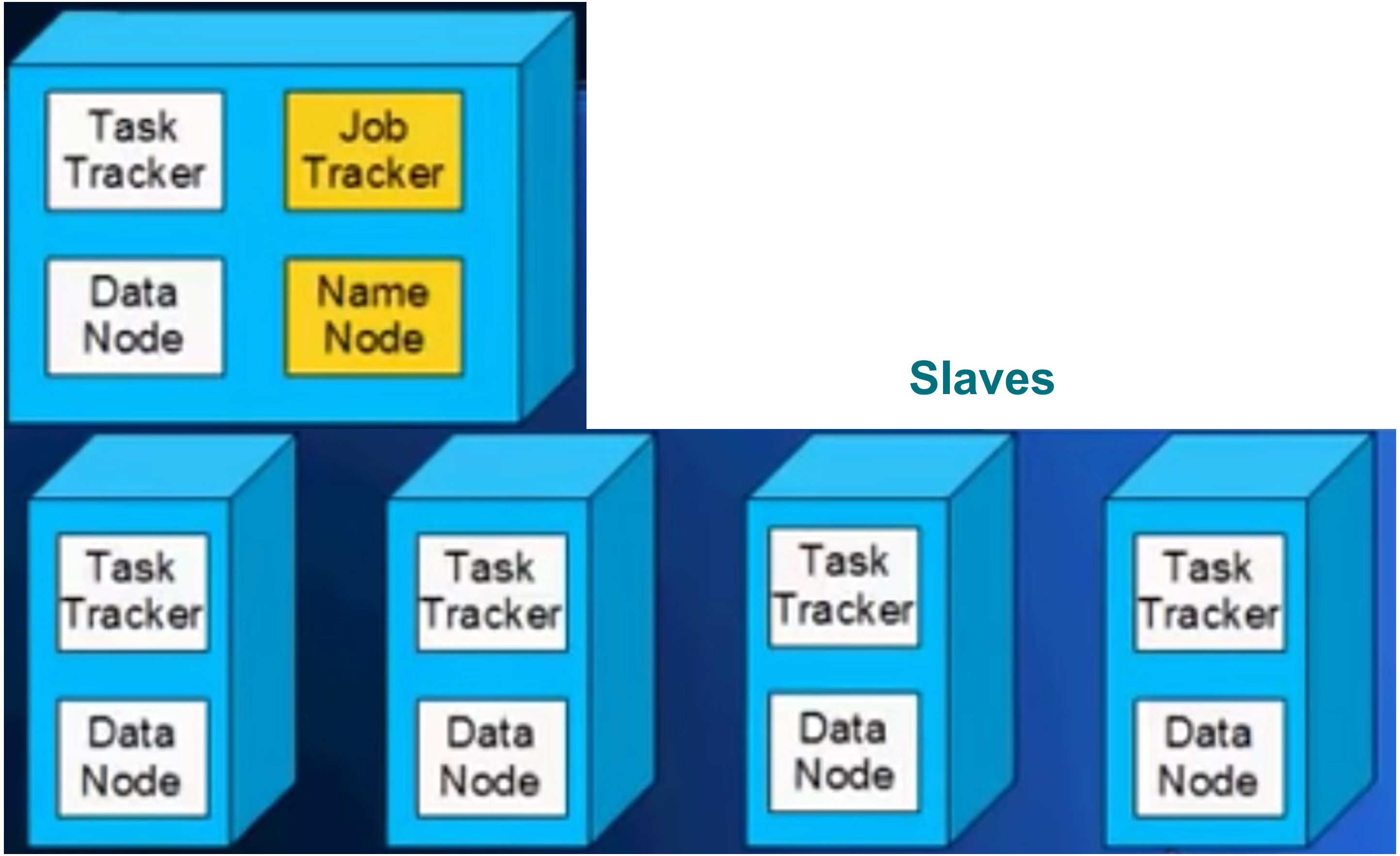


Slaves


Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node

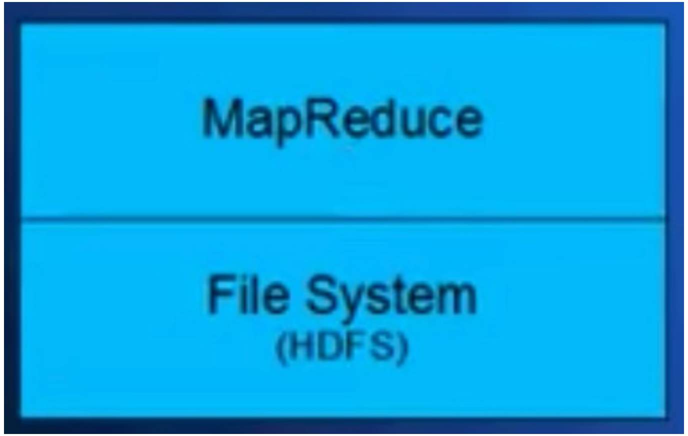
Master Slave Architecture

Master



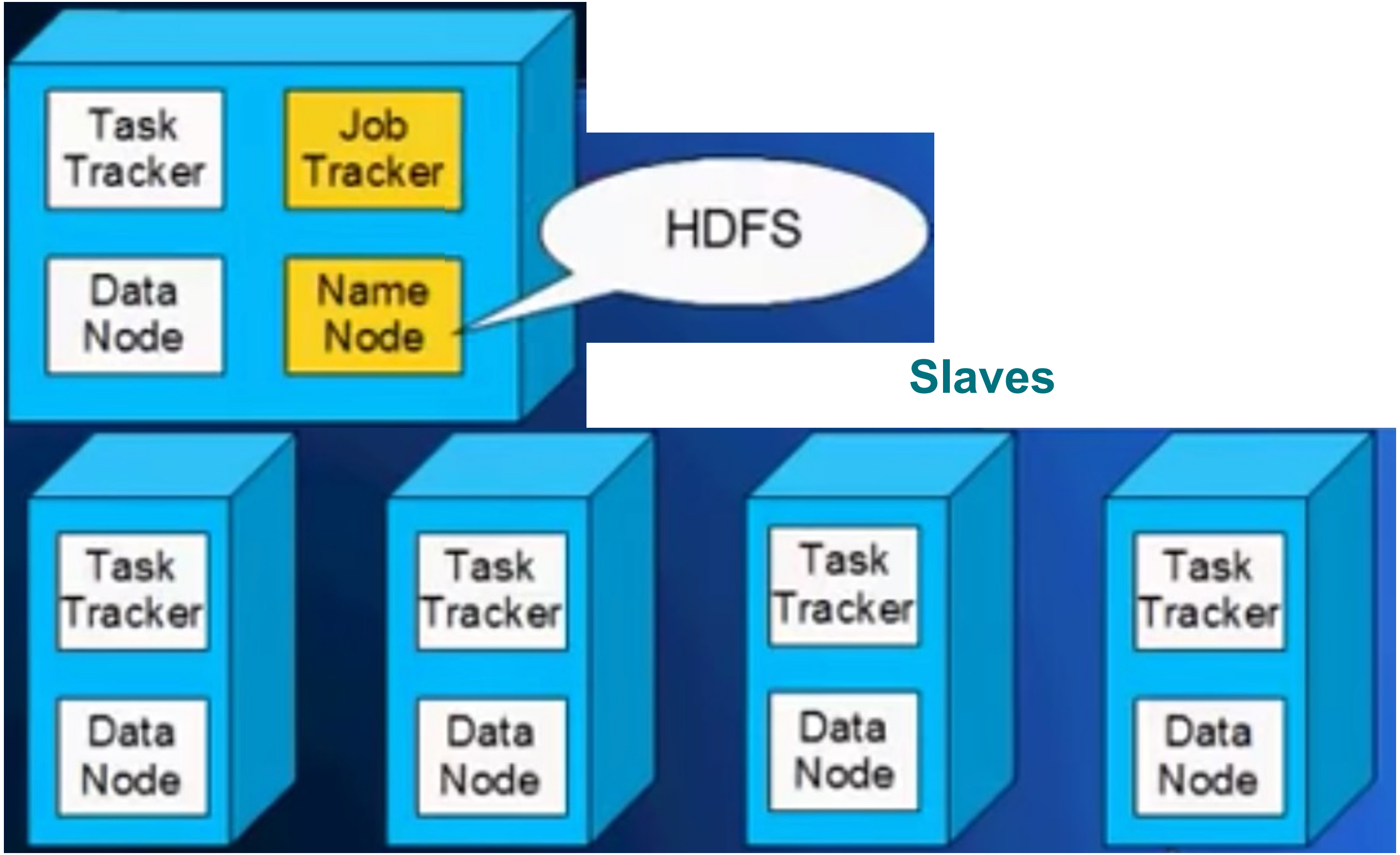
Slaves

Components



HDFS

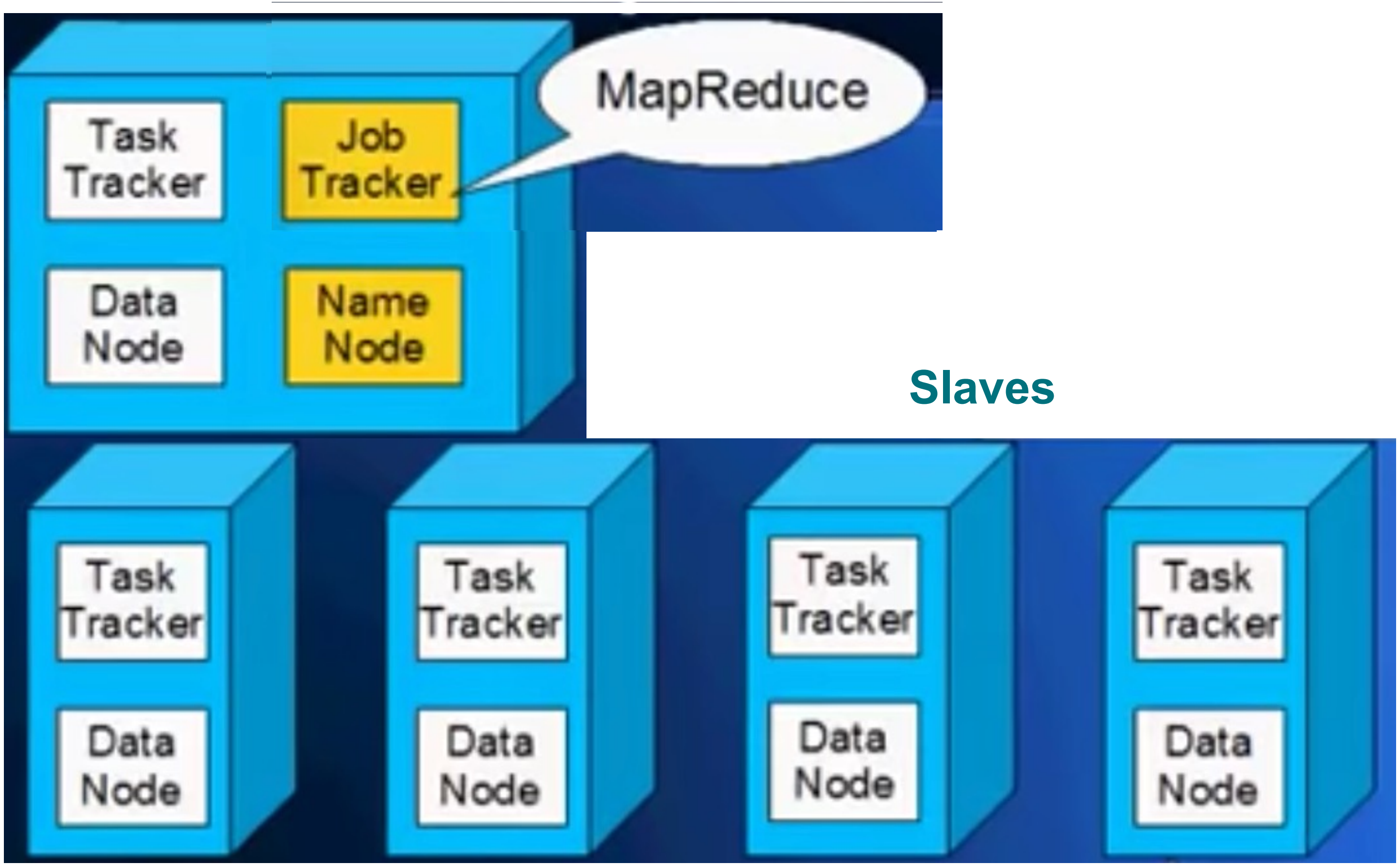
Master



Slaves

MapReduce

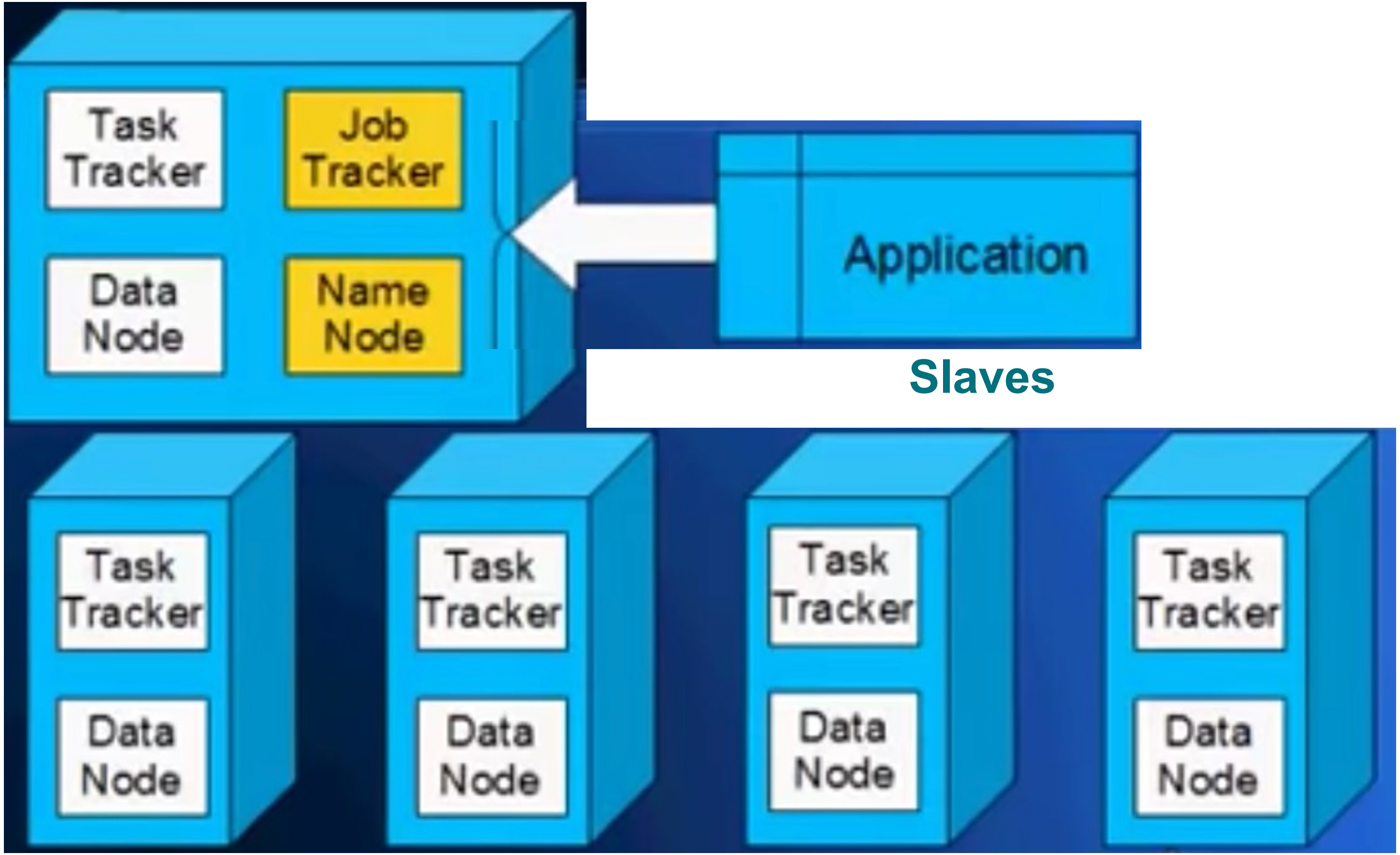
Master



Slaves

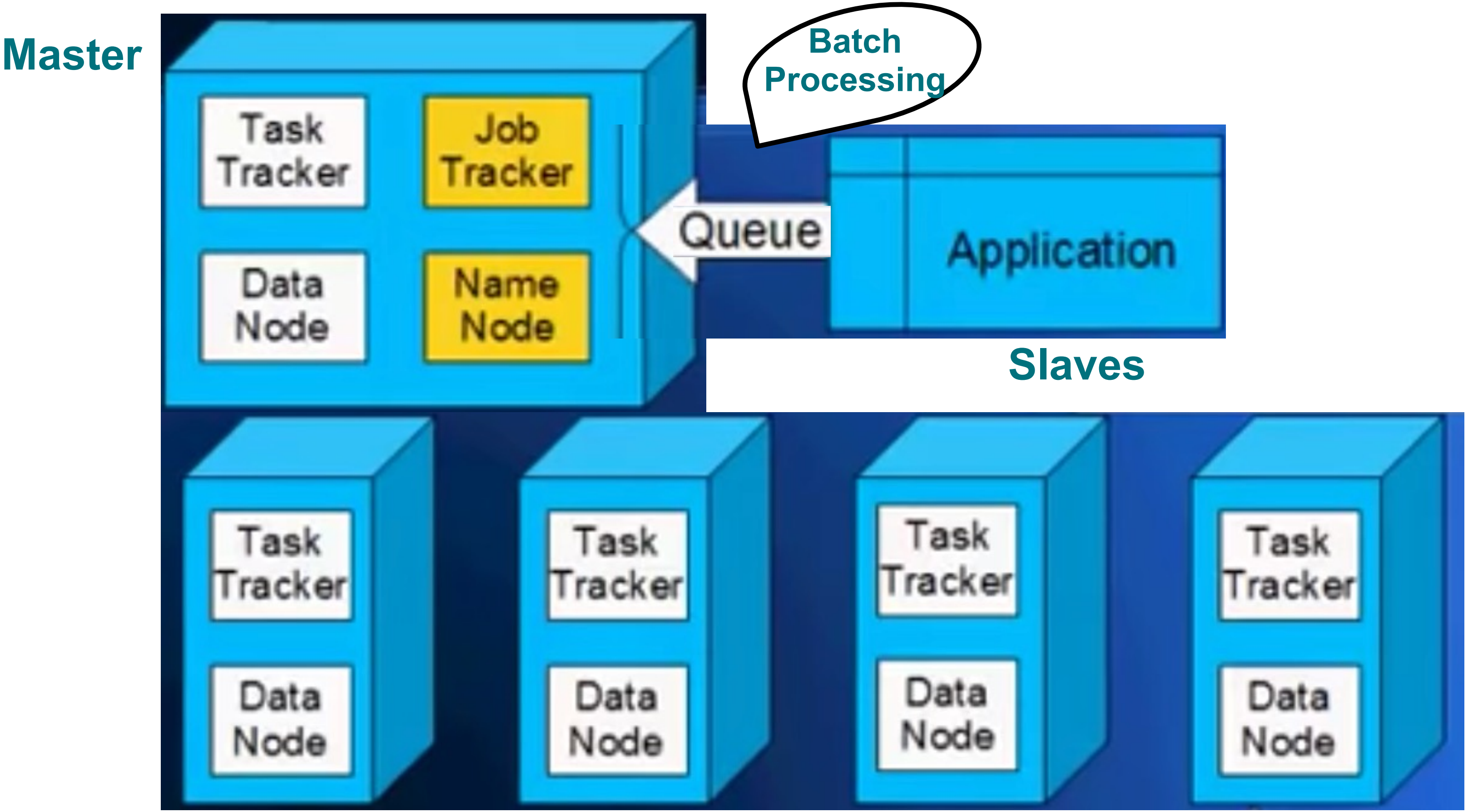
Applications

Master

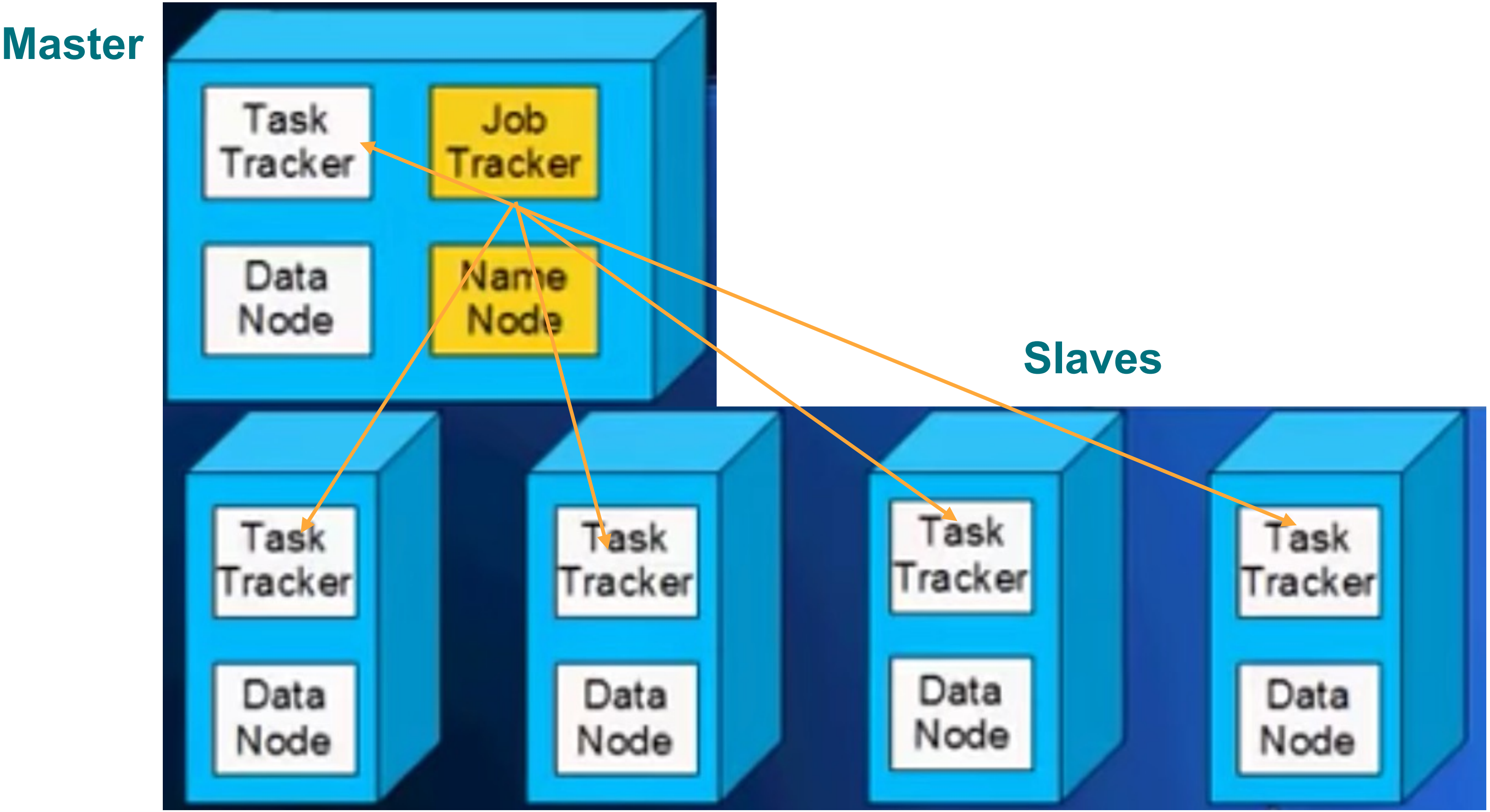


Slaves

Batch Processing

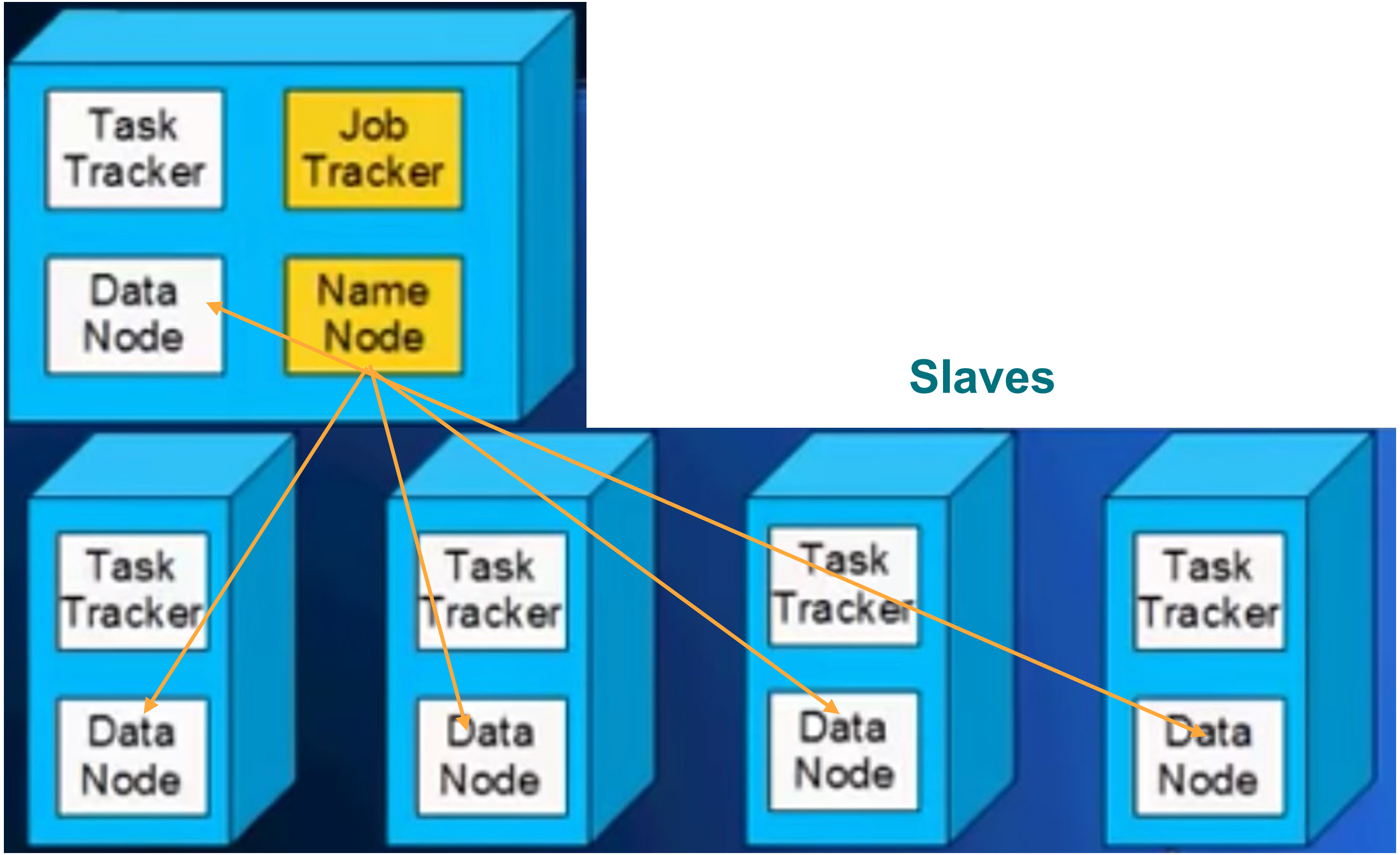


Job Tracker


Master
Task Tracker
Job Tracker
Data Node
Name Node
Slaves
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node

Name Node

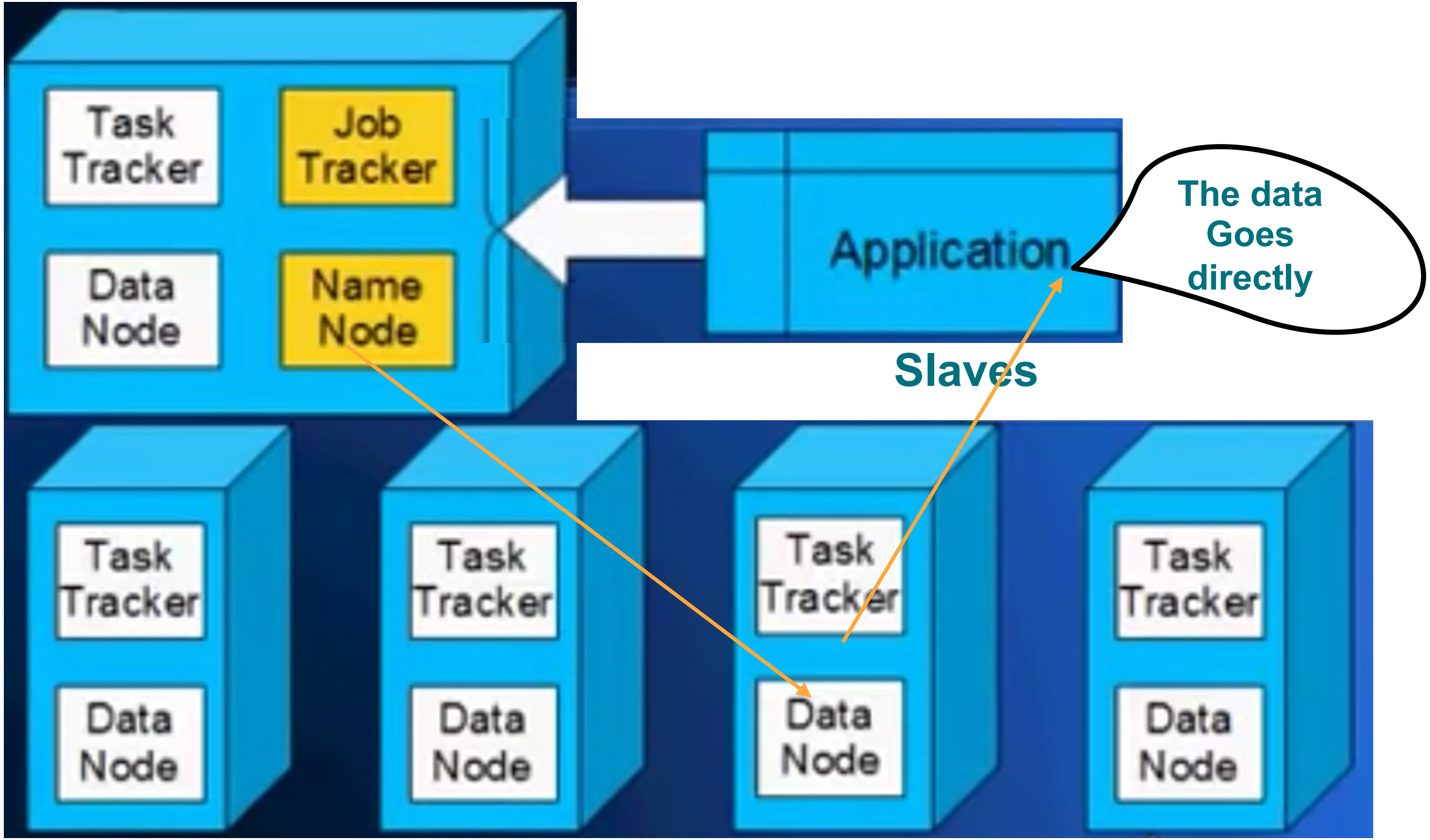
Master



Slaves

Data

Master

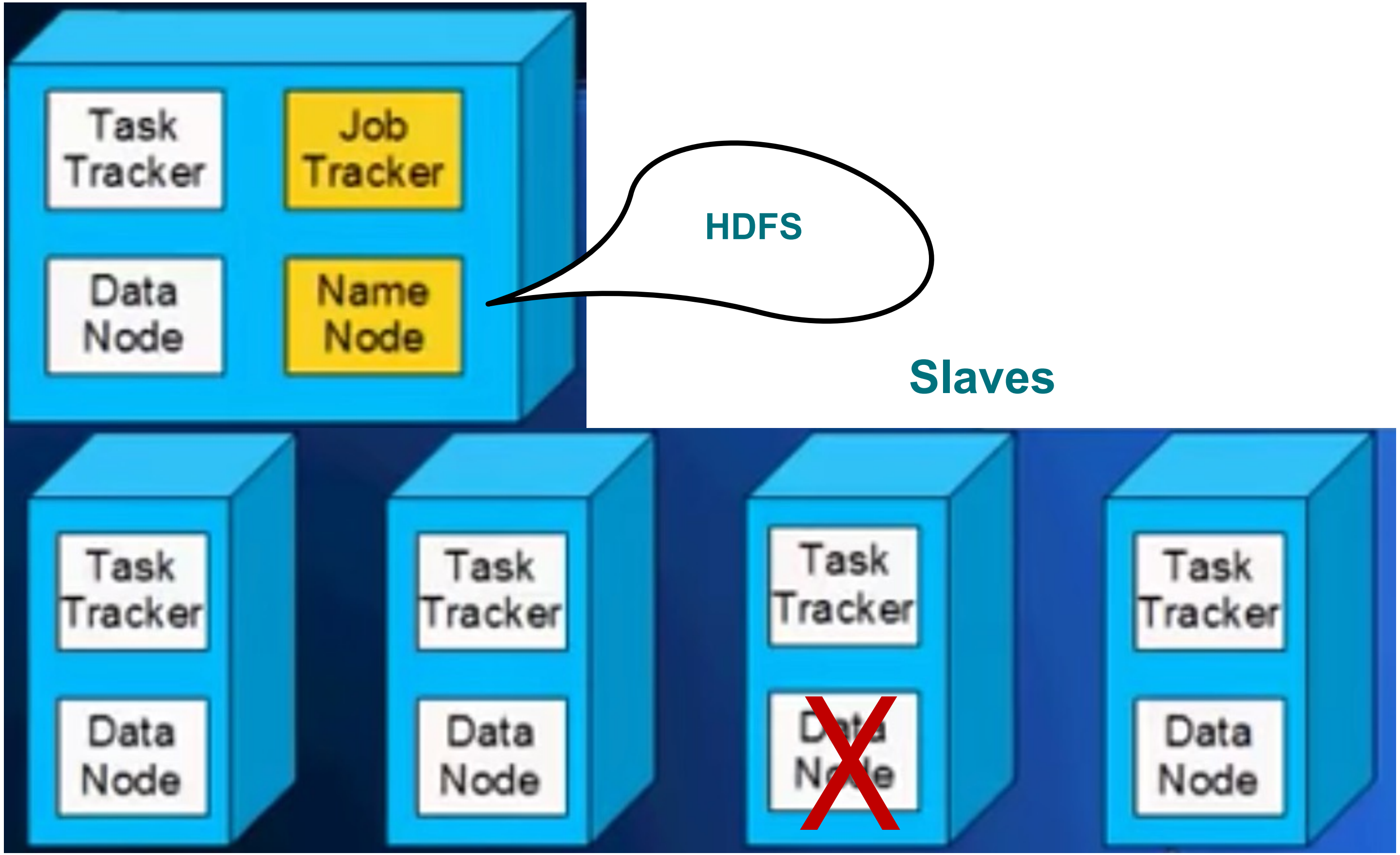


Slaves

The data Goes directly

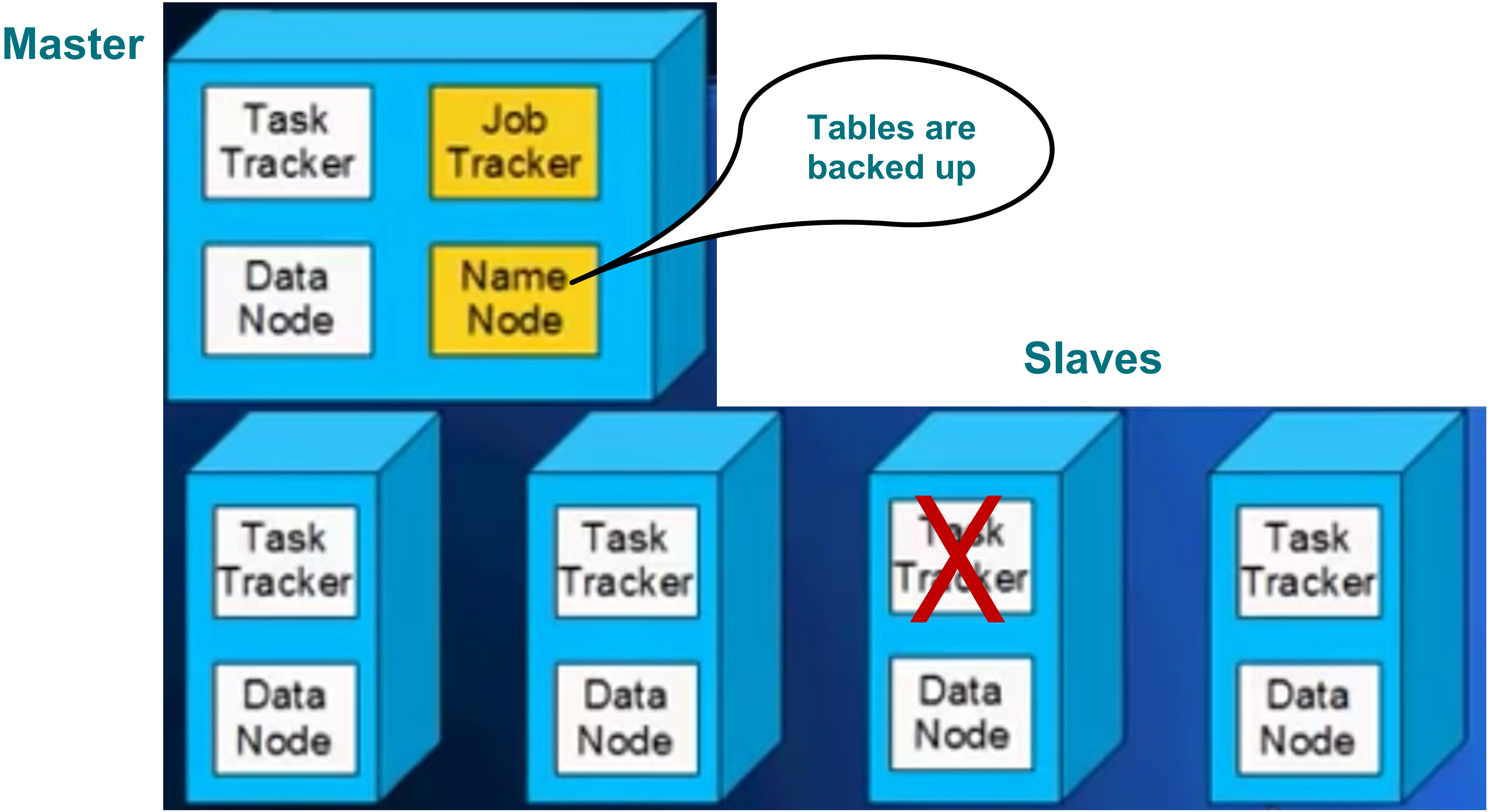
Fault Tolerance for Data

Master

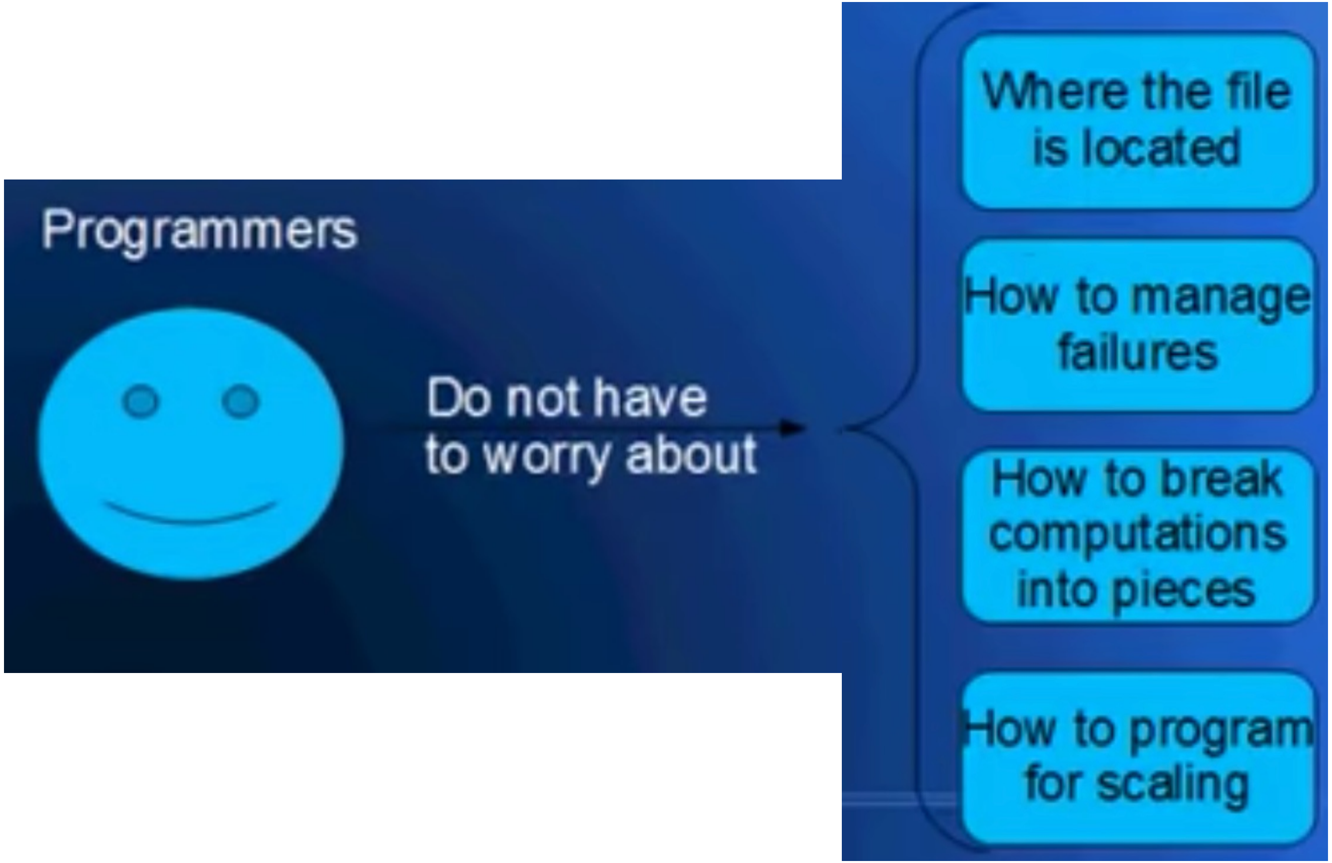


Slaves

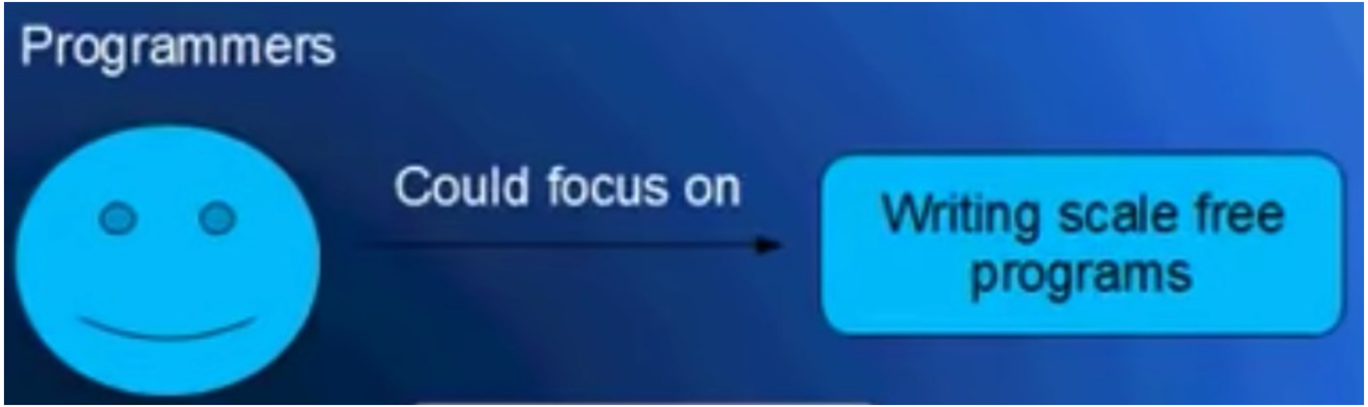
Master Backup



Easy Programming


Programmers
Do not have to worry about
Where the file is located
How to manage failures
How to break computations into pieces
How to program for scaling

Easy Programming

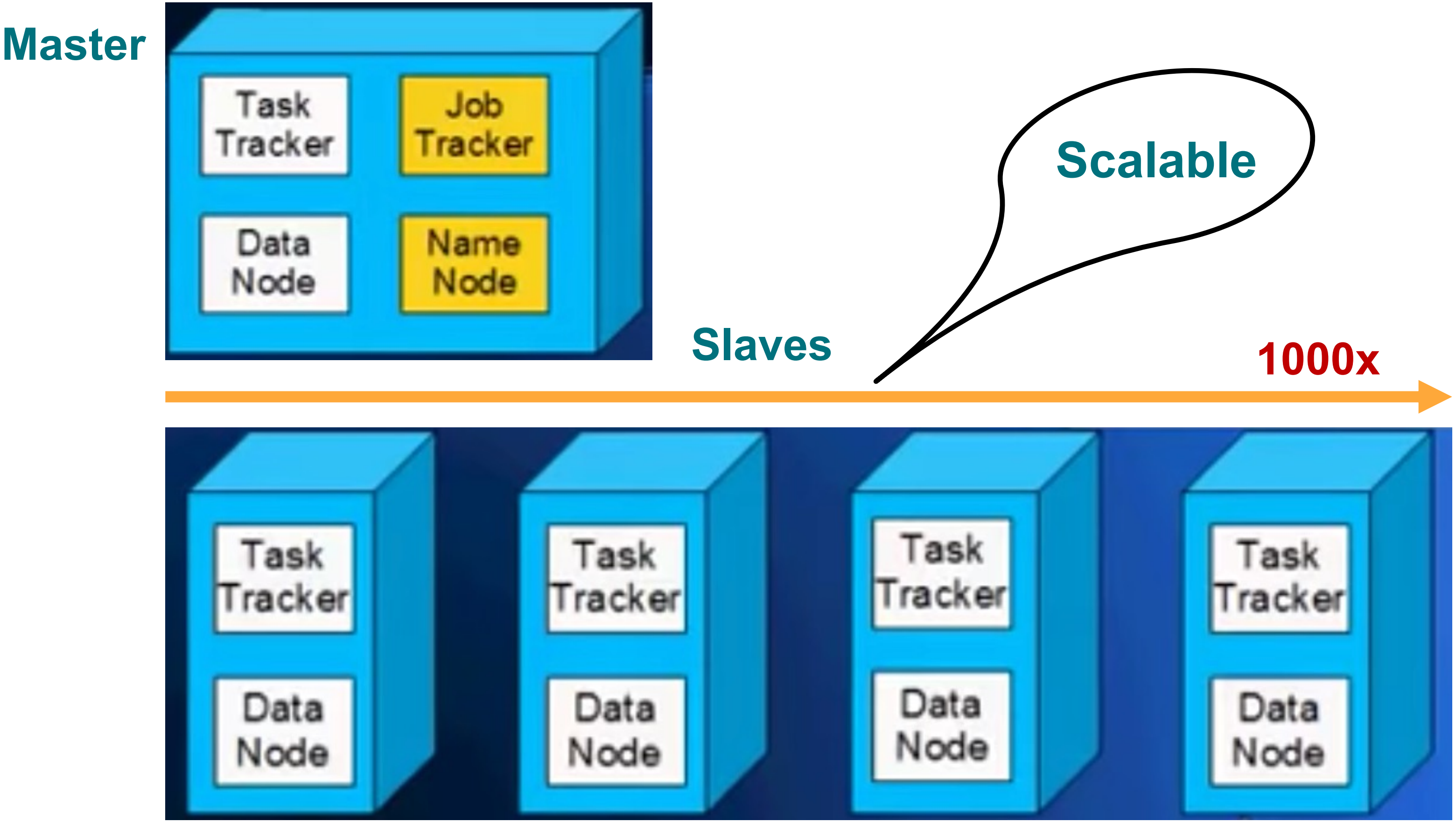


Programmers

Could focus on

Writing scale free programs

Scalable



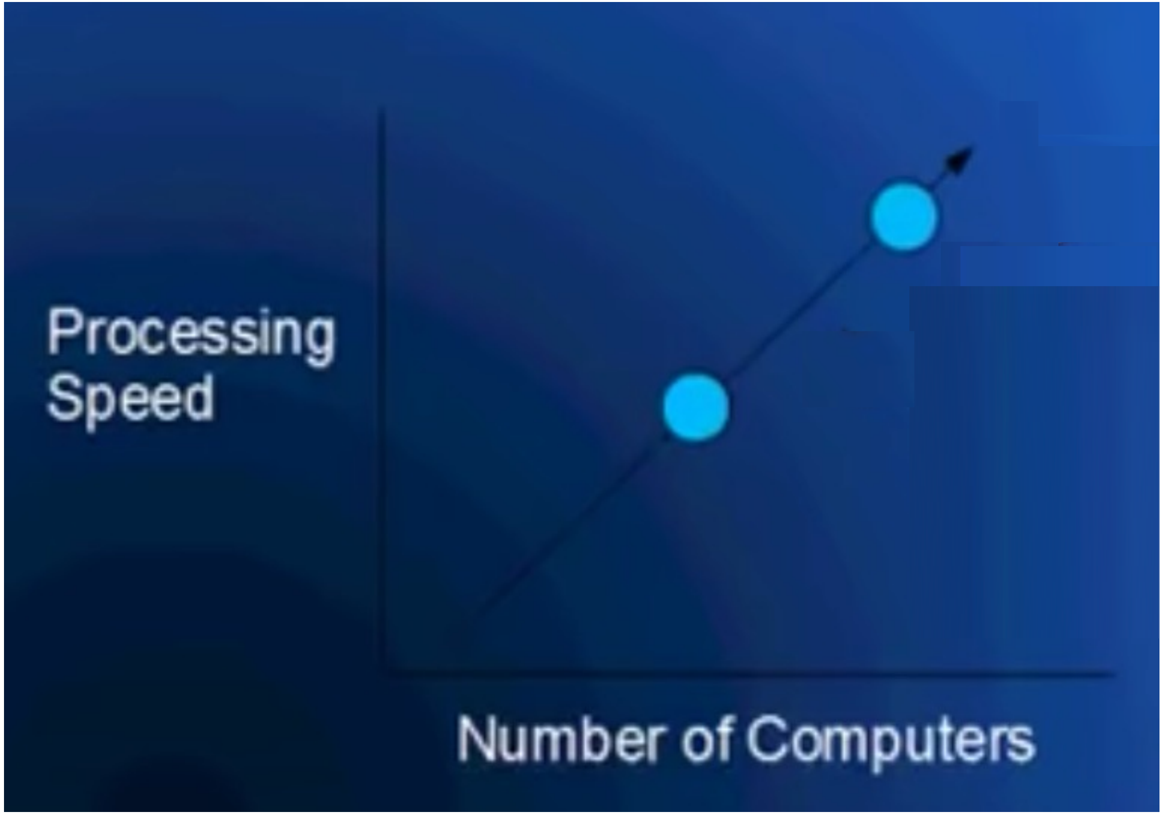
Master

Slaves

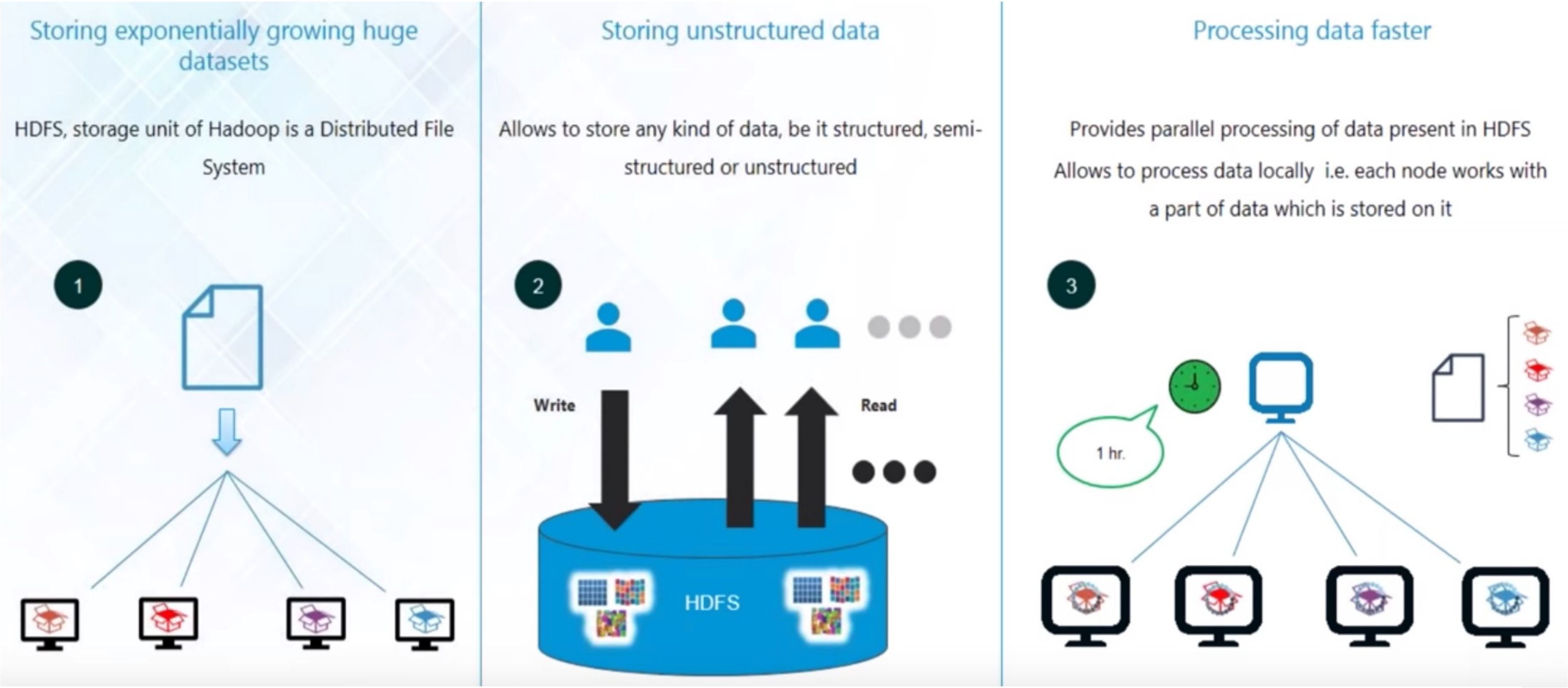
Scalable

1000x

Scalability Cost



Solving Big Data problem w/ Hadoop



Storing exponentially growing huge datasets

HDFS, storage unit of Hadoop is a Distributed File System

Storing unstructured data

Allows to store any kind of data, be it structured, semi-structured or unstructured

Processing data faster

Provides parallel processing of data present in HDFS

Allows to process data locally i.e. each node works with a part of data which is stored on it

Distributed Computing

University of CINCINNATI

Needs management of components for distribution of work:

- ➢ Compute resources (CPU/RAM)
- ➢ Long-Term Storage (Filesystem)
- ➢ User-supplied “algorithm” or “work”
- ➢ Scheduling/dependency

Hadoop Platform

University of CINCINNATI

Provides framework/API for distributed computing

- ➢ Hide away distributed back-end
- ➢ Provides consolidated abstraction for:
  - ○ Workload distribution
  - ○ Storage and replication
  - ○ Application development
  - ○ Fault tolerance

Hadoop Core Components

- Hadoop Common:
  - Common shared library
  - Runtime environment
- Hadoop Distributed File System (HDFS)
  - Storage abstraction
  - Fault tolerance
  - Data replication
- Hadoop YARN (Yet Another Resource Negotiator)
  - Compute management layer
  - Distributed OS

Hadoop Architecture

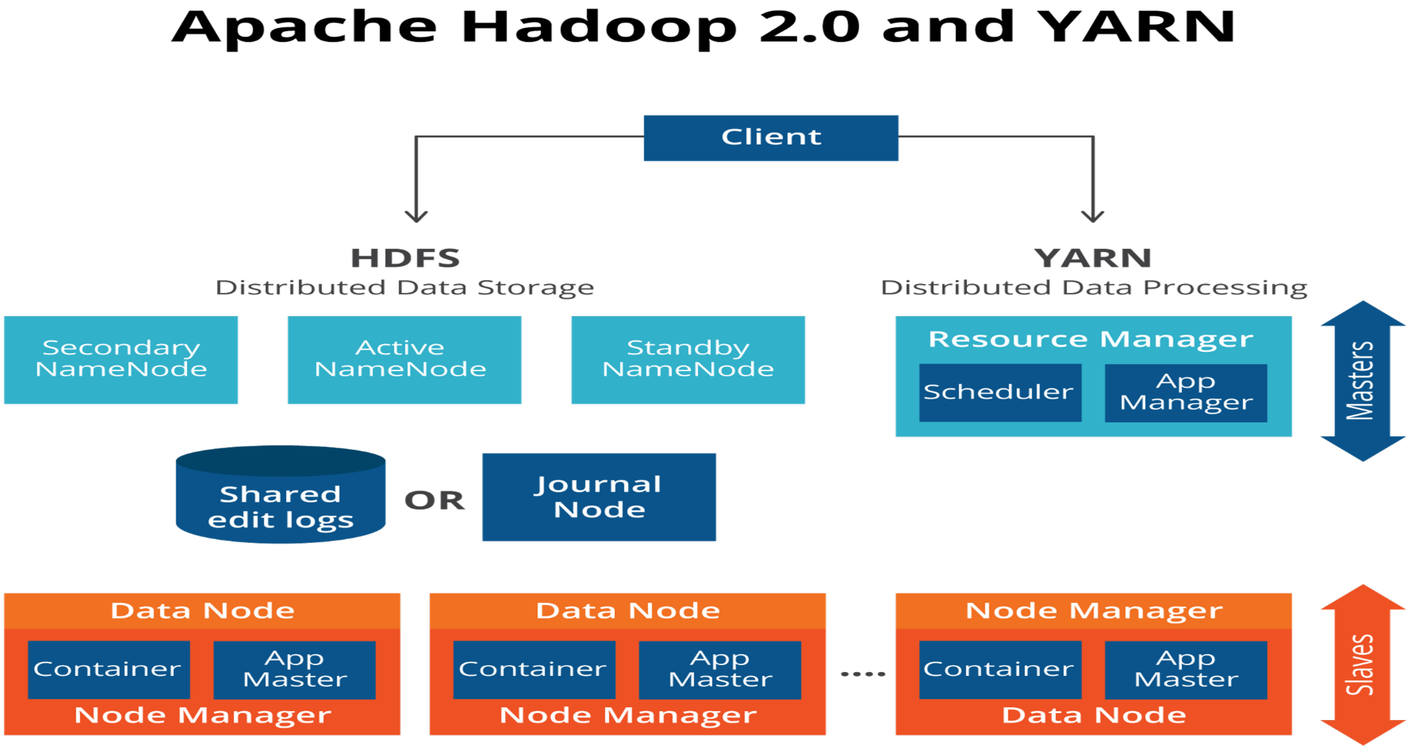


Image Link: https://goo.gl/images/VxQCe6

Hadoop Properties

- At its core - Java application thus platform independent
- Data-centric
- Core tools & frameworks
- Cluster built out of “nodes” that provide resources - execution time, disk space, etc.
- Application development system, to build distributed solutions to user data problems

Hadoop Common Components

- Basic primitives for programming environment:
  - Core data types (writable)
    - Text, Int (V), Boolean, Byte, Long (V), Double
  - Exceptions handling
  - Data structures
    - Arraywritable, TwoD Array Writable, Object Writable
  - IPC primitives
  - Logic, math, computation runtime
  - Data handling/encoding/transfer
    - Compressed files (.tar, .tar.gz, .tar.bz)

HDFS Architecture

Built from nodes, which serves processes deployed to nodes in your cluster:

- ➢ Namenode
  - ○ Metadata is stored here
  - ○ Provides logical links to your data block
- ➢ Datanode
  - ○ Files are broken into smaller pieces and stored across these nodes

HDFS Filesystem

Filesystem abstraction for Hadoop clusters Manage storage efficiently for:

- ➢ Very large files
- ➢ Streaming data access
- ➢ Built on commodity hardware
- ➢ High availability
- ➢ Concurrent access

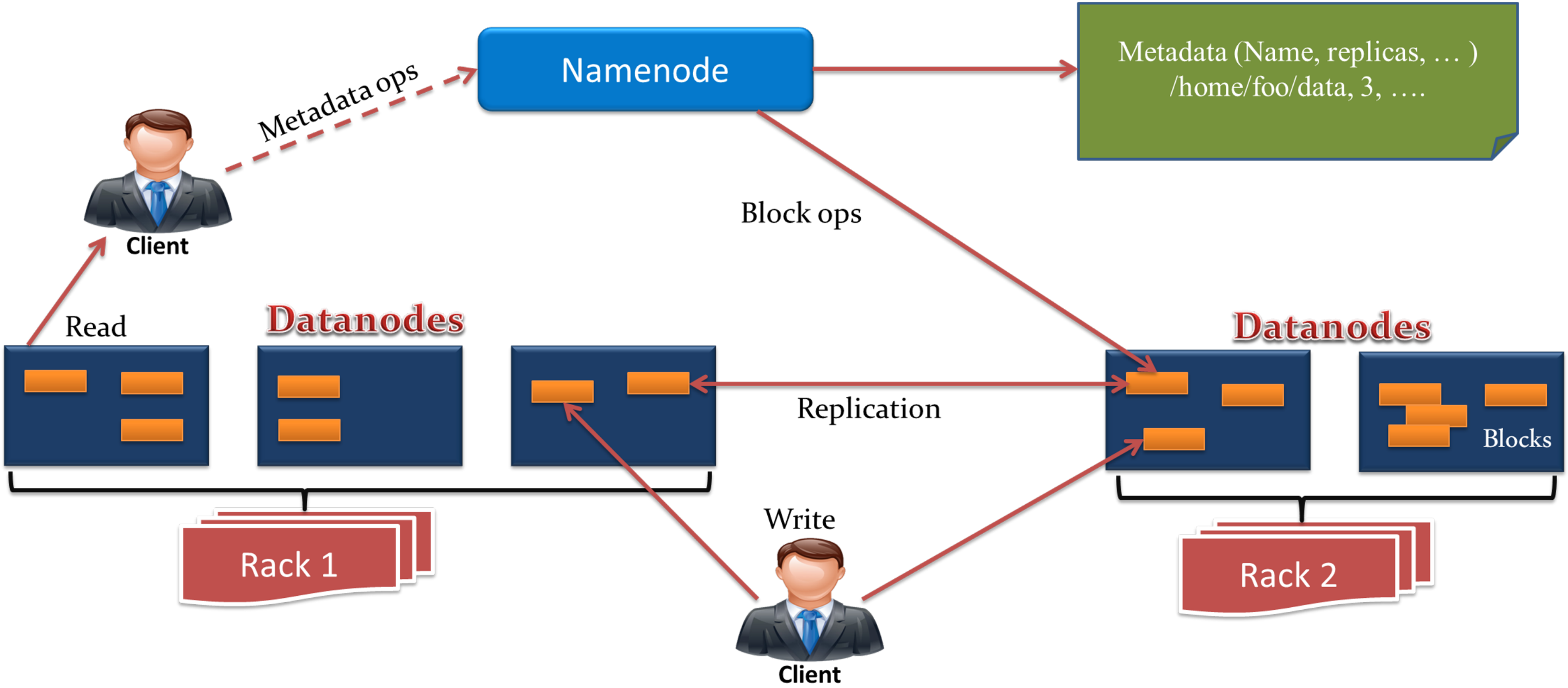
HDFS Limitations

Doesn’t work as well for

- ➢ Many small files
- ➢ Low-latency data access

In these cases, you may always use another storage abstraction or engine

HDFS Architecture

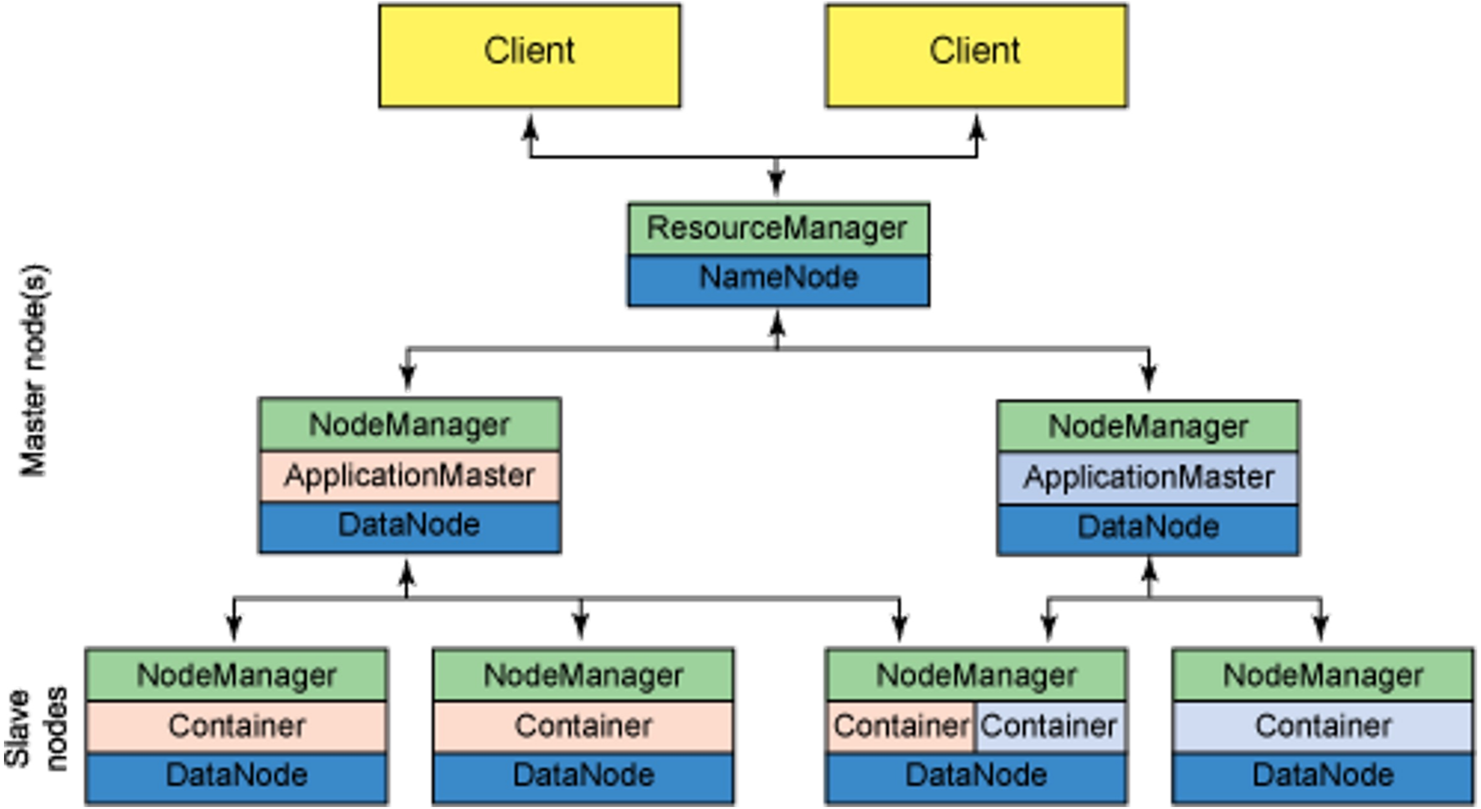


YARN Architecture

- As Resource Manager
  - Manages the compute resources in the cluster
  - Receives work from clients

- As NodeManager
  - Manages workload within an execution node in the cluster
  - Instantiates containers to execute workloads
  - Distributes to other Node Managers on-demand

YARN Architecture

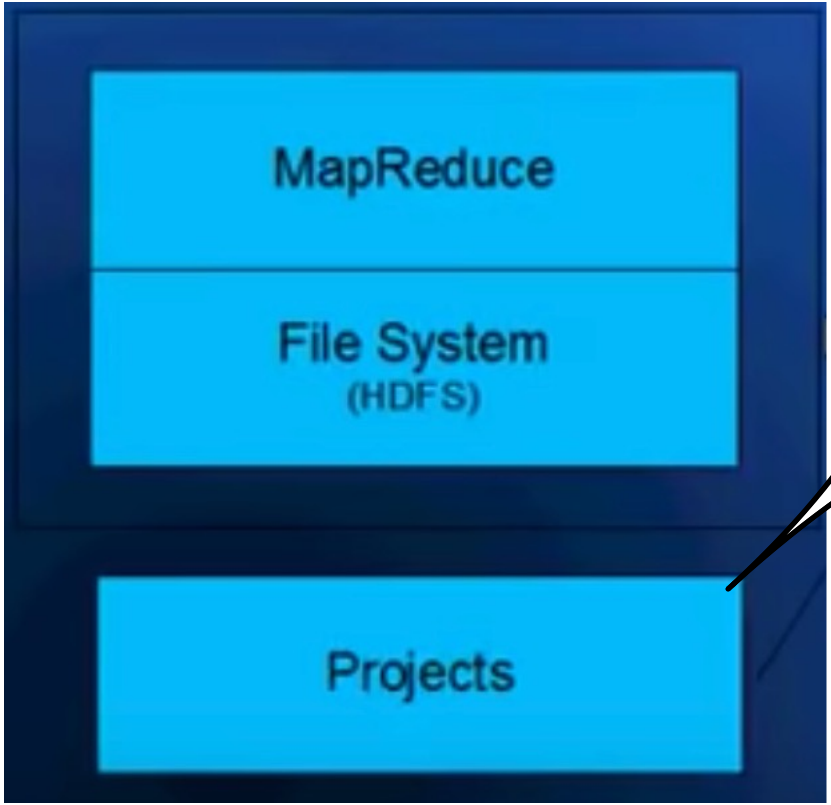


YARN Concepts

The platform development layer for Hadoop - or “Cluster OS”

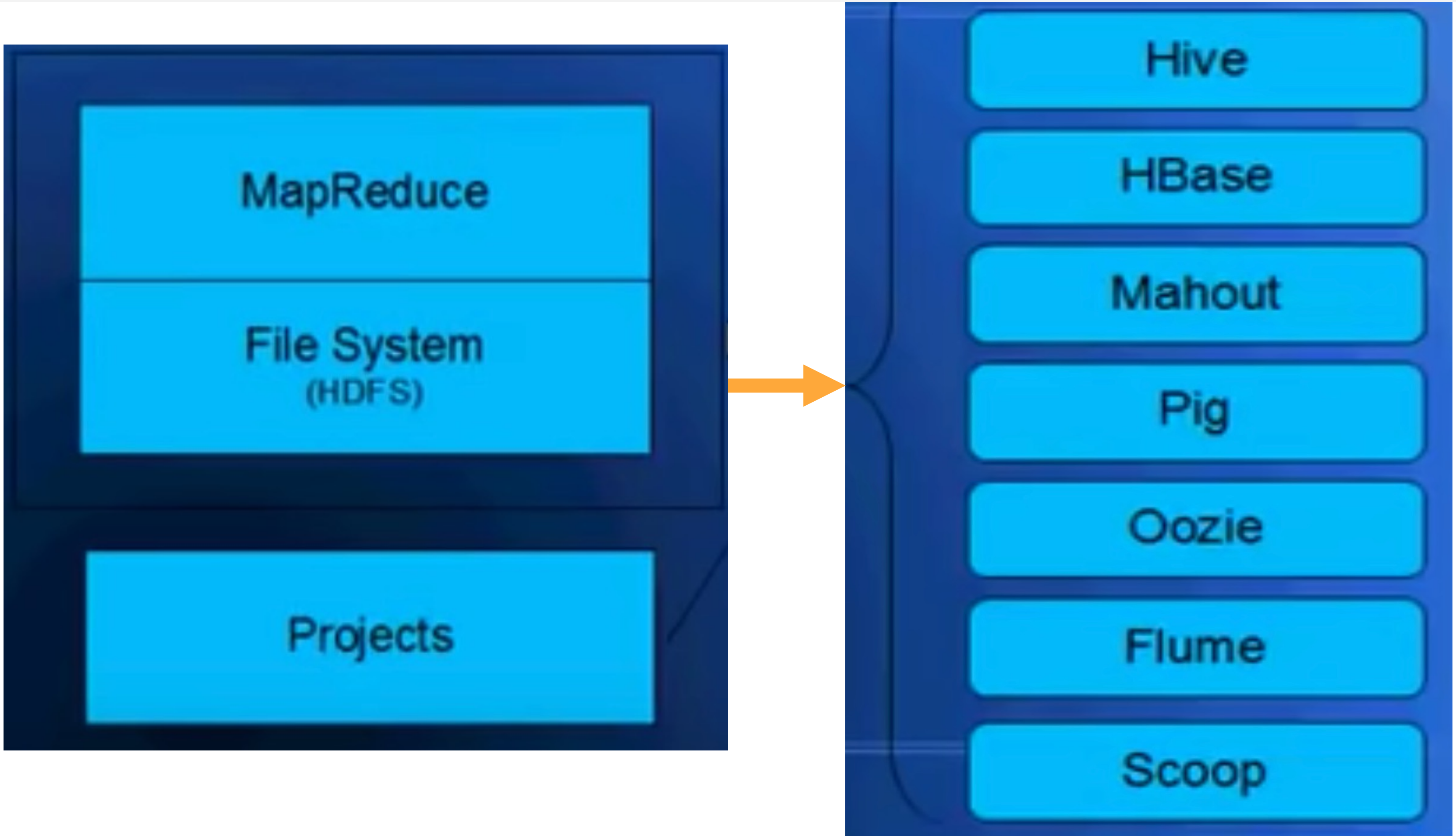
- In the context above,
    - It almost acts as a means of IaaS(M)
    - PaaS-like layers built atop this to provide “application interfaces” to Hadoop users.
- Examples include: MapReduce, Spark

Projects



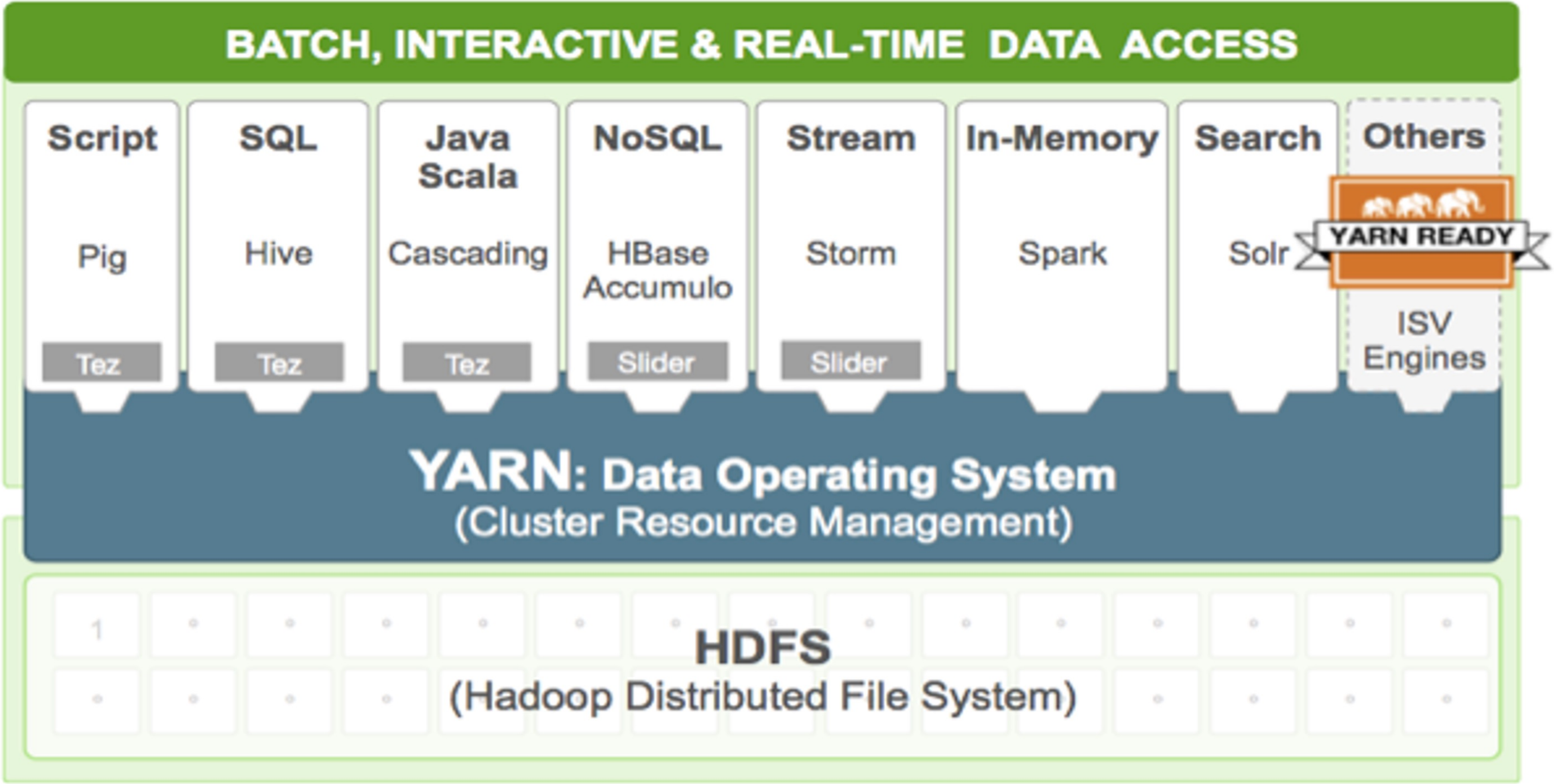
Provide additional functionality

Projects

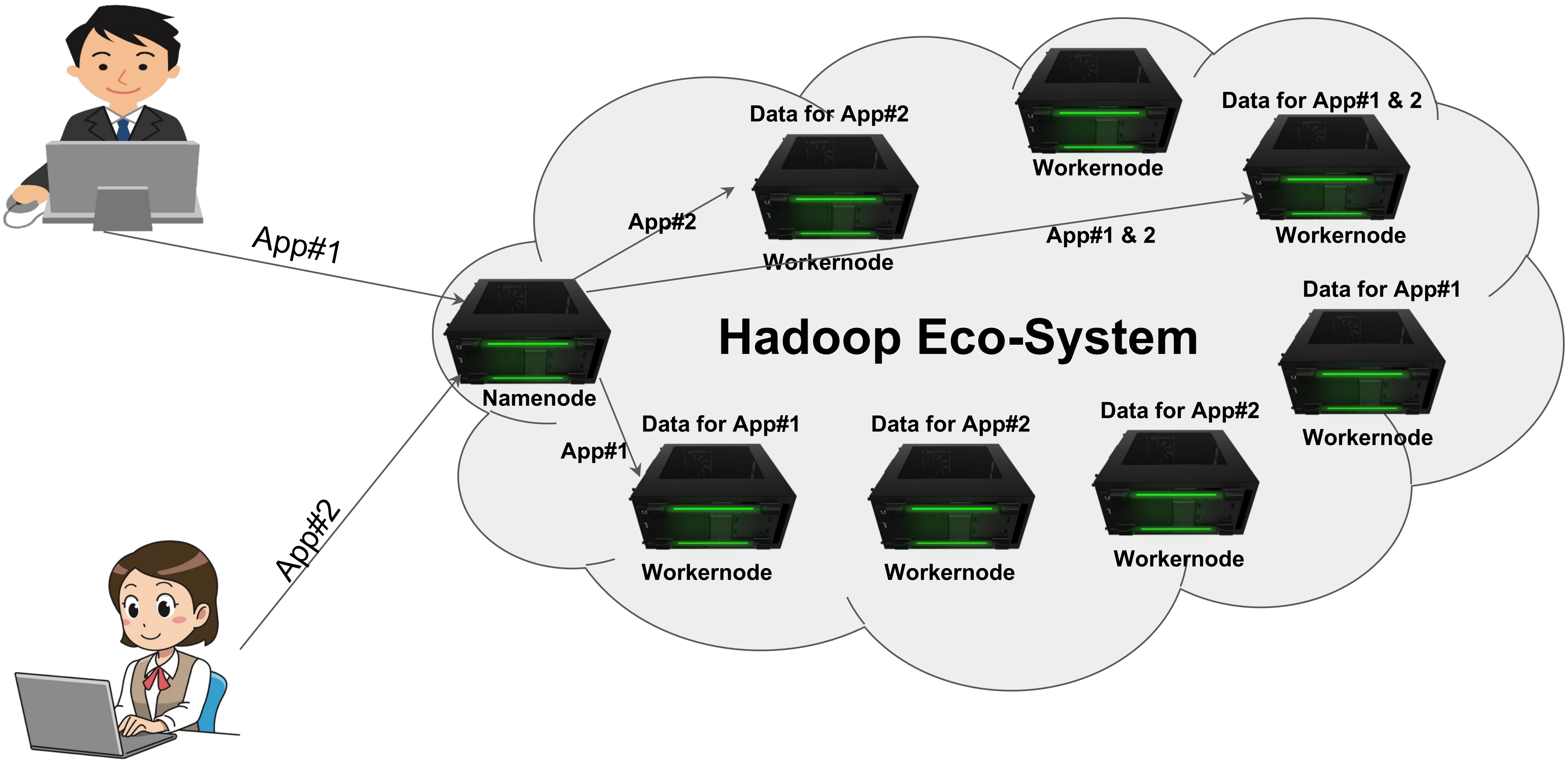

MapReduce
File System
(HDFS)
Projects
Hive
HBase
Mahout
Pig
Oozie
Flume
Scoop

User, Hadoop, YARN, HDFS

University of CINCINNATI



Application Flow in Hadoop


App#1
App#2
Namenode
App#2
Data for App#2
Workernode
Workernode
Data for App#1 & 2
App#1 & 2
Workernode
Data for App#1
Workernode
Hadoop Eco-System
App#1
Data for App#1
Workernode
Data for App#2
Workernode
Data for App#2
Workernode