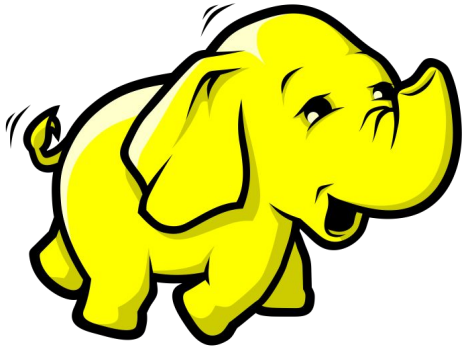


Apache Hadoop

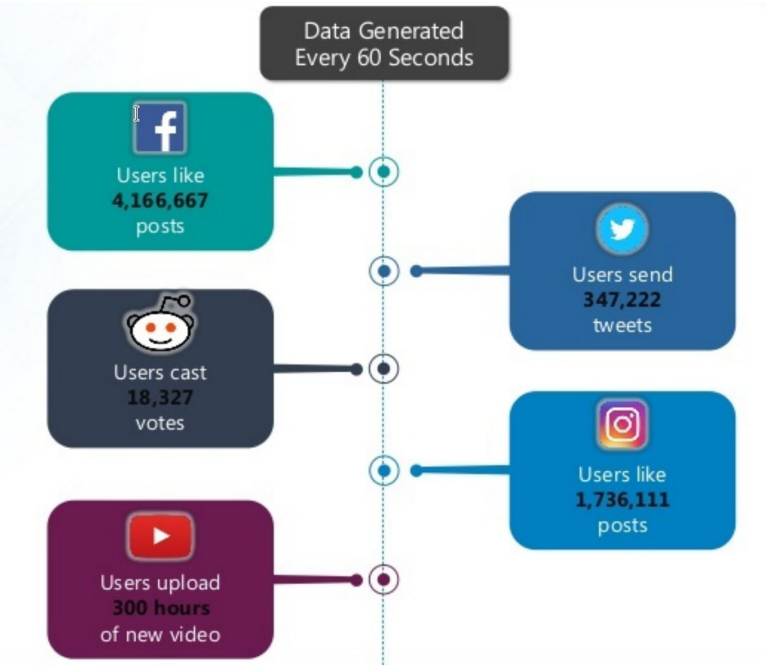
Distributed Framework for Big Data

Outline

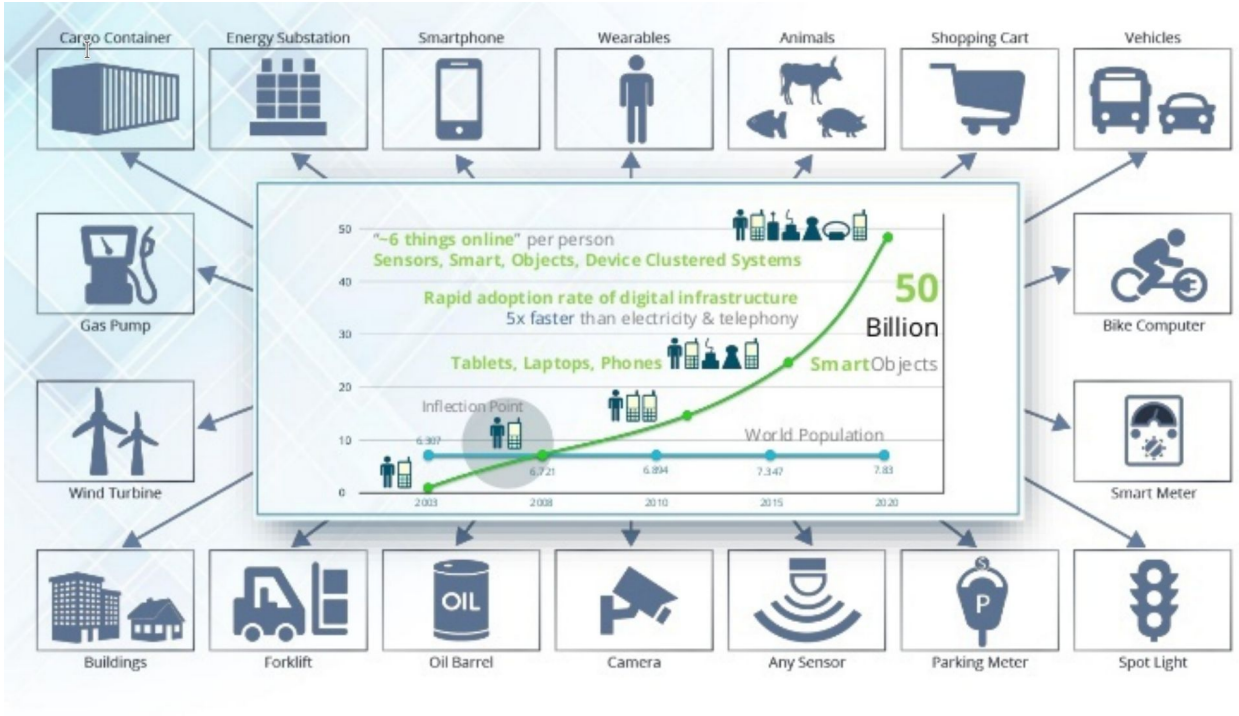
- Big data overview
- Hadoop Framework
- HDFS
- Map Reduce
- Yarn
- Apache Tez
- Zookeeper

Big Data

Big Data Growth Driver



Big Data Growth Driver

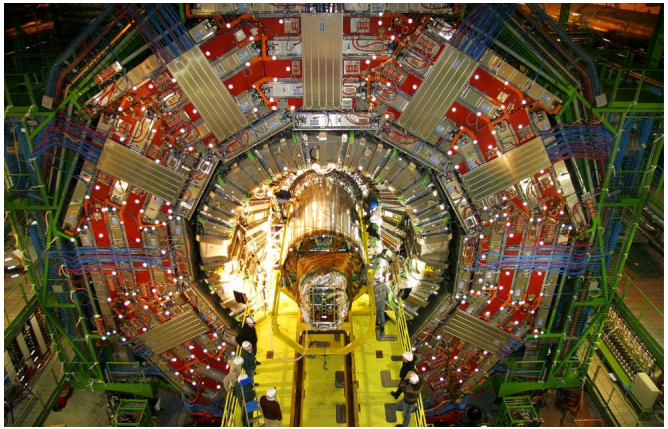


Big Data Growth Driver

The Earthscope is the world's largest science project. Designed to track North America's geological evolution, this observatory records data over 3.8 million square miles, amassing 67 terabytes of data. It analyzes seismic slips in the San Andreas fault, sure, but also the plume of magma underneath Yellowstone and much, much more.

Big Data Growth Driver

NYSE generates about one terabyte of new trade data per day to Perform stock trading analytics to determine trends for optimal trades.

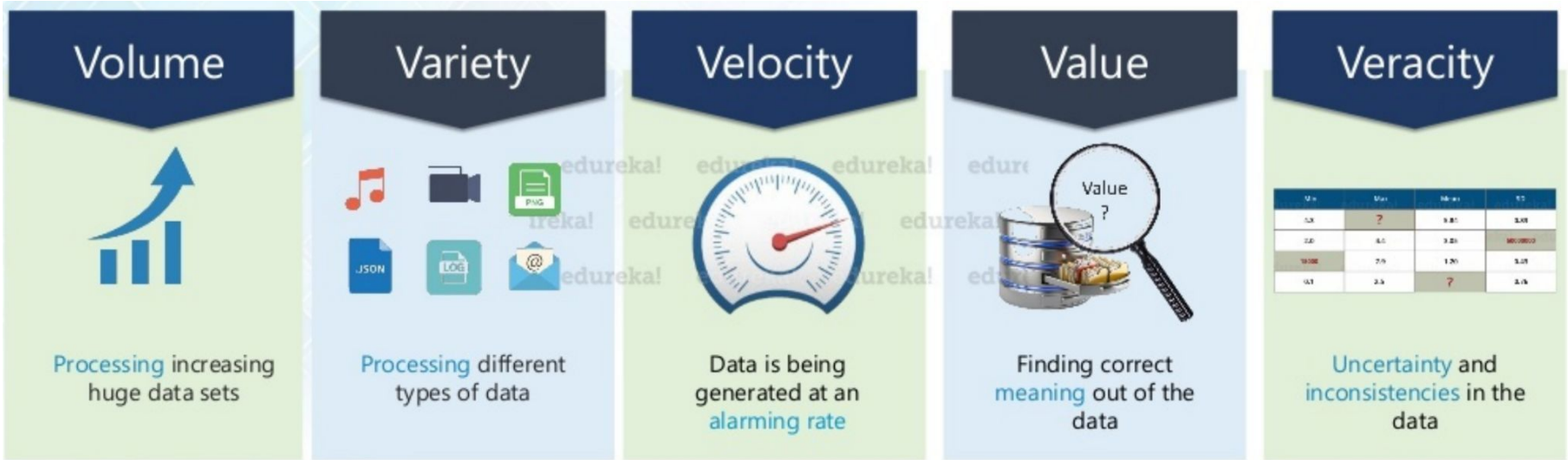


CERN’s Large Hydron Collider (LHC) generates 15 PB a year

What is Big Data

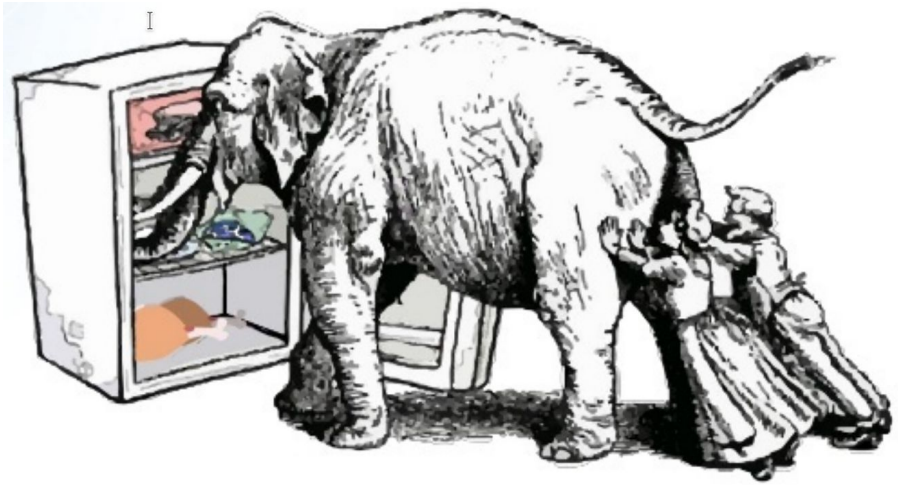
- Lots of Data (Terabytes or Petabytes)
- Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization.
- Systems / Enterprises generate huge amount of data from Terabytes to and even Petabytes of information.

5V Big Data


Volume
Processing increasing huge data sets
Variety
PNG
JSON
LOG
Processing different types of data
Velocity
Data is being generated at an alarming rate
Value
Value
?
Finding correct meaning out of the data
Veracity
Uncertainty and inconsistencies in the data

Big Data Problem

1. Storing huge and exponentially grow datasets:
    - Data generated past **2 years** is more than previous human race history in total
    - By 2020, Total data digital will grow **44 zetabytes** or **44 trillion Gigabytes**
    - By 2020, about 1.7 MB of new info will be created every second per person



Big Data Problem

2. Processing data have complex structures

Structured
- Organized data format
- Data schema is fixed
- Ex: RDBMS Data, etc

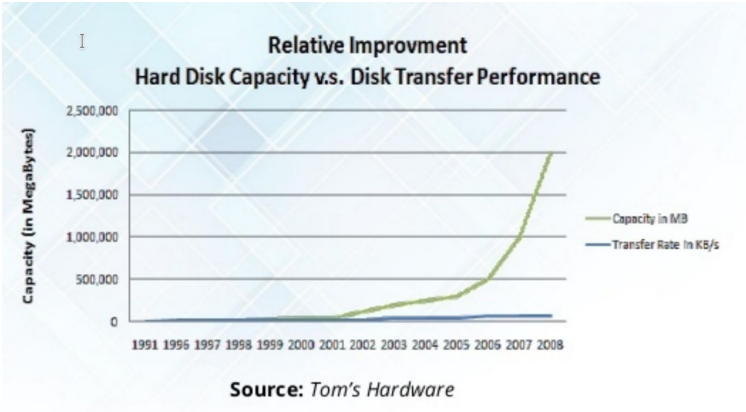
Semi-Structured
- Partial Organized
- Lack formal structure like data model
- Ex: XML, JSON files etc

Unstructured
- Un-organized data
- Unknown Schema
- Ex: Multimedia files, etc

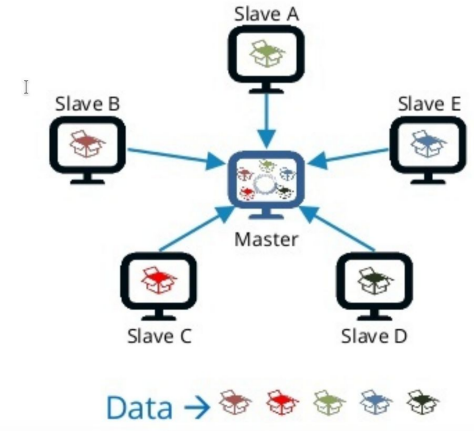
Big Data Problem

3. Processing data faster:

- Data is growing faster rate than of disk read/write



- Bringing huge amount data to computation unit become bottleneck



Big Data Problem

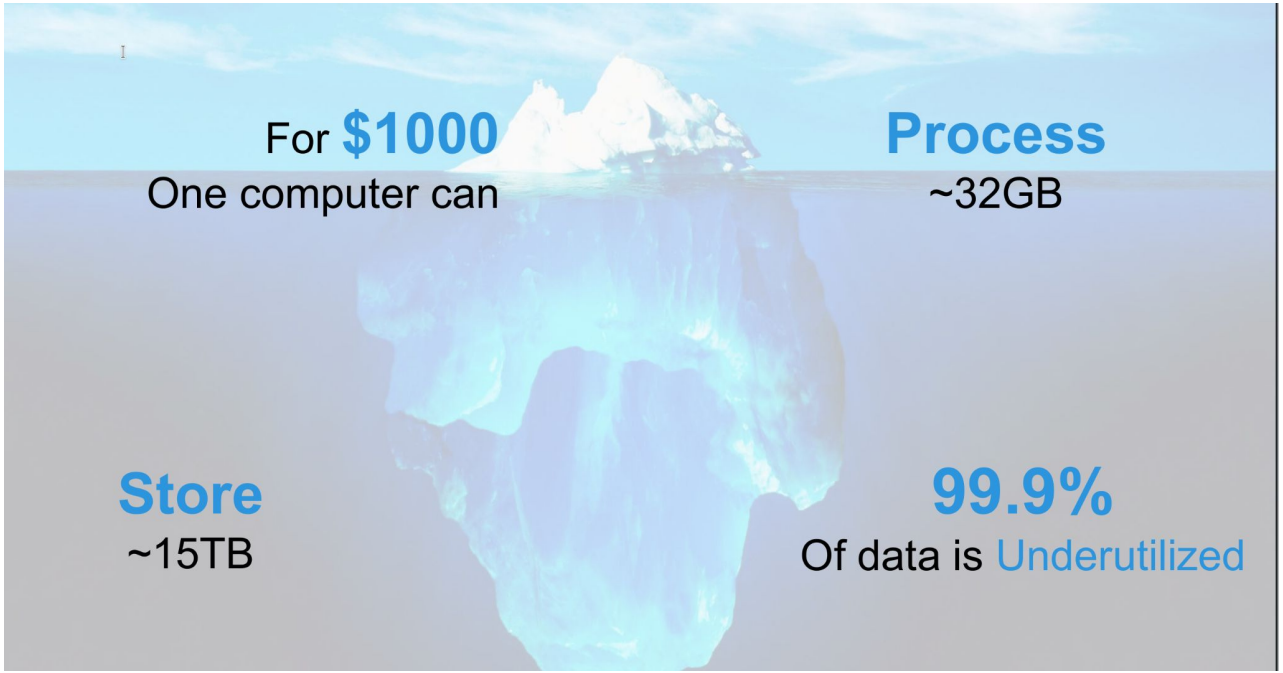
4. Storage & Memory B/W lagging CPU:

	CPU	DRAM	LAN	Disk
Annual bandwidth improvement (all milestones)	1.5	1.27	1.39	1.28
Annual latency improvement (all milestones)	1.17	1.07	1.12	1.11

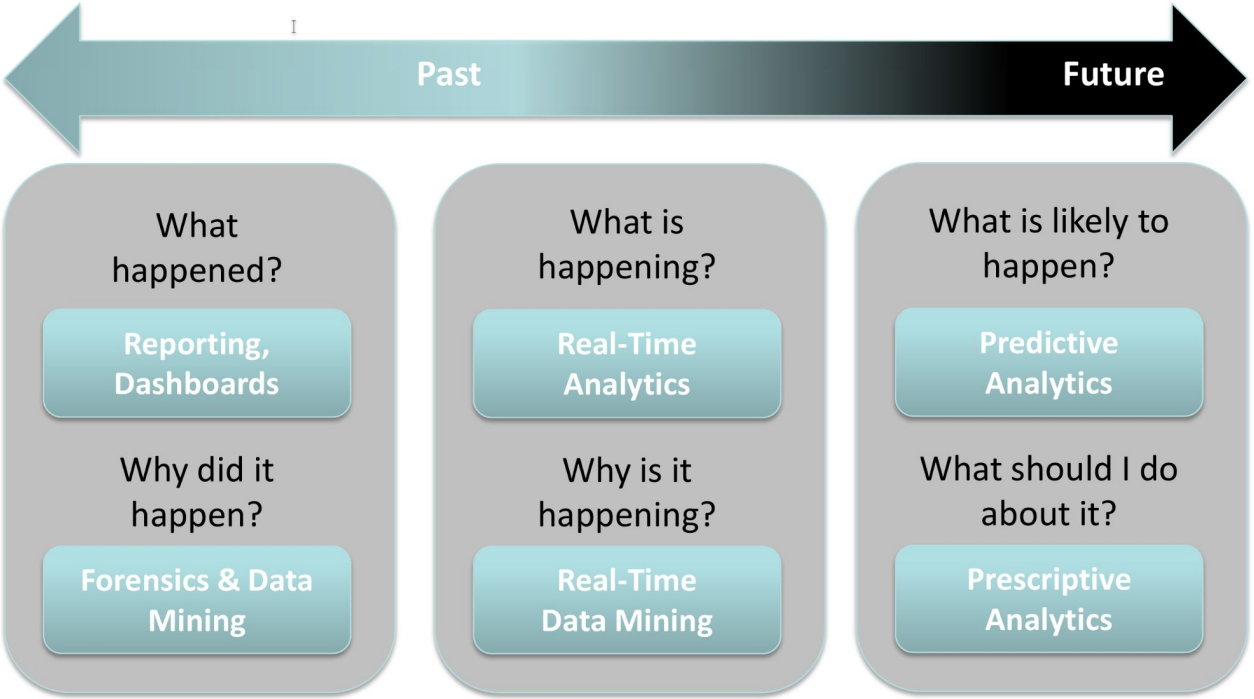
- CPU B/W requirements out-pacing memory and storage
- Disk & memory getting “further” away from CPU
- Large sequential transfers better for both memory & disk

Big Data Problem

5. Commodity Hardware Economics



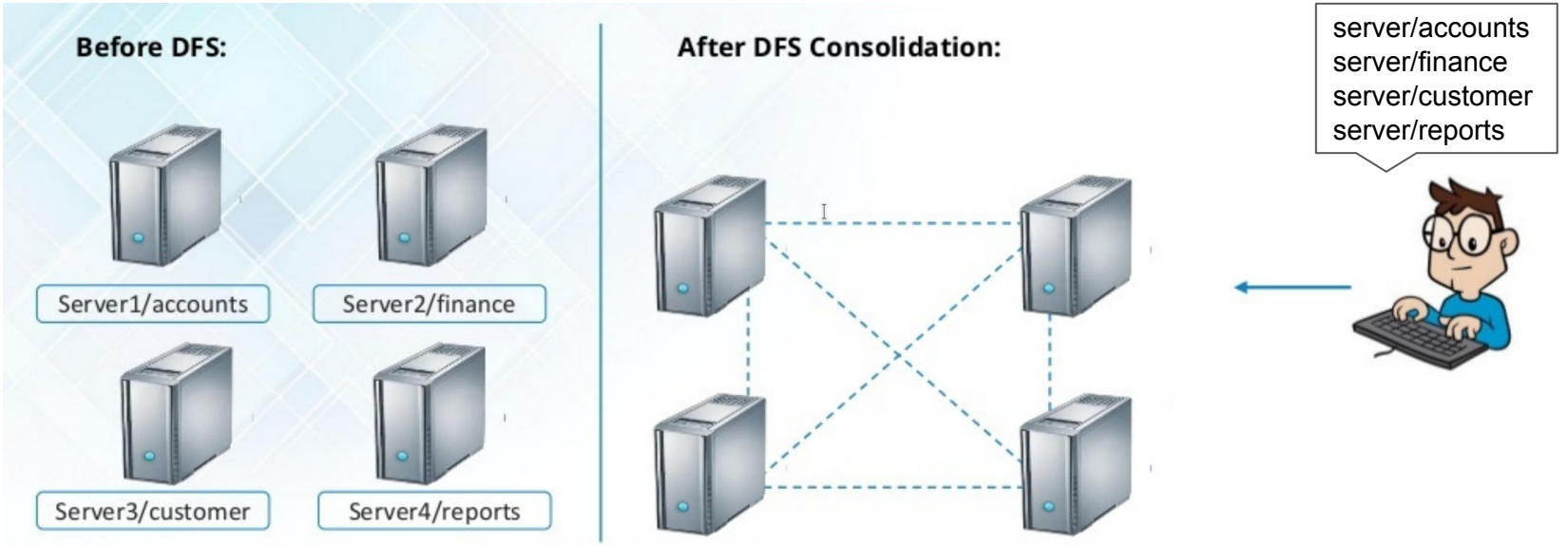
Questions from Businesses will Vary



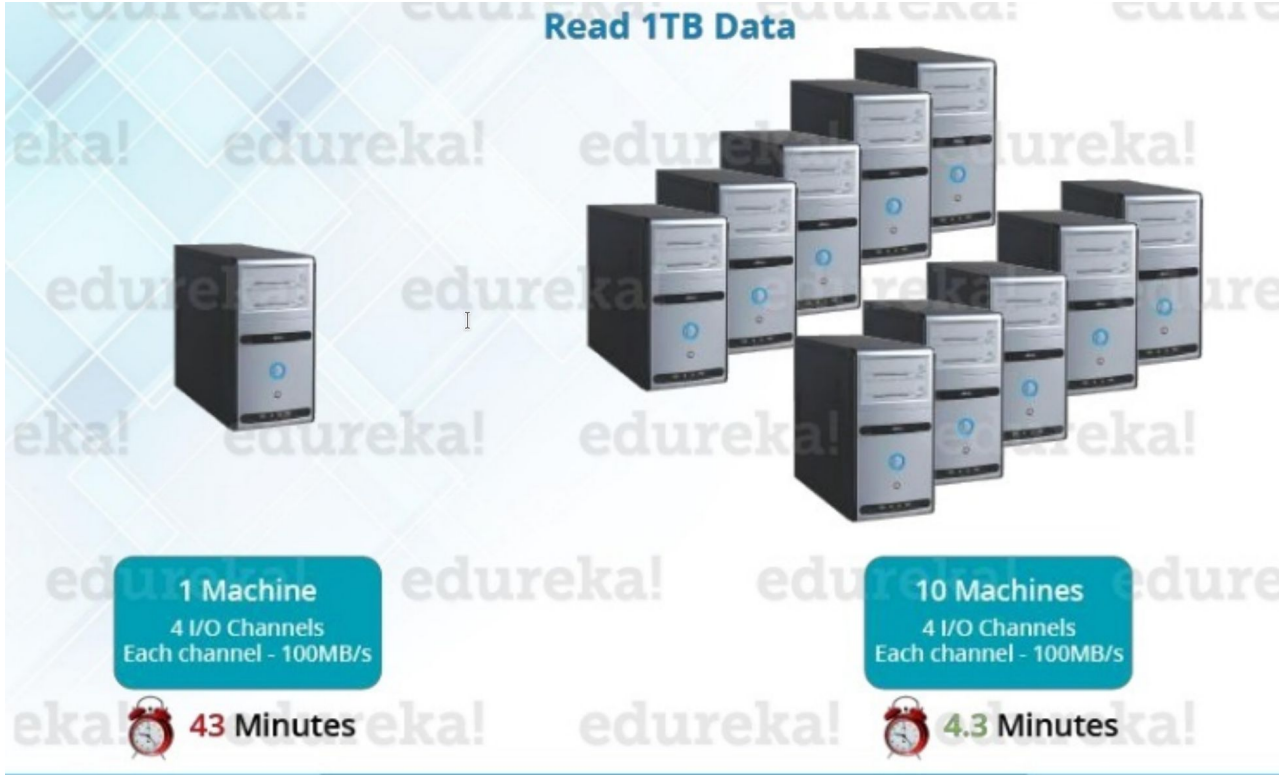
Before Moving to Big Data Solution Lets Understand What is **DFS**?

DFS - Distributed File System

- Storing and managing data (files and folders) across multiple computer/servers
- Provide abstraction of single large file system



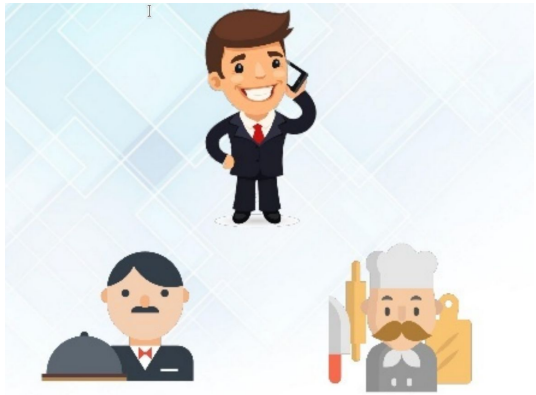
Why DFS?


Read 1TB Data
1 Machine
4 I/O Channels
Each channel - 100MB/s
43 Minutes
10 Machines
4 I/O Channels
Each channel - 100MB/s
4.3 Minutes

Big Data &
Traditional System

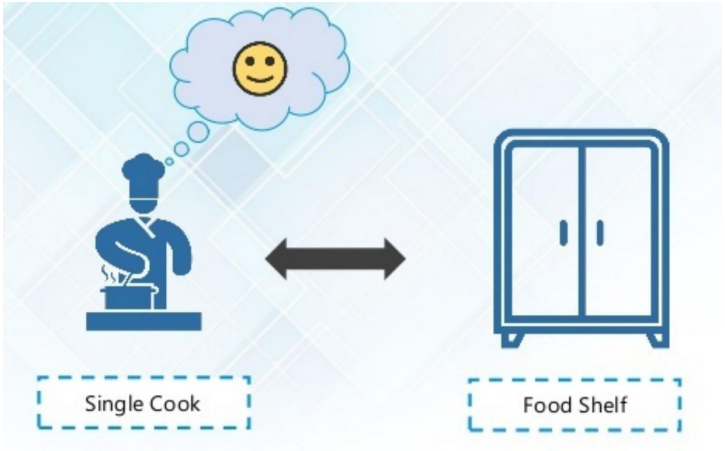
Story of Big Data and Traditional System

Bob has opened small restaurant
in the city

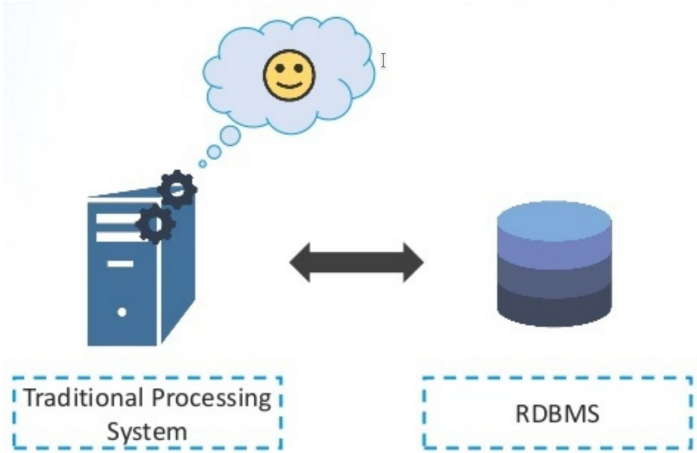


Story of Big Data and Traditional System

Traditional Scenario:
2 Orders per hour

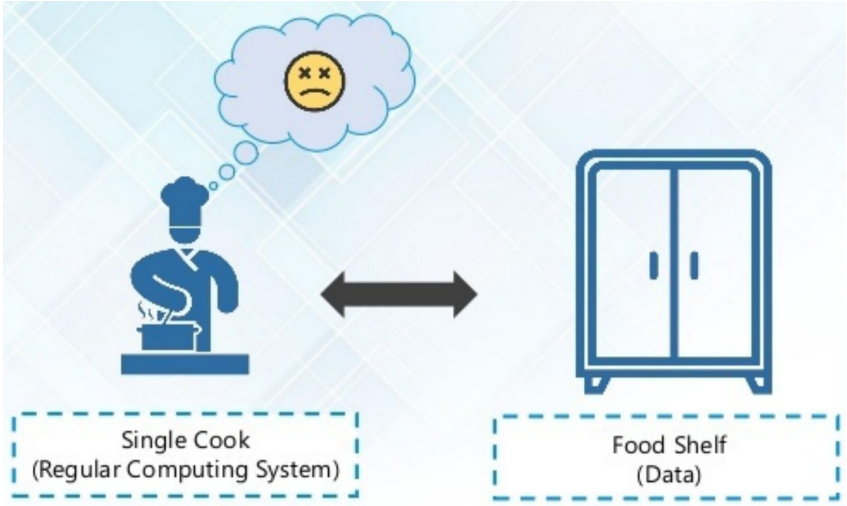


Traditional Scenario:
Data is generated in steady rate and in structured in nature

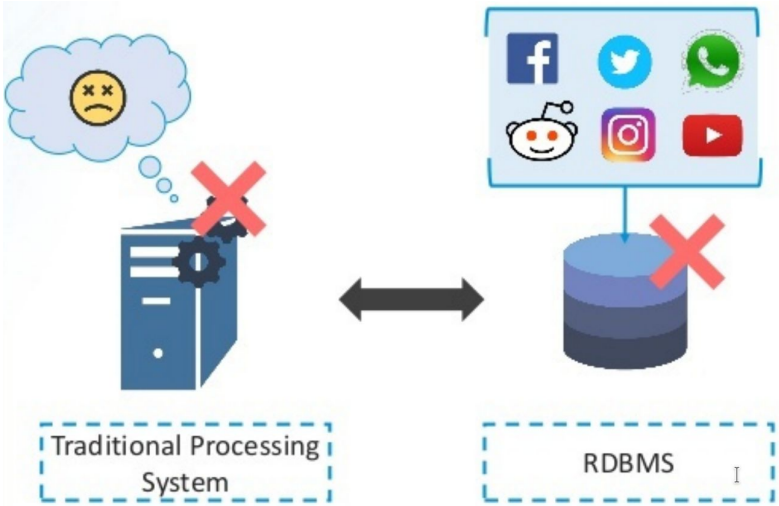


Story of Big Data and Traditional System

Traditional Scenario:
- 10 orders per hour
- Multiple menu introduced



Traditional Scenario:
Heterogenous data being generated at alarming rate

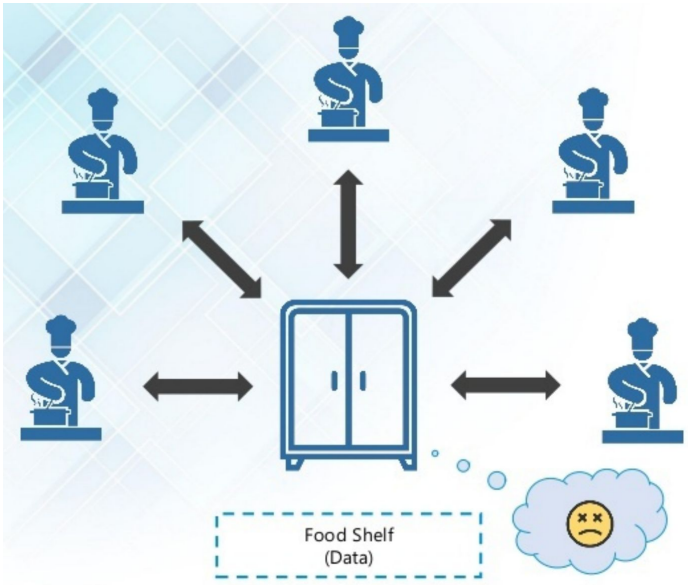


Issue 1. Too many orders

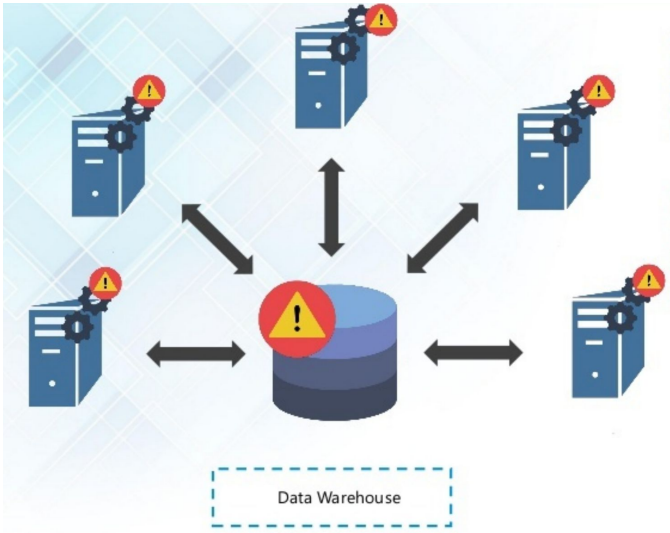
Solution: Hiring more cooks

Story of Big Data and Traditional System

Multiple Cooks cooking food:
Food shelf become the **bottleneck**



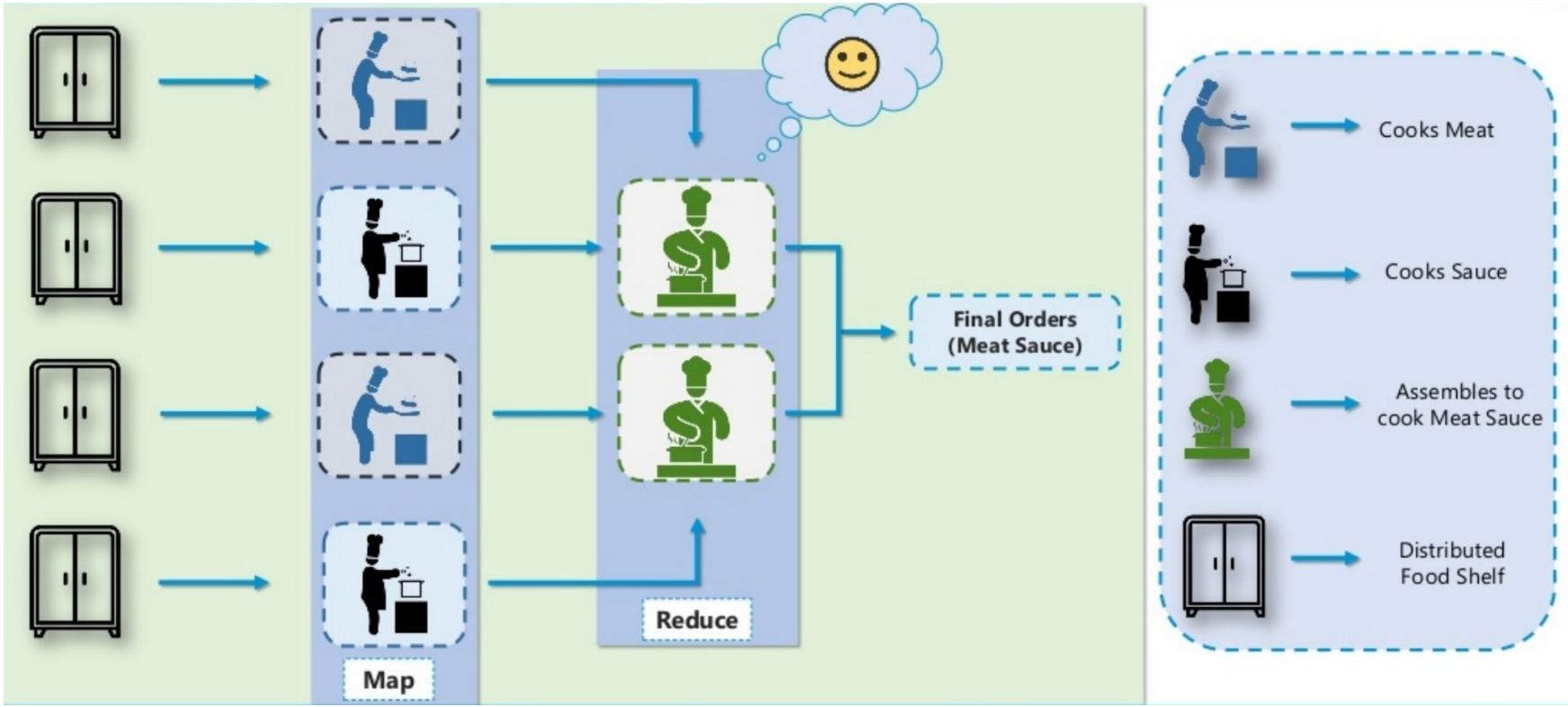
Multiple Processing Unit:
Bringing data to processing unit generated lot of Network overhead



Issue 2. Food shelf becomes **bottleneck**

Solution: Distributed and parallel approach

Distributed and parallel approach

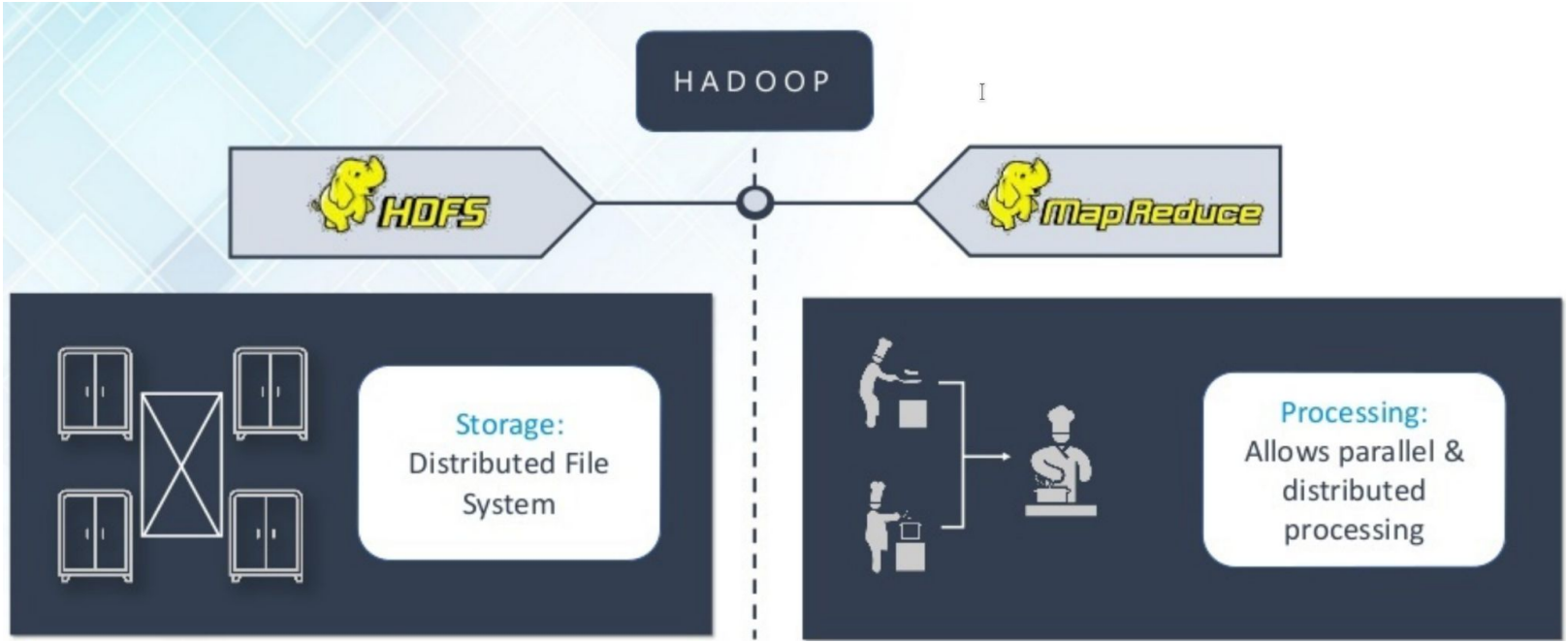


Apache Hadoop

Framework to process big data

Apache Hadoop

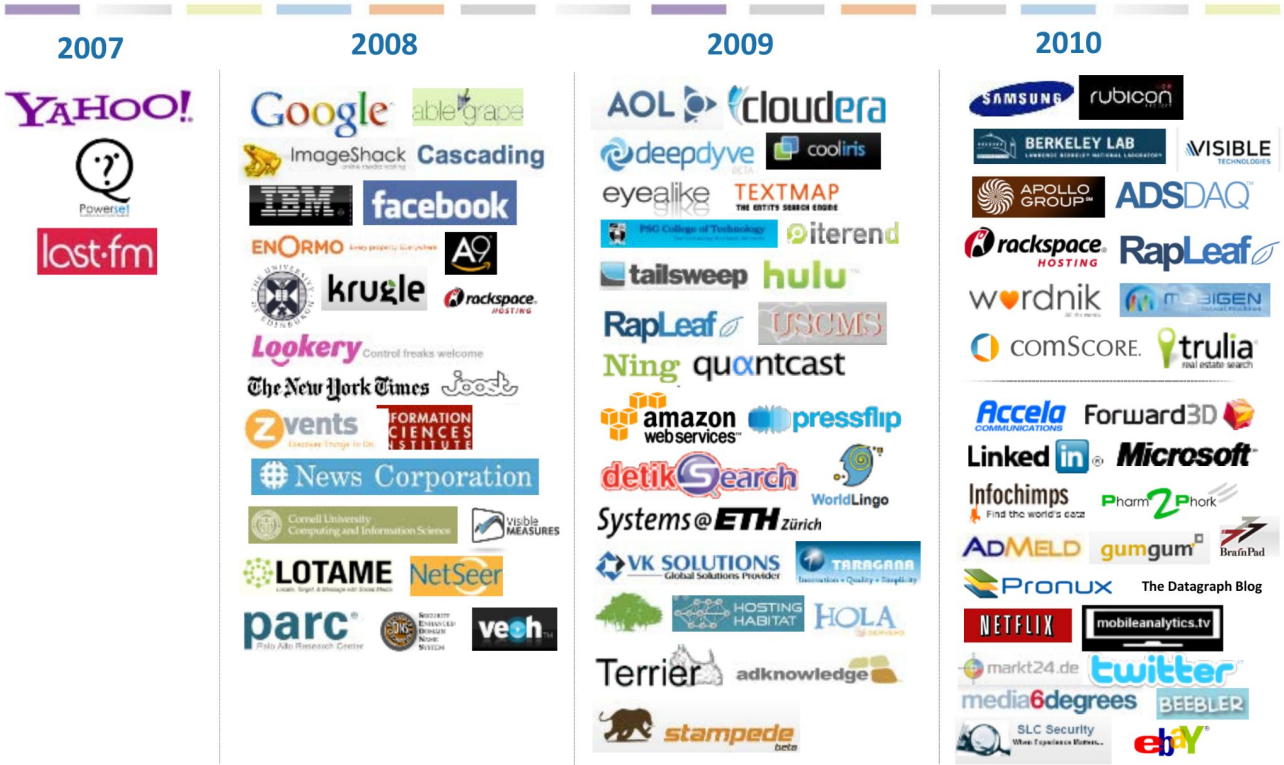
Hadoop is framework that allow us to store and process large data in parallel and distributed fashion



Apache Hadoop

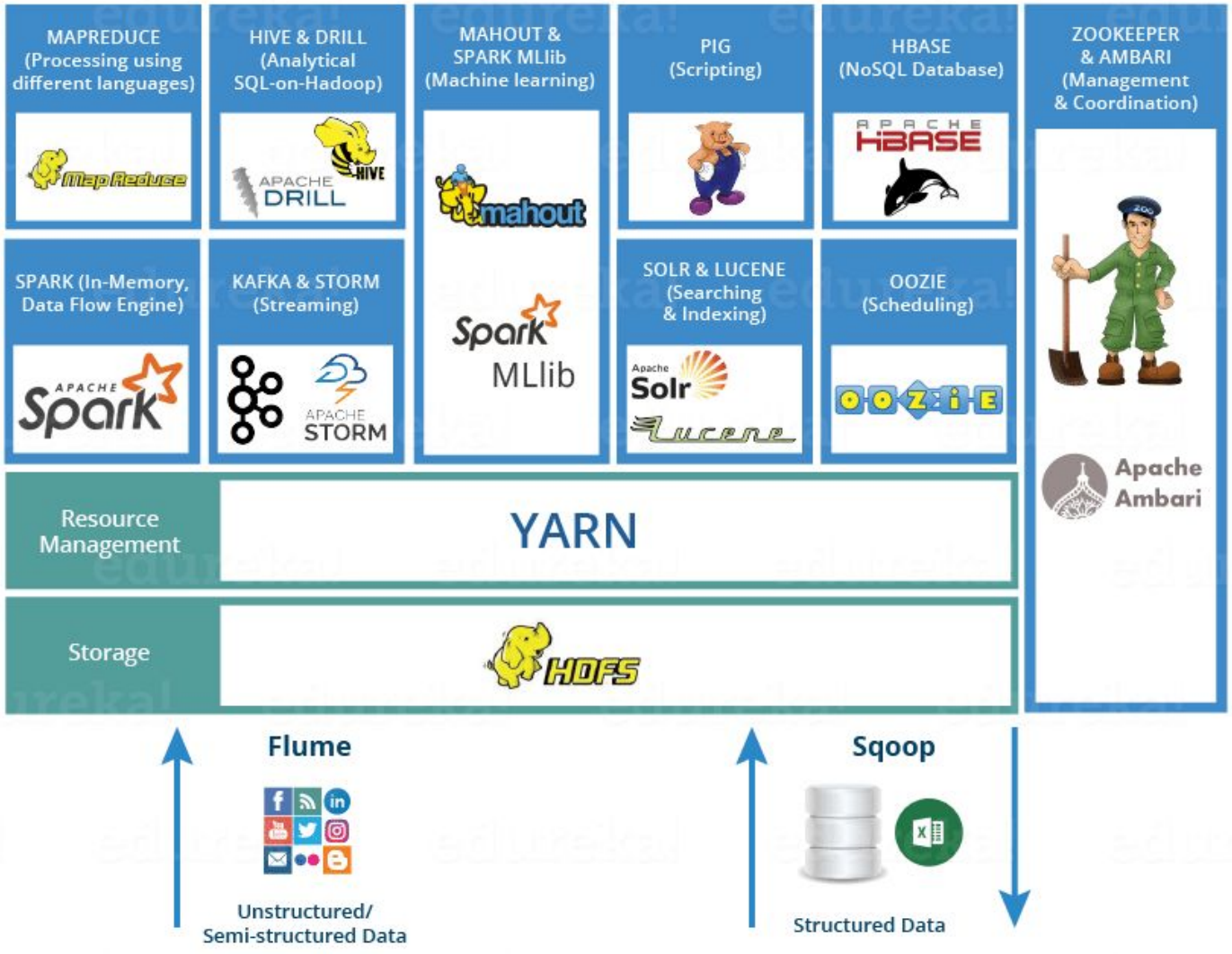
- A scalable fault-tolerant distributed system for data storage and processing
- Core Hadoop has two main components
  - Hadoop Distributed File System (HDFS):
    - self-healing, high-bandwidth clustered storage Reliable, redundant, distributed file system optimized for large files
  - MapReduce: fault-tolerant distributed processing Programming model
    - for processing sets of data
    - Mapping inputs to outputs and reducing the output of multiple Mappers to one (or a few) answer(s)
- Operates on unstructured and structured data
- A large and active ecosystem
- Open source under the friendly Apache License http://wiki.apache.org/hadoop/

Hadoop Adoption in Industry



Source: Hadoop Summit Presentations

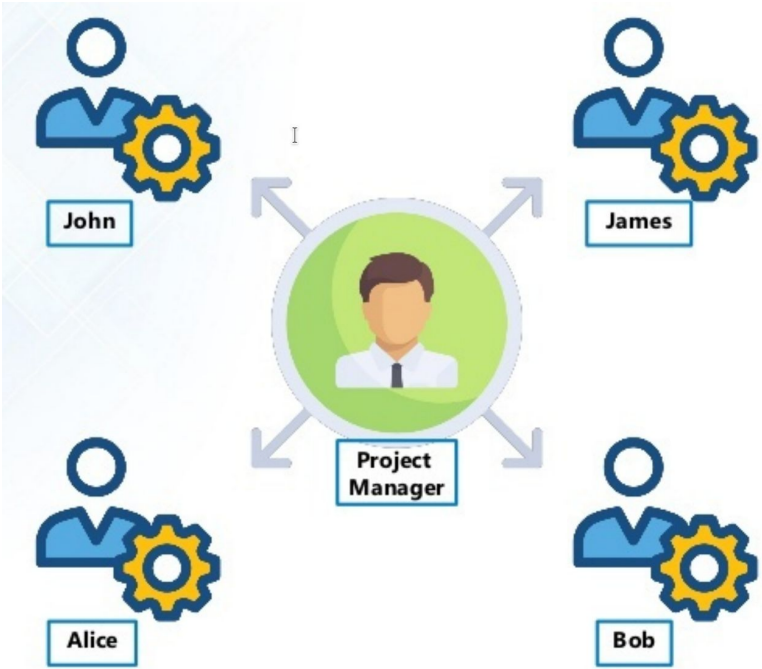
Hadoop Ecosystem



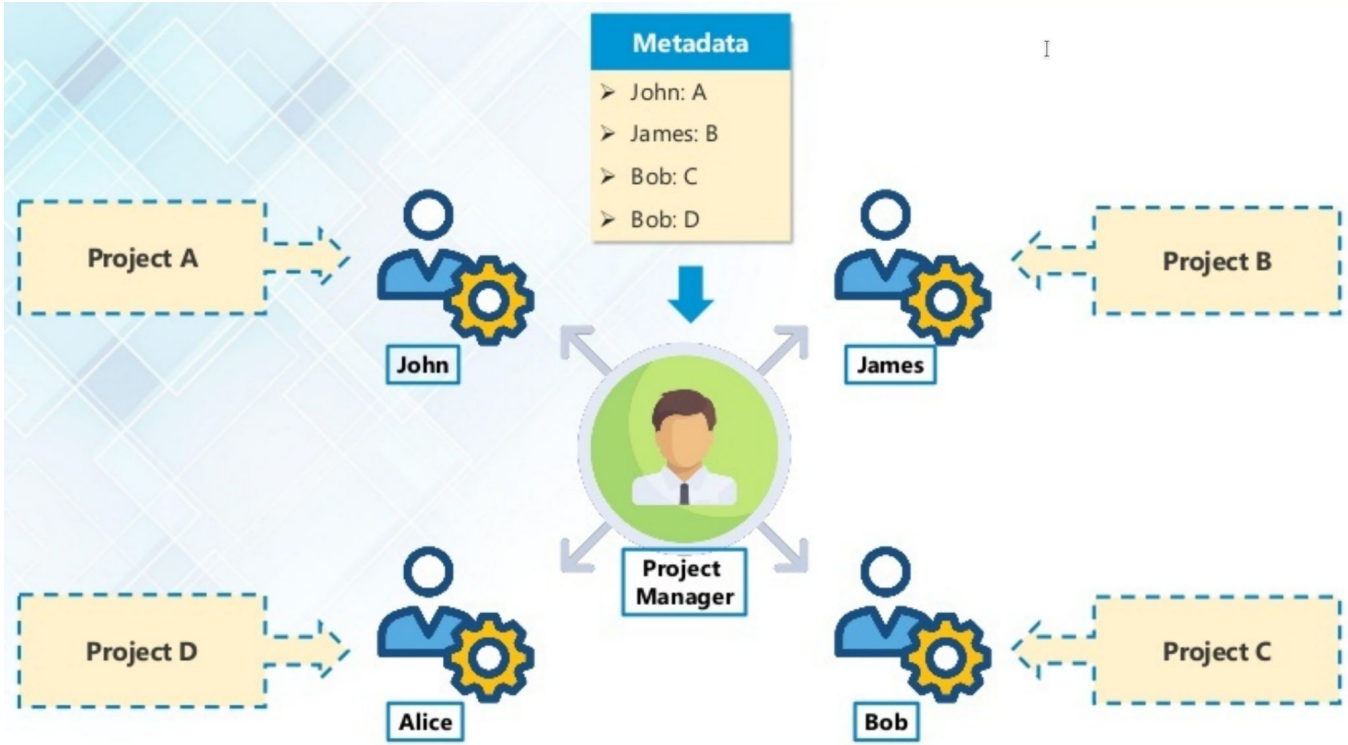
Hadoop: Master/Slave Architecture

Scenario:

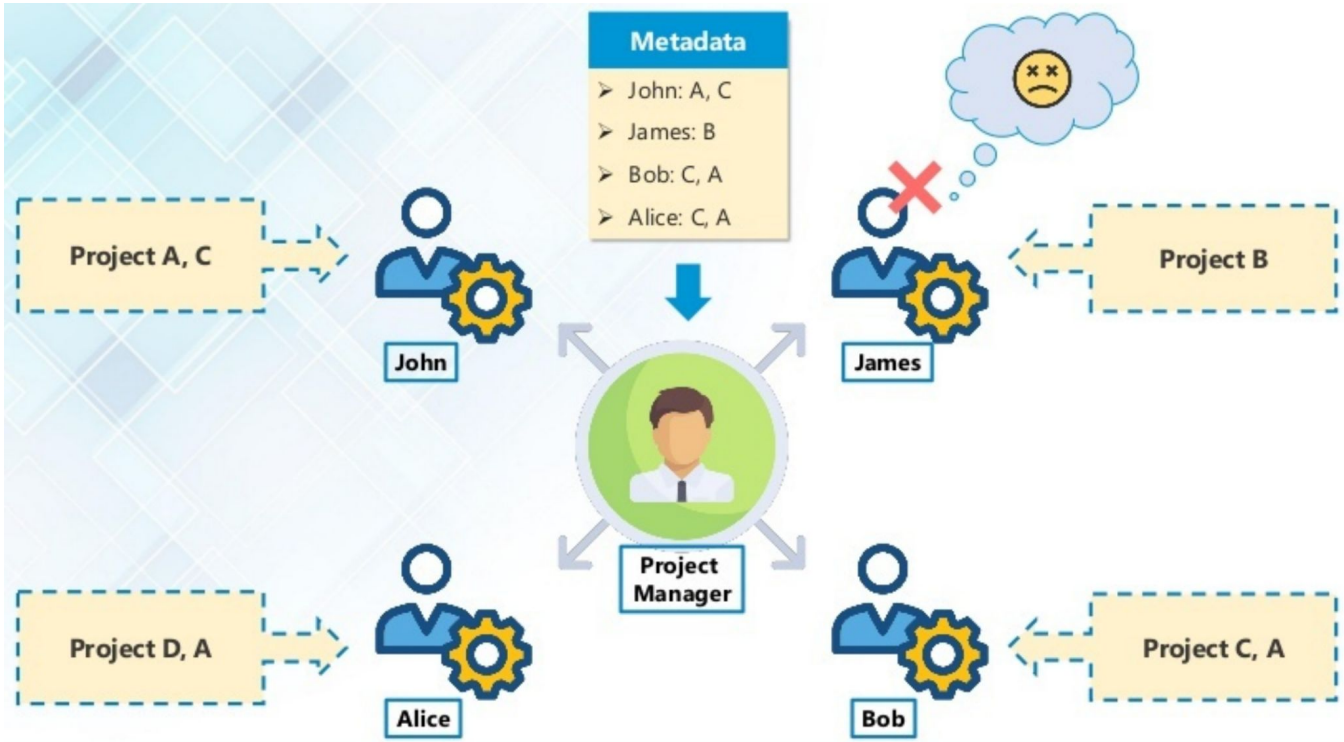
A project manager managing team of four employees. He assign project to each of them then track the progress



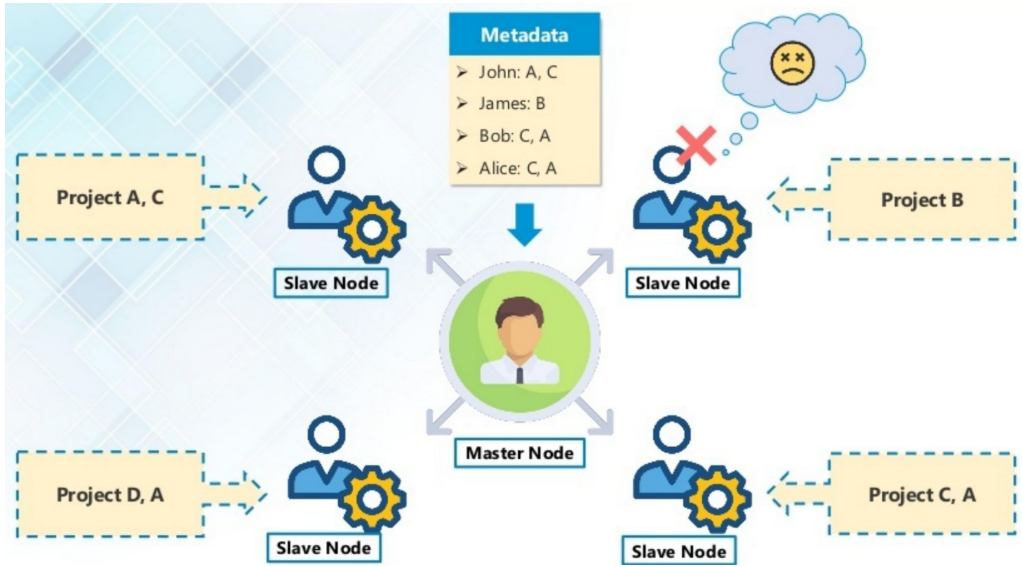
Hadoop: Master/Slave Architecture

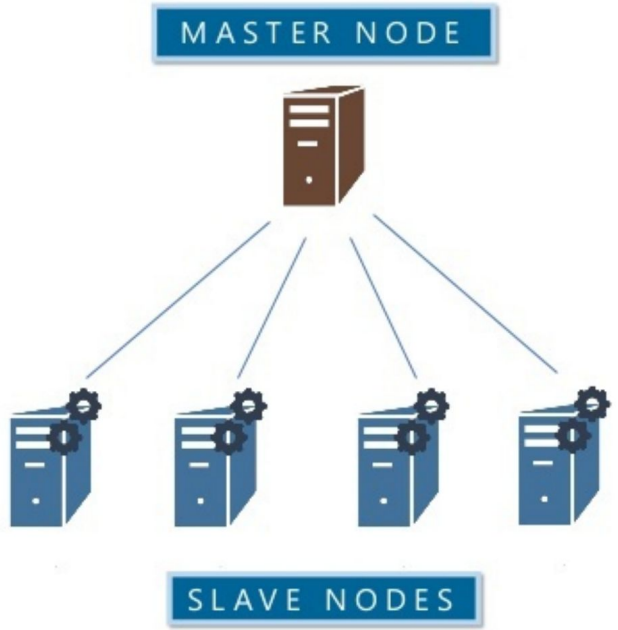


Hadoop: Master/Slave Architecture

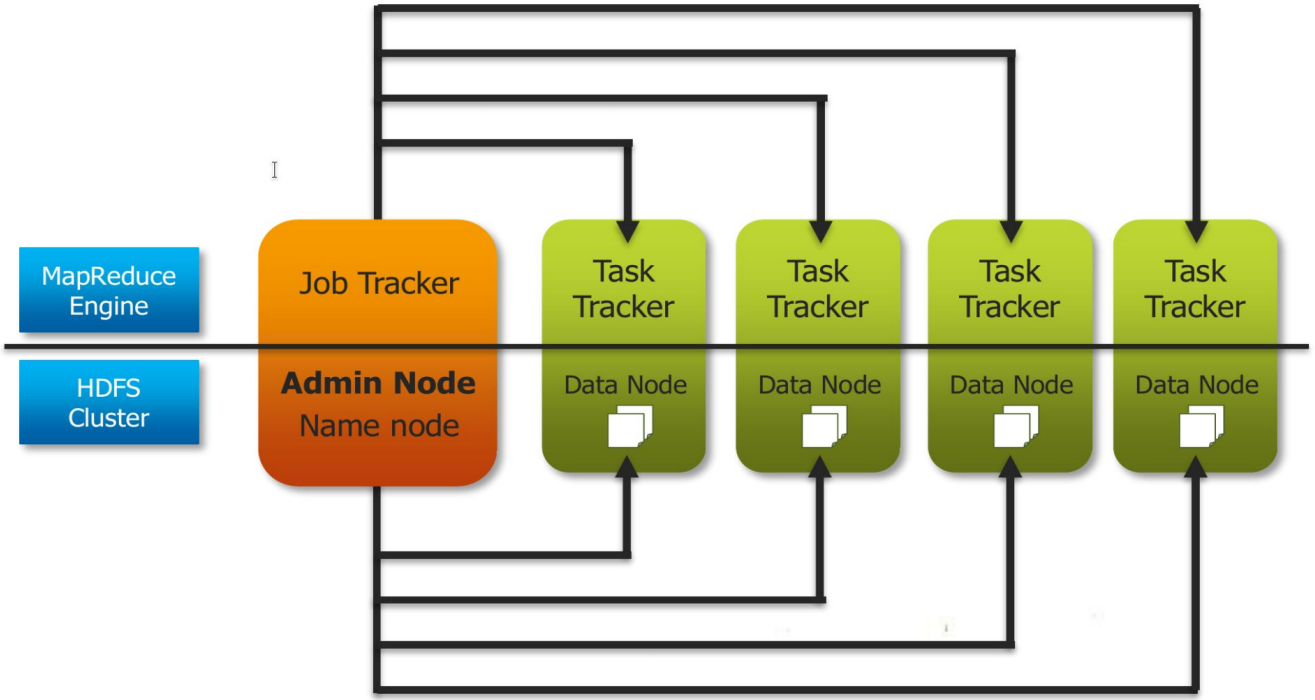


Hadoop: Master/Slave Architecture





Hadoop Architecture



Demo

- Show hadoop interface
- Show hadoop configuration
- Add new worker node
- Shutdown worker node

Quiz

- What is **5V of Big Data** and explain!
- Explain terms **“Bringing data to processing unit generated lot of Network overhead”**!
- What is **DFS** dan what the difference with previous approach?
- What is **two main component of Hadoop** and explain it?
- Explain briefly how hadoop could handle jobs in **distributed and parallel** way!

HDFS

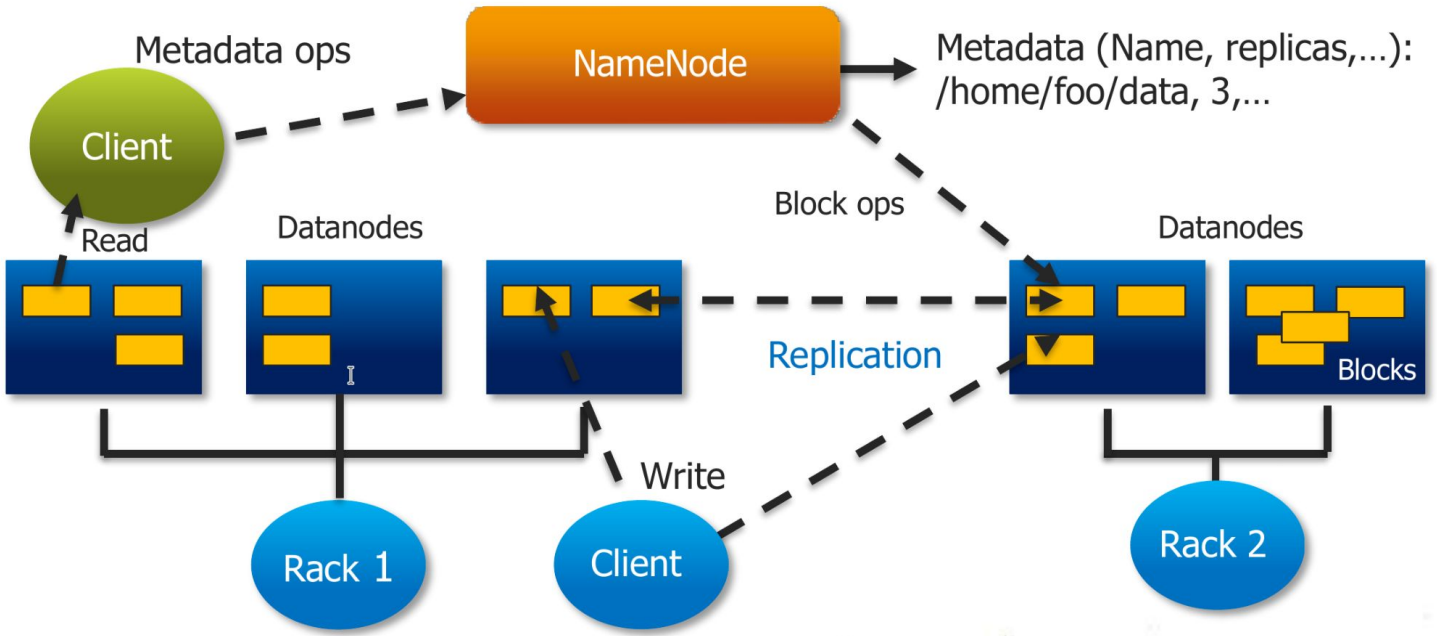
Hadoop Distributed File System

HDFS

Store large datasets in a distributed, scalable and fault-tolerant way:

- High throughput
- Very large files
- Streaming reads and writes (no edits)
- Write once, read many times

HDFS Architecture



HDFS Architecture

NameNode (Master):

master of the system maintains and manages the blocks which are present on the DataNodes

DataNodes (Slaves):

slaves which are deployed on each machine and provide the actual storage responsible for serving read and write requests for the clients

Name Node Metadata

Meta-data in Memory

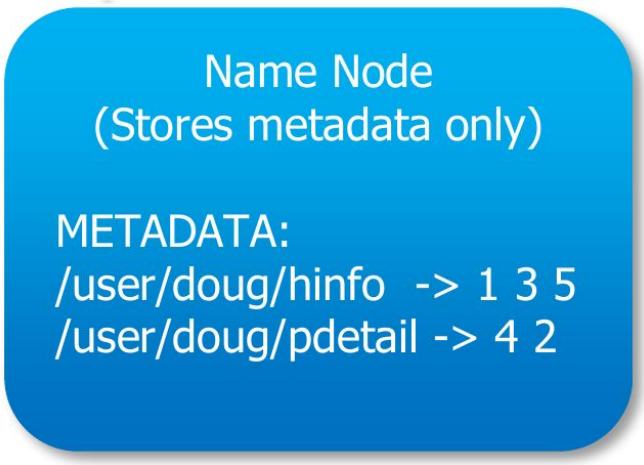
- The entire metadata is in main memory
- No demand paging of FS meta-data

Types of Metadata

- List of files
- List of Blocks for each file
- List of DataNode for each block
- File attributes, e.g. access time, replication factor

A Transaction Log

- Records file creations, file deletions. etc

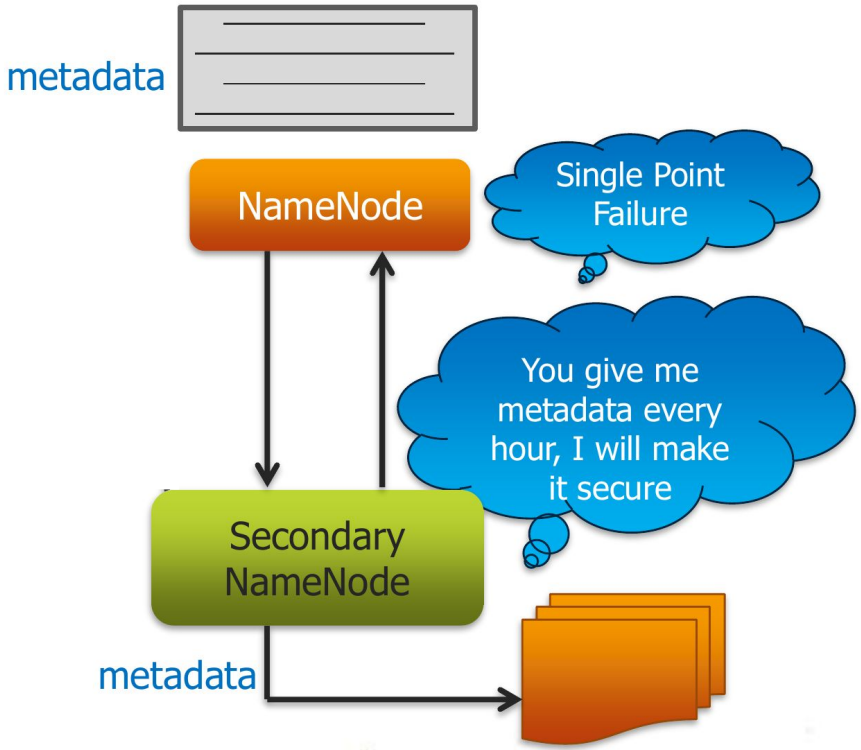


Name Node:

Keeps track of overall file directory structure and the placement of Data Block

Secondary Name Node

- Not a hot standby for the NameNode
- Connects to NameNode every hour*
- Housekeeping, backup of NameNode metadata
- Saved metadata can build a failed NameNode



HDFS Blocks

- Each files stores as blocks
- Default size of each blocks is 128 MB in Hadoop 2.x (64 MB in Hadoop 1.x)
- Let say, I have file example.txt with size 248 MB, below is how data will be stored in HDFS

What if file size is 600MB?

HDFS Blocks

- Minimize the overhead of a disk seek operation (less than 1%)
- A file is just “sliced” into chunks after each 128MB (or so)
- It does NOT matter whether it is text/binary, compressed or not
- It does matter later (when reading the data)

What if file size is 600MB?

HDFS Replication Blocks

The default replication factor is 3.

It can be changed per a file or a directory.

It can be increased for “hot” datasets (temporarily or permanently)

Trade-off between

Reliability, availability, performance Disk space

HDFS Block Placement

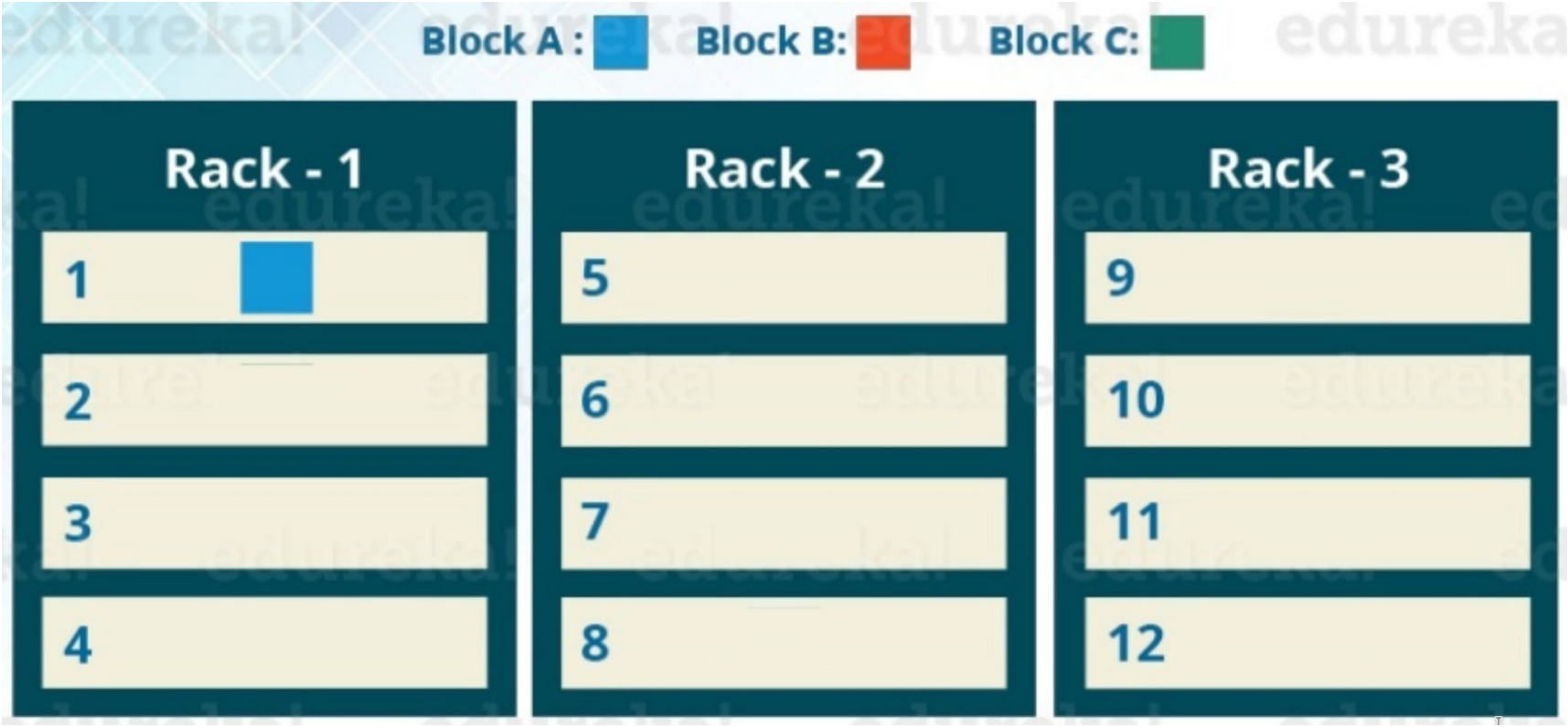
Pluggable (default in `BlockPlacementPolicyDefault.java`)

1st replica on the same node where a writer is located Otherwise “random” (but not too “busy” or almost full) node is used 2nd and the 3rd replicas on two different nodes in a different rack

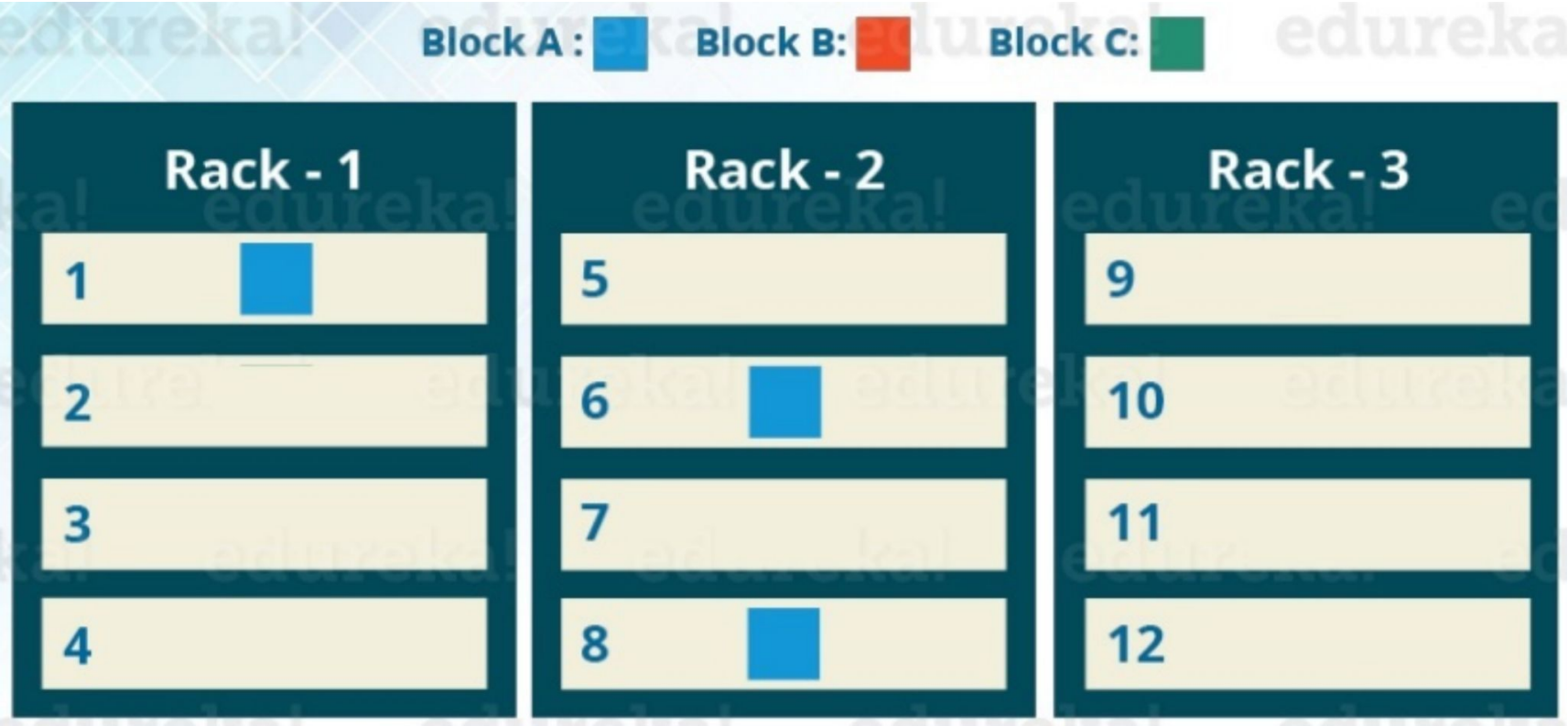
The rest are placed on random nodes. No DataNode with more than one replica of any block. No rack with more than two replicas of the same block (if possible)

HDFS Rack Awareness

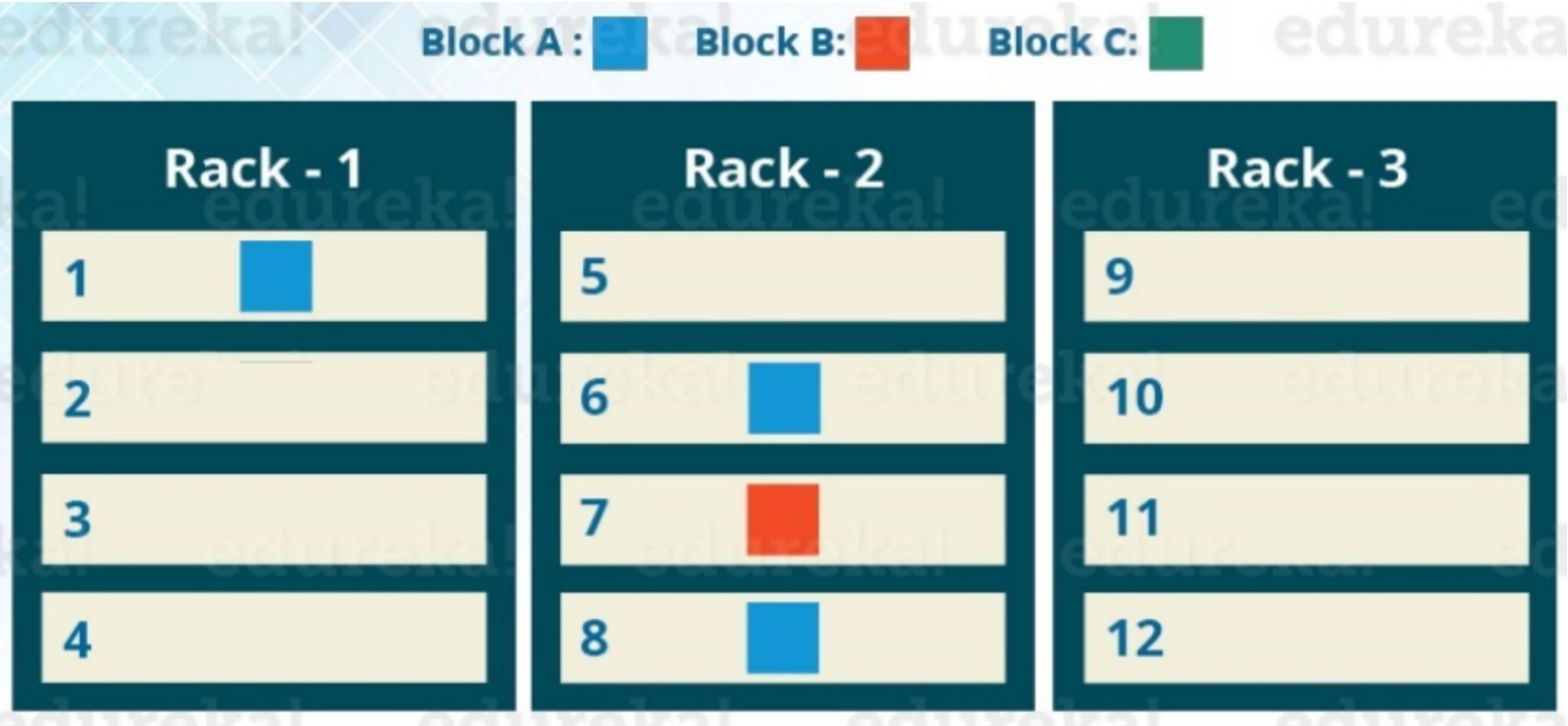
HDFS Rack Awareness



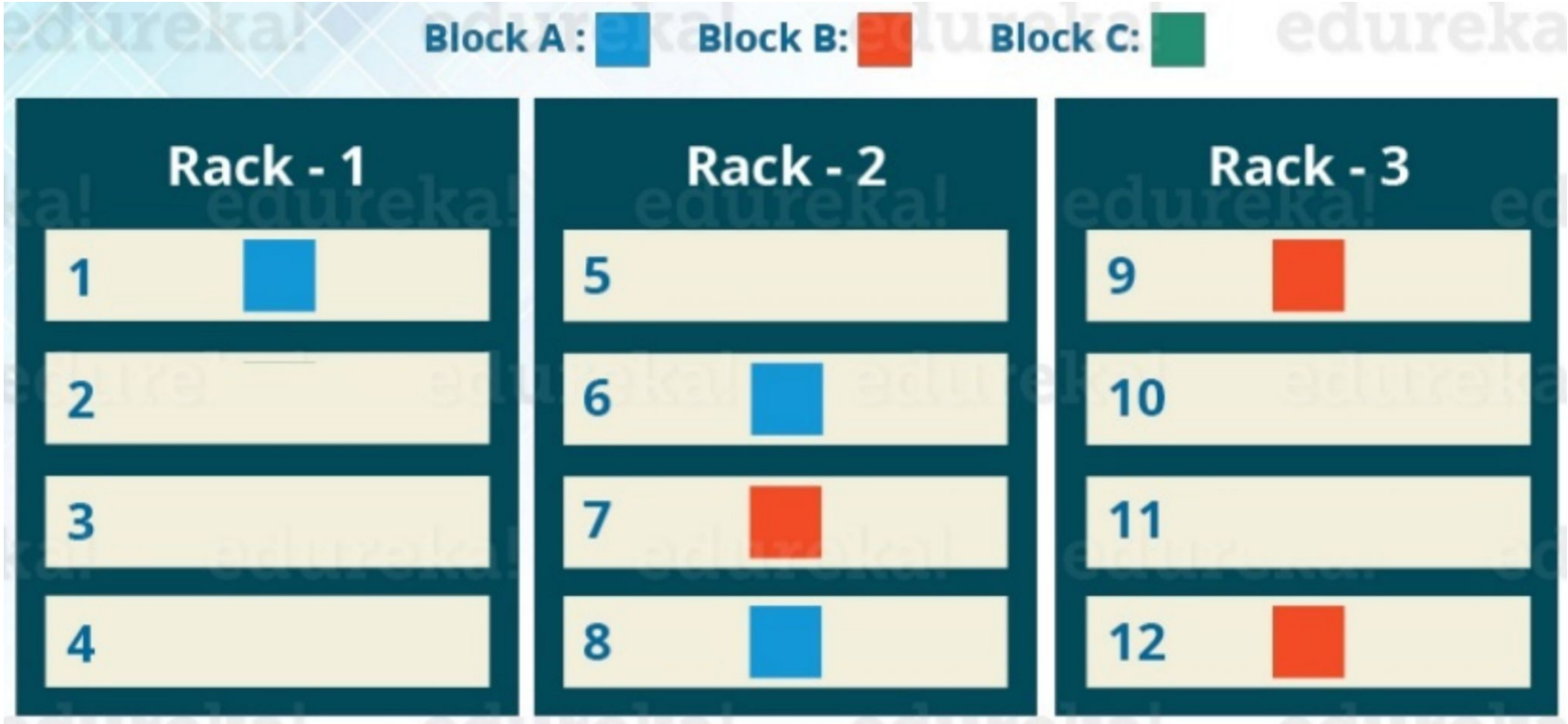
HDFS Rack Awareness



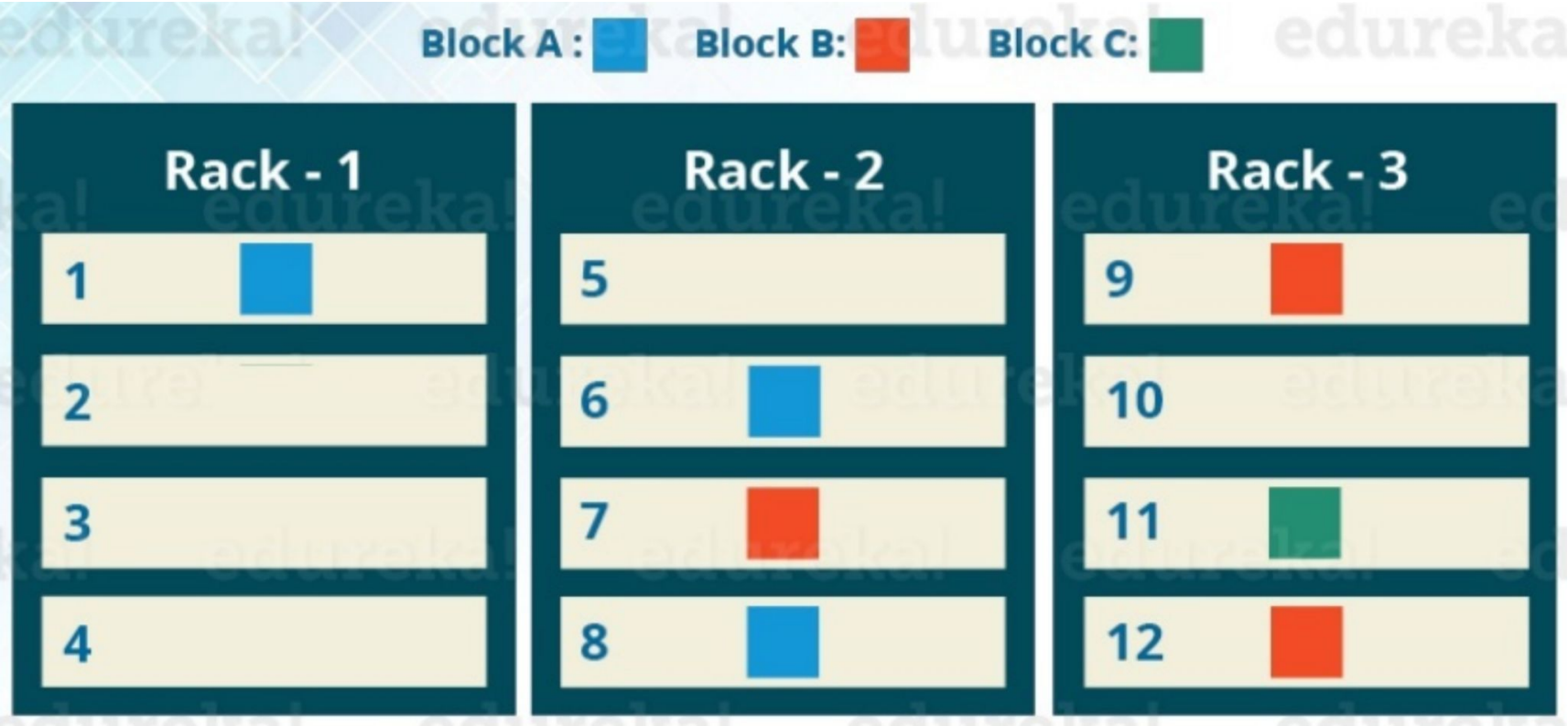
HDFS Rack Awareness



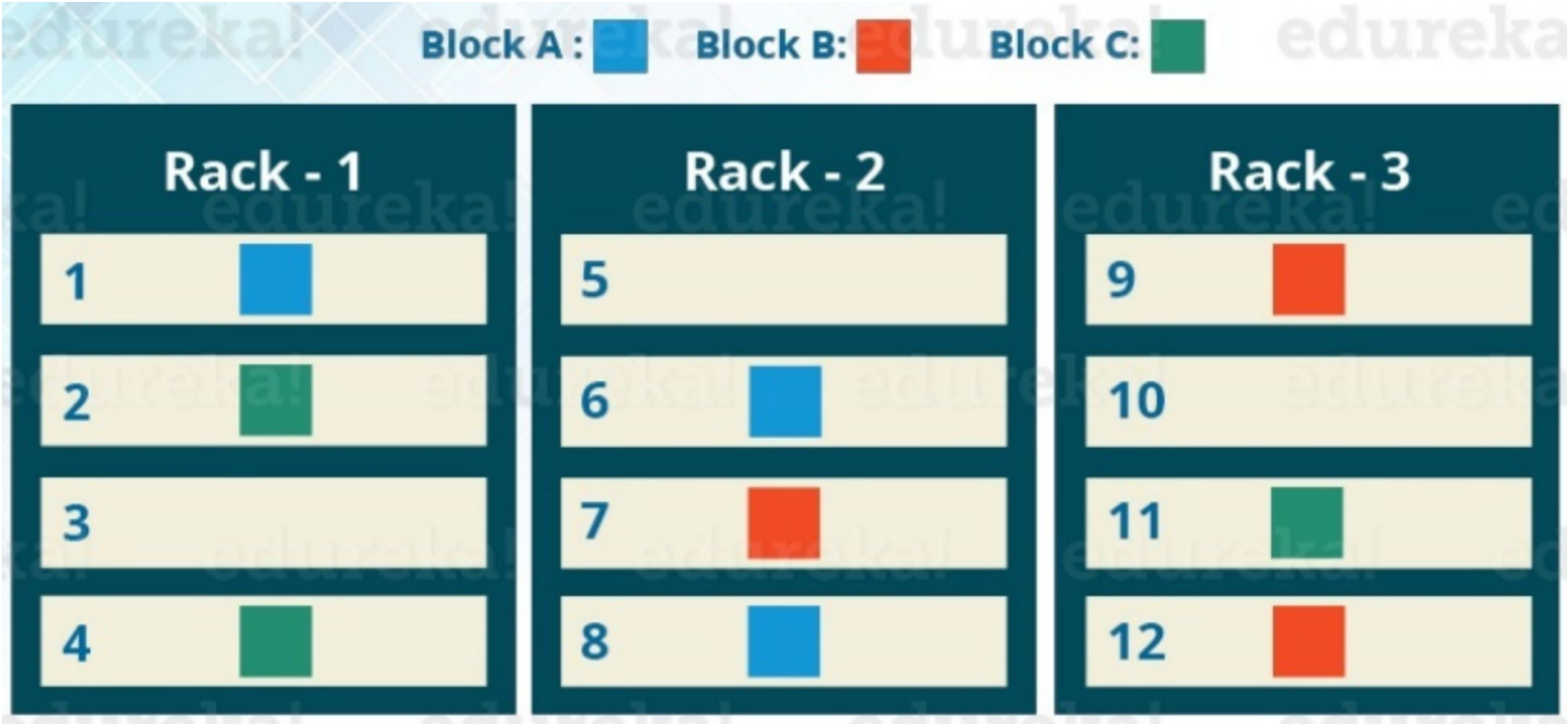
HDFS Rack Awareness



HDFS Rack Awareness



HDFS Rack Awareness



HDFS Balancer

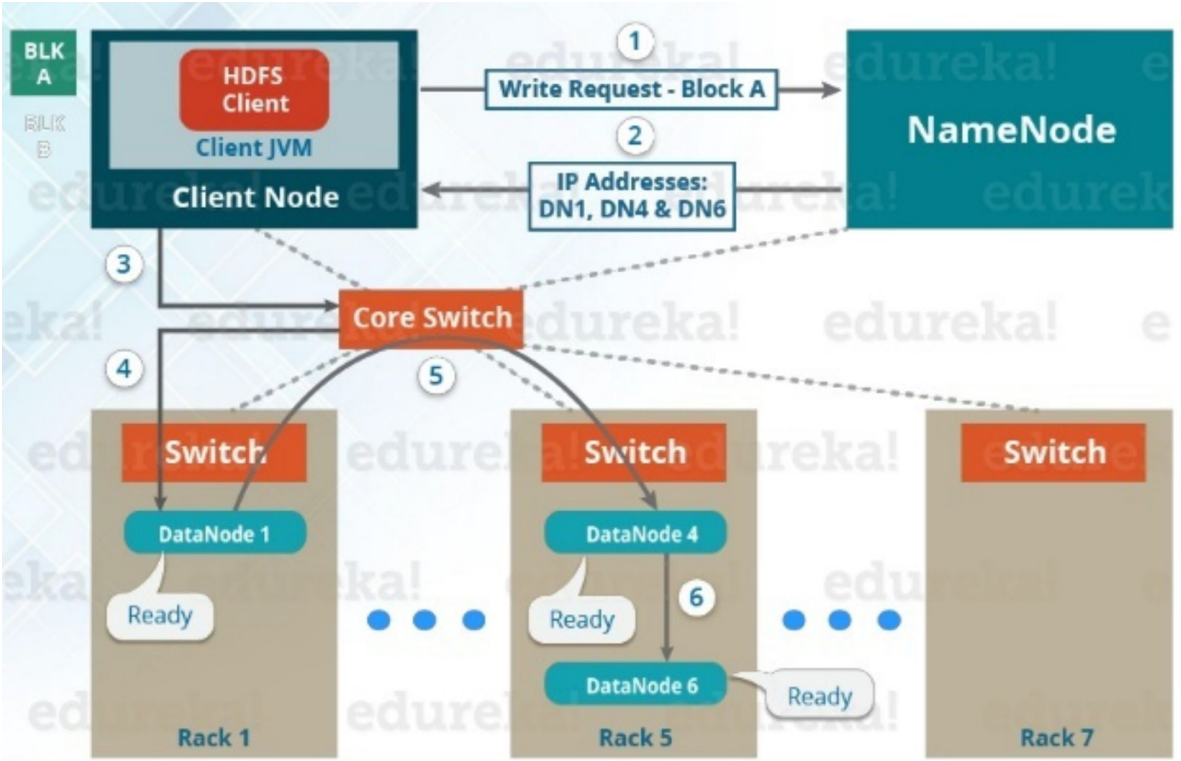
Moves block from over-utilized DNs to under-utilized DNs

Stops when HDFS is balanced

Maintains the block placement policy

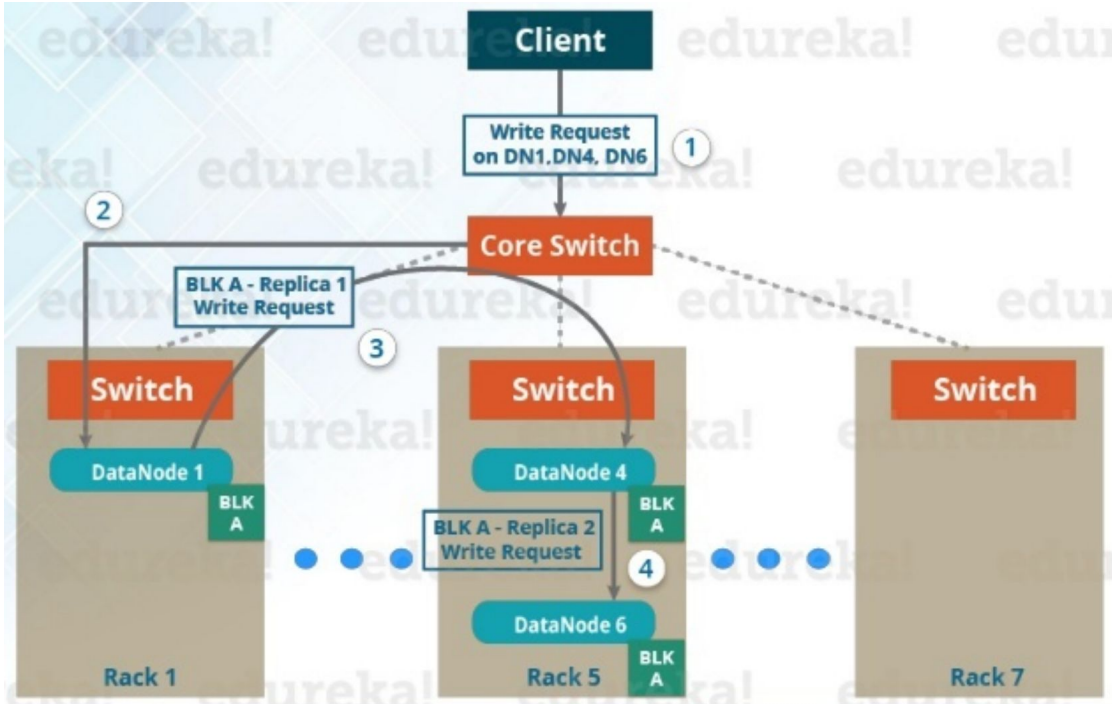
HDFS Write Mechanism

Pipeline Setup



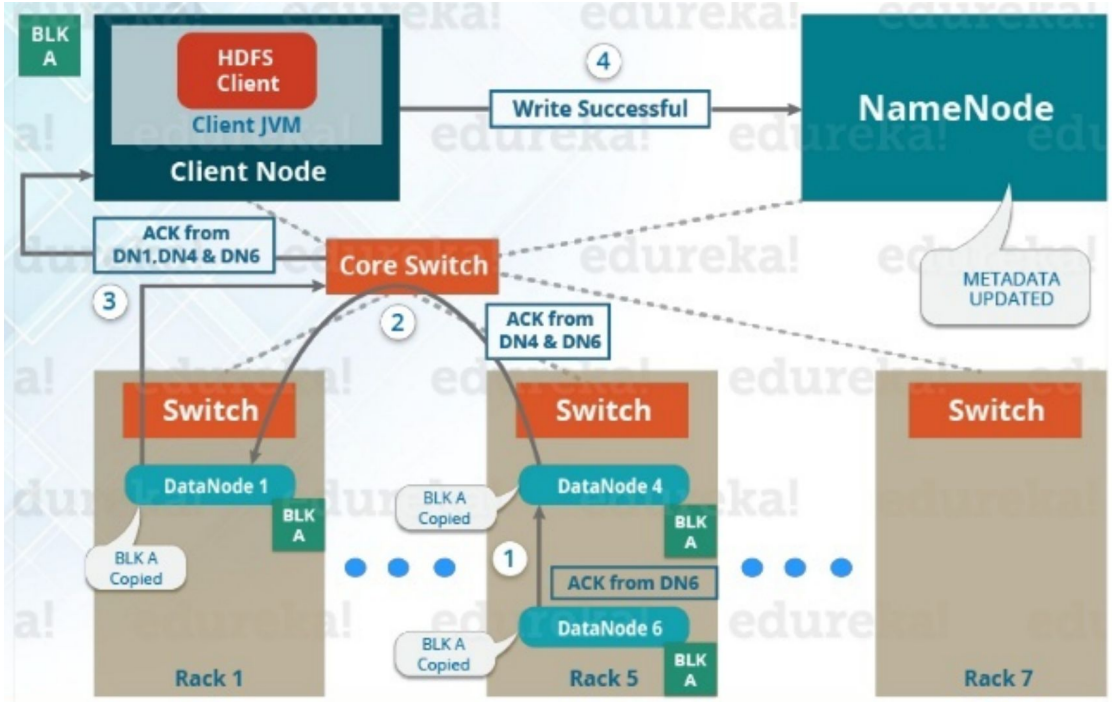
HDFS Write Mechanism

Write Replica



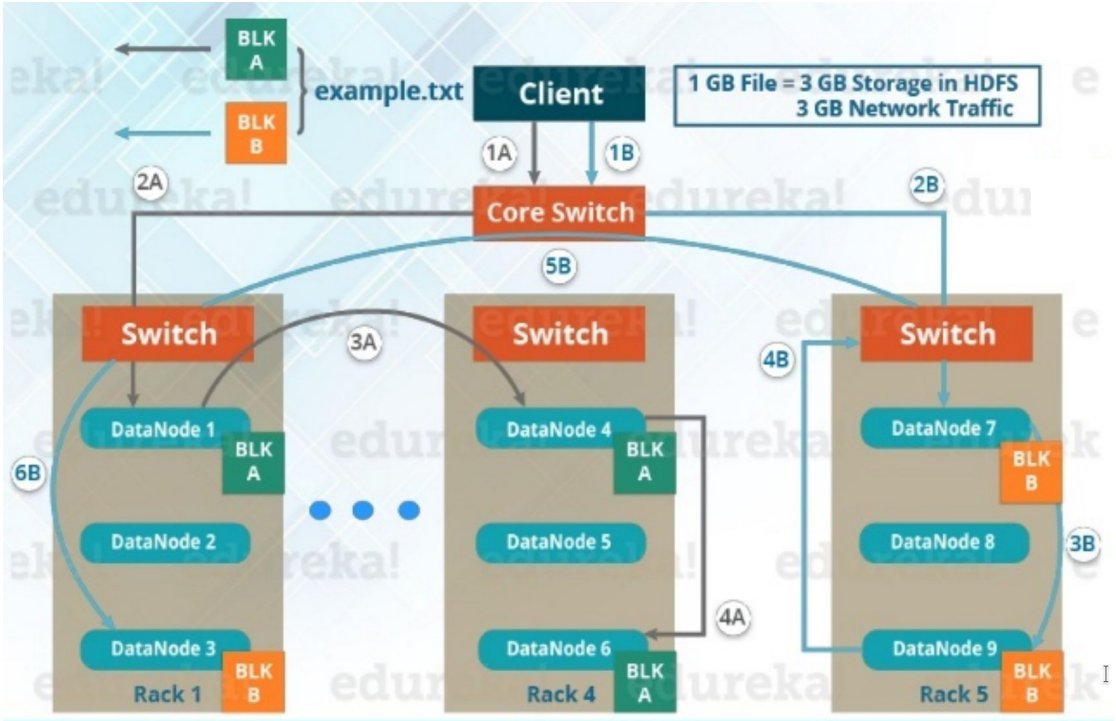
HDFS Write Mechanism

Acknowledgement

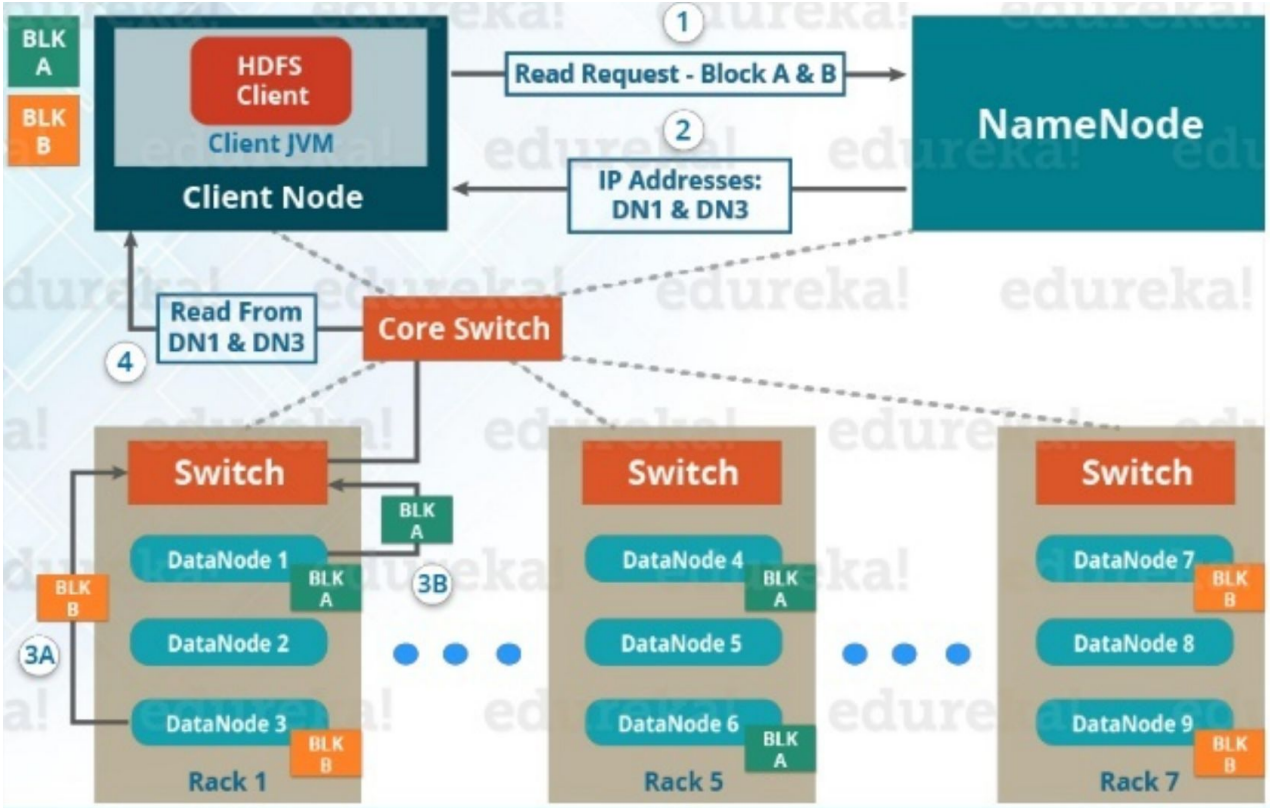


HDFS Write Mechanism

Multi block write



HDFS Read Mechanism



HDFS Missuse

- Low-latency requests
- Random reads and writes
- Lots of small files

Demo

- `hdfs dfs -ls /`
- `wget https://raw.githubusercontent.com/maxg203/Python-for-Beginners/master/shakespeare.txt`
- `hdfs dfs -mkdir /btpn-training`
- `hdfs dfs -put shakespeare.txt /btpn-training/.`
- `hdfs dfs -cp /btpn-training/shakespeare.txt /btpn-training/shakespeare2.txt`

`NameNode WebUI : http://btpn-training-m:9870/`

Quiz

- What is main component HDFS, explain each roles!
- What happen to secondary name node if main name node fail?
- What is HDFS blocks and how blocks is stored?
- What is HDFS balancer?

Map Reduce

Programming Model

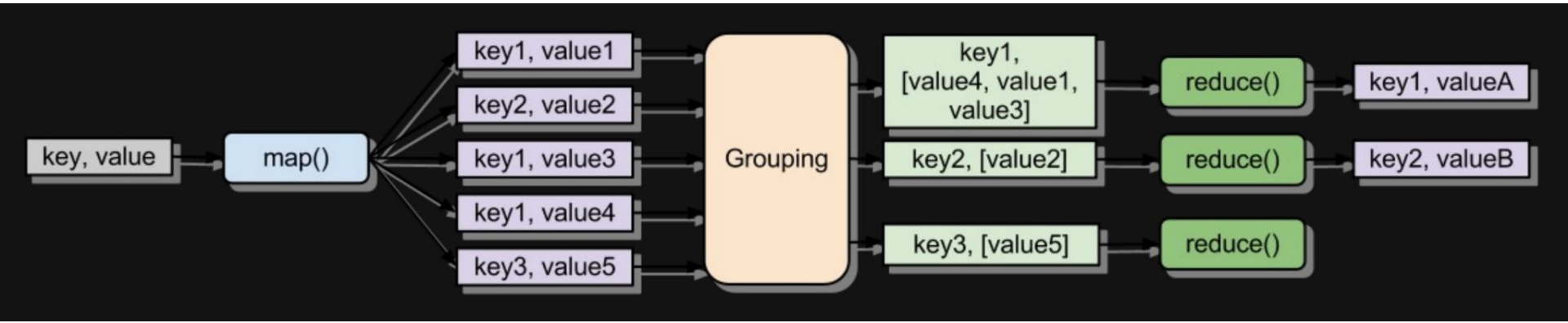
Map Reduce

Programming model inspired by functional programming map() and reduce() **functions processing <key, value> pairs**.

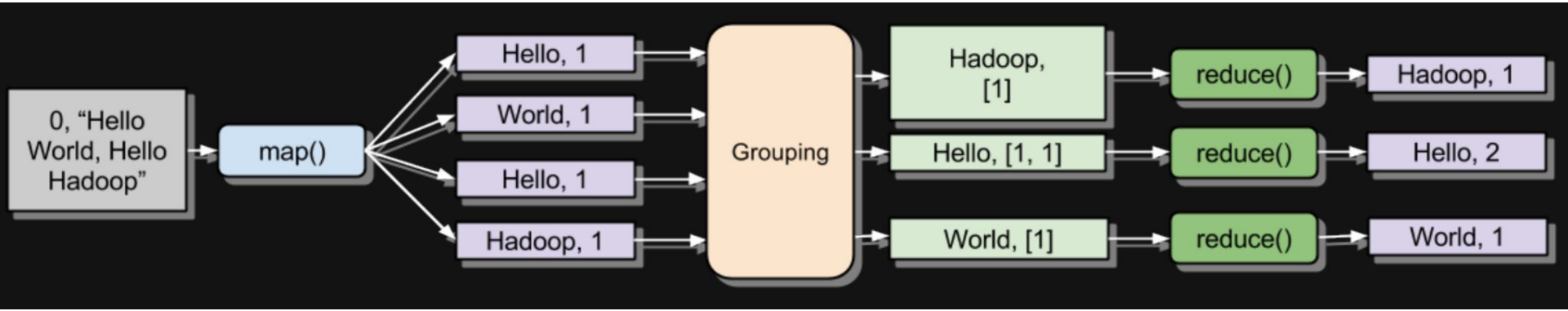
Useful for processing very large datasets in a **distributed way**

Simple, but **very expressible**

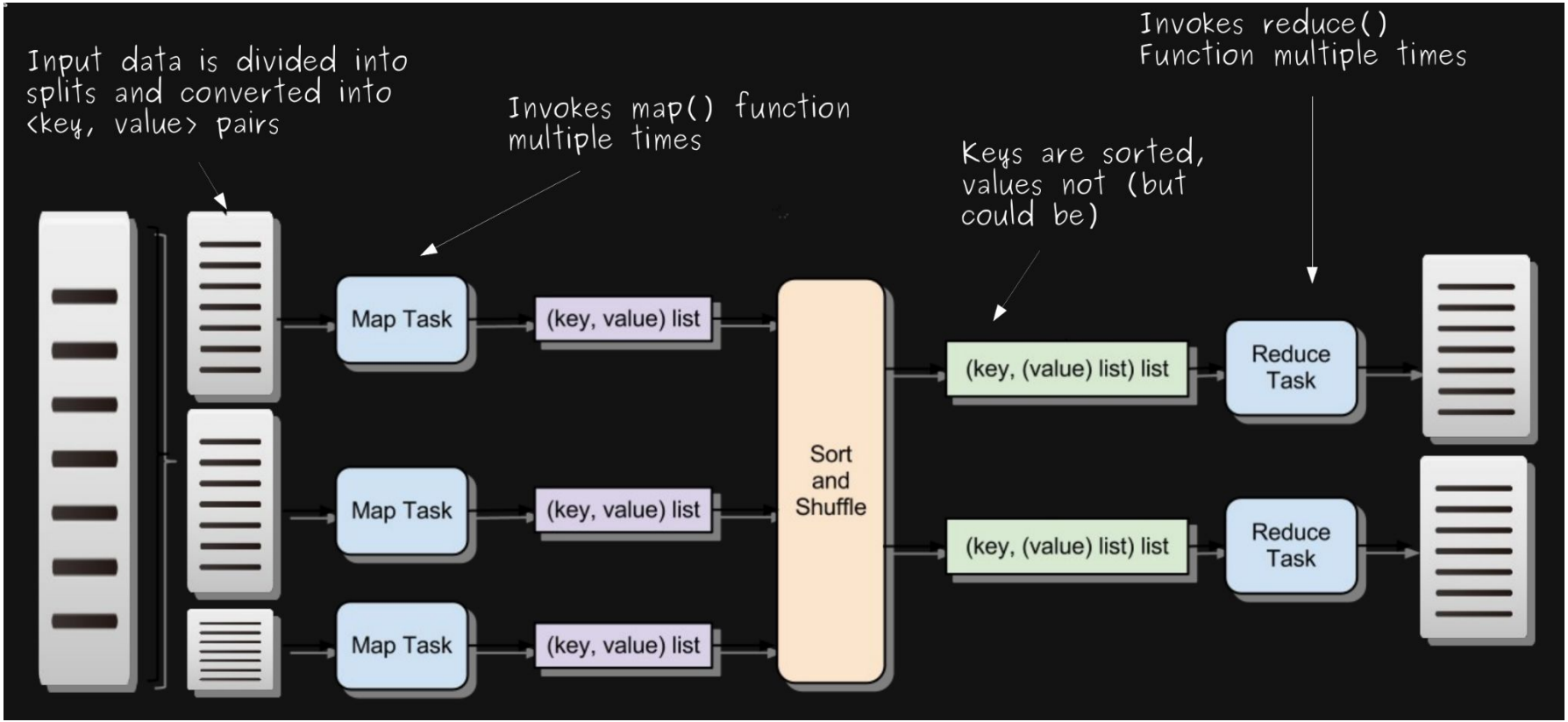
Map And Reduce Functions



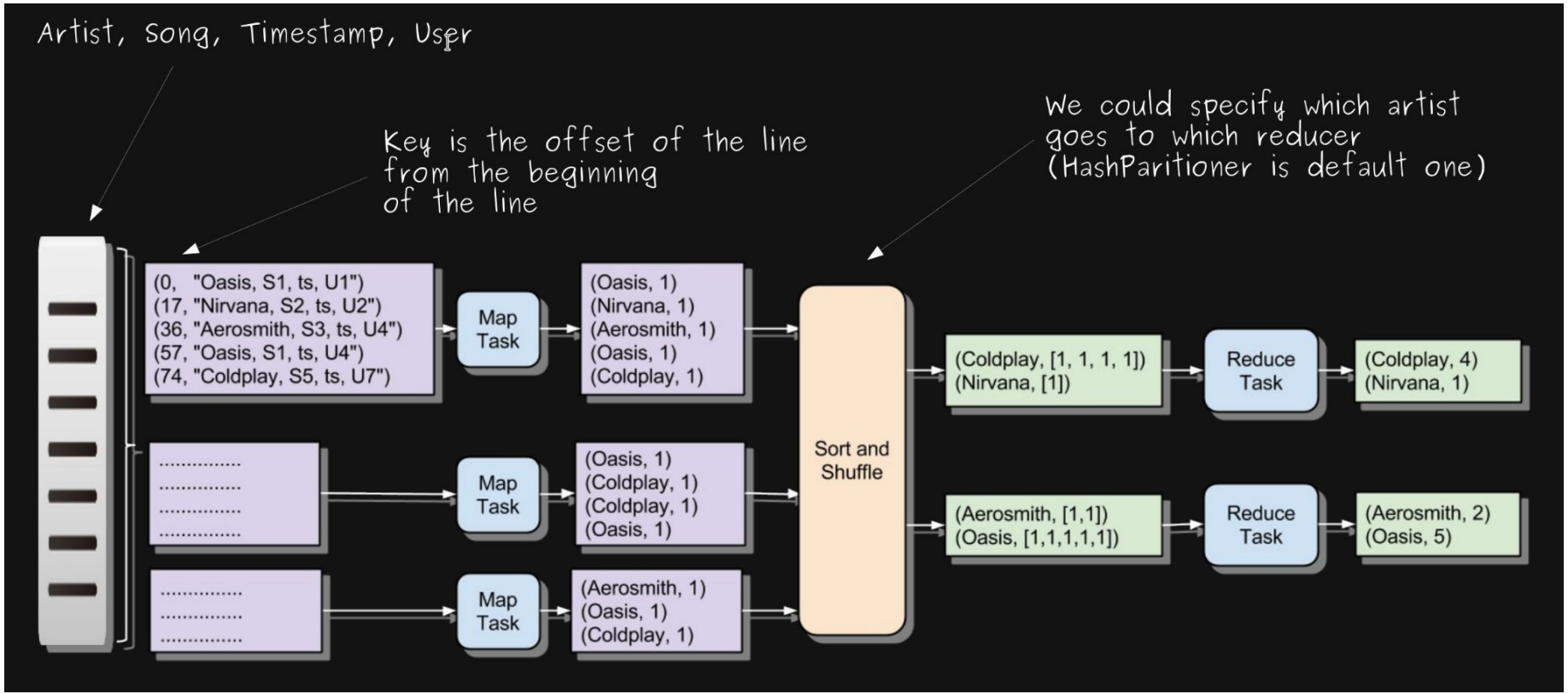
Map And Reduce Functions: Counting Words



MapReduce Job



MapReduce Example: Artist Count

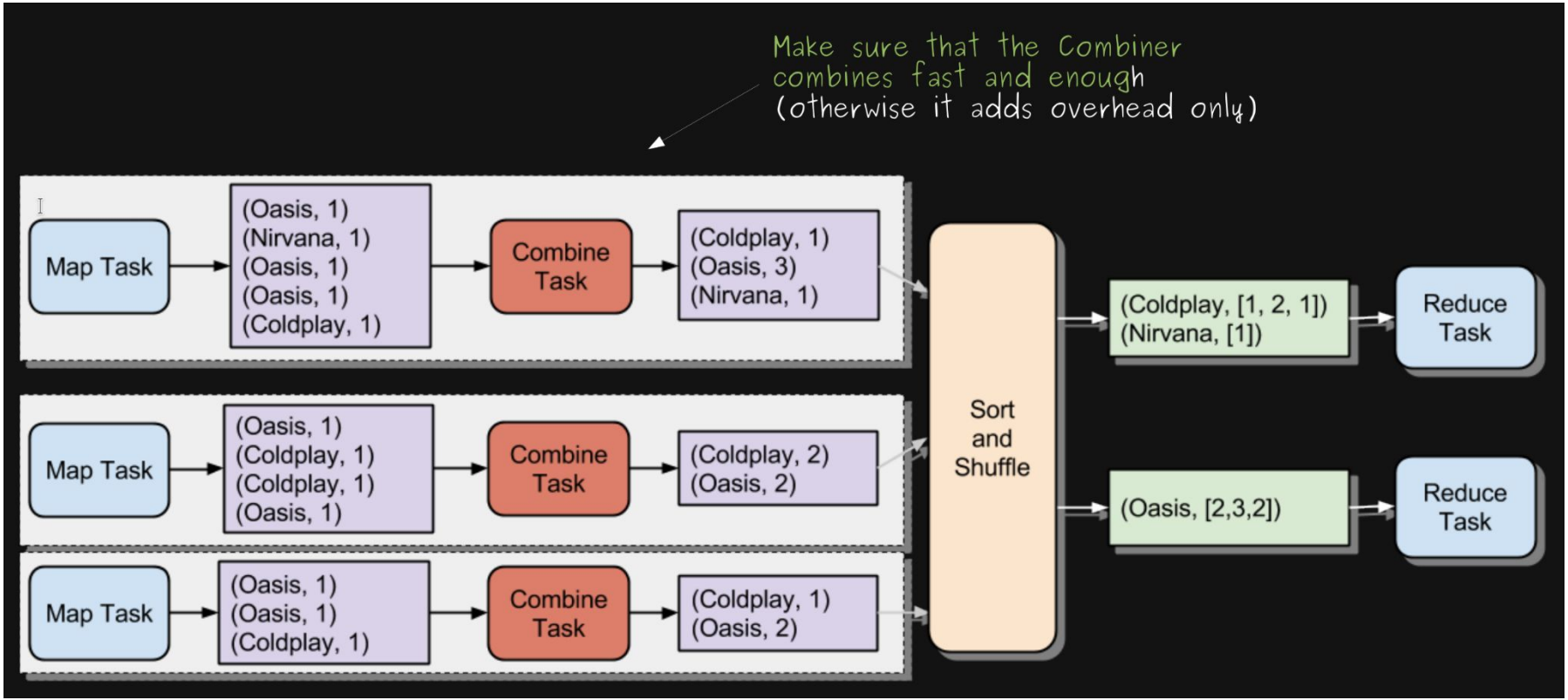


MapReduce Example: ArtistCount

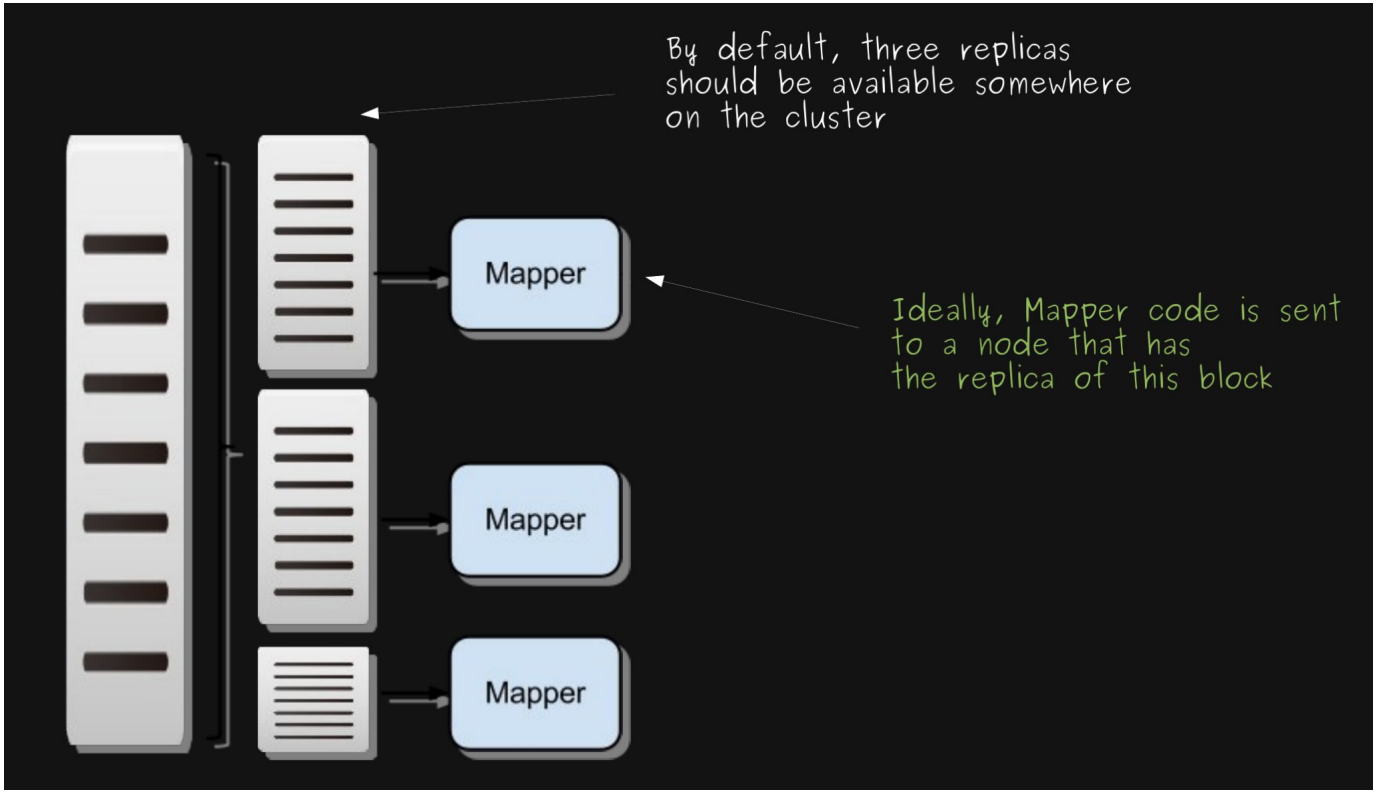
```
map(Integer key, EndSong value, Context context):
    context.write(value.artist, 1)


reduce(String key, Iterator<Integer> values, Context context):
    int count = 0
    for each v in values:
    count += v
    context.write(key, count)
```

Disclaimer:
Not actual programming language

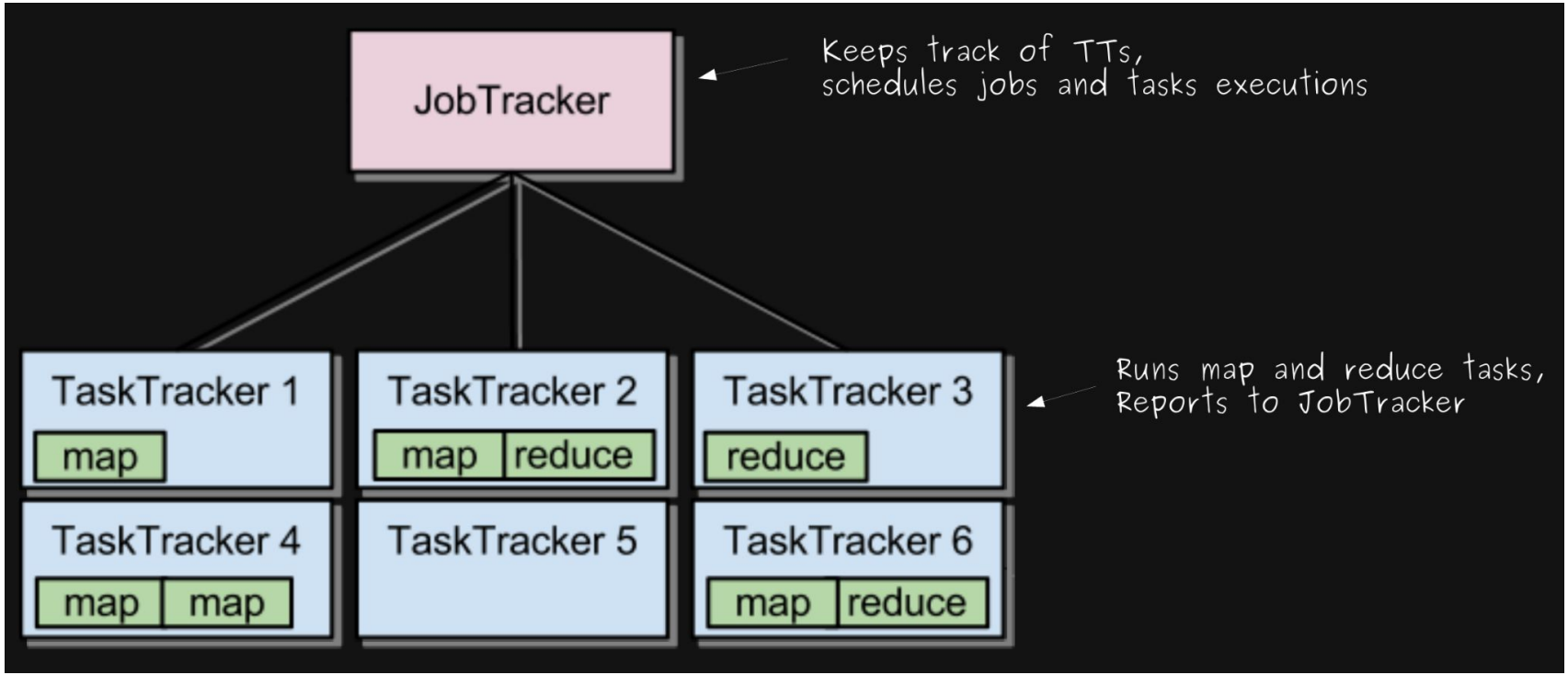
Map Reduce Combiner



Data Locality in HDFS and MapReduce



MapReduce Daemon



JobTracker Reponsibilities

Manages the computational resources

- Available TaskTrackers, map and reduce slots

Schedules all user jobs

- Schedules all tasks that belongs to a job
- Monitors tasks executions
- Restarts failed and speculatively runs slow tasks
- Calculates job counters totals

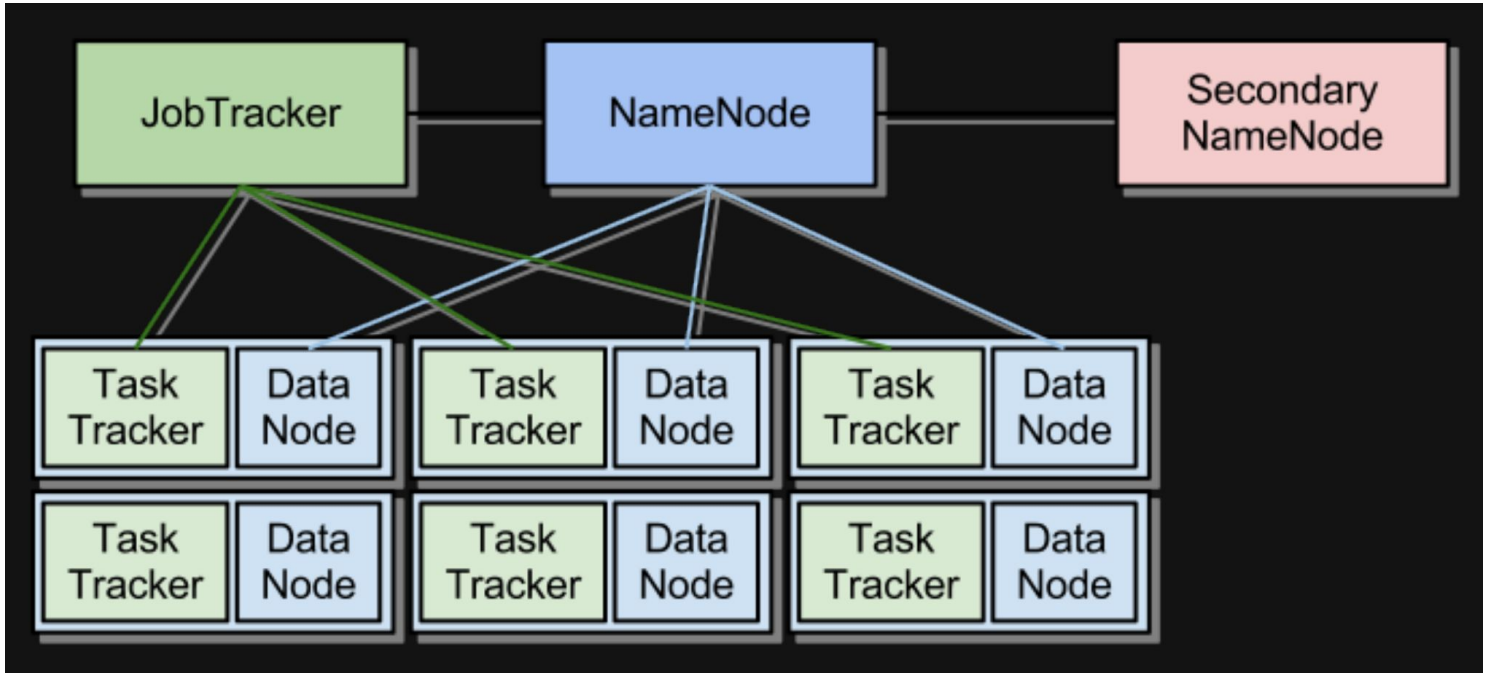
TaskTracker Reponsibilities

Runs map and reduce tasks

Reports to JobTracker

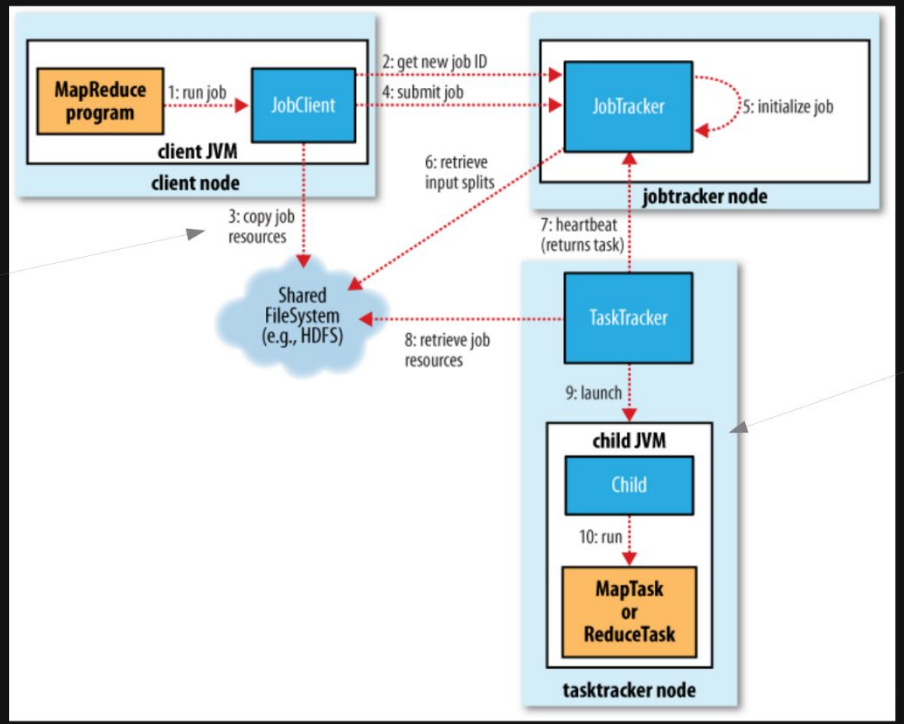
- Heartbeats saying that it is still alive
- Number of free map and reduce slots
- Task progress, status, counters etc

Hadoop Cluster



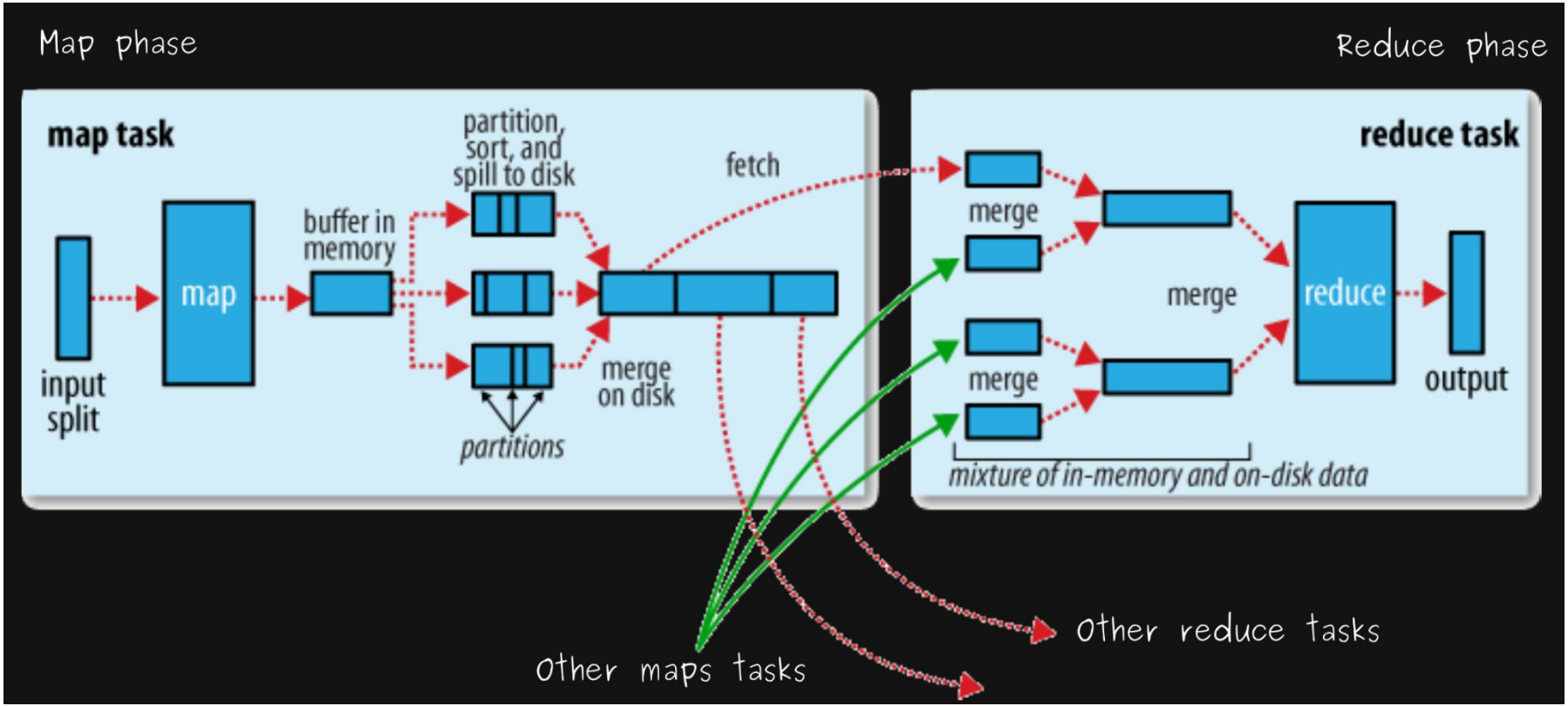
MapReduce Job Submission

They are copied with a higher replication factor (by default, 10)



Tasks are started in a separate JVM to isolate a user code form Hadoop code

MapReduce: Sort And Shuffle Phase



MapReduce: Partitioner

Specifies which Reducer should get a given <key, value> pair

- Aim for an even distribution of the intermediate data
- Skewed data may overload a single reducer
- And make a job running much longer

Speculative Execution

Scheduling a redundant copy of the remaining, long-running task

- The output from the one that finishes first is used
- The other one is killed, since it is no longer needed
- An optimization, not a feature to make jobs run more reliable

Only enable, if tasks often experience “external” problems e.g.

- Hardware degradation (disk, network card), system problems,
- Memory unavailability..

Otherwise:
can reduce overall throughput

Demo

- `mkdir tmp`
- `cp /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar ./tmp/`
- `unzip hadoop-mapreduce-examples.jar`
- `https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html`
- `yarn jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount /btpn-training/ /tmp/result`
- `hdfs dfs -get /tmp/result/* ./tmp/.`

Quiz

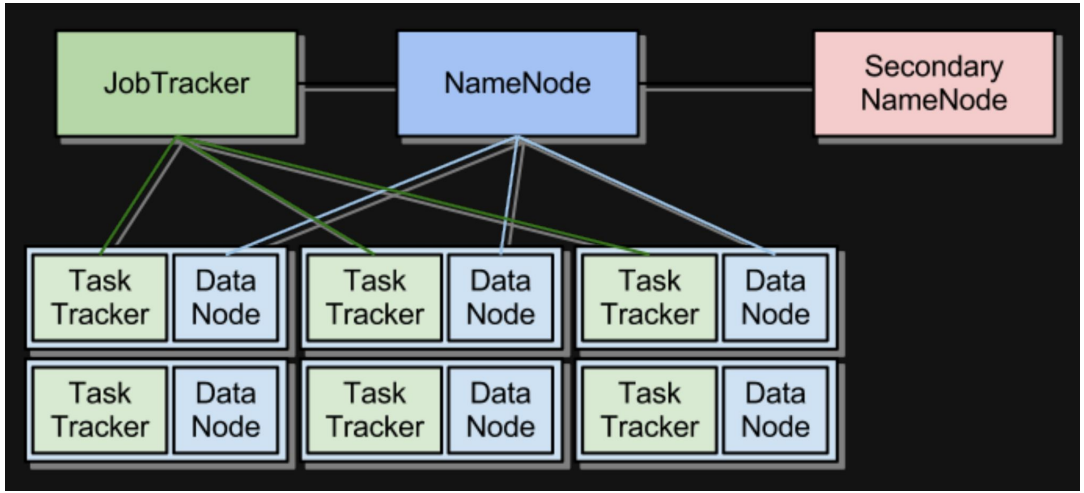
- What is MapReduce?
- What is MapReduce daemon?
- What data locality means?
- Why hadoop and not regular DFS?

YARN

Yet Another Resource Negotiator

“Classical” MapReduce Limitations

- Limited scalability
- Poor resource utilization
- Lack of support for the alternative frameworks
- Lack of wire-compatible protocols



Limited Scalability

Scales to only

- ~4,000-node cluster
- ~40,000 concurrent tasks

Smaller and less-powerful clusters must be created

Poor Cluster Resources Utilization

Cluster Summary (Heap Size is 12.58 GB/23.97 GB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node
4512	285	2223	188	4512	285	0	0	4512	3008	40.00

Map slots is full. New map task will need to wait this free even reduce slot is free.

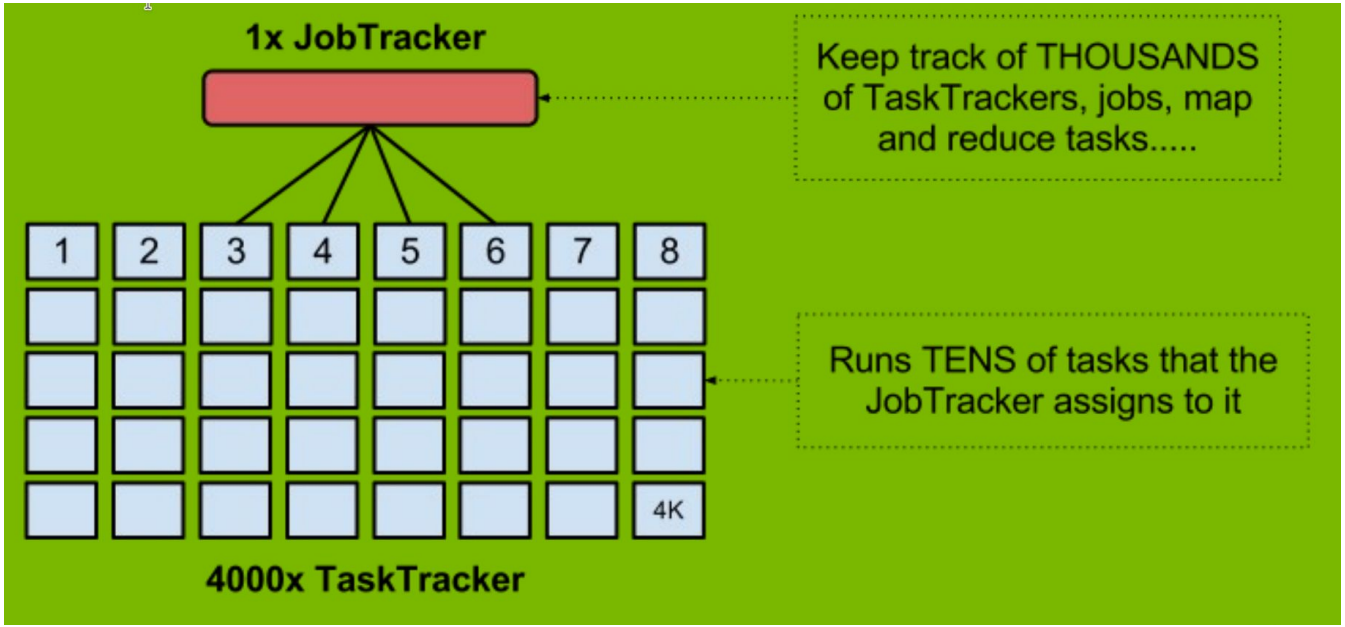
Capacity is hard coded value. If we changed this then task tracker need to be restarted

Resources Needs That Varies Over Time

How to hard-code the number of map and reduce slots efficiently?

Simple Observation

JobTracker has lots tasks...



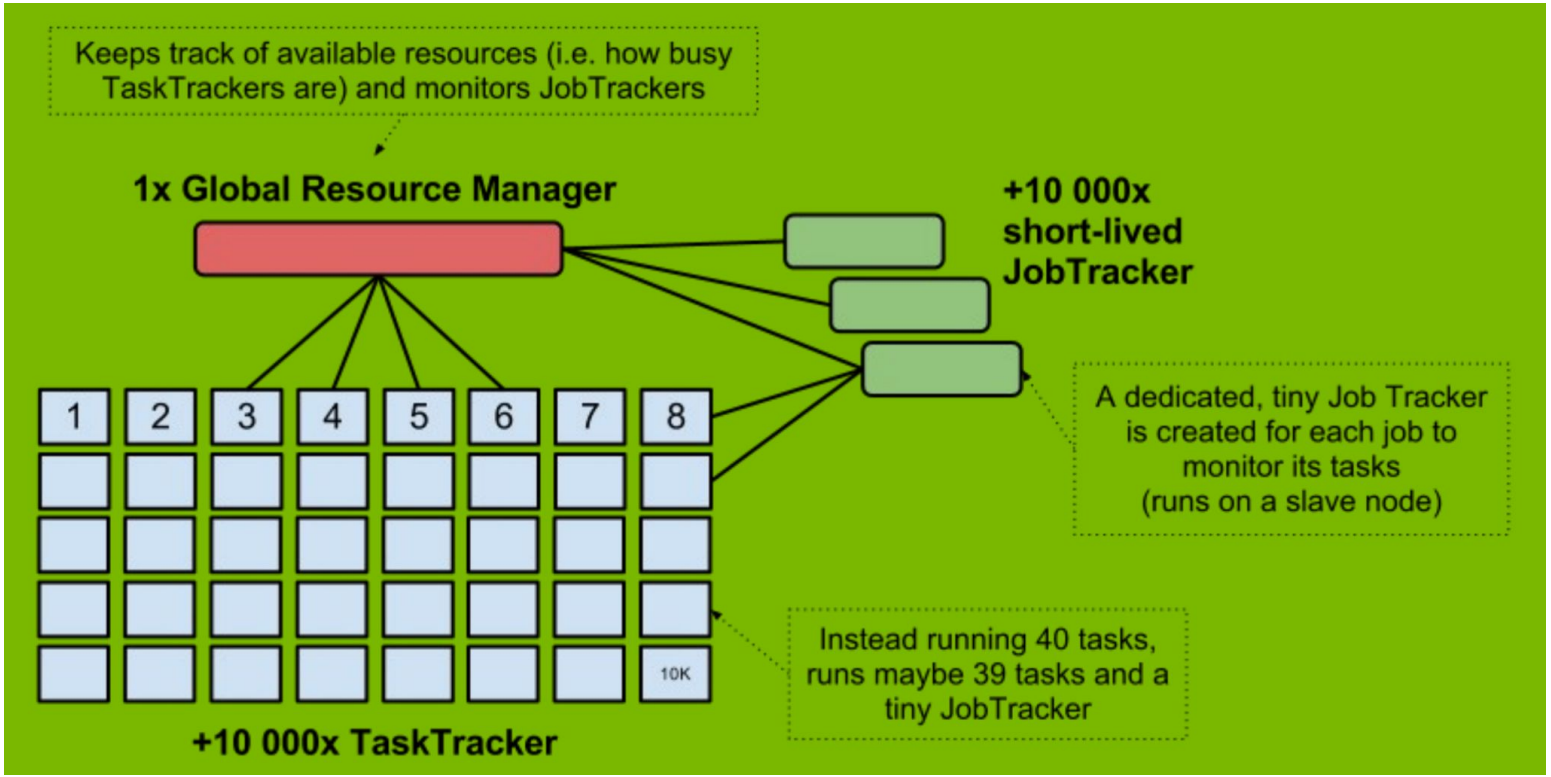
JobTracker Redesign Idea

Reduce responsibilities of JobTracker

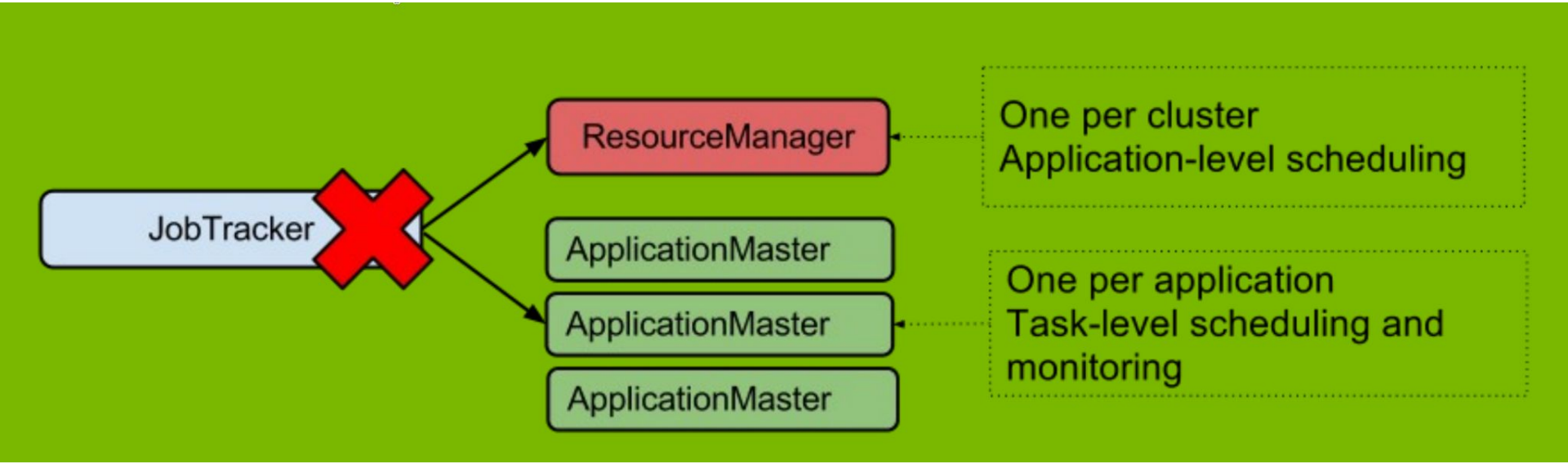
- Separate cluster resource management from job coordination
- Use slaves (many of them!) to manage jobs life-cycle

Scale to (at least) 10K nodes, 10K jobs, 100K tasks

Disappearance Of The Centralized JobTracker



Resource Manager and Application Master



YARN Applications

MRv2 (simply MRv1 rewritten to run on top of YARN)

No need to rewrite existing MapReduce jobs

Distributed shell

Spark, Apache S4 (a real-time processing)

Hamster (MPI on Hadoop)

Apache Giraph (a graph processing)

Apache HAMA (matrix, graph and network algorithms)

NodeManager

More flexible and efficient than TaskTracker

Executes any computation that makes sense to ApplicationMaster

- Not only map or reduce tasks

Containers with variable resource sizes (e.g. RAM, CPU, network, disk)

- No hard-coded split into map and reduce slots

NodeManager Containers

NodeManager creates a container for each task

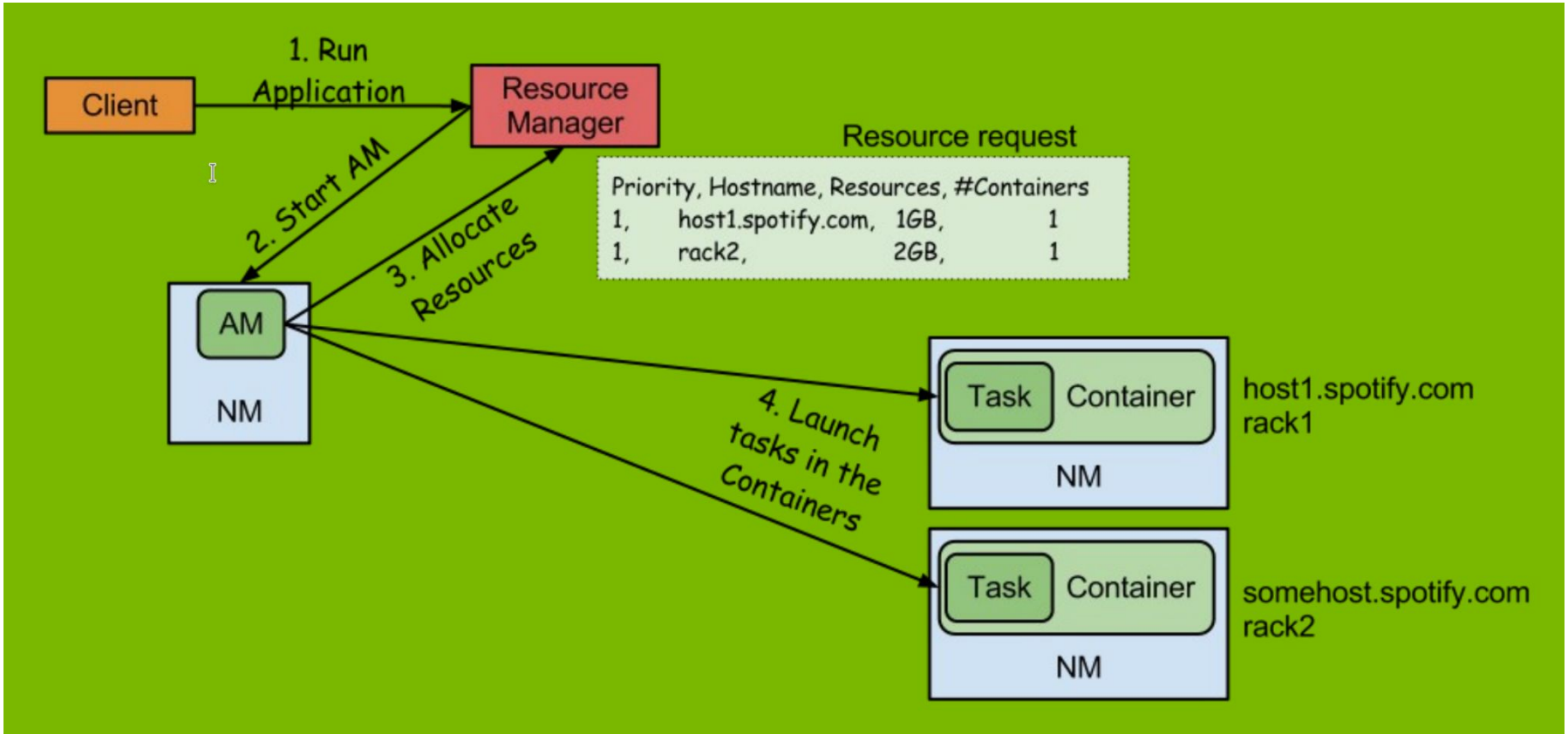
Container contains variable resources sizes

e.g. 2GB RAM, 1 CPU, 1 disk

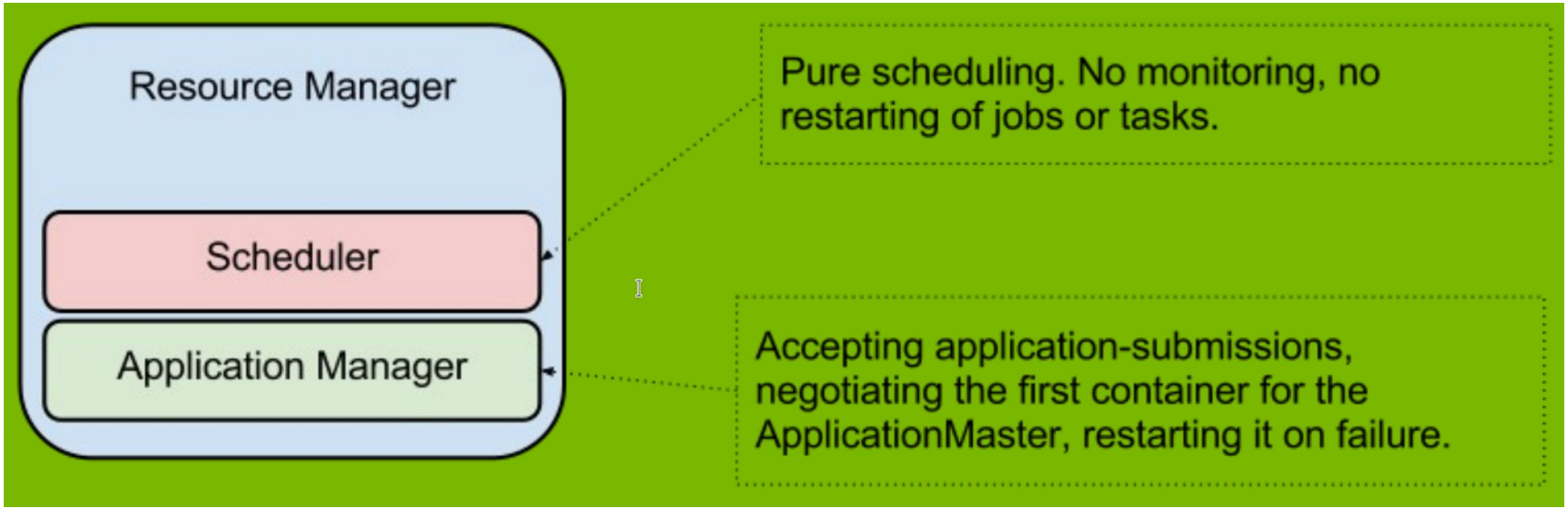
Number of created containers is limited

By total sum of resources available on NodeManager

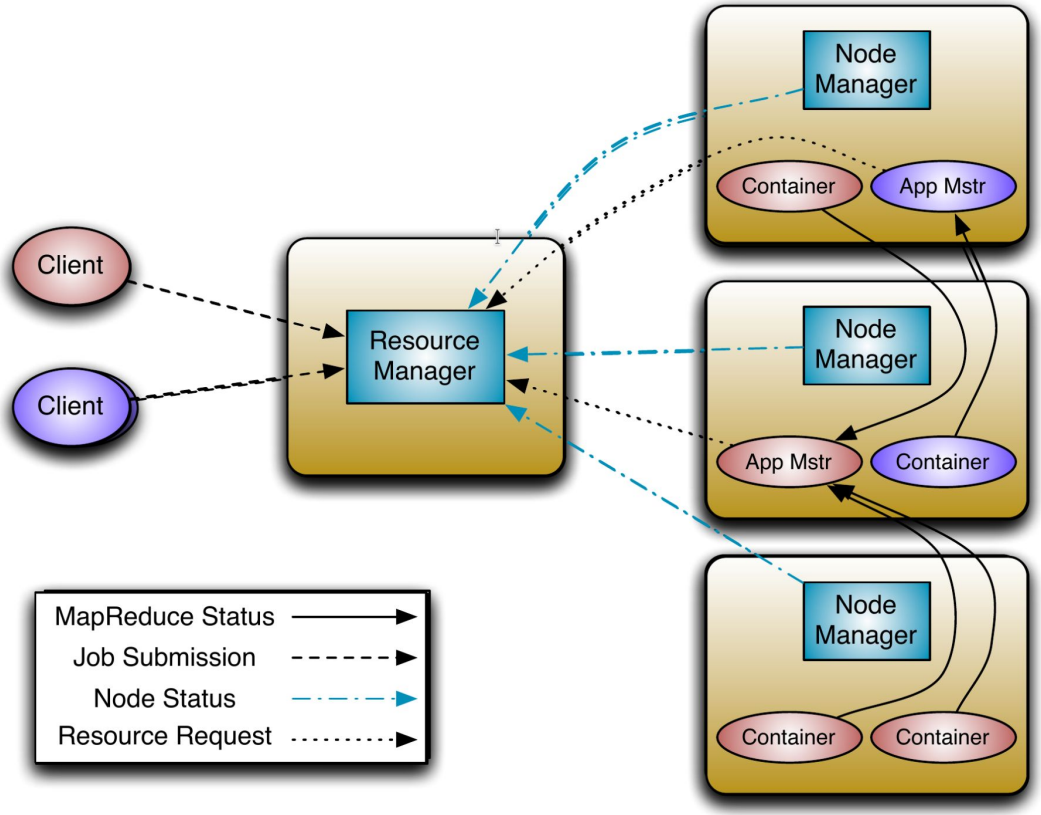
Application Submission In YARN



Resource Manager Components



Complete New Architecture



Yarn Improvement over MapReduce System

Utilization

- Generic resource model (Data Locality, Memory, CPU, Disk b/q, Network b/w)
- Remove fixed partition of map and reduce slot

Scalability

- Application life-cycle management is very expensive
- Partition resource management and application life-cycle management
- Application management is distributed
- Hardware trends - Currently run clusters of 4,000 machines
  - 6,000 2012 machines > 12,000 2009 machines
  - <16+ cores, 48/96G, 24TB> v/s <8 cores, 16G, 4TB>

Yarn Improvement over MapReduce System

Fault Tolerance and Availability

- Resource Manager
  - No single point of failure – state saved in ZooKeeper (coming soon)
  - Application Masters are restarted automatically on RM restart
- Application Master
  - Optional failover via application-specific checkpoint
  - MapReduce applications pick up where they left off via state saved in HDFS

Wire Compatibility

- Protocols are wire-compatible
- Old clients can talk to new servers
- Rolling upgrades

Yarn Improvement over MapReduce System

Innovation and Agility

- MapReduce now becomes a user-land library
- Multiple versions of MapReduce can run in the same cluster (a la Apache Pig)
- Faster deployment cycles for improvements
- Customers upgrade MapReduce versions on their schedule
- Users can customize MapReduce

Quiz

Why need to develop yarn?

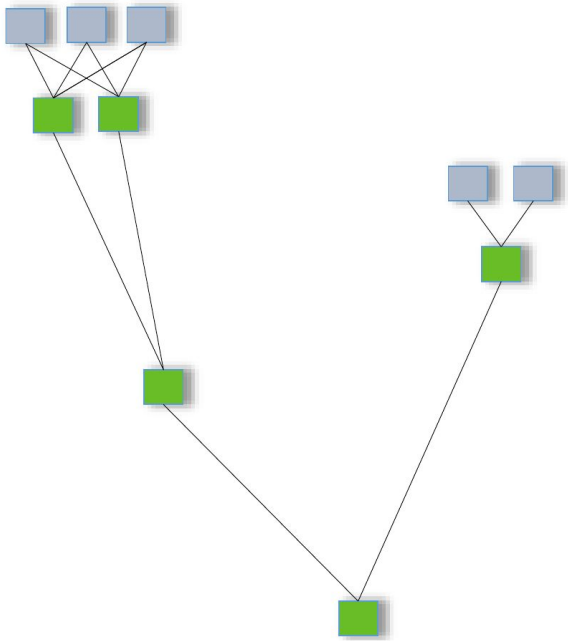
What is Node Manager?

What is Resource Manager?

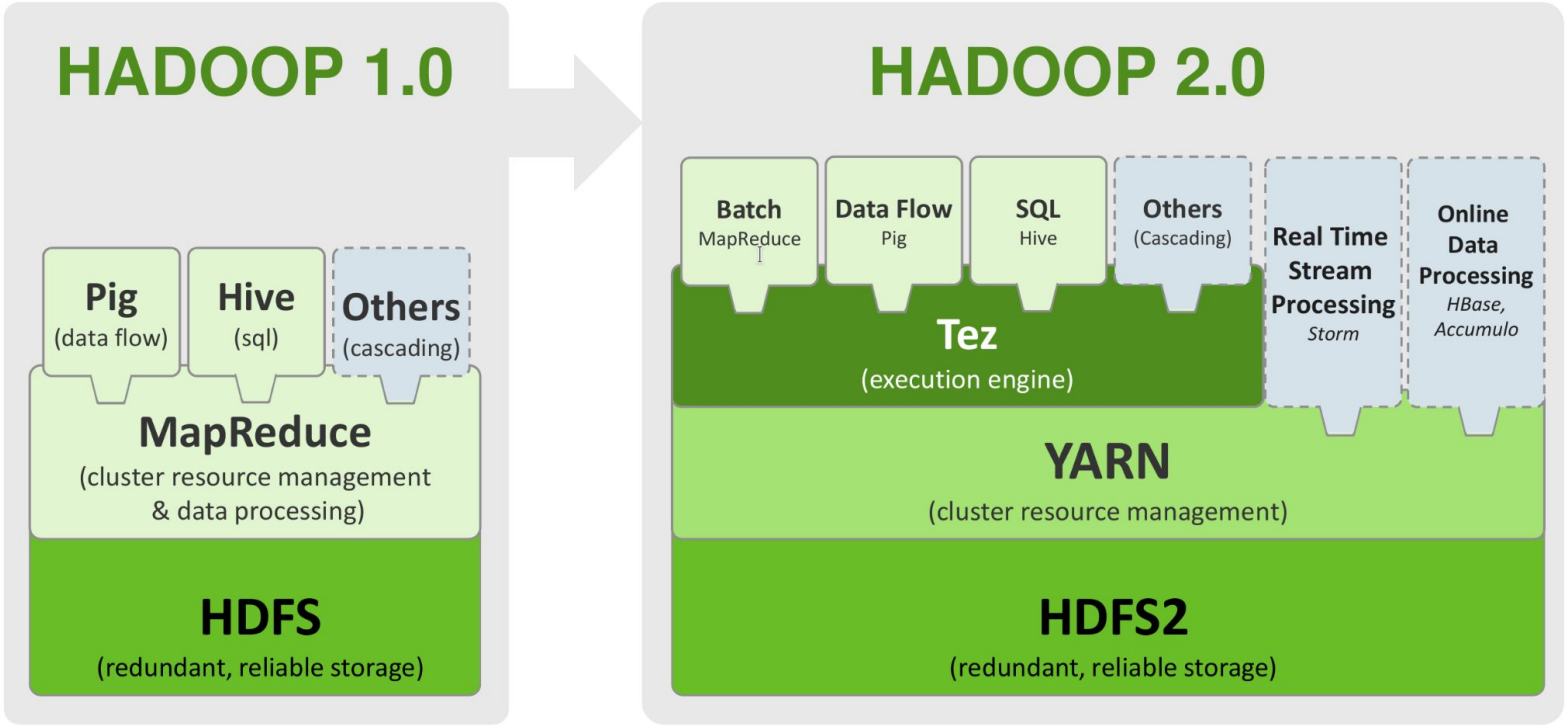
Apache Tez

Tez – Introduction

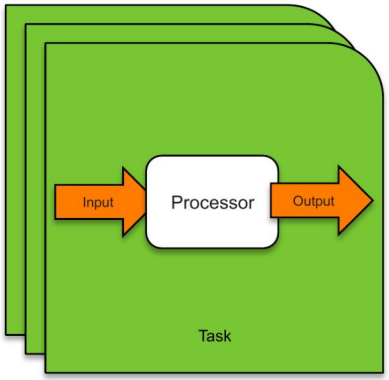
- Distributed execution framework targeted towards data-processing applications.
- Based on expressing a computation as a dataflow graph.
- Highly customizable to meet a broad spectrum of use cases.
- Built on top of YARN – the resource management framework for Hadoop.
- Open source Apache incubatorproject and Apache licensed.



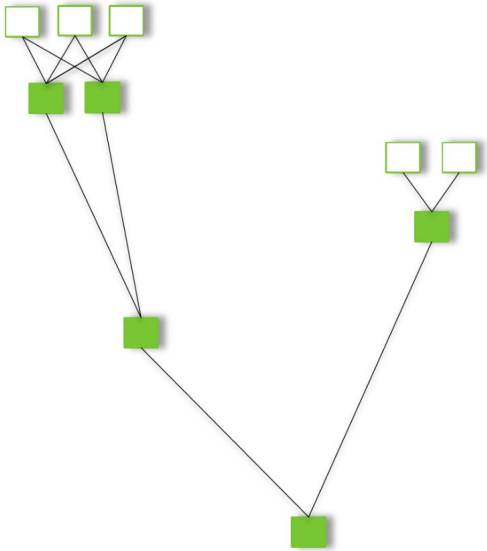
Hadoop 1.x vs Hadoop 2.x



Tez - Core Idea



Tez Task - <*Input, Processor, Output*>



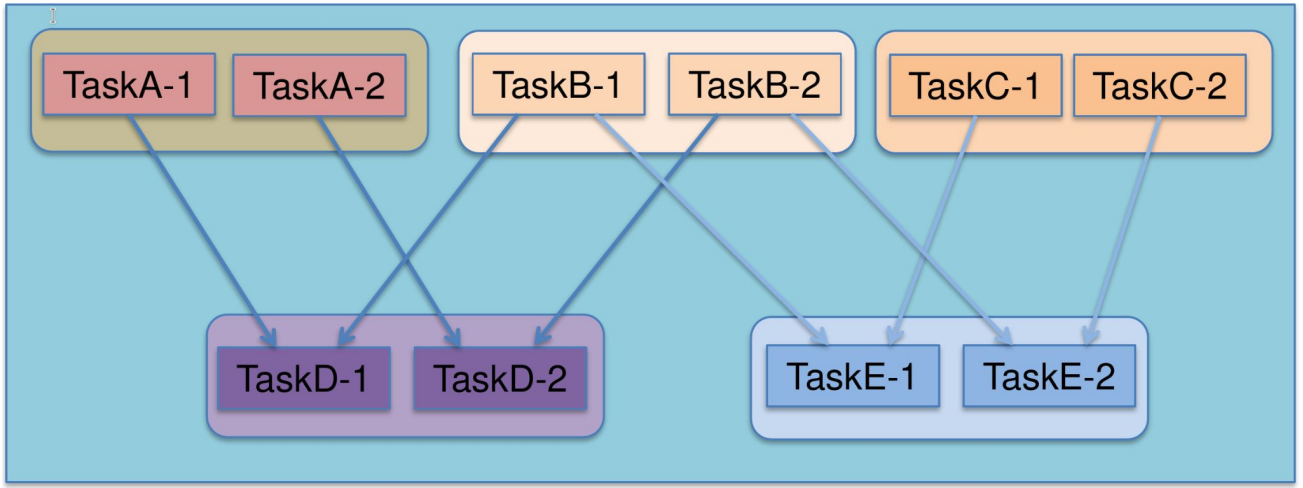
Tez Benefits

- Empowering end user
- Improve Performance

Empowering End User

Expressive dataflow definition API's

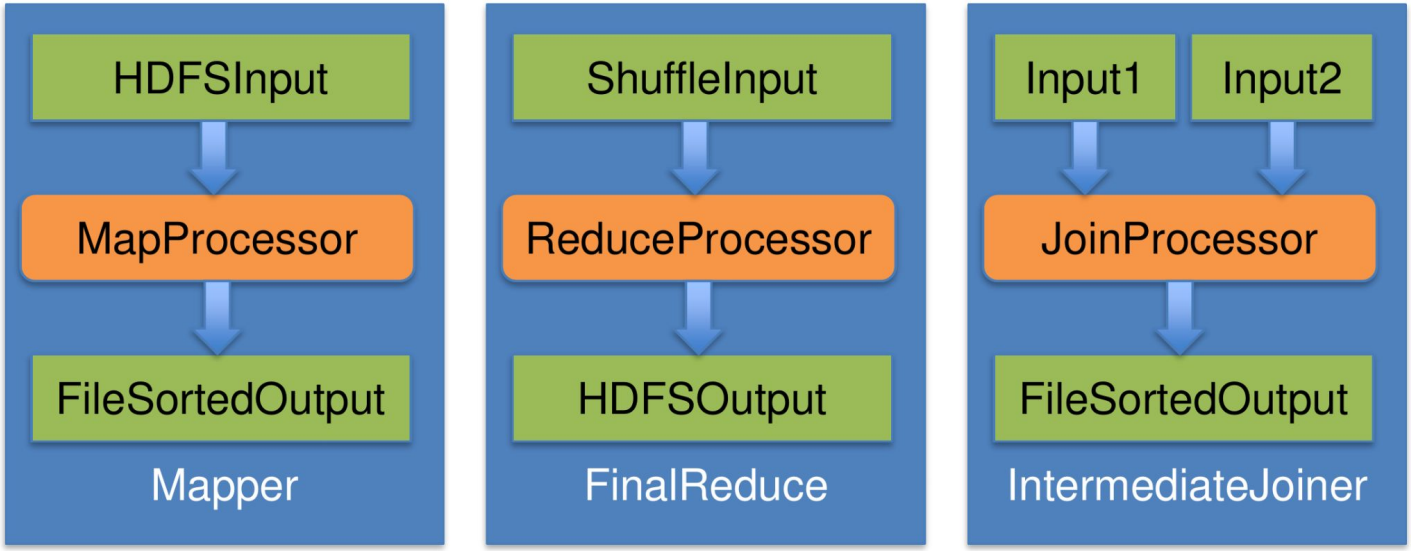
- Enable definition of complex data flow pipelines using simple graph connection API's. Tez expands the logical plan at runtime.
- Targeted towards data processing applications like Hive/Pig but not limited to it. Hive/Pig query plans naturally map to Tez dataflow graphs with no translation impedance.



Empowering End User

Flexible Input-Processor-Output runtime model

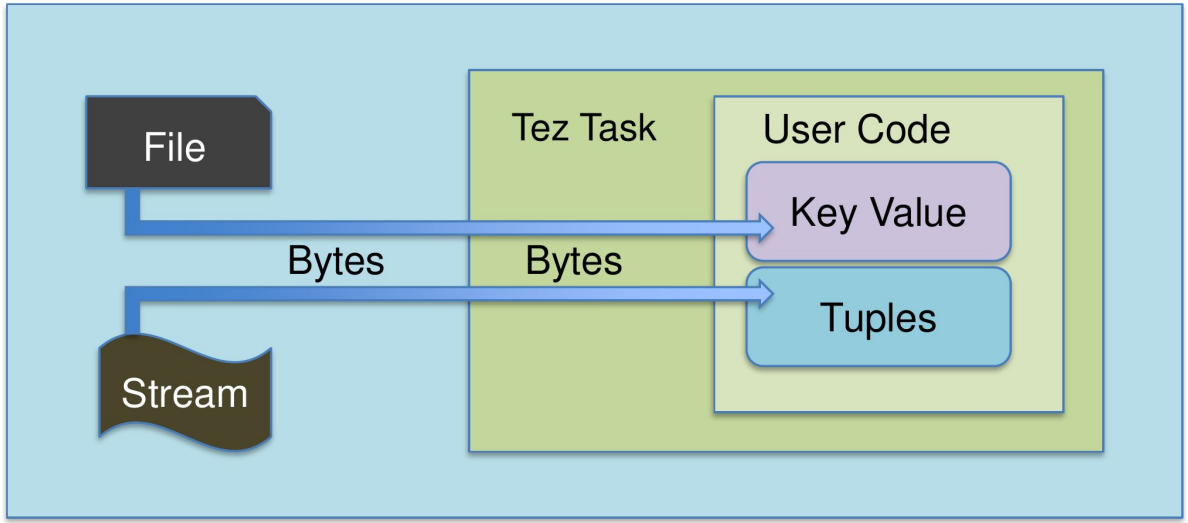
- Construct physical runtime executors dynamically by connecting different inputs, processors and outputs.
- End goal is to have a library of inputs, outputs and processors that can be programmatically composed to generate useful tasks.



Empowering End User

Data type agnostic

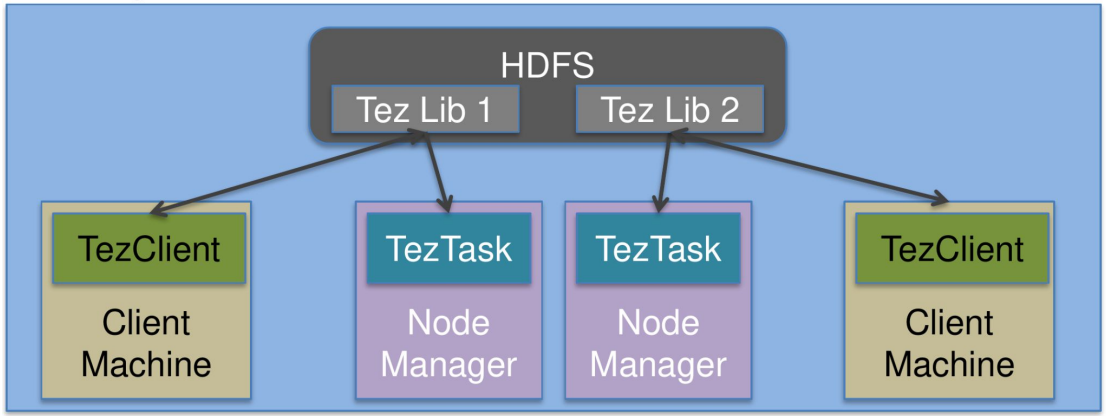
- Tez is only concerned with the movement of data. Files and streams of bytes.
- Does not impose any data format on the user application. MR application can use Key-Value pairs on top of Tez. Hive and Pig can use tuple oriented formats that are natural and native to them



Empowering End User

Simplifying deployment

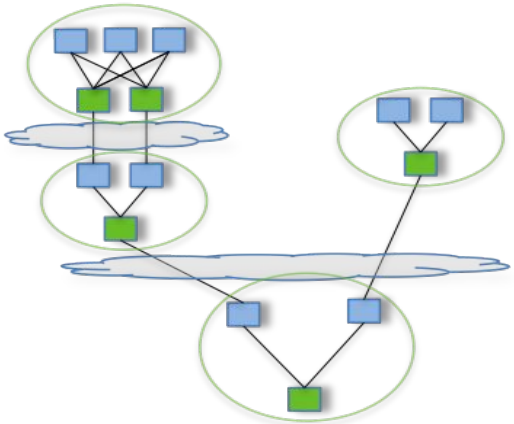
- Tez is a completely client side application.
- No deployments to do. Simply upload to any accessible FileSystem and change local Tez configuration to point to that.
- Enables running different versions concurrently. Easy to test new functionality while keeping stable versions for production.
- Leverages YARN local resources.



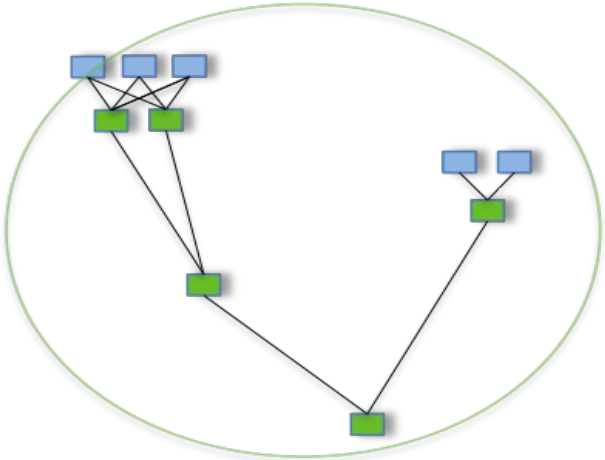
Tez Execution Performance

Performance MapReduce VS Tez

- Eliminate replicated write barrier between successive computations.
- Eliminate job launch overhead of workflow jobs.
- Eliminate extra stage of map reads in every workflow job.
- Eliminate queue and resource contention suffered by workflow jobs that are started after a predecessor job completes.



Pig/Hive - MR

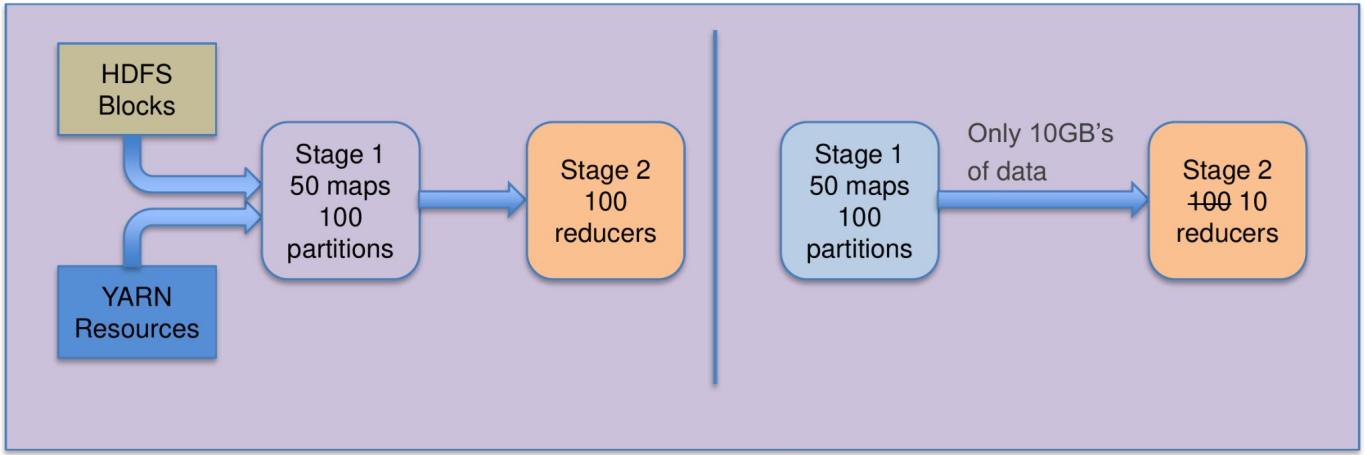


Pig/Hive - Tez

Tez Execution Performance

Plan reconfiguration at runtime

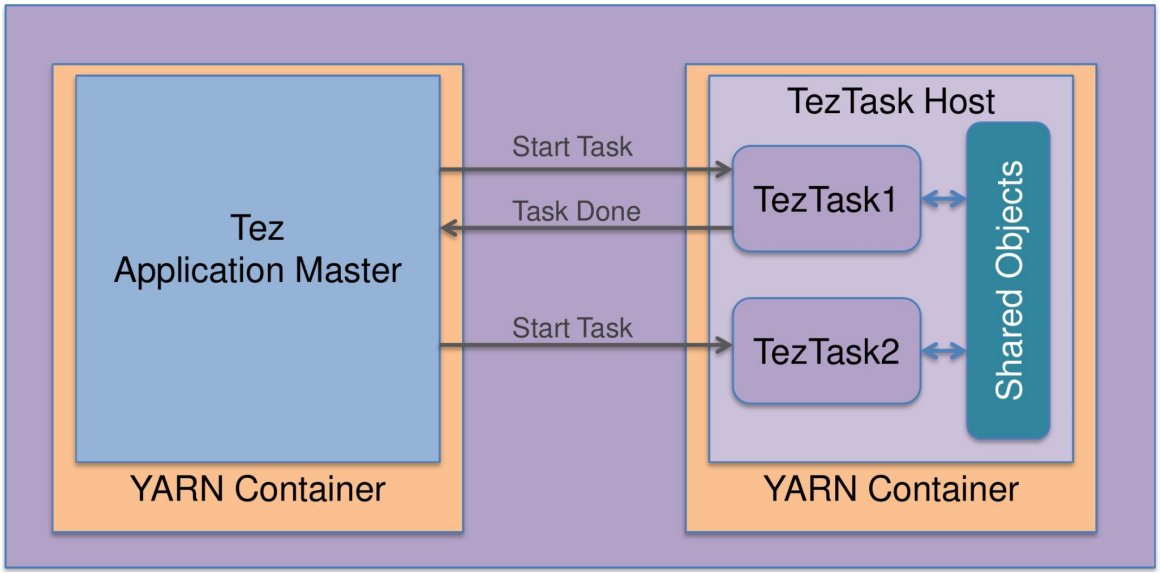
- Dynamic runtime concurrency control based on data size, user operator resources, available cluster resources and locality.
- Advanced changes in dataflow graph structure.
- Progressive graph construction in concert with user optimizer.



Tez Execution Performance

Optimal resource management

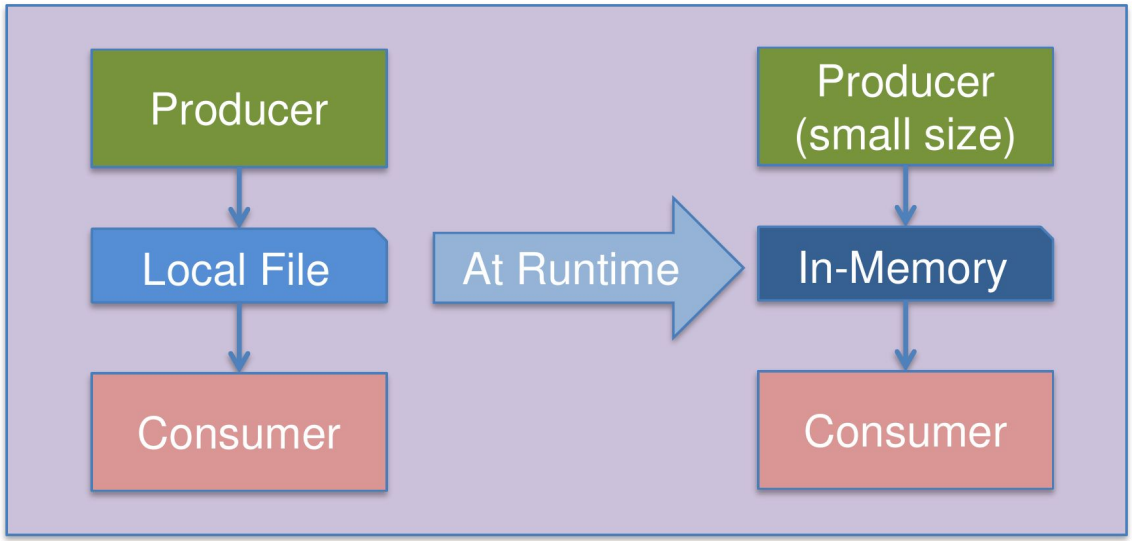
- Reuse YARN containers to launch new tasks.
- Reuse YARN containers to enable shared objects across tasks.



Tez Execution Performance

Dynamic physical data flow decisions

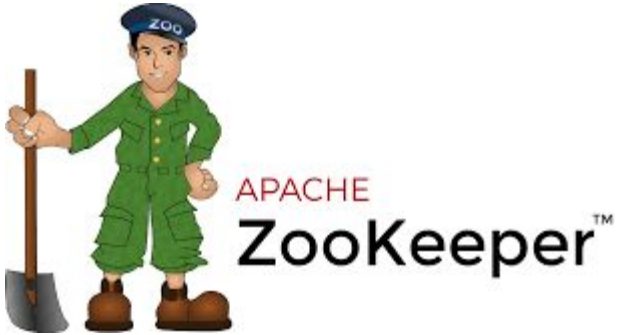
- Decide the type of physical byte movement and storage on the fly.
- Store intermediate data on distributed store, local store or in-memory.
- Transfer bytes via blocking files or streaming and the spectrum in between.



Quiz

What is Apache Tez?

What is main difference between Tez and MapReduce



Apache Zookeeper

What is Distributed System

A distributed system consists of multiple computers that communicate through a computer network and interact with each other to achieve a common goal.

Fallacies of Distributed Computing

- The network is reliable.
- Latency is zero.
- Bandwidth is infinite.
- The network is secure.
- Topology doesn't change.
- There is one administrator.
- Transport cost is zero.
- The network is homogeneous.

Coordination in distributed system

- Coordination: An act that multiple nodes must perform together.
- Examples:
  - Group membership
  - Locking
  - Publisher/Subscriber
  - Leader Election
  - Synchronization
- Getting node coordination correct is very hard!

ZooKeeper allows distributed processes to coordinate with each other through a shared hierarchical name space of data registers.

Overview – What is ZooKeeper?

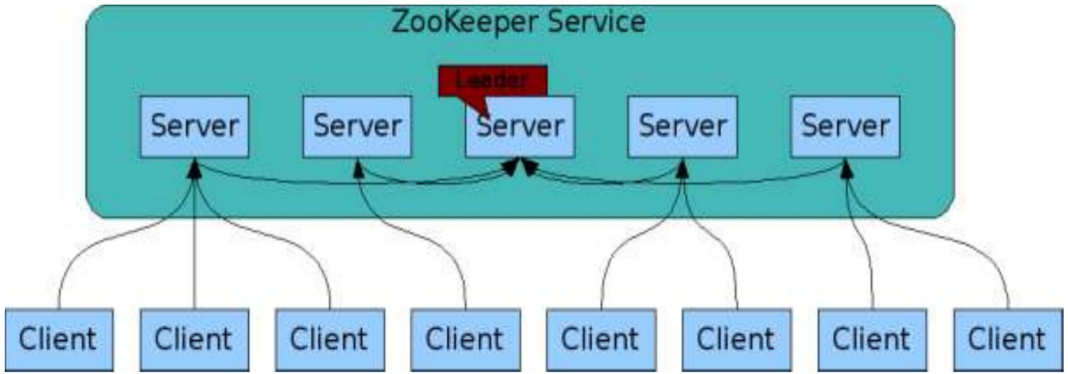
- An open source, high-performance coordination service for distributed application.
- Exposes common services in simple interface:
  - Naming
  - Configuration management
  - Locks & synchronization
  - Groups services
- Build your own on it for specific needs

ZooKeeper Use Cases

- Configuration Management
  - Cluster member nodes bootstrapping configuration from a centralized source in unattended way
- Distributed Cluster Management
  - Node join / leave
  - Node statuses in real time
- Naming service – e.g. DNS
- Distributed synchronization – locks, barriers, queues
- Leader election in a distributed system

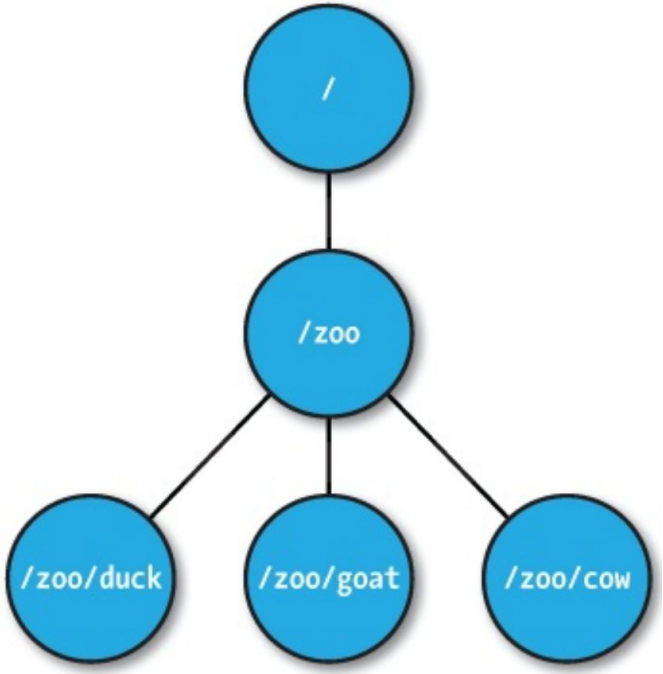
Zookeeper Services

- ZooKeeper Service is replicated over a set of machines
- All machines store a copy of the data (in memory)
- A leader is elected on service startup
- Clients only connect to a single ZooKeeper server & maintains a TCP connection.
- Client can read from any Zookeeper server, writes go through the leader & needs
- majority consensus.



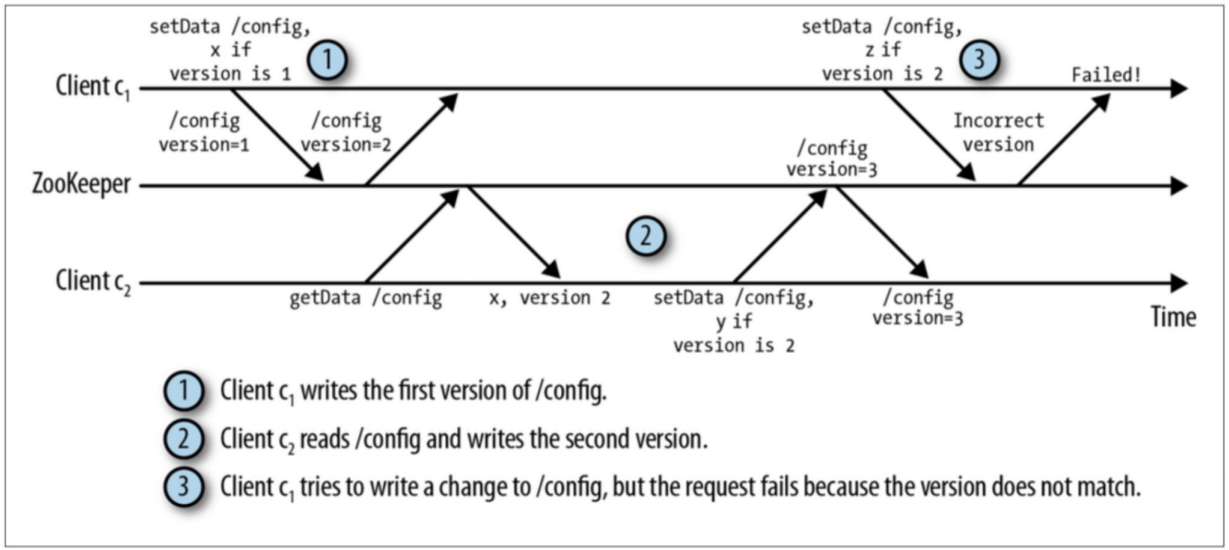
The ZooKeeper Data Model (ZDM)

- Hierarchal name space
- Each node is called as a ZNode
- Every ZNode has data (given as byte[]) and can optionally have children
- ZNode paths:
  - Canonical, absolute, slash-separated
  - No relative references
  - Names can have Unicode characters
- ZNode maintain stat structure



ZDM - Versions

- Eash Znode has version number, is incremented every time its data changes
- setData and delete take version as input, operation succeeds only if client's version is equal to server's one



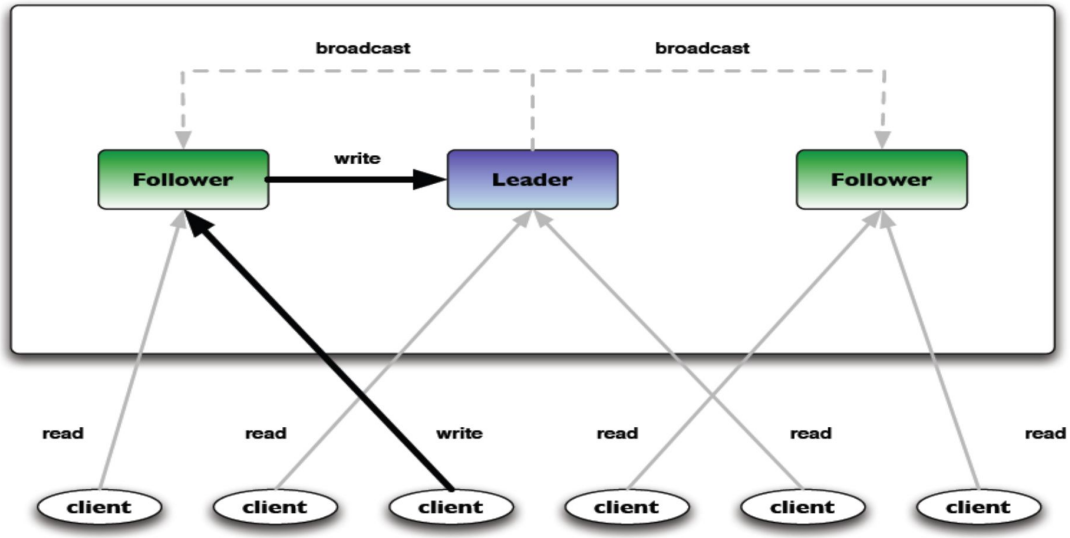
ZDM – Types of ZNode

- Persistent ZNode
  - Have lifetime in ZooKeeper’s namespace until they’re explicitly deleted (can be deleted by delete API call)
- Ephemeral ZNode
  - Is deleted by ZooKeeper service when the creating client’s session ends
  - Can also be explicitly deleted
  - Are not allowed to have children
- Sequential Znode
  - Is assigned a sequence number by ZooKeeper as a part of name during creation
  - Sequence number is integer (4bytes) with format of 10 digits with 0 padding. E.g. /path/to/znode-0000000001

ZDM – Znode Operations

Operation	Description
`create`	Creates a znode in a specified path of the ZooKeeper namespace
`delete`	Deletes a znode from a specified path of the ZooKeeper namespace
`exists`	Checks if a znode exists in the path
`getChildren`	Gets a list of children of a znode
`getData`	Gets the data associated with a znode
`setData`	Sets/writes data into the data field of a znode
`getACL`	Gets the **ACL** of a znode
`setACL`	Sets the **ACL** in a znode
`sync`	Synchronizes a client's view of a znode with ZooKeeper

ZDM – Znode – Reads & Writes



- Read requests are processed locally at the ZooKeeper server to which client is currently connected
- Write requests are forwarded to leader and go through majority consensus before a response is generated

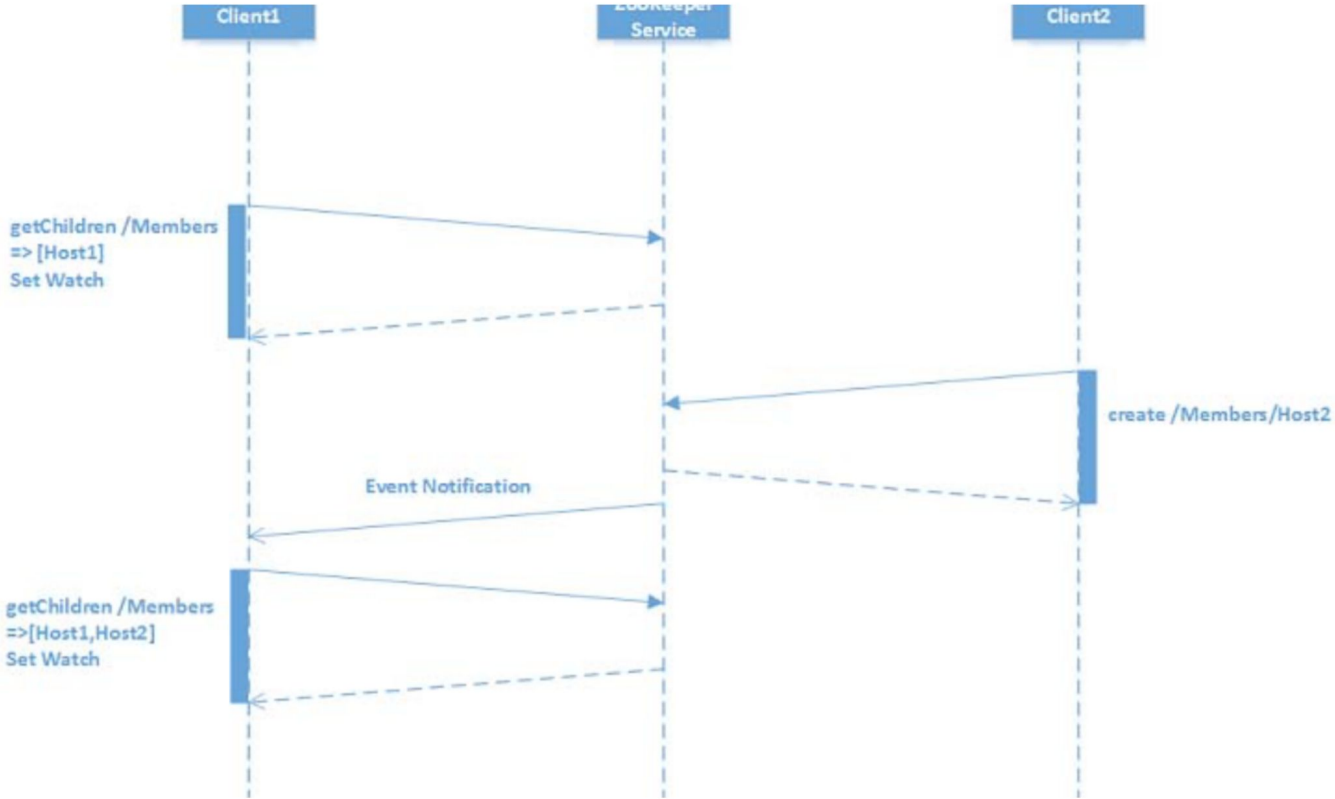
ZDM – Consistency Guarantees

- Sequential Consistency
- Atomicity
- Single System Image
- Reliability
- Timeliness (Eventual Consistency)

ZDM - Watches

- A watch event is one-time trigger, sent to client that set watch, which occurs when data for which watch was set changes.
- Watches allow clients to get notifications when a znode changes in any way (NodeChildrenChanged, NodeCreated, NodeDataChanged,NodeDeleted)
- All of read operations – getData(), getChildren(), exists() have option of setting watch
- ZooKeeper Guarantees about Watches:
    - Watches are ordered, order of watch events corresponds to the order of the updates
    - A client will see a watch event for znode it is watching before seeing the new data that corresponds to that znode

ZDM – Watches (cont)



Quiz

What is Apache Zookeeper?

Explain briefly how Zookeeper model it’s data!

How many types Zookeeper Data Model? and explain!

Explain how zookeeper data model watch works!