hadoop

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models.

It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

- Hadoop was created by **Doug Cutting** and **Mike Cafarella** in 2005. Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant.

Hadoop history

•**Hadoop is based on work done by Google in the early 2000s**
– Specifically, on papers describing the Google File System (GFS)
published in 2003, and MapReduce published in 2004.

•! **This work takes a radical new approach to the problem of distributed computing**
– Meets all the requirements-reliability, scalability etc

•! **Core concept: distribute the data as it is initially stored in the system**
– Individual nodes can work on data local to those nodes
– No data transfer over the network is required for initial Processing.

Hadoop

▪ Hadoop is the practical implementation in Java of Google's MapReduce model.

▪ Hadoop is open-source, developed by Yahoo!, now distributed by the Apache Foundation.

▪ A software "industry" has grown up around Hadoop

- Hadoop is now supplemented by a range of Cloud software projects, such as Pig, Hive and Zookeeper, etc. Most of these are also distributed by Apache.

Hadoop

- Hadoop is a MapReduce software platform.
    - Provides a framework for running MapReduce applications.
    - This framework understands and assigns work to the nodes in a cluster.
    - Handles the mapping and reduc(e)ing logistics.

- Currently takes custom functionality written in Java or Python.

- Can use an open-source Eclipse plug-in to interface with Hadoop.

- Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of commodity computers using a simple programming model.

- It is designed to scale up from single servers to thousands of machines, each providing computation and storage.

- Rather than rely on hardware to deliver high-availability, the framework itself is designed to detect and handle failures at the application layer, thus delivering a highly-available service on top of a cluster of computers.

Hadoop Architecture

Hadoop framework includes following four modules:

1. **Hadoop Common:** These are Java libraries and utilities required by other Hadoop modules. These libraries provides filesystem and OS level abstractions and contains the necessary Java files and scripts required to start Hadoop.

2. **Hadoop YARN(**Yet Another Resource Negotiator)**:** This is a framework for job scheduling and cluster resource management.

3. **Hadoop Distributed File System (HDFS):** A distributed file system that provides high-throughput access to application data.

4. **Hadoop MapReduce:** This is YARN-based system for parallel processing of large data sets.

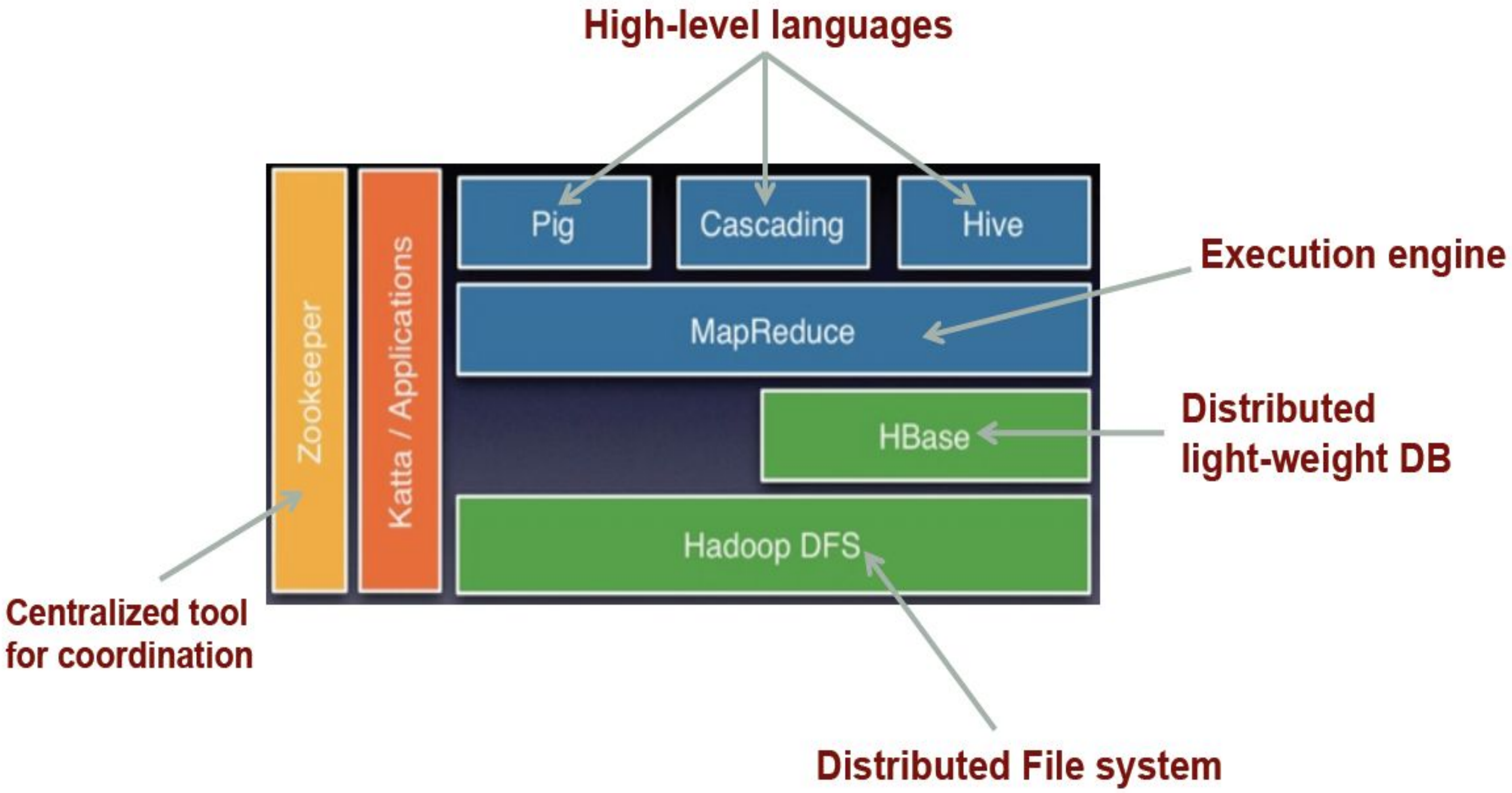
Hadoop Components

- **Hadoop consists of two core components**
  - The Hadoop Distributed File System (HDFS)
  - MapReduce Software Framework

- **There are many other projects based around core Hadoop**
  - Often referred to as the 'Hadoop Ecosystem'
  - Since 2012, the term "Hadoop" often refers not just to the base modules mentioned above but also to the collection of additional software packages that can be installed on top of or alongside Hadoop, such as Apache Pig, Apache Hive, Apache HBase, Apache Spark , Flume, Oozie, Sqoop, etc

- **A set of machines running HDFS and MapReduce is known as a Hadoop Cluster**
  - Individual machines are known as nodes
  - More nodes = better performance!

Hadoop: Big Picture



HDFS + MapReduce are enough to have things working

Who uses Hadoop?

- Amazon/A9
- Facebook
- Google
- New York Times
- yahoo!
- …. many more

A large ecosystem

Who uses Hadoop?

IBM

VISA

hp invent

SAMSUNG

The New York Times

facebook

SK telecom

YAHOO!

eHarmony

intel

rackspace HOSTING

comScore

Visible MEASURES

NETFLIX

NING

BERKELEY LAB

University of Phoenix

ebay

Autodesk

contextWeb

twitter

amazon.com

GROUPON

JPMORGAN CHASE & CO.

NAVTEQ

Aol Advertising.

Core Hadoop Concepts

! **Applications are written in high-level code**
– Developers do not worry about network programming, temporal(time-based) dependencies etc.

! **Nodes talk to each other as little as possible**
– Developers should not write code which communicates between nodes
– 'Shared nothing' architecture

! **Data is spread among machines in advance**
– Computation happens where the data is stored, wherever possible
– Data is replicated multiple times on the system for increased availability and reliability.

Fault Tolerance

! If a node fails, the master will detect that failure and re-assign the work to a different node on the system.

! Restarting a task does not require communication with nodes working on other portions of the data.

! If a failed node restarts, it is automatically added back to the system and assigned new tasks.

Advantages of Hadoop

- ✔Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.

- ✔Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.

- ✔Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.

- ✔Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

HDFS

- **Hadoop File System** was developed using distributed file system design. It runs on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.

- HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines.

- These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

HDFS

- Hadoop makes use of HDFS for data storage - the file system that spans all the nodes in a Hadoop cluster.

- It links together the file systems on many local nodes to make them into one big file system.

- HDFS assumes nodes will fail, so it achieves reliability by replicating data across multiple nodes.

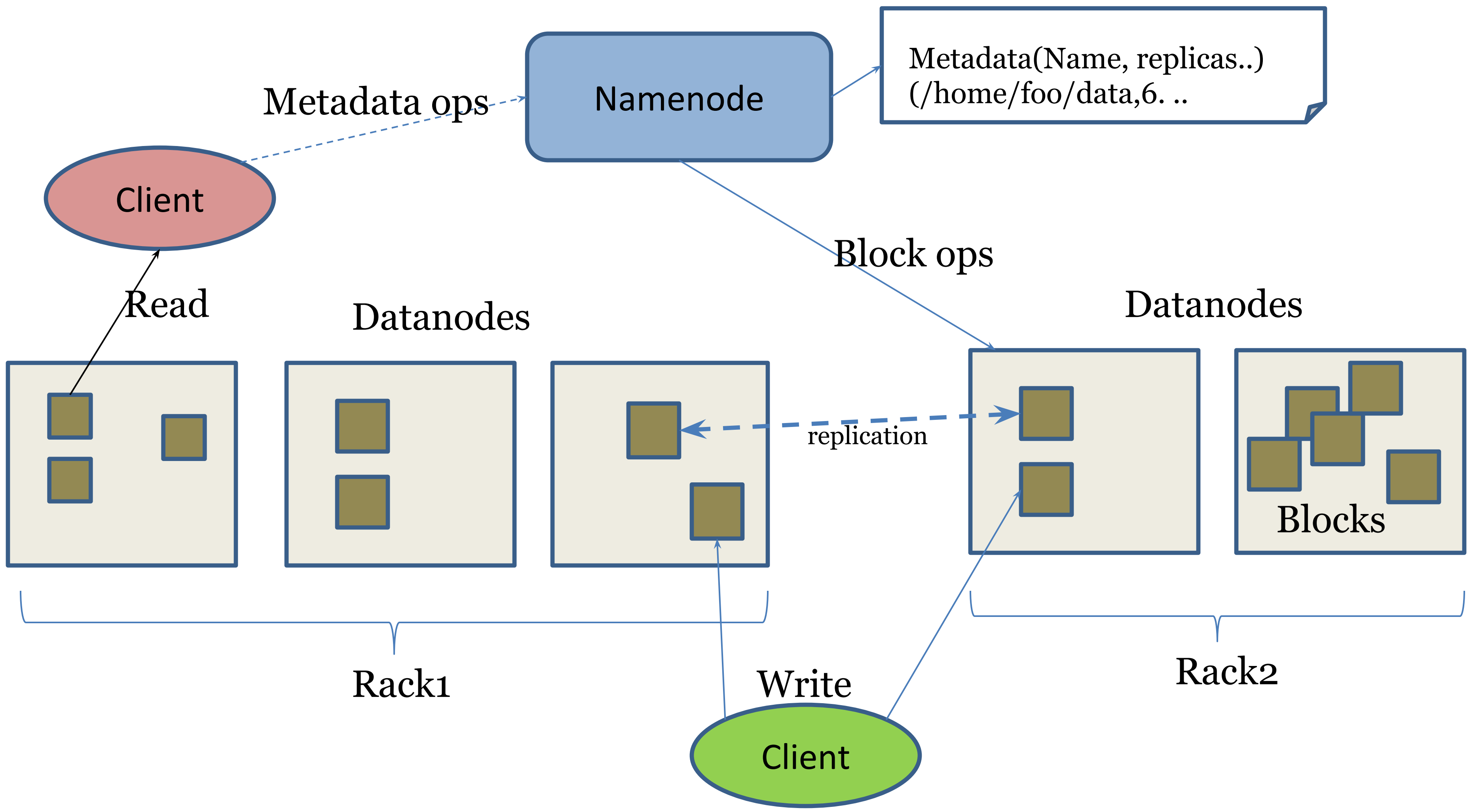
Basic Features: HDFS

- Highly fault-tolerant
  - Can handle disk crashes, machine crashes, etc...
- High throughput
- Suitable for applications with large data sets
- Can be built out of commodity hardware
- Based on Google's Filesystem GFS
- It is suitable for the distributed storage and processing.
- Hadoop provides a command interface to interact with HDFS.
- The built-in servers of namenode and datanode help users to easily check the status of cluster.
- HDFS provides file permissions and authentication.

Fault tolerance

- A HDFS instance may consist of thousands of server machines, each storing part of the file system’s data.
- Since we have huge number of components and that each component has non-trivial probability of failure means that there is always some component that is non-functional.
- Detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.

HDFS Architecture



HDFS Daemons

- **Filesystem cluster is managed by three types of processes**

– **Namenode**

- manages the File System's namespace/meta-data/file blocks
- Runs on 1 machine to several machines

– **Datanode**

- Stores and retrieves data blocks
- Reports to Namenode
- Runs on many machines

– **Secondary Namenode**

- Performs house keeping work so Namenode doesn’t have to require similar hardware as Namenode machine
- Not used for high-availability – not a backup for Namenode

Namenode

- **Master/slave architecture**
- HDFS cluster consists of a single **Namenode**, a master server that manages the file system namespace and regulates access to files by clients.

- **Functions of a NameNode:-**

✔ Regulates client's access to files.

✔ It also executes file system operations such as renaming, closing, and opening files and directories.

✔ Manages File System Namespace
  - Maps a file name to a set of blocks
  - Maps a block to the DataNodes where it resides

✔ Cluster Configuration Management

✔ Replication Engine for Blocks

Datanode

- There are a number of **DataNodes** usually one per node in a cluster.
- These nodes manage the data storage of their system.
- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

- The DataNodes manage storage attached to the nodes that they run on.
- HDFS exposes a file system namespace and allows user data to be stored in files.
- A file is split into one or more blocks and set of blocks are stored in DataNodes.
- DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

Block

- Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes.

- These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a **Block**.

- The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

- When the filesystem starts up it generates a list of all HDFS blocks and send this report to Namenode: **Blockreport.**

- **Block Report**
  - Periodically sends a report of all existing blocks to the NameNode

- A Block Server
  - Stores data in the local file system
  - Stores metadata of a block
  - Serves data and metadata to Clients

- Facilitates Pipelining of Data

  -Forwards data to other specified DataNodes

File system Namespace

- Hierarchical file system with directories and files
- Create, remove, move, rename etc.
- Namenode maintains the file system
- Any meta information changes to the file system recorded by the Namenode.
- An application can specify the number of replicas of the file needed: replication factor of the file. This information is stored in the Namenode.

Data Replication

- HDFS is designed to store very large files across machines in a large cluster.
- Each file is a sequence of blocks.
- Blocks are replicated for fault tolerance.
- Block size and replicas are configurable per file.
- The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster.
- BlockReport contains all the blocks on a Datanode.

Filesystem Metadata

- The HDFS namespace is stored by Namenode.
- Namenode uses a transaction log called the EditLog to record every change that occurs to the filesystem meta data.
  - For example, creating a new file.
  - Change replication factor of a file
  - EditLog is stored in the Namenode's local filesystem
- Entire filesystem namespace including mapping of blocks to files and file system properties is stored in a file **FsImage**. Stored in Namenode's local filesystem.

- Types of metadata
  - List of files
  - List of Blocks for each file
  - List of DataNodes for each block
  - File attributes, e.g. creation time, replication factor

- A Transaction Log
  - Records file creations, file deletions etc

Block Placement

- Current Strategy
  - One replica on local node
  - Second replica on a remote rack
  - Third replica on same remote rack
  - Additional replicas are randomly placed
- Clients read from nearest replicas

Heartbeats

- DataNodes send hearbeat to the NameNode
  - Once every 3 seconds
- NameNode uses heartbeats to detect DataNode failure

Replication Engine

- NameNode detects DataNode failures
  - Chooses new DataNodes for new replicas
  - Balances disk usage
  - Balances communication traffic to DataNodes

Secondary NameNode

- It is responsible for performing periodic checkpoints.So if the Namenode fails it can be replaced with a snapshot image stored by the Secondary Namenode checkpoints.
- Copies FsImage and Transaction Log from Namenode to a temporary directory.
- Merges FSImage and Transaction Log into a new FSImage in temporary directory.
- Uploads new FSImage to the NameNode
  - Transaction Log on NameNode is eliminate.

Goals of HDFS

- **Fault detection and recovery** : Since HDFS includes a large number of commodity hardware, failure of components is frequent. Therefore HDFS should have mechanisms for quick and automatic fault detection and recovery.

- **Huge datasets** : HDFS should have hundreds of nodes per cluster to manage the applications having huge datasets.

- **Hardware at data** : A requested task can be done efficiently, when the computation takes place near the data. Especially where huge datasets are involved, it reduces the network traffic and increases the throughput.

HDFS is Good for...

• **Storing large files**

– Terabytes, Petabytes, etc...

– 100MB or more per file

• **Streaming data**

– Write once and read-many times patterns

– Optimized for streaming reads rather than random reads

• **“Cheap” Commodity Hardware**

– No need for super-computers, use less reliable
commodity hardware

User Interface

- Commads 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

MapReduce

- MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner.

- MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

- Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

- **Map stage** : The map or mapper’s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.

- **Reduce stage** : This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.

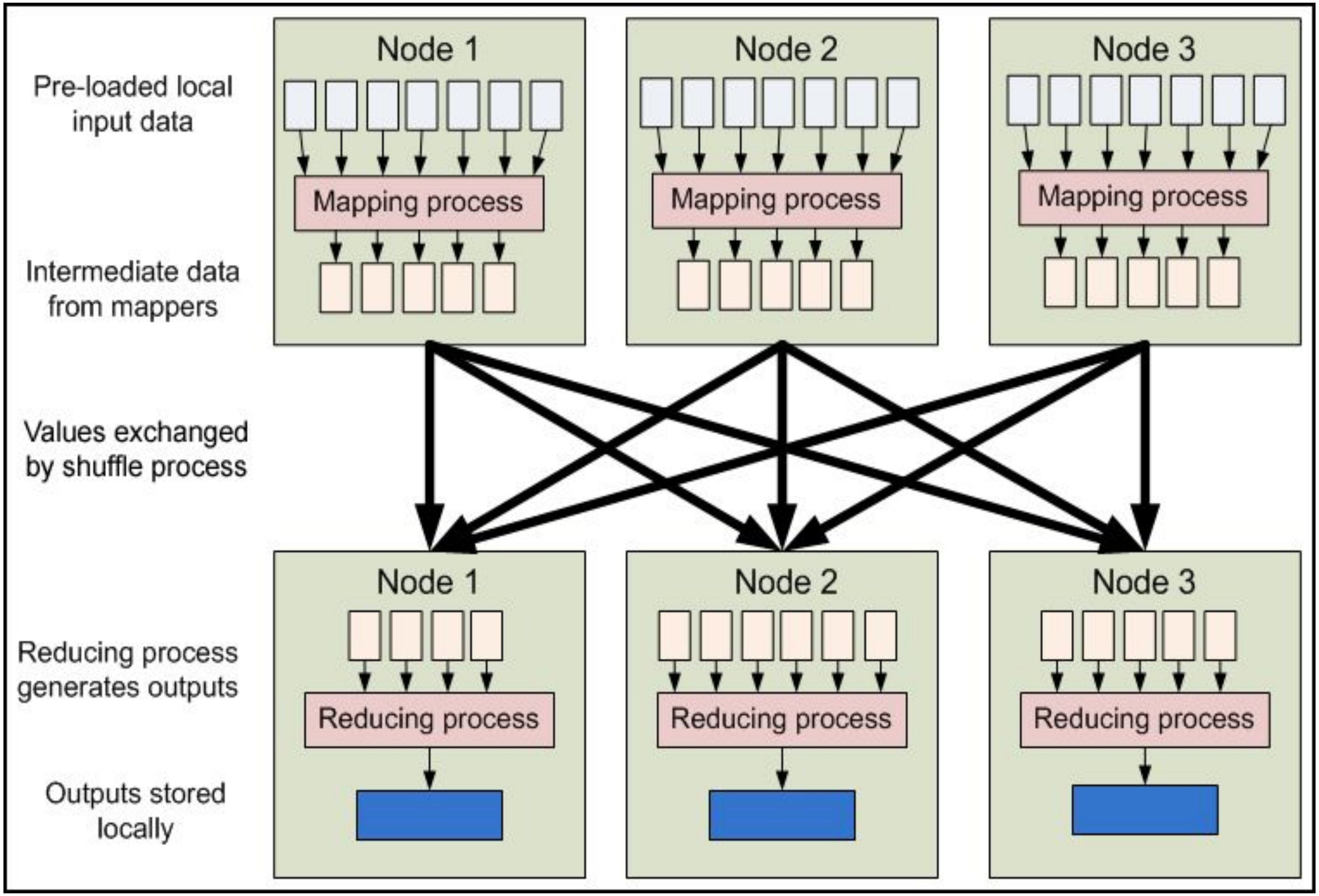
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.

- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.

- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.

- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

MapReduce - Dataflow



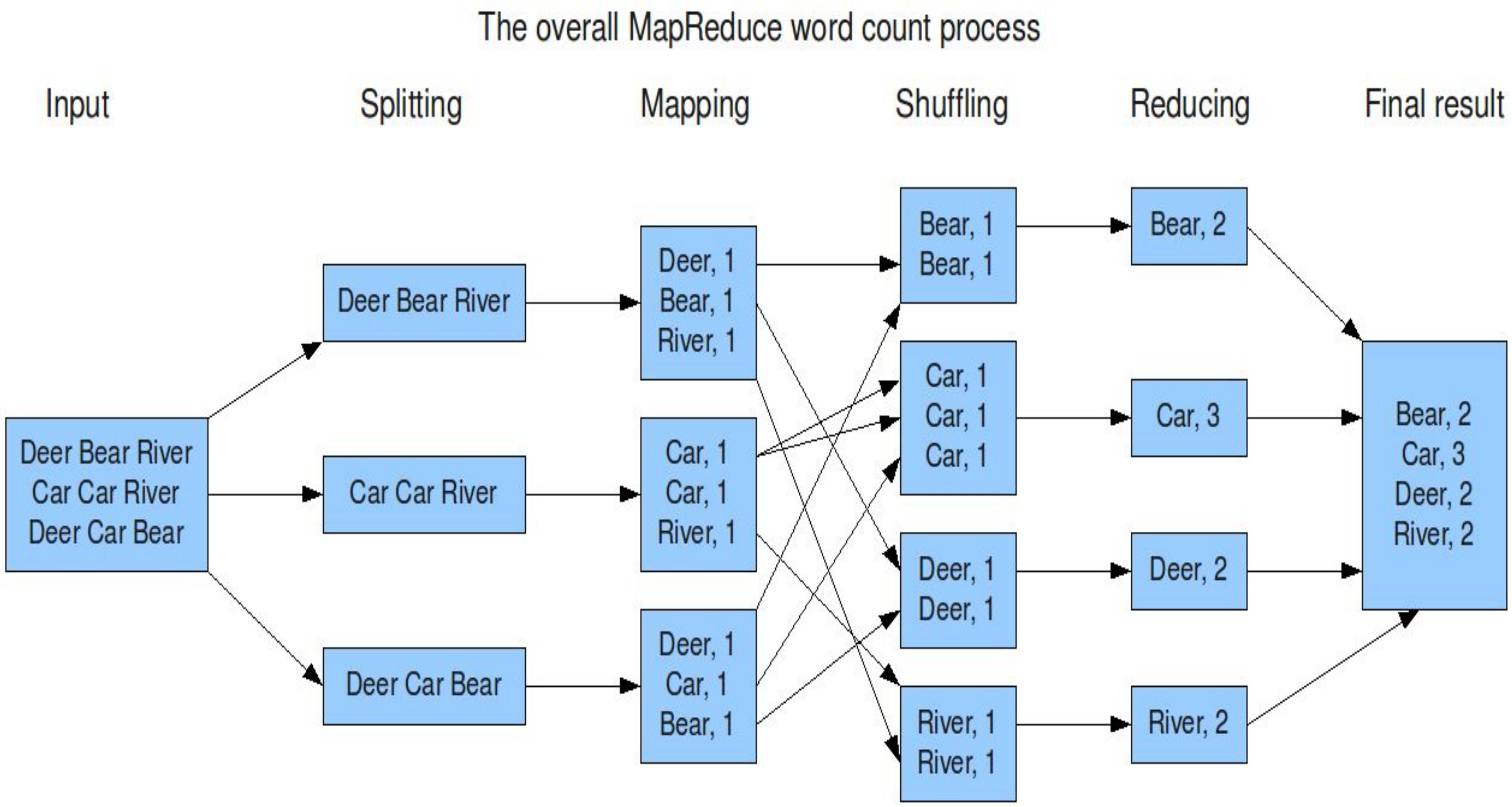
MapReduce - What?

- MapReduce is a programming model for efficient distributed computing
- It works like a Unix pipeline
  - **Input** | **Map** | Shuffle & Sort | **Reduce** | **Output**
- Efficiency from
  - Streaming through data
  - Pipelining
- A good fit for a lot of applications
  - Log processing
  - Web index building

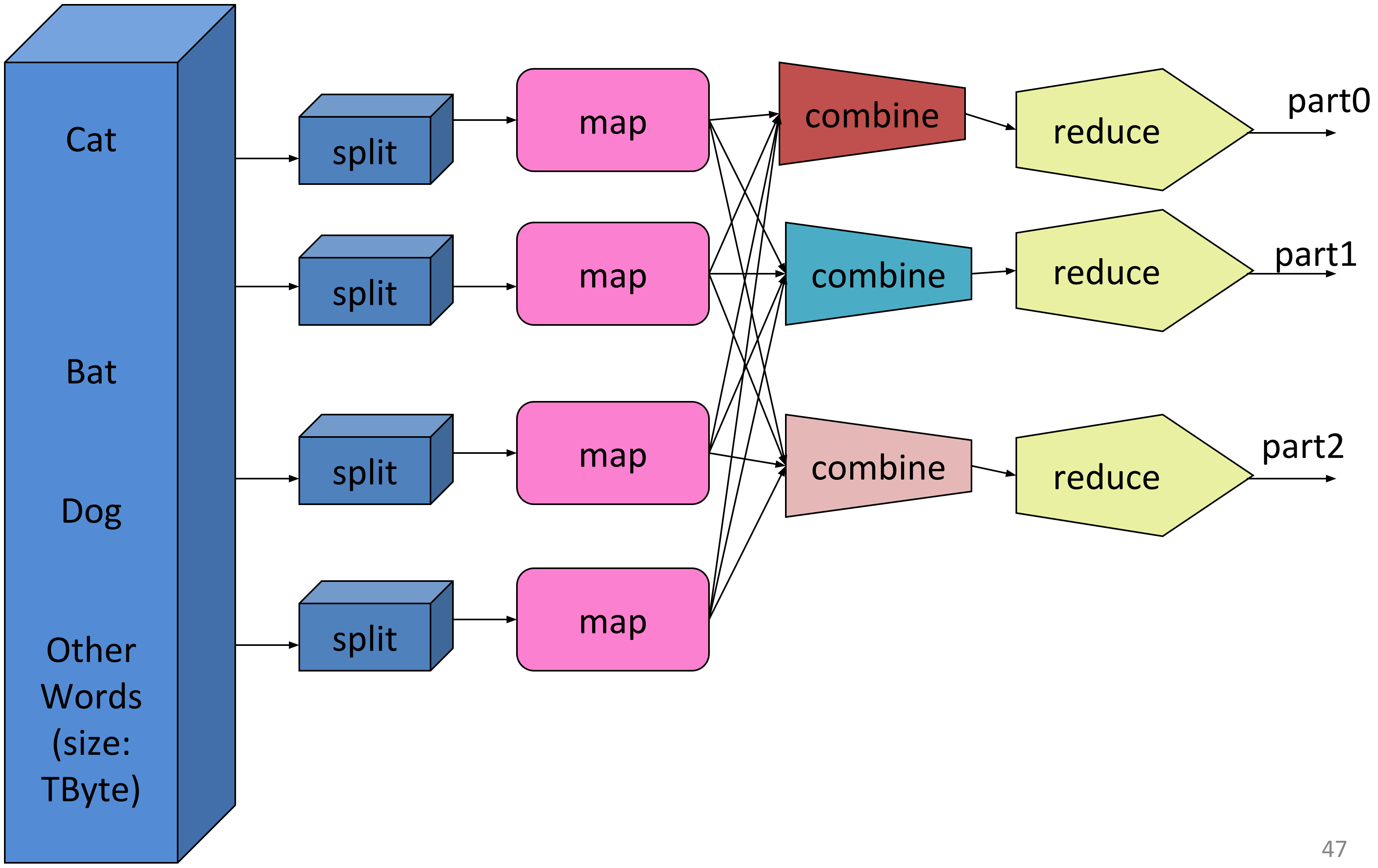
Word Count Example

- 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

Word Count Dataflow



MapReduce


Cat
Bat
Dog
Other Words (size: TByte)
split
split
split
split
map
map
map
map
combine
combine
combine
reduce
reduce
reduce
part0
part1
part2

References

- http://www.hadoopadmin.co.in/hadoop-administrator/mapreduce/
- https://www.tutorialspoint.com/hadoop/index.htm

END