MapReduce - Simplified Data Processing on Large Clusters

Download the paper if you haven't already.

- Salman Shah (@salman-bhai)
Web Enthusiasts' Club && IEEE NITK

Today’s Co-Speakers

Utkarsh Patil
@utkarsh23
Web Enthusiasts’ Club

Anumeha Agrawal
@anumehaagrawal
Web Enthusiasts’ Club

Today's Plan

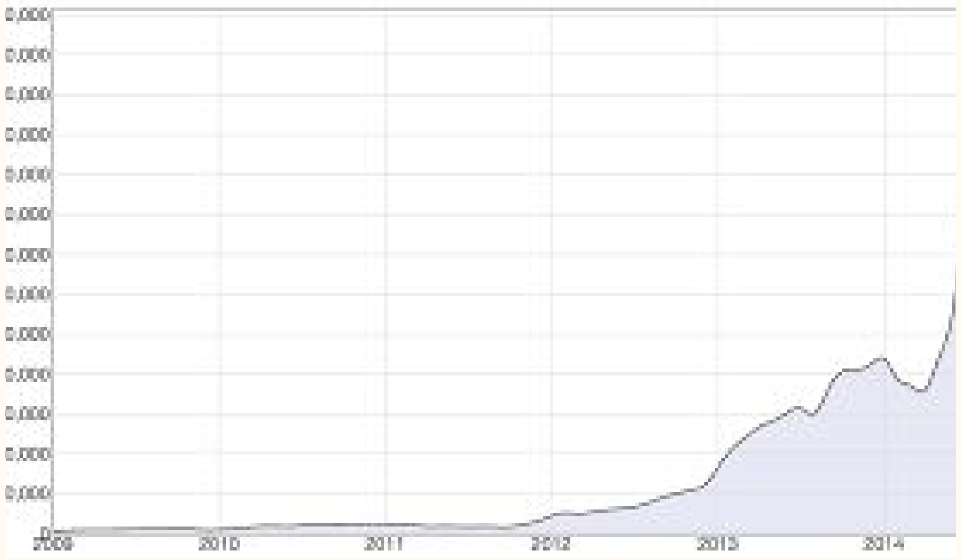
What all will we do in today's session?

- Introduction to MapReduce & Big Data
- MapReduce Architecture
- How to run code for MapReduce - Github Example

Introduction to MapReduce & Big Data

Big Data

1. Can be defined simply as “Data that is too big to be processed on a single machine”
2. Formal Definition can be given as “Big data is high **volume**, high **velocity**, and/or high **variety** information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization”



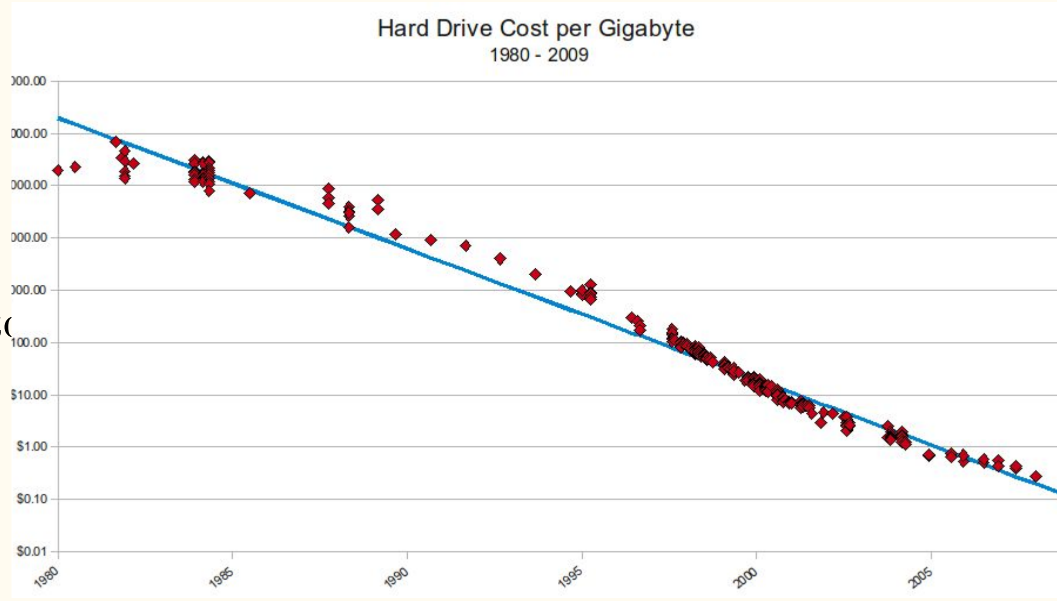
3 V’s of Big Data
Volume | Variety | Velocity

Volume

1. Volume refers to the size of the data that you are dealing with.

2. Price of storing data has dropped drastically from $100k/GB in 1980 t
$0.10/GB in 2013.

3. Storing data is as important as reading it and processing it efficiently.



Variety

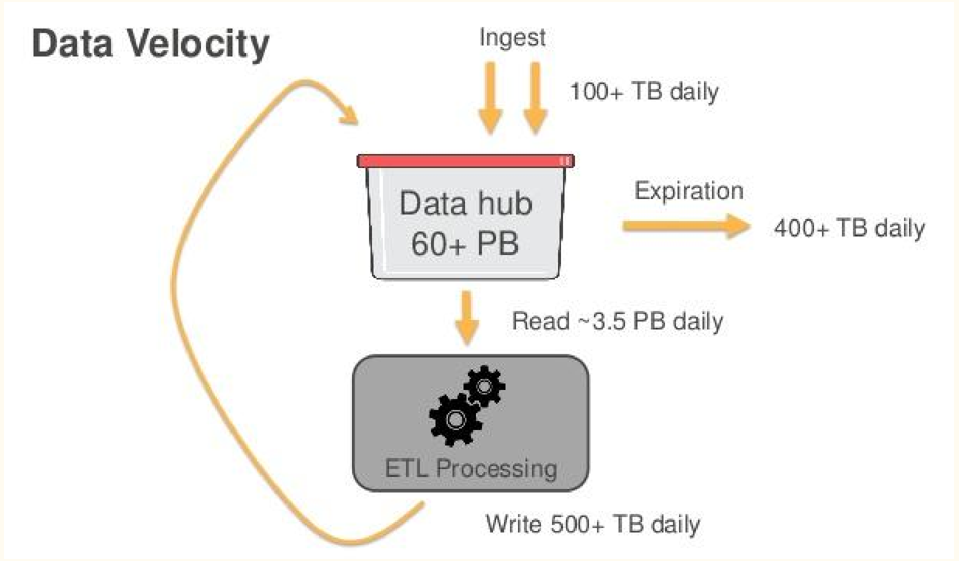
1. Variety refers to the fact that data comes from different sources and in different formats.

2. A lot of data that was earlier stored was in a pre-defined format such as tables in MySQL or Data Warehouses like Oracle. However data today is highly unstructured.

3. Example: Call Center, information stored in text as well as a MP3 recording. Thus several types of data formats can co-exist at the same time and these need to be stored.

Velocity

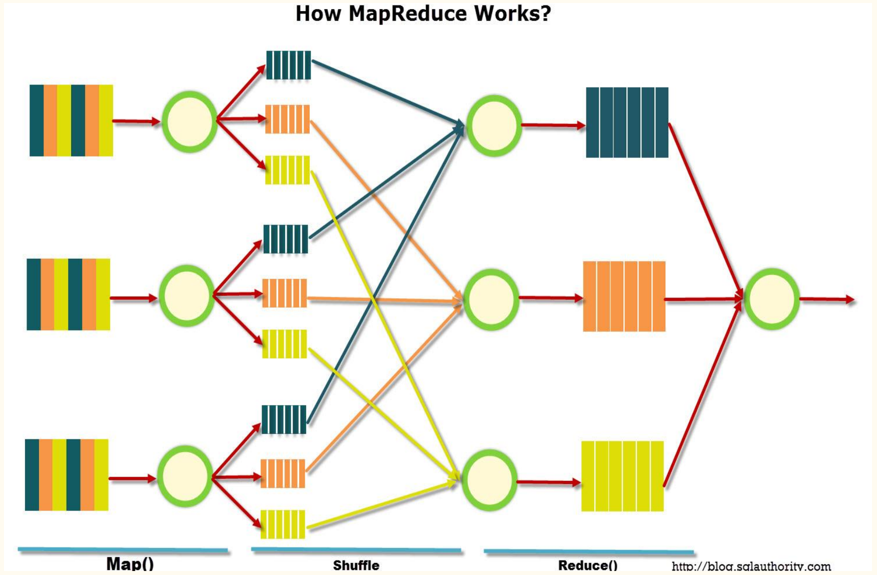
1. Velocity refers to speed at which data is being generated and speed at which it is made available.
2. Data needs to be able to be accepted and stored even if it is coming at the rate of several TBs/day.
3. If we can't store as it arrives, then we'll end up discarding it and it'll be of no use.
4. Example: Netflix gives users recommendations based on users previous data.



Map Reduce

1. MapReduce paradigm has created a huge impact in the Big Data world. Many systems including Hadoop MapReduce and Spark were developed using this paradigm.

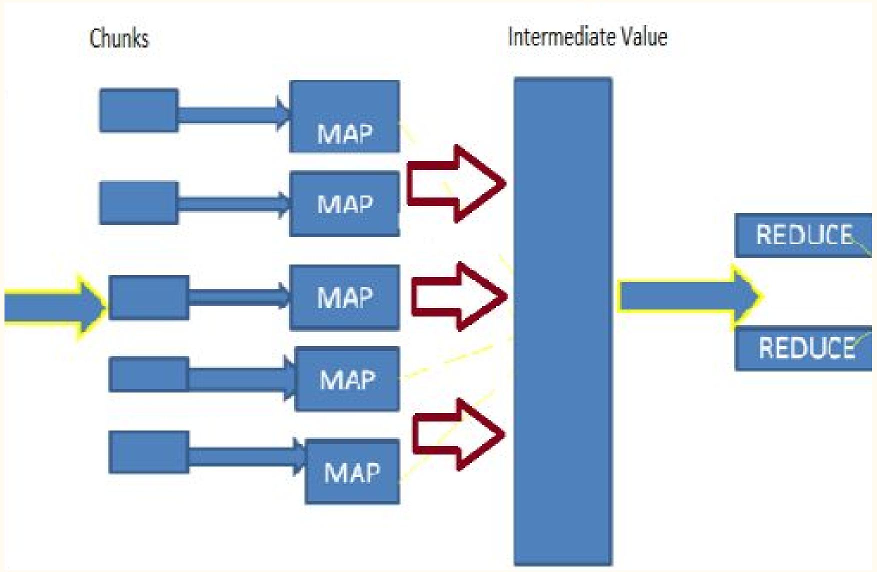
2. MapReduce was conceptualized to handle cases where computation to be performed was straightforward but parallelizing the computation and taking care of other aspects of distributed computing was a pain.



Why we need MapReduce?

1. If we had a big file, processing it serially from top to bottom would take a really long time.

2. However if the file were to be divided into chunks and then each chunk was to be processed in parallel, it would significantly reduce all our work.



MapReduce Architecture

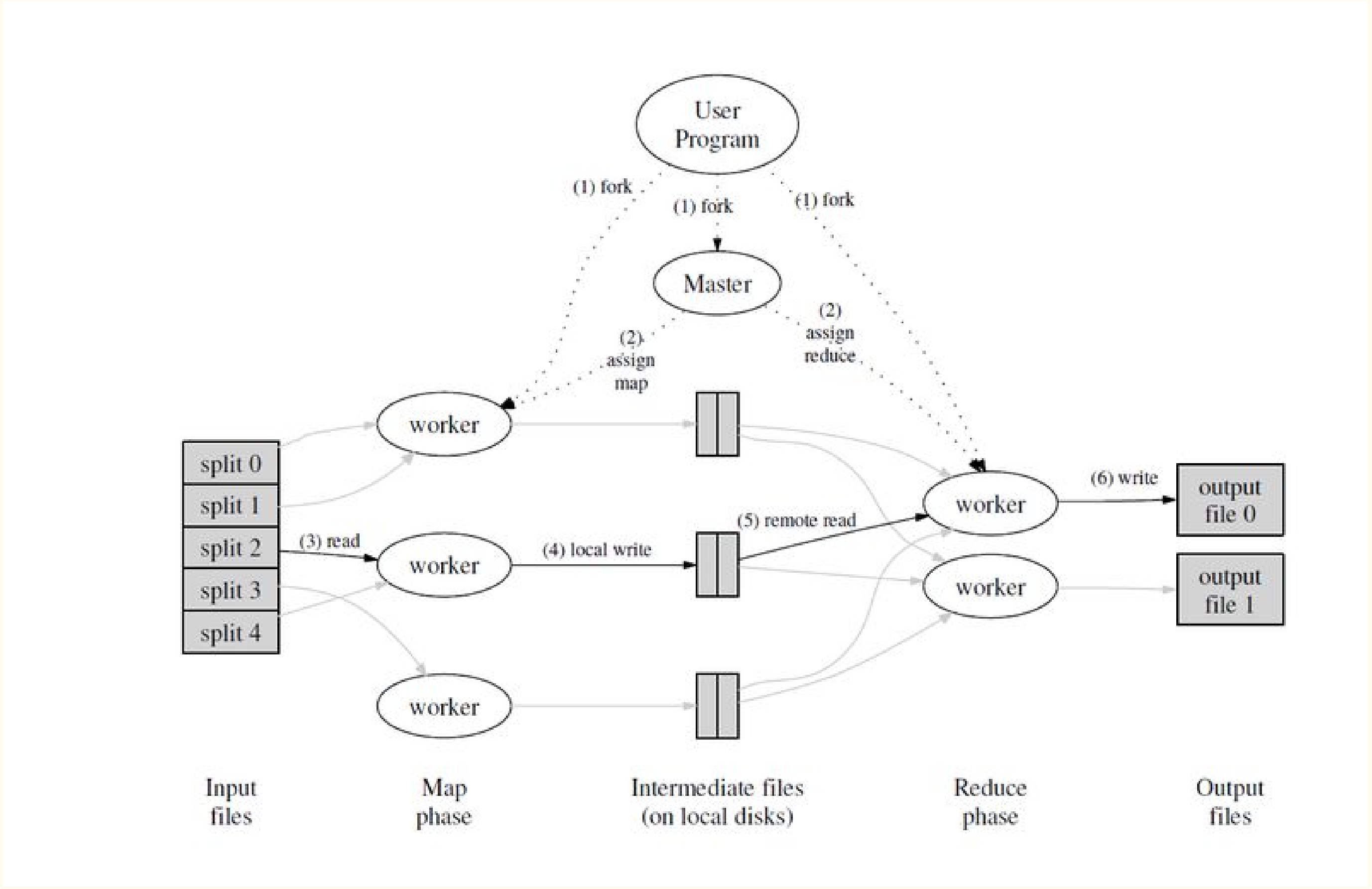
Implementation

How the Framework actually works

- Execution Overview
- Data Structures Used
- Fault Tolerance
- Locality
- Backup Tasks

Execution Overview

Ideally MapReduce allows the **Map invocation** to partition the input data into *M splits* and the **Reduce invocation** to partition the intermediate key space into *R partitions*.
The **final output** is combined from those *R partitions* into one combined output file.



Master assigning work to each of the worker nodes to either **execute Map or Reduce Task**
Reduce Task is assigned by master **only when there is an intermediate input** to be taken

Master Data Structures

1. For each map task and reduce task, the master stores the state (idle, in-progress, or completed), and the identity of the worker machine (for non-idle tasks).
2. For each completed map task, the master stores the locations and sizes of the R intermediate file regions produced by the map task. Updates are made to this region as and when the tasks are completed.

Fault Tolerance

Worker Failure

What happens when the Worker node fails?

Master pings worker periodically. If there is no response received from the worker the machine is down and the Master will reallocate the work to some other node.

The entire task is re-executed even if the failed node has executed 99.99% of the task as the output is stored in the local disk of the failed machine and it is inaccessible.

MapReduce allows for large scale worker failures. **Why?**

Master Node Failure

What happens when the Master Node fails?

In the previous implementation of MapReduce(mentioned in the paper), if the master dies then the entire process is aborted and and users can retry the MapReduce operation if necessary.

Locality - Managing the Network for MapReduce

Locality

We conserve network bandwidth by taking advantage of the fact that the input data is stored on the local disks of the machines that make up our cluster. We divide each file into 64 MB blocks, and stores several copies of each block (typically 3 copies) on different machines.

The MapReduce master takes the location information of the input files into account and attempts to schedule a map task on a machine that contains a replica of the corresponding input data.

Locality

Failing that, it attempts to schedule a map task near a replica of that task's input data (e.g., on a worker machine that is on the same network switch as the machine containing the data).

When running large MapReduce operations on a significant fraction of the workers in a cluster, most input data is read locally and consumes no network bandwidth.

Backup Tasks

Stragglers

One of the common problems that we face with MapReduce that increases the time for computation is with “Stragglers”, or simply put machines that take a long time to complete one of the last few computations.

Example: Machine with a bad disk that may read a file *very very slowly*. A machine with a capacity to read input from disk at 30MB/s may drop to 1MB/s for a host of reasons.

Solution to Stragglers

Whenever the task is close to completion the Master node schedules one other worker node (referred as the backup node), to complete the remaining *in-progress* tasks.

The task is then marked complete when either the primary or the backup node complete executing the task.

These are commonly referred to as **Backup tasks**.

Refinements

Certain refinements that can make MapReduce better

- Combiner Function
- Skipping Bad Records
- Local Execution
- Counters

Applications of MapReduce (Examples)

1. **Distributed Grep:** The map function emits a line if it matches a supplied pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.
2. **Count of URL Access Frequency:** The map function processes logs of web page requests and outputs hURL, 1i. The reduce function adds together all values for the same URL and emits a hURL, total counting pair.
3. **Reverse Web-Link Graph:** The map function outputs htarget, sourcei pairs for each link to a target URL found in a page named source. The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: htarget, list(source)i

Applications

Applying MapReduce for Real-Life Problems at Google

- Large Scale Machine Learning Problems
- Clustering problems for the Google News
- Extraction of data used to produce reports of popular queries
- Extraction of properties of web pages for new experiments and products

MapReduce Coding Example

A simple Python program to run MapReduce on your local system

https://github.com/WebClub-NITK/MapReduce

Hadoop

Apache Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model.

Hadoop Installation

Hadoop Installation on Ubuntu for Single Node clusters can be done via the following link

http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

Hadoop Demonstration

Dataset with over

Conclusion

- First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing.
- A large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production web search service, for sorting, for data mining, for machine learning, and many other systems.
- MapReduce scales to large clusters of machines comprising of thousands of machines and all can be used at the same time.