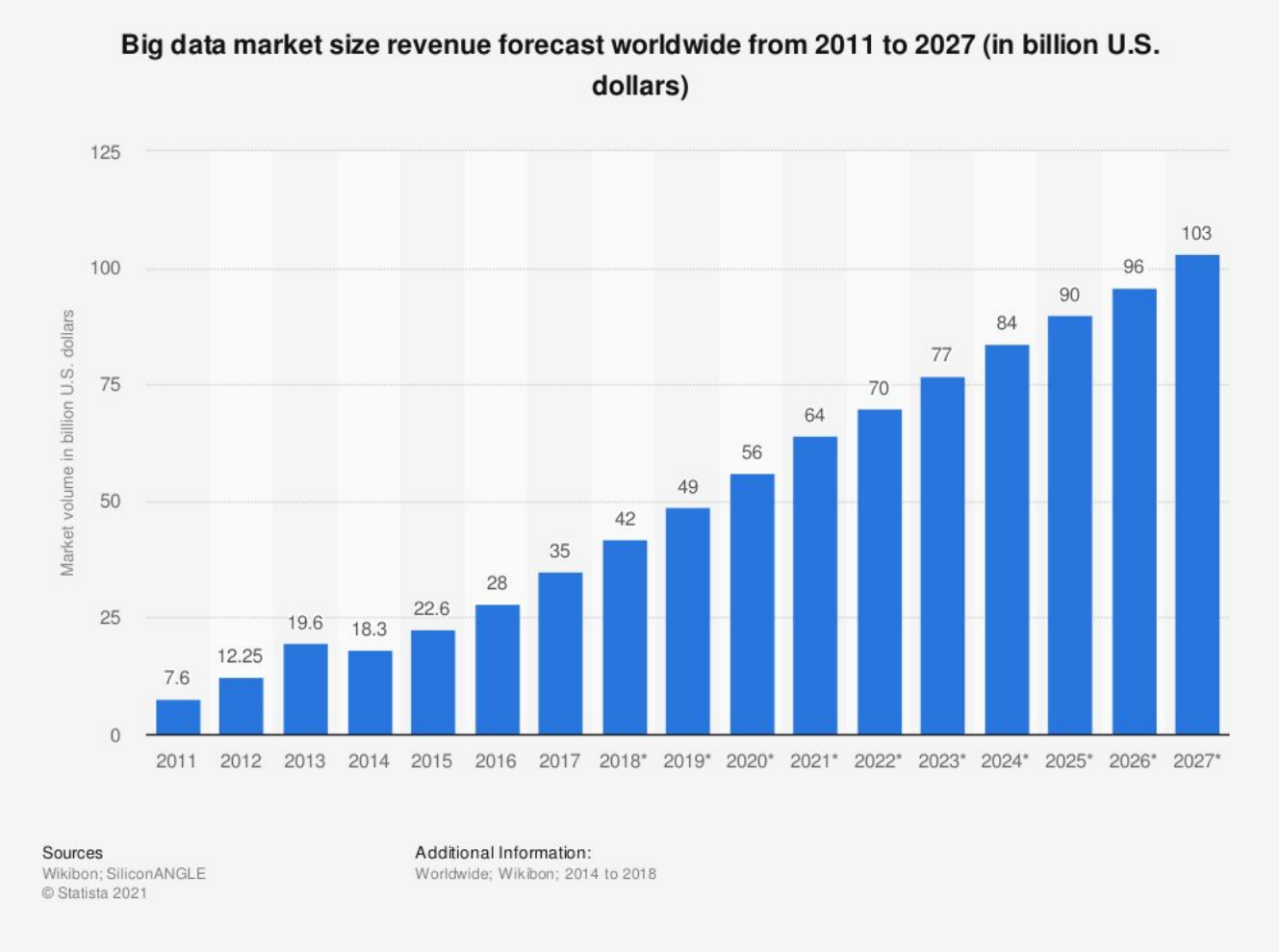
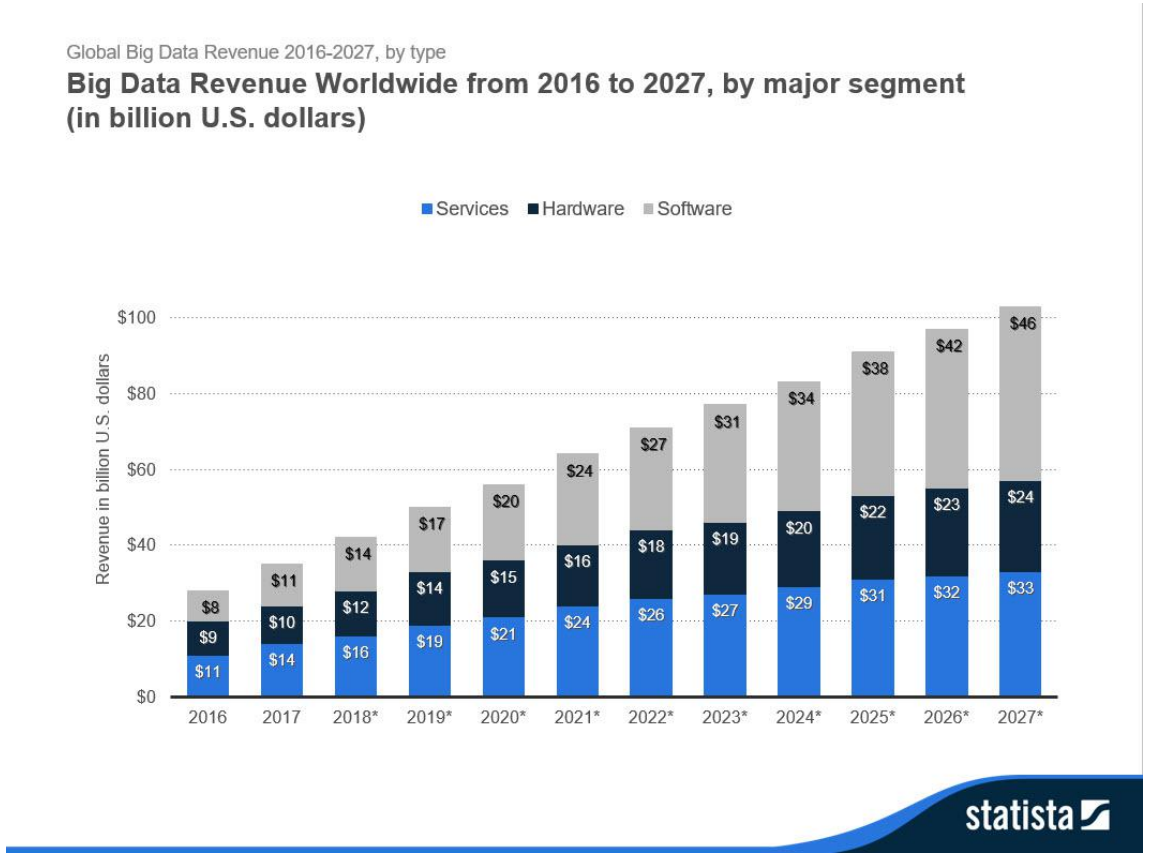
Skill Lab , V Sem

Introduction to Big Data Analytics

Introduction and Motivation





Introduction and Motivation

Key Factors of Machine Learning

1. Large Data Sets

Millions of labelled images, *thousands of hours* of speech

2. Improved Models and Algorithms

- Deep Neural Networks: *hundreds* of layers, *millions* of parameters

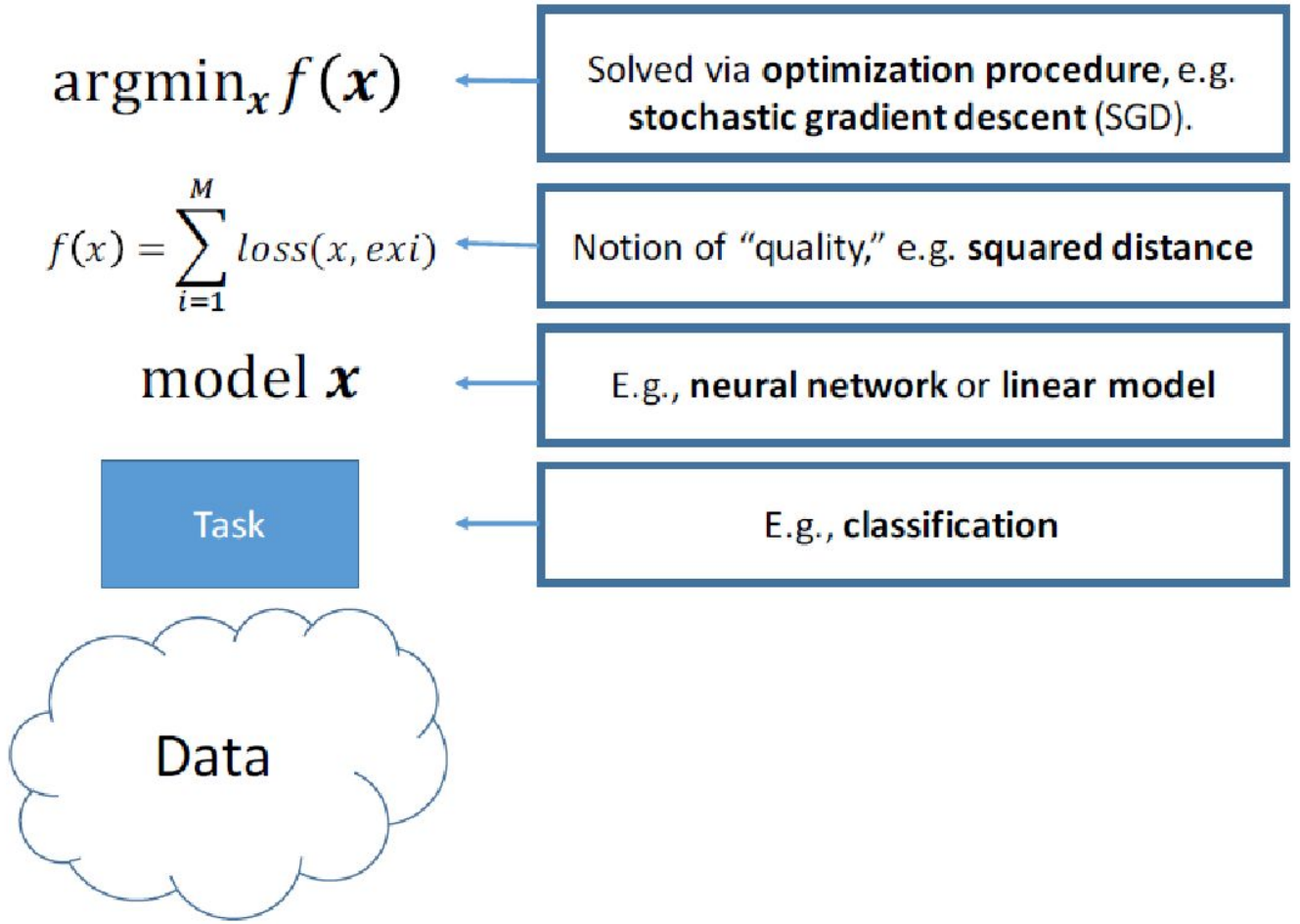
3. Efficient Computation for Machine Learning:

- Computational power for ML increased by ~100x since 2010
- Gains (GPU, CPU) almost stagnant in latest generations
- Computation times are extremely large anyway (days to weeks to months)

Solution: Distribute Machine Learning Applications to Multiple Processors and Nodes

Introduction and Motivation

Machine Learning in One Node



Distributed Machine Learning

$$\text{argmin}_x f(x) = f_1(x) + f_2(x)$$

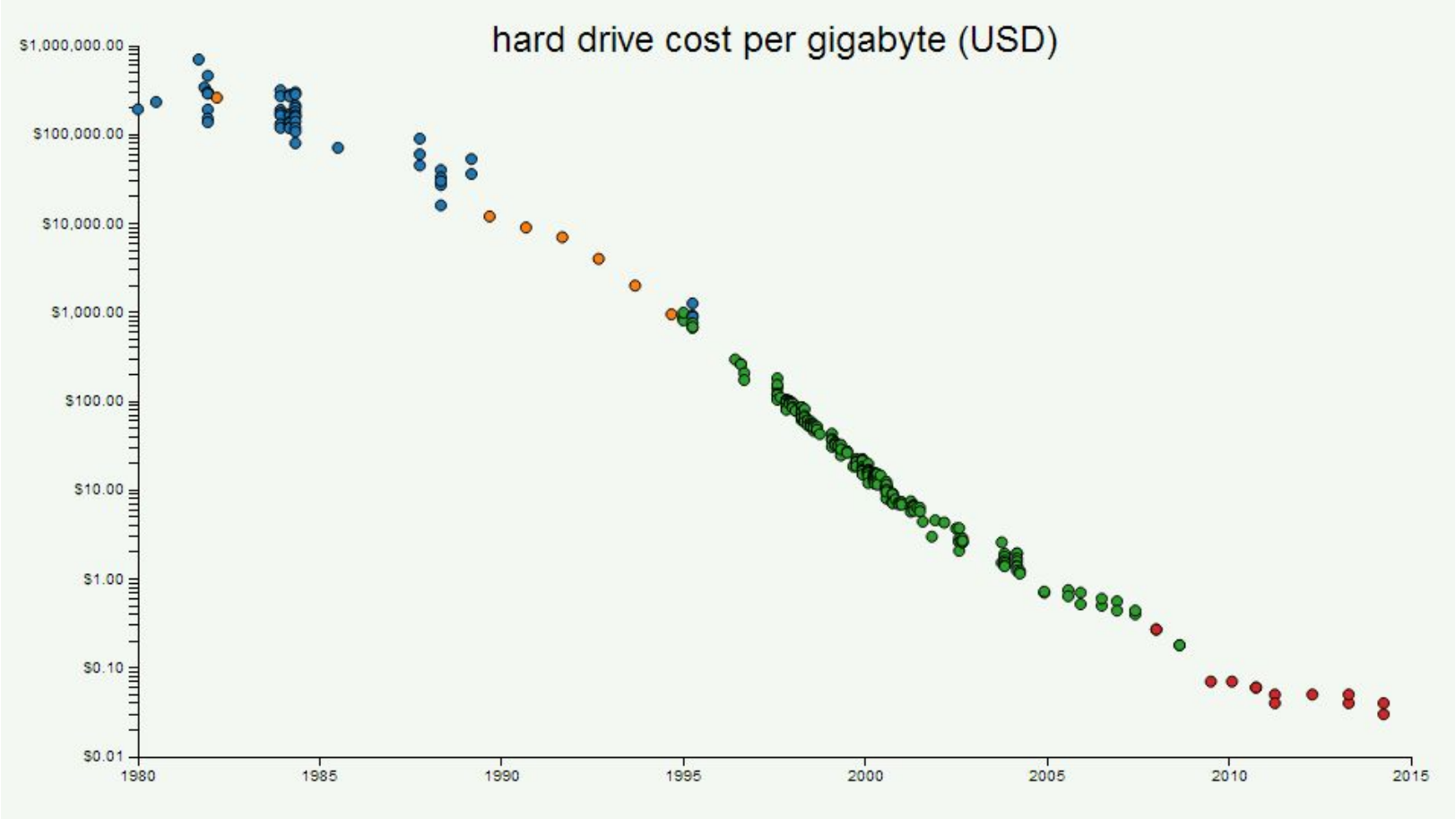
$$f_1(x) = \sum_{i=1}^{M/2} l(x, ei) \qquad f_2(x) = \sum_{i=\frac{M}{2}+1}^{M} l(x, ei)$$

Hadoop Overview

- Motivation for Hadoop
- HDFS
- Map Reduce
- The Hadoop Ecosystem

Slides credits: ClouderaAcademic Partners Program & Prof. De Liu, MSBA 6330 Harvesting Big Data

Cost of Data Storage



Traditional vehicles for storing big data are prohibitively expensive

- We must retain and process large amount of data to extract its value
- Unfortunately, traditional vehicles for storing this data have become prohibitively expensive
    - In dollar costs, $ per GB
    - In computational costs
        - The need for ETL jobs to summarize the data for warehousing

The Traditional Workaround

How has industry typically dealt with these problem?

- Perform an ETL (Extract, Transform, Load) on the data to summarize and denormalize the data, before archiving the result in a data warehouse
- Discarding the details
- Run queries against the summary data

Unfortunately, this process results in loss of detail

- But there could be real nuggets in the lost detail
- Value lost with data summarization or deletion
- Think of the opportunities lost when we summarize 10000 rows of data into a single record
- We may not learn much from a single tweet, but we can learn a lot from 10000 tweets

More data == Deeper understanding

- "There's no data like more data" (Moore 2001)
- "It's not who has the best algorithms that wins. It's who has the most data"

We Need A System That Scales

We’re generating too much data to process with traditional tools

Two key problems to address

- How can we reliably store large amounts of data at a reasonable cost?
- How can we analyze all the data we have stored?

HADOOP OVERVIEW

What is Hadoop?

‘Apache **Hadoop**is an open-source software framework written in Java for distributed storage and distributed processing of very large data sets on computer clusters built from commodity hardware.’

-Google

‘A free, open source software framework that supports data-intensive distributed applications.’

-Cloudera

‘Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a distributed computing environment. It is part of the Apacheproject sponsored by the Apache Software Foundation.’

-TechTarget

What is Hadoop?

- Hadoop is a free software package that provides tools that enable data discovery, question answering, and rich analytics based on very large volumes of information.

- All this in fractions of the time required by traditional databases.

What is Apache Hadoop

A system for providing scalable, economical data storage and processing

- Distributed and fault tolerant
- Harnesses the power of industry standard hardware

'Core' Hadoop consists of two main components

- Storage: The Hadoop Distributed File System (HDFS)
  - A framework for distributing data across a cluster in ways that are scalable and fast.
- Processing: MapReduce
  - A framework for processing data in ways that are reliable and fast
- Plus the infrastructure needed to make them work, including
  - Filesystem and administration utilities
  - Job scheduling and monitoring

Scalability

Hadoop is a distributed system

- A collection of servers running Hadoop software is called a **cluster**

Individual servers within a cluster are called **nodes**

- Typically standard rack-mount servers running Linux
- Each node both stores and processes data

Add more nodes to the cluster to increase scalability

- A cluster may contain up to several thousand nodes
    - Facebook and Yahoo are each running clusters in excess of 4400 nodes
- Scalability is linear
    - And its accomplished on standard hardware, greatly reducing the storage costs

Reliability / Availability

Reliability and availability are traditionally expensive to provide

- Requiring expensive hardware, redundancy, auxiliary resources, and training

With Hadoop, providing reliability and availability is much less expensive

- If a node fails, just replace it

At Google, servers are not bolted into racks, they are attached to the racks using Velcro, in anticipation of the need to replace them quickly

- If a task fails, it is automatically run somewhere else
- No data is lost
- No human intervention is required

In other words, reliability / availability are provided by the framework itself

Fault Tolerance

Adding nodes increases chances that any one of them will fail

- Hadoop builds redundancy into the system and handle failures automatically

Files loaded into HDFS are replicated across nodes in the cluster

- If a node fails, its data is re-replicated using one of the other copies

Data processing jobs are broken into individual tasks

- Each task takes a small amount of data as input
- Thousands of tasks (or more) often run in parallel
- If a node fails during processing, its tasks are rescheduled elsewhere
- Routine failures are handled automatically without any loss of data

How are reliability, availability, and fault tolerance provided?

A file in Hadoop is broken down into **blocks**which are:

- typically 64MB each (a typical windows block size is 4KB)
- Distributed across the cluster
- Replicated across multiple nodes of the cluster (Default: 3)

Consequences of this architecture:

- If a machine fails (even an entire rack) no data is lost
- Tasks can run elsewhere, where the data resides
- If a task fails, it can be dispatched to one of the nodes containing redundant copies of the same data block
- When a failed node comes back online, it can automatically be reintegrated with the cluster

HDFS: Hadoop Distributed File System

Provides inexpensive and reliable storage for massive amounts of data

- Optimized for a relatively small number of large files
- Each file likely to exceed 100 MB, multi-gigabyte files are common
- Store file in hierarchical directory structure
  e.g. , /sales/reports/asia.txt
- Cannot modify files once written
- Need to make changes? remove and recreate
- Use Hadoop specific utilities to access HDFS

HDFS Architecture (1 Of 3)

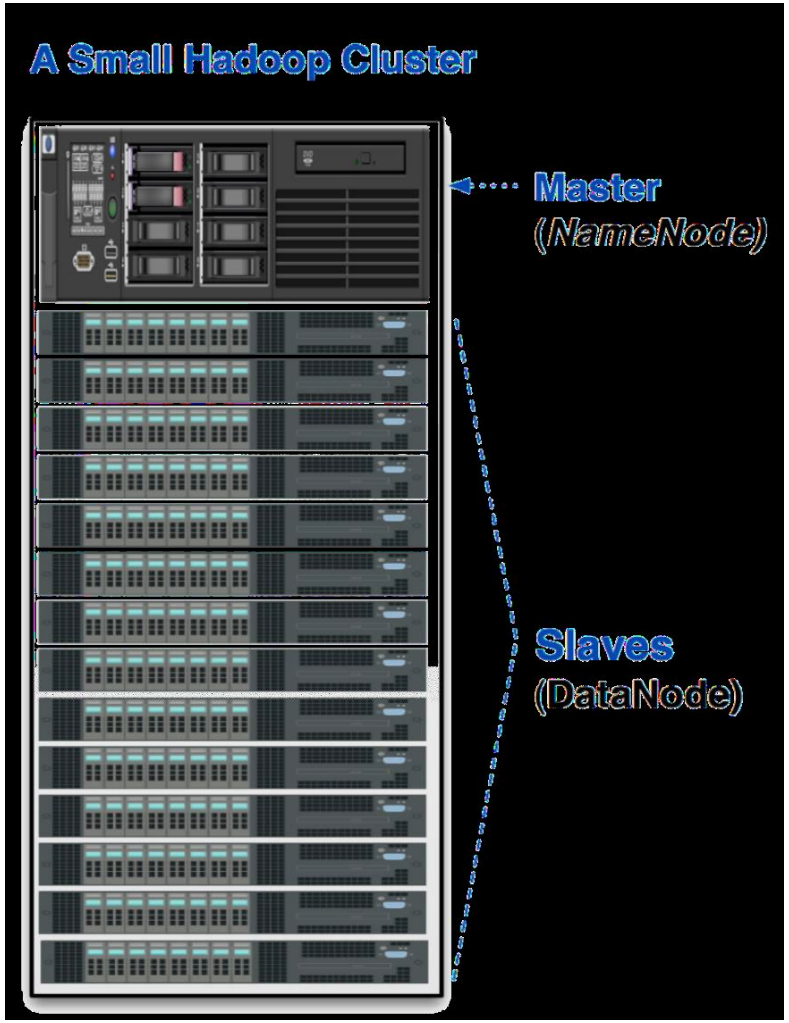
Hadoop has a master/slave architecture

HDFS master daemon: Name Node

- Manages namespace (file to block mappings) and metadata (block to machine mappings)
- Monitors slave nodes

HDFS slave daemon: Data Node

- Reads and writes the actual data



HDFS Architecture (2 Of 3)

Example:

The Name Node holds metadata for the two files

- Foo.txt (300MB) and Bar.txt (200MB)
- Assume HDFS is configured for 128MB blocks

The Data Nodes hold the actual blocks

- Each block is 128MB in size
- Each block is replicated three times on the cluster
- Block reports are periodically sent to the NameNode

21

HDFS Architecture (3 Of 3)

Optimal for handling millions of large files, rather than billions of small files, because:

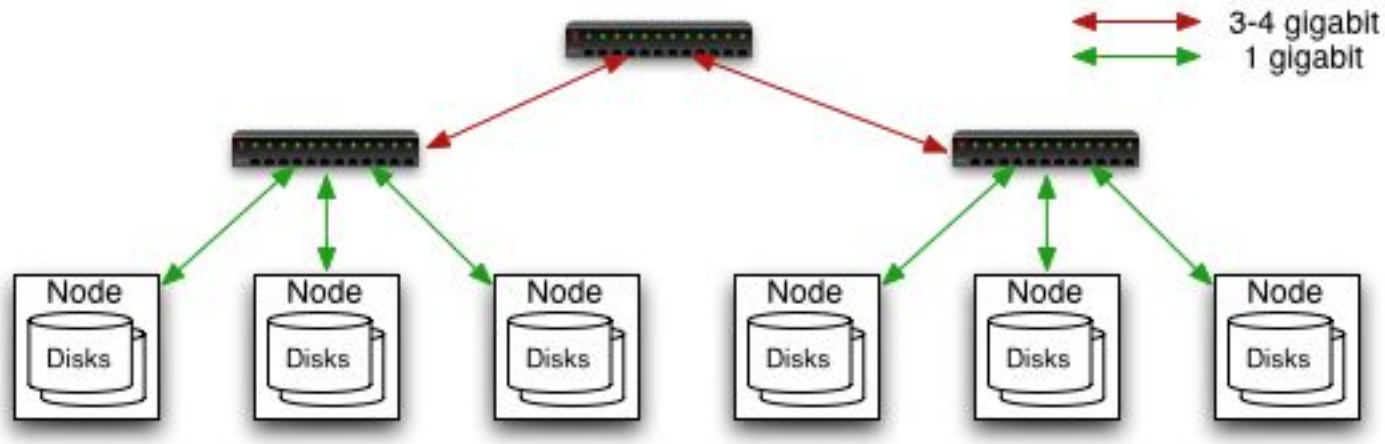
- In pursuit of responsiveness, the NameNodestores all of its file/block information
- Too many files will cause the NameNodeto run out of storage space
- Too many blocks (if the blocks are small) will also cause the NameNodeto run out of space
- Processing each block requires its own Java Virtual Machine (JVM) and (if you have too many blocks) you begin to see the limits of HDFS scalability

HDFS: Hadoop Distributed File System

Provides inexpensive and reliable storage for massive amounts of data

- Optimized for a relatively small number of large files
- Each file likely to exceed 100 MB, multi-gigabyte files are common
- Store file in hierarchical directory structure
  e.g. , /sales/reports/asia.txt
- Cannot modify files once written
- Need to make changes? remove and recreate
- Use Hadoop specific utilities to access HDFS

Commodity Hardware Cluster

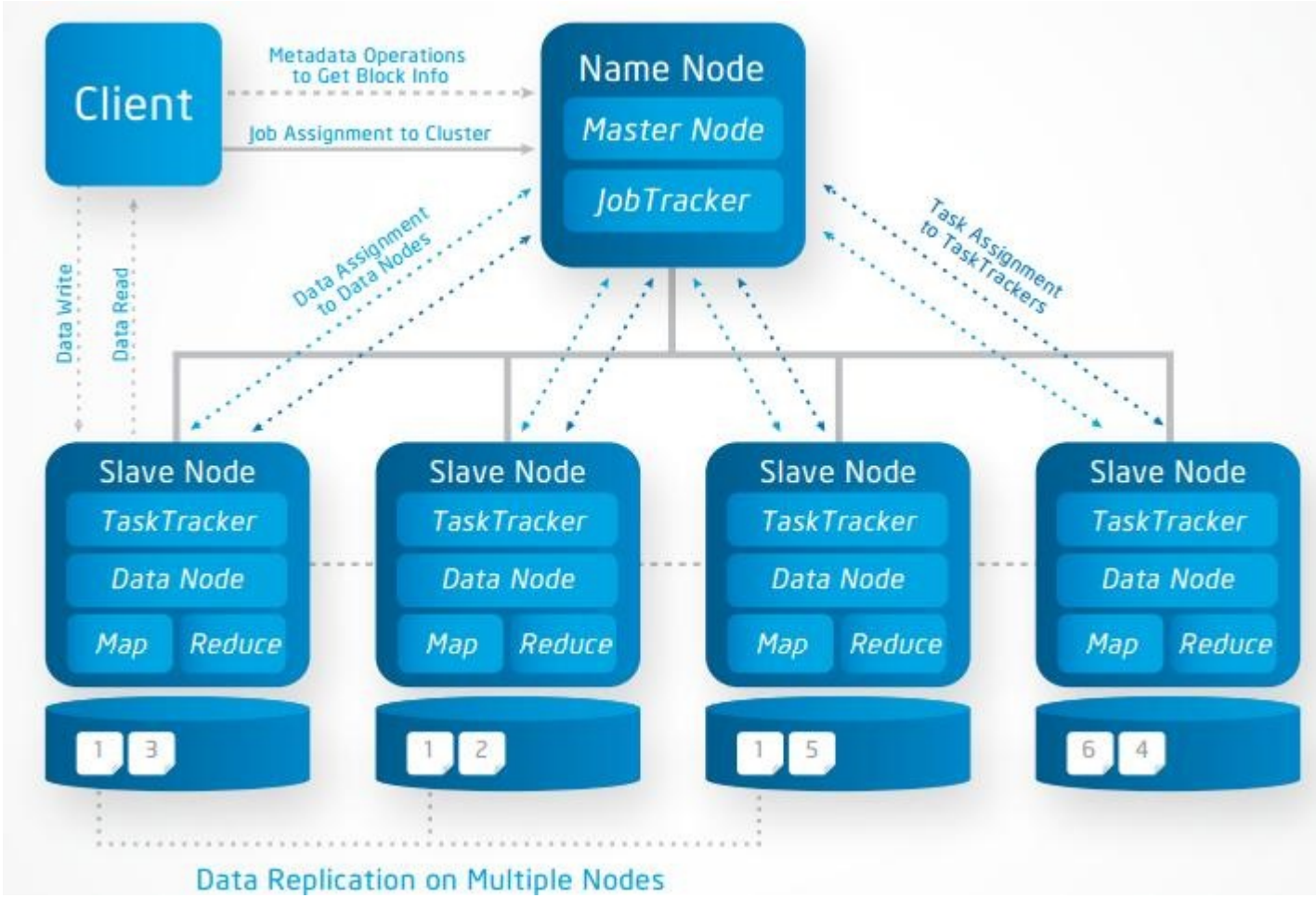


Typically in 2 level architecture

- Nodes are commodity PCs
- 40 nodes/rack
- Uplink from rack is 3-4 gigabit
- Rack-internal is 1 gigabit

HDFS Architecture - Read & Write Operations

- Name Node, Secondary Name Node and Job Tracker are known as Master daemons. Task Tracker and Data Nodes are known as slave daemons. Whenever a client submits a Job, the actual processing is going to take place in Task Tracker.
- Name Node consists of all Meta data information such as racks, blocks (r1, r2, b1, b2).
- NN sends its image file to secondary NN every 3sec (Heartbeat Mechanism). Name node is having special feature “Rack Awareness”, it will give information about nodes present in different racks of Hadoop cluster environment.
- Secondary Name Node (Snn) - If NN goes down, Snn will take the image instantaneously and acts like NN and performs the NN functionalities.
- Job Tracker (JT) - it is used for job scheduling and maintenance. It assigns different tasks to Task Tracker and monitors it.
- Task Tracker (TT) - Actual computational processing is going to happen here. It gets in contact with JT and has some heart beat mechanism between them.



Introducing Map Reduce

- Hadoop stores data, Map Reduce processes the data
- We typically process data in Hadoop using MapReduce
- MapReduce is not a language, it's a programming model
- MapReduce consists of two functions: map and reduce

27

Understanding Map and Reduce

The map function always runs first

- Typically used to “break down”
- Filter, transform, or parse data, e.g. Parse the stock symbol, price and time from a data feed
- The output from the map function (eventually) becomes the input to the reduce function

The reduce function

- Typically used to aggregate data from the map function

  e.g. Compute the average hourly price of the stock
- Not always needed and therefore optional
- You can run something called a “map-only” job

Shuffle and Sort

Between Map and Reduce tasks there is typically a hidden phase known as the "Shuffle and Sort"

- Which organizes map output for delivery to the reducer
- Each individual piece is simple, but collectively are quite powerful
- Analogous to a pipe / filter in Unix

Putting It All Together

Here's the data flow for the entire