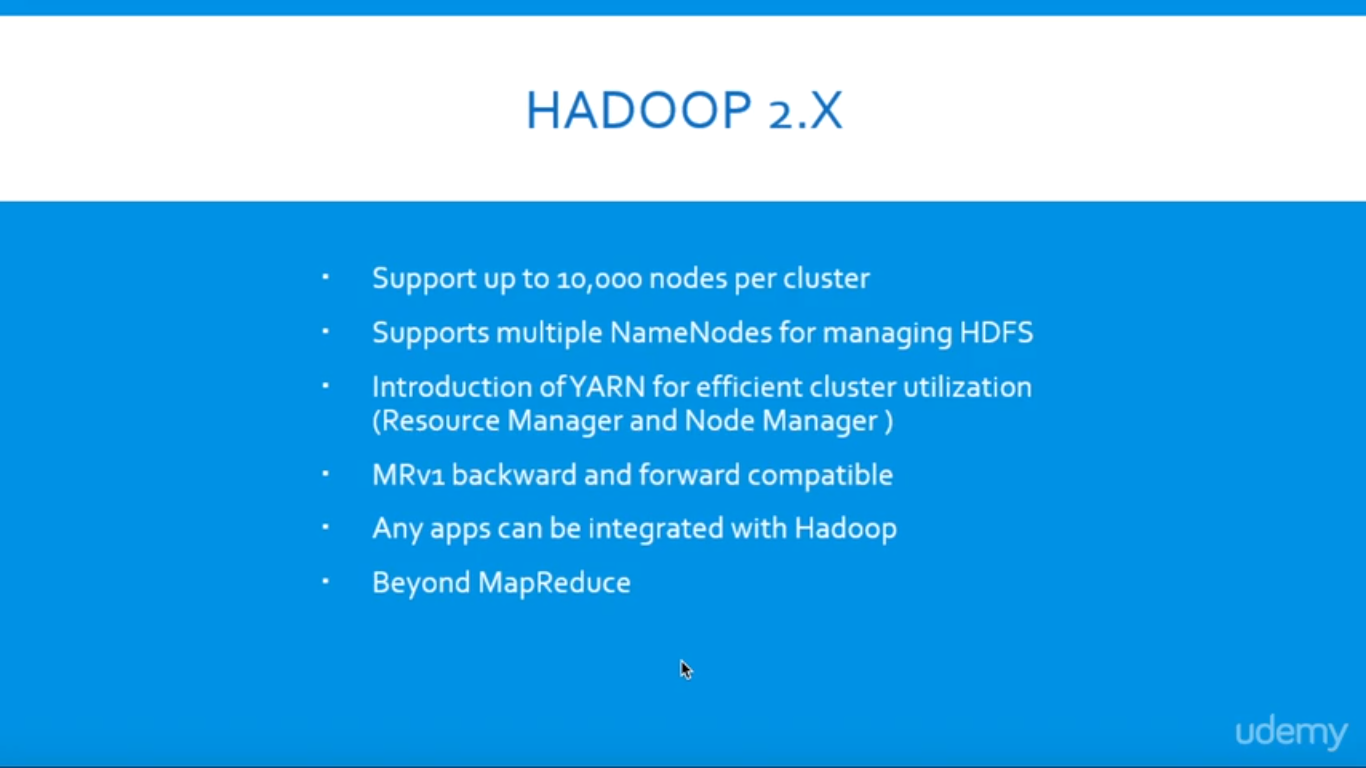
Hadoop 2.x

# Introduction

Apache Hadoop 2.0 represents a generational shift in the architecture of Apache Hadoop. With YARN, Apache Hadoop is recast as a significantly more powerful platform – one that takes Hadoop beyond merely batch applications to taking its position as a ‘data operating system’ where HDFS is the file system and YARN is the operating system.

YARN is a re-architecture of Hadoop that allows multiple applications to run on the same platform. With YARN, applications run “in” Hadoop, instead of “on” Hadoop:



The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker and TaskTracker into separate entities. In Hadoop 2.0, the JobTracker and TaskTracker no longer exist and have been replaced by three components:

* ResourceManager: a scheduler that allocates available resources in the cluster amongst the competing applications.
* NodeManager: runs on each node in the cluster and takes direction from the ResourceManager. It is responsible for managing resources available on a single node.
* ApplicationMaster: an instance of a framework-specific library, an ApplicationMaster runs a specific YARN job and is responsible for negotiating resources from the ResourceManager and also working with the NodeManager to execute and monitor Containers.

The actual data processing occurs within the Containers executed by the ApplicationMaster. A Container grants rights to an application to use a specific amount of resources (memory, cpu etc.) on a specific host.

YARN is not the only new major feature of Hadoop 2.0. HDFS has undergone a major transformation with a collection of new features that include:

* NameNode HA: automated failover with a hot standby and resiliency for the NameNode master service.
* Snapshots: point-in-time recovery for backup, disaster recovery and protection against use errors.
* Federation: a clear separation of namespace and storage by enabling generic block storage layer.

NameNode HA is achieved using existing components like ZooKeeper along with new components like a quorum of JournalNodes and the ZooKeeper Failover Controller (ZKFC) processes:

Federation enables support for multiple namespaces in the cluster to improve scalability and isolation. Federation also opens up the architecture, expanding the applicability of HDFS cluster to new implementations and use cases.

# HDFS File Read Workflow

Now let’s understand complete end to end HDFS data read operation. As shown in the above figure the data read operation in HDFS is distributed, the client reads the data parallelly from datanodes, the steps by step explanation of data read cycle is:

i) Client opens the file it wishes to read by calling open() on the*FileSystem* object, which for HDFS is an instance of *DistributedFileSystem*.

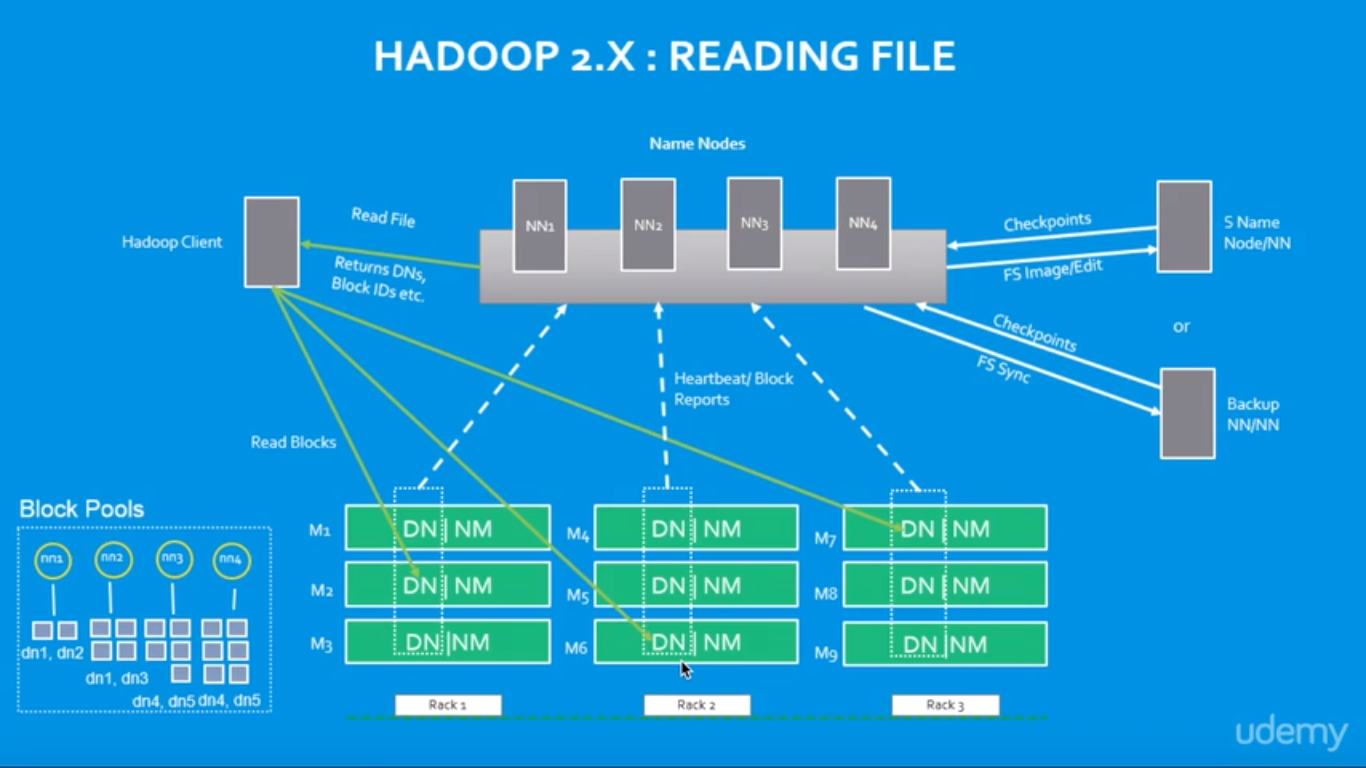
ii) *DistributedFileSystem* calls the namenode using RPC to determine the locations of the blocks for the first few blocks in the file. For each block, the namenode returns the addresses of the datanodes that have a copy of that block and datanode are sorted according to their proximity to the client.

iii)*DistributedFileSystem* returns a *FSDataInputStream* to the client for it to read data from. *FSDataInputStream*, thus, wraps the *DFSInputStream* which manages the datanode and namenode I/O. Client calls read() on the stream. DFSInputStream which has stored the datanode addresses then connects to the closest datanode for the first block in the file.

iv) Data is streamed from the datanode back to the client, as a result client can call read() repeatedly on the stream. When the block ends, DFSInputStream will close the connection to the datanode and then finds the best datanode for the next block.

v) If the *DFSInputStream* encounters an error while communicating with a datanode, it will try the next closest one for that block. It will also remember datanodes that have failed so that it doesn’t needlessly retry them for later blocks. The *DFSInputStream* also verifies checksums for the data transferred to it from the datanode. If it finds a corrupt block, it reports this to the namenode before the*DFSInputStream* attempts to read a replica of the block from another datanode.

vi) When the client has finished reading the data, it calls close() on the stream.



# Running a job on Hadoop

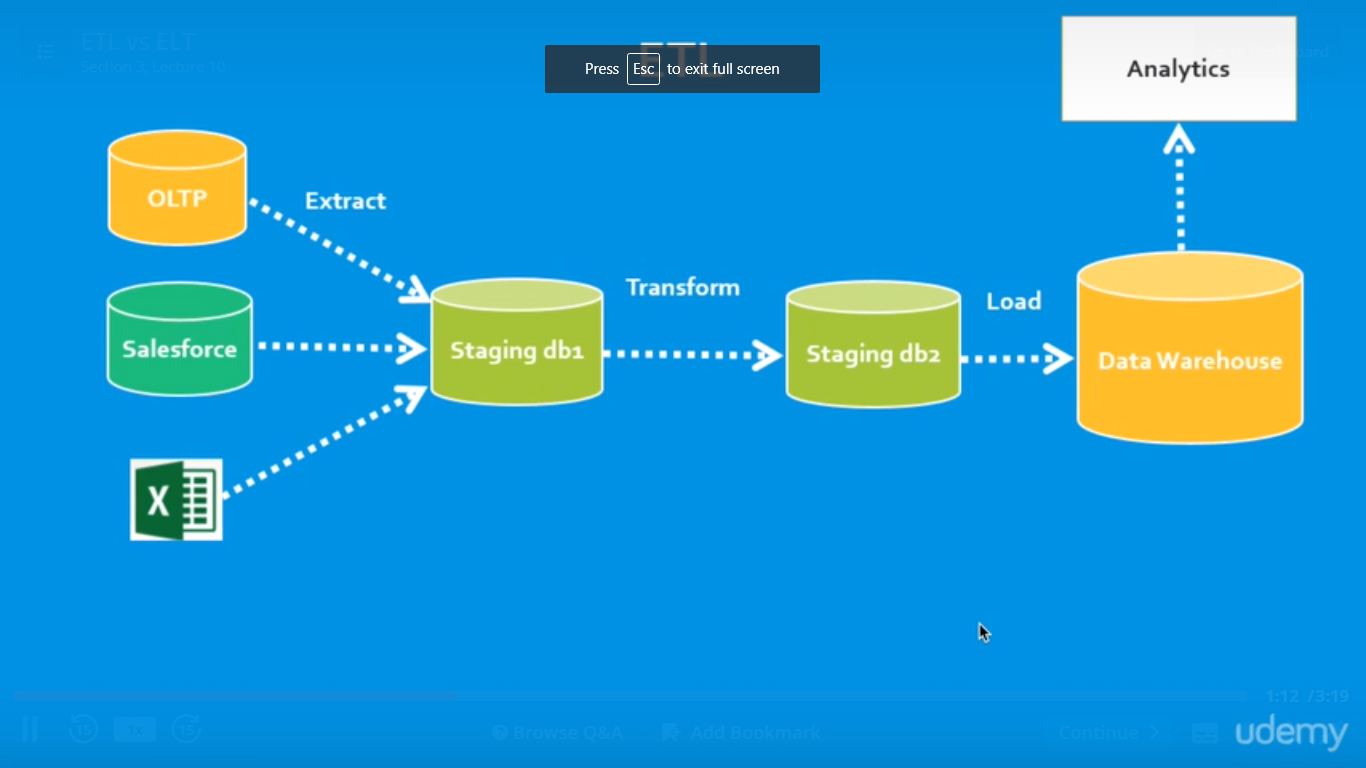
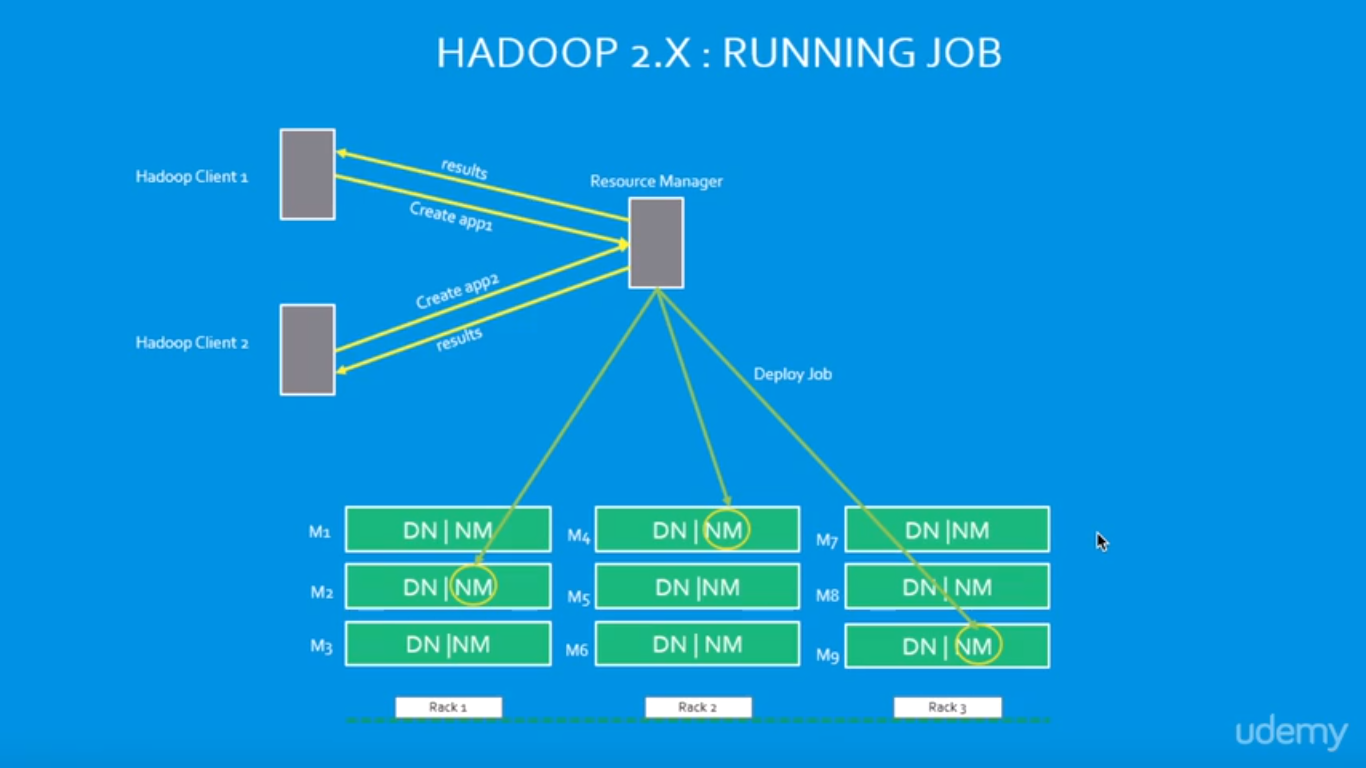
Before we jump into the programming our MapReduce, we may need to talk about the preparation steps that are commonly taken. Because MapReduce is usually operating on a huge data, we need to consider those steps before we actually do the MapReduce.

The underlying structure of the HDFS filesystem is very different from our normal file systems. The block sizes are quite a bit larger, and the actual block size for our clusters dependent on the cluster configuration as shown in the picture below: 64, 128, or 256 MB. So, we may need to have blocks with customized partitioned.

Another consideration is where we're going to retrieve our data from in order to perform the MapReduce operations or the parallel processing on it. Though we'll work with the core Hadoop filesystem, we may execute MapReduce algorithms against information stored on different locations such as native filesystem, cloud storage such as Amazon S3 buckets, or Windows Azure blobs.

Another considration is the output of the MapReduce job results are immutable. So, our output is a one-time output, and when a new output is generated, we have a new file name for it.

The last consideration in preparing for MapReduce is about the logic that we'll be writing, and it should fit our situation that we're trying to address. We'll be writing logic in some programming language, library, or tools to map our data to, and then reduce it, and then we have some output.

Note also that we'll be working with key-value pairs, so regardless of the format of the data coming in, we want to output key-value pairs. 

# ELT and ETL

These two definitions of ETL are what make ELT a bit confusing. ELT is a different way of looking at the tool approach to data movement. Instead of transforming the data before it’s written, ELT leverages the target system to do the transformation. The data is copied to the target and then transformed in place.

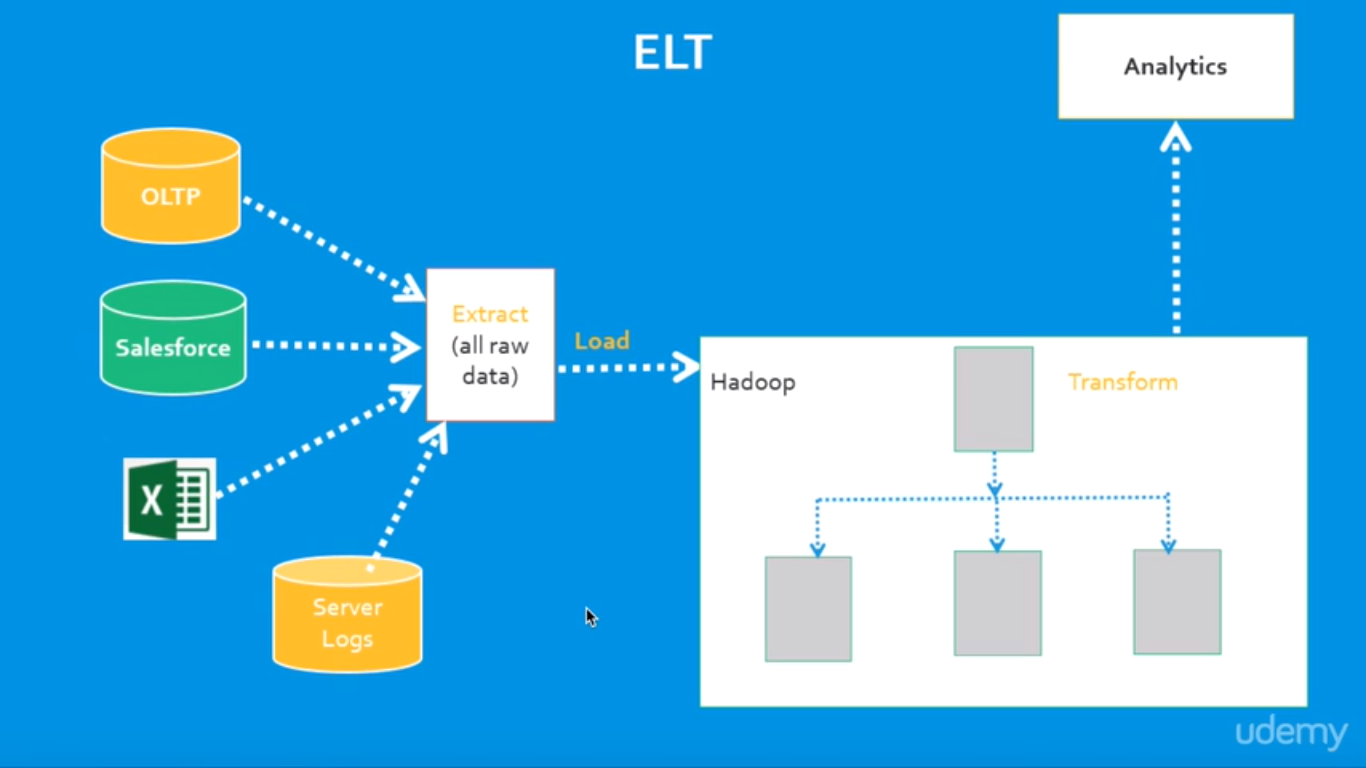
ELT makes sense when the target is a high-end data engine, such as a data appliance, Hadoop cluster, or cloud installation to name three examples.  If this power is there, why not use it?

ETL, on the other hand, is designed using a pipeline approach. While data is flowing from the source to the target, a transformation engine (something unique to the tool) takes care of any data changes.

Which is better depends on priorities. All things being equal, it’s better to have fewer moving parts. ELT has no transformation engine – the work is done by the target system, which is already there and probably being used for other development work. On the other hand, the ETL approach can provide drastically better performance in certain scenarios. The training and development costs of ETL need to be weighed against the need for better performance. (Additionally, if you don’t have a target system powerful enough for ELT, ETL may be more economical.

The specifics of ELT development will vary depending on the platform. For example, Hadoop clusters work by breaking a problem into smaller chunks, then distributing those chunks across a large number of machines for processing. The problem is solved faster because it’s being done in parallel. This requires careful design to make sure that the act of splitting the problem can be done without affecting the answer. Some problems can be easily split, others will be much harder.

In all cases, developers need to be aware of the nature of the system they’re using to perform transformations. Some systems (such as hardware appliances) have enough resources to handle nearly any transformation, but others require careful planning and design.

ELT is an excellent tactical tool for loading a data warehouse. It requires a powerful system in place as the target, but more and more warehouses are being built with such systems in mind to meet ever-growing analytic needs. As with any tool, knowing when to use it is at least as important as knowing how to use it. Ironsides can provide strategic direction and/or technical support in data integration and management. Contact us today to discuss which options fit your environment best. 

Different Vendors:

1.Amazon Elastic MapReduce (Amazon EMR)

Amazon Elastic MapReduce (EMR) is an Amazon Web Services ([AWS](http://whatis.techtarget.com/definition/Amazon-Web-Services-AWS)) tool for big data processing and analysis. Amazon EMR offers the expandable low-configuration service as an easier alternative to running in-house [cluster computing](http://searchdatacenter.techtarget.com/definition/cluster-computing).

Amazon EMR is based on Apache [Hadoop](http://searchcloudcomputing.techtarget.com/definition/Hadoop), a Java-based programming framework that supports the processing of large data sets in a [distributed computing](http://whatis.techtarget.com/definition/distributed-computing) environment. [MapReduce](http://searchcloudcomputing.techtarget.com/definition/MapReduce) is a software framework that allows developers to write programs that process massive amounts of unstructured data in parallel across a distributed cluster of [processors](http://searchcio-midmarket.techtarget.com/definition/processor) or stand-alone computers. It was developed at Google for indexing web pages and replaced their original indexing algorithms and [heuristics](http://whatis.techtarget.com/definition/heuristic) in 2004.

Amazon EMR processes big data across a [Hadoop cluster](http://searchbusinessanalytics.techtarget.com/definition/Hadoop-cluster) of virtual servers on Amazon Elastic Compute Cloud ([EC2](http://searchcloudcomputing.techtarget.com/definition/Amazon-Elastic-Compute-Cloud)) and Amazon Simple Storage Service (S3). The elastic in EMR's name refers to its dynamic resizing ability, which allows it to ramp up or reduce resource use depending on the demand at any given time.

2.Cloud Era

Cloudera Inc. is a [United States](https://en.wikipedia.org/wiki/United_States)-based software company that provides [Apache Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop)-based software, support and services, and training to business customers.

Cloudera's open-source Apache Hadoop distribution, CDH (Cloudera Distribution Including Apache Hadoop), targets enterprise-class deployments of that technology. Cloudera says that more than 50% of its engineering output is donated upstream to the various Apache-licensed open source projects (Apache Hive, Apache Avro, [Apache HBase](https://en.wikipedia.org/wiki/Apache_HBase), and so on) that combine to form the Hadoop platform.

Cloudera is also a sponsor of the [Apache Software Foundation](https://en.wikipedia.org/wiki/Apache_Software_Foundation).

3.IBM InfoSphere

InfoSphere DataStage is a powerful data integration tool.

It was acquired by IBM in 2005 and has become a part of IBM Information Server Platform. It uses a client/server design where jobs are created and administered via a Windows client against central repository on a server.

The IBM InfoSphere DataStage is capable of integrating data on demand across multiple and high volumes of data sources and target applications using a high performance parallel framework.

InfoSphere DataStage also facilitates extended metadata management and enterprise connectivity

It has three levels of Parallelism which are:

Pipeline Parallelism, Data Parallelism, Component Parallelism

Teradata Logo.png4.Teradata

Teradata Corporation is a provider of [database](https://en.wikipedia.org/wiki/Database)-related products and services. The company was formed in 1979 in [Brentwood, California](https://en.wikipedia.org/wiki/Brentwood,_California), as a collaboration between researchers at [Caltech](https://en.wikipedia.org/wiki/Caltech) and [Citibank](https://en.wikipedia.org/wiki/Citibank)'s advanced technology group.[[2]](https://en.wikipedia.org/wiki/Teradata#cite_note-2) The company was acquired by [NCR Corporation](https://en.wikipedia.org/wiki/NCR_Corporation) in 1991, and subsequently spun-off again as an independent public company on October 1, 2007.

The company produces a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system) of the same name, which it markets as a [data warehouse](https://en.wikipedia.org/wiki/Data_warehouse)

Teradata offers three main services to its customers: cloud and hardware-based data warehousing, business analytics, and consulting services.

the company launched Teradata Everywhere, which allows users to submit queries against public and private databases.

The service uses [massively parallel](https://en.wikipedia.org/wiki/Massively_parallel) processing across both its physical data warehouse and cloud storage, including managed environments such as [Amazon Web Services](https://en.wikipedia.org/wiki/Amazon_Web_Services), [Microsoft Azure](https://en.wikipedia.org/wiki/Microsoft_Azure), [VMware](https://en.wikipedia.org/wiki/VMware), and Teradata's Managed Cloud and IntelliFlex

 Teradata offers customers both [hybrid cloud](https://en.wikipedia.org/wiki/Hybrid_cloud) and [multi-cloud](https://en.wikipedia.org/wiki/Multicloud) storage.