Platforms to handle Big data

Dr. Jigna Ashish Patel
Assistant Professor, CSE Dept,
Institute of Technology,
Nirma University

Objective of the lecture

- Right platform
- Need of the application/algorithm
- Right decision

- How quickly do we need to get the results?
- How big is the data to be processed?
- Does the model building require several iterations or single iteration?

System/platform level requirements

- Will there be a need for more data processing capability in the future?
- Is the rate of data transfer critical for this application?
- Is there a need for handling hardware failures within the application?

Horizontal Scaling

- It involves distributing the workload across many servers which may be even commodity machines.
- It is also known as “scale out”, where multiple independent machines are added together in order to improve the processing capability.
- Typically, multiple instances of the operating system are running on separate machines.

Vertical Scaling

- Vertical Scaling involves installing more processors, more memory and faster hardware, typically, within a single server.
- It is also known as “scale up” and it usually involves a single instance of an operating system.

Table 1 A comparison of advantages and drawbacks of horizontal and vertical scaling

Scaling	Advantages	Drawbacks
Horizontal scaling	➔ Increases performance in small steps as needed	➔ Software has to handle all the data distribution and parallel processing complexities
	➔ Financial investment to upgrade is relatively less	➔ Limited number of software are available that can take advantage of horizontal scaling
	➔ Can scale out the system as much as needed	
Vertical scaling	➔ Most of the software can easily take advantage of vertical scaling	➔ Requires substantial financial investment
	➔ Easy to manage and install hardware within a single machine	➔ System has to be more powerful to handle future workloads and initially the additional performance in not fully utilized
		➔ It is not possible to scale up vertically after a certain limit

Horizontal Scaling Platforms

- Peer-to-Peer Network
- Apache Hadoop
- Apache Spark

Vertical Scaling Platforms

- High performance computing clusters
- Multicore CPU
- Graphics Processing Unit(GPU)
- Field Programmable gate arrays(FPGA)

Peer-to-Peer networks

- involve millions of machines connected in a network
- decentralized and distributed network architecture where the nodes in the networks (known as peers) serve as well as consume resources.
- oldest distributed computing platforms
- Message Passing Interface (MPI) for communication scheme used in such a setup to communicate and exchange the data between peers.
- Each node can store the data instances and the scale out is practically unlimited (can be millions of nodes).

Apache Hadoop

- open source framework for storing and processing large datasets using clusters of commodity hardware.
- Hadoop is designed to scale up to hundreds
- highly fault tolerant
- The Hadoop platform contains the following two important components: (1) HDFS (2) YARN

Apache Spark

- developed by researchers at the University of California at Berkeley. designed to overcome the disk I/O limitations
- ability to perform in-memory computations.
- allows the data to be cached in memory, thus eliminating the
- Hadoop’s disk overhead limitation for iterative tasks.
- supports Java, Scala and Python and for certain tasks
- it is tested to be up to 100× faster than Hadoop MapReduce

HPC clusters

- Known as blades or supercomputers, are machines with thousands of cores.
- They can have a different variety of disk organization, cache, communication mechanism etc.
- powerful hardware which is optimized for speed and throughput.
- They are not as scalable as Hadoop or Spark clusters but they are still capable of processing terabytes of data.

Multicore CPU

- Multicore refers to one machine having dozens of processing cores They usually have shared memory but only one disk.
- the number of cores per chip and the number of
- operations that a core can perform has increased significantly. Newer breeds of motherboards allow multiple CPUs within a single machine thereby increasing the parallelism.
- Until the last few years, CPUs were mainly responsible for accelerating the algorithms for big data analytics.

GPU

- It is designed to accelerate the creation of images in a frame buffer intended for display output
- GPUs were primarily used for graphical operations such as video and image editing, accelerating graphics-related processing etc. due to their massively parallel architecture, recent developments in GPU hardware and related programming frameworks have given rise to GPGPU
- In addition to the processing cores, GPU has its own high throughput DDR5 memory which is many times faster than a typical DDR3 memory.

FPGA

- highly specialized hardware units for specific applications
- FPGAs can be highly optimized for speed and can be orders of magnitude faster compared to other platforms for certain applications.
- Due to customized hardware, the development cost is typically much higher compared to other platforms.
- On the software side, coding has to be done in HDL with a low-level knowledge of the hardware which increases the algorithm development cost.

Comparison of platforms

System/Platform Level characteristics

- Scalability Horizontal scaling application boosts
- Data I/O performance Vertical scaling boosts
- Fault Tolerance Only peer to peer is worst otherwise all are good

Scalability

Platform	Scalability
Peer-to-Peer	* * * * *
Virtual Clusters(MapReduce/MPI)	* * * * *
Virtual Clusters(Spark)	* * * * *
HPC clusters (MPI/MapReduce)	* * *
Multicore(Multithreading)	* *
GPU(CUDA)	* *
FPGA(HDL)	*

Data I/O Performance

Platform	Data I/O performance
Peer-to-Peer	*
Virtual Clusters(MapReduce/MPI)	* *
Virtual Clusters(Spark)	* * *
HPC clusters (MPI/MapReduce)	* * * *
Multicore(Multithreading)	* * * *
GPU(CUDA)	* * * * *
FPGA(HDL)	* * * * *

Fault Tolerance

Platform	Fault Tolerance
Peer-to-Peer	*
Virtual Clusters(MapReduce/MPI)	* * * * *
Virtual Clusters(Spark)	* * * * *
HPC clusters (MPI/MapReduce)	* * * *
Multicore(Multithreading)	* * * *
GPU(CUDA)	* * * *
FPGA(HDL)	* * * *

Comparison of platforms

Application/Algorithm Level characteristics

- Real time processing Vertical is good and specifically GPU and FPGA
- Data size supported Horizontal is good and peer to peer best as it contains billion of nodes resulting in an infinite amount of storage
- Iterative task support Vertical is good

Real Time Processing

Platform	Real Time Processing
Peer-to-Peer	*
Virtual Clusters(MapReduce/MPI)	* *
Virtual Clusters(Spark)	* *
HPC clusters (MPI/MapReduce)	* * *
Multicore(Multithreading)	* * *
GPU(CUDA)	* * * * *
FPGA(HDL)	* * * * *

Data Size supported

Platform	Data Size supported
Peer-to-Peer	* * * * *
Virtual Clusters(MapReduce/MPI)	* * * *
Virtual Clusters(Spark)	* * * *
HPC clusters (MPI/MapReduce)	* * * *
Multicore(Multithreading)	* *
GPU(CUDA)	* *
FPGA(HDL)	* *

Iterative Task Support

Platform	Iterative task support
Peer-to-Peer	* *
Virtual Clusters(MapReduce/MPI)	* *
Virtual Clusters(Spark)	* * *
HPC clusters (MPI/MapReduce)	* * * *
Multicore(Multithreading)	* * * *
GPU(CUDA)	* * * *
FPGA(HDL)	* * * *

Peer-to-Peer

ADVANTAGES

- Handle huge data.

DISADVANTAGES

- Communication is Slower.
- Not suitable for iterative algorithms.
- **Fault-intolerant.**

Virtual Clusters using Map/Reduce:

ADVANTAGES

- Fault-Tolerant.
- **Scalable.**

DISADVANTAGES

- Takes time to handle large amount of data.
- **Not suitable for iterative algorithms.**

Virtual Clusters using Spark:

ADVANTAGES

- In-memory computations.
- **Data is cached in the memory.**

DISADVANTAGES

- **Not optimal for real time processing tasks.**

High Performance Computing Cluster

ADVANTAGES

- **Fault tolerance.**

DISADVANTAGES

- **Expensive.**

Multicore CPU's

ADVANTAGES

- **Parallel operations.**

DISADVANTAGES

- DDR3 - slower.
- **Limited number of cores in CPU.**

Graphics Processing Unit's

ADVANTAGES

- High throughput memory.
- DDR5 memory – faster.
- **Well suited for real-time applications.**

DISADVANTAGES

- Limited software that supports GPU.
- Limited algorithms that are portable to GPU's.
- **Memory Constraints, less likely to be scalable.**

Field Programmable Gate Arrays(FPGA)

ADVANTAGES

- **Highly optimized speed, faster.**

DISADVANTAGES

- Limited number of applications.
- Limited amount of memory.
- **Expensive and Complex.**

How will you choose one of platform for a particular criteria ?

Amount of Time

Criterion : 'Amount of time'

Choice Made: GPU

Reasons:

1) Optimized for Speed
2) Thousands of processing cores
3) High memory bandwidth, uses DDR5
4) Widely used in many machine learning Algorithms

Number of Iterations

Choice Made: GPU

Reasons:

1) Elimination of Data to be read/write from disc.

2) Data is processed in threads, multi core processing.

Practically proved with k-means algorithm.

Fault Tolerance

Definition: Fault tolerance is the ability of the system to perform properly in case of any of its components failure.

- Virtual Clusters (Map Reduce & Spark) : Built-in fault tolerance mechanism.
- Multicore CPU, GPU, FPGA : Very rarely prone to hardware failures.
- Peer-to-Peer: No built-in fault tolerance mechanism, Uses commodity machines which are highly probable for hardware failures.

Scalability

Definition: Scalability is defined as the ability of the system to cope up with the increased demands in data processing.

- Virtual clusters, Peer-to-Peer networks: Easy to scale out.
- High performance Computing Cluster, FPGA: Once deployed, expensive to scale up.
- GPU: Limitations on # of GPU's a machine can have makes it not effectively scalable.

Choice of platform

- Data size
- Speed/Throughput
- Training /Applying a model

K means clustering

The k-means Clustering Algorithm
Input : Data points D, Number of clusters k
Step 1: Initialize k centroids randomly
Step 2: Associate each data point in D with the nearest centroid. This will divide the data points into k clusters.
Step 3: Recalculate the position of centroids.
Repeat steps 2 and 3 until there are no more changes in the membership of the data points
Output : Data points with cluster memberships

K-means on MapReduce

k-means::Map
Input: Data points D, number of clusters k and centroids
1: for each data point d Є D do
2: Assign d to the closest centroid
Output: centroids with associated data points

k-means::Reduce
Input: Centroids with associated data points
1: Compute the new centroids by calculating the average of data points in cluster
2: Write the global centroids to the disk
Output: New centroids

K-means on MPI

k-means::MPI
Input: Data points D, number of clusters k
1: Slaves read their part of data
2: do until global centroids converge
3: Master broadcasts the centroids to the slaves
4: Slaves assign data instances to the closest centroids
5: Slaves compute the new local centroids and local cluster sizes
6: Slaves send local centroids and cluster sizes to the master
7: Master aggregates local centroids weighted by local cluster sizes into global centroids.
Output: Data points with cluster memberships

K-means on GPU

***k*-means::GPU**
Input: Data points D, number of clusters k
1: do until global centroids converge
2: Upload data points to each multiprocessor and centroids to the shared memory
3: Multiprocessor works with one data vector at a time and associate it with the closest centroid
4: Centroid recalculation is done on CPU