SAKARYA ÜNİVERSİTESİ

BIG DATA

TOO BIG TO IGNORE

SÜMEYYE KAYNAK

OUTLINE

SAKARYA ÜNİVERSİTESİ

Scaling platform

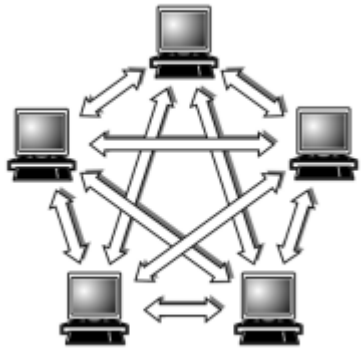
- Horizontal scaling platform
- Vertical scaling platform

SCALABILITY

Scalability	Pros	Cons
Horizontal scaling	• Increasing performance in small steps as needed • The financial investment required to improve performance is comparatively less. • The system can be scaled as needed.	• The software must handle all data distribution and parallel processing complexities. • There is a limited number of software available that can take advantage of horizontal scaling.
Vertical scaling	• Most software can easily take advantage of vertical scaling. • It's easy to manage and install hardware on a single machine.	• It requires significant financial investment. • The system must be more powerful to handle future workloads, and additional performance is wasted initially. • It is not possible to scale vertically after a certain limit.

HORIZONTAL SCALING PLATFORMS

Peer to peer network



Apache Hadoop

Apache Spark

VERTICAL SCALING PLATFORMS

- High performance computing (hpc) cluster
- Multicore Processors
- Graphics processing unit (GPU)
- Field Programmable Gate Arrays (FPGA)

PEER TO PEER NETWORK

- Typically, there are billions of networked machines.
- It has a decentralized and distributed network architecture.
- MPI (message pass-through interface) is used for communication.
- Each node has the ability to store and process data.
- Scaling is almost unlimited.

PEER TO PEER NETWORK

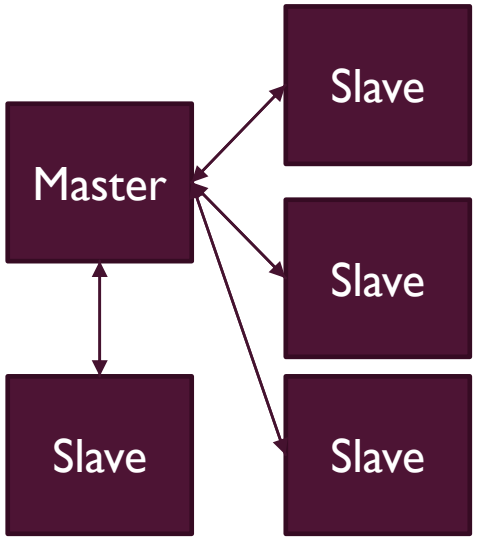
- Bottleneck may occur
- Broadcasting message are cheaper but aggregating data/result is costly.

APACHE HADOOP SOFTWARE ARCHITECTURE

- Hadoop is an open-source platform that enables the processing and storage of very large datasets.
- Hadoop is written in the Java programming language.
- The project is still maintained by Apache.

APACHE HADOOP SOFTWARE ARCHITECTURE

- Files that send divides blocks and form a cluster.



NAMENODE (MASTER NODE)

- The NameNode is the node that maintains and manages the blocks in the DataNode.
- Data exists only in DataNodes. In HDFS Architecture, user data never resides on the NameNode.

NAMENODE

NameNode functions:

- It is the node that maintains and manages DataNodes.
- Saves the metadata of all files stored in the cluster. For example; location of stored blocks, size of files, permissions etc.
- It regularly receives a Heartbeat and a block report from all DataNodes in the cluster to check that the DataNodes are alive.
- It keeps track of all blocks in HDFS.

DATANODE

- DataNodes are slave nodes in HDFS.

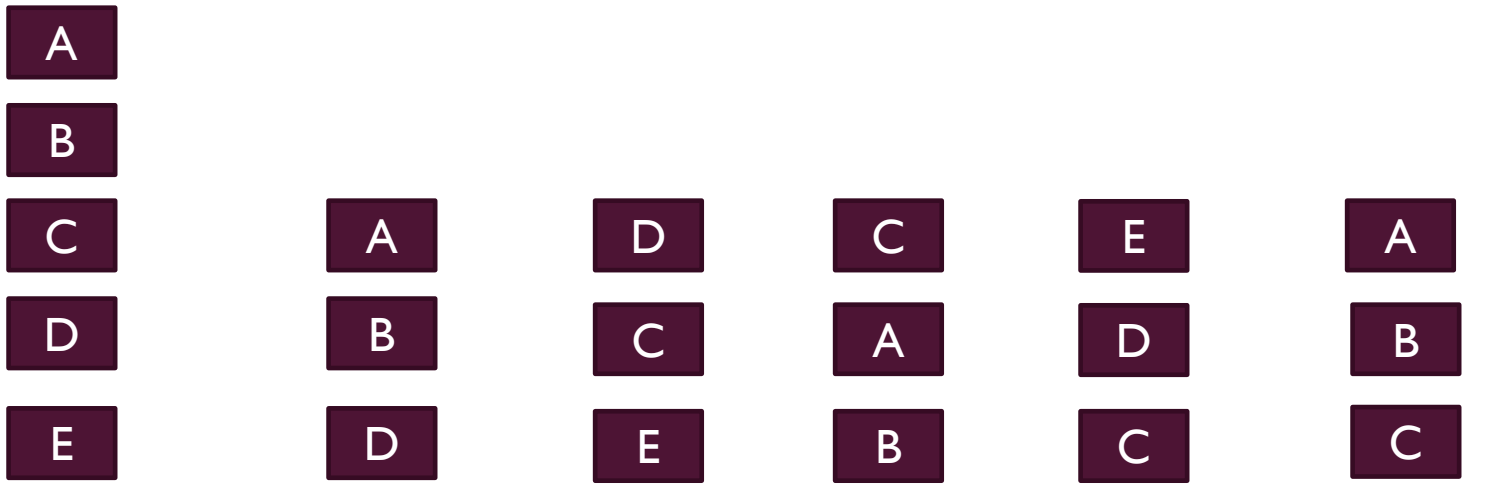
NameNode functions:

- Actual data is stored in DataNode.
- DataNodes handle read and write requests from file system clients.
- They periodically send a heartbeat to the NameNode to let them know it's alive.

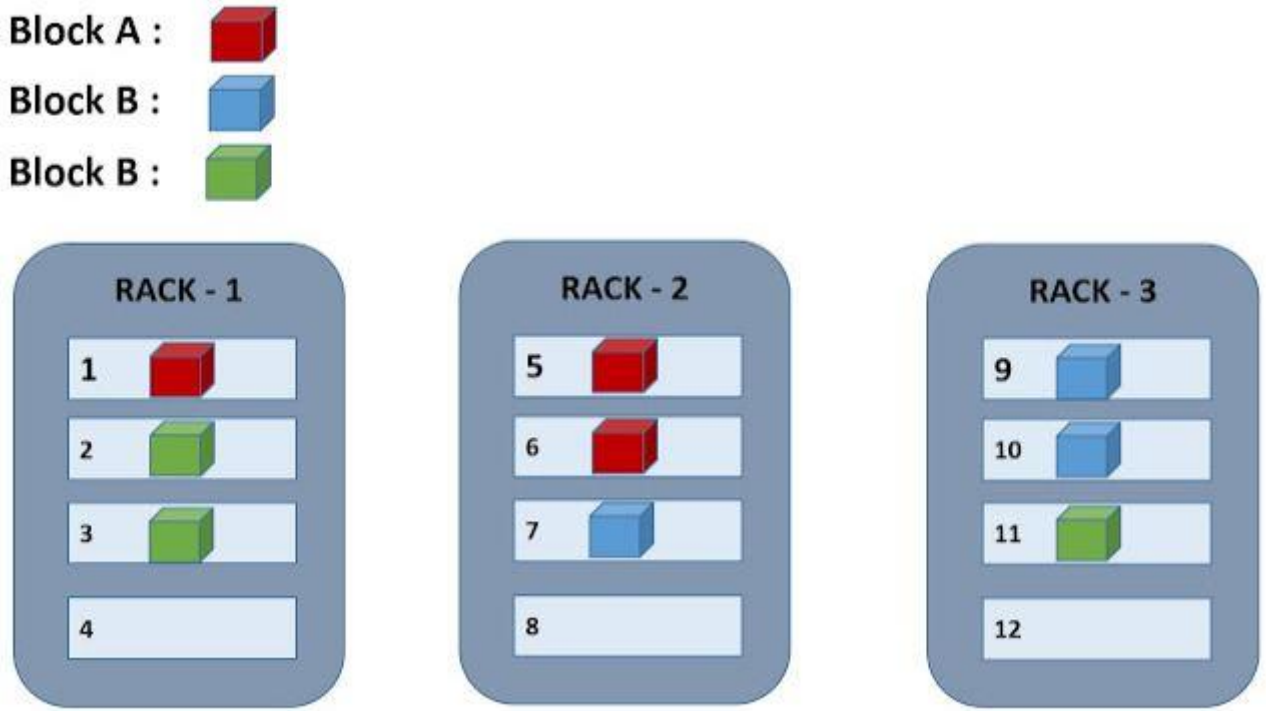
(By default, this frequency value is 3 seconds.)

APACHE HADOOP SOFTWARE ARCHITECTURE

- Data is stored in a cluster by multiplexing. **(replication factor)**



RACK AWARENESS



APACHE HADOOP SOFTWARE ARCHITECTURE

- The Hadoop project has 4 key components:
  - Hadoop Common
  - HDFS
  - HADOOP Yarn
  - Map-Reduce

APACHE HADOOP SOFTWARE ARCHITECTURE

- The Hadoop project has 4 key components:
  - **Hadoop Common**
  - HDFS
  - HADOOP Yarn
  - Map-Reduce

APACHE HADOOP SOFTWARE ARCHITECTURE

- The Hadoop project has 4 key components:
  - Hadoop Common
  - **HDFS**
  - HADOOP Yarn
  - Map-Reduce

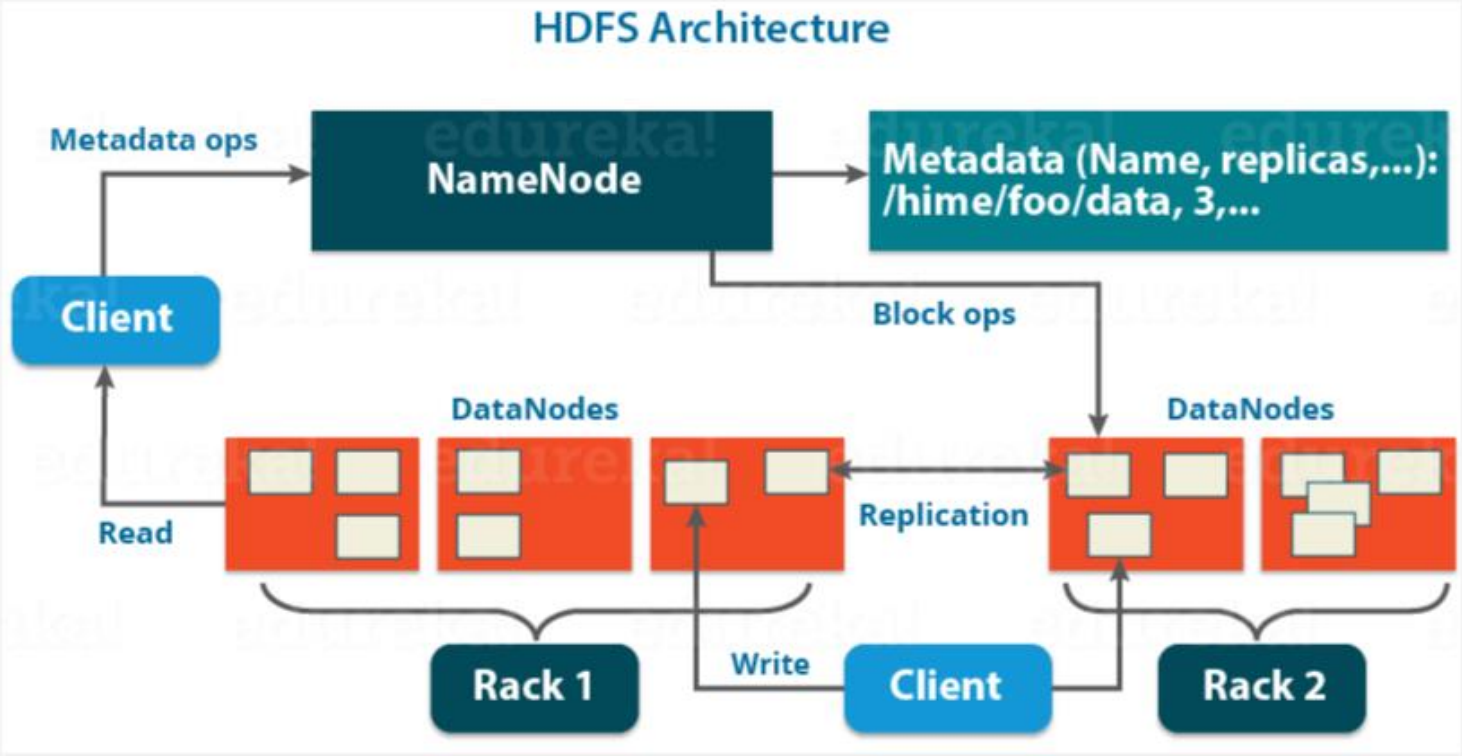
APACHE HADOOP SOFTWARE ARCHITECTURE

- The Hadoop project has 4 key components:
  - Hadoop Common
  - HDFS
  - **HADOOP Yarn**
  - Map-Reduce

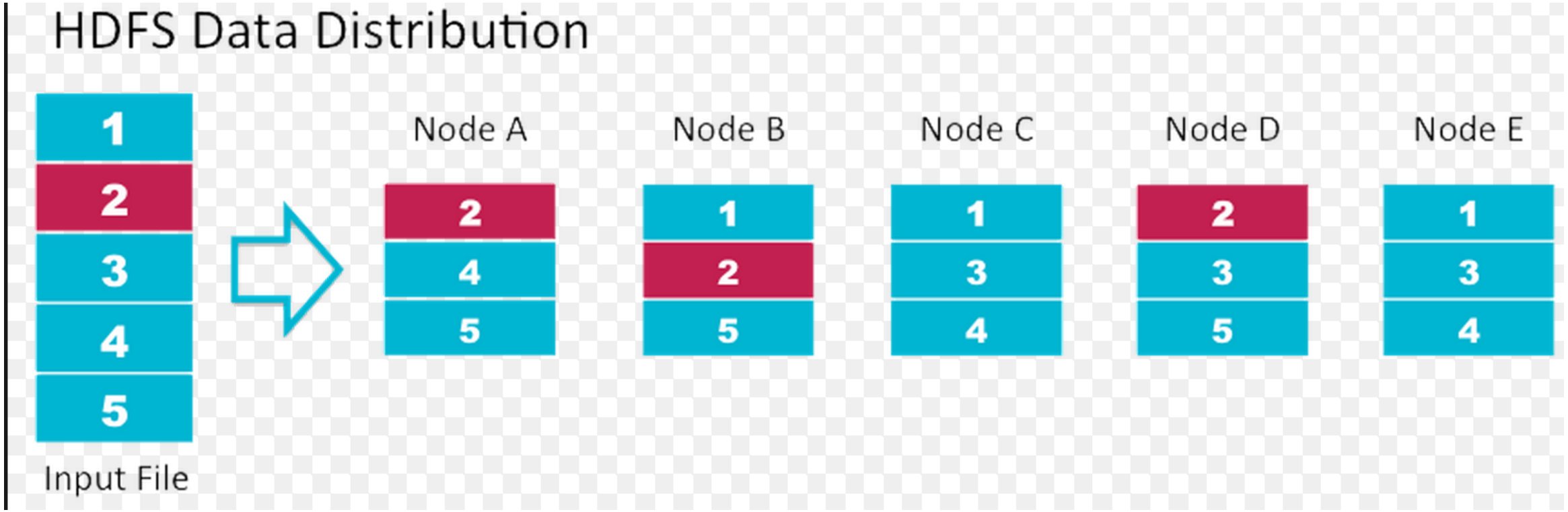
APACHE HADOOP SOFTWARE ARCHITECTURE

- The Hadoop project has 4 key components:
  - Hadoop Common
  - HDFS
  - HADOOP Yarn
  - **Map-Reduce**

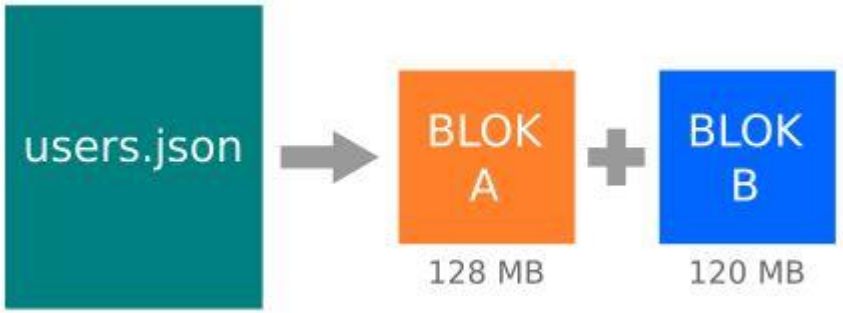
APACHE HADOOP SOFTWARE ARCHITECTURE



APACHE HADOOP SOFTWARE ARCHITECTURE



HDFS READ/WRITE



When data is wanted to be written to HDFS;

- First, the HDFS client makes a request to the NameNode to write the two blocks.
- The NameNode gives the client write permission and provides the IP addresses of the DataNodes to which the blocks will be copied.
- The selection of DataNodes was randomized based on availability, replication factor, and rack awareness.
- If the replication factor is 3, the NameNode provides 3 DataNode IPs for each block.

Blok A için = { DataNode 1, DataNode 4, DataNode 6 }

Blok B için = { DataNode 3, DataNode 7, DataNode 9 }

- Each block is copied in 3 different DataNodes.

MAP-REDUCE

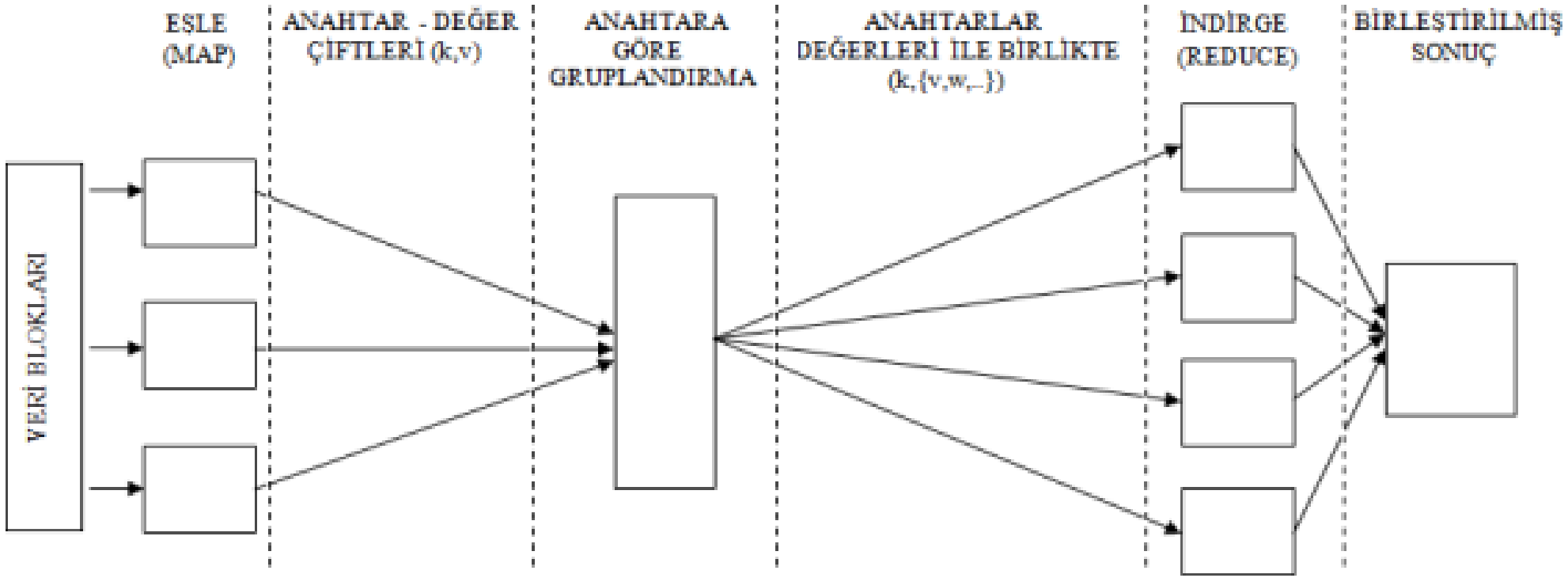
- Map-reduce is programming framework based on big data problems are first fragmented and then processed in parallel on many server.
- Is developed by Google.
- Is based on divide and conquer method.

MAP-REDUCE

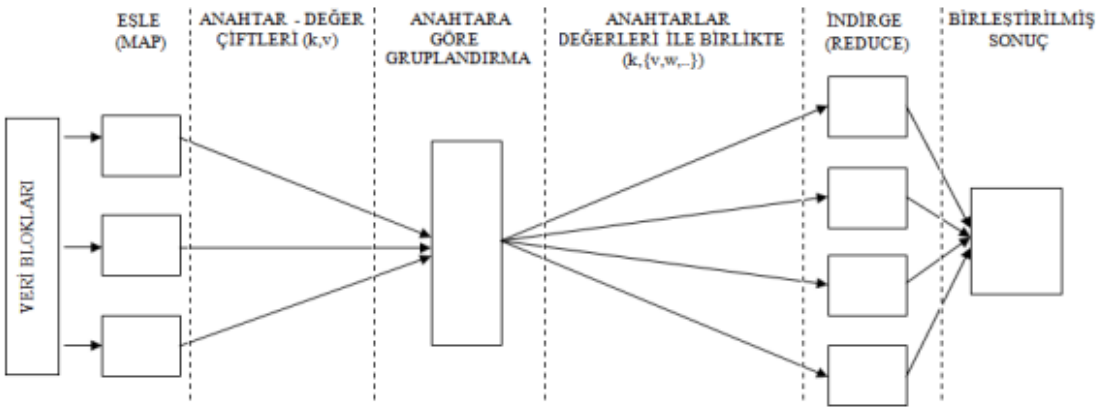
- In map phase, main node divides big problems to small and manageable sub-problem then deploys to worker nodes.
- Jobs in map phase is independent, so the jobs can run in parallel.

- In reduce phase, the completed jobs are combined according to the business logic and the result is obtained.

MAP-REDUCE

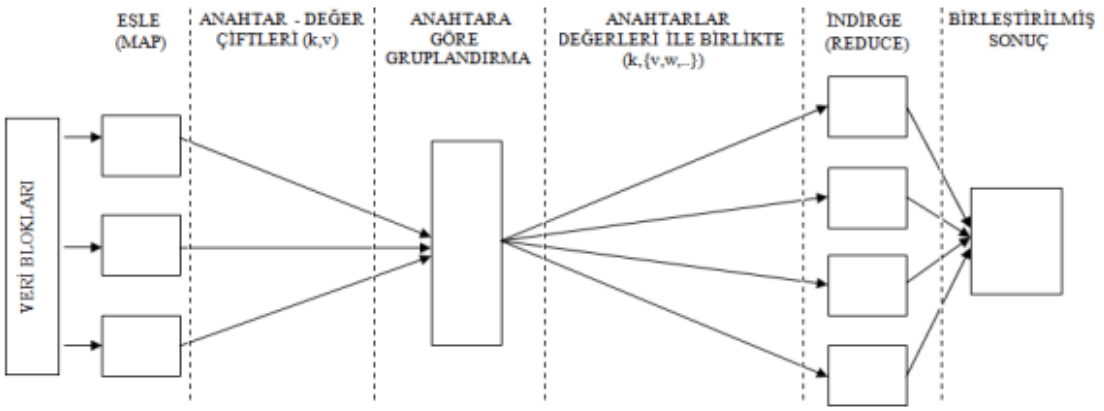

EŞLE
(MAP)
ANAHTAR - DEĞER
ÇİFTLERİ (k,v)
ANAHTARA
GÖRE
GRUPLANDIRMA
ANAHTARLAR
DEĞERLERİ İLE BİRLİKTE
(k,{v,w,..})
İNDİRGE
(REDUCE)
BİRLEŞTİRİLMİŞ
SONUÇ
VERİ BLOKLARI

MAP-REDUCE



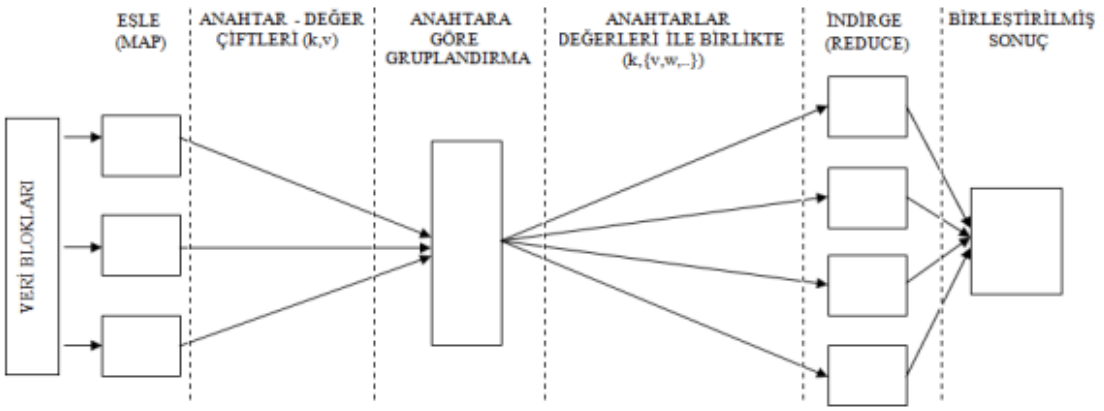
- Master node gets data inputs, divides to small, manageable sub-jobs and deploys to worker-node.
- The worker-node performs sub-jobs assigned to them, under job-follower control.

MAP-REDUCE



- The worker-node classifies the data in key-value format.
- Classified results is in local file system that can reach in reduce phase.

MAP-REDUCE



- In the reduce phases, master node get results came from the worker-node and reduce ‘value’ data according to ‘key’ prop.
- Are collected results coming from each node.

WORD-COUNT WITH MAP-REDUCE

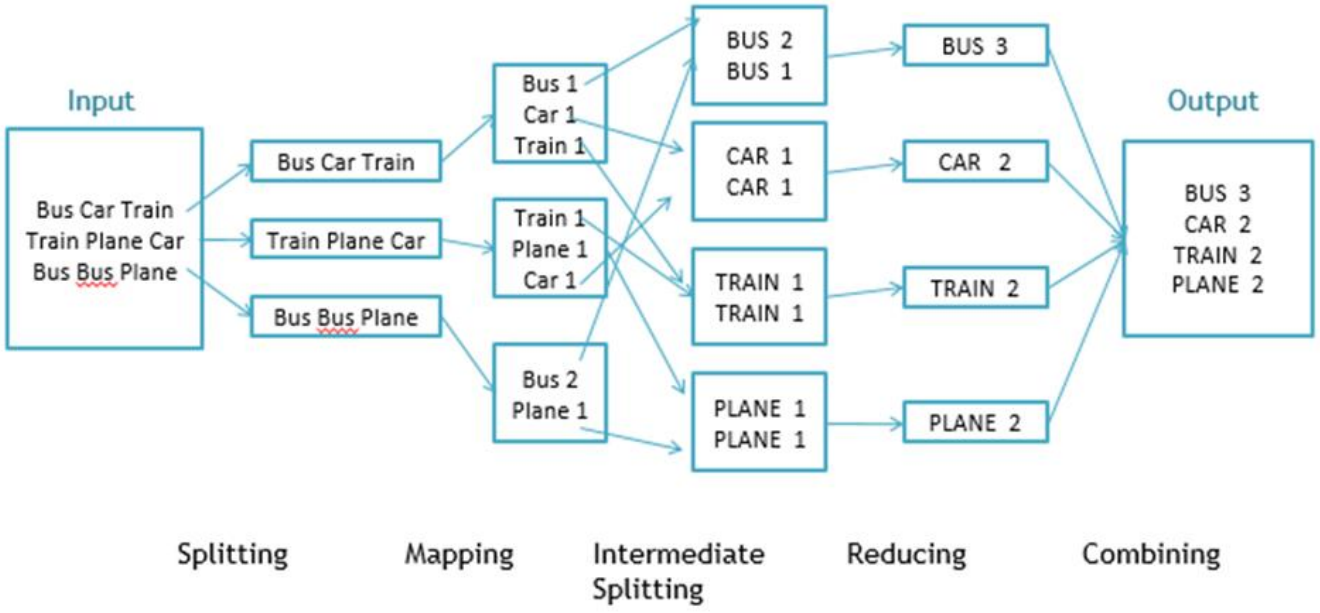


Fig. WorkFlow of MapReducing

WORD-COUNT WITH MAP-REDUCE

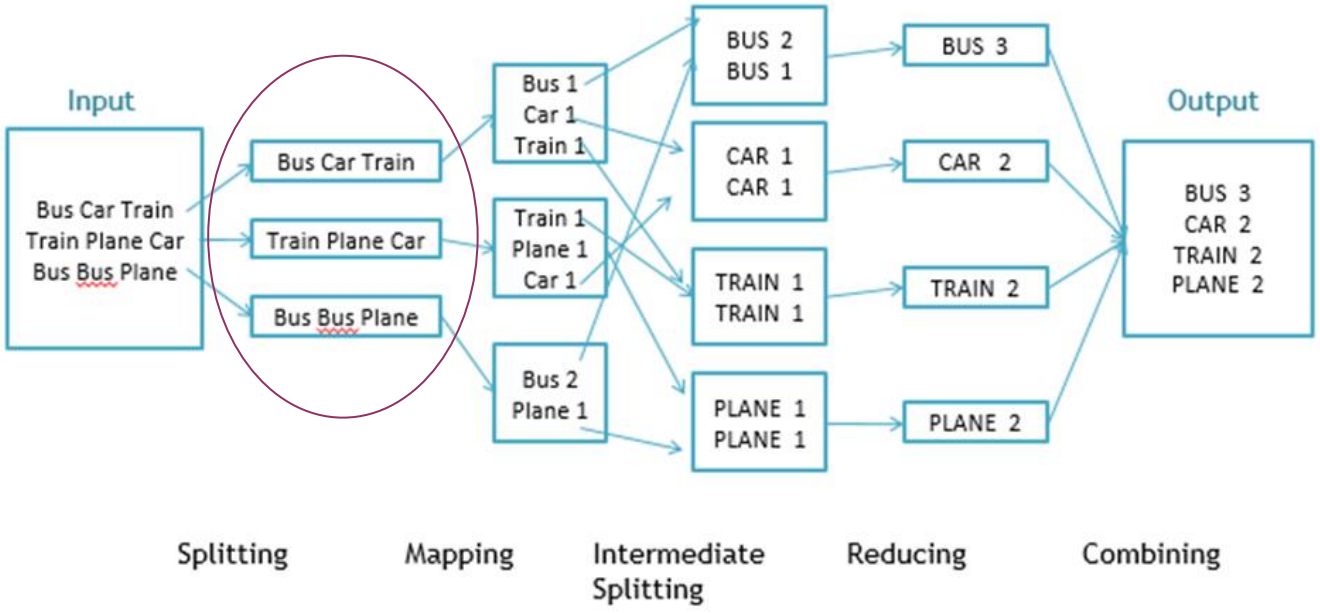
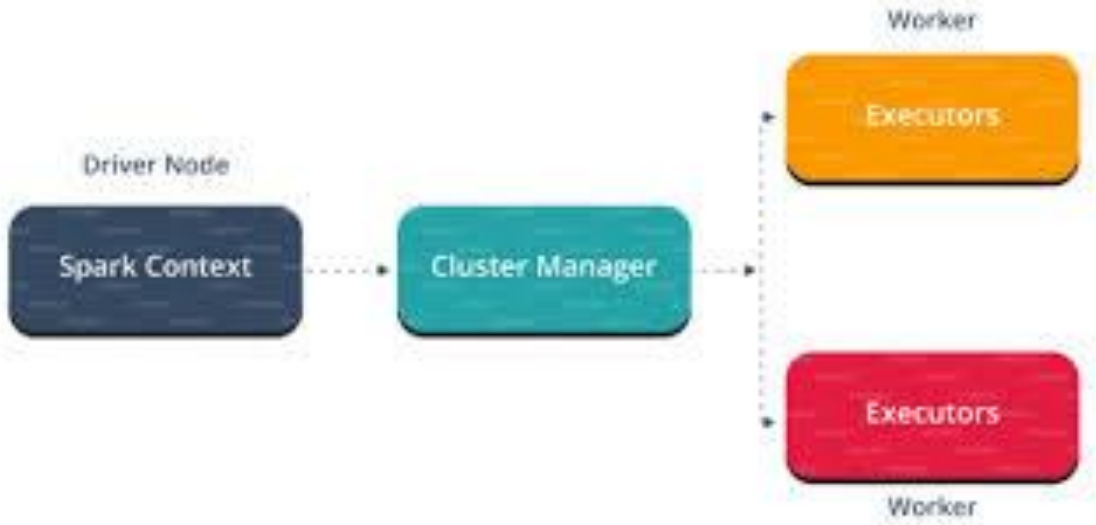


Fig. WorkFlow of MapReducing

SPARK

- It is the next generation paradigm developed for processing big data.
- It is an alternative to Map-Reduce.
- Written in Scala
- It supports Java, Scala and Python programming languages.

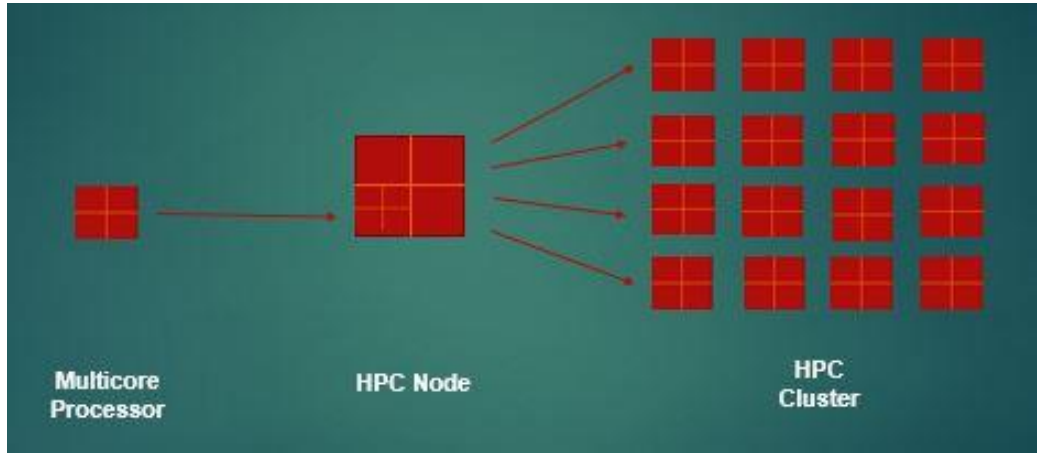


SPARK

- It can deliver up to 100x faster throughput than Hadoop Map-Reduce.
- Spark has a built-in machine learning library called MLlib, Hadoop has no such library.

HIGH PERFORMANCE COMPUTING (HPC) CLUSTER

- It is also called as supercomputer having thousand of processing cores.
- Includes powerful hardware
- It's not as scalable as Hadoop or Spark, but it can handle terabytes of data.
- The initial setup cost is quite high.
- The cost of scaling is high.
- MPI is used for communication.



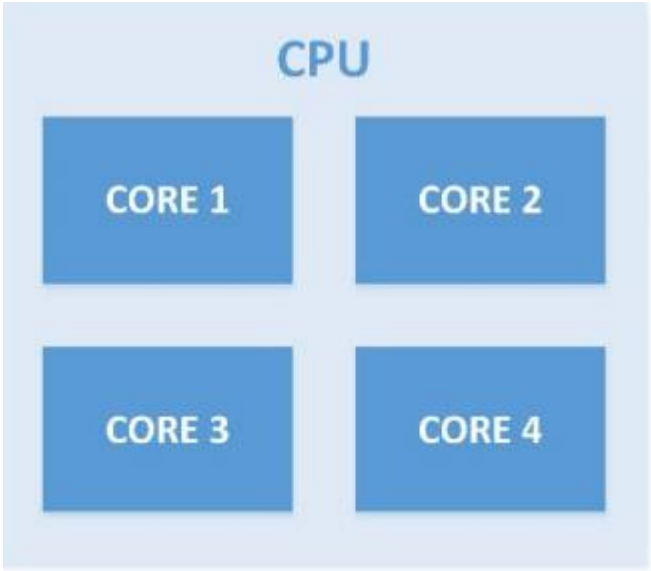
MULTI-CORE CPU

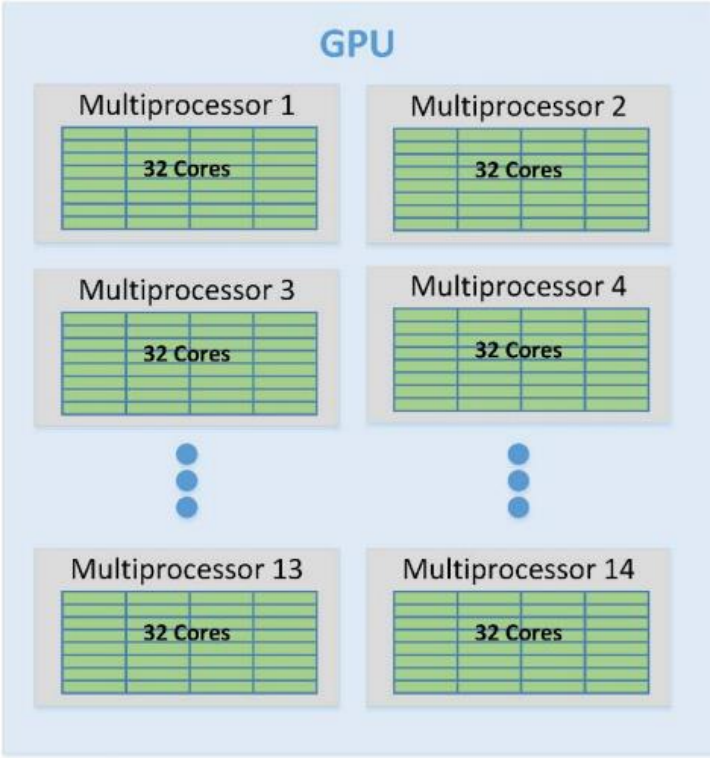
- A machine has dozens of processing core.
- Number of core per chip and the number of operation per core improve system performance.
- The next-generation mainboards allow that more CPU located in the single machine.
- Multithreading provide parallelism.
- Task should be divided in thread

GPU

- GPU is special hardware having parallel architecture.
- GPU architecture has more processing core.
- CPU architecture has its own DDR5 memory that is several time faster than the typical DDR3.
- NVIDIA CUDA is programming framework that is make simpler GPU programming.
- CUDA abstracts the user from low-level hardware details.

CPU-GPU


CPU
CORE 1
CORE 2
CORE 3
CORE 4


GPU
Multiprocessor 1
32 Cores
Multiprocessor 2
32 Cores
Multiprocessor 3
32 Cores
Multiprocessor 4
32 Cores
Multiprocessor 13
32 Cores
Multiprocessor 14
32 Cores

CPU & GPU

- Improvement in CPU is slower than improvement in GPU.
- The number of core in CPU is lesser than the number of core in GPU.
- The GPU is a good option, but the number of applications that take advantage of the GPU is quite limited.
- CPU provides task parallelism; GPU provides data parallelism.

FPGA

- FPGA is special hardware.
- FPGA is specially produced for specific application.
- The development cost is high.

COMPARISON OF BIG DATA PLATFORMS

Platforms (Communication Scheme)	System/Platform			Application/Algorithm		
	Scalability	Data I/O Performance	Fault Tolerance	Real-Time Processing	Data Size Supported	Iterative Task Support
Peer to Peer (TCP/IP)	★★★★★	★	★	★	★★★★★	★★
Virtual Clusters (MapRedce/MPI)	★★★★★	★★	★★★★★	★★	★★★★	★★
Virtual Clusters (Spark)	★★★★★	★★★	★★★★★	★★	★★★★	★★★
HPC Clusters (MPI/Mapreduce)	★★★	★★★★	★★★★	★★★	★★★★	★★★★
Multicore (Multithreading)	★★	★★★★	★★★★	★★★	★★	★★★★
GPU (CUDA)	★★	★★★★★	★★★★	★★★★★	★★	★★★★
FPGA (HDL)	★	★★★★★	★★★★	★★★★★	★★	★★★★

SCABILITY

Platforms (Communication Scheme)	System/Platform Scalability
Peer to Peer (TCP/IP)	★★★★★
Virtual Clusters (MapRedce/MPI)	★★★★★
Virtual Clusters (Spark)	★★★★★
HPC Clusters (MPI/Mapreduce)	★★★
Multicore (Multithreading)	★★
GPU (CUDA)	★★
FPGA (HDL)	★

Its scalability is quite high; it is relatively easy to expand to any size and add machines.

It can only be scaled up to a certain extent.

Scalability is limited by the number of GPUs and CPUs in a single machine.

It is quite costly to scale once deployed.

I/O PERFORMANCE

Platforms (Communication Scheme)	System/Platform Data I/O
Peer to Peer (TCP/IP)	★
Virtual Clusters (MapRedce/MPI)	★★
Virtual Clusters (Spark)	★★★
HPC Clusters (MPI/Mapreduce)	★★★★
Multicore (Multithreading)	★★★★
GPU (CUDA)	★★★★★
FPGA (HDL)	★★★★★

Network communication is slow and disk access is done.

Slower disk access.

Disk access is reduced, and system memory is used.

System memory is used.

DDR5 memory, which is faster than system memory, is used.

FAULT TOLERANT

Platforms (Communication Scheme)	System/Platform Fault Tolerance
Peer to Peer (TCP/IP)	★
Virtual Clusters (MapRedce/MPI)	★★★★★
Virtual Clusters (Spark)	★★★★★
HPC Clusters (MPI/Mapreduce)	★★★★
Multicore (Multithreading)	★★★★
GPU (CUDA)	★★★★
FPGA (HDL)	★★★★

It does not have a fault tolerance mechanism.

There is a fault tolerance mechanism inherent in the system.

Although these platforms do not have state of the art fault tolerance technology, they do have reliable and well-built hardware.

REAL-TIME PROCESSING

Platforms (Communication Scheme)	System/Platform
	Real-Time
Peer to Peer (TCP/IP)	★
Virtual Clusters (MapRedce/MPI)	★★
Virtual Clusters (Spark)	★★
HPC Clusters (MPI/Mapreduce)	★★★
Multicore (Multithreading)	★★★
GPU (CUDA)	★★★★★
FPGA (HDL)	★★★★★

There is no real-time data processing because the network communication is slow and has a generic hardware.

It does not contain powerful and optimized hardware. I/O operations are slow.

They have real-time processing capabilities. They have a lot of processor and memory bandwidth.

With thousands of processor cores and very high-speed memory, it is well suited for real-time processing.

DATA SIZE

Platforms (Communication Scheme)	System/Platform Data Size
Peer to Peer (TCP/IP)	★★★★★
Virtual Clusters (MapRedce/MPI)	★★★★
Virtual Clusters (Spark)	★★★★
HPC Clusters (MPI/Mapreduce)	★★★★
Multicore (Multithreading)	★★
GPU (CUDA)	★★
FPGA (HDL)	★★

It can handle petabytes of data

It can handle several Terabytes of data.

Not suitable for large-scale datasets. The multi-core system relies only on system memory, which can be up to a few hundred Gigabytes. Likewise, the on-board memory of the GPU is limited.

REPEATED TASK

Platforms (Communication Scheme)	System/Platform Iterative Tasks
Peer to Peer (TCP/IP)	★★
Virtual Clusters (MapRedce/MPI)	★★
Virtual Clusters (Spark)	★★★
HPC Clusters (MPI/Mapreduce)	★★★★
Multicore (Multithreading)	★★★★
GPU (CUDA)	★★★★
FPGA (HDL)	★★★★

P2P; has a large network communication load.

MapReduce; has disk input/output overhead.

Disc input/output load is reduced.

They are suitable for iterative operations. Iterative algorithms must be redesigned for each platform.