

Big Data Analytics

Scope

- Research issues
- Existing big data tools
- Big data architectures

Research issues

- Volume, Velocity, Variety
- Scalability
- Data integration
- Skills availability
- Solution cost
- Algorithms and tools redesign
- Applicability limits
- Making sense of the explosion of data
- Understanding a growing variety of data

Research issues

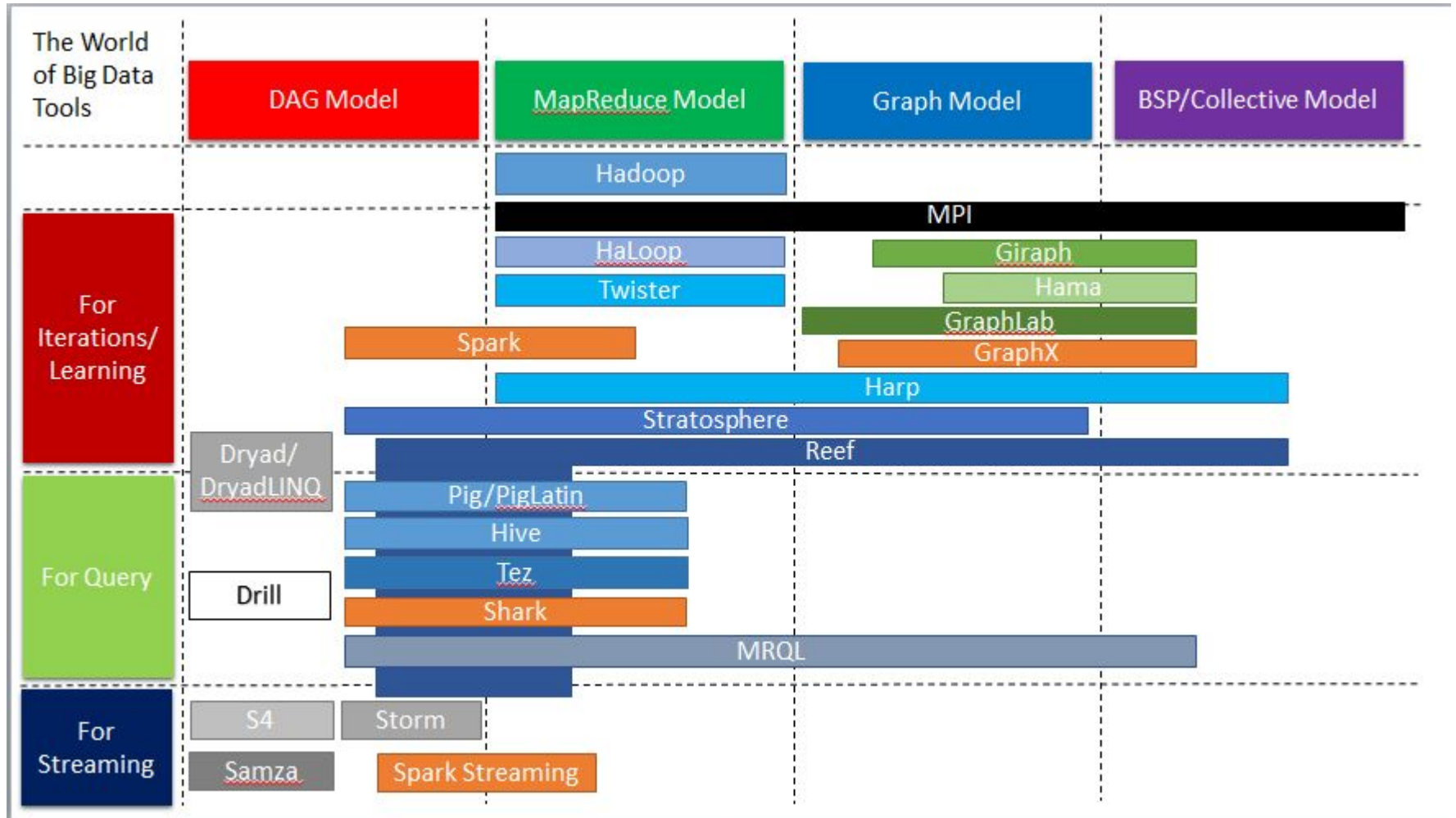
- *Enabling real-time analysis of data*
- *Achieving simplified deployment and management*
- *Complexity*
- *Privacy*
- *Access and share*
- *Data visualization:*
 - *Meeting the need for speed*
 - *Understanding the data*
 - *Addressing data quality*
 - *Displaying meaningful results*
 - *Dealing with outliers*

Big trends in big data analytics

- Big data analytics on cloud
- Hadoop: the new enterprise data operating systems
- Big data lake
- More predictive analysis
- Sql on hadoop
- Nosql
- Deep learning
- In-memory analytics

EXISTING BIG DATA TOOLS

World map of big data tools




Big Data Tools for HPC and Supercomputing

- MPI(Message Passing Interface, 1992)
 - Provide standardized function interfaces for communication between parallel processes.
- Collective communication operations
 - Broadcast, Scatter, Gather, Reduce, Allgather, Allreduce, Reduce-scatter.
- Popular implementations
 - MPICH (2001)
 - OpenMPI (2004)
 - <http://www.open-mpi.org/>

BIG DATA ANALYTICS TOOLS ON CLOUD

MapReduce Model

- Google MapReduce (2004)
 - Jeffrey Dean et al. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.
- Apache Hadoop (2005) 
 - <http://hadoop.apache.org/>
 - <http://developer.yahoo.com/hadoop/tutorial/>
- Apache Hadoop 2.0 (2012)
 - Vinod Kumar Vavilapalli et al. Apache Hadoop YARN: Yet Another Resource Negotiator, SOCC 2013.
 - Separation between resource management and computation model.

Key Features of MapReduce Model

- Designed for clouds
 - Large clusters of commodity machines
- Designed for big data
 - Support from local disks based distributed file system (GFS / HDFS)
 - Disk based intermediate data transfer in Shuffling
- MapReduce programming model
 - Computation pattern: Map tasks and Reduce tasks
 - Data abstraction: KeyValue pairs

Iterative MapReduce Model



- Twister (2010)

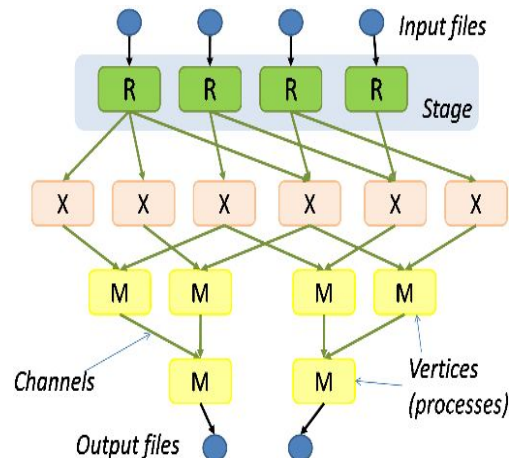
- Jaliya Ekanayake et al. Twister: A Runtime for Iterative MapReduce. HPDC workshop 2010.
- <http://www.iterativemapreduce.org/>
- Simple collectives: broadcasting and aggregation.

- HaLoop (2010)

- Yingyi Bu et al. HaLoop: Efficient Iterative Data Processing on Large clusters. VLDB 2010.
- <http://code.google.com/p/haloop/>
- Programming model $R_{i+1} = R_0 \cup (R_i \bowtie L)$
- Loop-Aware Task Scheduling
- Caching and indexing for Loop-Invariant Data on local disk

DAG (Directed Acyclic Graph) Model

- Dryad and DryadLINQ (2007)
 - Michael Isard et al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks, EuroSys, 2007.
 - <http://research.microsoft.com/en-us/collaboration/tools/dryad.aspx>



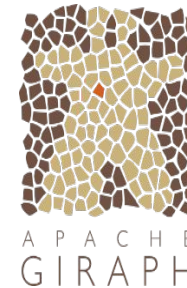
Model Composition



- Apache Spark (2010)
 - Matei Zaharia et al. Spark: Cluster Computing with Working Sets,. HotCloud 2010.
 - Matei Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012.
 - <http://spark.apache.org/>
 - Resilient Distributed Dataset (RDD)
 - RDD operations
 - MapReduce-like parallel operations
 - DAG of execution stages and pipelined transformations
 - Simple collectives: broadcasting and aggregation

Graph Processing with BSP model

- Pregel (2010)
 - Grzegorz Malewicz et al. Pregel: A System for Large-Scale Graph Processing. SIGMOD 2010.
- Apache Hama (2010)
 - <https://hama.apache.org/>
- Apache Giraph (2012)
 - <https://giraph.apache.org/>
 - Scaling Apache Giraph to a trillion edges
 - <https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920>



GraphLab (2010)



- Yucheng Low et al. GraphLab: A New Parallel Framework for Machine Learning. UAI 2010.
- Yucheng Low, et al. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. PVLDB 2012.
- <http://graphlab.org/projects/index.html>
- <http://graphlab.org/resources/publications.html>
- Data graph
- Update functions and the scope
- Sync operation (similar to aggregation in Pregel)

Collective Model

- Harp (2013)



- <https://github.com/jessezbj/harp-project>
- Hadoop Plugin (on Hadoop 1.2.1 and Hadoop 2.2.0)
- Hierarchical data abstraction on arrays, key-values and graphs for easy programming expressiveness.
- Collective communication model to support various communication operations on the data abstractions.
- Caching with buffer management for memory allocation required from computation and communication
- BSP style parallelism
- Fault tolerance with check-pointing

K-means Clustering Parallel Efficiency

- Shantenu Jha et al. A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures. 2014.

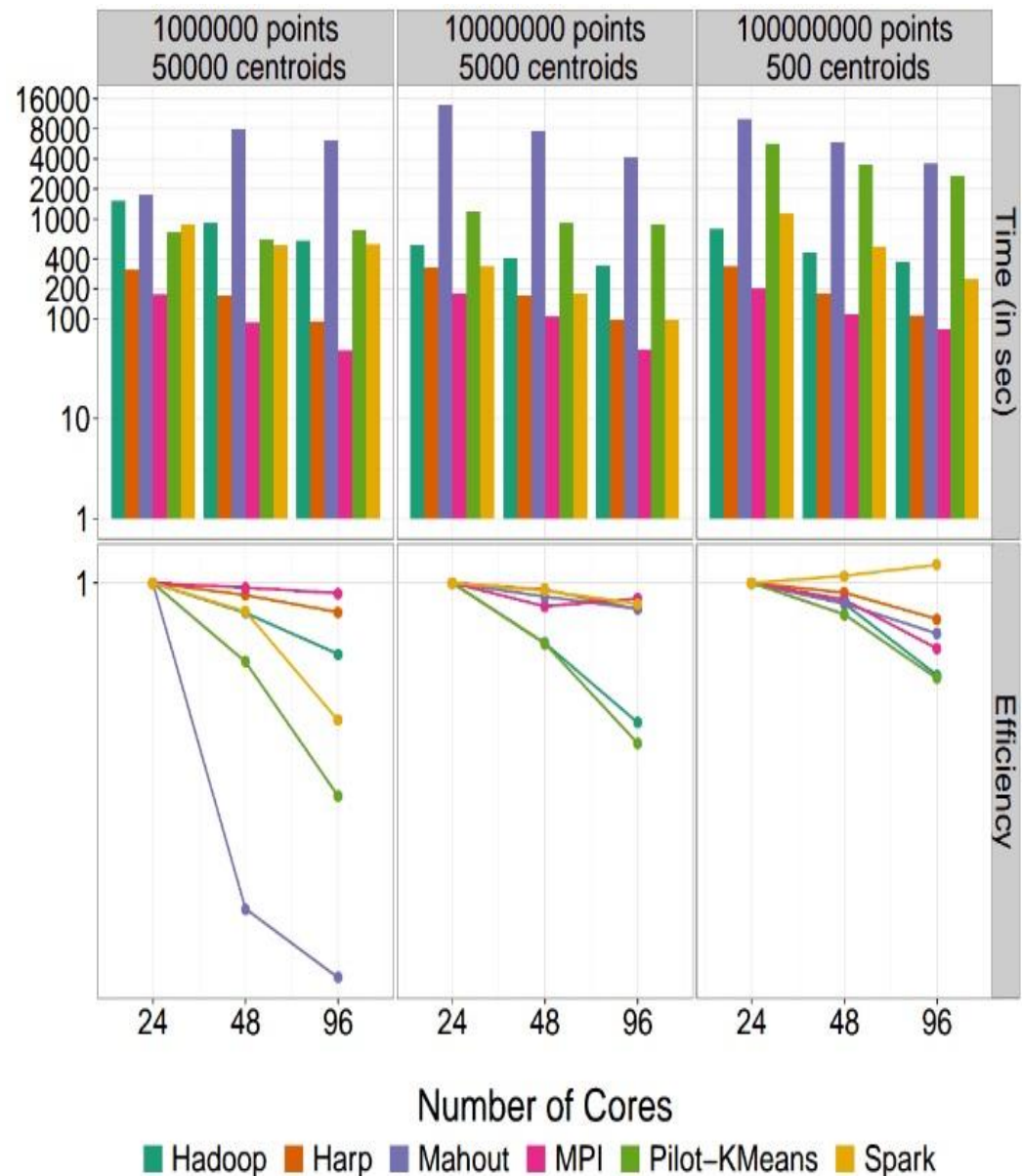


Fig. 5. Time-To-Completion for KMeans on Different Backends

Machine Learning on Big Data

- Mahout on Hadoop



- <https://mahout.apache.org/>

- MLlib on Spark


- <http://spark.apache.org/mllib/>

- GraphLab Toolkits

- <http://graphlab.org/projects/toolkits.html>

- GraphLab Computer Vision Toolkit

Query on Big Data

- Query with procedural language
- Google Sawzall (2003)
 - Rob Pike et al. Interpreting the Data: Parallel Analysis with Sawzall. Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure 2003.
- Apache Pig (2006) 
 - Christopher Olston et al. Pig Latin: A Not-So-Foreign Language for Data Processing. SIGMOD 2008.
 - <https://pig.apache.org/>

SQL-like Query



- Apache Hive (2007)
 - Facebook Data Infrastructure Team. Hive - A Warehousing Solution Over a Map-Reduce Framework. VLDB 2009.
 - <https://hive.apache.org/>
 - On top of Apache Hadoop
- Shark (2012)
 - Reynold Xin et al. Shark: SQL and Rich Analytics at Scale. Technical Report. UCB/EECS 2012.
 - <http://shark.cs.berkeley.edu/>
 - On top of Apache Spark
- Apache MRQL (2013)
 - <http://mrql.incubator.apache.org/>
 - On top of Apache Hadoop, Apache Hama, and Apache Spark

Other Tools for Query

- Apache Tez (2013)
 - <http://tez.incubator.apache.org/>
 - To build complex DAG of tasks for Apache Pig and Apache Hive
 - On top of YARN
- Dremel (2010) Apache Drill (2012)
 - Sergey Melnik et al. Dremel: Interactive Analysis of Web-Scale Datasets. VLDB 2010.
 - <http://incubator.apache.org/drill/index.html>
 - System for interactive query

Stream Data Processing

- Apache S4 (2011)
 - <http://incubator.apache.org/s4/>
- Apache Storm (2011)
 - <http://storm.incubator.apache.org/>
- Spark Streaming (2012)
 - <https://spark.incubator.apache.org/streaming/>
- Apache Samza (2013)
 - <http://samza.incubator.apache.org/>

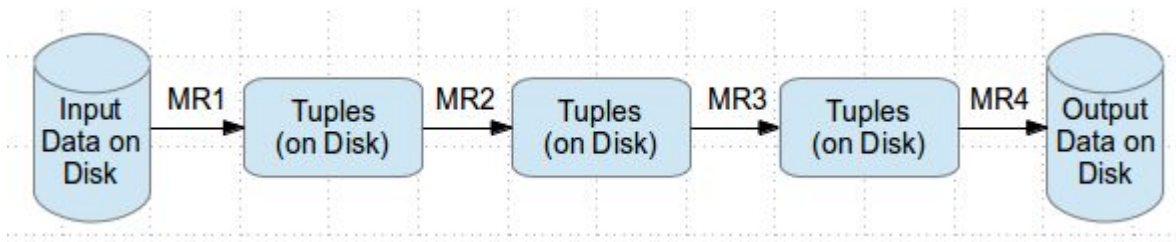
Apache Spark

In-Memory Data Processing

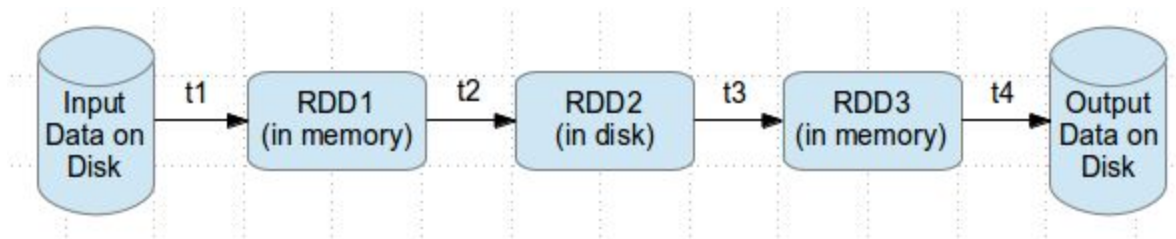
Why Spark ?

- Most of Machine Learning Algorithms are iterative because each iteration can improve the results
- With Disk based approach each iteration's output is written to disk making it slow

Hadoop execution flow



Spark execution flow



<http://www.wiziq.com/blog/hype-around-apache-spark/>

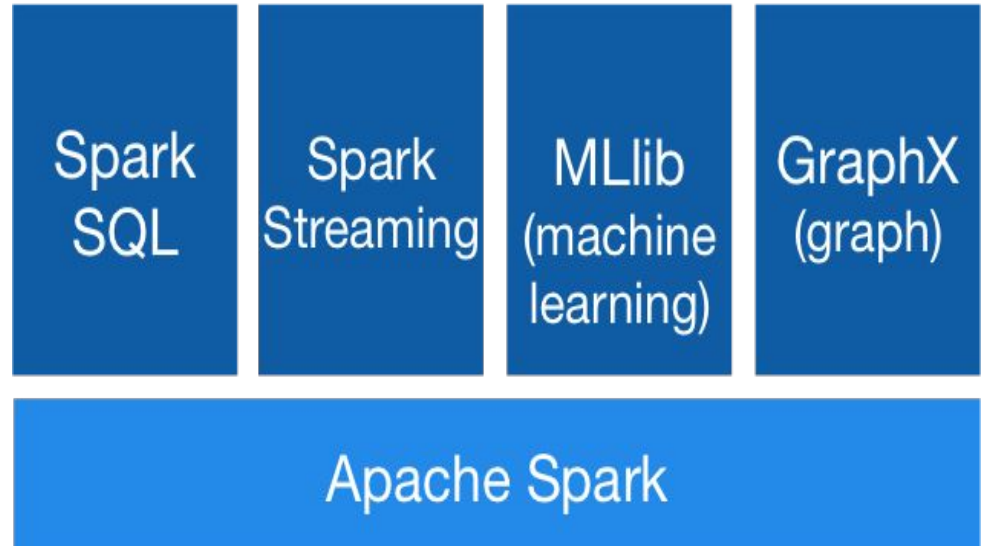
About Apache Spark



- Initially started at UC Berkeley in 2009
- Fast and general purpose cluster computing system
- 10x (on disk) - 100x (In-Memory) faster
- Most popular for running *Iterative Machine Learning Algorithms*.
- Provides high level APIs in
 - Java
 - Scala
 - Python
- Integration with Hadoop and its eco-system and can **read existing data**.
- <http://spark.apache.org/>

Spark Stack

- Spark SQL
 - For SQL and unstructured data processing
- MLib
 - Machine Learning Algorithms
- GraphX
 - Graph Processing
- Spark Streaming
 - stream processing of live data streams



<http://spark.apache.org>

How to select right platform/tools/technology??

Key questions

- 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 a single iteration?

Technological concern

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

Scalable platforms

Scaling is the ability of the system to adapt to increased demands in terms of data processing.

Horizontal Scaling

Horizontal scaling 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.

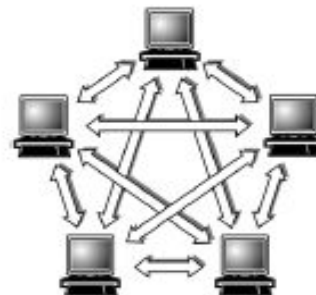
Big data Hardware platforms

Scaling	Advantages	Drawbacks
Horizontal scaling	<ul style="list-style-type: none">• Setup and upgrade cost is relatively less• More scalable• performance improvement is easy	<ul style="list-style-type: none">• Software has to handle all the data distribution and parallel processing complexities• Limited number of software are available that can take advantage of horizontal scaling
Vertical scaling	<ul style="list-style-type: none">• Management and installation of hardware is easy	<ul style="list-style-type: none">• Setup cost is high• System has to be more powerful to handle future workloads and initially the additional performance goes to waste• Scalable up to certain limit

Horizontal Scaling Platforms

- Some prominent horizontal scaling platforms:

- ◆ Peer to Peer Networks



- ◆ Apache Hadoop



- ◆ Apache Spark



Vertical Scaling Platforms

- Most prominent vertical scaling platforms:

- ◆ High Performance Computing Clusters (HPC)



- ◆ Multicore Processors



- ◆ Graphics Processing Unit (GPU)



- ◆ Field Programmable Gate Arrays (FPGA)



CPU Vs. GPU

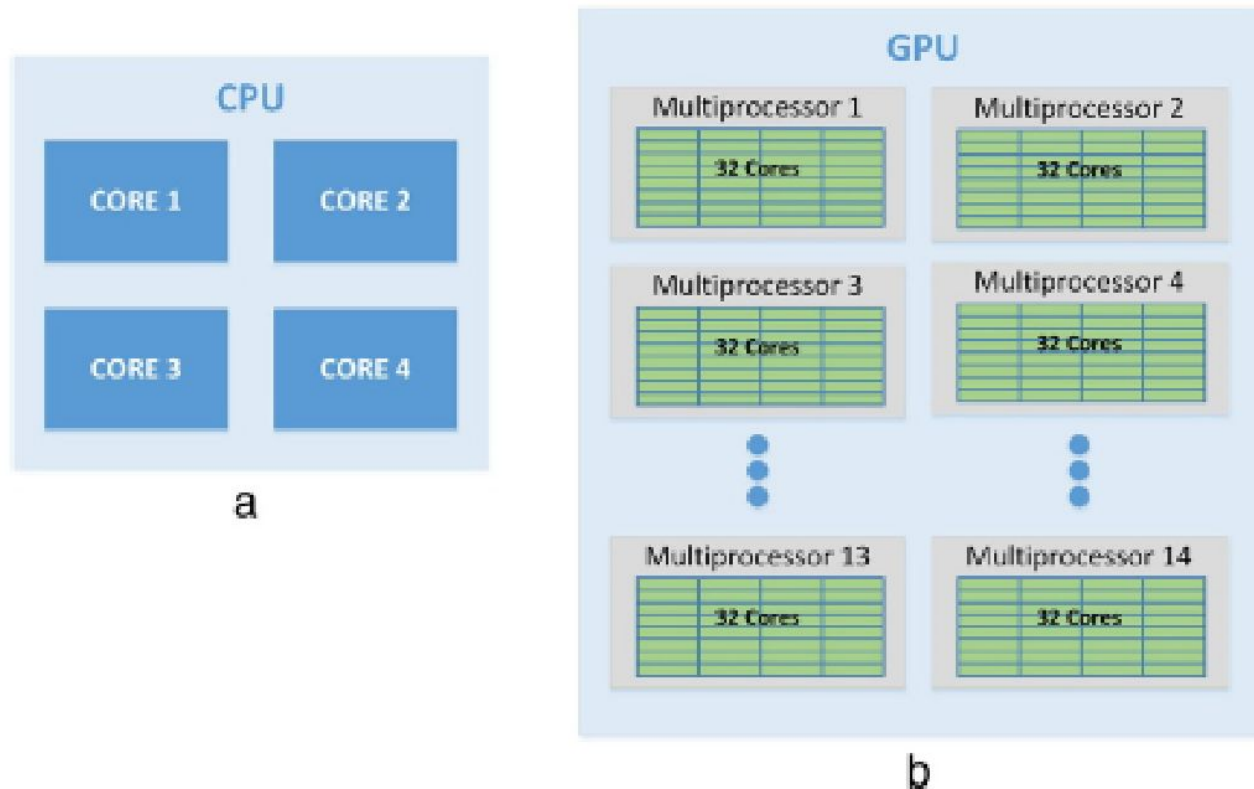


Figure 3 A comparison between the architectures of CPU(a) and GPU(b) showing the arrangement of processing cores.

Comparison analysis

Table 2 Comparison of different platforms (along with their communication mechanisms) based on various characteristics

Scaling type	Platforms (Communication Scheme)	System/Platform			Application/Algorithm		
		Scalability	Data I/O performance	Fault tolerance	Real-time processing	Data size supported	Iterative task support
Horizontal scaling	Peer-to-Peer (TCP/IP)	★★★★★	★	★	★	★★★★★	★★
	Virtual clusters (MapReduce/MPI)	★★★★★	★★	★★★★★	★★	★★★★	★★
	Virtual clusters (Spark)	★★★★★	★★★★	★★★★★	★★	★★★★	★★★★
Vertical scaling	HPC clusters (MPI/Mapreduce)	★★★★	★★★★	★★★★	★★★★	★★★★	★★★★
	Multicore (Multithreading)	★★	★★★★	★★★★	★★★★	★★	★★★★
	GPU (CUDA)	★★	★★★★★	★★★★	★★★★★	★★	★★★★
	FPGA (HDL)	★	★★★★★	★★★★	★★★★★	★★	★★★★