Platforms and Algorithms for Big Data Analytics

Chandan K. Reddy

Department of Computer Science

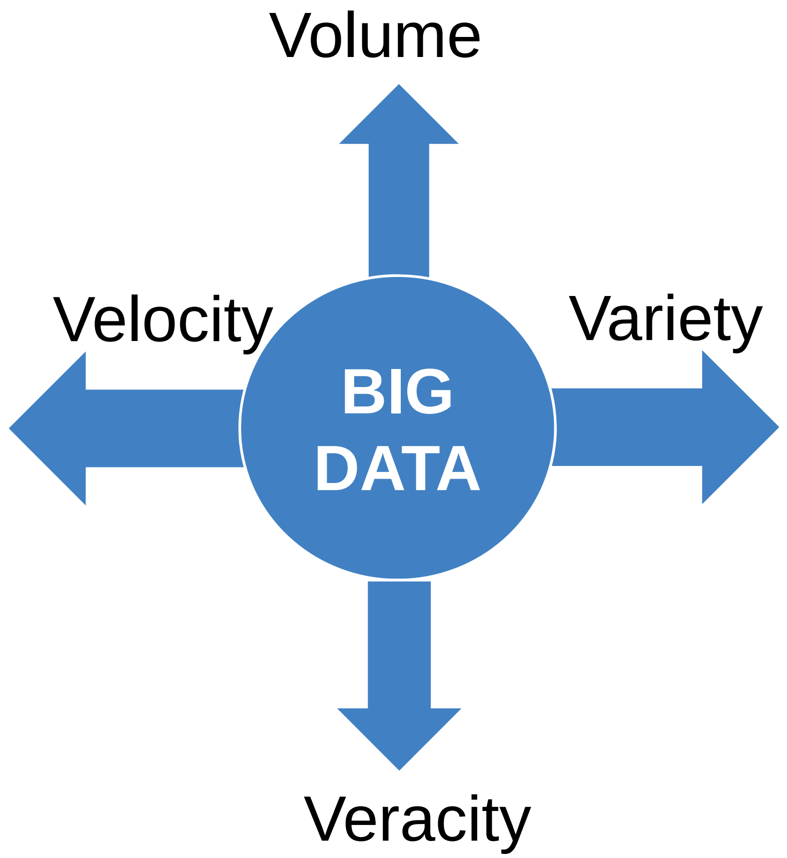
Wayne State University

http://www.cs.wayne.edu/~reddy/

http://dmkd.cs.wayne.edu/TUTORIAL/Bigdata/

What is Big Data?

A collection of large and complex data sets which are difficult to process using common database management tools or traditional data processing applications.



The four dimensions (V's) of Big Data

Big data is not just about size.

- Finds insights from complex, noisy, heterogeneous, streaming, longitudinal, and voluminous data.
- It aims to answer questions that were previously unanswered.

The challenges include capture, storage, search, sharing & analysis.

Data Accumulation !!!

- Data is being collected at rapid pace due to the advancements in sensing technologies.
- Storage has become extremely cheap and hence no one wants to throw away the data. The assumption here is that they will be using it in the future.
- Estimates show that the amount of digital data accumulated until 2010 has been gathered within the next two years. This shows the growth in the digital world.
- Analytics is still lagging behind compared to sensing and storage developments.

Why Should YOU CARE ?

- **JOBS !!**

- The U.S. could face a shortage by 2018 of 140,000 to 190,000 people with "deep analytical talent" and of 1.5 million people capable of analyzing data in ways that enable business decisions. (McKinsey & Co)
- Big Data industry is worth more than $100 billion
- Growing at almost 10% a year (roughly twice as fast as the software business)

- **Digital World in the future !!**

- The world will become more and more digital and hence big data is only going to get BIGGER !!
- This is an era of big data

Why we need more Powerful Platforms ?

- The choice of hardware/software platform plays a crucial role to achieve one's required goals.
- To analyze this voluminous and complex data, scaling up is imminent.
- In many applications, analysis tasks need to produce results in real-time and/or for large volumes of data.
- It is no longer possible to do real-time analysis on such big datasets using a single machine running commodity hardware
- Continuous research in this area has led to the development of many different algorithms and big data platforms

THINGS TO THINK ABOUT !!!!

- **Application/Algorithm level requirements…**
  - 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?

- **Systems/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?

Outline of this Tutorial

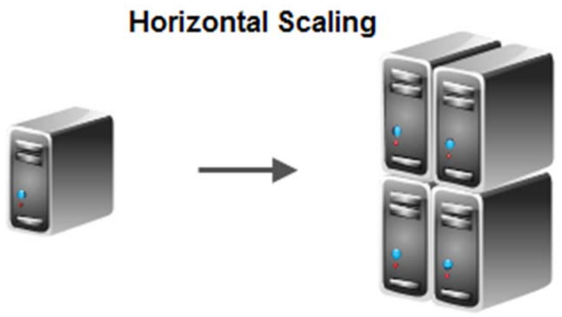
- Introduction
- Scaling
- Horizontal Scaling Platforms
  - Peer to Peer
  - Hadoop
  - Spark
- Vertical Scaling Platforms
  - High Performance Computing (HPC) Clusters
  - Multicore
  - Graphical Processing Unit (GPU)
  - Field Programmable Gate Array (FPGA)
- Comparison of Different Platforms
- Big Data Analytics on Amazon EC2 Clusters

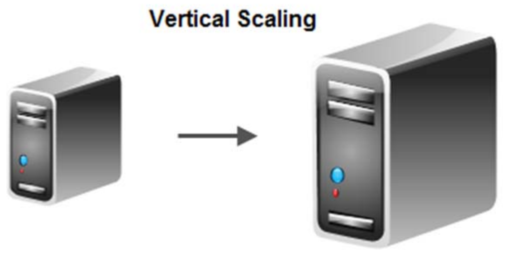
Outline

- Introduction
- Scaling
- Horizontal Scaling Platforms
  - Peer to Peer
  - Hadoop
  - Spark
- Vertical Scaling Platforms
  - High Performance Computing (HPC) clusters
  - Multicore
  - Graphical Processing Unit (GPU)
  - Field Programmable Gate Array (FPGA)
- Comparison of Different Platforms
- Big Data Analytics on Amazon EC2 Clusters

Scaling

- Scaling is the ability of the system to adapt to increased demands in terms of processing
- Two types of scaling :
  - **Horizontal Scaling**
    - Involves distributing work load across many servers
    - Multiple machines are added together to improve the processing capability
    - Involves multiple instances of an operating system on different machines
  - **Vertical Scaling**
    - Involves installing more processors, more memory and faster hardware typically within a single server
    - Involves single instance of an operating system



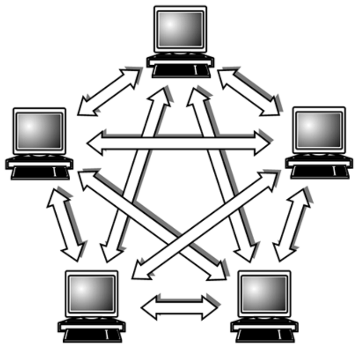


Horizontal vs Vertical Scaling

Scaling	Advantages	Drawbacks
Horizontal Scaling	→ Increases performance in small steps as needed → Financial investment to upgrade is relatively less → Can scale out the system as much as needed	→ 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	→ Most of the Software can easily take advantage vertical scaling → Easy to manage and install hardware within a single machine	→ Requires substantial financial investment → System has to be more powerful to handle future workloads and initially the additional performance goes to waste → It is not possible to scale up vertically after a 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)

Outline

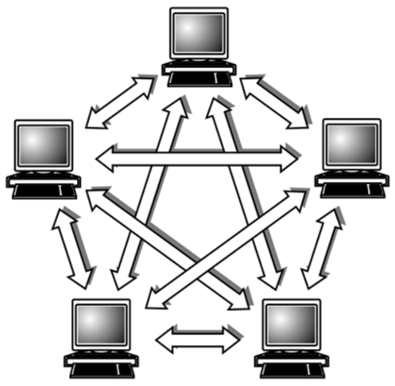
- Introduction
- Scaling
- Horizontal Scaling Platforms
  - Peer to Peer
  - Hadoop
  - Spark
- Vertical Scaling Platforms
  - High Performance Computing (HPC) clusters
  - Multicore
  - Graphical Processing Unit (GPU)
  - Field Programmable Gate Array (FPGA)
- Comparison of Different Platforms
- Big Data Analytics on Amazon EC2 Clusters

Peer to Peer Networks

- Typically involves millions of machines connected in a network
- Decentralized and distributed network architecture
- Message Passing Interface (MPI) is the communication scheme used
- Each node capable of storing and processing data
- Scale is practically unlimited (can be millions of nodes)

Main Drawbacks

- Communication is the major bottleneck
- Broadcasting messages is cheaper but aggregation of results/data is costly
- Poor Fault tolerance mechanism

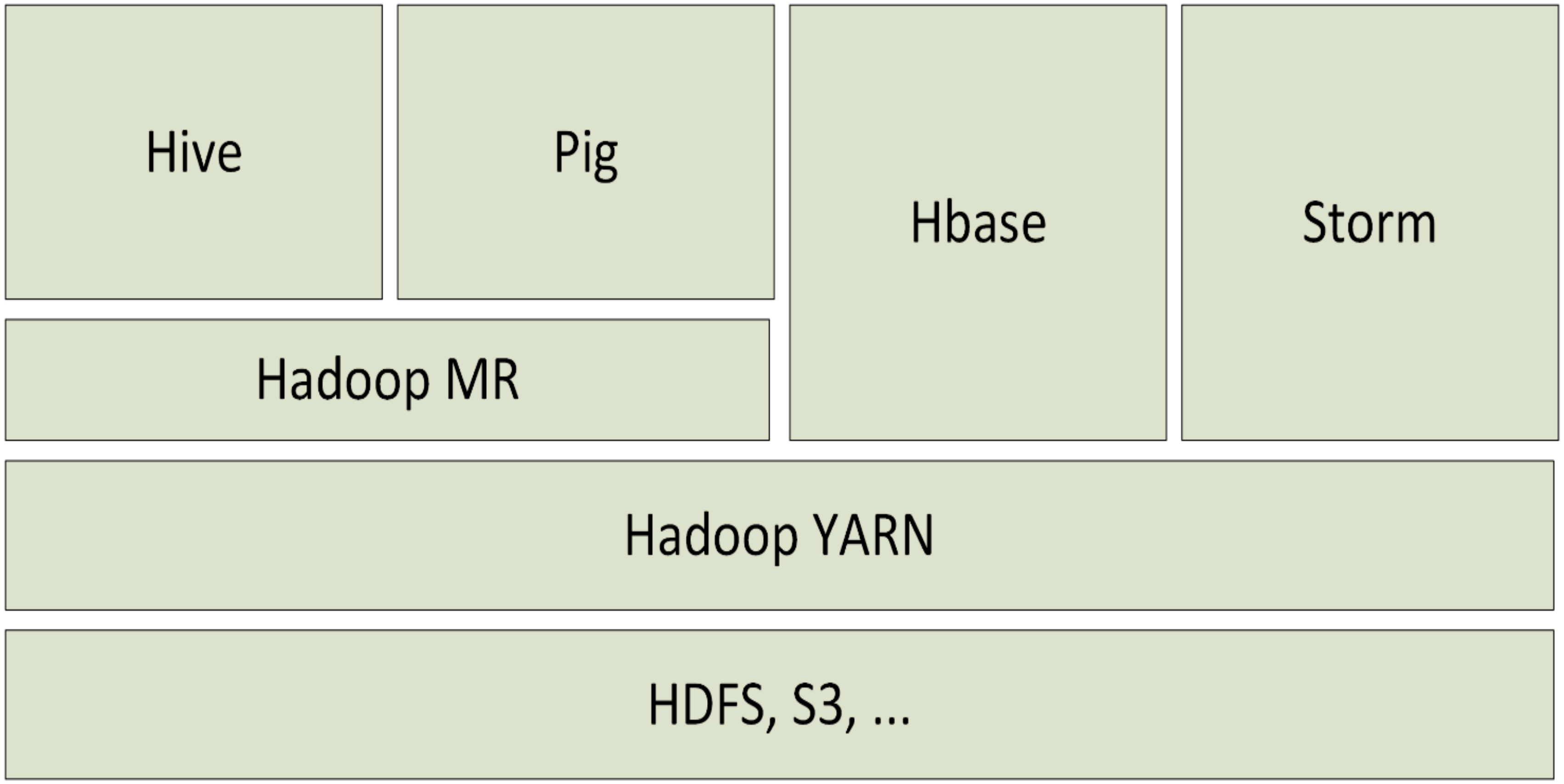


Apache Hadoop

- Open source framework for storing and processing large datasets
- Highly fault tolerance and designed to be used with commodity hardware

- Consists of two important components:
  - HDFS (Hadoop Distributed File System)
    - Used to store data across cluster of commodity machines while providing high availability and fault tolerance
  - Hadoop YARN
    - Resource management layer
    - Schedules jobs across the cluster

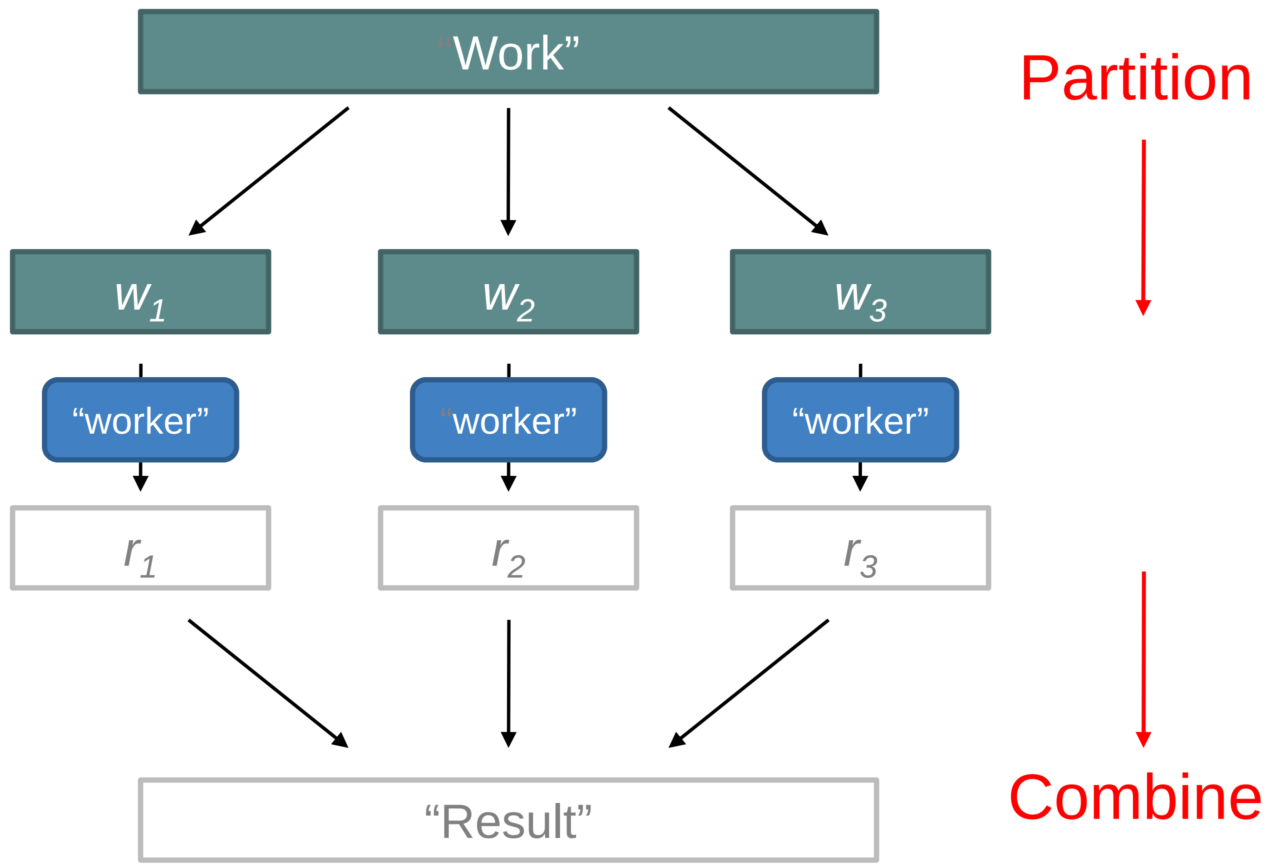
Hadoop Architecture



Hadoop MapReduce

- Basic data processing scheme used in Hadoop
- Includes breaking the entire scheme into mappers and reducers
  - Mappers read data from HDFS, process it and generate some intermediate results
  - Reducers aggregate the intermediate results to generate the final output and write it to the HDFS
- Typical Hadoop job involves running several mappers and reducers across the cluster

Divide and Conquer Strategy



MapReduce Wrappers

- Provide better control over MapReduce code
- Aid in code development
- Popular map reduce wrappers include:
  - Apache Pig
    - SQL like environment developed at Yahoo
    - Used by many organizations including Twitter, AOL, LinkedIn and more
  - Hive
    - Developed by Facebook

Both these wrappers are intended to make code development easier without having to deal with the complexities of MapReduce coding

Spark

- Next generation paradigm for big data processing
- Developed by researchers at University of California, Berkeley
- Used as an alternative to Hadoop
- Designed to overcome disk I/O and improve performance of earlier systems
- Allows data to be cached in memory eliminating the disk overhead of earlier systems
- Supports Java, Scala and Python
- Tested upto 100x faster than Hadoop MapReduce

Outline

High Performance Computing (HPC) Clusters

- Also known as Blades or supercomputers with thousands of processing cores
- Can have different variety of disk organization and communication mechanisms
- Contains well built powerful hardware optimized for speed and throughput
- Fault tolerance is not critical because of top quality high end hardware
- Not as scalable as Hadoop or Spark but can handle terabytes of data
- High initial cost of deployment
- Cost of scaling up is high
- MPI is typically the communication scheme used

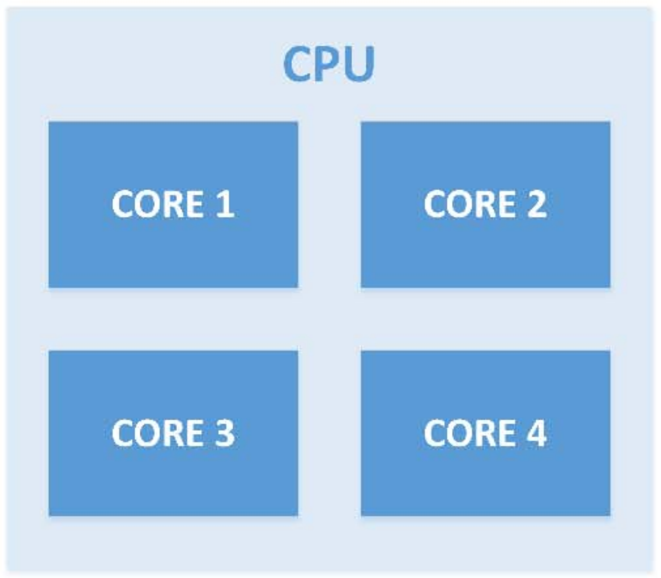
Multicore CPU

- One machine having dozens of processing cores
- Number of cores per chip and number of operations a core can perform has increased significantly
- Newer breed of motherboards allow multiple CPUs within a single machine
- Parallelism achieved through multithreading
- Task has to be broken into threads

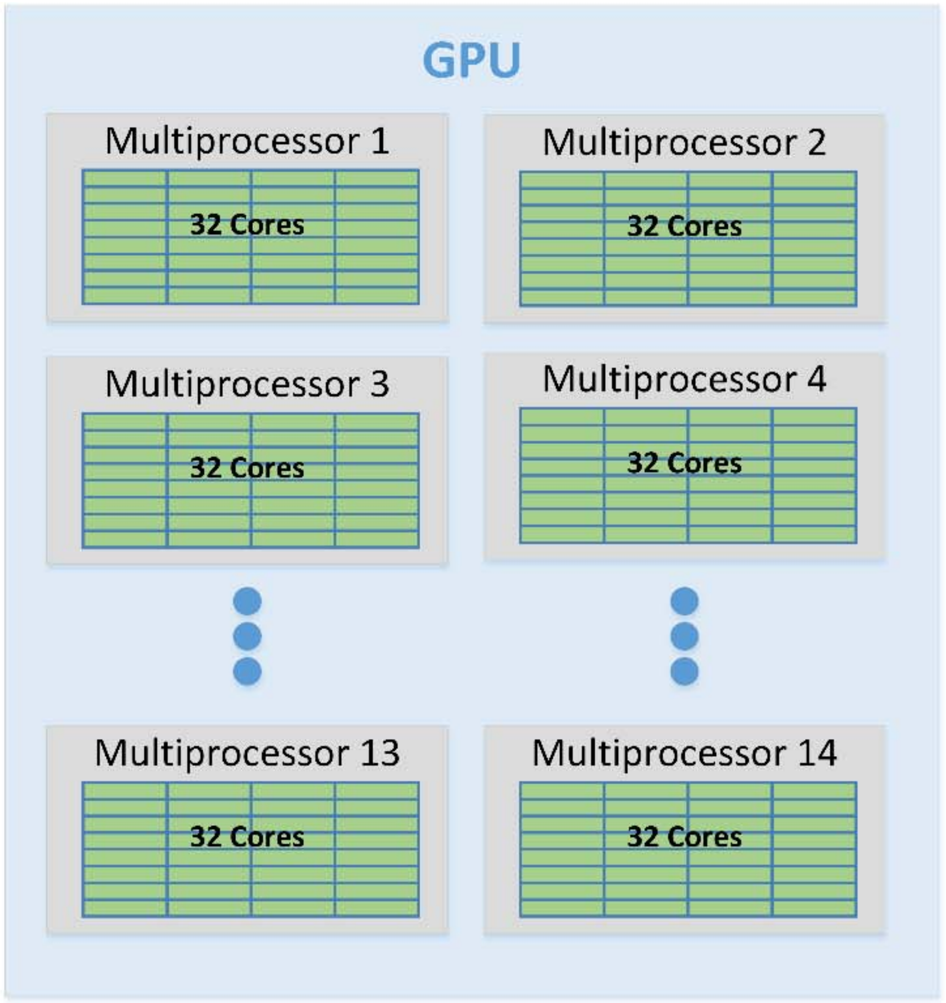
Graphics Processing Unit

- Specialized hardware with massively parallel architecture
- Recent developments in GPU hardware and programming frameworks has given rise to GPUGPU (general purpose computing on graphics processing units)
- Has large number of processing cores (typically around 2500+ currently)
- Has it's own DDR5 memory which is many times faster than typical DDR3 system memory
- Nvidia CUDA is the programming framework which simplifies GPU programming
- Using CUDA, one doesn't have to deal with low-level hardware details

GPU vs CPU Architecture



(a)



(b)

CPU vs GPU

- Development in CPU is rather slow as compared with GPU
- Number of cores in CPU is still in double digits while a GPU can have 2500+ cores
- Processing power of a current generation CPU is close to 10 Gflops while GPU can have close to 1000 Gflops of computing power
- CPU primarily relies on system memory which is slower than the GPU memory
- While GPU is an appealing option for parallel computing, the number of softwares and applications that take advantage of the GPU is rather limited
- CPU has been around for many years and huge number of software are available which use multicore CPU’s

Field Programmable Gate Arrays (FPGA)

- Highly specialized hardware units
- Custom built for specific applications
- Can be highly optimized for speed
- Due to customized hardware, development cost is much higher
- Coding has to be done in HDL (Hardware Description Language) with low level knowledge of hardware
- Greater algorithm development cost
- Suited for only certain set of applications

Outline

- Introduction
- Scaling
- Horizontal Scaling Platforms
  - Peer to Peer
  - Hadoop
  - Spark
- Vertical Scaling Platforms
  - High Performance Computing (HPC) clusters
  - Multicore
  - Graphical Processing Unit (GPU)
  - Field Programmable Gate Array (FPGA)
- Comparison of Different Platforms
- Big Data Analytics on Amazon EC2 Clusters

Comparison of Different Platforms

- Following characteristics are used for comparison:
  - System/Platform dependent
    - Scalability
    - Data I/O performance
    - Fault Tolerance
  - Application/Algorithm dependent
    - Real-time processing
    - Data size support
    - Support for iterative tasks

- Comparison is done using the star ratings
- 5 stars correspond to highest possible rating
- 1 star is the lowest possible rating

Comparison of 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)	★	★★★★★	★★★★	★★★★★	★★	★★★★

Scalability

- Ability of the system to handle growing amount of work load in a capable manner or to be enlarged to accommodate that growth.
- It is the ability to add more hardware to improve the performance and capacity of the system

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)	★

Highly scalable and it is relatively easy to add machines and extend them to any extent

Can only scale up to a certain extent

Limited number of GPUs and CPUs in a single machine

Once deployed, scaling up becomes costly

Data I/O Performance

- The rate at which the data is transferred to/from a peripheral device. In the context of big data analytics, this can be viewed as the rate at which the data is read and written to the memory (or disk) or the data transfer rate between the nodes in a cluster.

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)	★★★★★

Disk access and slow network communication

Slower disk access

Uses system memory; minimizes disk access

Uses system memory; usually within a single machine

Use DDR5 memory which is faster than system memory

Fault Tolerance

- The characteristic of a system to continue operating properly in the event of a failure of one or more components

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)	★★★★

Have no fault tolerance mechanism and use of commodity hardware makes them highly susceptible to system failures

Have in-built efficient fault tolerance mechanism

Although these platforms don't have state-of-the-art fault tolerance mechanisms, these have most reliable and well-built hardware which makes hardware failure an extremely rare event

Real-Time Processing

- The system’s ability to process the data and produce the results strictly within certain time constraints

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)	★★★★★

Slow for real-time data processing because of network overhead and commodity hardware

Slow in terms of data I/O and do not contain optimized and powerful hardware

Have reasonable real-time processing capabilities. They have many processing cores and high memory bandwidth

Well suited for real-time processing with thousands of processing cores and very high speed memory

Data Size Supported

- The size of the dataset that a system can process and handle efficiently

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)	★★

Can handle Petabytes of data and can scale out to unlimited number of nodes

Can handle around several Terabytes of data

Not suited for large-scale datasets. Multicore relies on system memory which can only be up to few hundred Gigabytes. Similarly, GPU has limited on board memory.

Iterative Task Support

- This is the ability of a system to efficiently support iterative tasks. Since many of the data analysis tasks and algorithms are iterative in nature, it is an important metric to compare different platforms, especially in the context of big data analytics

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)	★★★★

P2P has huge network communication overhead; MapReduce has disk I/O overhead

Reduces the disk I/O overhead

All the other platforms are suited for iterative processing. All the iterative algorithms cannot be easily modified for each of these platforms

Outline

- Introduction
- Scaling
- Horizontal Scaling Platforms
  - Hadoop
  - Peer to Peer
  - Spark
- Vertical Scaling Platforms
  - Graphical Processing Unit (GPU)
  - Multicore
  - High Performance Computing (HPC) clusters
  - Field Programmable Gate Array (FPGA)
- Comparison of Different Platforms
- Big Data Analytics on Amazon EC2 Clusters

K-Means and KNN Algorithms

K-MEANS CLUSTERING ALGORITHM

Basic K-Means Clustering Algorithm

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
```

- Starts by initializing the cluster centroids
- Each data point is associated with the nearest centroid in the step 2
- In Step 3, centroids are recalculated
- Step 2 and Step 3 are repeated until the centroids converge or till predefined number of iterations

K-Means clustering on Different Platforms

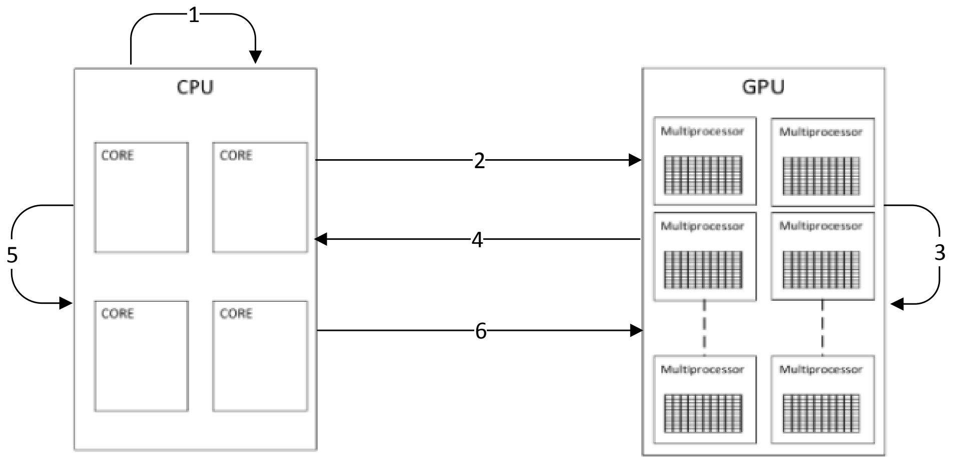
- Popular and widely used algorithm
- Contains critical elements that can demonstrate the ability of various platforms
- Characteristics include:
  - Iterative nature of the algorithm wherein the current iteration results are needed before proceeding to the next iteration
  - Compute-intensive task of calculating the centroids from a set of data points
  - Aggregation of the local results to obtain a global solution when the algorithm is parallelized

K-Means GPU Pseudo code

- **Input**: Data points D, Number of clusters K
- **Step** 1: Initialize K centroids randomly on CPU
- **Step** 2: Send K and D to GPU. Copy K in GPU shared memory of
  Multiprocessors.Divide and copy data points into memory of
  each processor.
- **Step** 3: Kernel process:

  Each processor computes the distance of each point in D;
  at a time with each centroid in K. Associate each data
  point with it's nearest centroid.
- **Step** 4: Send the association back to CPU
- **Step** 5: Re-compute the centroids with the given association on CPU
- **Step** 6: Send new K to GPU
- **Step** 7: Repeat Step 3 to Step 6 until global converge
- **Output**: Data points with cluster membership

K-Means CPU-GPU communication



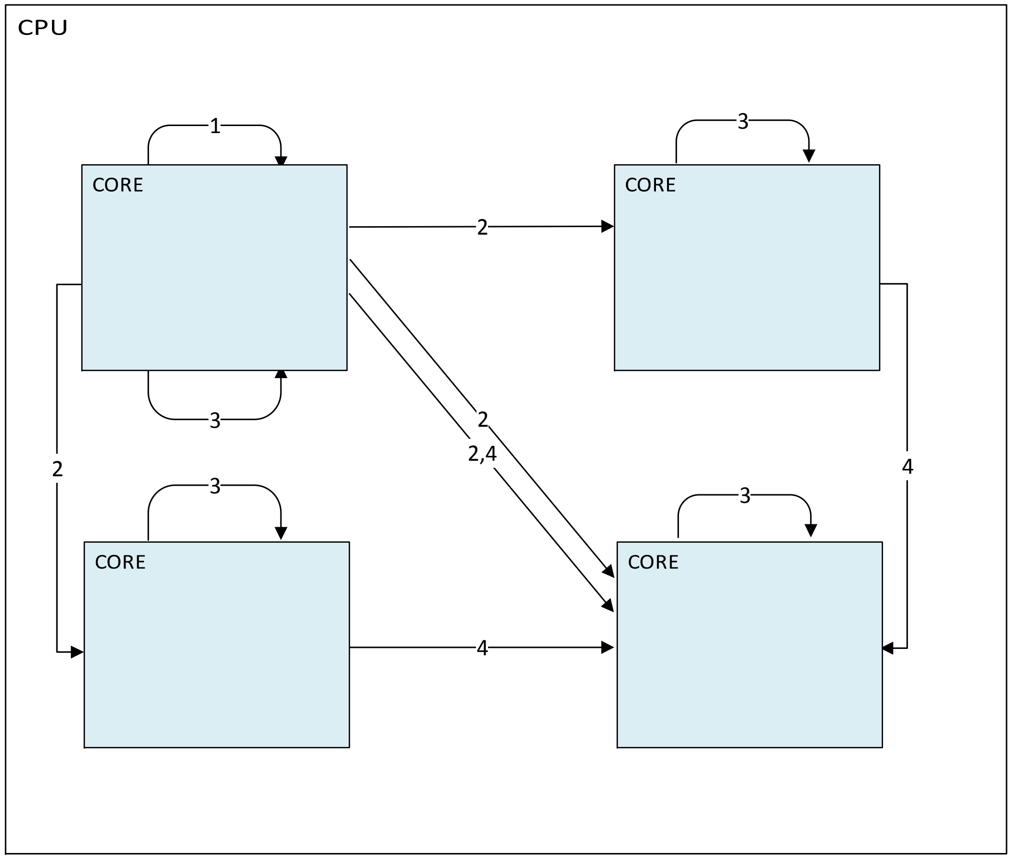
Step [illegible]: [illegible]
of M [illegible] processors. Divide and copy data points into
memory at each processor.

Each processor compute the distance of each point in D; with each centroid in K. Associate each data point with it's nearest centroid.

K-Means Multicore Pseudo code

- **Input**: Data points D, Number of clusters k
- **Step** 1: Initialize k centroids randomly
- **Step** 2: Send the centroids and the split the D into multiple cores
- **Step** 3: Associate each data point in D with the nearest centroid. This will divide the data points into k clusters.
- **Step** 4: Recalculate the position of centroids.
- **Step** 5: Repeat steps 2 to 4 until there are no more changes in the membership of the data points
- **Output**: Data points with cluster memberships

K-Means Multicore Communication



Step [illegible]: [illegible] nearest
multiple [illegible] data points [illegible]
clusters.

K-Means Mapreduce Pseudo code

- **Input**: Data points D, Number of clusters K
- **Step** 1: Copy K and D into memory. Initialize each centroid with 0 as data points.
- **Step** 2: Mapper:

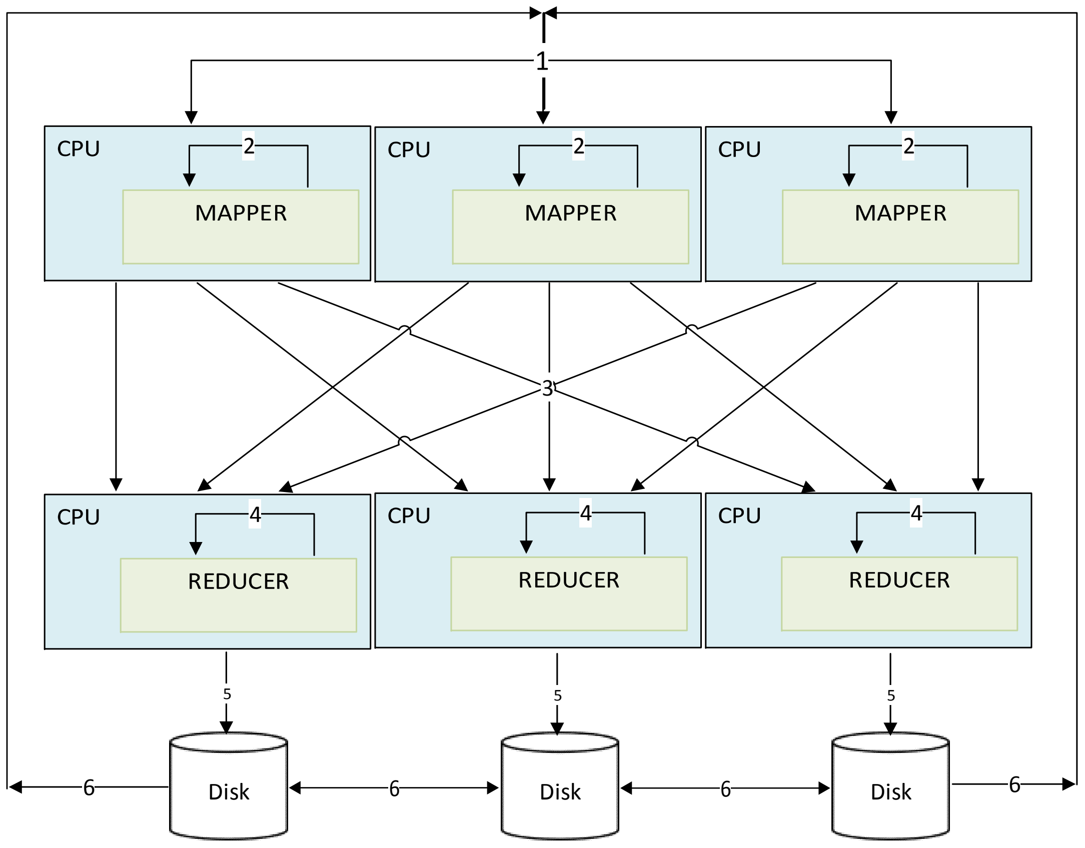
  Each map task computes the distance of each point with the centroid array. Assign data points to its nearest centroid.
- **Step** 3: Mapper Output:

  Output the key-value pair with key as centroid and value as the data points array.
- **Step** 4: Reducer:

  Combine all the values for each key (centroid) and compute the new centroid.
- **Step** 5: Reducer Output:

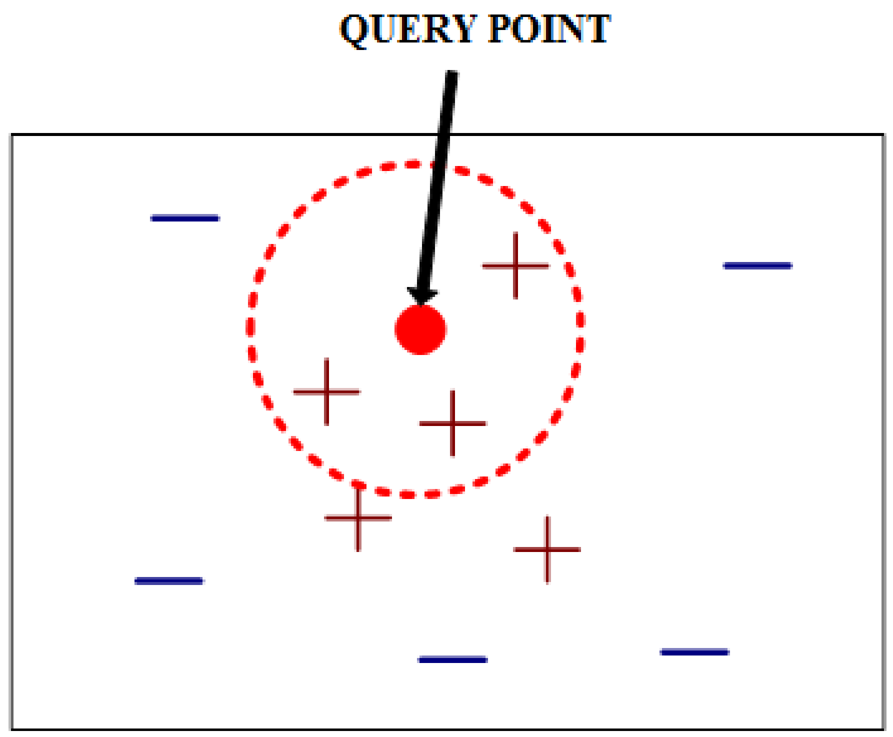
  Write the new centroids.
- **Step** 6: Repeat steps 1 to 5 until the centroid converges
- **Step** 7: Repeat steps 1 to 3 and write the mapper output.
- **Output**: Data points with cluster membership.

K-Means Mapper-Reducer Communication



Step 6: Repeat step 1 to 5 until the centroid converges. [illegible] with 0 as data p[illegible] [illegible]th

the [illegible] points to its nearest centroid.



K-NEAREST NEIGHBOR ALGORITHM

KNN GPU Pseudo code

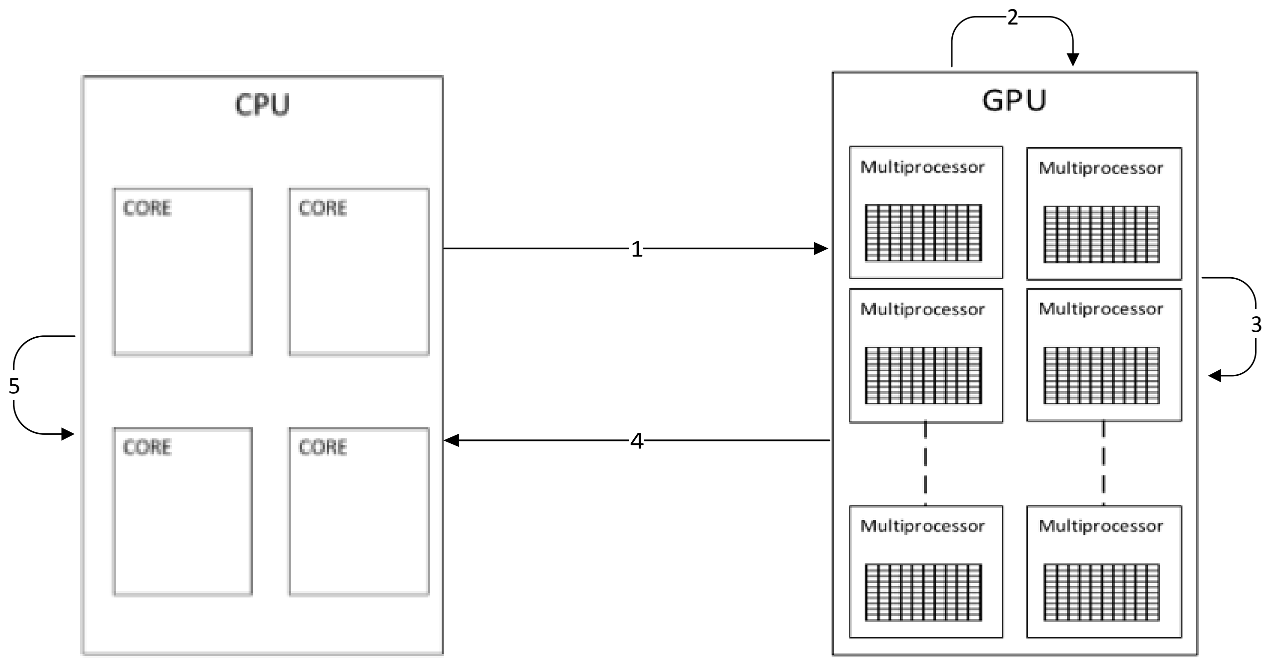
- **Input**: Reference points R and Query points Q
- **Step** 1: Send R and Q to GPU. Copy Q in the shared memory of the multiprocessors.
- **Step** 2: **Kernel** 1:

  Compute the distance matrix m × n where m are the query points and n are the points. Each processor computes the distance of one query point and with R.
- **Step** 3: **Kernel** 1:

  Send m × n matrix to kernel 2
- **Step** 3: **Kernel** 2:

  Each row in m × n matrix is provided to individual processor and is sorted parallelly using insertion sort. Along with it, the indices of each column are also sorted.
- **Step** 4: Send this m × n matrix back to CPU
- **Step** 5: Compute the K nearest neighbors
- **Output**: Query points with sorted distance to each reference point.

KNN CPU-GPU Communication



Step 2: **Kernel 1**: Copy R and Q in the shared memory of the multiprocessors. [illegible] computes the distance matrix [illegible] individual processors and [illegible] in parallel. Each processor computes the distances of each query point with R.

Step 3: **Kernel 2**: [illegible] query points and is sorted [illegible] using insertion sort. Along with it, the indices of each column are also sorted.

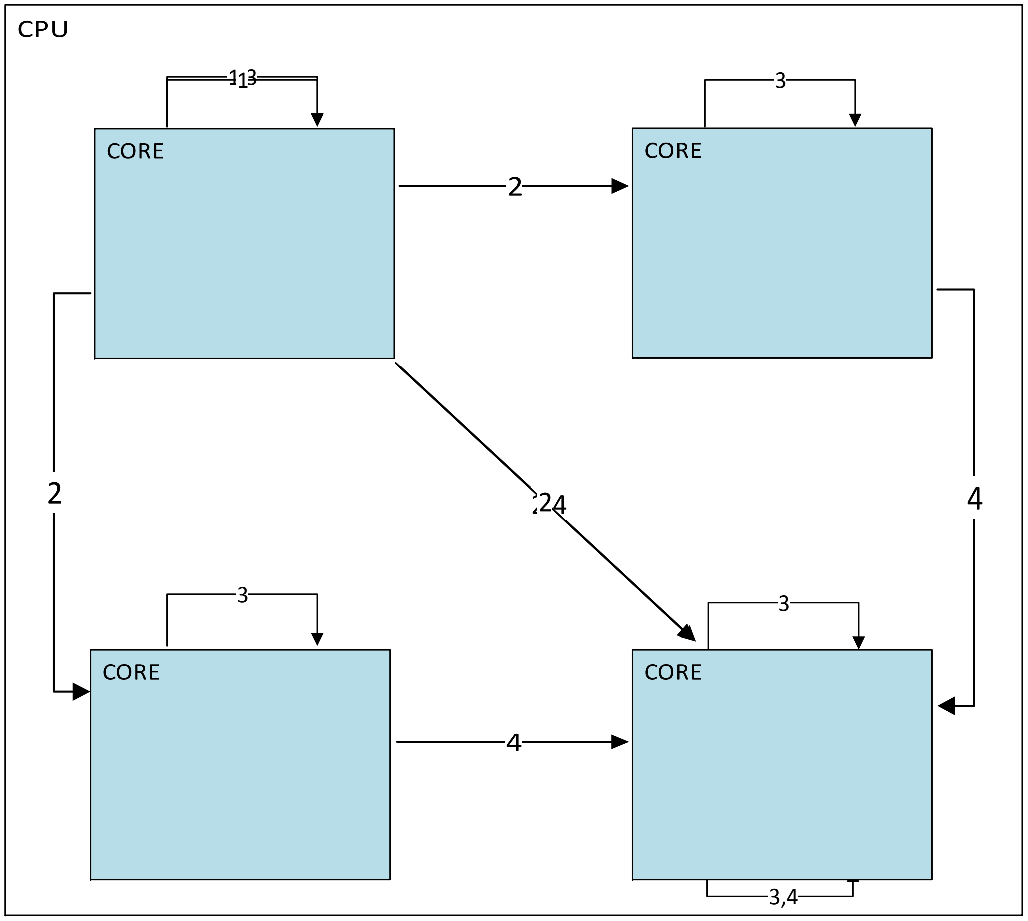
Step 4: Send k × m matrix back to CPU

Step 5: Compute the K nearest neighbors

KNN Multicore Pseudo code

- **Input**: Reference points R, Query points Q
- **Step** 1: Compute the distance of each point in Q with points in R.
- **Step** 2: Create a Distance matrix m × n where m are the query points and n are the reference points.
- **Step** 3: Sort each row in the matrix
- **Step** 4: Select the K nearest neighbors for each query point
- **Output**: Query point with K nearest neighbors

KNN CPU-Cores Communication



Step 4: Select the K nearest neighbors for each query point

points and Rn are the distance points.

KNN MapReduce Pseudo code

- **Input**: Reference points R, Query points Q and Nearest neighbor K
- **Step** 1: Copy Q and R into memory
- **Step** 2: Mapper:

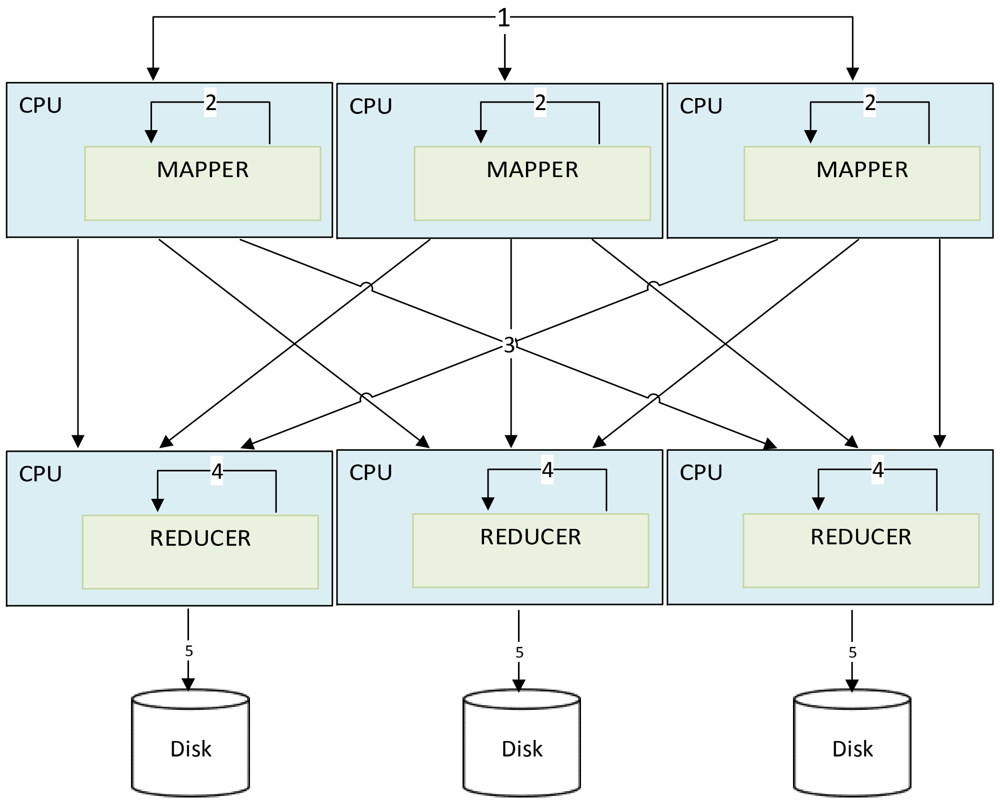
  Each map task computes the distance of each point in Q with each point in R
- **Step** 3: Mapper Output:

  Output the key-value pair with key as data point in Q and value vector of distances. The value is combination of reference point label and distance
- **Step** 4: Reducer:

  For each key, sort the value vector and find the K elements with smallest distance
- **Step** 5: Reducer Output:

  Write the associations to disk

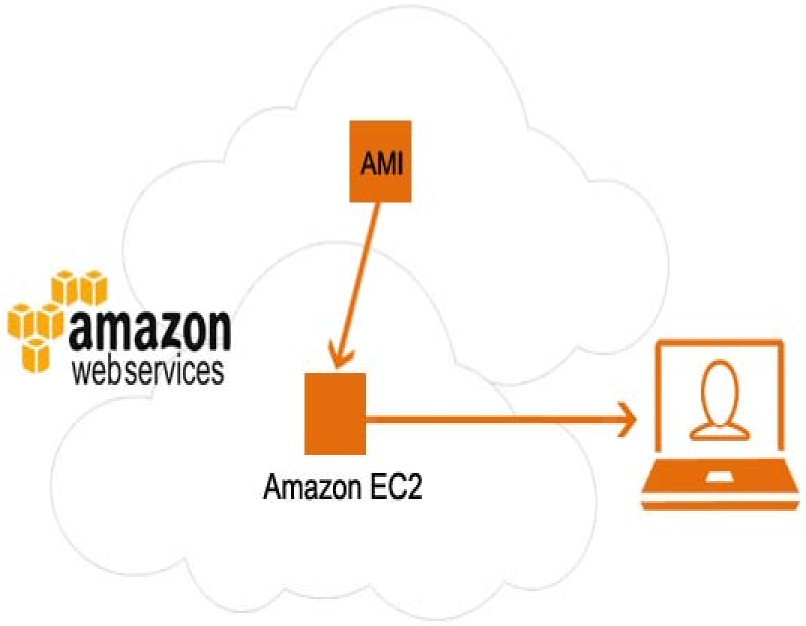
KNN Mapper-Reducer Communication



Step [illegible]: [illegible] into memory.

[illegible] [illegible]ces. The value is combination of reference point label and distance

Amazon Web Services



Amazon EC2

- Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud.
- Designed to make web-scale computing easier for developers.
- Simple web service interface allows you to obtain and configure capacity with minimal friction.
- Provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment.
- Reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change.
- Changes the economics of computing by allowing you to pay only for capacity that you actually use.
- Provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

Benefits

- **Elastic Web-Scale Computing**
  - Enables you to increase or decrease capacity within minutes.
  - You can commission thousands of server instances simultaneously.
  - Applications can automatically scale itself up and down depending on its needs.
- **Completely Controlled**
  - You have root access to each instance
  - You can stop your instance while retaining the data.
  - Instances can be rebooted remotely using web service APIs.
  - You also have access to console output of your instances.
- **Flexible Cloud Hosting Services**
  - You have the choice of multiple instance types, operating systems, and software packages.
  - It allows you to select a configuration of memory, CPU, instance storage, and the boot partition size that is optimal for your choice of operating system and application.
- **Reliable**
  - The service runs within Amazon's proven network infrastructure and data centers.
  - The Amazon EC2 Service Level Agreement commitment is 99.95% availability for each Amazon EC2 Region.

Benefits

- **Secure**
  - Amazon EC2 works in conjunction with Amazon VPC to provide security and robust networking functionality
  - Instances are located in a Virtual Private Cloud (VPC) with an IP range that you specify.
  - You decide which instances are exposed to the Internet and which remain private.
  - Security Groups and networks ACLs allow you to control inbound and outbound network access.
  - You can provision your EC2 resources as Dedicated Instances. Dedicated Instances are Amazon EC2 Instances that run on hardware dedicated to a single customer for additional isolation.
- **Inexpensive**
  - Pay only for what is used, without up-front or long-term commitments
  - **On-Demand Instances** let you pay for compute capacity by the hour with no long-term commitments.
  - **Reserved Instances** give you the option to make a low, one-time payment for each instance and in turn receive a significant discount on the hourly charge for that instance.
  - **Spot Instances** allow customers to bid on unused Amazon EC2 capacity and run those instances for as long as their bid exceeds the current Spot Price.
- **Easy to Start**
  - Choosing preconfigured software on Amazon Machine Images (AMIs), you can quickly deploy softwares to EC2 via 1-Click launch or with the EC2 console.

Using EC2 Services

- Instances are priced depending upon the configurations

Instance Type	Usage	Use cases	Price range
T2	General Purpose	Development environments, build servers, code repositories, low-traffic web applications, early product experiments, small databases.	$0.013 - $0.520 per hour
M3	General Purpose	Small and mid-size databases, backend servers for SAP, Microsoft SharePoint	$0.070 - $0.560 per hour
C3	Compute Optimized	High performance front-end fleets, web-servers, on-demand batch processing, distributed analytics, high performance science and engineering applications, ad serving, batch processing and distributed analytics.	$0.105 - $1.680 per hour
R3	Memory Optimized	High performance databases, distributed memory caches, in-memory analytics, genome assembly and analysis, larger deployments of SAP, Microsoft SharePoint, and other enterprise applications.	$0.175 - $2.800 per hour
G2	GPU	Game streaming, video encoding, 3D application streaming, GPGPU, and other server-side graphics workloads.	$0.650 per hour
I2	Storage Optimized	NoSQL databases, scale out transactional databases, data warehousing and cluster file systems.	[illegible]

EC2 Best Practices

- Make sure you choose the correct instance type, depending upon your use case
- Make sure you choose the correct OS and the appropriate amount of storage for your use case
- For Development purposes, choose the configurations which are provided for free. AWS provides a free tier for 750 hours each month for some configurations
- While using On-demand instances, make sure to stop the instances if not in use and restart them later as required
- Terminate the instances which you won’t be needing anymore to avoid being charged.

Big Data Platform Instance Configurations for AWS

AWS GPU Instance

- Type : g2.2xlarge (GPU)
- Instances used: 1
- Processor: Intel Xeon-E5-2670 (Sandy Bridge)
- CPU cores: 8
- Memory : 15 Gb
- GPU: NVIDIA with 1,536 CUDA cores and 4 Gb of video memory
- Instance storage: 60Gb SSD

AWS Multicore Instance

- Type : c3.2xlarge (Compute optimized)
- Instances used: 1
- Processor: Intel Xeon E5-2680 v2 (Ivy Bridge)
- CPU cores: 8
- Memory: 15 Gb
- Instance storage: 2 x 80Gb SSD

AWS Hadoop Instances

- Type : c3.xlarge (Compute Optimized)
- Instances used: 3
- Processor: Intel Xeon E5-2680 v2 (Ivy Bridge)
- CPU cores: 4
- Memory: 7.5 Gb
- Instance storage: 2 x 40Gb SSD

AWS Spark Instance

- Type : m3.large
- Instances used: 4
- Processor: Intel Xeon E5-2670 v2 (Ivy Bridge)
- CPU cores: 2
- Memory : 7.5 Gb
- Instance storage: 1 x 32Gb SSD

LAB SESSION

Conclusion

- Big Data is not about handling a particular challenge, rather it’s a field in itself.
- Big data can provide potentially revolutionary solutions to the problems where there are no answers yet.
- It can directly impact various disciplines especially in the way the data is currently being handled in those disciplines.
- Different platforms have different strengths and the choice of platforms can play a critical role in the eventual success of the application and/or algorithm used.
- Algorithms for Big Data Analytics are still at their infancy.

Thank You

Questions and Comments

Feel free to email questions or suggestions to

reddy@cs.wayne.edu

http://www.cs.wayne.edu/~reddy/