Module-4

Data-Intensive Computing

Map Reduce Programming

What is data-intensive computing?

- Data-intensive computing is concerned with production, manipulation, and analysis of large-scale data in the range of hundreds of megabytes (MB) to petabytes(PB) and beyond.
- The term dataset is commonly used to identify a collection of information elements that is relevant to one or more applications
- Datasets are often maintained in repositories, which are infrastructures supporting the storage, retrieval, and indexing of large amounts of information
- To help search ,relevant bits of information, we use metadata that is attached to datasets

Application Areas of Data Intensive Computing

- **Computational science** is one of the most popular ones.
- **Scientific simulations** and experiments are often keen to produce, analyze, and process huge volumes of data. Hundreds of gigabytes of data are produced every second by telescopes mapping the sky; the collection of images of the sky easily reaches the scale of petabytes over a year.
- **Bioinformatics applications** mine databases that may end up containing terabytes of data.
- **Earthquake simulators** process a massive amount of data, which is produced as a result of recording the vibrations of the Earth across the entire globe

Characterizing data-intensive computations

Data-intensive applications handle datasets on the scale of multiple terabytes and petabytes. Datasets are commonly persisted in several formats and distributed across different locations.

Challenges ahead

Open challenges in data-intensive computing given by Ian Gorton are:

- Scalable algorithms that can search and process massive datasets

- New metadata management technologies that can scale to handle complex, heterogeneous, and distributed data sources
- Advances in high-performance computing platforms aimed at providing a better support for accessing in-memory multi terabyte data structures
- High-performance, highly reliable, peta scale distributed file systems
- Data signature-generation techniques for data reduction and rapid processing
- New approaches to software mobility for delivering algorithms that are able to move the computation to where the data are located
- Specialized hybrid interconnection architectures that provide better support for filtering multi gigabyte data streams coming from high-speed networks and scientific instruments
- Flexible and high-performance software integration techniques that facilitate the combination of software modules running on different platforms to quickly form analytical pipelines

Historical perspective

Various stages in which data intensive computing developed is given below:

- The early age: high-speed wide-area networking
- Data grids
- Data clouds and "Big Data"

The early age: high-speed wide-area networking

- The evolution of **technologies, protocols, and algorithms** for data transmission and streaming has been an enabler of data-intensive computations
- In **1989, the first experiments** in high-speed networking as a support **for remote visualization** of scientific data led the way
- **In 1991 at Supercomputing conference practically remote visualization was implemented** by large developing a high-resolution magnetic resonance image, or MRI, scan of the human brain and was sent between the Pittsburgh Supercomputing Center (PSC) and Albuquerque, New Mexico, where the conference was held
- The project called Wide Area Large Data Object (WALDO) system was

developed, which was used to provide the following capabilities:

- automatic generation of metadata; automatic cataloguing of data and metadata
- facilitation of cooperative research by providing local and remote users access to data
- mechanisms to incorporate data into databases and other documents

- Another important milestone was set with the Clipper project, A collaborative effort of several scientific research laboratories, with the goal of designing and implementing a collection of independent but architecturally consistent service components

Data grids

With the advent of grid computing, huge computational power and storage facilities could be obtained by through heterogeneous resources across different domains. Within this context, data grids become important.

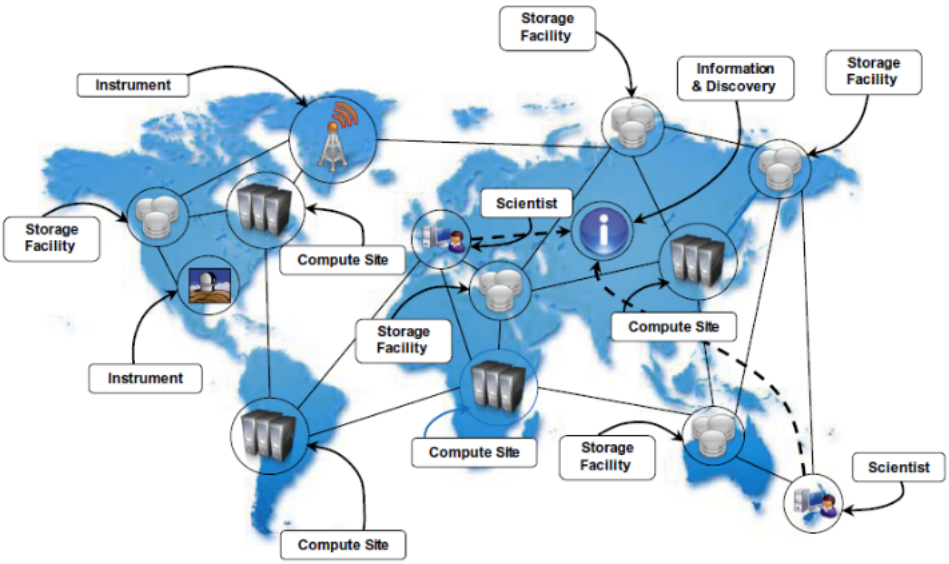
The components/players involved are:

- Instruments
- Repositories (Storage Facility)
- Information & Discovery
- Computing sites
- Scientists

Steps in Data Grid computing:

- Huge amounts of data are produced by **scientific instruments** (telescopes, particle accelerators, etc.).
- Next the information, which can be locally processed, is then stored in repositories and made available for experiments and analysis to scientists.
- Scientists can leverage specific **discovery and information services**, which help in determining the locations of the closest datasets of interest for their experiments.
- Datasets are replicated by the infrastructure to provide better availability in **storage facility**.
- Since processing of this information also requires a large computational power, specific **computing sites** are used to perform analysis and

experiments.



Challenges faced in implementing data grids

- Massive data sets: The size of data sets can easily be on the scale of gigabytes, terabytes, and beyond. It is therefore necessary to minimize latencies during bulk transfers and replicate content
- Shared data collections: Resource sharing includes distributed collections of data.

  For example, repositories can be used to both store and read data.
- Unified namespace: Data grids impose a unified logical name space where to locate data collections and resources.
- Access restrictions: Some users might want to ensure confidentiality for their data and restrict access to them to their collaborators. Authentication and authorization in data grids involve both coarse-grained and fine-grained access control over shared data collections

Data clouds and "Big Data"

- Earlier Large datasets have mostly been used only for scientific computing.
- This scenario has recently started to change as massive amounts of data are being produced, mined, and crunched by companies that provide Internet services such as searching, online advertising, and social media.

- It is critical for such companies to efficiently analyze these huge datasets because they constitute a precious source of information about their customers.
- They use Log analysis is an example of a data-intensive operation. Companies such as Google have a massive amount of data in the form of logs that are daily processed using their distributed infrastructure.

Support of Cloud to Data intensive Applications

Cloud technologies support data-intensive computing in several ways:

- By providing a large amount of compute instances on demand, which can be used to process and analyze large datasets in parallel.
- By providing a storage system optimized for keeping large blobs of data and other distributed data store architectures.
- By providing frameworks and programming APIs optimized for the processing and management of large amounts of data.

Technologies for data-intensive computing

- Storage systems
- Programming Platforms

Initially we concentrate on Storage Systems supporting Data Intensive Computing:

Various Storage systems can be classified into two categories:

1. High-performance distributed file systems and storage clouds
2. NO-SQL or Not a SQL systems

High-performance distributed file systems and storage clouds

Various Distributed file systems are discussed one by one here:

Lustre:

- The Lustre file system is a massively parallel distributed file system that covers the needs of a small workgroup of clusters to a large-scale computing cluster.
- The file system is used by several of the Top 500 supercomputing systems,

including the one rated the most powerful supercomputer in the June2012 list

- It is designed to provide access to peta bytes (PBs) of storage to serve thousands of clients with an I/O throughput of hundreds of gigabytes persecond (GB/s).

Google File System(GFS):

- GFS is the storage infrastructure that supports the execution of distributed applications in Google's computing cloud.
- The system has been designed to be a fault- tolerant, highly available, distributed file system built on commodity hardware and standard Linux operating systems.

Various assumptions made while designing GFS are:

- The system is built on top of commodity hardware that often fails.
- The system stores a modest number of large files; multi-GB files are common
- The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.
- The workloads also have many large, sequential writes that append data to files.
- High-sustained bandwidth is more important than low latency

IBM General Parallel File System(GPFS):

- GPFS is the high-performance distributed file system developed by IBM that provides support for the RS/6000 supercomputer and Linux computing clusters.
- GPFS is a multi platform distributed file system built over several years of academic research and provides advanced recovery mechanisms.

- GPFS is built on the concept of shared disks, in which a collection of disks is attached to the file system nodes by means of some switching fabric.
- The file system makes this infrastructure transparent to users and stripes large files over the disk array by replicating portions of the file to ensure high availability.
- By means of this infrastructure, the system is able to support petabytes of storage, which is accessed at a high throughput and without losing consistency of data

Amazon Simple Storage Service(S3):

- Amazon S3 is the online storage service provided by Amazon.
- Even though its internal details are not revealed, the system is claimed to support high availability, reliability, scalability, infinite storage, and low latency at commodity cost.
- The system offers a flat storage space organized into buckets, which are attached to an Amazon Web Services (AWS) account.
- Each bucket can store multiple objects, each identified by a unique key. Objects are identified by unique URLs and exposed through HTTP, thus allowing very simple get-put semantics

Limitation of S3:

- Security is limitation of S3 as the visibility and accessibility of objects are linked to AWS accounts, and the owner of a bucket can decide to make it visible to other accounts or the public.
- It is also possible to define authenticated URLs, which provide public access to anyone for a limited (and configurable) period of time

No SQL systems

- The term Not Only SQL (No SQL) was originally coined in 1998 to identify a relational database that did not expose a SQL interface to manipulate and query data
- Instead used a set of UNIX shell scripts and commands to operate on text files containing the actual data.

- In 2009,the term No SQL was reintroduced with the intent of labelling all those database management systems that didnot use a relational model but provided simpler and faster alternatives for data manipulation.
- Now a days, the term NoSQL is a big umbrella encompassing all the storage and database management systems that differ in some way from the relational model.

  According to Wikipedia various types of NoSQL can be identified here is the list:

  - Document stores (Apache CouchDB, SimpleDB,Terrastore).
  - Graphs (AllegroGraph, Neo4j, FlockDB,Cerebrum).
  - Key-value stores
  - Multivalue databases (OpenQM,RocketU2,OpenInsight).
  - Object databases (ObjectStore,JADE,ZODB).
  - Tabularstores (GoogleBigTable,HadoopHBase,Hypertable).
  - Tuple stores (ApacheRiver).

Here is Few NOSQL systems discussed:

ApacheCouchDBandMongoDB:

- They are the two examples of document stores.
- Both provide a schema-less store they are organized into a collection of key-value fields.
- The value of each field can be of type string, integer,float, date,or anarrayof values.
- The databases expose a RESTful interface and represent data in JSON format.
- Both allow querying and indexing data by using the Map Reduce programming model,
- Use JavaScript as a base language for data querying and manipulation rather than SQL.

- o CouchDB ensures ACID- Atomicity Consistency Integrity & Durability properties on data.
- o MongoDB supports sharding, it is the ability to distribute the conten t of a collection among different nodes

Amazon Dynamo:

- o It is the distributed key-value store that supports the management of information of several of the business services offered by Amazon.

- o The architecture of the Dynamo system, shown in Figure 8.3, is composed of a collection of storage peers organized in a ring that shares the key space for a given application.
- o The key space is partitioned among the storage peers, and the keys are replicated across the ring, avoiding adjacent peers.
- o Each peer is configured with access to a local storage facility where original objects and replicas are stored.
- o Furthermore, each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.
- o Dynamo implements the capability of being an always-writablestore

Limitation of dynamo

- o The downside of such an approach is the *simplicity* of the storage model, which requires applications to build their own data models on top of the simple building blocks provided by the store.
- o For example, No support for referential integrity constraints, relationships are embedded in the storage model, and therefore join operations are not supported

The Diagram of Amazon Dynamo

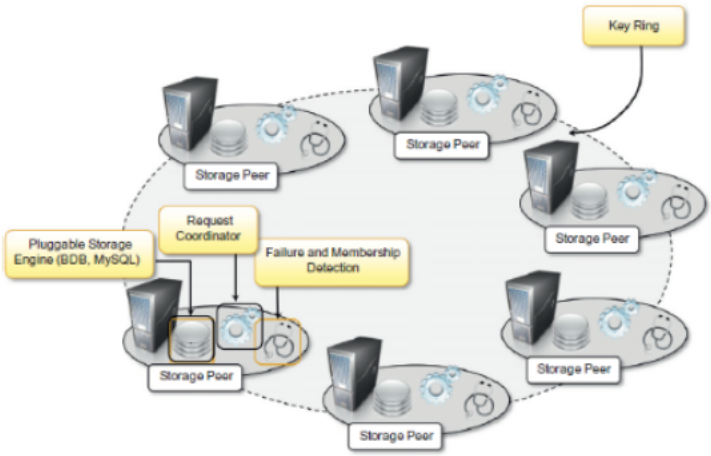


FIGURE 8.3
Amazon Dynamo architecture.

GoogleBigtable.

- Bigtable is the distributed storage system designed to scale up to peta- bytes of data across thousands of servers.
- Bigtable organizes the data storage in tables of which the rows are distributed over the distributed file system supporting the middleware, which is the Google File System.
- From a logical point of view, a table is a multidimensional sorted map indexed by a key that is represented by a string of arbitrary length.
- Here A table is organized into rows and columns; columns can be grouped in column family

Working of Google Big Table

- Figure 8.4 gives an overview of the infrastructure that enables Bigtable.
- Bigtable identifies two kinds of processes: master processes and tablet server processes.
- A tablet server is responsible for serving the requests for a given tablet that is a contiguous partition of rows of a table. Each server can

manage multiple tablets (commonly from 10 to 1,000).

- The master server is responsible for keeping track of the status of the tablet servers and of the allocation of tablets to tablet servers. The server constantly monitors the tablet servers to check whether they are alive, and find out they are reachable or not reachable

- Chubby is a distributed, highly available, and persistent lock service—supports the activity of the master and tablet servers

- At the very bottom layer, the data are stored in the Google File System in the form of files, and all the update operations are logged into the file for the easy recovery of data in case of failures or when tablets need to be reassigned to other servers.

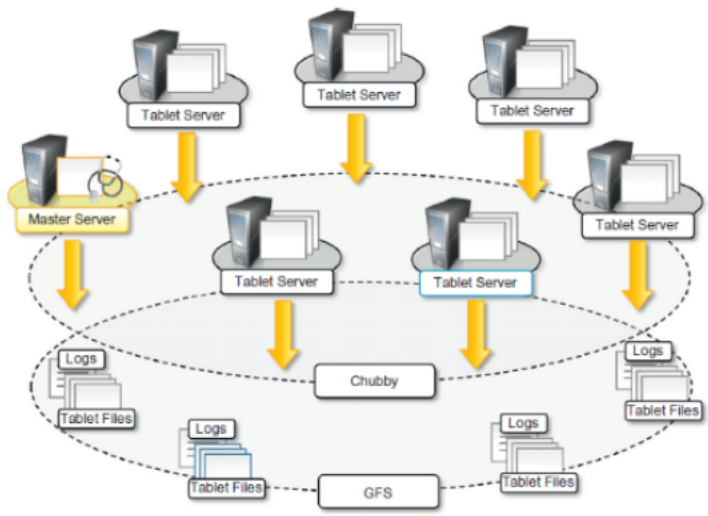


FIGURE 8.4
Bigtable architecture.

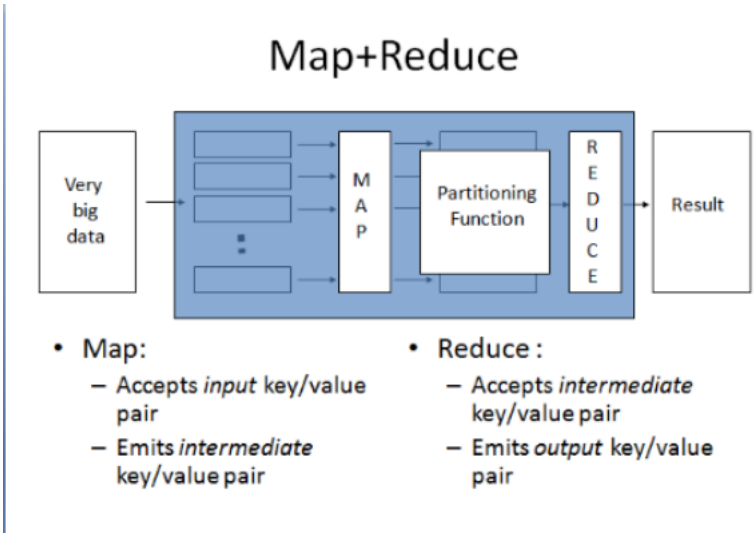
ApacheCassandra:

- It is a distributed object store for managing large amounts of structured data spread across many commodity servers.
- The system is designed to avoid a single point of failure and offer a highly reliable service.

- Cassandra was initially developed by Facebook; now it is part of the Apache incubator initiative.
- Currently, it provides storage support for several very large Web applications such as Facebook, Digg, andTwitter.
- It follows a fully distributed design of Amazon Dynamo, and Google Bigtable, from which it inherits the “column family” concept.
- The data model exposed by Cassandra is based on the concept of a table that is implemented as a distributed multidimensional map indexed by a key.
- In terms of the infrastructure, Cassandra is very similar to Dynamo. It has been designed for incremental scaling, and it organizes the collection of nodes sharing a key space into a ring

Programming platforms

Map Reduce Technique For Big Data in nutshell



Here is a pseudocode to count the different words from various documents using map reduce technique.

Diagrammatic representation of map-reduce and partition is given below:

Map-Reduce Diagrammatic representation

Parallel Execution