SNS COLLEGE OF TECHNOLOGY

COIMBATORE-35

DEPARTMENT OF INFORMATION TECHNOLOGY

19ITE305 – BIG DATA ANALYTICS

UNIT II: INTRODUCTION TO TECHNOLOGY LANDSCAPE

Topic 3: RDBMS Versus Hadoop, Distributed Computing Challenges

RDMS (Relational Database Management System): RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties required for designing a database. The purpose of RDBMS is to store, manage, and retrieve data as quickly and reliably as possible.

Hadoop: It is an open-source software framework used for storing data and running applications on a group of commodity hardware. It has large storage capacity and high processing power. It can manage multiple concurrent processes at the same time. It is used in predictive analysis, data mining and machine learning. It can handle both structured and unstructured form of data. It is more flexible in storing, processing, and managing data than traditional RDBMS. Unlike traditional systems, Hadoop enables multiple analytical processes on the same data at the same time. It supports scalability very flexibly. Below is a table of differences between RDBMS and Hadoop:

S.No.	RDBMS	Hadoop
1.	Traditional row-column based databases, basically used for data storage, manipulation and retrieval.	An open-source software used for storing data and running applications or processes concurrently.
2.	In this structured data is mostly processed.	In this both structured and unstructured data is processed.

S.No.	RDBMS	Hadoop
3.	It is best suited for OLTP environment.	It is best suited for BIG data.
4.	It is less scalable than Hadoop.	It is highly scalable.
5.	Data normalization is required in RDBMS.	Data normalization is not required in Hadoop.
6.	It stores transformed and aggregated data.	It stores huge volume of data.
7.	It has no latency in response.	It has some latency in response.
8.	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
9.	High data integrity available.	Low data integrity available than RDBMS.
10.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

Distributed Computing Challenges

What is a Distributed System?

A distributed system is a system that utilizes multiple networked computers which together work toward a common purpose/goal. Computers on distributed systems are able to communicate and synchronize their actions through a communication network by passing each other "messages". Additionally, different computers may serve different specific functions by hosting different components - these different computers have a separate memory and run on their own operating systems. Additionally, redundancies are always implemented to ensure that if one systems component fails, the entire system does not.

Distributive systems can be found in various telecommunications networks and network applications. Such as: telephone or cellular networks and peer-to-peer networks or massive multiplayer games.

Note: Distributed computing studies distributed systems. A program that runs on a distributed system is known as a distributed program.

Benefits

- Resiliency - With multiple computers, redundancies are implemented to ensure that a single failure doesn't equate to systems-wide failure
- Resource/Data sharing - Resources are available to multiple users
- Speed - Multiple computers are faster than one (usually)
- Scalability - Easier to scale

Distributed Systems: Challenges, Failures

Challenges and Failures of a Distributed System are:

- Heterogeneity
- Scalability
- Openness
- Transparency
- Concurrency
- Security
- Failure Handling

Heterogeneity

Heterogeneity refers to the differences that arise in networks, programming languages, hardware, operating systems and differences in software implementation. For example, there are different hardware devices, tablets, mobile phones, computers, and etc.

Some challenges may present themselves due to heterogeneity. When programs are written in different languages or developers utilize different implementations (data structures, etc.) problems will arise when the computers try to communicate with each other. Thus it is important to have common standards agreed upon and adopted to streamline the process. Additionally, when we consider mobile code - code that can be transferred from one computer to the next - we may encounter some problems if the executables are not specified to accommodate both computers' instructions and specifications.

Scalability

A program is scalable if a program does not need to be redesigned to ensure stability and consistent performance as its workload increases. As such, a program (distributed system in our case) should not have a change in performance regardless of whether it has 10 nodes or 100 nodes.

As a distributed system is scaled, several factors need to be taken into account: size, geography, and administration. The associated problem with size is overloading. Overloading refers to the degradation of the system that refers as the system's workload increases (increase in number of users, resourced consumed, etc.). Secondly, with geography, as the distance that our distributed system encompasses increases, the reliability of our communication may break down. Additionally, as a distributed system is scaled, we may have to implement controls in the system; however, this may devolve into what we can effectively call an administrative mess.

Openness

The openness of distributed systems refers to the system's extensability and ability to be reimplemented. More specifically, the openness of a distributed system can be measured by three characteristics: interoperability, portability, and extensability as we previously mentioned. Interoperability refers to the system's ability to effectively interchange information between computers by standardization, portability refers to the system's ability to properly function on different operating systems, and extensability allows developers to freely add new features or easily reimplement existing ones without impairing functionality. Additionally, open distributed systems implement open interfaces, which comes with many challenges - in this case having well-defined interfaces may present themselves as challenges.

Transparency

A problem with transparency may arise with distributed systems due to the nature of the system's complexity. In this context, transparency refers to the distributed system's ability to conceal its complexity and give off the apperance of a single system. And when we discuss transparency, we must also discuss to what extent.

Concurrency

This discusses the shared access of resources which must be made available to the correct processes. Problems may arise when multiple processes attempt to access the same resources at the same time, thus steps need to be taken to ensure that any manipulation in the system remains in a stable state; however the illusion of simultaneous execution should be preserved. We refer to these preventative measure as concurrency control. Concurrency control should be implemented to ensure that processes are executed in a synchronous manner.

Security

Security is comprised of three key components: availability, integrity, and confidentiality. In a similar fashion, authentication and authorization are paramount - an entity must be verifiably authenticated as claimed and privileges must be appropriate delegated based on authority.
These concepts are related, availability regards the authenticated and authorized users, integrity protects through encryption and other methods, and confidentiality ensures that resources are not needlessly disclosed or made available.
Security is especially important in distributive systems due to their association with sensitive and private data. Take payment and transactions information, for example.

Failure Handling

Failures, like in any program, are a major problem. However, in distributive systems, with so many processes and users, the consequences of failures are exacerbated. Additionally, many problems will arise due to the nature of distributive systems. Unexpected edge cases may present themselves in which the system is ill-equipped for, but developers must account for. Failures can occur in the software, hardware, and in the network; additionally the failure can be partial, causing some components to function and other to not. However, the most important part in failure handling is recognizing that not every failure can be accounted for. Thus, implementing processes to detect, monitor, and repair systems failures is a core feature in failure handling/ management.