# Part A Question bank.

# 1. Describe IAAS, PAAS, SAAS. What are the main differences between them.

IAAS(infrastructure as a service) – deliver of compter infrastructure as a service. Network operators use it (control over applications, data, runtime, middleware, operation system)

PAAS(platform as a service) – delivery of computer platform for custom software development as a service (control over applications and data) app developers

SAAS(software as a service) apps through browser end users use it (everything is automatically controlled)

# 2. What is Hadoop and what is the main purpose of it. (lecture 6)

Hadoop is defined as a software utility that uses a network of many computers to solve problems involving a huge amount of computation and data, these data can be structured or unstructured, and hence it provides more flexibility for collecting, processing, analysing, and managing data. It has an open-source distributed framework for the distributed storage, managing, and processing of the big data application in scalable clusters of computer servers.

# 3. Describe the Monoliths and Microservices and state their main differences.

Advantages of microservices over monoliths include

– Support for scaling

• Scale vertically rather than horizontally

• Support for change – Support hot deployment of updates

• Support for reuse – Use same web service in multiple apps – Swap out internally developed web service for externally developed web service

• Support for separate team development – Pick boundaries that match team responsibilities

• Support for failure

# 4. What are the two fundamental aspects of Rest Design?

Resources: Every distinguishable entity is a resource. A resource may be a Web site, an HTML page, an XML document, a Web service, a physical device, etc.

• URLs Identify Resources: Every resource is uniquely identified by a URL.

# 5. What is hot swapping?

A hot swap describes the act of removing components from or plugging them into a computer system while the power remains switched on. This means that parts can be changed without shutting down or rebooting a computer or server.

# 6. Describe and give examples for parallel computing

In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem.

– To be run using multiple CPUs

– A problem is broken into discrete parts that can be solved in parallel – Each part is further broken down to a series of instructions

Examples:

Bitcoin - blockchain tech that uses multiple computers to validate transactions

Smartpohones – have 6core or more

# 7. Flynn’s Classical Taxonomy and describe Flynn’s matrix (lecture 3)

**Flynn's taxonomy** distinguishes multi-processor computer architectures according to how they can be classified along the two independent dimensions of Instruction and Data. Each of these dimensions can have only one of two possible states: Single or Multiple.

**Flynn’s matrix** has 4 possible classifications:

1. SISD - Single instruction, single data

2. SIMD - Single instruction, multiple data

3. MISD - Multiple instruction, single data

4. MIMD - Multiple instruction, multiple data

# 8. Memory Architectures and examples for each (lecture 3)

**Shared Memory:**

Multiple processors can operate independently but share the same memory resources.

Changes in a memory location effected by one processor are visible to all other processors.

Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.

**Distributed Memory:**

Distributed memory systems require a communication network to connect inter-processor memory.

Processors have their own local memory. Memory addresses in one processor do not map to another processor – no concept of global address space across all processors.

Each processor operates independently. Changes it makes to its local memory have no effect on the memory of other processors.

When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.

**Hybrid Distributed-Shared Memory:**

The largest and fastest computers in the world today employ both shared and distributed memory architectures.

The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global.

The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.

# 9. Transactions and their specialty in data recovery. (lecture 4)

A distributed transaction is a set of operations on data that is performed across two or more data repositories (especially databases). It is typically coordinated across separate nodes connected by a network but may also span multiple databases on a single server.

**Flat transaction** send out requests to different servers and each request is completed before client goes to the next one.

**Nested transaction** allows sub-transactions at the same level to execute concurrently.

# 10.Communication in two phase commit protocol (state all the required steps to start the communication and close the communication). (lecture 4)

**In the first phase**, each participant votes for the transaction to be committed or aborted. Once voted to commit, not allowed to abort it. So before votes to commit, it must ensure that it will eventually be able to carry out its part, even if it fails and is replaced.

A participant is said to be in a prepared state if it will eventually be able to commit it. So each participant needs to save the altered objects in the permanent storage device together with its status-prepared.

**In the second phase**, every participant in the transaction carries out the joint decision. If any one participant votes to abort, the decision must be to abort. If all the participants vote to commit, then the decision is to commit the transaction.

The problem is to ensure that all of the participants vote and that they all reach the same decision. It is an example of consensus. It is simple if no error occurs. However, it should work when servers fail, message lost or servers are temporarily unable to communicate with one another.

# 11.Concurrency Control in Distributed Transaction. (lecture 4)

Concurrency control for distributed transactions: each server applies local concurrency control to its own objects, which ensure transactions serializability locally.

However, the members of a collection of servers of distributed transactions are jointly responsible for ensuring that they are performed in a serially equivalent manner. Thus, global serializability is required.

# 12.Brewer’s Theorem. Give all possible combinations and state why they exist. (lecture 5)

In theory it is impossible to fulfill all 3 requirements.

CAP provides the basic requirements for a distributed system to follow 2 of the 3 requirements.

CA - Single site cluster, therefore all nodes are always in contact. When a partition occurs, the system blocks.

CP -Some data may not be accessible, but the rest is still consistent/accurate.

AP - System is still available under partitioning, but some of the data returned may be inaccurate.

# 13.Examples for NoSQL database and state advantages and challenges of it over RDBMS. (lecture 5)

**Advantages:**

“Classical” database administrators scale up – buy bigger servers as database load increases, Scaling out – distributing the database across multiple hosts as load increases

Volumes of data that are being stored have increased massively, Opens new dimensions that cannot be handled with RDBMS

Automatic repair, distribution, tuning vs. expensive, highly trained DBAs of RDBMS

Based on commodity servers, less costs per transaction/second

Non-existing/relaxed data schema, structural changes cause no overhead

**Challenges:**

Still in pre-production phase, Key features yet to be implemented

Mostly open source, result from start-ups, Limited resources or credibility

Require lot of skill to install and effort to maintain

Focused on web apps scenarios, Limited ad-hoc querying, Even a simple query requires significant programming expertise

Few number of NoSQL experts available in the market

# 14.State what is the limited Taxonomy of NoSQL and give example if you can. (lecture 5)

Examples:

Document Stores – MongoDB, OrientDB,RavenDB

Graph Databases – Neo4j, Infinite Graph, OrientDB, FlockDB

Key-Value Stores – Redis, MemcachedDB, HamsterDB

Columnar Databases- Google ’ s BigTablem HyperTable, Cassandra, SimpleDB

# 15.What is Row-Oriented Database and what is Column Oriented Database. Compare them and give examples of their use cases.

Row oriented databases(traditional databases) rows are stored in sequence which means rows immediately follow each another. All columns in a single row are stored together on the same page as log as the row size is smaller than page size.

Examples: PostgreSQL and MySQL databases

Column oriented databases are used when there are hundreds of rows and columns on table and if you make complex query the information you need will be in several blocks and to get a result you should go through whole database and composite index won’t work for us there. Columnar databases store data from one column together on disk. This means that all the names form one group while columns such as gender or age form another.

Examples: Amazon Redshift and BigQuery

|  |  |
| --- | --- |
| **Row-oriented** | **Column-oriented** |
| Inserting and deleting data is fast and easy | Inserting and deleting data could lead to a negative performance impact, especially if the table has many columns |
| The best solution for transactional processing (OLTP) applications | The best solution for analytical processing (OLAP) applications |
| Aggregating data is slow and inefficient | The best solution for aggregating data |
| Insufficient compression | High compression because of data similarity |
| Requires more space to store data | Requires less space to store data |

# 16.Read and Write operations in Hadoop. Provide step-by-step operations over performing these actions.

**Hadoop write operation step by step:**

1. HDFS client creates Distributed FileSystem

2. ClientNode creates NameNode

3. HDFS client writes FSDataOutputStream

4. FSDataOutputStream writes packet DataNode

5. DataNode acks packet FSDataOutputStream

6. HDFS client close FSDataOutputStream

7. ClientNode copletes NameNode

**Hadoop read operation step by step:**

1. HDF client opens Distributed FileSystem

2. Distributed FileSystem gets block location to NameNode

3. HDF client reads FSDataInputStream

4. FSDataInputStream reads DataNode

5. FSDataInputStream reads DataNode

6. HDF client close FSDatainputStream

# 17.HBase. Give a comparison of HBase and RDBMS. (lecture 5)

HBase is the Hadoop distributed, scalable, big data store database. Apache HBase is an open-source, distributed, versioned, non-relational database modeled after Google's Bigtable: A Distributed Storage System for Structured Data. Hbase is needed for random, real time read/write access to big data.

Hbase open new dimensions that can’t be handled with RDBMS. HBase is schemaless compared to rdbms which means Hbase gives more freedom and flexibility. For example, in Hbase we can easily change data storage through the process when we learn more about project

# 18.What is MapReduce and specify each phase to execute it.

Hadoop MapReduce’s programming model facilitates the processing of big data stored on HDFS.

By using the resources of multiple interconnected machines, **MapReduce effectively handles a large amount of structured and unstructured data**

At a high level, MapReduce breaks input data into fragments and distributes them across different machines.

The input fragments consist of key-value pairs. Parallel map tasks process the chunked data on machines in a cluster. The mapping output then serves as input for the reduce stage. The reduce task combines the result into a particular key-value pair output and writes the data to HDFS.

The Hadoop Distributed File System usually runs on the same set of machines as the MapReduce software. When the framework executes a job on the nodes that also store the data, the time to complete the tasks is reduced significantly.

# 19.Scheduling in Yarn. Describe and explain.

A scheduler typically handles the resource allocation of the jobs submitted to YARN. In simple words — for example - if a computer app/service wants to run and needs 1GB of RAM and 2 processors for normal operation - it is the job of YARN scheduler to allocate resources to this application in accordance to a defined policy. There are three types of schedulers available in YARN: FIFO, Capacity and Fair.

1. FIFO - runs the applications in submission order by placing them in a queue. Application submitted first, gets resources first and upon completion, the scheduler serves next application in the queue. However, FIFO is not suited for shared clusters as large applications will occupy all resources and queues will get longer due to lower serving rate.

# 20.State at features and components of HIVE

Apache Hive supports analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3 filesystem and Alluxio. It provides a SQL-like query language called HiveQL[9] with schema on read and transparently converts queries to MapReduce, Apache Tez[10] and Spark jobs. All three execution engines can run in Hadoop's resource negotiator, YARN (Yet Another Resource Negotiator). To accelerate queries, it provided indexes, but this feature was removed in version 3.0 [11] Other features of Hive include:

Different storage types such as plain text, RCFile, HBase, ORC, and others.

Metadata storage in a relational database management system, significantly reducing the time to perform semantic checks during query execution.

Operating on compressed data stored into the Hadoop ecosystem using algorithms including DEFLATE, BWT, snappy, etc.

Built-in user-defined functions (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.

SQL-like queries (HiveQL), which are implicitly converted into MapReduce or Tez, or Spark jobs.

By default, Hive stores metadata in an embedded Apache Derby database, and other client/server databases like MySQL can optionally be used.[12]

The first four file formats supported in Hive were plain text,[13] sequence file, optimized row columnar (ORC) format[14][15] and RCFile.[16][17] Apache Parquet can be read via plugin in versions later than 0.10 and natively starting at 0.13.[18][19]

COMPONENTS

1. **Hive Client**
2. **Hive Services**
3. **Processing and Resource Management**
4. **Distributed Storage**

### [ps2id id=’Hive-Client’ target=”/]Hive Client

Hive supports applications written in any language like Python, Java, C++, Ruby, etc. using JDBC, ODBC, and Thrift drivers, for performing queries on the Hive. Hence, one can easily write a hive client application in any language of its own choice.

Hive clients are categorized into three types:

#### 1. Thrift Clients

The Hive server is based on Apache Thrift so that it can serve the request from a thrift client.

#### 2. JDBC client

Hive allows for the Java applications to connect to it using the JDBC driver. JDBC driver uses Thrift to communicate with the Hive Server.

#### 3. ODBC client

# 21.RDD in Spark, Advantages and Disadvantages of it.

The primary abstraction the Spark is the concept of RDD, which Spark uses to achieve Faster and efficient [MapReduce](https://en.wikipedia.org/wiki/MapReduce) operations. Resilient Distributed Dataset (RDD) is the fundamental data structure of Spark. They are immutable Distributed collections of objects of any type. As the name suggests is a Resilient (Fault-tolerant) records of data that resides on multiple nodes.

## ****What are the key features?****

The key features of resilient distributed dataset are:

### ****Lazy Evaluation****

All Transformations in the [Apache Spark](https://www.xenonstack.com/blog/apache-spark-architecture/) are lazy, which means that they do not compute the results as and when stated in Transformation statements. Instead, they Keep track of the Transformation tasks using the concept of DAG (Directed Acyclic Graphs). Spark computes these Transformations when an action requires a result for the driver program.

### ****In-Memory Computation****

Spark uses in-memory computation as a way to speed up the total processing time. In the in-memory computation, the data is kept in RAM (random access memory) instead of the slower disk drives. This is very helpful as it reduces the cost of memory and allows for pattern detection, analyzes large data more efficiently. Main methods that accompany this are  cache()  and  persist()  methods.

### ****Fault Tolerance****

The fault-tolerant as they can track data lineage information to allow for rebuilding lost data automatically on failure. To achieve fault tolerance for the generated RDD’s, the achieved data is replicated among various Spark executors in worker nodes in the cluster.

### ****Immutability****

As it is effortless to share the immutable data safely among several processes, it turns out to be a very valid option. Immutability simply rules out lots of potential problems due to various updates from varying threads at once. Having Immutable data is safer to share across processes. But, these are not just immutable but also deterministic functions of their inputs which makes recreating the RDD parts possible at any given instance. We can think of its not only as a collection of data but a recipe for building new data from other data.

### ****Partitioning****

There are generally collections of various data items of massive volumes, that cannot fit into a single node and have to be partitioned across multiple nodes. Spark automatically does this partitioning of Resilient Distributed Datasets and distributes these partitions across different nodes. Key points related to these partitions are

* Each node in a spark cluster contains one or more partitions.
* Partitions in do not span multiple machines.
* The number of barriers in Spark is configurable and should be chosen efficiently.
* By increasing the number of executors on the cluster, parallelism can be increased in the system.

### ****Location Setup capability****

These are capable of clear placement preference to compute partitions. Placement preference refers to the defining information about the location of it. Here the DAG comes into play and places the partitions in a way that the task is close to the data it needs. Hence, the Speed of computation is increased.

# 22.List at least three common actions in Apache spark

# Commonly Used Actions

# Count() Returns the number of elements in the dataset

# Reduce(func) Aggregate the element of the dataset using a function func (which takes two

# arguments and returns one). The function should be commutative and

# associative so that it can be computed correctly in parallel

# Collect() Return all the elements of the dataset as an array at the driver program. This

# is usually useful after a filter or other operation that returns a sufficiently small

# subset of the data

# Take(n) Returns an array with first n elements

# First() Returns the first element of the dataset

# TakeOrdered(n,

# [ordering])

# Returns first n elements of RDD using natural order or custom operator