The Ultimate Big Data Masters Program (Cloud Focused)

Big Data Introduction
==========
what is big data?

3 V's is considered to be big data

1. Volume - Huge amount of data

2. Variety - Structured, Semi structured, Unstructured

Structured -
RDBMS tables - Database tables (Mysql)
for example an employee table in a mysql database

Semi Structured -
CSV, Json, XML
Unstructured -
log files, image, video
3. Velocity - The speed at which new data is coming in

Amazon Sale - how many iphones are sold in last 1 hour...

4. Veracity - The quality/correctness of data

5. Value - The Data should be able to generate value...

why are we going to learn a new technology stack to handle big data?

our traditional technologies will not be able to handle this big data

Monolythic vs Distributed System
==========
Mono means one

One big system holding a lot of power

Resources
==========
Compute - CPU cores (4 cores)
Memory - RAM (8 GB RAM)
Storage - Hard disk (1 TB harddisk)

X resources ~ Performance Y

2X resources < performance 2Y

Monolythic is not a scalable system

Distributed system
==========

Cluster - a group of machines (nodes)

5 node cluster

node1 node2 node3 node4 node5

in distributed if you double the resources the performance will double

2X resources ~ performance 2Y

all good big data systems are based on distributed architecture and not on monolythic because you get true scalability with distributed systems.

1. Distributed Storage

2. Distributed Processing/Computation

3. Scalability

Things we talked about
==========

1. what is big data - 5 V's

2. why do we need a completely new technology stack

3. Monolythic vs Distributed Architecture

4. Storage, Processing, Scalability

Overview of Hadoop
==========

Hadoop is a framework to solve big data problems

2007 - Hadoop 1.0

2012 - Hadoop 2.0

current version - Hadoop 3

3 core components
==========

HDFS / Mapreduce / YARN

HDFS - Hadoop distributed file system (Distributed Storage)

Mapreduce - Distributed Processing

YARN - yet another resource negotiator (Resource manager)

20 node hadoop cluster

writing a mapreduce code is really really hard - Java (Spark)

sqoop - data movement (Azure Datafactory)

pig - a scripting language , mainly used to clean the data..

Hive - provides a sql kind of interface to query your data

Oozie - workflow scheduler (Azure Datafactory)

Hbase - is a noSQL database (Cosmosdb)

Employee table - Hive
==========
10 lakh records

1
2
3.
.
.
.
1000000

select * from employee where employee id = 800000

NoSQL (Random access would be possible)

Challenges with Hadoop
==========
Mapreduce is very slow and its really hard to write the code

we need to learn so many different components and each has its own big learning curve..
On-premise

==========

what is hadoop

the core components of hadoop

some of the ecosystem technologies

Challenges of hadoop

Cloud and It's Advantages
==========
On-Premise vs Cloud
==========
50 node hadoop cluster

you buy 50 nodes

you procure a space to keep those nodes

cooling equipment

Advantage of Cloud

==========

Scalable
capex vs opex
Agility
Geo Distribution
Disaster recovery
Cost effective

3 types of cloud

==========

1. Public Cloud - Amazon AWS/ Microsoft Azure/ Google GCP
2. Private Cloud - within the organization
3. Hybrid cloud - Public + Private

Amazon AWS

Microsoft Azure
Google GCP

On Premise vs Cloud

Types of cloud

Advantages of cloud

Major Cloud providers

What is Apache Spark

==========

Hadoop - HDFS/Mapreduce/YARN

Mapreduce is the biggest bottleneck

Apache Spark

==========

General Purpose
In Memory - 10X to 100X faster than mapreduce
Compute engine

Hadoop is dead and spark is replacement of hadoop?

storage - HDFS
compute - MR
resource management - YARN

HDFS/Apache Spark/YARN

I need 2 things to work with

==========

Storage - HDFS / Amazon S3 / Azure ADLS Gen2 / GCS / local storage

Resource manager - YARN / Mesos / Kubernetes

Plug and play compute engine

In which language we write spark code?

python / Scala / Java / R

Python is easy to learn and the support for data science is the best when we talk about python

spark with python - pyspark

Big Data the Big Picture

==========

Data Engineering Flow

Multiple Sources -> Ingestion -> DataLake -> Processing -> Serving layer

for the serving layer a database or a datawarehouse is the best fit.

Hadoop

=======

Mysql db -> Sqoop -> HDFS -> Mapreduce/Spark -> Hive/Hbase

(employee) import file processing dwh

when to use a datawarehouse/database for serving layer - when you to do reporting

vs

when to use a nosql for your serving layer - when you are building a custom UI

Cloud

======

Azure Cloud

==========

Multiple Sources -> Azure DataFactory (ADF) -> ADLS Gen2 -> Azure Databricks/Azure Synapse -> Azure SQL / Cosmosdb

AWS Cloud

==========

Multiple Sources -> AWS Glue -> Amazon S3 -> AWS Databricks/Athena/Redshift -> AWS RDS / DynamoDB

Ingestion - Sqoop / ADF / AWS Glue

DataLake - HDFS / ADLS Gen2 / Amazon S3

Compute - Mapreduce / Spark / Azure Databricks / Synapse / Athena / Redshift

Serving - Hbase / Azure SQL / CosmosDB / AWS RDS / DynamoDB

Computation
==========
Serverless or Serverful
==========
AWS Athena vs AWS Redshift
==========
Amazon S3

Athena charges you based on amount of data scanned

you will use resources from a shared pool

$5 for 1 TB of data scanned

if your query scan 100 gb of data

.5$

AWS Redshift
==========
you turn on a dedicated cluster

it will charge you a lot...

serverless vs serverful

serverless will be less expensive

in serverless since resources are not dedicated you cannot get a guaranteed performance.
AWS - Athena vs Redshift

Azure - Synapse Serverless vs Synapse Dedicated Pool

Azure Datafactory for Ingestion

ADLS Gen2 - for datalake storage

Azure Databricks - for processing

Pyspark - for processing

Synapse - for processing

cosmosDB - for serving

Azure Datafactory - for Scheduling

Internals of HDFS

==========

block size - 128 mb

1 name node

multiple data nodes

data nodes hold the actual data

name node holds the metadata table and knows what is kept where

whenever a client wants to read a file... the request goes to the namenode.. namenode will check its table and tell the client from where to read...

how to make sure that we do not lose the data even when a data node goes down?
Replication factor - by default is 3

there is a concept of heartbeat.. every datanode sends a heartbeat to namenode every 3 seconds indication its alive...
if namenode does not receive heartbeat for 10 consecutive times (30 seconds)

Whats the need of a Data Engineer
==========
TRADITIONAL: MULTIPLE SOURCES -> ETL -> DWH

MODERN: MULTIPLE SOURCES -> ELT -> DATALAKE -> SPARK -> SERVE

Informatica

Talend

ADF

AWS Glue

========

concept of data locality - process the data on machine where it is kept

hadoop works on the principal of data locality.

=======