An

Industrial Synopsis Report

BIG DATA ANALYTICS

Submitted in for the partial fulfilment of the degree

By

Mitali Jhalani

Regn. No. 21MCAN196

JECRC™
UNIVERSITY
BUILD YOUR WORLD

CELEBAL
TECHNOLOGIES

Under the Guidance of

Faculty Internship Guide

Name: Mr. Manoj Tiwari

Industry Guide

Name: Mr. Kratik Mehta

Internship Organisation Details

Celebal Technologies is a premier software services company in the field of Data Science, Big Data and Enterprise Cloud. Celebal Technologies helps you to discover the competitive advantage by employing intelligent data solutions using cutting-edge technology solutions that can bring massive value to your organization. The core offerings are around "Data to Intelligence", wherein we leverage data to extract intelligence and patterns thereby facilitating smarter and quicker decision making for clients.

Celebal Technologies solutions are powered by Robotics, Artificial Intelligence and Machine Learning algorithms which offer improved business efficiency in the interconnected world. Companies clients are Product ISVs and delivery organisations across the globe who use our niche expertise in product development as well as Enterprise project implementations. We have adopted the highest standards of service quality and operational excellence, enabling its clients across a wide range of industries to transform into a data-driven enterprise. Our tailor-made solutions help enterprises maximise productivity, improve speed and accuracy.

Our adept team industry experts have strong expertise in various industry-standard technologies along with niche emerging areas including Blockchain, Analytics & Visualization, IoT, Chatbot, Data Science- AI & ML. We have hands-on experience in developing and performing supply chain analytics and media greenlighting solutions, connect the dots across data pits for generating actionable insights. We help modernize legacy and on-premise applications by leveraging modern cloud paradigms such as Microservices, Advance Analytics, API Management and other Cloud-native services.

With Celebal Technologies, who understands the core value of modern analytics over the enterprise, we help the business in improving business intelligence and more data-driven in architecting solutions. Our analytics team caters to ad-hoc and defined business analytics

Industry Mentor Profile

Name : Mr. Kratik Mehta

Designation : Architect (Big Data & EDW)

Experience :6+ years.

Technical Expertise :

- Project Management.
- Databricks
- Pyspark
- Tech Lead
- SQL, PYTHON, Azure

Achievements :

- Built a unified team culture.

Projects :

- Databricks migration and data quality

Internship Profile

Position Overview:

As a Data Analyst (DataBricks), you will collect, aggregate, store, and reconcile data in support of Customer's business decisions. You will design and build data pipelines, data streams, data service APIs, data generators and other end-user information portals and insight tools.

Responsibilities:

Plan, build and implement data solutions based on DataBricks

Design and build Modern Data Pipelines and Data Streams

Develop and Maintain Data Warehouse

Design and create ETL processes supplying Data Warehouse

Implement effective metrics and monitoring process

Preprocess of structured and unstructured data, create queries in databases

Requirements:

Contract of employment or B2B contract

For contract of employment: creative right up to 80%, 100% paid sick leave, annual bonus, private medical care for you and your family, cafeteria platform with sport card, internal rewards and referral program

Opportunity to work for Microsoft’s Global Alliance Partner of the Year (17 of the last 20 years!)

Exceptional development and training with a minimum 80 hours/year of training and paid Microsoft certifications

Dedicated career adviser to encourage your progression, engaged and helpful coworkers genuinely interested in you

Technical Skills:

Experience working with Azure Databricks

Familiarity with PySpark

Fluent Polish and English, both verbal and written

Mastery of SQL (T-SQL or PL SQL preferred)

Knowledge of at least one component: Azure Data Factory, Azure Data Lake, Azure SQL DW, Azure SQL

Experience with any programming language used for data engineering purpose: Python, Scala, R, Java etc

Ability to conduct data profiling, cataloging, and mapping for technical design and construction of technical data flows

MODE OF INTERNSHIP -OFFLINE.

LOCATION- 7th floor Corporate Tower JLN Marg, near Jawahar Circle, Malviya Nagar, Jaipur, Rajasthan 302017

Literature Review(Technology Used)

Databricks:

Databricks is a unified set of tools for building, deploying, sharing, and maintaining enterprise-grade data solutions at scale. The Databricks Lakehouse Platform integrates with cloud storage and security in your cloud account, and manages and deploys cloud infrastructure on your behalf.

Technology related to databricks:

Big Data:

The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. This is also known as the three Vs.

Put simply, big data is larger, more complex data sets, especially from new data sources. These data sets are so voluminous that traditional data processing software just can't manage them. But these massive volumes of data can be used to address business problems you wouldn't have been able to tackle before.

- Volume: The amount of data matters. With big data, you'll have to process high volumes of low-density, unstructured data. This can be data of unknown value, such as Twitter data feeds, clickstreams on a web page or a mobile app, or sensor-enabled equipment. For some organizations, this might be tens of terabytes of data. For others, it may be hundreds of petabytes.
- Velocity: Velocity is the fast rate at which data is received and (perhaps) acted on. Normally, the highest velocity of data streams directly into memory versus being written to disk. Some internet-enabled smart products operate in real time or near real time and will require real-time evaluation and action.
- Variety: Variety refers to the many types of data that are available. Traditional data types were structured and fit neatly in a relational database. With the rise of big data, data comes in new unstructured data types. Unstructured and semistructured data types, such as text, audio, and video, require additional preprocessing to derive meaning and support metadata.

The history of big data:

- **Although the concept of big data itself is relatively new, the origins of large data sets go back to the 1960s and '70s when the world of data was just getting started with the first data centers and the development of the relational database.**
- **Around 2005, people began to realize just how much data users generated through Facebook, YouTube, and other online services. Hadoop (an open-source framework created specifically to store and analyze big data sets) was developed that same year. NoSQL also began to gain popularity during this time.**
- **The development of open-source frameworks, such as Hadoop (and more recently, Spark) was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it's not just humans who are doing it.**
- **With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of machine learning has produced still more data**
- **While big data has come far, its usefulness is only just beginning. Cloud computing has expanded big data possibilities even further. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data. And graph databases are becoming increasingly important as well, with their ability to display massive amounts of data in a way that makes analytics fast and comprehensive**

- **Pyspark:** Apache Spark is written in Scala programming language. PySpark has been released in order to support the collaboration of Apache Spark and Python, it actually is a Python API for Spark. In addition, PySpark, helps you interface with Resilient Distributed Datasets (RDDs) in Apache Spark and Python programming language. This has been achieved by taking advantage of the Py4j library.

- **PySparkSQL**

- A PySpark library to apply SQL-like analysis on a huge amount of structured or semi-structured data. We can also use SQL queries with PySparkSQL. It can also be connected to Apache Hive. HiveQL can be also be applied. PySparkSQL is a wrapper over the PySpark core. PySparkSQL introduced the DataFrame, a tabular representation of structured data that is similar to that of a table from a relational database management system.

- **MLlib**

- MLlib is a wrapper over the PySpark and it is Spark's machine learning (ML) library. This library uses the data parallelism technique to store and work with data. The machine-learning API provided by the MLlib library is quite easy to use. MLlib supports many machine-learning algorithms for classification, regression, clustering, collaborative filtering, dimensionality reduction, and underlying optimization primitives.

- **Database:** A database is the collection of inter-related data which helps in the efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, views, schemas, reports, etc.

- **MongoDB:** MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for the storage and retrieval of data.

- **SQL:** Structured Query Language is a standard Database language that is used to create, maintain, and retrieve the relational.

- HADOOP:

- Apache Hadoop is an open source framework that is used to efficiently store and process large datasets ranging in size from gigabytes to petabytes of data. Instead of using one large computer to store and process the data, Hadoop allows clustering multiple computers to analyze massive datasets in parallel more quickly.

- Hadoop consists of four main modules:

- Hadoop Distributed File System (HDFS) – A distributed file system that runs on standard or low-end hardware. HDFS provides better data throughput than traditional file systems, in addition to high fault tolerance and native support of large datasets.
- Yet Another Resource Negotiator (YARN) – Manages and monitors cluster nodes and resource usage. It schedules jobs and tasks.
- MapReduce – A framework that helps programs do the parallel computation on data. The map task takes input data and converts it into a dataset that can be computed in key value pairs. The output of the map task is consumed by reduce tasks to aggregate output and provide the desired result.
- Hadoop Common – Provides common Java libraries that can be used across all modules.
- How Hadoop Works
- Hadoop makes it easier to use all the storage and processing capacity in cluster servers, and to execute distributed processes against huge amounts of data. Hadoop provides the building blocks on which other services and applications can be built.
- Applications that collect data in various formats can place data into the Hadoop cluster by using an API operation to connect to the NameNode. The NameNode tracks the file directory structure and placement of “chunks” for each file, replicated across DataNodes. To run a job to query the data, provide a MapReduce job made up of many map and reduce tasks that run against the data in HDFS spread across the DataNodes. Map tasks run on each node against the input files supplied, and reducers run to aggregate and organize the final output.
- The Hadoop ecosystem has grown significantly over the years due to its extensibility. Today, the Hadoop ecosystem includes many tools and applications to help collect, store, process, analyze, and manage big data. Some of the most popular applications are:

- Spark – An open source, distributed processing system commonly used for big data workloads. Apache Spark uses in-memory caching and optimized execution for fast performance, and it supports general batch processing, streaming analytics, machine learning, graph databases, and ad hoc queries.
- Presto – An open source, distributed SQL query engine optimized for low-latency, ad-hoc analysis of data. It supports the ANSI SQL standard, including complex queries, aggregations, joins, and window functions. Presto can process data from multiple data sources including the Hadoop Distributed File System (HDFS) and Amazon S3.
- Hive – Allows users to leverage Hadoop MapReduce using a SQL interface, enabling analytics at a massive scale, in addition to distributed and fault-tolerant data warehousing.
- HBase – An open source, non-relational, versioned database that runs on top of Amazon S3 (using EMRFS) or the Hadoop Distributed File System (HDFS). HBase is a massively scalable, distributed big data store built for random, strictly consistent, real-time access for tables with billions of rows and millions of columns.
- Zeppelin – An interactive notebook that enables interactive data exploration.
- YARN stands for “Yet Another Resource Negotiator“. It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN was described as a “Redesigned Resource Manager” at the time of its launching, but it has now evolved to be known as large-scale distributed operating system used for Big Data processing.

YARN also allows different data processing engines like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS (Hadoop Distributed File System) thus making the system much more efficient. Through its various components, it can dynamically allocate various resources and schedule the application processing. For large volume data processing, it is quite necessary to manage the available resources properly so that every application can leverage them.

YARN Features: YARN gained popularity because of the following features-

Scalability: The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.

Compatibility: YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.

Cluster Utilization:Since YARN supports Dynamic utilization of cluster in Hadoop, which enables optimized Cluster Utilization.

Multi-tenancy: It allows multiple engine access thus giving organizations a benefit of multi-tenancy.

Client: It submits map-reduce jobs.

Resource Manager: It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:

Scheduler: It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.

Application manager: It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.

Node Manager: It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.

Application Master: An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.

Container: It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

MAP REDUCE:

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. As the processing component, MapReduce is the heart of Apache Hadoop. The term "MapReduce" refers to two separate and distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

The reduce job takes the output from a map as input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

MapReduce programming offers several benefits to help you gain valuable insights from your big data:

Scalability. Businesses can process petabytes of data stored in the Hadoop Distributed File System (HDFS).

Flexibility. Hadoop enables easier access to multiple sources of data and multiple types of data.

Speed. With parallel processing and minimal data movement, Hadoop offers fast processing of massive amounts of data.

Simple. Developers can write code in a choice of languages, including Java, C++ and Python.

- **An example of MapReduce**

- **This is a very simple example of MapReduce. No matter the amount of data you need to analyze, the key principles remain the same.**

- **Assume you have five files, and each file contains two columns (a key and a value in Hadoop terms) that represent a city and the corresponding temperature recorded in that city for the various measurement days. The city is the key, and the temperature is the value. For example: (Toronto, 20). Out of all the data we have collected, you want to find the maximum temperature for each city**

across the data files (note that each file might have the same city represented multiple times).

- **Using the MapReduce framework, you can break this down into five map tasks, where each mapper works on one of the five files. The mapper task goes through the data and returns the maximum temperature for each city.**

- **For example, the results produced from one mapper task for the data above would look like this: (Toronto, 20) (Whitby, 25) (New York, 22) (Rome, 33)**

- **Assume the other four mapper tasks (working on the other four files not shown here) produced the following intermediate results:**

- **(Toronto, 18) (Whitby, 27) (New York, 32) (Rome, 37) (Toronto, 32) (Whitby, 20) (New York, 33) (Rome, 38) (Toronto, 22) (Whitby, 19) (New York, 20) (Rome, 31) (Toronto, 31) (Whitby, 22) (New York, 19) (Rome, 30)**

- **All five of these output streams would be fed into the reduce tasks, which combine the input results and output a single value for each city, producing a final result set as follows: (Toronto, 32) (Whitby, 27) (New York, 33) (Rome, 38).**

- **As an analogy, you can think of map and reduce tasks as the way a census was conducted in Roman times, where the census bureau would dispatch its people to each city in the empire. Each census taker in each city would be tasked to count the number of people in that city and then return their results to the capital city. There, the results from each city would be reduced to a single count (sum of all cities) to determine the overall population of the empire. This mapping of people to cities, in parallel, and then combining the results (reducing)**

is much more efficient than sending a single person to count every person in the empire in a serial fashion..

RDD:

RDDs are the main logical data units in Spark. They are a distributed collection of objects, which are stored in memory or on disks of different machines of a cluster. A single RDD can be divided into multiple logical partitions so that these partitions can be stored and processed on different machines of a cluster.

RDDs are immutable (read-only) in nature. You cannot change an original RDD, but you can create new RDDs by performing coarse-grain operations, like transformations, on an existing RDD.

An RDD in Spark can be cached and used again for future transformations, which is a huge benefit for users. RDDs are said to be lazily evaluated, i.e., they delay the evaluation until it is really needed. This saves a lot of time and improves efficiency.

Resilience: RDDs track data lineage information to recover lost data, automatically on failure. It is also called fault tolerance.

Distributed: Data present in an RDD resides on multiple nodes. It is distributed across different nodes of a cluster.

Lazy evaluation: Data does not get loaded in an RDD even if you define it. Transformations are actually computed when you call action, such as count or collect, or save the output to a file system.

Immutability: Data stored in an RDD is in the read-only mode—you cannot edit the data which is present in the RDD. But, you can create new RDDs by performing transformations on the existing RDDs.

In-memory computation: An RDD stores any immediate data that is generated in the memory (RAM) than on the disk so that it provides faster access.

Partitioning: Partitions can be done on any existing RDD to create logical parts that are mutable. You can achieve this by applying transformations to the existing partitions.

Transformations

ThesearefunctionsthatacceptstheexistingRDDsasinputandoutputoneormoreRDDs. However, thedataintheexistingRDDinSparkdoesnotchangeasitisimmutable.
Someofthetransformationoperationsareprovidedinthetablebelow:

Function	Description
map()	ReturnsanewRDDbyapplyingthefunctiononeachdataelement
filter()	ReturnsanewRDDformedbyselectingthoseelementsofthesourceonwhichthefunctionreturnstrue
reduceByKey()	Aggregatesthevaluesofakeyusingafunction
groupByKey()	Convertsa (key, value) pairintoa (key, <iterablevalue>) pair
union()	ReturnsanewRDDthatcontainsallelementsandargumentsfromthesourceRDD
intersection()	ReturnsanewRDDthatcontainsanintersectionoftheelementsinthedatasets

Actions

ActionsinSparkarefunctionsthatreturntheendresultofRDDcomputations.
ItusesalineagegraphtoloaddataontotheRDDinaparticularorder.
Afterallofthetransformationsaredone, actionsreturnthefinalresulttotheSparkDriver.
Actionsareoperationsthatprovidenon-RDDvalues.
SomeofthecommonactionsusedinSparkaregivenbelow:

Function	Description
count()	Getsthenumberofdataelementsinan RDD
collect()	Getsallthedataelementsinan RDD asanarray
reduce()	AggregatesdataelementsintoanRDDbytakingtwoargumentsandreturningone
take(n)	Fetchesthefirstnelementsofan RDD
foreach(operation)	Executestheoperationforeachdataelementin an RDD
first()	Retrievesthefirstdataelementofan RDD

CreatinganRDD

AnRDDcanbecreatedinthreeways. Let’sdiscussthemonebyone.

By Loading an External Dataset

YoucanloadanexternalfileontoanRDD. Thetypesoffilesyoucanloadarecsv, txt, JSON, etc.
Hereistheexampleofloadingatextfileontoan RDD:

Loading an External Dataset

ByParallelizingtheCollectionofObjects

WhenSpark’sparallelizemethodisappliedtoagroupofelements, anewdistributeddatasetiscreated. ThisdatasetisanRDD.

MakeyourselfjOb-readywiththesetopSparkInterviewQuestionsandAnswerstoday!

Below, youcanseehowtocreateanRDDbyapplyingtheparallelizemethodtoacollectionthatconsistsofsixelements:

ParallelizingtheCollectionofObjects

ByPerformingTransformationsontheExistingRDDs

OneormoreRDDscanbecreatedbyperformingtransformationsontheexistingRDDsasmentionedearlierinthistutorialpage. Thebelowfigureshowshowamap() functioncanbeusedtocreateanRDD:

PerformingTransformationsontheExistingRDDs

However, thedatainsideRDDsarenotalwaysorganizedorstructuredsincethedataisstoredfromdifferentsources.

InafurthersectionofthisApacheSparktutorial, youwilllearnaboutSparkSQLthatorganizesdataintorowsandcolumns. Besides, youwillcometoknowaboutSparkSQLlibrariesthatprovideAPIstoconnecttoSparkSQLthroughJDBC/ODBCconnectionsandperformqueries (tableoperations) onstructureddata, whichisnotpossibleinanRDDinSpark. Staytuned!