Big Data Summer Training

BigData Analytics-BigData Platforms

Prepared and Presenting By: Amrit Chhetri (Certified BigData Analyst),
Principal Techno-Functional Consultant and Principal IT Security Consultant

knowledgelab SM
Incubated @ STEP IIT (Kgp)

BigData Project Activities

Agendas/Modules:

- Project Team Members
- Sample SRS Documents

Project Team Members:

- Project will be given to team and each will have maximum of 3 trainees or students
- Students should start giving the name of team-members by 12 July 2016 and completed by 13 July, 2016
- Each team will be presented by a team leader and he/she will be responsible for any kind of interaction or communication
- Innovations on every area of BigData project will bear some extra rewards.
- Each team will put 20 minutes for 2 days to create the template version of SRS and need submit the template by 14 July 2016.

knowledgelab SM

Incubated @ STEP IIT (Kgp)

Sample SRS Documents:

- SRS (Software Requirement Specification) is an important document in BigData Analytics and almost every BigData Project contains the common section and only functional requirements and software platform selected will be different
- It should contain summarized table of BigData Platform select for the project and implementation details
- In SRS of BigData Project, the details of Datawarehouse to store data is very crucial
- It should also contain the process of requirement analysis under Functional Requirement/Analysis Section

Hadoop-BigData Core Engine-II

Agendas/Modules:

- BigData Vs. Traditional Analytics
- Hadoop Introduction
- Hadoop MapReduce-II
- MapReduce on Cloudera
- MapReduce on Apache Hadoop
- BigData Architecture-Case Studies

BigData Vs. Traditional Analytics:

- BigData Analytics Solution should support:
  - Larger volume data, different types of data, complex data-model and scalability
- Traditional Analytics/BI Vs. BigData Analytics:
  - BigData Analytics handles larger volume of data but not the Traditional BI/Analytics
  - Hadoop, BigData's core engine is scalable, robust and fault-proof, by default
  - Scalability is an issue with Traditional Analytics but scalability is the main feature of Hadoop
  - Traditional Open Source BI is complex to design but BigData is easy to architect

Hadoop MapReduce-II:

- The intermediate results produced by reduce() function of Reducer are saved on HDFS
- MapReduce storage file system, HDFS has characteristics need for larger volume of data
  - Distributed storage :
  - Snapshots :
  - Federations :
- YARN is new in Hadoop and ....
- Hadoop 2.0 has three new components –Resource Manager, Node Manager and Application Manager

MapReduce on Cloudera:

- The steps to MapReduce on Java using Cloudera Quick Start VM:
  - Open a Linux Terminal on Cloudera QuickStart VM
  - Change Directory to /usr/lib/hadoop-mapreduce/
  - Type jsp localhost to check Hadoop
  - Execute, # hadoop jar WordCount.jar WordCountDriver , (Reports missing files)
  - Create two text files:
  - echo "Senssor is important elements of IoT"> /home/cloudera/testfile1
  - echo "IoT is the future of intelligent computation"> /home/cloudera/testfile2
  - Create input directory on HDFS,# hdfs dfs -mkdir /user/cloudera/input
  - Copy the files to Hadoop input directory
  - # hdfs dfs -put /home/cloudera/testfile1 /user/cloudera/input
  - # hdfs dfs -put /home/cloudera/testfile2 /user/cloudera/input
  - Execute,#hadoop jar WordCount.jar WordCountDriver /user/cloudera/input /user/cloudera/output
  - Check output directory, # hdfs dfs -ls /user/cloudera/output

MapReduce on Apache Hadoop:

- The steps to MapReduce on Java using Apache Hadoop (Custom Configuration)
  - Open Linux Terminal and change Directory to /usr/local/hadoop/sbin
  - Type jsp localhost to check Hadoop
  - Execute, # hadoop jar WordCount.jar WordCountDriver , (Reports missing files)
  - Create two text files:
  - echo "Senssor is important elements of IoT"> /home/apachehadoop/testfile1
  - echo "IoT is the future of intelligent computation"> /home/apachehadoop/testfile2
  - Create input directory on HDFS,# hdfs dfs -mkdir /user/apachehadoop/input
  - Copy the files to Hadoop input directory
  - # hdfs dfs -put /home/apachehadoop/testfile1 /user/apachehadoop/input
  - # hdfs dfs -put /home/apachehadoop/testfile2 /user/apachehadoop/input
  - Execute,#hadoop jar WordCount.jar WordCountDriver /user/apachehadoop/input /user/apachehadoop/output
  - Check output directory, # hdfs dfs -ls /user/aachehadoop/output

BigData Architecture-Case Studies:

- BigData Analytics is designed with Hadoop and which always robust by default
- Different set of Tools or Frameworks are used in designing BigData Analytics
- Selection of right set of Tools or Frameworks is essentials to delivered great BigData Analytics
- Some of successful BigData Platform stacks are:
  - Usage and Bill Analysis: JSON, Amazon Redshift, Amazon SQS,Amazon S3, Elastic Beantalk, Jaspersoft BI, Python
  - Click Stream Analysis : Flume, Hadoop, Amazon S3, HBase, Oozie, Hive

Advanced Pythone

Agendas/Modules:

- Python In-Built Functions
- Python on Linux- Terminal and PyDev
- Python Collection Module

Python In-Built Functions:

- Python has huge set of built-in and here are some examples:
  - random.random() : Generates values between 0.0 to 1.0
  - random.randint(min, max) : Generates integer number randomly between minimum and maximum values
  - math.sqrt(number): Return the square root of the given function
  - s.getcwd() or os.getcwdu() : Gives current working directory
  - os.system() : Allows to run OS commands
  - Example:

```
import random ;import math ;import os
x=int(input("Enter First Number :"))
y=int(input("Enter Second Number: "))
print("Power:", pow(x,y))
print("Factorial:", math.factorial(x))
print(random.randint(y,y))
os.system("netstat -an ")
```

Python on Linux-Terminal & PyDev:

- Python on Linux is most used combination of Data Science
- In Linux, Python can be installed in two ways:
  - Using IDE : Eclipse and Pydev
  - On Terminal using standard editor : gedit and python command
- In Linux, Python script can be executed as $ python <scriptname.py> arg1 arg2
- Steps to configure Python to run with Eclipse on Linux are:
  - Install JDK 1.7 or 1.8
  - Install Eclipse Mars 2
  - Open Eclipse and click on Help -> New Software -> click on 'Add' button and give http://pydev.org/updates to install Pydev plugin
  - In preferences, click Pydev and add python executable file

BigData, ETL and Analytics Designing

Agendas/Modules:

- ETL with Sqoop
- ETL with Talend

ETL With Sqoop:

- Sqoop is Data Integration Service developed on Open Source Hadoop Technology
- Sqoop is meant to transform data between Hadoop Cluster and Database using JDBC ,in bi-direction
- Scoop- Data Integration Steps- Moving to HDFS:
  - import --connect jdbc:mysql://<DB IP>/database --table orders --username <DB User> –P
- Data Integration Steps- Moving to Hive:
  - #sqoop import --hive-import --create-hive-table --hive-table orders –connect jdbc:mysql://<DB IP>/database --table orders --username <DB User> -P <password>

ETL with Talend Studio:

- Talend is the world-class ETL tool available for BigData and the Data Systems
- It is used to move data to BigData Warehouse designed using HBase, Hive and other technology
- Steps to transform data using Talend Studio:
  - Install JDK on Windows or Linux (or Ubuntu)
  - Install Talend Studio to and run it
  - Create a transformation and create sources for targeted source like HBase, Hive and others
  - Draw the transformation mapping on main screen
  - Now, either schedule the job to run in future and to run immediately

Hive-Query Processing Engine

Agendas/Modules:

- Introduction to Hive
- Introduction to HiveQL
- Writing HiveQL Statements

Introduction to Hive:

- DW System for processing un-structured data
- DW Software for querying and managing large datasets
- DW Infrastructure built on top of Hadoop
- High Level Data Processing/Query Language
- Works on top of HDFS(Hadoop Distributed File System) of Hadoop
- Supports HDFS as Storage, MapReduce, Execution and HiveQL for Query execution
- Stores Schemas or Meta-Data on Database
- Derby is default database for Hive but it supports others too(Oracle MySQL)
- Support Apache Spark(In-Memory, Machine Learning and Graphing API)
- Hive Components are:
  - Driver ,Query Compiler , Optimizer
  - NameNode
  - HiveServer2 Engine

Introduction to HiveQL:

- HiveQL is SQL-like data querying language
- HiveQL runs slowly using MapReduce of Hadoop
- HiveQL is similar to SQL but functions and architectural flow is completely different
- Driver of Hive sends the Hive Query to optimizer before actually it runs
-

Writing HiveQL Statements:

- Sample Examples:
  - CREATE : CREATE logs(ip string, size string, time string) ROW FORMAT DELIMITED FIELDS TERMINATED BY ' ‘
  - LOADING : LOAD DATA LOCAL INPATH 'log.log' OVERWRITE INTO TABLE logs;
  - SELECTING : SELECT * FROM logs;
  - UPDATE : UPDATE logs SET ip='192.168.2.10' WHERE time='2:30';
  - ALTER : ALTER TABLE logs COLUMN MODIFY (time STRING);
  - WHERE : select * from logs WHERE Ip='192.168.2.10';
  - GROUP BY : SELECT COUNT(*), logs.ip,count(*) FROM logs logs GROUP BY logs.ip;
- HiveQL execution uses Hadoop MapReduce Services and it slower than standard SQL statements