**EXPERIMENT 5**

|  |  |
| --- | --- |
| **Objective :**  **Theory:**  **Performance:**  **Deliverables:**  **Summary :** | **Programming exercise and analytics of a dataset using Pig.**  Apache Pig is a high-level language platform developed to execute queries on huge datasets that are stored in HDFS using Apache Hadoop. It is similar to SQL query language but applied on a larger dataset and with additional features. The language used in Pig is called PigLatin. It is very similar to SQL. It is used to load the data, apply the required filters and dump the data in the required format. It requires a Java runtime environment to execute the programs. Pig converts all the operations into Map and Reduce tasks which can be efficiently processed on Hadoop. It basically allows us to concentrate upon the whole operation irrespective of the individual mapper and reducer functions.  Pig is generally used with **Hadoop**; we can perform all the data manipulation operations in Hadoop using Apache Pig.  To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.  To analyze data using **Apache Pig**, programmers need to write scripts using Pig Latin language. All these scripts are internally converted to Map and Reduce tasks. Apache Pig has a component known as **Pig Engine** that accepts the Pig Latin scripts as input and converts those scripts into MapReduce jobs. Why Do We Need Apache Pig? Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.   * Using **Pig Latin**, programmers can perform MapReduce tasks easily without having to type complex codes in Java. * Apache Pig uses **multi-query approach**, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the   development time by almost 16 times.   * Pig Latin is **SQL-like language** and it is easy to learn Apache Pig when you are familiar with SQL. * Apache Pig provides many built-in operators to support data operations like joins, filters,   ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.  [Download](https://drive.google.com/file/d/1JfiudjZwsWzn6dXYf4RRKkcvdNsenYWh/view?usp=sharing) HDP Hortonworks sandbox from given link and import into Virtualbox.    Type in browser 127.0.0.1:8888 for accessing HDP  To explore HDP Sandbox navigate to 127.0.0.1:8000 by entering username as hue and password 1111    After Successful login you will get below screen    you can also login to your HDP sandbox by CLI, type  **user: sudo ssh root@127.0.0.1 -p 2222**  **password : hadoop**    To start service of Ambari run script as  **[root@sandbox ~]# ./start\_ambari.sh**    After Starting service of Ambari open UI of Ambari from browser 127.0.0.1:8080 and login with username and password as **admin**    After successful login you able to see Dashboard of Ambari which having all Hadoop ecosystem components monitoring as shown below:    You can start all Hadoop ecosystem components by selecting start all services from Action menu as shown:    If you need to start only selected services then you have to select that particular service and then from service action menu you need to start.  **Let’s demonstrate Pig and clean the dataset provided on the Moodle.**  Upload the [data](https://drive.google.com/drive/folders/1nBtKnheR7jO8XHDOPi1lkIugBlS7HKp5?usp=drive_link) to the file system using Ambari interface. Task: Write a Pig script that satisfies all of the following criteria:  * Load all of the data in /user/hue/flight\* * Remove all rows in the flightdelays data where the DepTime column equals the string "NA". * The output should only contain the Year, Month, DayofMonth, DepTime, UniqueCarrier, FlightNum, ArrDelay, Origin and Dest * Store the result as comma-separated records in a new directory in HDFS named   /user/hue/flightdelays\_clean   * Save the script in a file named Task01.pig.  Solution Pig Query: A = LOAD '/user/hue/flight\*' USING PigStorage(',') AS(Year, Month, DayofMonth, DayOfWeek, DepTime, CRSDepTime, ArrTime, CRSArrTime, UniqueCarrier, FlightNum, TailNum, ActualElapsedTime, CRSElapsedTime, AirTime,ArrDelay, DepDelay, Origin, Dest, Distance, TaxiIn, TaxiOut, Cancelled, CancellationCode, Diverted, CarrierDelay, WeatherDelay, NASDelay, SecurityDelay, LateAircraftDelay);  B = FILTER A BY (chararray)DepTime != 'NA';  C = FOREACH B GENERATE Year, Month, DayofMonth, DepTime, UniqueCarrier, FlightNum,ArrDelay,Origin,Dest;  STORE C INTO '/user/hue/flightdelays\_clean' USING PigStorage(',');  Write a solution query to Pig Query editor and click on Execute button for run that.    After Successful execution you can see newly created directory flightdelays\_clean whcih contains output files.    flightdelays\_clean whcih contains output file as **part-m-00000**    Click on **part-m-00000** file to view output which having clean dataset.  Step wise Screen shots of the Pig script execution.  Thus, we have learnt to analyse and clean the larger sets of data representing them into data flows using Pig. |