ANALYSING H1B VISA DATA USING HADOOP ECOSYSTEM

Project: H1B Visa Data Analysis

Name: Sharvari Jagdish Bhovad

Registration No: R180010900078

Centre: Borivali, Mumbai

INTRODUCTION

Data was born with the introduction of computer. Every time when computer was used it generated some information, this information is nothing but data. This data also has data of its own, known as metadata. In earlier phase of computing, data was stored on flat files. Storing data on flat files had many drawbacks. On flat files each fields of information are separated by space, tab, comma, semicolon or any other symbol. These data was less organized, insecure and not reliable.

With increase in data and need for storing data on better, reliable and secure platform, concept of database was introduced. Database is a collection of data, which is organized into files called tables. These tables provide a systematic way of accessing, managing, and updating data. A relational database is one that contains multiple tables of data that relate to each other through special key fields. Relational databases are far more flexible (though harder to design and maintain) than what are known as flat file databases, which contain a single table of data. Relational databases require a relational database management system (RDBMS) to manage and access the data.
These traditional database are been used for long time. Most of these data was structured, or neatly organized in databases. Then the world went digital and the internet came along. It was no more about just storing data but also about studying and analyzing it.

Data was no more in structured format. It started coming in different shape and formats, which is known as unstructured data. Unstructured data is generated by all our digital interactions, from email to online shopping, text messages to tweets, Facebook updates to YouTube videos. And the number of gadgets recording and transmitting data, from smartphones to intelligent fridges, industrial sensors to CCTV cameras, has also proliferated globally, leading to an explosion in the volume of data. These data sets are now so large and complex that we need new tools and approaches to make the most of them. Such huge volume of data is called big data. Using traditional data with unstructured data has some limitation.

Problem faced by traditional data storage system:-

- It only stores structured data.
- Traditional storage cost too much.
- Traditional storage doesn't scale too much.
- It works better when the volume of data is low (in Gigabytes).
- It doesn't support Online Analytical Processing, which is used in Data Mining techniques.
- Takes lot of time to process huge amount of data.

This is where Hadoop comes into picture.

HADOOP

Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It is part of the Apache project sponsored by the Apache Software Foundation.

<u>Hadoop advantage over traditional system</u>:-

- Scalable

  Hadoop is a highly scalable storage platform, because it can stores and distribute very large data sets across hundreds of inexpensive servers that operate in parallel.
  Hadoop enables businesses to run applications on thousands of nodes involving many thousands of terabytes of data.

- Cost effective

  Hadoop also offers a cost effective storage solution for businesses' exploding data sets.

- Flexible

  Hadoop enables businesses to easily access new data sources and tap into different types of data (both structured and unstructured) to generate value from that data.
  Hadoop can be used for a wide variety of purposes, such as log processing, recommendation systems, data warehousing, and market campaign analysis and fraud detection.

- Fast

  Hadoop's unique storage method is based on a distributed file system that basically 'maps' data wherever it is located on a cluster.
  The tools for data processing are often on the same servers where the data is located, resulting in much faster data processing. Hadoop is able to efficiently process terabytes of data in just minutes, and petabytes in hours.

- Resilient to failure

  A key advantage of using Hadoop is its fault tolerance. When data is sent to an individual node, that data is also replicated to other nodes in the cluster, which means that in the event of failure, there is another copy available for use.

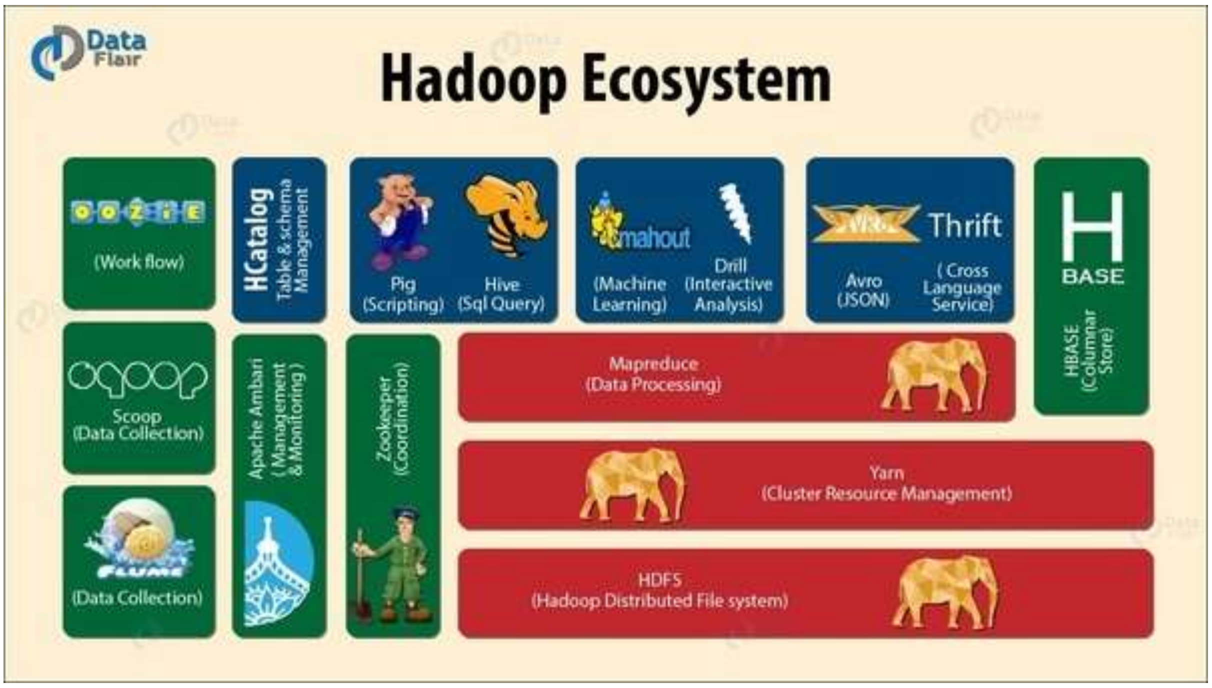
Hadoop makes it possible to run applications on systems with thousands of commodity hardware nodes, and to handle thousands of terabytes of data. Its distributed file system facilitates rapid data transfer rates among nodes and allows the system to continue operating in case of a node failure. This approach lowers the risk of catastrophic system failure and unexpected data loss, even if a significant number of nodes become inoperative. Consequently, Hadoop quickly emerged as a foundation for big data processing tasks, such as scientific analytics, business and sales planning, and processing enormous volumes of sensor data, including from internet of things sensors.

Hadoop is increasingly becoming the go-to framework for large-scale, data-intensive deployments. Hadoop is built to process large amounts of data from terabytes to petabytes and beyond. With this much data, it’s unlikely that it would fit on a single computer's hard drive, much less in memory. The beauty of Hadoop is that it is designed to efficiently process huge amounts of data by connecting many commodity computers together to work in parallel. Using the MapReduce model, Hadoop can take a query over a dataset, divide it, and run it in parallel over multiple nodes. Distributing the computation solves the problem of having data that’s too large to fit onto a single machine.

Apache Hadoop controls costs by storing data more affordably per terabyte than other platforms. Instead of thousands to tens of thousands per terabyte, Hadoop delivers compute and storage for hundreds of dollars per terabyte.

Fault tolerance is one of the most important advantages of using Hadoop. Even if individual nodes experience high rates of failure when running jobs on a large cluster, data is replicated across a cluster so that it can be recovered easily in the face of disk, node or rack failures.

HADOOP ECOSYSTEM



<u>Data Collection</u>

Flume	Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.
Sqoop	Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases

<u>Storage</u>

Hdfs	The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It provides high-performance access to data across Hadoop clusters.
HBase	Apache HBase is an open source NoSQL database that provides real-time read/write access to those large datasets.

<u>Resource Management</u>

Yarn	Apache Hadoop YARN (Yet Another Resource Negotiator) is a cluster management technology. YARN is the prerequisite for Enterprise Hadoop, providing resource management and a central platform to deliver consistent operations, security, and data governance tools across Hadoop clusters.

Coordination

Zookeeper	ZooKeeper is an open source Apache project that provides centralized infrastructure and services that enable synchronization across an Apache Hadoop cluster. Apache ZooKeeper is an open source file application program interface (API) that allows distributed processes in large systems to synchronize with each other so that all clients making requests receive consistent data.

Processing

MapReduce	Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data(multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

Analysis

Hive	Hive is an open-source data warehouse system for querying and analyzing large datasets stored in Hadoop files.
Pig	Apache Pig is an open-source technology that offers a high-level mechanism for the parallel programming of MapReduce jobs to be executed on Hadoop clusters.

AP ACHE PIG

Pig is a high level scripting language that is used with Apache Hadoop. Pig enables data workers to write complex data transformations without knowing Java. Pig's simple SQL-like scripting language is called Pig Latin, and appeals to developers already familiar with scripting languages and SQL.

Pig is complete; it allows us to do all required data manipulations in Apache Hadoop with Pig. Through the User Defined Functions (UDF) facility in Pig, Pig can invoke code in many languages like JRuby, Jython and Java. We can also embed Pig scripts in other languages. Pig ca nbe used as a component to build larger and more complex applications that tackle real business problems.

Pig works with data from many sources, including structured and unstructured data, and stores the results into the Hadoop Data File System.

Pig scripts are translated into a series of MapReduce jobs that are run on the Apache Hadoop cluster.

Advantage of Pig

- Pig Hadoop follows a multi query approach thus it cuts down on the number times the data is scanned.
- Pig Hadoop is very easy to learn read and write if you are familiar with SQL.
- Pig provides the users with a wide range of nested data types such as Maps, Tuples and Bags that are not present in MapReduce along with some major data operations such as Ordering, Filters, and Joins.
- Performance of Pig is on par with the performance of raw Map Reduce.

APACHE HIVE

Apace Hive is a data warehouse system that is often used with an open-source analytics platform called Hadoop. Hadoop has become a popular way to aggregate and refine data for businesses. Hadoop users may use tools like Apache Spark or MapReduce to compile data in precise ways before storing it in a file handling system called HDFS. From there, the data can go into Apache Hive for central storage.

Advantage of Hive

Hive Hadoop provides the users with strong and powerful statistics functions.

- Hive Hadoop is like SQL, so for any SQL developer the learning curve for Hive will almost be negligible.
- Hive Hadoop can be integrated with HBase for querying the data in HBase whereas this is not possible with Pig. In case of Pig, a function named HbaseStorage () will be used for loading the data from Hbase.

Both Apache Pig and Apache Hive internally run MapReduce mechanism.

MAPREDUCE

MapReduce is a core component of the Apache Hadoop software framework.

MapReduce is the data processing layer of Hadoop. It is a software framework for easily writing applications that process the vast amount of structured and unstructured data stored in the Hadoop Distributed Filesystem (HDFS). It processes the huge amount of data in parallel by dividing the job (submitted job) into a set of independent tasks (sub-job). By this parallel processing speed and reliability of cluster is improved. We just need to put the custom code (business logic) in the way map reduce works and rest things will be taken care by the engine.

How MapReduce works

The original version of MapReduce involved several component daemons, including:

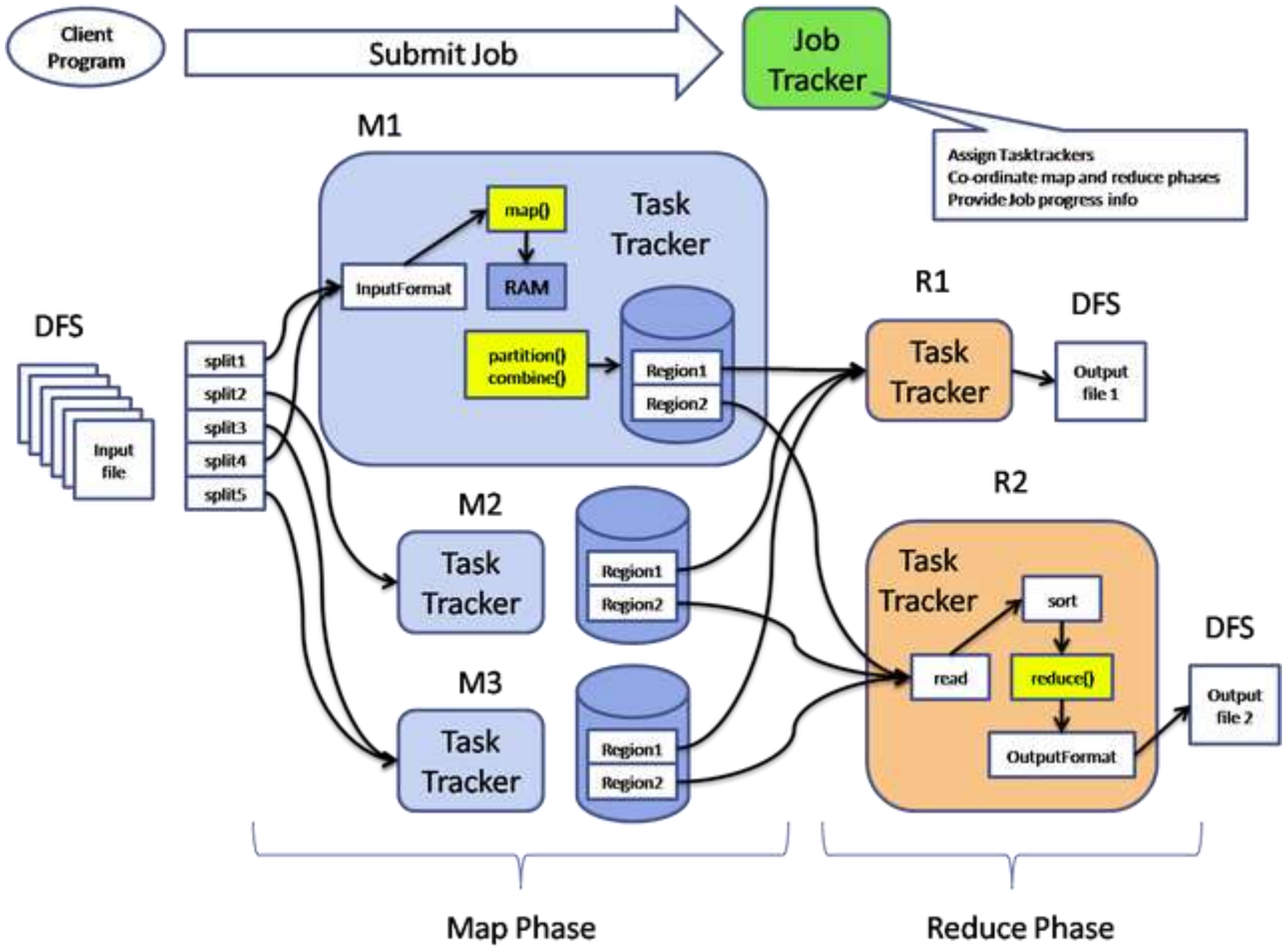
- JobTracker - the master node manages all the jobs and resources in a cluster.
- TaskTrackers - agents deployed to each machine in the cluster to run the map and reduce tasks
- JobHistory Server -- a component that tracks completed jobs and is typically deployed as a separate function or with JobTracker.

With the introduction of MapReduce and Hadoop version 2, previous JobTracker and TaskTracker daemons have been replaced with components of Yet Another Resource Negotiator (YARN), called ResourceManager and NodeManager.

- ResourceManager runs on a master node and handles the submission and scheduling of jobs on the cluster. It also monitors jobs and allocates resources.
- NodeManager runs on slave nodes and interoperates with Resource Manager to run tasks and track resource usage. NodeManager can employ other daemons to assist with task execution on the slave node.

To distribute input data and collate results, MapReduce operates in parallel across massive cluster sizes. Because cluster size doesn't affect a processing job's final results, jobs can be split across almost any number of servers. Therefore, MapReduce and the overall Hadoop framework simplify software development.

HDFS ARCHITECTURE FLOW



<u>There are 4 entities involved in classing mapreduce</u>

- ➢ The Job client who submits the job.
- ➢ The Job tracker handles overall execution of job. It is a java application with main class JobTracker.
- ➢ The task tracker, who run the individual tasks. It is a java application with main class TaskTracker.
- ➢ The shared file system, provided by HDFS

<u>The following are the main phases in map reduce</u>

<u>Job Submission</u>
The submit() method on job creates an internal instance of JobSubmitter and calls submitJobInternal() method on it. Having submitted the job, waitForCompletion() polls the job's progress once a second.
On calling this method following happens.

- ➢ It goes to JobTracker and gets a jobId for the job
- ➢ Perform checks if the output directory has been specified or not. If specified, whether the directory already exists or is new. And throws error if any such thing fails.

- Computes input split and throws error if it fails to do so, because the input paths don't exist.
- Copies the resources to JobTracker file system in a directory named after Job Id. These resources include configuration files, job jar file, and computed input splits. These are copied with high redundancy by default a factor of 10.
- Finally it calls submitJob() method on JobTracker.

Job Initialization

Job tracker performs following steps

- Creates bookkeeping object to track tasks and their progress
- For each input split creates a map tasks.
- The number of reduce tasks is defined by the configuration mapred.reduce.tasks set by setNumReduceTasks().
- Tasks are assigned task ID's at this point.
- In addition to this 2 other tasks are created: Job initialization task and Job clean up task, these are run by tasktrackers
- Job initialization task , based on output committed, like in case of FileOutputCommitter, creates the output directory to store the tasks output as well as temporary output
- Job clean up tasks which delete the temporary directory after the job is complete.

Task Assignment

TaskTracker sends a heartbeat to jobtracker every five seconds. This heartbeat serves as a communication channel, will indicate whether it is ready to run a new task. They also send the available slots on them.
Here is how job allocation takes place.

- JobTracker first selects a job to select the task from, based on job scheduling algorithms.
- The default scheduler fills empty map task before reduce task slots.
- For a map task then it chooses a tasktracker which is in following order of priority: data-local, rack-local and then network.
- For a reduce task, it simply chooses a task tracker which has empty slots.
- The number of slots which a task tracker has depends on number of cores.

Task Execution

Following is how a job is executed
Setup:

- TaskTracker copies the job jar file from the shared filesystem (HDFS) and any files needed to run the tasks from distributed cache to TaskTracker's local file system.
- Tasktracker creates a local working directory and un-jars the jar file into the local file system.
- It then creates an instance of TaskRunner.

Action:

- Tasktracker starts TaskRunner in a new JVM to run the map or reduce task.
- Separate process is needed so that the TaskTracker does not crash in case of bug in user code or JVM.
- The child process communicates it progress to parent process umbilical interface.
- Each task can perform setup and cleanup actions, which are run in the same JVM as the task itself, based on OutputComitter.
- In case of speculative execution the other tasks are killed before committing the task output, only one of the duplicate tasks is committed.
- Even if the map or reduce tasks is run via pipes or via socket as in case of streaming, we provide the input via stdin and get output via stdout from the running process.

Job/Task Progress

Here is how the progress is monitored of a job/task

- JobClient keeps polling the JobTracker for progress.
- Each child process reports its progress to parent task tracker.
- If a task reports progress, it sets a flag to indicate that the status change should be sent to the task tracker. The flag is checked in a separate thread every 3 seconds, and if set it notifies the task tracker of the current task status.
- Task tracker sends its progress to JobTracker over the heartbeat for every five seconds. Counters are sent less frequently, because they can be relatively high-bandwidth.
- JobTracker then assembles task progress from all task trackers and keeps a holistic view of job.
- The Job receives the latest status by polling the job tracker every second.

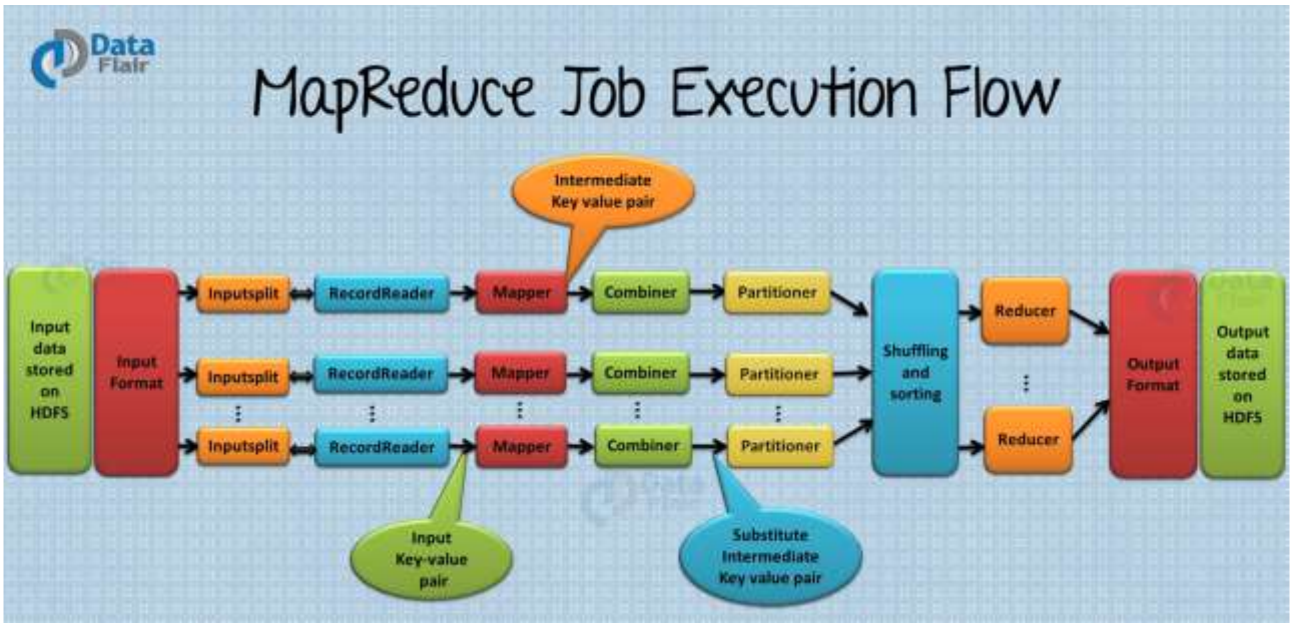
Job Completion

On Job Completion the cleanup task is run.

- Task sends the task tracker job completion. This in turn is sent to job tracker.
- Job Tracker then sends the job completion message to client, when it polls.
- Job tracker cleans up its working state for the job and instructs task trackers to do the same; it cleans up all the temporary directories.
- This causes jobclient’s waitForJobToComplete() method to return.

MapReduce Job Execution Flow

In Hadoop, MapReduce works by breaking the data processing into two phases: Map phase and Reduce phase. The map is the first phase of processing, where we specify all the complex logic/business rules/costly code. Reduce is the second phase of processing, where we specify light-weight processing like aggregation/summation.


Data Flair
MapReduce Job Execution Flow
Intermediate Key value pair
Input data stored on HDFS
Input Format
Inputsplit
RecordReader
Mapper
Combiner
Partitioner
Shuffling and sorting
Reducer
Output Format
Output data stored on HDFS
Input Key-value pair
Substitute Intermediate Key value pair

1. Input Files

The data for a MapReduce task is stored in input files, and input files typically lives in HDFS. The format of these files is arbitrary, while line-based log files and binary format can also be used.

2. InputFormat

Now, InputFormat defines how these input files are split and read. It selects the files or other objects that are used for input. InputFormat creates InputSplit.

3. InputSplits

It is created by InputFormat, logically represent the data which will be processed by an individual Mapper. One map task is created for each split; thus the number of map tasks will be equal to the number of InputSplits. The split is divided into records and each record will be processed by the mapper.

4. RecordReader

It communicates with the InputSplit in Hadoop MapReduce and converts the data into key-value pairs suitable for reading by the mapper. By default, it uses TextInputFormat for converting data into a key-value pair. RecordReader communicates with the InputSplit until the file reading is not completed. It assigns byte offset (unique number) to each line present in the file. Further, these key-value pairs are sent to the mapper for further processing.

5. Mapper

It processes each input record (from RecordReader) and generates new key-value pair, and this key-value pair generated by Mapper is completely different from the input pair. The output of Mapper is also known as intermediate output which is written to the local disk. The output of the Mapper is not stored on HDFS as this is temporary data and writing on HDFS will create unnecessary copies (also HDFS is a high latency system). Mappers output is passed to the combiner for further process.

6. Combiner

The combiner is also known as 'Mini-reducer'. Hadoop MapReduce Combiner performs local aggregation on the mappers' output, which helps to minimize the data transfer between mapper and reducer. Once the combiner functionality is executed, the output is then passed to the partitioner for further work.

7. Partitioner

Hadoop MapReduce, Partitioner comes into the picture if we are working on more than one reducer (for one reducer partitioner is not used).

Partitioner takes the output from combiners and performs partitioning. Partitioning of output takes place on the basis of the key and then sorted. By hash function, key (or a subset of the key) is used to derive the partition.

According to the key value in MapReduce, each combiner output is partitioned, and a record having the same key value goes into the same partition, and then each partition is sent to a reducer. Partitioning allows even distribution of the map output over the reducer.

8. Shuffling and Sorting

Now, the output is Shuffled to the reduce node (which is a normal slave node but reduce phase will run here hence called as reducer node). The shuffling is the physical movement of the data which is done over the network. Once all the mappers are finished and their output is shuffled on the reducer nodes, then this intermediate output is merged and sorted, which is then provided as input to reduce phase.

9. Reducer

It takes the set of intermediate key-value pairs produced by the mappers as the input and then runs a reducer function on each of them to generate the output. The output of the reducer is the final output, which is stored in HDFS.

10. RecordWriter

It writes these output key-value pair from the Reducer phase to the output files.

11. OutputFormat

The way these output key-value pairs are written in output files by RecordWriter is determined by the OutputFormat. OutputFormat instances provided by the Hadoop are used to write files in HDFS or on the local disk. Thus the final output of reducer is written on HDFS by OutputFormat instances.

Hence, in this manner, a Hadoop MapReduce works over the cluster.

With the help of mapreduce concept and by using various hadoop ecosystems we will further analyze a big volume of data.

PROJECT DEFINITION

The H-1B is a non-immigrant visa in the United States; it is designed to bring foreign professionals with college degrees and specialized skills to fill jobs when qualified Americans cannot be found. However the outsourcing companies in recent years have dominated the entire H1B visa program.

We will analyze the H1B visa data from 2011 to 2016 and find out how this visa program has been used around the Globe.

Following analysis will provide us stats like:-

- Which city or state in US is popular for specified skill set?
- Which profession or job position is in demand?
- Which company files most petition for h1b_visa.
- Increase or decrease in petition filled.

Here, we are using MapReduce, Apache hive, Apache pig and Sqoop to analyze this data.

HARDWARE REQUIREMENT

Minimum RAM required: 4GB (Suggested: 8GB)

Minimum Free Disk Space: 25GB

Minimum Processor i3 or above

Operating System of 64bit (Suggested)

SOFTWARE REQUIREMENT

UBUNTU	- version 14.04.1
JAVA	- version 1.7.0_65
HADOOP	- version 2.6.0
PIG	- version 0.13.0
HIVE	- version 1.2.1
MySQL	- version 5.6.33
SQOOP	- version 1.4.6

PROJECT DESCRIPTION

The H1B is an employment-based, non-immigrant visa category for temporary foreign workers in the United States. For a foreign national to apply for H1B visa, an US employer must offer a job and petition for H1B visa with the US immigration department. This is the most common visa status applied for and held by international students once they complete college/ higher education (Masters, Ph.D.) and work in a full-time position.

<u>The dataset description is as follows:</u>

<u>The columns in the dataset include:</u>

- CASE_STATUS:
- Status associated with the last significant event or decision. Valid values include “Certified”, “Certified-Withdrawn,” Denied,” and “Withdrawn”.

  - Certified: Employer filed the LCA, which was approved by DOL
  - Certified Withdrawn: LCA was approved but later withdrawn by employer
  - Withdrawn: LCA was withdrawn by employer before approval
  - Denied: LCA was denied by DOL

- EMPLOYER_NAME: Name of employer submitting labor condition application.

- SOC_NAME: the Occupational name associated with the SOC_CODE.

- SOC_CODE is the occupational code associated with the job being requested for temporary labor condition, as classified by the Standard Occupational Classification (SOC) System.

- JOB_TITLE: Title of the job

- FULL_TIME_POSITION: Y = Full Time Position; N = Part Time Position

- PREVAILING_WAGE: Prevailing Wage for the job being requested for temporary labor condition. The wage is listed at annual scale in USD. The prevailing wage for a job position is defined as the average wage paid to similarly employed workers in the requested occupation in the area of intended employment. The prevailing wage is based on the employer’s minimum requirements for the position.

- YEAR: Year in which the H1B visa petition was filed.

- WORKSITE: City and State information of the foreign worker’s intended area of employment

- lon: longitude of the Worksite

- lat: latitude of the Worksite

PROJECT OUTLINE

Title	Big Data Analysis in Hadoop on H1B_Visa Data
Inputs	H1B Visa Data
Data Elements	Employee name, soc name, case status, worksite, job title, year, wages ,full time position and soc code.
Analysis Relevance	Employment, Migration
Purpose	To analyze the effect of H1B_Visa on employment, wages, demand, new job positions and migration.
Methodology	Agile

1 a) Is the number of petitions with Data Engineer job title increasing over time? Analyze average percentage increase over year

Expected output
Input: - all records
Output: - year – percentage increase

Technology used: - MapReduce

steps involved
Step 1:- filter DATA ENGINEER record
Step 2:- count DATA ENGINEER for each year
Step 3:- find average growth: ((next_year_count - current_year_count) / current_year_count) * 100;
Step 4:- analyze every year average increasing record

Mapper

Input key-value:	key - offset key
	value - all records
Process:	filter 'DATA ENGINEER' record from value
Output key-value:	key – year
	value – 1

Reducer

Input key-value:	key - year
	value – 1
Process:	count record for each year
Output key-value:	key – year
	value – total number of petition for each year

Cleanup

avggrowth= (nextYear-currentYear)*100/currentYear);

Observation
There is increase in data engineer's job, except for 2014-2015 where there is slight dip in percentage but rebounding back in 2015-2016 with average percentage increase of 99%.
Data Engineer positions have observed an exponential growth in the last 6

years.

b) Find top 5 job titles that are having highest average growth in applications. [ALL]

Expected output
Input: - all records
Output: - year – percentage increase

technology used:- MapReduce

steps involved
Step 1:- count number of job title for each year
Step 2:- find average growth: ((next_year_count - current_year_count) / current_year_count) * 100;
Step 3:- analyze every year average increasing record

Job1:-

Mapper

Input key-value:	key - offset key value - all record
Output key-value:	key – job_title value – year

Reducer

Input key-value:	key - job_title value – year
Process:	count record for each year ((next_year_count - current_year_count) / current_year_count) * 100;
Output key-value:	key – percentage increase value value – job_title

Job2:- For Sorting

Mapper

Input key-value:	key - offset key value – percentage increase value + job_title

Output key-value:	key – percentage increase value
	value – job_title

Reducer

Input key-value:	key – percentage increase value
	value – job_title
Process:	print only 5 value using counter
Output key-value:	key – job_title
	value – percentage increase value

Observation:
BUSINESS ANALYST 2, SENIOR SYSTEMS ANALYST JC60 , PROGRAMMER/ DEVELOPER, BUSINESS SYSTEMS ANALYST 2, SOFTWARE DEVELOPER 2 are the top 5 job_title

2 a) Which part of the US has the most Data Engineer jobs for each year?

Expected output
Input: - all records
Output: - year – city/state – total jobs

Technology used: - hive, pig
Here, I have stored filtered data of 'DATA ENGINEER' into directory, and then used Pig command to run further query.
Filtering and running only pig command took more time i.e. 4 min 32 sec
While filtering and storing data in different directory using hive and processing pig commands on it reduced lot of execution time. Time taken- 1 min 06 sec
Reason for such big gap in execution timing is due to amount of query on which pig is processing.

In first case, pig processed number of records = 3002445
In second case pig processed number of records = 1543

Steps involved

Step: - 1. Filter records with job_title='data engineer' and case_status='certified' [HIVE]
Step: - 2. Count number of job for each worksite for each year [PIG]
Step: - 3. Find maximum number of job for each year
Step: - 4. Analyze final output

Observation
SEATTLE, WASHINGTON has most number of data engineer jobs for each year in US.

b) Find top 5 locations in the US who have got certified visa for each year. [Certified]

Expected output
Input: - all records
Output: - year – worksite – total count

Technology used: - pig

Following question we have to group by on two keys, i.e. year and worksite
As well as to display top 5 records for each year.
The bag feature of pig provides a good solution to transfer data from one bag to other easily.
In these case, we are using LIMIT to get top 5 records from bag that generates ordered record on total count which is further flatten and whole output is send to one bag(works like sub query, bag inside bag).

Steps involved
STEP 1:- read all record into first bag
STEP 2:- take only those column that are needed
STEP 3:- filter the record and keep record having case_status as 'CERTIFIED'
STEP 4:- group the record on year and worksite
STEP 5:- count number of jobs at each worksite for each year
STEP 6:- display top 5 records for each year

Observation
(New York, New York), (Houston, Texas), (Chicago, Illinois), (San Jose,California) and (Atlanta, Georgia) have consistently been top 5 locations in US having certified visa. With (New york, New York) topping the list every year.

3) Which industry (SOC_NAME) has the most number of Data Scientist positions?

Expected output
Input: - all records
Output: - soc_name   total position

Technology used: - hive
Here we have to find records with case_status 'CERTIFIED'. To improve query performance in hive, I created another table name h1b_partitioned. This table is partitioned on case_status column and bucketed on year. Running query on partitioned and bucketed table reduced the time required for execution.

Time Taken
Running query on normal table (h1b_final):96.918 sec
Running query on portioned (h1b_partitioned): 49.666   seconds

Steps involved
STEP 1:- convert data to uppercase, since soc_name data is in both upper and lower case.
STEP 2:- keep data where soc_name= 'DATA SCIENTIST' and case_status= 'CERTIFIED' and filter out rest records
STEP 3:- count number of data scientist in each soc_name by group by on soc_name.
STEP 4:- order by data in descending order to get top most soc_name with certified data scientist
STEP 5:- display only top record.

Observation
Statisticians have most number of data scientist position with total count of 705.

4) Which top 5 employers file the most petitions each year? - Case Status – ALL

Expected output
Input: - all records
Output: - emp_name   total_petition

Technology used: - pig
Following question we have to group by on two keys, i.e. year and emp_name
As well as to display top 5 employer records for each year.
The bag feature of pig provides a good solution to transfer data from one bag to other easily.
In these case, we are using LIMIT to get top 5 records from bag that generates ordered record on total count which is further flatten and whole output is send to one bag(works like sub query, bag inside bag).

Steps involved
STEP 1:- insert all record into bag
STEP 2:- insert only those column that are need (year, emp_name) and filter out rest
STEP 3:- group the record using two key years, emp_name
STEP 4:- count the number of record of emp_name for each year. This will provide count of   all record
STEP 5:- to get top 10 records for each year group by output of above record on year.
STEP 6:- for each year order the record on total number petition in descending order and then display top 10 records from ordered set.

Observation
For 2011 TATA CONSULTANCY SERVICES LIMITED was top employer. Since 2012 INFOSYS LIMITED has been consistent top employer with most petitions.
The Top 5 list is dominated by the Indian IT companies.

5) Find the most popular top 10 job positions for H1B visa applications for each year?

a) For all the applications

Expected output
Input: - all records
Output: - job_title total job

Technology used: - pig
Following question we have to group by on two keys, i.e. year and job_title
As well as to display top 10 records for each year.
The bag feature of pig provides a good solution to transfer data from one bag to other easily.
In these case, we are using LIMIT to get top 10 records from bag that generates ordered record on total count which is further flatten and whole output is send to one bag(works like sub query, bag inside bag).

Steps involved
STEP 1:- insert all record into bag
STEP 2:- insert only those column that are need (year, job_title) and filter out rest
STEP 3:- group the record using two key years, job_title
STEP 4:- count the number of record of job_title for each year. This will provide count of all record
STEP 5:- to get top 10 records for each year group by output of above record on year.
STEP 6:- for each year order the record on total number petition in descending order and then display top 10 records from ordered set.

Observation
Programmer Analyst job is the most popular job.

b) Find the most popular top 10 job positions for H1B visa applications for each year? For only certified applications.

Expected output
Input: - all records
Output: - job_title total job

Technology used: - hive
This question needed case_status as 'CERTIFIED'. To improve query performance, we will process this query on partitioned and bucketed table name h1b_partitioned. This table is partitioned on case_status column and bucketed on year. Running query on following table reduced the time required for execution time.

Time taken
Original table -- 67.109 sec
Partitioned and bucketed table – 47.66 sec

Steps involved
STEP 1:- Ask the user to enter year and pass the parameter to query
STEP 2:- filter the record having case_status as certified and year entered by user
STEP 3:- group the record using two key years, job_title
STEP 4:- count the number of record of job_title for each year by grouping on year and job_title
STEP 5:- order the record on total number of job_title in descending order
STEP 6:- display top 10 records for that particular year

Observation
Most of the top 10 job for each year are dominated by jobs from computer related field. Programmar Analyst topped the list.

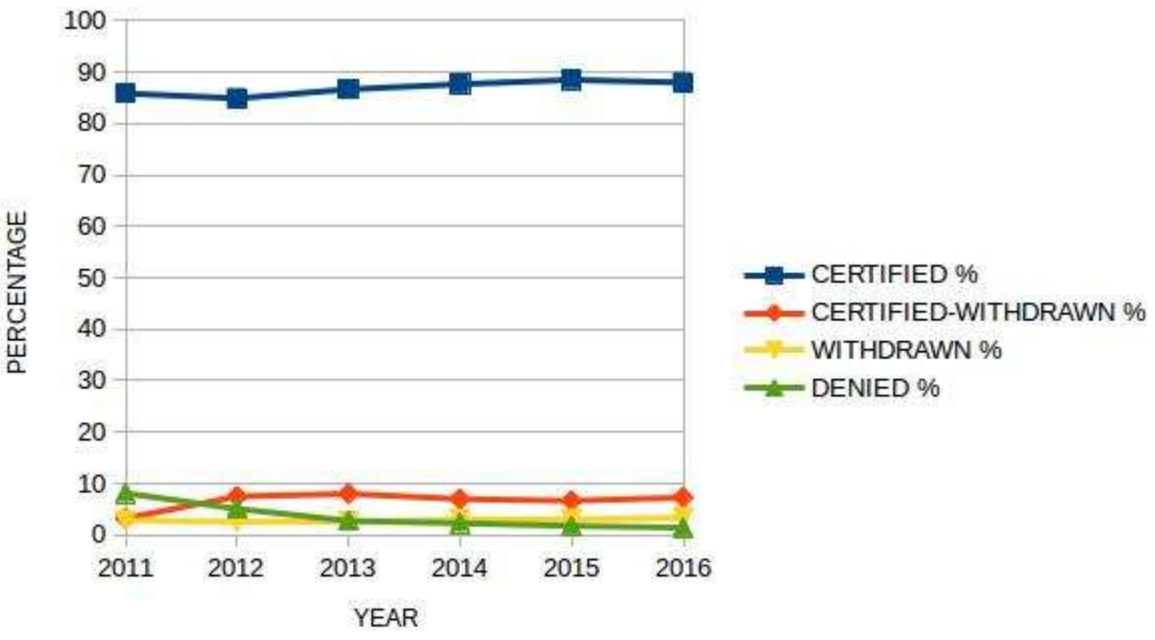
6) Find the percentage and the count of each case status on total applications for each year. Create a line graph depicting the pattern of all the cases over the period of time.

Expected output
Input: - all records
Output: - year case_status total

Technology used: - hive
In this question we just have to find total count and percentage of each case status for every year. It doesn't require any complex analysis; it can easily be done using simple hive query and sub query.

Steps involved
Step 1:- get year and case_status record
Step 2:- calculate count and percentage for each case_status for each year
Step 3:- display the data in order of year
Step 4:- create line graph.

Observation



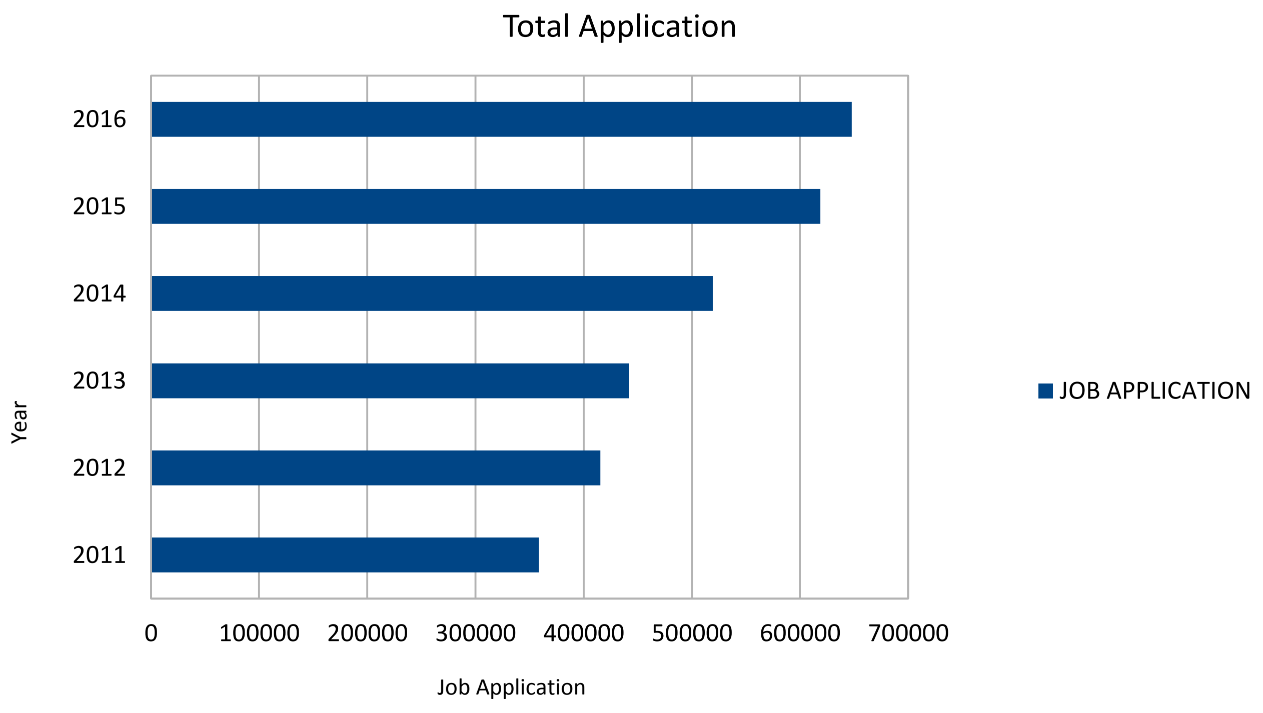
7) Create a bar graph to depict the number of applications for each year [All]

Expected output
Input: - all records
Output: - year total_application

Technology used: - hive
In this question we just have to find total count of application for every year. It doesn't require any complex analysis; it can easily be done using simple hive query.

Steps involved
STEP 1:- count the number of application for each year by grouping on year
STEP 2:- order the record on yearly basis.
STEP 3:- display the record on bar graph

Observation
Job applications are continuously increasing for each year.



8) find the average Prevailing Wage for each Job for each Year (take part time and full time separate). Arrange the output in descending order - [Certified and Certified Withdrawn.]

The wage received by an employee might depend on whether the position is a Full-Time position or a Part-Time Position. Thus we will find out wages separately for Full-Time position and Part-Time Position

Expected output
Input: - all records
Output: - year case_status total

Technology used: - hive

Steps involved

STEP 1:- filter record to get certified and certified-withdrawn
STEP 2:- ask user for year and full_time_position
STEP 3:- find average prevailing wage for each job for particular year and full_time_position using AVG() funtion.

Observation
Expectedly, the wage for Full time positions is higher than for part-time positions.

9) Which are the employers along with the number of petitions who have the success rate more than 70% in petitions? (Total petitions filed 1000 OR more than 1000)?

Expected output
Input: - all records
Output: - emp_name total successrate

Technology used: - mapreduce

Steps involved
STEP 1:- columns needed employer_name,case_status
STEP 2:- count total number of petition of each employer
STEP 3:- count number of petition with case_status as CERTIFIED and CERTIFIED-WITHDRAWN
STEP 4:- use only those records with total >= 1000
STEP 5:- calculate SUCCESS RATE % = (Certified + Certified Withdrawn)/Total x 100
STEP 6:- display only those records successrate >70

Mapper

Input key-value:	key - offset key
	Value - all records
Output key-value:	key – emp_name
	value – case_status, 1

Reducer

Input key-value:	key – emp_name
	value – case_status, 1
Process:	find total case_status
	Find total number of certified case_status
	Find total number of certified-withdrawn case_status
	Check if sum is greater than or equal to 1000
	Then
	Calculate SUCCESS RATE%=(Certified+Certified Withdrawn)/Total x 100
	Display the records having success rate% greater than 70
Output key-value:	key – emp_name
	value – total case_status, successrate %

Observation

There are total 42 employees with success rate is more than 70%.

10) Which are the job positions along with the number of petitions which have the success rate more than 70% in petitions (total petitions filed 1000 OR more than 1000)?

Expected output
Input: - all records
Output: - jobtitle total successrate

Technology used: - mapreduce

Steps involved
STEP 1:- columns needed job_title,case_status
STEP 2:- count total number of petition for each job_title
STEP 3:- count number of petition with case_status as CERTIFIED and CERTIFIED-WITHDRAWN
STEP 4:- use only those records with total >= 1000
STEP 5:- calculate SUCCESS RATE % = (Certified + Certified Withdrawn)/Total x 100
STEP 6:- display only those records successrate >70

Mapper

Input key-value:	key - offset key value - all records
Output key-value:	key – job_title value – case_status, 1

Reducer

Input key-value:	key – job_title value – case_status, 1
Process:	find total case_status Find total number of certified case_status Find total number of certified-withdrawn case_status Check if sum is greater than or equal to 1000 Then Calculate SUCCESS RATE%=(Certified+Certified Withdrawn)/Total x 100 Display the records having success rate% greater than 70
Output key-value:	key – job_title Value – total case_status, successrate %

Observation

There are total 309 job_title with success rate is more than 70%.

11) Export result for question no 10 to MySql database.

Technology used: - sqoop

Steps involved

STEP 1:- create a database in MySQL

STEP 2:- create a table in that particular database. Schema of table should be same as input data.

STEP 3:- run Sqoop export command to export data stored on HDFS to MySQL table.

Observation

We can transfer data from HDFS to MySQL database.

CONCLUSION

Following is the conclusion that we can draw based on the task performed by us:

MapReduce code written in java makes the complex analysis quite easy. Codes required to be written to collect user inputs and performing complex join operations are handled efficiently using this approach.

Pig is very efficient when we have to send output of one job to another, bag feature of pig is helpful in such cases which is not possible with hive and difficult and lengthy process using MapReduce.

Hive feature of partitioned and bucketing helps in reducing execution time if implemented correctly and it also provides the ability to analyze data by writing SQL like queries called HiveQL.

WEBOGRAPHY

https://intellipaat.com/tutorial/hadoop-tutorial/

https://data-flair.training/blogs/hadoop-hdfs-architecture/

https://www.dezyre.com/hadoop-tutorial/big-data-hadoop-tutorial

https://hortonworks.com/tutorial/beginners-guide-to-apache-pig/

http://searchdatamanagement.techtarget.com/definition/Apache-Hive