International Journal of Computer Science and Engineering Research and Development (IJCSERD)

ISSN Print: 2248-9363
ISSN Online: 2248-9371

Volume 1 Issue 2, June-December (2015), pp 01-08

IJCSERD

© PRJ PUBLICATION

ADVOCACY IN IMPROVING FOOD SECURITY USING PREDICTIVE ANALYSIS

Dr.R.Renuga
UG HOD, CIT Sandwich Polytechnic College
Civil Aerodrome Post, Coimbatore-641014.

ABSTRACT

Food security is, “when at all times, all people have access to sufficient, safe, affordable and nutritious food to maintain a healthy life”. Increase in agricultural productivity is one of the solution for food security issues. This should be based on integrating inputs and outputs-the supply of high yielding varieties of seeds, fertilizers, and irrigation, supported by credit alongside remunerative output prices. A “green revolution" is essential to stimulate food production in India. Farmers necessarily require a timely advice to predict the future crop productivity and an analysis is to be made in order to help them to maximize their crop production. In this proposed work, Hadoop, an open source framework is used to store and process the crop related data sets in a Distributed Environment .Hive is used to develop SQL type scripts to facilitate the analysis of datasets and predict patterns by facilitating map reduce operations. This approach will aid in developing innovative proposals to enhance production rate and prevent food scarcity.

Keywords-Mapper,Reducer,Slave node,Master Node,Job tracker,Task Tracker.

1. INTRODUCTION

The concept of food security is defined as including both physical and economic access to food that meets people's dietary needs as well as their food preferences. India is second largest country in the world in the manner of population. Food security happens when all people at all times have access to enough food that should affordable, safe and healthy, meets specific dietary needs, and produced in ways that are environmentally sound. Recent study says that most of Indian peoples are struggling with the bread and butter due to the continuously increasing prices of food grain, vegetables, pulses and other cereals.

Food security is a complex sustainable development issue, linked to health through malnutrition, but also to sustainable economic development, environment, and trade. The issue of food security, especially in a developing nation like India, raises the twin problems of uncertain food production and unequal food distribution. The impact of unequal food distribution can have adverse effects on the rural and urban population living below the poverty

line. Food insecurity is not only economic problem but also problem of non-humanity approach in India.

It may be possible to make food security in India in good manner. If proper planning of food grain production and fair practices in food market is done. The root causes of food insecurity include, poverty, corruption, national policies that do not promote equal access to food for all, environmental degradation, barriers to trade, insufficient agricultural development, population growth, low levels of education, social and gender inequality, poor health status, cultural insensitivity, and natural disasters.

Increase in agricultural productivity is one of the solution for food security issues. This should be based on integrating inputs and outputs-the supply of high yielding varieties of seeds, fertilizers, and irrigation, supported by credit alongside remunerative output prices.
The main motive of the project is to analyze the various factors influencing rice and wheat cultivation over the past five years as mentioned above, address the twin problems namely, uncertain food production and Unequal food distribution by predicting its future cultivation and consumption, thereby giving a refined report containing useful methodologies that can be implemented to prevent food scarcity

Rice and Wheat are the highly essential staple food crops in India that forms the concrete base for livelihood. The various factors influencing rice and wheat cultivation are

- Rainfall
- Irrigation
- Production and Consumption rate
- Monsoon
- Economic status of farmers

In this proposed work, Hadoop, an open source framework is used to store and process the crop related data sets in a Distributed Environment. Hive is used to develop SQL type scripts to facilitate the analysis of datasets and predict patterns by facilitating map reduce operations. This approach will aid in developing innovative proposals to enhance production rate and prevent food scarcity.

2. INTRODUCTION TO HIVE:

Today's era as we know is the Data Era. We see data all over places right from our cell phones to the high end enterprise servers, from social networking sites to search engines and from text messages to video calls the data is growing and with the need to process that in cost and time effective manner is increasing.Using traditional data management systems, it is difficult to process Big Data. Therefore, the Apache Software Foundation introduced a framework called Hadoop to solve Big Data management and processing challenges.Hadoop is an open-source framework to store and process Big Data in a distributed environment. It contains two modules, one is MapReduce and another is Hadoop Distributed File System (HDFS). Hive is a data warehousing tool based on Hadoop. As we know Hadoop provides massive scale out on distributed infrastructure with high degree of fault tolerance for data storage and processing. Hadoop uses MapReduce algorithm to process huge amount of data with minimal cost as it does not require high end machines to process such amount of data. Hive processor converts most of its queries into a Map Reduce job which runs on Hadoop cluster. Hive is designed for easy and effective data aggregation, ad-hoc querying and analysis of huge volumes of data. Even though Hive gives SQL dialect it does not give SQL like latency as it ultimately runs Map Reduce programs underneath. As we all know, Map Reduce framework is built for batch

processing jobs it has high latency, even the fastest hive query would take several minutes to get executed on relatively smaller set of data in few megabytes. Hive is best suited for Data Warehousing Applications where data is stored, mined and reporting is done based on processing. As most Data Warehousing applications are based on relational database models, Hive bridges the gap between these applications and Hadoop.

3. SYSTEM ARCHITECTURE

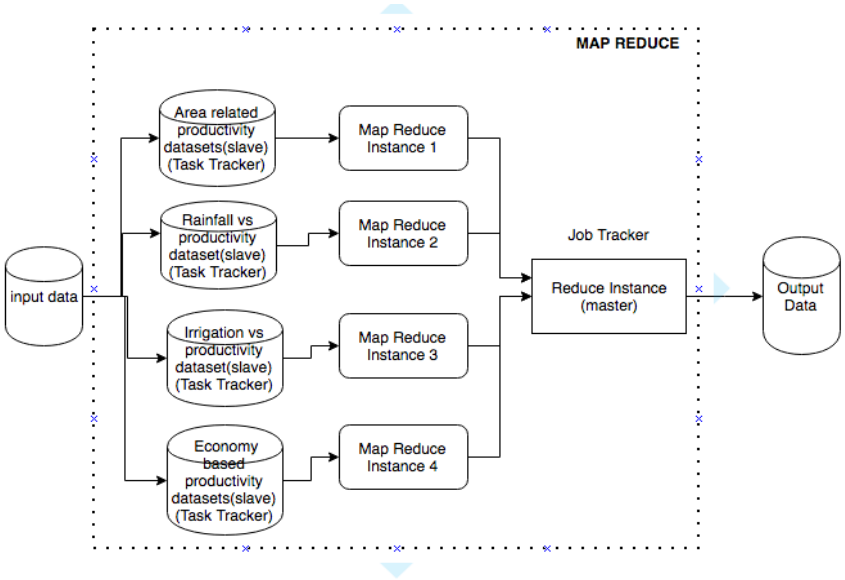

Figure 1 System Architecture

The complete architecture of the proposed system is depicted in Fig.1.One of the systems is configured as the Master node.Four other systems are configured as the Slave nodes.These Master and Slave nodes form the heart of the distributed system that are connected over a network to enable communication between them.The Master node consists of both the Name node and Data nodes while the Slave node consists of only the Data nodes(HDFS Layer).The Master node also acts as the slave while performing the Map Reduce operations.The Master Node consists of both the Job Tracker and the TaskTracker but the Slave node consists of only the TaskTracker(Map Reduce Layer).The client triggers different types of queries(in terms of Hive queries) to the Master node that will spot the locality of the essential data in the respective Data nodes using the metadata from the Name node.Now, the map reduce operations are done using the Job and the Task Trackers and finally the results are collaborated and forwarded to the client.In short, the layers of the proposed architecture could be classified as follows(Bottom-up Architecture):

Layer 1:HDFS Layer

Layer 2:MapReduce Layer

Layer 2.1:Mapping

Layer 2.2:Reducing

Layer 3:Result transmission Layer

4. SYSTEM DESIGN

4.1. Dataset Distribution

The operations of the different components mentioned in the System Architecture is explained henceforth.The Data node is the storehouse of all the datasets.Since the proposed system deals with big data we use four different Slave machines(Data nodes) into which the datasets are split and stored.For instance,the productivity related to area is stored in data node 1,followed by the rainfall in data node 2, irrigation in data node 3 and economy based productivity in data node 4.The Name node that is present in the Master will contain the metadata about the Data nodes,(i.e) address of all the datasets indicating in which Data node the required dataset is available.JobTracker receives the requests for MapReduce execution from the client and talks to the NameNode to determine the location of the data.Now, the job tracker will know when to contact area dataset,rainfall dataset,irrigation dataset and economy dataset.Each of these datasets will have a task tracker.Further,JobTracker finds the best TaskTracker nodes to execute tasks based on the data locality (proximity of the data). Finally,JobTracker monitors the individual TaskTrackers.It collects the results from area,rainfall,irrigation and economy datasets and then submits back the overall status of the job back to the client.TaskTracker runs on DataNode.Mapper and Reducer tasks are executed on DataNodes administered by TaskTrackers.Further, the job tracker sends the results from each of them as input to the reduce task handled by the TaskTracker present in the master node.TaskTracker will be in constant communication with the JobTracker signalling the progress of the task in execution.Finally, the actual result for the query,(for example the reason for the highest productivity) is obtained by the job tracker and passed on to the client.JobTracker process is critical to the Hadoop cluster in terms of MapReduce execution.When the JobTracker is down, HDFS will still be functional but the MapReduce execution cannot be started and the existing MapReduce jobs will be halted.TaskTracker failure is not considered fatal. When a TaskTracker becomes unresponsive, JobTracker will assign the task executed by the TaskTracker to another node.

Sub-division	Monthly rainfall							
	January		February		March		April	
	2007	2008	2007	2008	2007	2008	2007	2008
1	2	3	4	5	6	7	8	9
1.Andaman and Nicobar Islands	4.8	16.6	1.6	55.7	13.5	116.3	55.5	203.9
2.Arunachal Pradesh	17.0	72.9	77.9	33.4	52.9	118.5	274.1	113.0
3.Assam and Meghalaya	1.2	28.8	75.6	13.9	24.0	116.0	236.4	165.6
4.Nagaland,Manipur, Mizoram, and Tripura	0.4	30.7	60.0	10.8	25.3	42.2	171.6	32.0
5.Sub-Himalayan,West Bengal & Sikkim	0.6	27.2	90.8	8.4	37.6	67.8	144.9	140.3
6.Gangetic West Bengal	3.2	52.9	56.8	12.3	38.2	12.2	36.2	46.8
7.Orissa	0.9	33.0	34.3	14.9	8.9	24.9	22.7	34.4
8.Jharkhand	0.0	10.1	74.4	4.6	16.9	23.2	15.7	17.3
9.Bihar	0.0	38.0	16.4	5.8	18.7	1.7	10.4	16.4
10.Uttar Pradesh East	0.1	2.0	50.9	3.4	25.5	0.0	2.8	5.2
11.Uttar Pradesh West	0.6	0.2	63.1	0.1	35.7	0.0	0.9	6.1
12.Uttaranchal	4.9	15.4	131.5	10.2	104.5	5.5	18.2	31.6
13.Haryana,Chandigarh & Delhi	1.8	2.7	87.9	2.5	49.3	0.0	1.9	16.4
14.Punjab	0.9	14.7	71.4	6.6	57.4	0.0	6.3	31.8

Table –I Sample Plan-wise Area, Production and Productivity

Five Year Plan	Area		Production		Productivity	
	M. ha	% increase over previous Plan year	M. tones	% increase over previous Plan year	Kg/ha	%increase over previous Plan year
First Plan						
(1951-52 to 1955-56)	30.68	-	25.03	-	816	-
Second Plan						
(1956-57 to1960 -61)	33.14	8.0	30.34	21.2	915	12.1
Third Plan						
(1961-62 to1965 -66)	35.62	7.5	35.15	15.9	987	7.9
Annual Plan						
(1966 -67)	35.25	-	30.44	-	863	-
(1967 -68)	36.44	-	37.61	-	1032	-
(1968 -69)	36.97	-	39.76	-	1076	-
Fourth Plan						
1969 -70 to 1973-74)	37.60	5.6	41.80	18.9	1112	12.7
Fifth Plan						
(1974 -75 to1978 -79)	39.33	4.6	47.34	13.3	1204	8.3
Annual Plan						
(1979 -80)	39.42	-	42.33	-	1074	-
Sixth Plan						
(1980-81 to1984 -85)	40.30	2.5	54.49	15.1	1352	12.3
Seventh Plan						
(1985-86 to1989-90)	41.00	1.7	65.06	19.4	1587	17.4
Annual Plan						
(1990 -91)	42.69	-	74.29	-	1740	-
(1991 -92)	42.65	-	74.68	-	1751	-
Eighth Plan						
(1992-93 to1996 -97)	42.68	4.1	78.74	21.0	1845	16.3
Ninth Plan						
(1997-98 to 2001-02)	44.60	4.5	87.32	10.9	1958	6.1
Tenth Plan 2002-03 to2006-07	42.63	-4.4	85.73	-1.8	2011	2.7

Table -II Sample rainfall statistics

Year/State/ Union Territory	Rice 2008-09 ('000 hectare)	Rice 2009-10 ('000 hectare)	Wheat 2008-09 ('000 hectare)	Wheat 2009-10 ('000 hectare)
Andhra Pradesh	4249	3354	12	9
Arunachal Pradesh *	54	51	NA	1
Assam *	249	179	NA	NA
Bihar*	1981	1821	1938	2021
Chhattisgarh	1284	1218	67	74
Goa	16	16	NA	NA
Gujarat *	473	430	980	788
Haryana	1199	1205	2438	2474
Himachal Pradesh *	50	50	73	71
Jammu & Kashmir*	233	237	81	87
Jharkhand *	26	22	50	44
Karnataka	1121	1112	144	161

Table –III Sample production rates

4.2. Mapping

Mapping operation could be explained using an example.The dataset that relates rainfall and production rate has been loaded into the DataNode 1 followed by that of irrigation and production rate into DataNode 2, Economic status and production rate into DataNode 3 ,Exports and production rate into DataNode 4.These datasets include details of all the states and the states could be repeated in each of them. For instance, if the aim is to find the reason behind the highest production rate for a particular state,then each of the mapper will perform the querying on its own dataset and portray the highest production rate from its dataset for the requested state.Hence, at the end of all mapping operations, each of the mapper will give out a result indicating the highest production rate in its own dataset.There ends the process of mapping and reducing commences.

4.3. Reducing

Reducing operation will take up the output of each of the mapper as its input and query for the highest production rate from them and produce the final result.This final result will give us the reason for the highest production rate in a particular state,thereby achieving our target.

4.4. Advantage and Automation of Map Reduce Operation

Since the datasets are distributed to different data nodes, querying could be done parallely on different data nodes and finally the results are clubbed together. This satisfies the purpose of using the Hadoop Distributed File System by enhancing the speed of the process.Instead of coding meticulously to achieve map reduce operations, Hive has been used that will automate the map reduce operations that could be best explained using optimization of joins. The Common join operation is executed as follows.When two large data tables need to do a join, there will be two different Mappers to sort these tables based on the join key and emit an intermediate file, and the Reducer will take the intermediate file as input file and do the real join work. This join mechanism (referred to as Common Join) is perfect with two large data size. But if one the join tables is small enough to fit into the Mapper's memory, then there is no need to launch the Reducer. Actually, the Reducer stage is very expensive for the performance because the Map/Reduce framework needs to sort and merge the intermediate files. So the basic idea of Map Join is to hold the data of small table in Mapper's memory and do the join work in Map stage, which saves the Reduce stage.The conversion of Common join to Map join that will enable the automation of Map Reduce operation is explained as follows.For the Map Join, the query processor should know which input table the big table is. The other input tables will be recognized as the small tables during the execution stage and these tables need to be held in the memory. However, in general, the query processor has no idea of input file size during compiling time because some of the tables may be intermediate tables generated from sub queries. So the query processor can only figure out the input file size during the execution time. Hence, during the compile time, the query processor will generate a Conditional Task, which contains a list of tasks and one of these tasks will be resolved to run during the execution time. It means the tasks in the Conditional Task's list are the candidates and one of them will be chosen to run during the run time. First, the original Common Join Task should be put into the list. Also the query processor will generate a series of Map Join Task by assuming each of the input tables may be the big table. For example, select * from src1 x join src2y on x.key=y.key. Both table src2 and src1 may be the big tables, so it will generate two Map Join Task. One is assuming src1 as the big table and the other assuming src2 as the big table.During the execution stage, the Conditional Task can know exactly the file size of each input table, even the table is a intermediate table. If all the tables are too large to be converted into map join, then just run the Common Join Task as previously. If one of the tables is large and others are small enough to run Map Join, then the Conditional Task will pick the corresponding Map Join Local Task to run. By this mechanism, it can convert the Common Join into Map Join automatically and dynamically.

5. IMPLEMENTATION

To implement the proposed system a distributed, multi-node Apache Hadoop cluster backed by the Hadoop Distributed File System (HDFS), running on Ubuntu Linux is set up. Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the

MapReduce computing paradigm. Hadoop's HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware.
To set up a multi-node clustered structure, the individual single-nodes are first configured. The single-nodes are configured containing their corresponding datasets with same settings so that they can be merged to achieve the final multi-node cluster setup. From these nodes, one of the node is configured to be a master (acts also as a slave). Then the machines are connected to the same network by configuring their IP addresses. After the machines are under the same network, the slaves can be accessed by the master using the public SSHkey to access the data available in various nodes and process them in the master node. The master node will run the "master" daemons for each layer: NameNode for the HDFS storage layer, and JobTracker for the MapReduce processing layer. All other machines will run the "slave" daemons: DataNode for the HDFS layer, and TaskTracker for MapReduce processing layer. Basically, the "master" daemons are responsible for coordination and management of the "slave" daemons while the latter will do the actual data storage and data processing work.

Before starting the multi-node cluster, we must format Hadoop's Distributed File System (HDFS) via the NameNode. Starting the cluster is performed in two steps.

- Begin with starting the HDFS daemons: the NameNode daemon is started on master, and DataNode daemons are started on all slaves.
- Start the MapReduce daemons: the JobTracker is started on master, and TaskTracker daemons are started on all slaves.

Then the map reduce operations are performed on the different datasets and predictive analysis is done to improve food security. After the predictions are done the multi-node cluster is stopped by stopping the MapReduce daemons (the JobTracker is stopped on master, and TaskTracker daemons are stopped on all slaves) and the HDFS daemons (the NameNode daemon is stopped on master, and DataNode daemons are stopped on all slaves).

6. RESULT

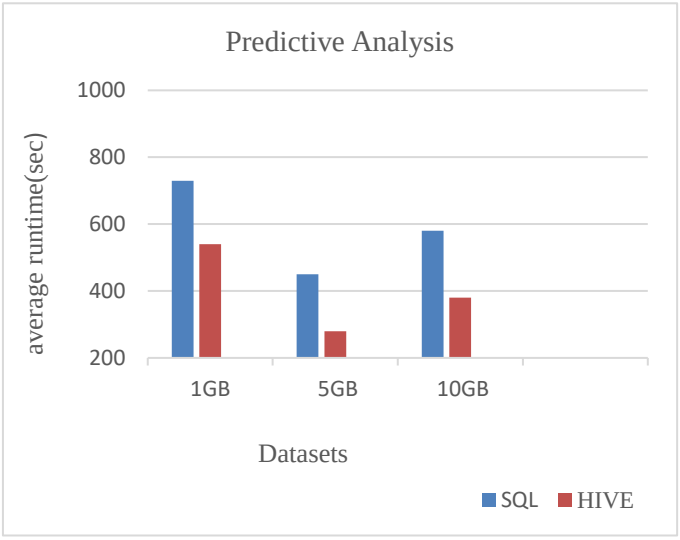

Figure 2 Dataset vs Time

The experimental results shows that this distributed approach is effective and significantly improve the response time. It is shown in Fig.2.

7. CONCLUSION

The proposed system does the predictive analysis only using two important food crops wheat and rice. In future it can be extended to all grains.

REFERENCES

[1] Clark Bradley, Ralph Hollinshead, Scott Kraus, Jason,Lefler, Roshan Taheri"Data Modeling Considerations in Hadoop and Hive", a technical paper from SAS, October 2013 .

[2] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy . "Hive – A Petabyte ScaleData Warehouse Using Hadoop ",IEEE 26th International Conference on Data Engineering (ICDE) pp. 996-1005, Long Beach, CA, 2010

[3] Carletto, Calogero, Alberto Zezza, and Raka Banerjee. "Towards better measurement of household food security: Harmonizing indicators and the role of household surveys." Global food security 2.1 (2013)

[4] Hive wiki athttp://hive.apache.org/.

[5] Running Hadoop on Ubuntu Linux tutorial at http://www.michael.noll/com/tutorials

[6] Hive Map-Reduce Tutorial at https://www.tutorialspoint.com/hive/

[7] Dataset collected from https://data.gov.in