

## **Leveraging Map Reduce With Hadoop for Weather Data Analytics**

Riyaz P.A.<sup>1</sup>, Surekha Mariam Varghese<sup>2</sup>

<sup>1,2.</sup> (Dept. of CSE, Mar Athanasius College of Engineering, Kothamangalam, Kerala, India)

---

**Abstract :** Collecting, storing and processing of huge amounts of climatic data is necessary for accurate prediction of weather. Meteorological departments use different types of sensors such as temperature, humidity etc. to get the values. Number of sensors and volume and velocity of data in each of the sensors makes the data processing time consuming and complex. Leveraging MapReduce with Hadoop to process the massive amount of data. Hadoop is an open framework suitable for large scale data processing. MapReduce programming model helps to process large data sets in parallel, distributed manner. This project aims to build a data analytical engine for high velocity, huge volume temperature data from sensors using MapReduce on Hadoop.

**Keywords –** Bigdata, MapReduce, Hadoop, Weather, Analytics, Temperature

---

### **I. Introduction**

Big Data has become one of the buzzwords in IT during the last couple of years. Initially it was shaped by organizations which had to handle fast growth rates of data like web data, data resulting from scientific or business simulations or other data sources. Some of those companies' business models are fundamentally based on indexing and using this large amount of data. The pressure to handle the growing data amount on the web e.g. lead Google to develop the Google File System [1] and MapReduce [2].

Bigdata and Internet of things are related areas. The sensor data is a kind of Bigdata. The sensor data have high velocity. The large number of sensors generates high volume of data. While the term "Internet of Things" encompasses all kinds of gadgets and devices, many of which are designed to perform a single task well. The main focus with respect to IoT technology is on devices that share information via a connected network.

India is an emerging country. Now most of the cities have become smart. Different sensors employed in smart city can be used to measure weather parameters. Weather forecast department has begun collect and analysis massive amount of data like temperature. They use different sensor values like temperature, humidity to predict the rain fall etc. When the number of sensors increases, the data becomes high volume and the sensor data have high velocity data. There is a need of a scalable analytics tool to process massive amount of data.

The traditional approach of process the data is very slow. Process the sensor data with MapReduce in Hadoop framework which remove the scalability bottleneck. Hadoop is an open framework used for Bigdata analytics. Hadoop's main processing engine is MapReduce, which is currently one of the most popular big-data processing frameworks available. MapReduce is a framework for executing highly parallelizable and distributable algorithms across huge data sets using a large number of commodity computers. Using Mapreduce with Hadoop, the temperature can be analyse without scalability issues. The speed of processing data can increase rapidly when across multi cluster distributed network.

In this paper a new Temperature Data Analytical Engine is proposed, which leverages MapReduce with Hadoop framework of producing results without scalability bottleneck. This paper is organized as follows: in section II, the works related to MapReduce is discussed. Section III describes the Temperature Data Analytical Engine implementation. Section IV describes the analytics of NCDC Data. Finally the section V gives the conclusion for the work.

### **II. Related Works**

#### **2.1 Hadoop**

Hadoop is widely used in big data applications in the industry, e.g., spam filtering, network searching, click-stream analysis, and social recommendation. In addition, considerable academic research is now based on Hadoop. Some representative cases are given below. As declared in June 2012, Yahoo runs Hadoop in 42,000 servers at four data centers to support its products and services, e.g., searching and spam filtering, etc. At present, the biggest Hadoop cluster has 4,000 nodes, but the number of nodes will be increased to 10,000 with the release of Hadoop 2.0. In the same month, Facebook announced that their Hadoop cluster can process 100 PB data, which grew by 0.5 PB per day as in November 2012. Some well-known agencies that use Hadoop to conduct distributed computation are listed in [3].

In addition, many companies provide Hadoop commercial execution and/or support, including Cloudera, IBM, MapR, EMC, and Oracle. According to the Gartner Research, Bigdata Analytics is a trending topic in 2014 [4]. Hadoop is an open framework mostly used for Bigdata Analytics. MapReduce is a programming paradigm associated with the Hadoop.

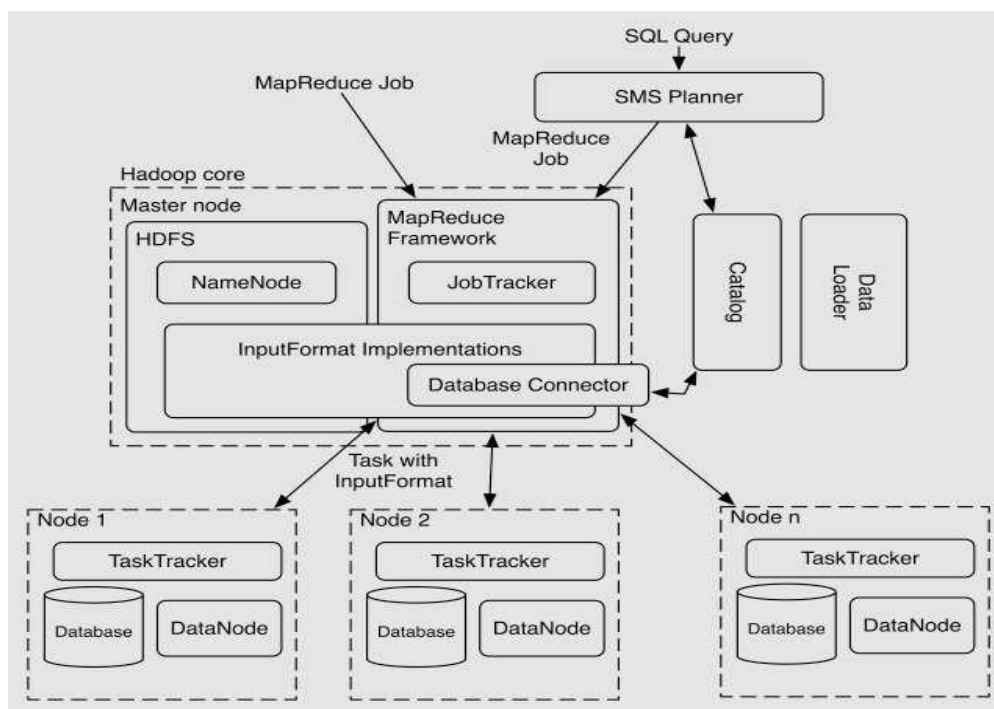


Fig.1: Hadoop Environment

Some of its characteristics of Hadoop are following. It is an open-source system developed by Apache in Java. It is designed to handle very large data sets. It is designed to scale to very large clusters. It is designed to run on commodity hardware. It offers resilience via data replication. It offers automatic failover in the event of a crash. It automatically fragments storage over the cluster. It brings processing to the data. Its supports large volumes of files into the millions. The Hadoop environment as shown in Figure 1

## 2.2 Map Reduce

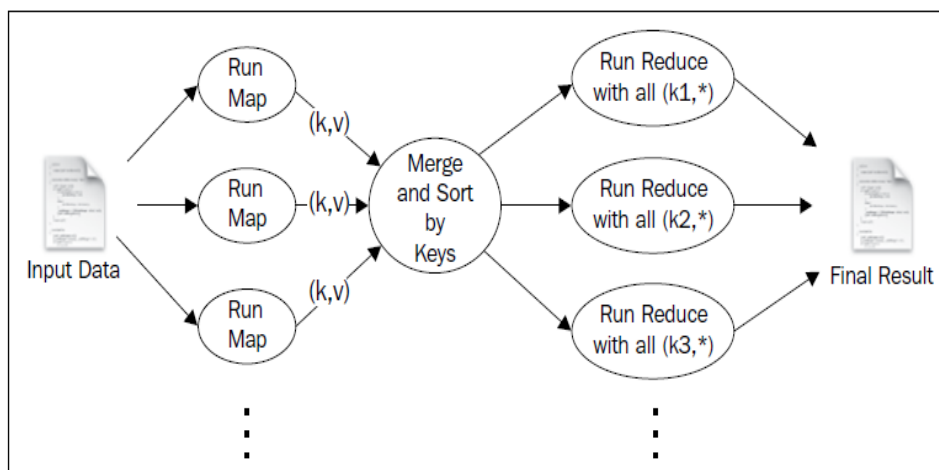
Hadoop using HDFS for data storing and using MapReduce to processing that data. HDFS is Hadoop's implementation of a distributed filesystem. It is designed to hold a large amount of data, and provide access to this data to many clients distributed across a network [5]. MapReduce is an excellent model for distributed computing, introduced by Google in 2004. Each MapReduce job is composed of a certain number of map and reduce tasks. The MapReduce model for serving multiple jobs consists of a processor sharing queue for the Map Tasks and a multi-server queue for the Reduce Tasks [6].

To run a MapReduce job, users should furnish a map function, a reduce function, input data, and an output data location as shown in figure 2. When executed, Hadoop carries out the following steps:

Hadoop breaks the input data into multiple data items by new lines and runs the map function once for each data item, giving the item as the input for the function. When executed, the map function outputs one or more key-value pairs. Hadoop collects all the key-value pairs generated from the map function, sorts them by the key, and groups together the values with the same key.

For each distinct key, Hadoop runs the reduce function once while passing the key and list of values for that key as input. The reduce function may output one or more key-value pairs, and Hadoop writes them to a file as the final result. Hadoop allows the user to configure the job, submit it, control its execution, and query the state. Every job consists of independent tasks, and all the tasks need to have a system slot to run [6]. In Hadoop all scheduling and allocation decisions are made on a task and node slot level for both the map and reduce phases [7].

There are three important scheduling issues in MapReduce such as locality, synchronization and fairness. Locality is defined as the distance between the input data node and task-assigned node. Synchronization is the process of transferring the intermediate output of the map processes to the reduce process



**Fig.2:** MapReduce Framework

As input is also consider as a factor which affects the performance. Fairness finiteness have trade-offs between the locality and dependency between the map and reduce phases. Due to the important issues and many more problems in scheduling of MapReduce, the Scheduling is one of the most critical aspects of MapReduce. There are many algorithms to solve this issue with different techniques and approaches. Some of them get focus to improvement data locality and some of them implements to provide Synchronization processing and many of them have been designed to minimizing the total completion time.

E.Dede et.al [8] proposed MARISSA allows for: Iterative application support: The ability for an application to assess its output and schedule further executions. The ability to run a different executable on each node of the cluster. The ability to run different input datasets on different nodes. The ability for all or a subset of the nodes to run duplicates of the same task, allowing the reduce step to decide which result to select.

Thilina Gunarthane et.al [9] introduce AzureMapReduce, a novel MapReduce runtime built using the Microsoft Azure cloud infrastructure services. AzureMapReduce architecture successfully leverages the high latency, eventually consistent, yet highly scalable Azure infrastructure services to provide an efficient, on demand alternative to traditional MapReduce clusters. Further evaluate the use and performance of MapReduce frameworks, including AzureMapReduce, in cloud environments for scientific applications using sequence assembly and sequence alignment as use cases.

Vinay Sudakaran et.al [10] introduce analyses of surface air temperature and ocean surface temperature changes are carried out by several groups, including the Goddard institute of space studies (GISS) [11] and the National Climatic Data Center based on the data available from a large number of land based weather stations and ship data, which forms an instrumental source of measurement of global climate change.

Uncertainties in the collected data from both land and ocean, with respect to their quality and uniformity, force analysis of both the land based station data and the combined data to estimate the global temperature change. Estimating long term global temperature change has significant advantages over restricting the temperature analysis to regions with dense station coverage, providing a much better ability to identify phenomenon that influence the global climate change, such as increasing atmospheric CO<sub>2</sub> [12]. This has been the primary goal of GISS analysis, and an area with potential to make more efficient through use of MapReduce to improve throughput.

Ekanayake et al. [13] evaluated the Hadoop implementation of MapReduce with High Energy Physics data analysis. The analyses were conducted on a collection of data files produced by high energy physics experiments, which is both data and compute intensive. As an outcome of this porting, it was observed that scientific data analysis that has some form of SPMD (Single- Program Multiple Data) algorithm is more likely to benefit from MapReduce when compared to others. However, the use of iterative algorithms required by many scientific applications were seen as a limitation to the existing MapReduce implementations.

### III. Implementation

The input weather dataset contain the values of temperature, time, place etc. The input weather dataset file on the left of Figure 3 is split into chunks of data. The size of these splits is controlled by the InputSplit method within the FileInputFormat class of the Map Reduce job. The number of splits is influenced by the HDFS block size, and one mapper job is created for each split data chunk. Each split data chunk that is, “a record,” is sent to a Mapper process on a Data Node server. In the proposed system, the Map process creates a series of key-value pairs where the key is the word, for instance, “place” and the value is the temperature. These key-value pairs are then shuffled into lists by key type. The shuffled lists are input to Reduce tasks, which

reduce the dataset volume by the values. The Reduce output is then a simple list of averaged key-value pairs. The Map Reduce framework takes care of all other tasks, like scheduling and resources.

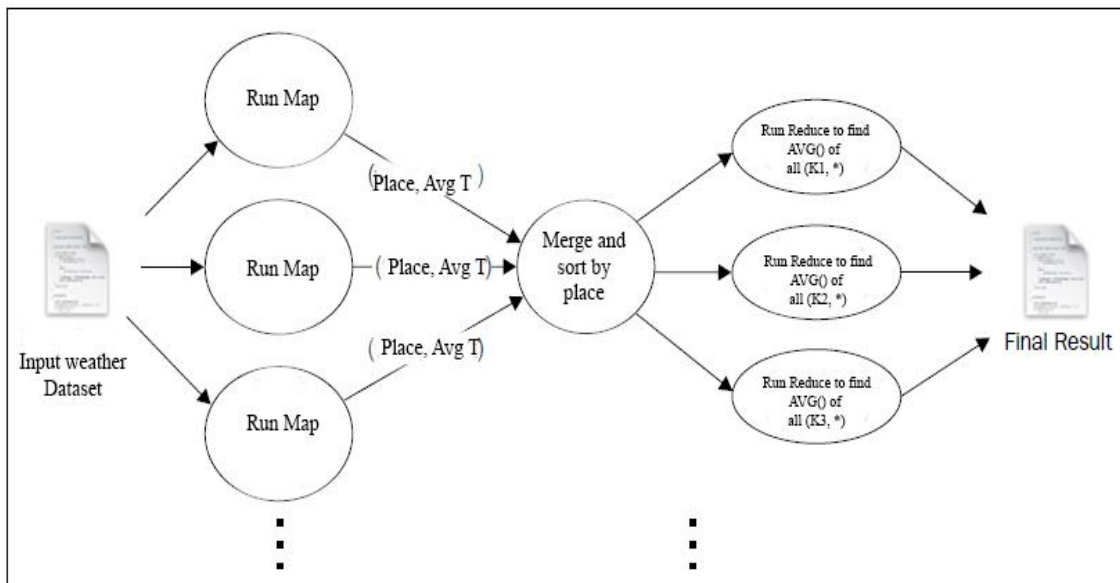


Fig.3: Proposed MapReduce Framework

The proposed MapReduce application is shown in figure 3. The mapper function will find the average temperature and associated with the key as place. The all values are combined ie reduce to form the final result. The average temperature, maximum temperature, minimum temperature etc can be calculated in the reduce phase.

### 3.1 Driver Operation

The driver is what actually sets up the job, submits it, and waits for completion. The driver is driven from a configuration file for ease of use for things like being able to specify the input/output directories. It can also accept Groovy script based mappers and reducers without recompilation.

### 3.2 Mapper Operation

The actual mapping routine was a simple filter so that only variables that matched certain criteria would be sent to the reducer. It was initially assumed that the mapper would act like a distributed search capability and only pull the <key, value> pairs out of the file that matched the criteria. This turned out not to be the case, and the mapping step took a surprisingly long amount of time.

The default Hadoop input file format reader opens the files and starts reading through the files for the <key, value> pairs. Once it finds a <key, value> pair, it reads both the key and all the values to pass that along to the mapper. In the case of the NCDC data, the values can be quite large and consume a large amount of resources to read and go through all <key, value> pairs within a single file.

Currently, there is no concept within Hadoop to allow for a “lazy” loading of the values. As they are accessed, the entire <key, value> must be read. Initially, the mapping operator was used to try to filter out the <key, value> pairs that did not match the criteria. This did not have a significant impact on performance, since the mapper is not actually the part of Hadoop that is reading the data. The mapper is just accepting data from the input file format reader.

Subsequently, modified the default input file format reader that Hadoop uses to read the sequenced files. This routine basically opens a file and performs a simple loop to read every <key, value> within the file. This routine was modified to include an accept function in order to filter the keys prior to actually reading in the values. If the filter matched the desired key, then the values are read into memory and passed to the mapper. If the filter did not match, then the values were skipped. There is some values in the temperature data may be null. In the mapper the null values were filtered.

Based on the position of sensor values, the mapper function assigns the values to the <Key, Value> pairs. Place id can be used as Key to get the entire calculation of a place. Alternatively combination of place and date can be used as Key. The values are temperature in an hour.

### 3.3 Reduce Operation

With the sequencing and mapping complete, the resulting <key, value> pairs that matched the criteria to be analyzed were forwarded to the reducer. While the actual simple averaging operation was straightforward and relatively simple to set up, the reducer turned out to be more complex than expected. Once a <key, value>

object has been created, a comparator is also needed to order the keys. If the data is to be grouped, a group comparator is also needed. In addition, a partitioner must be created in order to handle the partitioning of the data into groups of sorted keys.

With all these components in place, Hadoop takes the <key, value> pairs generated by the mappers and groups and sorts them as specified as shown in figure 4. By default, Hadoop assumes that all values that share a key will be sent to the same reducer. Hence, a single operation over a very large data set will only employ one reducer, i.e., one node. By using partitions, sets of keys to group can be created to pass these grouped keys to different reducers and parallelize the reduction operation. This may result in multiple output files so that an additional combination step may be needed to handle the consolidation of all results.

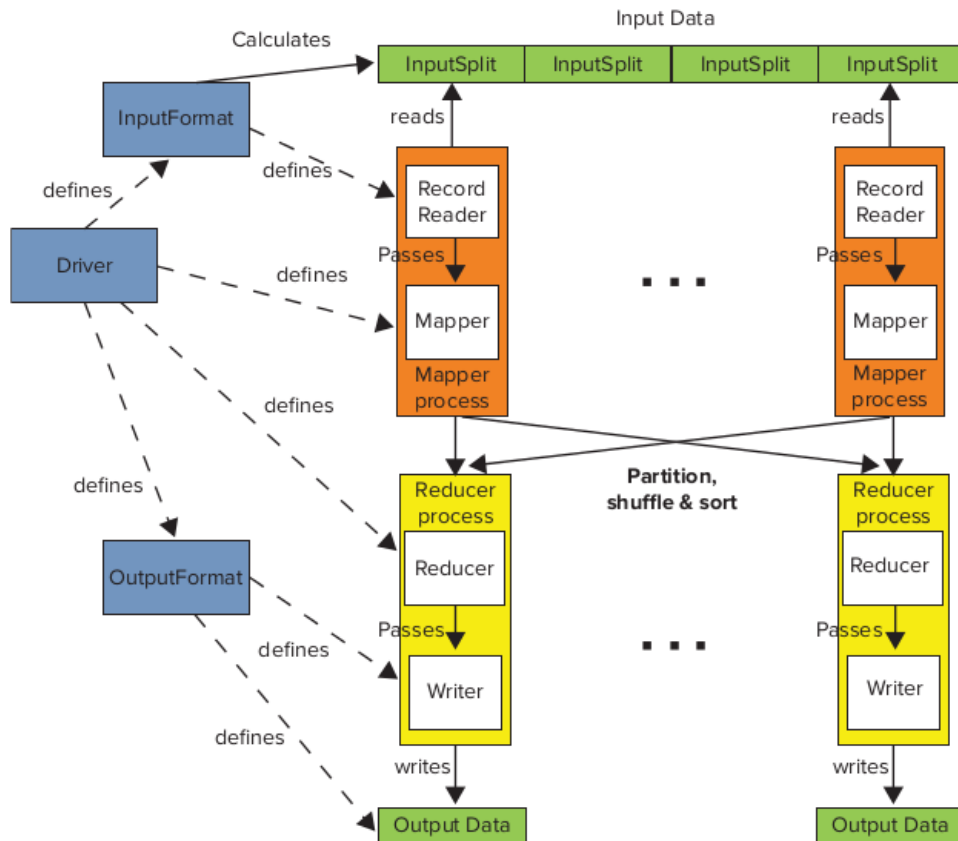


Fig 4 : Inside Mapreduce

### 3.4 Map Reduce Process

MapReduce is a framework for processing highly distributable problems across huge data sets using a large number of computers (nodes). In a “map” operation the head node takes the input, partitions it into smaller sub-problems, and distributes them to data nodes. A data node may do this again in turn, leading to a multi-level tree structure. The data node processes the smaller problem, and passes the answer back to a reducer node to perform the reduction operation. In a “reduce” step, the reducer node then collects the answers to all the sub-problems and combines them in some way to form the output, the answer to the problem it was originally trying to solve. The map and reduce functions of Map-Reduce are both defined with respect to data structured in <key, value> pairs.

The following describes the overall general MapReduce process that is executed for the averaging operation, maximum operation and minimum operation on the NCDC data:

The NCDC files were processed into Hadoop sequence files on the HDFS Head Node. The files were read from the local NCDC directory, sequenced, and written back out to a local disk.

The resulting sequence files were then ingested into the Hadoop file system with the default replica factor of three and, initially, the default block size of 64 MB.

The job containing the actual MapReduce operation was submitted to the Head Node to be run.

Along with the JobTracker, the Head Node schedules and runs the job on the cluster. Hadoop distributes all the mappers across all data nodes that contain the data to be analyzed.

On each data node, the input format reader opens up each sequence file for reading and passes all the <key,value> pairs to the mapping function.

The mapper determines if the key matches the criteria of the given query. If so, the mapper saves the <key, value> pair for delivery back to the reducer. If not, the <key, value> pair is discarded. All keys and values within a file are read and analyzed by the mapper.

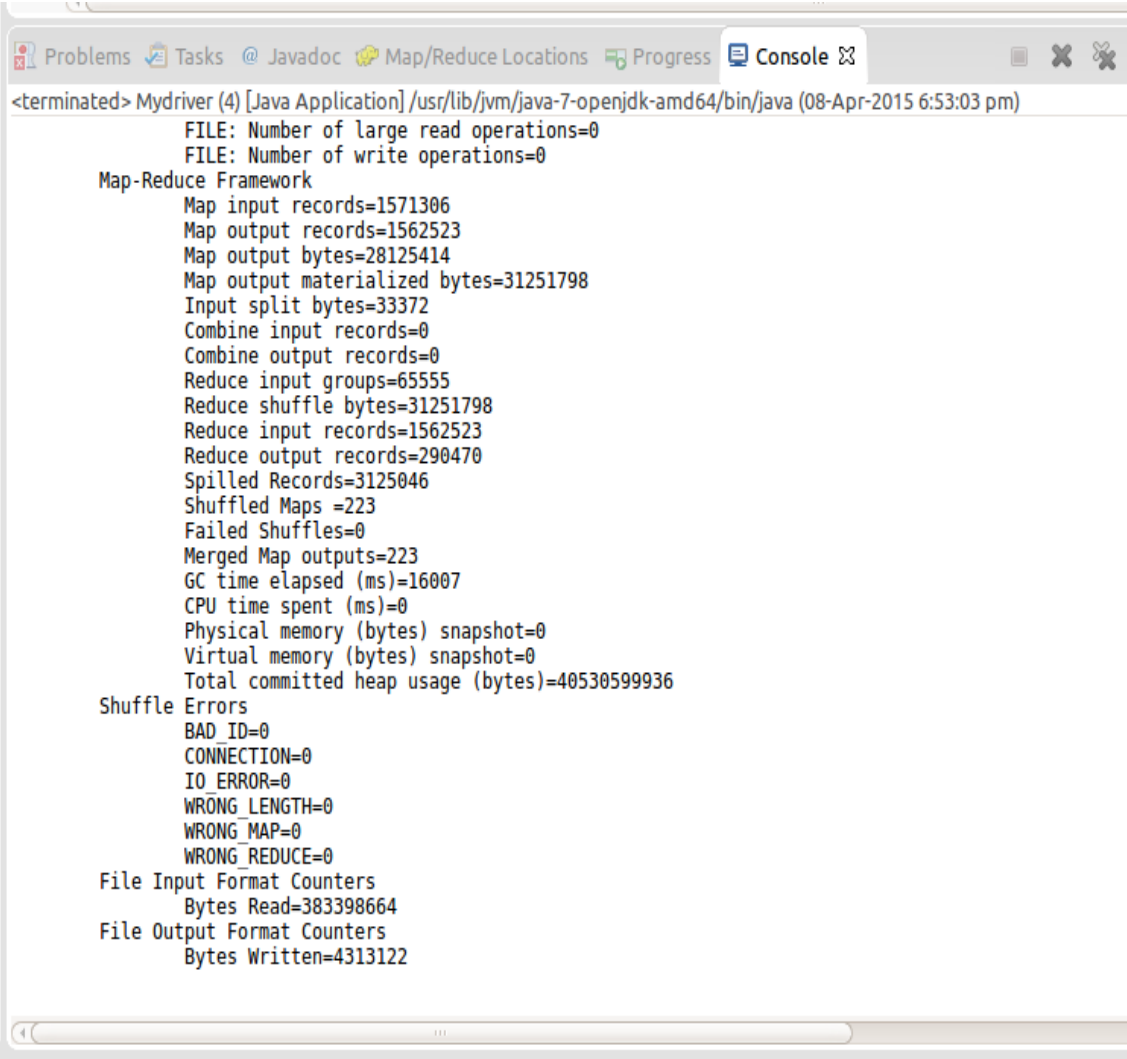
Once the mapper is done, all the <key, value> pairs that match the query are sent back to the reducer. The reducer then performs the desired averaging operation on the sorted <key, value> pairs to create a final <key, value> pair result.

This final result is then stored as a sequence file within the HDFS.

#### IV. Analysing NCDC Data

National Climatic Data Center (NCDC) have provide weather datasets [14]. Daily Global Weather Measurements 1929-2009 (NCDC, GSOD) dataset is one of the biggest dataset available for weather forecast. Its total size is around 20 GB. It is available on amazon web services [15].The United States National Climatic Data Center (NCDC), previously known as the National Weather Records Center (NWRC), in Asheville, North Carolina is the world's largest active archive of weather data. The Center has more than 150 years of data on hand with 224 gigabytes of new information added each day. NCDC archives 99 percent of all NOAA data, including over 320 million paper records; 2.5 million microfiche records; over 1.2 petabytes of digital data residing in a mass storage environment. NCDC has satellite weather images back to 1960.

Proposed System used the temperature dataset of NCDC in 2014. The records are stored in HDFS. They are splitted and goes to different mappers. Finally all results goes to reducer. Due to practical limit, the analysis is executed in Hadoop standalone mode. MapReduce Framework execution as shown in figure 5. The results shows that adding more number of systems to the network will speed up the entire data processing. That is the major advantage of MapReduce with Hadoop framework.



```
<terminated> Mydriver (4) [Java Application] /usr/lib/jvm/java-7-openjdk-amd64/bin/java (08-Apr-2015 6:53:03 pm)
FILE: Number of large read operations=0
FILE: Number of write operations=0
Map-Reduce Framework
Map input records=1571306
Map output records=1562523
Map output bytes=28125414
Map output materialized bytes=31251798
Input split bytes=33372
Combine input records=0
Combine output records=0
Reduce input groups=65555
Reduce shuffle bytes=31251798
Reduce input records=1562523
Reduce output records=290470
Spilled Records=3125046
Shuffled Maps =223
Failed Shuffles=0
Merged Map outputs=223
GC time elapsed (ms)=16007
CPU time spent (ms)=0
Physical memory (bytes) snapshot=0
Virtual memory (bytes) snapshot=0
Total committed heap usage (bytes)=40530599936
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=383398664
File Output Format Counters
Bytes Written=4313122
```

Fig 5 : MapReduce Framework Execution

## V. Conclusion

In traditional systems, the processing of millions of records is time consuming process. In the era of Internet of things, the meteorological department uses different sensors to get the temperature, humidity etc values. Leveraging Mapreduce with Hadoop to analyze the sensor data alias the bigdata is an effective solution.

MapReduce is a framework for executing highly parallelizable and distributable algorithms across huge data sets using a large number of commodity computers. Using Mapreduce with Hadoop, the temperature can be analyse effectively. The scalability bottleneck is removed by using Hadoop with MapReduce. Addition of more systems to the distributed network gives faster processing of the data.

With the wide spread employment of these technologies throughout the commercial industry and the interests within the open-source communities, the capabilities of MapReduce and Hadoop will continue to grow and mature. The use of these types of technologies for large scale data analyses has the potential to greatly enhance the weather forecast too.

## References

- [1] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google File System. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP '03, pages 29–43, New York, NY, USA, October 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945450.
- [2] Jeffrey Dean and Sanjay Ghemawat. MapReduce: A Flexible Data Processing Tool. Communications of the ACM, 53(1):72–77, January 2010. ISSN 0001-0782. doi: 10.1145/1629175.1629198.
- [3] Wiki (2013). Applications and organizations using hadoop. <http://wiki.apache.org/hadoop/PoweredBy>
- [4] Gartner Research Cycle 2014, <http://www.gartner.com>
- [5] K. Morton, M. Balazinska and D. Grossman, “Paratimer: a progress indicator for MapReduce DAGs”, In Proceedings of the 2010 international conference on Management of data, 2010, pp.507–518.
- [6] Lu, Wei, et al. “Efficient processing of k nearest neighbor joins using MapReduce”, Proceedings of the VLDB Endowment, Vol. 5, NO. 10, 2012, pp. 1016-1027.
- [7] J. Dean and S. Ghemawat, “Mapreduce: simplied data processing on large clusters”, in OSDI 2004: Proceedings of 6th Symposium on Operating System Design and Implementation.
- [8] E. Dede, Z. Fadika, J. Hartog, M. Govindaraju, L. Ramakrishnan, D.Gunter, and R. Canon. Marissa: Mapreduce implementation for streaming science applications. In E-Science (e-Science), 2012 IEEE 8<sup>th</sup> International Conference on, pages 1 to 8, Oct 2012.
- [9] Thilina Gunarathne, Tak-Lon Wu, Judy Qiu, Geoffrey Fox, MapReduce in the Clouds for Science, 2nd IEEE International Conference on Cloud Computing Technology and Science, DOI 10.1109/CloudCom.2010.107
- [10] Vinay Sudhakaran, Neil Chue Hong, Evaluating the suitability of MapReduce for surface temperature analysis codes, DataCloud-SC '11 Proceedings of the second international workshop on Data intensive computing in the clouds, Pages 3-12, ACM New York, NY, USA ©2011
- [11] Hansen, J., R. Ruedy, J. Glascoe, and M. Sato. “GISS analysis of surface temperature change.” J. Geophys.Res.,104, 1999: 30,997-31,022.
- [12] Hansen, James, and Sergej Lebedeff. “Global Trends of Measured Surface Air Temperature.” J. Geophys. Res., 92,1987: 13,345-13,372.
- [13] Jaliya Ekanayake and Shrideep Pallickara, MapReduce for Data Intensive Scientific Analyses, eScience, ESCIENCE '08 Proceedings of the 2008 Fourth IEEE International Conference on eScience, Pages 277-284, IEEE Computer Society Washington, DC, USA ©2008
- [14] National Climatic Data Centre , <http://www.ncdc.noaa.gov>
- [15] Daily Global Weather Measurements 1929-2009 (NCDC,GSOD) dataset in Amazon web services , <http://www.amazon.com/datasets/2759>