International Journal of Current Advanced Research
ISSN: O: 2319-6475, ISSN: P: 2319-6505, Impact Factor: SJIF: 5.995
Available Online at www.journalijcar.org
Volume 7; Issue 2(K); February 2018; Page No. 10369-10373
DOI: http://dx.doi.org/10.24327/ijcar.2018.10373.1754

Research Article

IMPLIMENTATION OF MAPR OVER MULTICORE FOR ANALYSIS OF METEOROLOGICAL DATA

Garima Sharma and Seema Maitrey

Krishna Institute of Eng & Technology, Duhai Gzb, India

ARTICLE INFO

Article History:
Received 19th November, 2017
Received in revised form 27th December, 2017
Accepted 4th January, 2018
Published online 28th February, 2018

Key words:
Weather forecasting, meteorology, Mapreduce, phoenix++, Multicore.

ABSTRACT

Nowadays, large amount of data is generating day by day in every field like scientific research, medical, meteorological, climate, finance or marketing. To analyse the pentabytes of data across an extremely wide increasing wealth of weather variable, we need to build a platform for this, which is extremely flexible & scalable. Mapreduce is the most suitable & efficient parallel programming pattern for processing the big data analysis. There are so many frameworks have implemented to achieve high performance for the retrieval of information. This paper explains the analysis of weather forecasting data stored at NCDC and later processed by Mapreduce Framework. This paper also explains different frameworks of mapreduce for parallelizing computing heterogeneous & multi-core environments such as phoenix & phoenix++.

INTRODUCTION

The meteorological data analyses consisting of analyzing the voluminous data collected from various instruments. To attain the scalability & better performance for analysing the meteorological data, there must be some efficient parallel/concurrent algorithms & frameworks to retrieve the best result in efficient amount of time. Weather forecasting is always a big challenge for the meteorologists to predict the state of atmosphere at some point in future. As prediction of weather forecasts plays a very important role for individuals, organisation, farmers, flights, residents of coastal areas [1]. As the cities are getting smarter day by day, there are various latest technologies emerges in this field like Doppler radar, observation tools and sensor technologies. All these generate high volume of data. So there is a need of scalable analytical tools to process this massive amount of data generated day by day. As the traditional approaches to process this massive amount of data is very slow [2]. The mapreduce technique attains a lot of attention in the field of meteorological data for its applicability in large parallel data analyses [3]. There are different implementations of mapreduce for parallelizing computing heterogeneous & multi-core environments such as phoenix & phoenix++. The multi-core Mapreduce framework is being designed to use the shared memory of system. All map & reduce tasks operated on the shared data structures that must be stored in the main memory. This framework is available for the programmers in the form of libraries where as hadoop implementation provides an API via java run time system in the form of a set & classes & objects. This approach helps in fast communication between the tasks through shared memory [4]. Considering the high performance, phoenix++ outperforms all its predecessors and it achieves a 28.5x speedup while executing on a single machine (not distributed). This motivates us to use hadoop for distributed memory & phoenix++ for shared memory multi-core architecture [5].

Corresponding author: **Garima Sharma**
Krishna Institute of Eng & Technology, Duhai Gzb, India

Existing implementations/related works

Table 1 Related work on meteorological data

Authors	Year	Model explanantion
C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis.	2007	proposed the first version of Phoenix as an optimized implementation of MapReduce for shared-memory architectures, with a C-based programming interface
R. M. Yoo, A. Romano, and C. Kozyrakis	2009	Phoenix project with new optimizations for multi-core architectures with non-uniform memory access.
J. Talbot, R. M. Yoo, and C. Kozyrakis	2011	Proposed Phoenix++, consisting of a completely rewritten version of Phoenix, implemented in C++ and taking advantage of the language's capabilities for modularity and code reuse in order to allow a more adaptive programming interface.
Fraley, C., Raftery, A. E., Gneiting, T., Sloughter, J. M., &Berrocal, V. J.	2011	proposed two R packages for probabilistic weather forecasting, ensemble BMA
White, T.	2012	Hadoop and MapReduce is used for weather forecasting it is more efficient than the existing techniques.
Li, L., Ma, Z., Liu, L., & Fan, Y.	2013	ARIMA model prediction algorithm to constructs meteorological data storage and mining platform based on the Hadoop framework.
Ahmed, A., Rashid, A., &Iqbal, S.	2014	Analyzed a simple probabilistic weather forecasting model for Islamabad weather

		using PRISM model checker.
Dagade, V., Lagali, M., Avadhani, S., &Kalekar, P	2015	Proposed data analysis using Apache Hadoop and Apache Spark.
Riyaz, P.A.,& Varghese, S. M.	2015	The Big data collected by NCDC build a data analytical engine for high velocity, huge volume temperature data from sensors using MapReduce on Hadoop
Krishna, G. V.	2015	weather forecasting conditions depending upon the space and time dependency along with the climatic variable using ARIMA ALGO

Literature gap

Some researches worked on improving hadoop's performance on single-node level, mainly by avoiding some internal mechanisms such as message passing & replication, not needed for non- distributed environments& harnessing the computation power of multi-core. These researches also keep the hadoop's programming interface, therefore not adding a new abstraction layer. These evaluation shows that phoenix++ still outperforms in the single-node level. [6][7][8].

Preliminaries

In this section we are going to present some preliminaries. We first introduce the mapreduce technique functionality, multi-core mapreduce framework and phoenix++. Then we are also going to explain the working of mapreduce & multi-core framework that are going to implement to fulfil our implementation.

The mapreduce model. Mapreduce is a programming model proposed by the Google for large scale data processing in distributed computing environment. This model is derived from the functional programming concepts [9]. Each mapreduce job is composed of map & reduces tasks. The Hadoop breaks the input data into multiple data items and run the map function once for each item by giving the item as the input for the function. When executed, the map function output one or more key-value pairs. Hadoop collect all the key-value pairs generated from the map function, sorts them by the key and group together the values with same key.

Table 2 phases of mapreduce technique [10]

Mappers	Required to generate an arbitrary number of intermediate pairs.
Reducers	Applied to all intermediate values associated with the same intermediate key.
Partitioners	Its main job is to divide the intermediate key space, and then to assign the intermediate key-value pairs to reducers.
Combiners	• Combiners are an (optional) optimization • Before performing the phase of shuffle and sort, it allows the local aggregation of data. • Essentially, combiners are used to save bandwidth, e.g.: word count program.

There are various features & principle of mapreduce which make it usable along with many research fields such as, Low-cost unreliable commodity hardware, extremely scalable RAIN cluster, simplified and restricted yet powerful, highly parallel yet abstracted, high throughput, shared-disk storage yet shared-Nothing computing etc [11].

Multi-core mapreduce framework: The multi-core Mapreduce framework is being designed to use the shared main memory of the system. Mapreduce frameworks for Multi-core architectures make the use of pointers. The main functions in the entire framework provide pointer to the input/output data buffers.

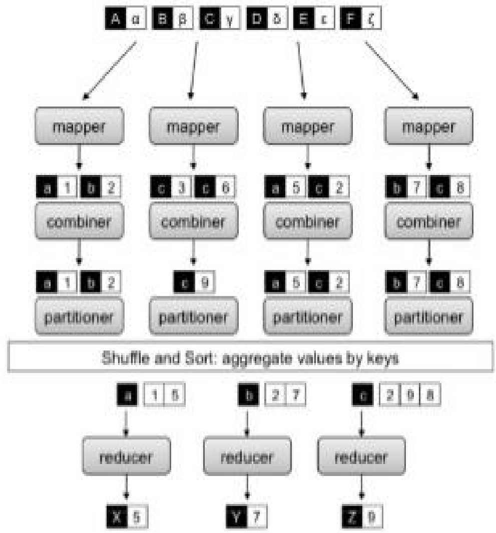


Fig 1 working of mapreduce technique

The use of pointer allows the data manipulation through all the phases without the need for copying large amounts of data [12]. For instance, keys are not copied to intermediate data structures. Instead, pointers to keys in the original input are used. There is an additional benefit to simplify the implementation for any key data type & size.

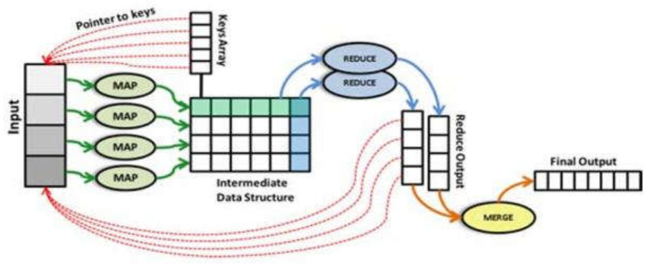


Fig 2 Multicore mapreduce framework

Pheonix++. To improve execution speed through modularity in critical sections phoenix++ was developed.

- It provides a flexible intermediate key/value storage abstraction that permits to adapt the specific implementation to the features of the workload.
- It includes a more effective combiner implementation that can minimize memory usage.

The phoenix++ considers different types of key distributions & basic execution parameters, which directly use the shared memory. Phoenix ++ classifies Mapreduce applications by the number of keys that map tasks will emit:

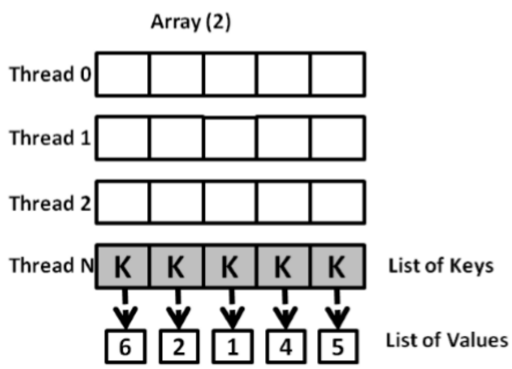


Fig 3 phoenix++ fixed-size thread-local array

Common Array (1:1) – This is a non- blocking array structure shared across all threads. In this each emitted key is unique and all threads can write into the same array without any synchronization.

Array (*: k) - any map task can emit any of the fixed number of keys which requires the keys to be integers within an a priori range. It provides an increase in performance in the cases where there is a small computation per task.

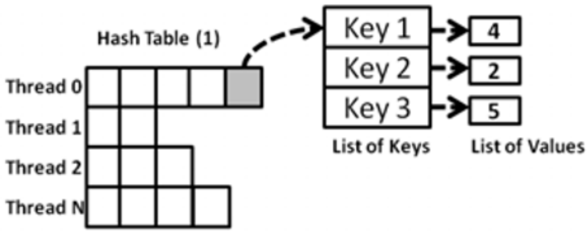


Fig 4 phoenix ++ variable-width hash table

Hash Table (* :*) - any map task can edit any key, where number of keys is not known before execution. This is the main improvement from the previous phoenix version which improves the search in large key arrays. Even in the presence of an unexpectedly large number of keys, it ensures the insertion complexity kept in order of O (1).

There is a combiner function in the phoenix++ that performs a local reduction per thread, after each key/value pair is emitted by the map function. This combiner function reduces the amount of memory required when having multiple values for the same key inside each map tasks. It is also having encapsulated task chunking mechanism due to which User-exposed functions called with one task at a time & compilation time optimization eliminate overheads.

Comparative analysis of mapreduce frameworks

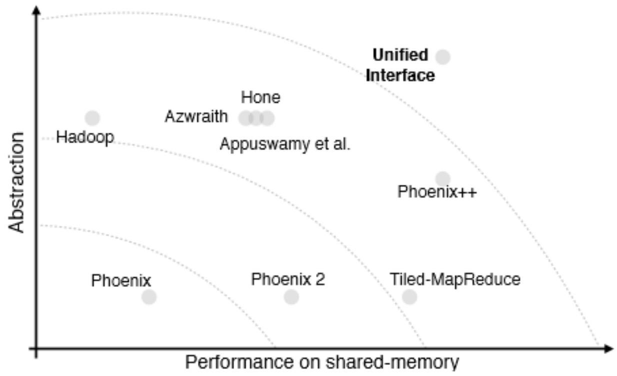


Fig 5 relationship graph between abstraction & performance on the programming interface design goals.

Proposed work: National Climate Data centre (NCDC) have provided huge amount of historical weather datasets. Daily global weather measurement 1929-2016 (NCDC) dataset is one of the biggest historical weather dataset available for the weather forecasting. In this proposed system, we use the dataset of NCDC contain the parameters like city name, city code, temperature, prediction etc. The Centre 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.

Problem statement

1. The system receives temperatures of various cities (Austin, Boston,etc) of USA captured at regular intervals of time on each day in an input file.
2. System will process the input data file and generates a report with Maximum and Minimum temperatures of each day along with time.
3. Then it will generate a separate output report for each city.

Steps for the analysis of approach

1. First from NCDC site we download the input file which contains the temperature statistics with time from multiple cities
2. Apply MAPR technique on those clusters of the core that are going to work parallel.
3. In the MAPR there are two functions MAP & REDUCE which works as follow:-

The mapper class followed by the type of map method parameters.

- Public class whetherforecastmapper extends Mapper <object,text,text,text> Object(KeyIn)- Offset for each line, line number 1,2…. Text(ValueIn)- Whole string for each line Text(KeyOut)- city information with date information as string. Text(Value Out)- Temperature & time information which need to be passed to reducer as string.
- Public void reduce(Text key, Iterable<text> values, context context) Text key is value of mapper output i.e. city & date information Iterable<text> value- in this values stores multiple temperature values for a given city & date.

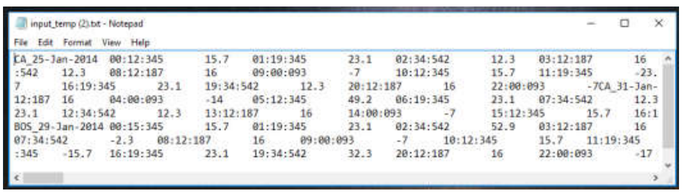
Context object is where reducer write its processed outcome & finally written in file.

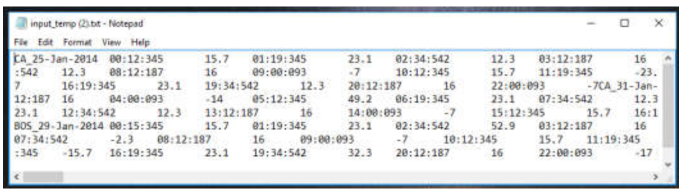
Reducer Generates output file. For a given key, there are two entries (max temperature time & mini temp time). Iterate value list, split value & get temperature & time value. Compare temperature value & create actual value string which reducer writes in appropriate file.

1. Finally, the setup generates the output for each city in which a maximum & minimum temperature is generated at a particular time for a city.
2. According to those temperatures we can predict the upcoming temperature analysis for that city.

The result is analysed by both the mapreduce technique as well as phoenix++.

Input file


```
input_temp (2).txt - Notepad
File Edit Format View Help
CA_25-Jan-2014 00:12:345      15.7    01:19:345       23.1    02:34:542       12.3    03:12:187       16
:542    12.3    08:12:187       16      09:00:093       -7      10:12:345       15.7    11:19:345       -23.
7       16:19:345       23.1    19:34:542       12.3    20:12:187       16      22:00:093       -7CA_31-Jan-
12:187  16      04:00:093       -14     05:12:345       49.2    06:19:345       23.1    07:34:542       12.3
23.1    12:34:542       12.3    13:12:187       16      14:00:093       -7      15:12:345       15.7    16:1
BOS_29-Jan-2014 00:15:345     15.7    01:19:345       23.1    02:34:542       52.9    03:12:187       16
07:34:542       -2.3    08:12:187       16      09:00:093       -7      10:12:345       15.7    11:19:345
:345    -15.7   16:19:345       23.1    19:34:542       32.3    20:12:187       16      22:00:093       -17
```



```
input_temp (2).txt - Notepad
File Edit Format View Help
CA_25-Jan-2014 00:12:345      15.7    01:19:345       23.1    02:34:542       12.3    03:12:187       16
:542    12.3    08:12:187       16      09:00:093       -7      10:12:345       15.7    11:19:345       -23.
7       16:19:345       23.1    19:34:542       12.3    20:12:187       16      22:00:093       -7CA_31-Jan-
12:187  16      04:00:093       -14     05:12:345       49.2    06:19:345       23.1    07:34:542       12.3
23.1    12:34:542       12.3    13:12:187       16      14:00:093       -7      15:12:345       15.7    16:1
BOS_29-Jan-2014 00:15:345     15.7    01:19:345       23.1    02:34:542       52.9    03:12:187       16
07:34:542       -2.3    08:12:187       16      09:00:093       -7      10:12:345       15.7    11:19:345
:345    -15.7   16:19:345       23.1    19:34:542       32.3    20:12:187       16      22:00:093       -17
```


Fig 6 input weather forecasting data

Step 1: Start local host.

After local host is successfully running. Go to step 2

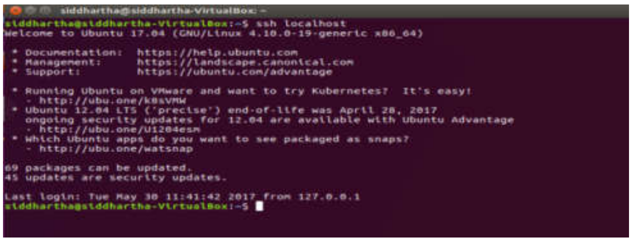


Fig 7 Start local host

Step 2: format the name node before running the demons.

Step 3: Start the DFS & MAPRED demons and check if they are running from JPS command.

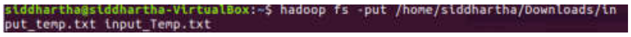


Fig 8 formatting the name node

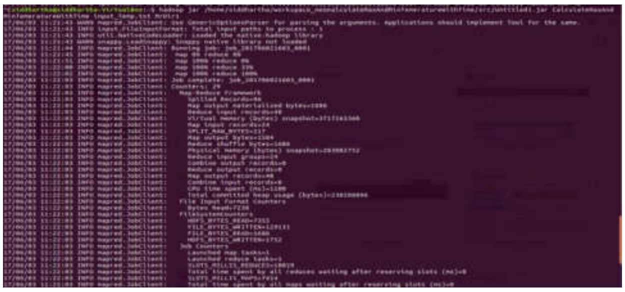


Fig 9 Start the DFS & MAPRED daemons

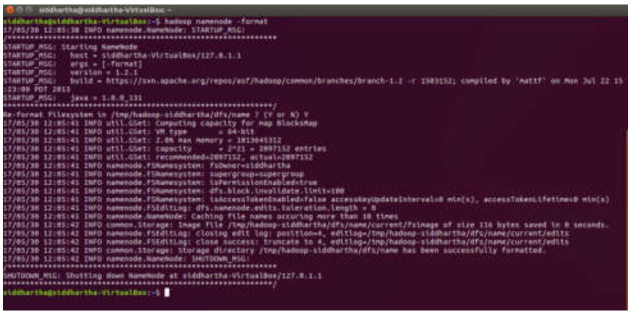


Fig 10 Start the DFS and MAPRED Daemons

Step 4: place the input file inside the hadoop file system using the put command.

Step 5: run the produced JAR file of project using the following commands.

Output

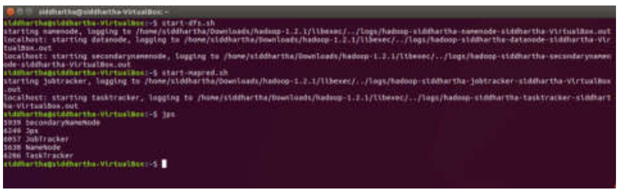


Fig 11 Running the produced JAR file

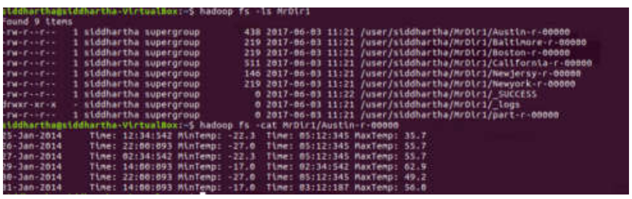


Fig 12 Final Output

CONCLUSION

In the traditional systems, the processing of millions of records is a time consuming process. In the era of Internet of things, the meteorological department uses different sensors to get the temperature, humidity values etc. We have proposed Multicore technology with MAPR for analysis & prediction of meteorological data.

MAPR is a frame work for highly parallel & distributed systems across huge dataset. Using the mapreduce along with Multicore gives the best performance in the system. As it overcomes the limitations of the MAPR. According to the mapreduce technique there is a need of many nodes over which the data is distributed on a large scale. The framework of Multicore over mapreduce includes the necessary optimisation features to make it highly efficient & scalable while analysing and retrieval of data from large data set.

Phoenix++ outperforms all its predecessors and it achieves a 28.5x speedup while executing on a single machine (not distributed). This motivates us to use hadoop for distributed memory & phoenix++ for shared memory multi-core architecture.

Future scope

The future scope relates with the working on improving hadoop's performance on single-node level. This research work can be future carried out for predicting several other conditions regarding weather forecasting and modified can be used to make prediction in other fields also.

References

1. Veershetty Dagade, Mahesh Lagali, Supriya Avadhani, Priya Kalekar, "Big Data Weather Analytics Using Hadoop", *International Journal of Emerging Technology in Computer Science & Electronics* (IJETCSE) ISSN: 0976-1353 Volume 14 Issue 2 – APRIL 2015
2. Riyaz P.A., Surekha Mariam Varghese "Leveraging Map Reduce With Hadoop for Weather Data Analytics", IOSR *Journal of Computer Engineering* (IOSR-JCE), Volume 17, Issue 3, Ver. II (May – Jun. 2015), PP 06-12
3. Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox, "MapReduce for Data Intensive Scientific Analyses", Fourth IEEE International Conference on eScience 2008
4. Rong chen, haibo chen, Binyu Zang. "tiled mapreduce: Optimizing resource usage of data-parallel application on Multicore with tiling", In proceeding of the 19th international conference on parallel architectures & compilation techniques, PACT 10, pages 523 534, New York NY, USA, 2010. ACM
5. Daniel Adornes, Dalvan Griebler, Cleverson Ledur, Luiz Gustavo Fernandes, "A Unified MapReduce Domain-Specific Language for Distributed and Shared Memory Architectures".
6. Z. Xiao, H. Chen, and B. Zang. A Hierarchical Approach to Maximizing MapReduce Efficiency. In Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques, PACT '11, pages 167-168, Washington, DC, USA, October 2011. IEEE Computer Society.

7. R. Appuswamy, C. Gkantsidis, D. Narayanan, O. Hodson, and A. Rowstron. Scale-up vs Scale-out for Hadoop: Time to Rethink? In Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC '13, pages 20:1-20:13, Santa Clara, CA, October 2013. ACM.
8. K. A. Kumar, J. Gluck, A. Deshpande, and J. Lin. Optimization Techniques for "Scaling Down" Hadoop on Multi-Core, Shared-Memory Systems. In Proceedings of the 17th International Conference on Extending Database Technology, EDBT '14, pages 13-24, Athens, Greece, 2014. Open Proceedings.
9. Jaliya Ekanayake, Shrideep Pallickara, and Geoffrey Fox, "MapReduce for Data Intensive Scientific Analyses", Fourth IEEE International Conference on eScience, 2008 IEEE
10. Seema Maitreya, C.K. Jha (2015) "MapReduce: Simplified Data Analysis of Big", Elsevier Procedia Computer Science 57 (2015) 563-571
11. Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, D. Stott Parker (2007), "MapReduce-Merge: Simplified Relational Data Processing on Large Clusters", June12-14,2007 ACM.
12. Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis. "Evaluating mapreduce for multi-core & multiprocessor system", pages 13 24. In HPCA 07: Proceeding of the 13th International Symposium on high performance computer Architecture, IEEE computer Society, 2007

How to cite this article:

Garima Sharma and Seema Maitrey (2018) 'Implimentation of Mapr Over Multicore for Analysis of Meteorological Data', *International Journal of Current Advanced Research,* 07(2), pp. 10369-10373.
DOI: http://dx.doi.org/10.24327/ijcar.2018.10373.1754

****