International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 9 5629 - 5633

Load Balancing Through Map Reducing Application Using CentOS System

Nidhi Sharma
Research Scholar, Suresh Gyan Vihar University,
Jaipur (India)

Bright Keswani
Associate Professor, Suresh Gyan Vihar University,
Jaipur (India)

Abstract:- The main aim is to design an algorithm for Map-Reduce application, with a ballot count application, which explains how the big data can be handled by the individual Mappers and how data is split using map() function. The output of Mappers and is given to the Reducer and reduce () function is applied. The execution takes set of input key/value pairs, and gives a set of output key/value pairs. Map, accepts the input from the user, processes it and generates a set of intermediate key/value pairs. The Map Reduce library groups all the intermediate values of the same category and passes it to the Reduce function. The Reduce function takes the intermediate key and a set of values associated with the key. It groups together all the associated values and generates a smaller set of values. Finally, only zero or one output value is generated by the reducer. The iterate is responsible for accepting the intermediate key and its associated value.

Keywords:- *Ad hoc, Map Reduce, CentOS System, Cloud Computing, Load Balancing etc.*

*****

I. INTRODUCTION

Cloud computing is one of the emerging technology in the current era. It helps the users to share resources, information and devices, so, reduced time and cost involved. Capital and Operational cost is also greatly reduced. It works on multi-tier architecture. Internet network works on the basis of cloud strategy.

The major issue to be addressed in cloud is the "Load Balancing". Load balancing is the method of distributing the services equally throughout the cloud. The cost of the cloud may increase if it holds idle resources. So, instead of allocating the jobs to the same resources again and again and increasing the pressure on a particular system, it is profitable to distribute the job throughout the cloud as shown in Fig:1.

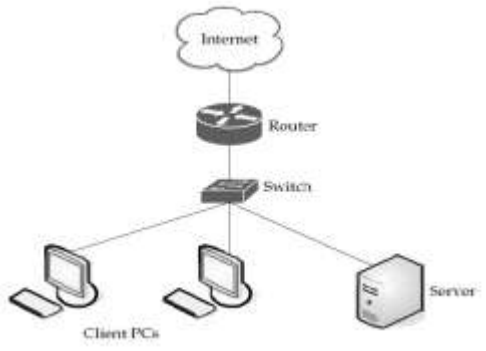

Figure 1: Distribution of jobs

Work flow defines a set of jobs and its dependency. In XML, workflow is represented as a Directed Acyclic Graph (DAG). The application/task is represented as node and the dependency of execution among the tasks is represented as "Directed Edge". Dependency exists with the data involved, its execution order and dataflow from one task to another. Certain tasks can be processed in parallel. Cloud computing, using services reduce the effort involved to access high performance computing and also information storage. The advantages are configurability, scalability and reliability with the high performance. The cost of execution of a job on a cloud is depending on the amount of computation done and the resources consumed. Map-Reduce model of programming consists of two functions- "Map" and "Reduce". The function of "Map" executes a block of inputs thus giving output as sequence of pairs, whereas the function "Reduce" processes single key function and its associated set of values.

Google for the usage of web-indexing technologies applied Map-Reduce operation. This is used to monitor the web-pages viewed and generate inverted-indices and summarize the web-pages for search-results. Later the methodology was applied to larger processes like query processing for both raw and derived data. This made many others use this process for their own set of evaluation of the organizational data. Map-Reduce help in managing large volumes of the data and with larger data sets which had dependencies existing among them. The aim of this work is to generate an algorithm using Map-Reduce model for the application ballot-count which had big set of input data. There are two main processes, one is to group ballot-count in different regions and create the total ballot count and other is to use combiner utility function to document the data. The functions map() and reduce() are mainly used. For processing three functions map(); combine(); and reduce() are used.

II. LOAD BALANCING

It is a process of distributing the total load to the different nodes of the whole system to make effective resource utilize. It is used to improve the time response of the task, and at the same time we will remove a condition in which some nodes become over loaded whereas others are not. A dynamic natured load balanced algorithm does not use the previous state or nature of the system, i.e., depending upon the present state of the system. The important points to remember while writing this type of algorithm are: load estimation, load comparison, system stability, system performance, action between the nodes, transferred work's behavior and select nodes etc. This load can be considered in types of load of CPU, memory used ratio, delay or load of Network.

III. DISTRIBUTED SYSTEM OF BALANCING LOAD FOR CLOUD

There is a requirement for balancing of load in the complicated and big systems. To make balancing of load in the cloud, a thing that has to be done is, using such techniques by which components of the system of cloud acts as the balanced load of the full system. For this purpose, we can discuss three solutions which can be used in a distributed purposely: algorithm for honeybee foraging, a biased and choose at random samples at a random walk process and Active Clustering.

IV. TECHNIQUE OF HONEYBEE FORAGING

This algorithm derived from the nature of honey bees for searching and making food. A class of honey bees called forager bees which can search for sources of food, and on finding it, they return to their beehive to tell to everyone by a dance called "waggle dance". The steps of that dance, gives the location of food and their quantity and also the distance of the food source from their beehive. Foragers then show the way to scout bees to the food location and then start to make it. Then they come back to their beehive and start doing waggle dance, which shows quantity of food left and so results in more using and this course of action for the source of food.

In balancing of load's case, according to the increment and decrement in the demand in web servers, the services are allocated dynamically to meet the changing user demands. The servers are categorized under "Virtual Servers (VS)", each and every Virtual Servers have its own virtual service buffers. Each and every server processes a request from its buffer which calculates a benefit or profit, which is same to the quality of honey bees as shows in their dance. Benefit of this technique can be measured by the time amount which is used by CPU to process a request. The honey bees dance floor is same as an advert board in this case. The benefit of the whole colony can be advertised by that advert board.

Each and every server has to take the job of either a "forager role" or "scout role". After process a request, the server can post their probability of px as well as their benefit on the board. A buffer can be chosen by the server of Virtual server by a probability of px expressing forage/explore nature, and it can also check all advertisements and process it, and thus showing scout nature.

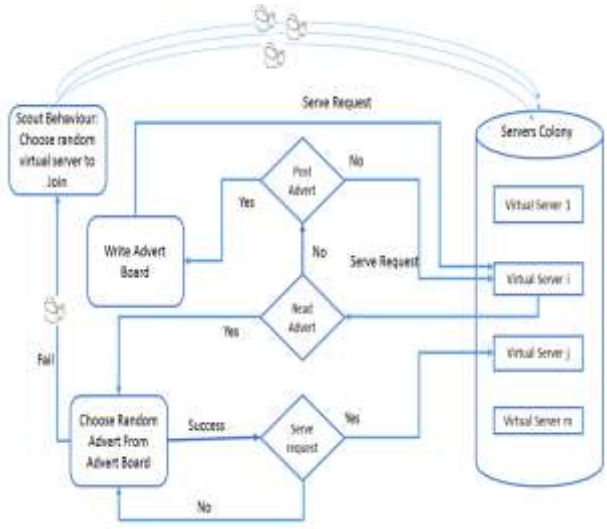

Figure 2: Honey Bee Foraging Technique

V. MAP REDUCE: ISOLATED PROCESSES

Hadoop is designed to efficiently process large volumes of information by connecting many commodity computers together to work in parallel. The theoretical 1000-CPU machine described earlier would cost a very large amount of money, far more than 1,000 single-CPU or 250 quad-core machines. Hadoop will tie these smaller and more reasonably priced machines together into a single cost-effective compute cluster. Hadoop controls the all communication perform by the processes and each alone record is controlled by a work in the state of being separate from one and another. While it is like a large demerit at first, it makes the full framework much better. Hadoop will not execute like any program and divide it across the cluster. "Map Reduce" is a particular programming model is used to conform to the written programs.

In "Map Reduce", all records are executed in the state of being separate by works called as "Mappers". The output of the Mappers is brought together to a other set of jobs called "Reducers", where different mappers results can be combined together as shown in Figure 3.

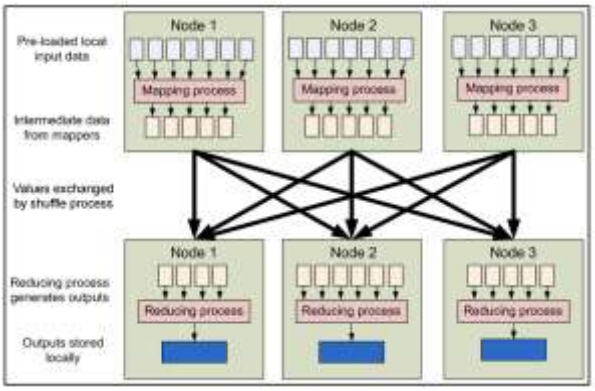

Figure 3: Mapping and Reducing Process in Map Reduce

Different nodes in the Hadoop cluster still interact with another node. However, more normal divided systems where application makers exactly marshal byte streams from one node to another node over the sockets or by the MPI buffers. Communication performed in the Hadoop is implicit communication. Packets of data can be labeled with key names which tell Hadoop how to transfer the bits of information data to a common final node. Hadoop controls all the cluster topology issues and data transfer internally.

By jamming the node to node communication, Hadoop makes the distributed system much easier. Individual failures of nodes can be done around by re-starting jobs on the other systems. Since jobs of user-level do not interact explicitly with one to other, there is no need to exchange messages by user programs, same as nodes also do not need to roll back to checkpoints which are pre-arranged to partially restart the calculation. The rest workers keep operate as here nothing went wrong, leaving the challenging points of partially begin the program again to the underlying layer of Hadoop.

VI. FLATSCALABILITY IN HADOOP

One of the big merits of using Hadoop as compare to another distributed systems is the flat scalability curve. Processing Hadoop on a less amount of information data and on a less number of nodes may not be makes particularly stellar performance, the overhead used in starting Hadoop programs is quite high. Other parallel/distributed programming patterns like MPI (Message Passing Interface) can perform much satisfactory on two, four, or possibly a dozen system machines. In spite of the fact that the effort of co-ordinating task to less number of machines can be well-performed by such machines, the cost paid in performance and developing effort incrementing non-linearly.

A program written in divided frameworks apart from Hadoop may require huge amounts of refactoring when measuring from ten to hundred or thousand machine systems. This can be involved taking the program be written again and again many times; basic elements of its design can also bound the scale to which the application can increase its strength.

Hadoop is mainly developed to have a curve of flat scalability. After Hadoop program is being written and performing on ten nodes, if any job is need for that program to execute on a big amount of hardware. The amount of growth magnitude are manage with very less re-job is needed for the applications. The records and hardware sources manages by the given Hadoop platform. The growth of the performance of the system which is proportionate to the large number of system available to the system which is dependable to the system.

VII. MAP-REDUCE SEARCH METHODOLO-GY

The job of the Hadoop system is implemented as workload. The system nodes of map jobs and reduce jobs are depends on each other. Basically the communication exists between map jobs and reduces jobs. The job of the reducers is fixed in the basic Hadoop system. Hash partitioned gives Beforehand and the reducer to the value pair. The function hash practitioner specifies the key to return back value among 0 and 1 which is sent to that reducer. Reducer load unbalancing exists in the system as mostly reducers receive much more key value pairs. This imbalance can be reduced by multiprocessor scheduling algorithm and a reduction in the gain of the execution time as we gives roles to the reducer as shown in Fig: 4.

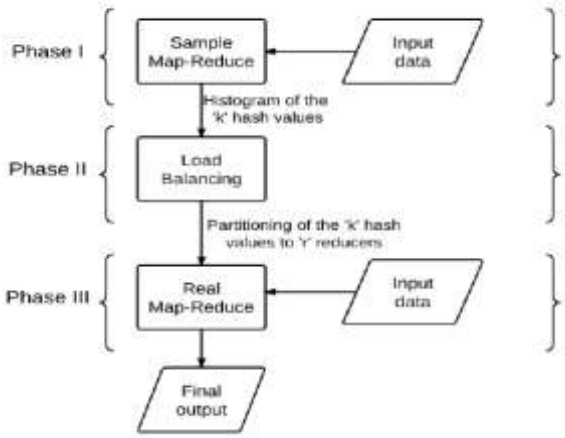

Figure 4: Input-Output phases of Map Reducing Technique

VIII. PROBLEM STATEMENT

- Input: text documents in large numbers
- Task: Calculate the word count on all of the documents

 ISSN: 2321-8169
5629 - 5633

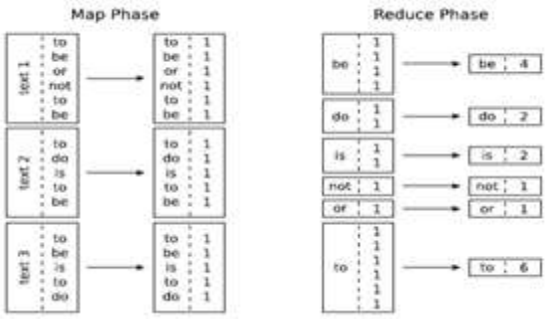

Figure 5: Map and Reduce Phase Working

IX. PROBLEM SOLUTION

- Mapper: every word in the document gives (word, "1")
- Reducer: Sum all occurrences' of word and outputs (word, total_count)

The two functions that associate to the Map-Reduce

programming model are as follow:

- Map Function:
  (K_{in}, V_{in}) -----→list (K_{inter}, V_{inter})
- Reduce function:
  $(K_{inter}, list(V_{inter}))$---------→ list (K_{out}, V_{out})

X. RESULTS

To initiate the project first we have to start our CentOS system at our virtual machine that is Oracle VM virtual box supported by Horton works Sandbox as shown in Figure 6.

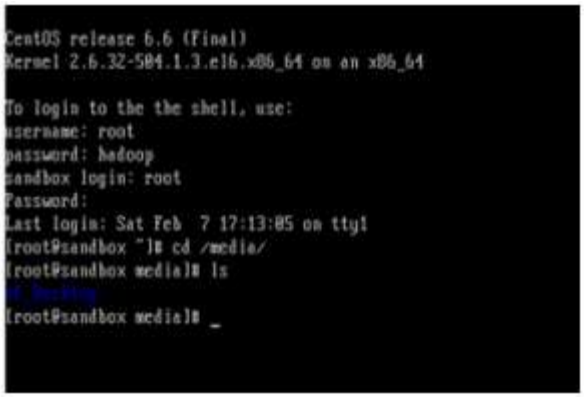

Figure 6: CentOS login Window

After initiation and login formalities the system will now process the files as per run command. For this first we have to import all input files in job browser and then have to run the program file to get our output information. The output window is shown in Figure 7.

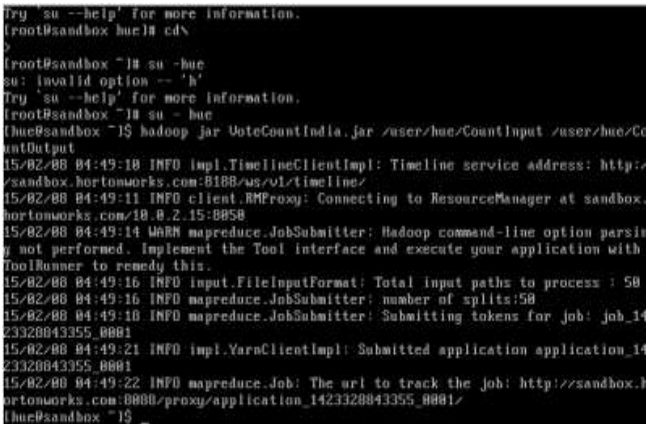

Figure 7: Output Window

The following output produced after calculating all the system inputs as shown in Figure 8.

Figure 8: Result Window

XI. CONCLUSION

It is helpful in usage of web-indexing technologies applied Map-Reduce operation. This is used to monitor the web-pages viewed and generate inverted-indices and summarize the web-pages for search-results. Later the methodology was applied to larger processes like query processing for both raw and derived data. This made many others use this process for their own set of evaluation of the organizational data. Map-Reduce were helpful in managing large volumes of the data and with larger data sets which had dependencies existing among them. The aim of this work is to generate an algorithm using Map-Reduce model for the application ballot-count which had big set of input data. There are two main processes, one is to group ballot-count in different regions and create the total ballot count and other is to use combiner utility function to document the data. The functions map() and reduce() are mainly used. For processing three functions map(); combine(); and reduce() are used.

XII. REFFERENCES

[1] Hadoop, http://hadoop.apache.org/mapreduce/.
[2] D. Abadi et al. Column-Oriented Database Systems. PVDLB,2(2):1664–1665, 2009.
[3] F. N. Afrati and J. D. Ullman. Optimizing Joins in a Map-Reduce Environment. In EDBT, pages 99–110, 2010.
[4] S. Babu. Towards automatic optimization of MapReduce programs. In SOCC, pages 137–142, 2010.

[5] S. Blanas et al. A Comparison of Join Algorithms for Log Processing in MapReduce. In SIGMOD, pages 975–986, 2010.
[6] J. Dean and S. Ghemawat. MapReduce: A Flexible Data Processing Tool. CACM, 53(1):72–77, 2010.
[7] J. Dittrich, J.-A. Quian´e-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop++: Making a Yellow Elephant Run Like a Cheetah (Without It Even Noticing). PVLDB, 3(1):519–529, 2010.
[8] J. Dittrich, J.-A. Quian´e-Ruiz, S. Richter, S. Schuh, A. Jindal, and J. Schad. Only Aggressive Elephants are Fast Elephants. PVLDB, 5, 2012.
[9] A. Floratou et al. Column-Oriented Storage Techniques for MapReduce. PVLDB, 4(7):419–429, 2011.
[10] A. Gates et al. Building a HighLevel Dataflow System on Top of MapReduce: The Pig Experience. PVLDB, 2(2):1414–1425, 2009.
[11] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google file system. In SOSP, pages 29–43, 2003.
[12] H. Herodotou and S. Babu. Profiling, What-if Analysis, and Cost-based Optimization of MapReduce Programs. PVLDB, 4(11):1111–1122, 2011.
[13] M. Isard et al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In EuroSys, pages 59–72, 2007.
[14] E. Jahani, M. J. Cafarella, and C. R´e. Automatic Optimization for MapReduce Programs. PVLDB, 4(6):385–396, 2011.
[15] D. Jiang et al. The Performance of MapReduce: An In-depth Study. PVLDB, 3(1-2):472–483, 2010.
[16] A. Jindal, J.-A. Quian´e-Ruiz, and J. Dittrich. Trojan Data Layouts: Right Shoes for a Running Elephant. In SOCC, 2011.
[17] J. Lin et al. Full-Text Indexing for Optimizing Selection Operations in Large-Scale Data Analytics. MapReduce Workshop, 2011.
[18] Y. Lin et al. Llama: Leveraging Columnar Storage for Scalable Join Processing in the MapReduce Framework. In SIGMOD, 2011
[19] D. Logothetis et al. Stateful Bulk Processing for Incremental Analytics. In SoCC, pages 51–62, 2010.
[20] A. Okcan and M. Riedewald. Processing Theta-Joins Using MapReduce. In SIGMOD, pages 949–960, 2011.
[21] A. Pavlo et al. A Comparison of Approaches to Large-Scale Data Analysis. In SIGMOD, pages 165–178, 2009.
[22] J.-A. Quian´e-Ruiz, C. Pinkel, J. Schad, and J. Dittrich. RAFTing MapReduce: Fast Recovery on the RAFT. ICDE, 2011.
[23] A. Thusoo et al. Data Warehousing and Analytics Infrastructure at Facebook. In SIGMOD, 2010.
[24] A. Thusoo et al. Hive – A Petabyte Scale Data Warehouse Using Hadoop. In ICDE, 2010.
[25] S. Wu, F. Li, S. Mehrotra, and B. C. Ooi. Query Optimization for Massively Parallel Data Processing. In SOCC, 2011.
[26] M. Zaharia et al. Improving MapReduce Performance in Heterogeneous Environments. In OSDI, 2008.