International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 5 373 - 377

Improving Energy Efficiency of MapReduce Framework using Dynamic Scheduling of Work

Prashant Sugandhi
Computer Department
Dr. D Y Patil School of Engineering,
Pune, India.
prashant.sugandhi1@gmail.com

Harshit Karnewar
Computer Department
Dr. D Y Patil School of Engineering,
Pune, India.
hnk1510@gmail.com

Cheryl Joseph
Computer Department
Dr. D Y Patil School of Engineering,
Pune, India.
cj4783@gmail.com

Prof. Jayashree Chaudhari
Computer Department
Dr. D Y Patil School of Engineering,
Pune, India.
jayashree.chaudhari@dypic.in

***Abstract*—** Most common huge volume data processing programs do counting, sorting, merging etc. Such programs require to perform first a computation on each record that is it requires to map an operation to each record. Then combine the output of these operations in appropriate way to get the answer that is apply a reduce operation to groups of records. MapReduce runtime environment takes care of parallelizing their execution and coordinating their inputs/outputs. Here we are concern about energy efficiency in MapReduce framework so we are proposing dynamic scheduling of workload which offers dynamic load balancing method. Load balancing is the methodology of distributing the load among different node of a distributed framework to enhance both resource usage and reaction time while likewise keeping away from a circumstance where a percentage of the node are intensely stacked while different node are sit out of gear or doing next to no work. An answer for unbalance circumstance is to utilize parallelization approaches yet at the same time node will stay overwhelming. In this paper, we propose an integrated. We are proposing a methodology where the MapReduce concept introduced into the MongoDB with NoSQL as a back end to implement the MapReduce.

Keywords- *MapReduce, MongoDB, NoSQL, Distributed database, Dynamic Load balancing.*

*****

I. INTRODUCTION

Now days, distributed database is spread over the system where application is required and one of the most challenging issue in distributed system is unbalance nodes. To comprehend Load scheduling, it is important to comprehend load. Load may be characterize as number of assignments are running in line, CPU usage, I/O use, measure of free CPU time/memory, and so on or any mix of the above pointers. Load balancing should be possible among interconnected workstations in a system or among individual processors in a parallel machine. Load balancing is only the assignment of errands or employments to processors to expand general processor usage and throughput.

We are proposing a methodology where the MapReduce concept introduced into the MongoDB with NoSQL as a back end to implement the MapReduce.

MongoDB is a schema-free document-oriented database written in C++. It is developed in an open-source project and primarily driven by the company 10gen Inc. It also offers professional services around MongoDB. According to its developers the main goal of MongoDB is to close the gap between the fast and highly scalable key-value-stores and feature-rich traditional RDBMSs relational database management systems. MongoDB name derived from the adjective humongous [15] [16].

In this paper, we proposed a robust architecture which schedules the queries for scalability and distribution of equal work load. Registered incoming CEP queries in system are transferred to idle nodes or not heavily node. To improve both resource utilization and performance scheduler is responsible to distribute the load among various nodes and make it balance. Queries are applied to scheduler module of system, then scheduler schedules the query to lightly weighted node with respect to CPU capacity, memory consumption. Incoming queries are delivered to queue; if queue is full or reaching to limit size in queue measure will be taken. Overload situation can kill the nodes and results into data loss but this will be avoided by using replication method. We proposed a system which plans the query for scalability and conveyance of equivalent work load. Enrolled approaching query in framework are exchanged to node having lower load or not vigorously node. To enhance both asset use and execution scheduler is dependable to disperse the load among different node and make it adjust. Query are connected to scheduler module of framework, then scheduler plans the question to lightly weighted queue regarding CPU limit, memory

utilization. Approaching inquiries are conveyed to Queue; if queue is full or coming as far as possible size in queue measure will be taken. Over-load circumstance can execute the nodes and results into fault tolerance yet this will be dodged by utilizing replication strategy. The proposed system maintain ready queue, use it to queue incoming query when nodes are stacked and none of node is free. At last event is generate if no selection is conceivable. [1]

The rest of the paper is organized as follows. Section II discusses literature review, Section III describes the problem statement, and Section IV discusses proposed method. We conclude the paper in last section.

II. BACKGROUND AND MOTIVATION

Objective of load balancing is as per the following [5]

- Even appropriation of load to every asset.
- Minimization of handling time for every work
- Maximum utilization of each resource.
- Energy consumption minimization

There are numerous methodologies taken inside the writing for learning load balancing. In appropriated systems adjusting the node in a versatile way enhances the system execution impressively

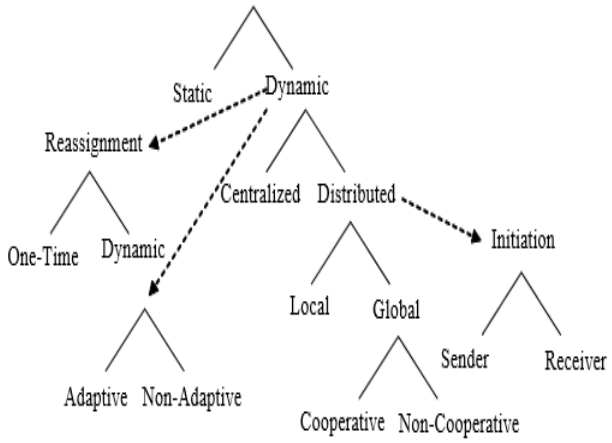


Fig 1- Taxonomy of Load Balancing Scheme

Load balancing is a system to upgrade assets, using parallelism, misusing throughput impromptu creation, and to decrease reaction time through a proper dispersion of the application [4]. Parallelization methodologies are utilized for load balancing as a part of which load is parallelize through numerous machines or centers. Indeed with parallelization approach still database questions are calm overwhelming obligation forms [4]. The essential things to consider while growing such calculation are : estimation of load, examination of load, soundness of diverse framework, execution of framework, cooperation between the nodes, way of work to be exchanged, selecting of node and numerous different ones. This node considered can be regarding CPU load, measure of memory utilized, postpone or Network load.

III. PROBLEM DEFINITION

In previous load balancing studies, a typical approach is to use a straightforward model of the distributed system with assumptions like large communication bandwidth measure, negligible load balancing overheads, homogenous workload, and to go looking for complex load balancing algorithms whose viability is questionable**.** Proposed System will balance Workload by queuing the queries based on capacity of Node also Enhancing Scalability and throughput in dynamic environment by scheduling the load, control the load, to attain optimum scalability even in heavy queries without loss of data even in system crash, and alert system and automatic transfer of queries to node with available capacity with lowest turnaround time. Proposed System aims to achieve scalability and efficiency through architecture for load balancing and query processing in Distributed Database schema.

IV. PROPOSED SYSTEM

Our system tend to style it to efficiently method streams of data queries and stream-DB workloads, using any hardware and stream package. As a demonstration and test scenario take into account a student database with student detail-records (SDR) and at an equivalent time massive databases holding past data and services outline records. Figure 1, shows process to design Dynamic Load balancing algorithm to attain scalability even in heavy queries, avoid fault tolerance when system crash, event generation and handling, job submission control, Overload control The system is comprised by:

- Detection of Overload
- Scheduling of Query
- Load Shedding and Event generation

A. DETECTION OF OVERLOAD

Data distribution node distributes incoming CEP to all processing nodes. Each processing node has its queue with limiting size so new data is added to it. Each node monitors its own queue to handle overload problem when it reaches to certain size estimated by an admin node is submitted to scheduler and it. Replicate data to improve data availability and query response time. Performance is improved because replicated fragment is stored at the nodes where they are frequently needed.

B. SCHEDULING THE QUERY

Proposed System schedules the incoming data, decision of distribution of query are depend upon load balancing algorithm. A number of load balancing algorithm are there like Round Robin (RR), Least Weighted (LW) etc. Proposed system is based on least work Least Work based on the number of queries running (LWRn). This algorithm requires knowledge about the number of queries running at each node, and chooses the node with less queries at the assignment instant. Finally, the

Least Weight (LW) algorithm needs to measure current load in terms of parameters such as CPU, memory and IO in order to determine the less loaded node, then it assigns the query to the less-loaded node.

C. OVERLOAD DETECTION

When a new query arrives at the scheduler it is send to the node with less load. If the queue of the processing node reaches a limit size, then the query is removed from it, and put to run in the ready node, ready node becomes a processing node. Elasticity and scalability is achieved by adding new nodes to the set of ready-nodes.

When a node has a small minimum number of queries and minimum load, the resource is de-provisioned. The node tries to free resources by submitting the queries to the scheduler. If it gets free, the node will be set on standby as a ready-node.

D. EVENT HANDLING AND ALERT

Every time a P/C queue reaches the maximum size (configurable parameter), queries removal or load shedding decisions need to be made. If all the previews options are exhausted and the system is still overloaded it will alert the administrator, indicating the node and queries in overload condition. The administrator can decide to add more ready-nodes, remove more queries.

V. ALGORITHM

In this section we describe the algorithm used in system. Workload refers to database sub queries. In section IV-A, we describe Scheduling algorithm and in Section IV- B we describe load shedding algorithm.

A. Scheduling Algorithm

The following is scheduling algorithm. The variables Times means how many times query was reschedule if it is zero scheduler will schedule it to best node.

Step1: Start
Step2: Accept Query as QUERY
Step3:3.1Check the number of times QUERY has been relocated
3.2 If Relocation Times
Go to step 4
Else if Relocation Times = 1
Go to step 5
Else if Relocation Times = 2
Go to step 6
Step 4: 4.1 Node= the least utilized node from the pool of nodes
4.2 Send QUERY to Node
4.3 Update the Relocation Times of QUERY = 2
4.4 Update the Relocation Times of previous QUERY=1
4.5 Go to step 7
Step 5:5.1 Send QUERY to Node
5.2 Update the Relocation Times of QUERY = 1
5.3 Go to step 7
Step 6: 6.1 Count the number of Ready Nodes available in the Ready Node Pool
6.2 If Ready node is not available
Go to step 7
Else
Send query to any one of the available Ready Node
Go to step 7
Step 7: Stop

B. Load Shedding Algorithm

In this section we design algorithm for handling the overload condition, when overload detected in many node or one node but query location is unable to solve the problem. Algorithm has following steps

Step 1: Start
Step 2: if size of P/C Queue > Assigned maximum size
Go to step 3
Else
Go to step 9
Step 3: Get the target load shedding value
Step 4: If current load shedding val < target load shedding val
Go to step 5
Else
Go to step 6
Step 5: 5.1 Set current load shedding value = minimum of (target load shedding value current load shedding value + 5% of current load shedding value)
5.2 Go to step 9
Step 6: 6.1 Check status of Query to check whether Query dropping is enabled
6.2 If Query drop enabled
Go to step 7
Else
Go to step 8
Step 7: 7.1 Remove Query
7.2 Set current load shedding value = 0
7.3 Go to step 9
Step 8: 8.1 Alert Administrator about failure in load shedding
8.2Go to step 9
Step 9: Stop

VI. EXPERIMENTS

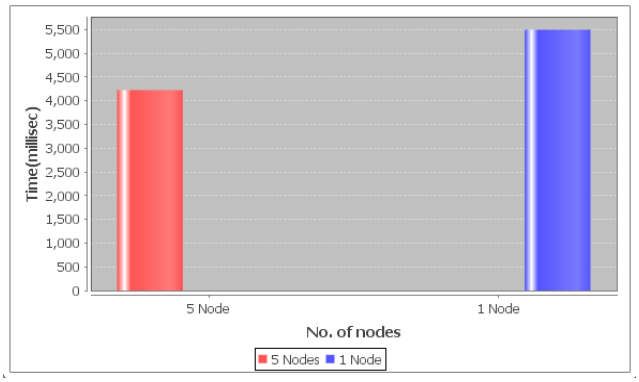
In this Section we describe experimental setup, result and analysis of the system doesn't require any specific hardware to run, and any standard machine is capable of running the application. Different hardware and software tools employed in developing decision making system is
1. Hardware Environment

- Processor – Intel Core or higher version
- RAM – 1 GB or more
- Hard Disk – 100 GB or more

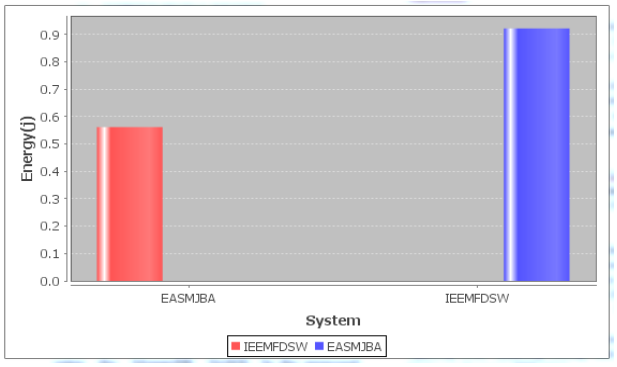
2. Software environment

- Operating System – Linux
- Technology - Java and J2EE
- IDE - Eclipse
- Database – NoSQL

VII. EXPERIMENTAL RESULT



Graph 4.1 Time required to process query



Graph 4.2 Time required to process query

Graph 4.1 shows the time required to process the queries with 5 nodes and single node

Graph 4.2 shows the Energy required to process the query IEEMFDSW consumes less energy than EASMJBA. Total battery power is 19V. And energy consumed for processing query is 0.007A.

Energy in Watt = 19V x 0.007A which is 0.133W.

Energy in joules = 0.133W x 4.221 Sec (time required to execute query) which is 0.561393 j.

VIII. CONCLUSION

MapReduce concept to get the approximate results using the MongoDB NoSQL. In the proposed methodology, a large set of data given to the MongoDB. The MapReduce model is simple to use. It allows scaling of applications across massive clusters of machines comprising thousands of nodes, with fault-tolerance inbuilt for ultra-fast performance.

A solution for any loaded system, is to parallelize the load though many machines or cores, however nodes can still overload. Hence an integrated approach is proposed to increase scalability of query processing with robust architecture for overload mitigation, scalability, various researchers has put forward mechanism for load balancing in networking and cloud environment but this approach provides unique and integrated approach and considers many factors in unison to provide best possible results. In future it is possible to contribute this work to dataware houses.

REFERENCES

[1] Pedro Martins, Maryam Abbasi, Pedro Furtado , “AuDy: Automatic Dynamic Least-Weight Balancing for Stream Workloads Scalability”, 2014 IEEE International Congress on Big Data.

[2] Report on An Evaluation of Load Balancing Algorithms for Distributed Systems” by Kouider Benmohammed-Mahieddine.

[3] L. F. Cabrera, "The Influence of Workload on Load Balancing Strategies," Proc. of the 1986 Summer USENIX Conference, pp. 446-458 (June 1986).

[4] Yunhua Deng, Rynson W.H. Lau, “Heat diffusion based dynamic load balancing for distributed virtual environments”, in: Proceedings of the17th ACM Symposium on Virtual Reality Software and Technology, ACM, 2010.

[5] Bin Dong, Xiuqiao Li, Qimeng Wu, Limin Xiao, Li Ruan, “A dynamic and adaptive load balancing strategy for parallel file system with large-scale I/O servers”, J. Parallel Distribution Computing,2012.

[6] Raj, G. Punjab Tech. Univ., Jalandhar, India Singh, D. Bansal, A. "Load balancing for resource provisioning using Batch Mode Heuristic Priority in Round Robin (PBRR) Scheduling" IEEE Confluence 2013: The Next Generation Information Technology Summit (4th International Conference).

[7] Suchi Johari, Arvind Kumar, “Algorithmic Approach for Applying Load Balancing During Task Migration in Multi-core System”, IEEE 2012.

[8] Parveen Jain, Daya Gupta, “ An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service” ,IJRTE may 2009.

[9] Nithya Kuriakose, Ms. Shinu Acca Mani “Survey on Load Rebalancing For Distributed File Systems in Clouds,

IOSR p- ISSN: 2278-8727Volume 16, Issue 2, Ver. III (Mar-Apr. 2014), PP 81-86.

[10] Suriya Mary, Guru Rani."Survey on Novel Load Rebalancing for Distributed File Systems" , International Journal of Computer Science and Mobile Computing, December 2013.

[11] Suriya Mary , Vairachilai "Dynamic Load Rebalancing by Monitoring the Elastic Map Reduce Service in Cloud", International Journal of Innovative Research in Science, Engineering and Technology, may 2014

[12] Smita Salunkhe, S. S. Sannakki "Load Rebalancing for Distributed File Systems in Clouds" International Journal of Advance Foundation and Research in Computer (IJAFRC) Volume 2,Special Issue (NCRTIT 2015), January 2015. ISSN 2348 – 4853.

[13] Article by Mayanka Katyal, Atul Mishra, "A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment" Article can be accessed online at http://www.publishingindia.com.

[14] Article by Anisn Nasir on **"**Load Balancing in Stream Processing Engines"

[15] 10gen, Inc: mongoDB. 2010. Available http://www.mongodb.org.

[16] Online Aggregation Using MapReduce in MongoDB B RamaMohan A Govardhan Dept. of CSE, JNTUH College of Engineering Hyd Professor, School of Information Technology JNT University Hyderabad, India JNT University Hyderabad, India