Energy Efficient Scheduling of MapReduce over Big Data

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

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

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

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

Abstract**—** The majority of large-scale data intensive applications carried out by information centers are based on MapReduce or its open-source implementation, Hadoop. Such applications are carried out on rich clusters requiring ample amounts of energy, helping the energy costs an appreciable fraction of the data centers overall costs. Therefore, reducing the energy consumption when carrying out each MapReduce task is a critical worry for data centers. In this paper, we advise a framework for mending the energy efficiency of MapReduce applications, while satisfying the (SLA) Service Level Agreement. We first prototype the problem of energy-aware scheduling of a single MapReduce task as an Integer Program. After that we court two algorithms, known as MapReduce scheduling algorithms and load scheduling algorithm, that find the assignments of map and reduce tasks to the machines plenty in order to reduce the energy consumed when carrying out the application. The energy aware configuration and scheduling will improve the energy efficiency of MapReduce clusters thus help in reduction of the service costs of the data-centers.

Keywords- *MapReduce, big data, reducing energy consumption, scheduling*

*****

I. INTRODUCTION

Several businesses and organizations are faced with a never growing need for analyzing the unprecedented a mounts of available data. Such need challenges existing methods, and requires novel approaches and technologies in order to cope with the complexities of big data processing. One of the major challenges of processing data intensive applications is minimizing their energy costs. Electricity used in US data centers in 2010 accounted for about 2 percent of the total electricity used nationwide. In addition, the energy consumed by the data centers is growing at over 15 percent annually, and the energy costs make up about 42 percent of the data centers' operating costs. Considering that server costs are consistently falling, it should be no surprise that in the near future a big percentage of the data centers' costs will be energy costs. Therefore, it is critical for the data centers to minimize their energy consumption when offering services to customers. Big data applications run on large clusters within data centers, where their energy costs make energy efficiency of executing such applications a critical concern. MapReduce and its open-source implementation, Hadoop, have emerged as the leading computing platforms for big data analytics. For scheduling multiple MapReduce jobs, Hadoop originally employed a FIFO scheduler. To over-come the issues with the waiting time in FIFO, Hadoop then employed the Fair Scheduler. These two schedulers, however, do not consider improving the energy efficiency when executing MapReduce applications. Improving energy efficiency of MapReduce applications leads to a significant reduction of the overall cost of data centers. In this paper, we design MapReduce scheduling algorithms that improve the energy efficiency of running each individual application, while satisfying the service level agreement (SLA).

II. LITERATURE SURVEY

[1] The results in the existing system show that make span minimization is not necessarily the best strategy to consider when scheduling MapReduce jobs for energy efficiency in data centers. This is due to the fact that data centers are obligated to deliver the requested services according to the SLA, where such agreement may provide significant optimization opportunities to reduce energy costs. Such reduction in energy costs is a great incentive for data centers to adopt our proposed scheduling algorithms.

Due to the increasing need for big data processing and the widespread adoption of MapReduce and its open source implementation Hadoop for such processing, improving MapReduce performance with energy saving objectives can have a significant impact in reducing energy consumption in data centers. G. Eason, B. Noble, and I. N. Sneddon show that there are significant optimization opportunities within the MapReduce framework in terms of reducing energy consumption. G. Eason, B. Noble, and I. N. Sneddon proposed two energy-aware MapReduce scheduling algorithms, EMRSA-I and EMRSA-II, that schedule the individual tasks of a MapReduce job for energy efficiency while meeting the application deadline. Both proposed algorithms provide very fast solutions making them suitable for execution in real-time settings. G. Eason, B. Noble, and I. N. Sneddon performed experiments on a Hadoop cluster to determine the energy consumption of several MapReduce benchmark applications such as Tera-Sort, Page Rank, and K-means clustering. Then used this data in an extensive simulation study to analyse the performance of EMRSA-I and EMRSA-II. The results showed that the proposed algorithms are capable of obtaining near optimal solutions leading to significant energy savings.

In the future, G. Eason, B. Noble, and I. N. Sneddon plan to design and implement a distributed scheduler for multiple MapReduce jobs with the primary focus on energy consumption.

[2] Most of the current day applications process large amounts of data. There were different trends in computing like mainframes, parallel computing, cluster computing, grid computing as per the requirement of the data size and execution speed. Cloud computing is the new era of computing where efficient utilization of resources can be done with no compromise on data size, execution time and cost of execution. Map Reduce is a programming model which is widely used for processing large scale data intensive applications in cluster, cloud environments. A. Sree Lakshmi, Dr M. BalRaju, Dr N. Subhash Chandra have discussed various scheduling algorithms of map reduce tasks. The default schedulers available with Hadoop can be improved to make it more efficient for the cloud environments.

In current day world as there is huge increase in volumes of data and big data has become an important point of research. This paper has discussed scheduling of Map Reduce parallel applications on cloud. There has been an active research in the area of scheduling of the map and reduce tasks to virtual machines to improve the performance of map reduce applications. Most of the scheduling algorithms concentrate on map tasks data locality. Scheduling can be made efficient by using the knowledge of data locality of the intermediate data generated by the map tasks. This knowledge helps out to reduce the intermediate network traffic during the reduce phase and there by speeding the execution of map reduce applications.

[3] Energy efficiency has become the center of attention in emerging data center infrastructures as increasing energy costs continue to outgrow all other operating expenditures. Nezih Yigitbasi, Kushal Datta, Nilesh Jain and Theodore Willke investigate energy aware scheduling heuristics to increase the energy efficiency of MapReduce workloads on heterogeneous Hadoop clusters comprising both low power (wimpy) and high performance (brawny) nodes. Nezih Yigitbasi, Kushal Datta, Nilesh Jain and Theodore Willke first make a case for heterogeneity by showing that low power Intel Atom processors and high performance Intel Sandy Bridge processors are more energy efficient for I/O bound workloads and CPU bound workloads, respectively. Then present several energy efficient scheduling heuristics that exploit this heterogeneity and real-time power measurements enabled by modern processor architectures. Through experiments on a 23-node heterogeneous Hadoop cluster we demonstrate up to 27% better energy efficiency with our heuristics compared with the default Hadoop scheduler.

[4] Power consumption has become a critical issue in large scale clusters. Existing solutions for addressing the servers' energy consumption suggest "shrinking" the set of active machines, at least until the more power-proportional hardware devices become available. This paper demonstrates that leveraging the sleeping state, however, may lead to unacceptably poor performance and low data availability if the distributed services are not aware of the power management's actions. Therefore, Nedeljko Vasic, Martin Barisits, Vincent Salzgeber, Dejan Kostic present an architecture for cluster services in which the deployed services overcome this problem by actively participating in any action taken by the power management. Nedeljko Vasic, Martin Barisits, Vincent Salzgeber, Dejan Kostic propose, implement, and evaluate modifications for the Hadoop Distributed File System and the MapReduce clone that make them capable of operating efficiently under limited power budgets.

Nedeljko Vasic, Martin Barisits, Vincent Salzgeber, Dejan Kostic demonstrate that important classes of distributed applications do not gracefully operate with limited power budgets. We believe that energy-aware design for cluster applications and services and their active participation in power management actions will be required for reliable, high performance, and low cost data centers. Nedeljko Vasic, Martin Barisits, Vincent Salzgeber, Dejan Kostic therefore propose a new approach for making cluster applications energy aware, and demonstrate the efficiency of our approach using a prototype implementation of HDFS and MapReduce.

III. Proposed Work

Our system tends 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, event generation and handling, job submission control.

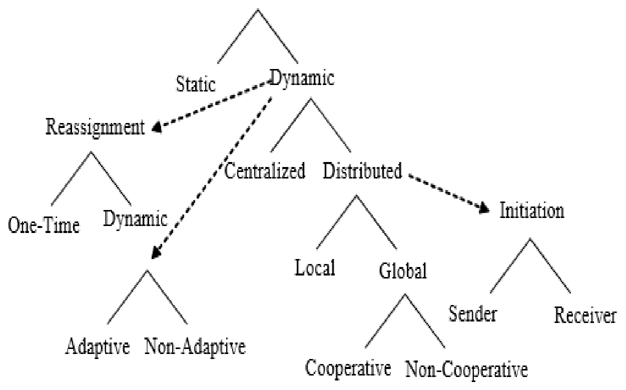

Figure 1

- Scheduling of Query
- Load Scheduling and Event generation

A. 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 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.

B. 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.

C. EVENT HANDLING AND ALERT

Every time a P/C queue reaches the maximum size (configurable parameter), queries removal or load scheduling 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.

D. 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 scheduling algorithm.

1. 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.

```
Step 1. Start
Step 2. Accept Query as QUERY
Step 3. 3.1. Check 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
```

2. Load Scheduling 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 scheduling value
Step 4. If current load scheduling val < target load scheduling
           val
        Go to step 5
        Else
        Go to step 6
Step 5. 5.1. Set current load scheduling value = minimum of
             (target load scheduling value current load
             scheduling value + 5% of current load
             scheduling 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 scheduling value = 0
        7.3. Go to step 9
Step 8. 8.1. Alert Administrator about failure in load
           scheduling
        8.2. Go to step 9
Step 9. Stop
```

3. EMRSA-X

```
1. Create an empty priority queue Q^m
2. Create an empty priority queue Q^r
3. for all j ∈ Ado
4. $ecr_j^m = \min_{\forall i \in M} \frac{e_{ij}}{p_{ij}}$, for EMRSA-I; or
   $ecr_j^m = \frac{\sum \forall i \in M \frac{e_{ij}}{p_{ij}}}{M}$, for EMRSA-II
5. $Q^m.enqueue(j, ecr_j^m)$
6. for all j ∈ B do
7. $ecr_j^m = \min_{\forall i \in R} \frac{e_{ij}}{p_{ij}}$
   $ecr_j^m = \frac{\sum \forall i \in R \frac{e_{ij}}{p_{ij}}}{R}$
8. $Q^r.\ enqueue(j, ecr_j^r)$
9. $D^m \leftarrow \infty;$     $D^r \leftarrow \infty;$
10. while Q^m is not empty and Q^r is not empty, do
```

```
11. $j^m = Q^m.extractMin()$
12. $j^r = Q^r.extractMin()$
13. $f = \frac{\sum \forall i \in M^{p_i j^m}}{\sum \forall i \in R^{p_i j^r}}$
14. $T^m$ : sorted unassigned map tasks i ∈ M based
        on $p_i j^m$.
15. $T^r$ : sorted unassigned reduce tasks i ∈ R based
        on $p_i j^r$ .
16. if $T^m = \phi$ and $T^r = \phi$ then break
17. ASSIGN-LARGE ()
18. ASSIGN-SMALL ()
19. if $D^m = \infty$ then
20.     $D^m = D - p^r$
21.     $D^r = p^r$
22. if $T^m \neq \phi$ or $T^r \neq \phi$ then
23. No feasible schedule
24. return
25. Output: X, Y
```

The ordering induced by these metrics on the set of slots determines the order in which the slots are assigned to tasks, that is, a lower ecr_j^m means that slot j has a higher priority to have a map task assigned to it. Similarly, a lower ecr_j^m means that slot j has a higher priority to have a reduce task assigned to it.

In addition, EMRSA-X uses the ratio of map and reduce processing times, denoted by f, in order to balance the assignment of map and reduce tasks. The ratio f is defined as follows:

$$f = \frac{\sum \forall i \in M^{p_i j^m}}{\sum \forall i \in R^{p_i j^r}}$$

This ratio is used in the task assignment process in each iteration of EMRSA-X. As we already mentioned, we use job profiling of production jobs to estimate the processing time of map and reduce tasks. This information, extracted from job profiling (i.e., the values of $p_i j^m$ and $p_i j^r$) is used by EMRSA-X to compute the ratio f.

4. ASSIGN-LARGE ()

```
1. $i^m = \arg\min_{t \in T^m} p_{tj^m}$
2. $i^r = \arg\min_{t \in T^r} p_{tj^r}$
3. $p^m = 0$; $p^r = 0$
4. if $p_{i^m j^m} + p_{i^r j^r} \leq D$ and $p_{i^m j^m} \leq D^m$ and
       $p_{i^r j^r} \leq D^r$ then
5. $T^m = T^m \setminus \{i\}$
6. $T^r = T^r \setminus \{i\}$
7. $p^m = p_{i^m j^m}$
8. $p^r = p_{i^r j^r}$
9. $X_{i^m j^m} = 1$
10. $Y_{i^r j^r} = 1$
11. do
12. $i^m = \arg\min_{t \in T^m} p_{tj^m}$
13. $i^r = \arg\min_{t \in T^r} p_{tj^r}$
14. if $f > 1$ then
15. while $\frac{p^m + p_{i^m j^m}}{p^r} < f$ and $p^m + p^r + p_{i^m j^m} \leq D$
        and $p^m + p_{i^m j^m} \leq D^m$ and $T^m \neq \phi$ do
16. $T^m = T^m \setminus \{i\}$
17. $p^m = p^m + p_{i^m j^m}$
18. $X_{i^m j^m} = 1$
19. $i^m = \arg\min_{t \in T^m} p_{tj^m}$
20. Balance the assignment of reduce tasks (repeat lines
        15-19 for reduce tasks).
21. else
22. The code for $f < 1$ is similar to lines 15-20 and is
        not presented here.
23. while $p^m + p^r + p_{i^m j^m} + p_{i^r j^r} \leq D$ and
        $p^m + p_{i^m j^m} \leq D^m$ and $p^r + p_{i^r j^r} \leq D^r$ and
        ($T^m \neq \phi$ or $T^r \neq \phi$)
```

5. ASSIGN-SMALL ()

```
1. {Assign small map tasks}
2. $i = \arg\min_{t \in T^m} p_{tj^m}$
3. while $p^m + p^r + p_{ij^m} \leq D$ , $p^m + p_{ij^m} \leq D^m$ &
       $T^m \neq \phi$ do
4. $T^m = T^m \setminus \{i\}$
5. $p^m = p^m + p_{ij^m}$
6. $X_{ijm} = 1$
7. $i = \arg\min_{t \in T^m} p_{tj^m}$
8. {Assign small reduce tasks}
9. $i = \arg\min_{t \in T^r} p_{tj^r}$
10. while $p^m + p^r + p_{ij^r} \leq D$ , $p^m + p_{ij^m} \leq D^r$ &
       $T^r \neq \phi$ do
11. $T^r = T^r \setminus \{i\}$
12. $p^r = p^r + p_{ij^r}$
```

13. $Y_{ij^r} = 1$

14. $i = \arg\min_{t \in T^r} {}_{ptj^r}$

IV. CONCLUSION

A solution for any big data system is to MapReduce and parallelizes 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, Query processing control, 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 data ware houses.

V. Acknowledgment

I wish to express my sincere thanks to the guide Prof. Jayashree Chaudhari and Head of Department, Prof. Soumitra Das, as well as our director Dr. S.S. Sonavane and last but not least, the staff members of Computer Department for their support.

VI. REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] Scheduling of Parallel Applications Using Map Reduce On Cloud: A Literature Survey -A.Sree Lakshmi, Dr.M.BalRaju, Dr.N.Subhash Chandra

[3] Energy Efficient Scheduling of MapReduce Workloads on Heterogeneous Clusters -NezihYigitbasi, KushalDatta, Nilesh Jain and Theodore Willke

[4] Making Cluster Applications Energy-Aware -NedeljkoVasic, Martin Barisits,Vincent Salzgeber, DejanKostic