DOI 10.29042/2019-4864-4869

Helix Vol. 9 (2): 4864- 4869

A Novel Priority Based Hadoop Energy Efficient Job Scheduling and Migration Technique with Multi Level Queue on YARN Scheduler

[1]G. Joel Sunny Deol, [2]Dr.O.NagaRaju

[1]Research Scholar, Dept of CSE, Acharya Nagarjuna University

[2]Asst. Professor, Department of Computer Science, Government Degree College, Macherla-522426

Email: sunnydeolgosu@gmail.com

Received: 18th March 2019, Accepted: 10th April 2019, Published: 30th April 2019

Abstract

The process or the framework for MapReduce works in two parts as Mapper and Reducer. The reducer algorithm analyzes the input from the tasks characteristics and generate recommendations for the applicable allocation of the work and the mapper algorithm analyses the perfect or the best fit for the task or the programming running on the Hadoop clusters. The primary challenge is to manage the migration of the virtual machines to make these arrangements suitable to the Hadoop scheduling capabilities. Hence the demand from the research is to justly the Hadoop scheduling capabilities and test the performances of the scheduler strategies for diversified workloads. Also, it is important to design a virtual machine migration algorithm to justify the demands of low power consumptions. Accordingly, this work also coined an energy efficient technique for Hadoop MapReduce jobs scheduling and migration technique. The work results into a novel algorithm and provide significant improvement of the energy consumption. The outcome of the work also analyzes the improvement of other performance parameters like identification of ill-scheduled job and total execution time. This work demonstrates a significant 30% reduction of energy with nearly 40% reduction in job identification and migration time.

Keywords

Hadoop, MapReduce, YARN, FIFO, FAIR, Capacity, Hybrid, LATE, HDFS, Multi-Level Queue, Allocation Policy

Introduction

The Apache Hadoop framework provides the best mashup of data and processing capabilities. The highest rated demand of enterprise applications like robustness and scalabilities can be achieved through Hadoop [Fig – 1] provided highly reliable and low cost storage structures [1]. The storage provided is HDFS enabling to host a large amount of data and process the jobs using MapReduce framework enabling the distributed and parallel processing of the Job [2]. The MapReduce framework [Fig – 2] is first introduced by Google in the early 2000. The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types [3]. MapReduce libraries have been written in many programming dialects, with various dimensions of advancement. A prominent open-source execution that has bolster for disseminated rearranges is a piece of Apache Hadoop. The name MapReduce initially alluded to the restrictive Google innovation, yet has since been genericized [4]. The decrease work takes the yield from a guide as info and joins those information tuples into a littler arrangement of tuples. As the arrangement of the name MapReduce infers, the diminish work is constantly performed after the guide work [5].

.

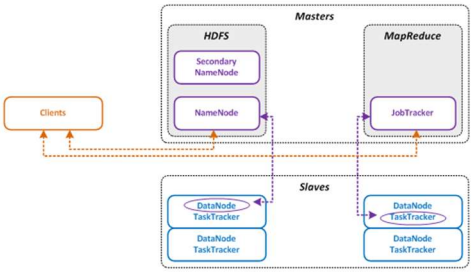

Figure 1: Hadoop Architecture

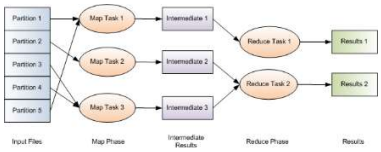

Figure 2: MapReduce Job Processing

The utilization of custom JavaScript capacities give adaptability to outline tasks. For example, when handling a report, the guide capacity can make more than one key and esteem mapping or no mapping. Guide decrease activities can likewise utilize a custom JavaScript capacity to make last changes to the outcomes toward the finish of the guide and lessen task, for example, play out extra computations [6]. As indicated by The Apache Software Foundation, the essential target of Map/Reduce is to part the information informational index into free lumps that are prepared in a

 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

totally parallel way. The Hadoop MapReduce system sorts the yields of the maps, which are then contribution to the lessen errands. Regularly, both the info and the yield of the activity are put away in a record framework [7].

Related Works

FIFO Scheduler: Schedules the tasks with the relevancy of arrival to the job queue. [6].

The FIFO scheduler performance needs to be analyzed on the following criterions:

Parameter Name	Performance Measure
Response Time	Expected to be High for better performance in Hadoop framework.
Job Heterogeneity	Expected to be heterogeneous in job nature.

The challenges identified for FIFO are been addressed in other advanced scheduling techniques.

Fair Scheduler: Reasonable planning is a technique for allocating assets to employments with the end goal that all occupations get, by and large, an equivalent offer of assets after some time. At the point when there is a solitary occupation running, that activity utilizes the whole group. At the point when different occupations are submitted, undertakings openings that free up are doled out to the new employments, so each employment gets generally a similar measure of CPU time. Dissimilar to the default Hadoop scheduler, which shapes a line of occupations, this gives short employments a chance to complete in sensible time while not starving long occupations. It is likewise a simple method to share a group between various of clients. Reasonable sharing can likewise work with occupation needs - the needs are utilized as loads to decide the part of all out process time that each activity gets [7 - 9].

The performance of the Fair Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Number of Pool	Expected to be equal to the number of users or number of clusters.
Number of Concurrent Jobs per Pool	Expected to be higher for maximum resource utilization.

The challenges identified for FAIR are been addressed in other advanced scheduling techniques.

Capacity Scheduler: The Capacity Scheduler is intended to permit sharing a vast group while giving every association a base limit ensure. The focal thought is that the accessible assets in the Hadoop Map-Reduce bunch are apportioned among various associations who on the whole store the group dependent on figuring needs. There is an additional advantage that an association can get to any overabundance limit no being utilized by others. This gives flexibility to the associations in a savvy way [10 - 12].

The performance of the Capacity Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Number of Queues	Expected to be equal to the number of users or number of clusters.
Number of Concurrent Jobs per Queues	Expected to be higher for maximum resource utilization.

The challenges identified for Capacity Scheduler are been addressed in other advanced scheduling techniques.

Dynamic Priority Based Hybrid Scheduler: Assign the tasks to the workload manager based on the increasing priority submitted by the user or based on the wait time. [12].

The performance of the Dynamic Priority Based Hybrid Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Waiting Time	Expected to be minimum concerning priority
Job Priority	Expected to be heterogeneous for the set of priorities extracted from the jobs
Job Locality	Expected to be higher to achieve the fundamental principle of data locality

The challenges identified in the framework are been addressed in proposed technique.

LATE Scheduler: LATE (Longest Approximate Time to End) scheduler executes SE which backs up the errand expected to finish last [13].

The performance of the LATE Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Remaining Time to Complete	Expected to be minimum concerning to priority
Job Priority	Expected to be heterogeneous for the set of priorities extracted from the jobs
Job Locality	Expected to be higher to achieve the fundamental principle of data locality

The challenges identified for LATE are been addressed in other advanced scheduling techniques.

SAMR Scheduler: The work of Chen at al. [14] utilizes the Hadoop node characteristics like processing speed.

The performance of the SAMR Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Total Job Completion Time	Expected to be minimum concerning to priority
Job Priority	Expected to be heterogeneous for the set of priorities extracted from the jobs

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

The challenges identified for SAMR are been addressed in other advanced scheduling techniques.

Delay Scheduler: Another work of Zaharia et al. [15] exhibits the activity booking for the particular employments where the guideline of region can't be acquired as the occupations asked for at a few bunches where the information is absent privately called Delay scheduler.

The performance of the Delay Scheduler needs to be identified based on the following parameters:

Parameter Name	Performance Measure
Waiting Time	Expected to be minimum concerning to priority
Job Locality	Expected to be higher to achieve the fundamental principle of data locality

The challenges identified for Delay Scheduler are been addressed in other advanced scheduling techniques.

Context Aware Scheduler: The work of Kumar et al. [16] demonstrates the context aware scheduler.

The performance of the Context Aware Scheduler needs to be identified based on the following parameters [17] [18]:

Parameter Name	Performance Measure
Job Priority	Expected to be heterogeneous for the set of priorities extracted from the jobs
Total Job Completion Time	Expected to be minimum concerning to priority
Total number of Job Migrations	Expected to be maximum for better results
Total Energy Cost	Expected to be minimum
Waiting Time	Expected to be minimum based on the job priority
Job Locality	Expected to be higher to achieve the fundamental principle of data locality

The challenges identified for Context Ware Scheduler are been addressed in novel proposed framework.

Performance Evaluation Matrix

In this section of the work, the novel proposed performance evaluation metric is proposed [Table 1]:

Parameter	Details
Local Storage cost	The local cost of the storage systems in case of locality of the job is given preference. The value is extracted from the File System Counters by the parameter FILE_BYTES_READ and FILE_BYTES_WRITTEN
HDFS Storage cost	The local cost of the storage systems in case of locality of the job is not given preference. The value is extracted from the File System Counters by the parameter HDFS_BYTES_READ and HDFS_BYTES_WRITTEN
Map Job Waiting Time	The Jobs waiting to get appropriate slots. The value is extracted from the Job Counters by the parameter FALLOW_SLOTS_MILLIS_MAPS & SLOTS_MISSIS_MAPS
Reduce Job Waiting Time	The Jobs waiting to get appropriate slots. The value is extracted from the Job Counters by the parameter FALLOW_SLOTS_MILLIS_REDUCES & SLOTS_MISSIS_REDUCES
Total Job Time	Total Job execution time is been calculated. The value is extracted from the Job Counters by the parameter SLOTS_MISSIS_MAP & SLOTS_MISSIS_REDUCES
Total Job waiting Time	Total Job waiting time is been calculated. The value is extracted from the Job Counters by the parameter FALLOW _SLOTS_MISSIS_MAP & FALLOW _SLOTS_MISSIS_REDUCES

Table 1: YARN - Performance Evaluation Matrix and Parameters

The migration algorithms are also expected to be evaluated against the other methods, thus another metric is proposed here [Table 2]:

Parameter	Details	Optimality Expectation
Hosts	Number of physical systems in the data centers	-
VMs	Virtual Machines utilized	-
Simulation Duration	Time for the observation	-
Energy Measurement	Measurement of energy consumed during the migration process	Low
Selection time	Source physical machine selection time	Low
Reallocation Time	Source to destination node migration time	Low
Total execution time	Duration of the total migration process	Low

Table 2: Job Migration - Performance Evaluation Matrix and Parameters

The results of this experiment are been discussed over the same parameters.

Experimental Setup

The Experimental setup demonstrated in this experiment is classified into two major categories as:

Capacity Scheduler Setup: During the configuration of Capacity Scheduler, the below configurations are been utilized. capacity-scheduler.xml file configuration for three test users as root1, TestUser1, TestUser2

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

```
<property>
<name>yarn.scheduler.capacity.maximum-applications</name>
<value>10000</value>
<description>
    Maximum number of applications that can be pending and running.
</description>
</property>
<property>
<name>yarn.scheduler.capacity.maximum-am-resource-percent</name>
<value>0.1</value>
<description>
    Maximum percent of resources in the cluster which can be used to run
    application masters i.e. controls number of concurrent running
    applications.
</description>
</property>
<!-- New Queue Code-->
<property>
<name>yarn.scheduler.capacity.root.queues</name>
<value>alpha,beta,default</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.alpha.capacity</name>
<value>20</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.beta.capacity</name>
<value>30</value>
</property>
<property>
<name>yarn.scheduler.capacity.root.default.capacity</name>
<value>50</value>
</property>
<property>
<name>yarn.scheduler.capacity.queue-mappings</name>
<value>u:TestUser1:alpha,u:TestUser2:beta</value>
</property>
<property>
<name>yarn.scheduler.capacity.queue-mappings-override.enable</name>
<value>false</value>
</property>
<!-- End New Queue Code-->
<property>
<name>yarn.scheduler.capacity.resource-calculator</name>
<value>org.apache.hadoop.yarn.util.resource.DefaultResourceCalculator</value>
<description>
    The ResourceCalculator implementation to be used to compare
    Resources in the scheduler.
    The default i.e. DefaultResourceCalculator only uses Memory while
DominantResourceCalculator uses dominant-resource to compare
    multi-dimensional resources such as Memory, CPU etc.
</description>
</property>
</configuration>
```

The experimental setup is visualized here:

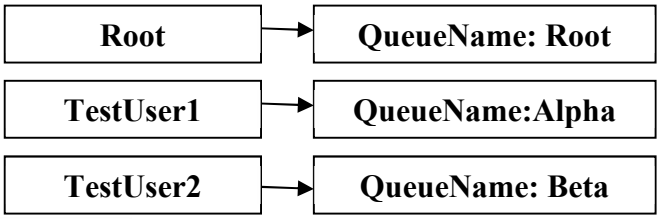

Figure 3: Capacity Scheduler Multi Queue Setup

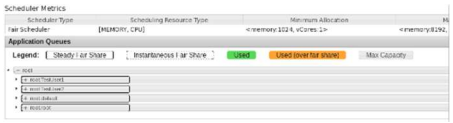

Figure 4: Fair Scheduler Multi Queue Setup

Results and Discussion

The multilevel queue configuration follows the below acronyms for Hadoop YARN scheduler scheduling techniques [Table 3]

Used Name in this Work	Scheduler Technique	Scheduling Policies	Number of Concurrent Users	Number of Concurrent Jobs per User
CS - User:1 - Job:1	Capacity Scheduler	First In First Out	1	1
CS - Users: 2 - Jobs: 2	Capacity Scheduler	First In First Out	2	2
CS - Users: 3 - Jobs: 3	Capacity Scheduler	First In First Out	3	3
FS - User:1 - Job:1 - FIFO	Fair Scheduler	First In First Out	1	1
FS - Users: 2 - Jobs: 2 - FIFO	Fair Scheduler	First In First Out	2	2
FS - Users: 3 - Jobs: 3 - FIFO	Fair Scheduler	First In First Out	3	3
FS - User:1 - Job:1 - FAIR	Fair Scheduler	Fair	1	1
FS - Users: 2 - Jobs: 2 - FAIR	Fair Scheduler	Fair	2	2
FS - Users: 3 - Jobs: 3 - FAIR	Fair Scheduler	Fair	3	3

Table 3: List of Job / Task Scheduling Techniques

Firstly, this work demonstrates the results of configuration with the first type scheduler on the proposed metric[Table 4]:

Experiment Name	Storage_Cost (KB)	HDFS_Storage _Cost (KB)	Map Job Waiting Time (MSec)	Reduce Job Waiting Time (MSec)	Total_Job_Time (MSec)	Total Job waiting Time (MSec)
CS - User:1 - Job:1	8494	208	0	0	1011018	0
FS - User:1 - Job:1 - FIFO	8493	208	0	0	2194453	0
FS - User:1 - Job:1 – FIFO	8493	208	0	0	2510011	0

Table 4: Performance Comparison for Configuration Set – 1

 Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

Henceforth, it is natural to realize the improvements in case of CS for the specified settings [Fig - 5].

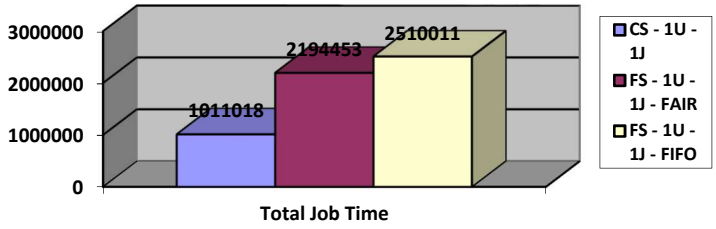

Figure 5: CS User:1 -Job:1, FS-User:1 -Job:1 – FIFO and FS - User:1-Job:1–FAIR Setup Job Completion Time

Next, this work demonstrates the results of configuration with second type scheduler on the proposed metric [Table 5] and [Table 6]:

Experiment Name	Storage_Cost (KB)	HDFS_Storage_Cost (KB)	Map Job Waiting Time (MSec)	Reduce Job Waiting Time (MSec)	Total_Job_Time (MSec)	Total Job waiting Time (MSec)
CS - Users: 2 - Jobs: 2	8494	208	0	0	1562530	0
FS - Users: 2 - Jobs: 2 – FIFO	8493	208	0	0	2217508	0
FS - Users: 2 - Jobs: 2 – FIFO	4595	123	0	0	1160195	0

Table 5: Performance Comparison for Configuration Set – 2 (User – 1)

Experiment Name	Storage_Cost (KB)	HDFS_Storage_Cost (KB)	Map Job Waiting Time (MSec)	Reduce Job Waiting Time (MSec)	Total_Job_Time (MSec)	Total Job waiting Time (MSec)
CS - Users: 2 - Jobs: 2	4595	123	0	0	556563	0
FS - Users: 2 - Jobs: 2 – FIFO	4595	123	0	0	1207336	0
FS - Users: 2 - Jobs: 2 – FIFO	8493	208	0	0	1738994	0

Table 6: Performance Comparison for Configuration Set – 2 (User – 2)

It is natural to realize that the configuration works better for CS and FS [Fig - 6] and [Fig – 3].

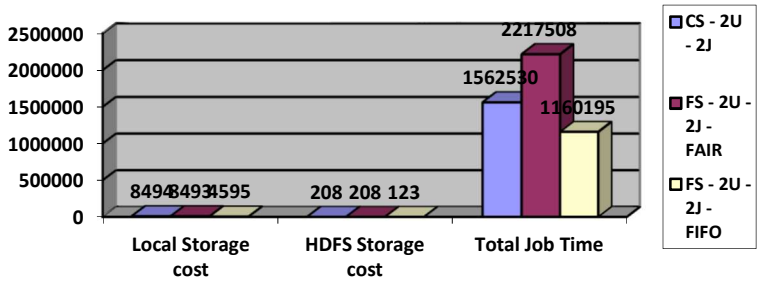

Figure 6: CS-Users: 2-Jobs: 2, FS-Users: 2-Jobs: 2–FIFO and FS-Users: 2-Jobs: –FAIR Setup Local Storage Cost, HDFS Storage Cost and Job Completion Time–User

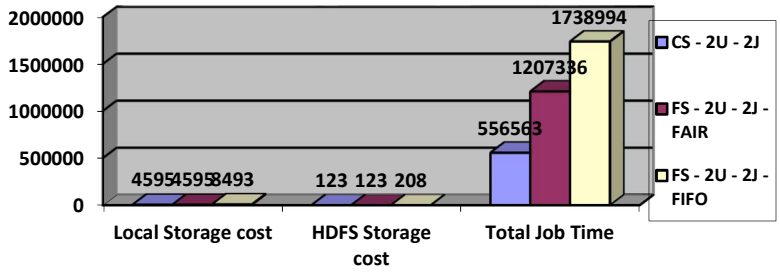

Figure 7: CS-Users: 2-Jobs: 2, F -Users: 2-Jobs: 2–FIFO and FS-Users: 2-Jobs: 2–FAIR Setup Local Storage Cost, HDFS Storage Cost and Job Completion Time–User 2

Conclusion

The work exhibits the current methods with comprehension of the importance in time and information multifaceted nature worldview like FIFO scheduler, Fair Scheduler, Capacity Scheduler, Dynamic Priority Based Hybrid Scheduler,

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495

LATE Scheduler, SAMR Scheduler, Delay Scheduler and Context Aware Scheduler. The not well apportioned Job Identification calculation is been tried and the work delivers considerable lot of test results to demonstrate the curiosity and enhancement. The outcome demonstrates almost a huge 30% enhancement of vitality utilization over the current procedures and 88.89% lesser than the current systems alongside a critical 70% enhancement over the current methods for Job Reallocation Time. Critical accomplishment of this work additionally incorporates the begetting the novel execution assessment lattice for near comprehension of the proposed and existing procedure. The ultimate result of the work exhibits the centrality of line booking to accomplish comparative execution.

References

1. N Nandakumar and Y. Nandita, " A Survey on Data Mining Algorithms on Apache Hadoop Platform", International Journal of Emerging Technology and Advanced Engineering, Vol. 4, NO. 1, January 201, pp. 563-566.
2. Z. Tang, L. Jiang, J. Zhou, K. Li, and K. Li, "A self-adaptive scheduling algorithm for reduce start time ", Future Generation Computer Systems, 2014.
3. 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, 2017, pp.507–518.
4. 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.
5. J. Dean and S. Ghemawat, "Mapreduce: simplied data processing on large clusters", in OSDI 2004: Proceedings of 6th Symposium on Operating System Design and Implemention, (New York), ACM Press, 2017, pp. 137–150.
6. Hadoop, "Hadoop home page."http://hadoop.apache.org/.
7. Hadoop's Fair Scheduler. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.
8. B.P Andrews and A. Binu, " Survey on Job Schedulers in Hadoop Cluster ", IOSR Journal of Computer Engineering, Vol.15, NO. 1, Sep. - Oct. 2016, pp. 46-50.
9. The Apache Hadoop Project. http://www.hadoop.org.
10. J. Chen, D. Wang and W. Zhao, " A Task Scheduling Algorithm for Hadoop Platform ", JOURNAL OF COMPUTERS, VOL. 8, NO. 4, APRIL 2016, pp. 929-936.
11. N. Tiwari, " Scheduling and Energy Efficiency Improvement Techniques for Hadoop Mapreduce: State of Art and Directions for Future Research (Doctoral dissertation, Indian Institute of Technology, Bombay Mumbai).
12. P. Nguyen, T. Simon, M. Halem, D. Chapman and Q. Le, " A hybrid scheduling algorithm for data intensive workloads in aMapReduce environment", In: Proceedings of the 2012 IEEE/ ACM fifth international conference on utility and cloud computing. Washington, DC,USA: IEEE computer society; UCC'12, 2015, p. 161-168.
13. M. Zaharia, A. Konwinski, A.D. Joseph, R.Katz and I. Stoica, "Improving MapReduce performance in heterogeneous environments "In: OSDI 2008: 8th USENIX Symposium on Operating Systems Design and Implementation 2012.

  Helix E-ISSN: 2319-5592; P-ISSN: 2277-3495