A Novel Multilevel Queue based Performance Analysis of Hadoop Job Schedulers

C. Sreedhar¹*, N. Kasiviswanath¹ and P. Chenna Reddy²

¹Department of Computer Science and Engineering, G Pulla Reddy Engineering College, Kurnool - 518007, Andhra Pradesh, India; csrgprec@gmail.com, nkasivaswanath@yahoo.com
²Department of Computer Science and Engineering, JNT University Anantapur, Anantapuram - 515002, Andhra Pradesh, India; pcreddy1@rediffmail.com

Abstract

**Objectives:** In this paper, we discuss on the importance of multilevel queues in scheduling Hadoop mapreduce jobs.

**Methods/Statistical analysis:** Modifications are done on HDFS and yarn configuration files to suit the multilevel queues. This work constitutes the performance analysis of various existing job schedulers such as FIFO, Fair and Capacity schedulers.

**Findings:** Significant achievements are achieved which includes performance evaluation metrics for comparative understanding of the proposed and existing techniques. The final outcome of the work demonstrates the need for multilevel queue scheduling with allocation policies and the optimal placement of jobs in queues.

**Application/Improvements:** With the adoption of multilevel queue scheduling, there is a significant improvement in placing jobs in multilevel queues for the jobs submitted by the users.

**Keywords:** Capacity, Fair, FIFO, Hadoop, Mapreduce, Multilevel Queues

1. Introduction

Big data has become the recent buzzword due to its capabilities of handling huge volumes of data. Over the past decade, the amount of data created every year is exponentially growing and at a faster rate. 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, highly reliable and low cost storage structures. Hadoop is the product of advances in development of parallel, grid and distributed technologies. The MapReduce framework was first introduced by Google in the early 2000. MapReduce is a framework for processing parallelizable problems across volumes of data using clusters of nodes. The jobs submitted by the users in MapReduce framework are divided into map and reduce jobs. The map jobs are handled by process sharing queues and the reduce jobs are handled by the server sharing queues. After submitting the jobs into the MapReduce framework the map and the reduce jobs have been allocated to the node level slots for the execution based on the availability. The Hadoop framework encounters three major challenges as, locality defined the distance between the job and the data for the job to processes, synchronization defined synchronization of the data from the map and reduce jobs and finally the fairness defined comparative fairness between the locality and the synchronization. However, the most critical challenge is the fairness of the framework. Many algorithms have been defined in order to solve this problem. Few of them have been understood and analyzed in later part of this work.

Hence this work demonstrates the use of multilevel queues for YARN based scheduling techniques to enhance the job performance.

The rest of this paper is organized as follows. Section 2 highlights the related work in the literature for job schedulers in Hadoop. The parameters to be considered for evaluating performance metrics of job schedulers are illustrated in Section 3. The experimental setup is provided

*Author for correspondence*
in Section 4. Results and Discussion are illustrated in Section 5. The conclusion and future enhancements are discussed in Section 6.

This study will help in identifying the significance of the proposed framework and will generate the parameters for performance comparison. The scheduling of the work is the most problem-resolving component in MapReduce. However, the ill management of the scheduling techniques may lead to high time complexity, low data locality or synchronization problems. In this work, we analyze the existing algorithms and techniques available in Hadoop framework with understanding of the significance in time and data complexity paradigm.

1.1 FIFO Scheduler

FIFO is the default job scheduler in Hadoop framework and schedules the jobs based on the arrival into the queue. The job considered being the oldest and residing in to the queue would be scheduled first. The FIFO scheduler performance needs to be analyzed on the following criterions as described in Table 1. The disadvantages of FIFO are to serve one job at one time so that small jobs have to suffer from starvation.

1.2 Fair Scheduler

The basic principle of Fair scheduler is to identify the available map - reduce slots and then allocate the slots to the queued jobs with a fair sharing mechanism such that all the jobs get same amount of CPU times. The objective of the Fair scheduler algorithm is to do an equal distribution of computing resources among users/jobs in the system. The Fair scheduler deploys resource pools for each user and sometimes for each cluster. The fair scheduler can limit the number of concurrently running jobs from each user and from each pool. If there is a single job running, the job uses the entire cluster.

The performance of the Fair Scheduler needs to be identified based on the following parameters as shown in Table 2:

1.3 Capacity Scheduler

The fundamental technique used for capacity scheduler is very similar to the Fair scheduler with a difference; jobs are allocated to the queues instead of pools. The configuration of the queues is such that when a job is received to YARN, the YARN searches for the queue with the highest available capacity and assigns the job. Also the YARN for capacity scheduler re-assigns the job based on the availability of the memory in the queues. Workload energy aware All-in-Strategy proposes a balanced workload of jobs submitted by the user in energy aware perspective.

The performance of the Capacity Scheduler is identified based on the following parameters as shown in Table 3:

---

**Table 1. Criterions for the performance of FIFO scheduler**

| Parameter Name | Performance Measure | FIFO Scheduler Measure / Contra |
|----------------|----------------------|---------------------------------|
| Response Time  | Expected to be High for better performance in Hadoop framework. | The response time is significantly low for jobs with small Reduce components. |
| Job Heterogeneity | Expected to be heterogeneous in job nature. | The performance is better in homogeneity compared to heterogeneity. |

**Table 2. Criterions for the performance of fair scheduler**

| Parameter Name | Performance Measure | Fair Scheduler Measure / Contra |
|----------------|----------------------|---------------------------------|
| Number of Pool | Expected to be equal to the number of users or number of clusters. | Job heterogeneity and the locality of the data decide the performance measure of the framework. |
| Number of Concurrent Jobs per Pool | Expected to be higher for maximum resource utilization. | Higher number of concurrent jobs increases the amount of data transfer for shared resource jobs. |

**Table 3. Criterions for the performance of capacity scheduler**

| Parameter Name | Performance Measure | Capacity Scheduler Measure / Contra |
|----------------|----------------------|-----------------------------------|
| Number of Queues | Expected to be equal to the number of users or number of clusters. | Job heterogeneity and the locality of the data decide the performance measure of the framework. |
| Number of Concurrent Jobs per Queues | Expected to be higher for maximum resource utilization. | Higher number of concurrent jobs increases the amount of data transfer for shared resource jobs. |
1.4 Dynamic Priority based Hybrid Scheduler

Dynamic priority based hybrid scheduler is another scheduling technique based on the job priority to reduce the waiting time of the job with higher priority. High priority jobs are the limitations in this algorithm due to locality of jobs in case of over allocations of the internal queues. This algorithm is designed for data intensive workloads and tries to maintain locality of data in the process of job execution.

The performance of the Dynamic Priority Based Hybrid Scheduler needs to be identified based on the following parameters as shown in Table 4:

1.5 LATE Scheduler

Longest Approximate Time to End (LATE) for the heterogeneous Hadoop jobs is a technique based on the shortest job first algorithm. In this framework submitted and slow running jobs are analyzed based on the remaining time to complete and the jobs are scheduled to higher ranked nodes for faster competition. Nevertheless, the localities of the jobs are certainly being overlooked to prioritize the job competition time.

The performance of the LATE Scheduler needs to be identified based on the following parameters as shown in Table 5:

2. Performance Evaluation Metrics

The following parameters are to be considered for the job schedulers. Table 6 proposes the performance evaluation and the optimality expectations.

3. Experimental Setup

The experimental setup demonstrated in this experiment is classified into two major categories: Capacity scheduler setup and Fair scheduler setup. Modifications are done on the following configuration files to suit to the Multilevel Queues namely: hdfs-site.xml, yarn-site.xml, capacity-scheduler.xml file configuration for three test users as root1, TestUser1 and TestUser2 as shown in Figure 3.

The allocation rule can be identified here, where the jobs submitted by the respective users will be allocated to the specified queue by default. Figure 2 depicts capacity scheduler multilevel queue setup.

| Parameter Name | Performance Measure | Dynamic Priority Based Hybrid Scheduler Measure / Contra |
|----------------|---------------------|--------------------------------------------------------|
| Waiting Time   | Expected to be minimum based on the job priority | Job heterogeneity and the locality of the data decide the performance measure of the framework. |
| Job Priority   | Expected to be heterogeneous for the set of priorities extracted from the jobs | Higher priority of the jobs are considered to be a limitation for the job locality in case of over allocation of the local queues. |
| Job Locality   | Expected to be higher to achieve the fundamental principle of data locality | In case with high number of jobs in queues, higher priority jobs are not given priority in this framework. |

| Parameter Name | Performance Measure | LATE Scheduler Measure / Contra |
|----------------|---------------------|--------------------------------|
| Remaining Time to Complete | Expected to be minimum based on the job priority | Job heterogeneity and locality of the data decide the performance measure of the framework. |
| Job Priority | Expected to be heterogeneous | Higher priority jobs are not given priority due to job locality. |
| Job Locality | Expected to be higher to achieve the fundamental principle of data locality | High priority jobs are to be given priority based on the needs with jobs of higher priority can be run immediately. |

During the Fair scheduler the following modifications are done on the configuration files to suit to the Multilevel Queues namely: hdfs-site.xml, yarn-site.xml, capacity-scheduler.xml file configuration for three test users as root1, TestUser1 and TestUser2 as shown in Figure 3.

4. Results and Discussion

The multilevel queue configuration follows the below acronyms for Hadoop YARN scheduler scheduling techniques shown in Table 7.
Table 6. Performance evaluation metrics and parameters

| Parameter Name | Performance Measure | LATE Scheduler Measure / Contra |
|----------------|---------------------|---------------------------------|
| 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 | Expected to be Low to achieve higher job locality |
| 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 | Expected to be Low to achieve lower network usage |
| Job Locality | Expected to be higher to achieve the fundamental principle of data locality | High priority jobs are to be given priority based on the needs so that jobs with higher priority can be run immediately. |
| 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 | Expected to be Minimum |
| 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 | Expected to be Minimum |
| Total Job Time | Total Job execution time is calculated. The value is extracted from the Job Counters by the parameter SLOTS_MISSIS_MAP & SLOTS_MISSIS_REDUCES | Expected to be Minimum |
| Total Job waiting Time | Total Job waiting time is calculated. The value is extracted from the Job Counters by the parameter FALLOW_SLOTS_MISSIS_MAP & FALLOW_SLOTS_MISSIS_REDUCES | Expected to be Minimum |

Figure 1. Allocation rule for multilevel queue.

Figure 2. Capacity scheduler multilevel queue setup.

Table 7. List of scheduling techniques used for performance comparisons

| Used Name in this Work | Scheduler Technique | Scheduling Policies | Number of Concurrent Users | Number of Concurrent Jobs per User |
|------------------------|---------------------|---------------------|---------------------------|----------------------------------|
| CS1U1J | Capacity Scheduler | First In First Out | 1 | 1 |
| CS2U2J | Capacity Scheduler | First In First Out | 2 | 2 |
| CS3U3J | Capacity Scheduler | First In First Out | 3 | 3 |
| FS1U1J - FIFO | Fair Scheduler | First In First Out | 1 | 1 |
| FS2U2J - FIFO | Fair Scheduler | First In First Out | 2 | 2 |
| FS3U3J - FIFO | Fair Scheduler | First In First Out | 3 | 3 |
| FS1U1J - FAIR | Fair Scheduler | Fair | 1 | 1 |
| FS2U2J - FAIR | Fair Scheduler | Fair | 2 | 2 |
| FS3U3J - FAIR | Fair Scheduler | Fair | 3 | 3 |
Firstly, this work demonstrates the results of configuration with CS - 1U - 1J, FS - 1U - 1J – FIFO and FS - 1U - 1J – FAIR where the performance evaluation parameters such as Local Storage cost, HDFS Storage cost, Map Job Waiting Time, Reduce Job Waiting Time, Total Job Time and Total Job waiting Time are being compared as depicted in Table 8.

Hence this clearly demonstrates with a proper configuration of the queue, the waiting times for the jobs are nearly zero. The results are also demonstrated visually depicted in Figure 4.

Next, this work demonstrates the results of configuration with CS - 2U - 2J, FS - 2U - 2J – FIFO and FS - 2U - 2J – FAIR where the performance evaluation parameters such as Local Storage cost, HDFS Storage cost, Map Job Waiting Time, Reduce Job Waiting Time, Total Job Time and Total Job waiting Time are compared for User - 1 as shown in Table 9 and User – 2 in Table 10.

The results clearly demonstrate the reduction of the local and HDFS storage cost for the configuration, where multiple users submit multiple jobs with weighted queue configurations. The waiting time remains zero. The results are also demonstrated visually for User - 1 in Figure 5 and User - 2 in Figure 6.

Next, this work demonstrates the results of configuration with CS - 3U - 3J, FS - 3U - 3J – FIFO and FS - 3U - 3J – FAIR where the performance evaluation parameters such as Local Storage cost, HDFS Storage cost, Map Job Waiting Time, Reduce Job Waiting Time, Total Job Time and Total Job waiting Time are compared for User - 1 shown in Table 11, User - 2 shown in Table 12 and User – 3 shown in Table 13:

The results clearly demonstrate the reduction of the local for the configuration, where multiple users submit multiple jobs with weighted queue configurations. The waiting time increases significantly for Fair Scheduler with FIFO configuration and remains zero for other configurations. The results are also demonstrated visually for User - 1 shown in Figure 7, User - 2 in Figure 8 and User – 3 in Figure 9.

**Table 8. Performance comparison for configuration set – 1**

| Experiment Name | Local 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) |
|-----------------|------------------------|------------------------|-----------------------------|-------------------------------|----------------------|-----------------------------|
| CS1U1J          | 8494                   | 208                    | 0                           | 0                             | 1011018              | 0                           |
| FS1U1J - FIFO   | 8493                   | 208                    | 0                           | 0                             | 2194453              | 0                           |
| FS1U1J – FIFO   | 8493                   | 208                    | 0                           | 0                             | 2510011              | 0                           |

**Table 9. Performance comparison for configuration set – 2**

| Experiment Name | Local 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) |
|-----------------|------------------------|------------------------|-----------------------------|-------------------------------|----------------------|-----------------------------|
| CS2U2J          | 8494                   | 208                    | 0                           | 0                             | 1562530              | 0                           |
| FS2U2J – FIFO   | 8493                   | 208                    | 0                           | 0                             | 2217508              | 0                           |
| FS2U2J – FIFO   | 4595                   | 123                    | 0                           | 0                             | 1160195              | 0                           |
Table 10. Performance comparison for configuration set – 2 (User-2)

| Experiment Name | Local 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) |
|------------------|-------------------------|------------------------|---------------------------|-----------------------------|----------------------|-----------------------------|
| CS2U2J           | 4595                    | 123                    | 0                         | 0                           | 556563               | 0                           |
| FS2U2J – FIFO    | 4595                    | 123                    | 0                         | 0                           | 1207336              | 0                           |
| FS2U2J – FIFO    | 8493                    | 208                    | 0                         | 0                           | 1738994              | 0                           |

Table 11. Performance comparison for configuration set – 3 (User-1)

| Experiment Name | Local 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) |
|------------------|-------------------------|------------------------|---------------------------|-----------------------------|----------------------|-----------------------------|
| CS3U3J           | 8494                    | 208                    | 0                         | 0                           | 764193               | 0                           |
| FS3U3J – FIFO    | 8493                    | 208                    | 77878                     | 0                           | 814368               | 77878                       |
| FS3U3J – FIFO    | 8493                    | 208                    | 0                         | 0                           | 2173455              | 0                           |

Table 12. Performance comparison for configuration set – 3 (User-2)

| Experiment Name | Local 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) |
|------------------|-------------------------|------------------------|---------------------------|-----------------------------|----------------------|-----------------------------|
| CS3U3J           | 4595                    | 123                    | 0                         | 0                           | 389306               | 0                           |
| FS3U3J – FIFO    | 4595                    | 123                    | 0                         | 0                           | 370165               | 0                           |
| FS3U3J – FIFO    | 4595                    | 123                    | 0                         | 0                           | 1253263              | 0                           |

Table 13. Performance comparison for configuration set – 3 (User-3)

| Experiment Name | Local 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) |
|------------------|-------------------------|------------------------|---------------------------|-----------------------------|----------------------|-----------------------------|
| CS3U3J           | 4596                    | 124                    | 0                         | 0                           | 635978               | 0                           |
| FS3U3J – FIFO    | 4595                    | 124                    | 0                         | 0                           | 1143084              | 0                           |
| FS3U3J – FIFO    | 4595                    | 124                    | 0                         | 0                           | 1225295              | 0                           |

Figure 4. CS1U1J, FS1U1J-Fair, FS1U1J-FIFO setup job completion time.

Figure 5. CS2U2J, FS2U2J-Fair, FS2U2J-FIFO setup local storage cost, HDFS storage cost and job completion time – User-1.
5. Conclusion and Future Work

The work analyzed the Hadoop and MapReduce framework for better understanding of the scheduling techniques. The outcome also includes the existing techniques with understanding of the significance in time and data complexity paradigm like FIFO scheduler, Fair Scheduler, Capacity Scheduler, Dynamic Priority Based Hybrid Scheduler and LATE Scheduler. The work proposes a novel multi-level queue allocation technique with custom policies.

Significant achievement also includes performance evaluation metrics for comparative understanding of the proposed and existing technique. The final outcome of the work demonstrates the significance of queue scheduling with allocation policies for the optimal placement of jobs in queues.

The future scope of this work clearly lay down the base for deriving novel techniques for optimal placement of jobs based on priority and to also consider the need to reduce energy.

6. References

1. Peisker A, Dalai S. Data analytics for rural development. Indian Journal of Science and Technology. 2015 Feb; 8(S4):50–60.
2. Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop distributed file system. Sunnyvale, California USA, IEEE; 2010. p.1–10.
3. Yuan D, Yang Y, Liu X, Chen J. A data placement strategy in scientific cloud workflows. Future Generation Computer Systems. 2010; 26(8):1200–14.
4. Morton K, Balazinska M, Grossman D. Paratimer: A progress indicator for MapReduce DAGs. In Proceedings of the International Conference on Management of Data, USA; 2010. p.507–18.
5. Wei L. Efficient processing of k nearest neighbor joins using MapReduce. Proceedings of the VLDB Endowment. 2012; 5(10):1016–27.
6. Dean J, Ghemawat PS. Mapreduce: Simplified data processing on large clusters. OSDI, Proceedings of 6th Symposium on Operating System Design and Implementation, ACM Press, USA. 2004; 6:1–10.
7. Hadoop job scheduling with the fair and capacity schedulers, Hadoop Summit[Internet]. [cited 2009 Jun 10]. Available from: https://people.csail.mit.edu/matei/talks/2009/hadoop_summit_fair_scheduler.pdf.
8. Gowthami M, Kilal JB, Jayashree R. Scheduling job queue on hadoop using hybrid Hadoop fair sojourn protocol. Indian Journal of Science and Technology. 2016 Apr; 9(16):1–4.
9. Pastorelli M, Barbuzzi A, Carra D, Amico MD, Michiardi P. Practical size-based scheduling for MapReduce workloads. CoRR; 2013. p.1–12.
10. Sreedhar C, Kasiviswanath N, Reddy PC. A survey on big data management and job scheduling. International Journal of Computer Applications. 2015; 130(13):41–9.
11. Chen J, Wang D, Zhao W. A task scheduling algorithm for Hadoop platform. Journal of Computers. 2013; 8(4):929–36.

12. Lang W, Jignesh M, Patel P. Energy management for MapReduce clusters. Proceedings of the VLDB Endowment. 2010; 3(1–2):129–39.

13. Nguyen P, Simon T, Halem M, Chapman D, Le PQ. A hybrid scheduling algorithm for data intensive workloads in a MapReduce environment. Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing, Washington, DC, USA; 2012. p. 161–7.

14. Zaharia M, Konwinski A, Joseph AD, Katz R, Stoica P. Improving MapReduce performance in heterogeneous environments. OSDI 8th USENIX Symposium on Operating Systems Design and Implementation; 2008. p. 29–42.