ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

# A Job Utility and Size-based Scheduler for Meeting the Client's Job Requirements in Hadoop

Aditi Jain[1*], Sanjay Jain[2], DA Mehta[1]

Shri Govindram Seksaria Institute of Technology and Science, Indore, Madhya Pradesh, India[1]

Shri Vaishnav Polytechnic College, Indore, Madhya Pradesh, India[2]

E-mail: aditijain0817@gmail.com

**Abstract:** Hadoop, an implementation of MapReduce paradigm, is an open-source powerful parallel processing framework for handling big data on distributed commodity hardware clusters such as Clouds. Proper scheduling of jobs on such a distributed cluster is an important factor in determining the clusters' performance. Proper scheduling of jobs in Hadoop cluster requires usage of efficient algorithms that should focus on meeting job requirements like job deadline, job priority etc. provided by clients and also, the improvement in the average job response time in the cluster. Client's requirement on job completion is an important way to measure the service quality which the client obtains from the cloud. Utility of a job denotes the quality of service requirements between client and service provider. Existing job schedulers in Hadoop (viz., FIFO, Fair Scheduler, Capacity Scheduler) usually ignore job's requirements (like job deadline, job priority etc.) specified by clients. There is a need of a scheduler that schedules jobs efficiently considering the clients' job requirements. The problem addressed in this work is of scheduling jobs taking into account the job requirements specified by client. In order to satisfy the client-specified job requirements, the scheduling algorithm calculates the utility value of each job using the job requirements specified by clients and the estimated job size. The results show an increase in the percentage by which jobs in the proposed scheduler are meeting client's job requirements when compared to the default scheduler in Hadoop.

**Keywords:** Energy efficient algorithm; Manets; Total transmission energy; Maximum number of hops; Network lifetime

## I. INTRODUCTION

To overcome the problem of Big Data storage and processing, Apache developed Hadoop, an open source framework which breaks down big data problems into smaller ones so that analysis could be done quickly and cost effectively. Hadoop parallelizes data processing across several computing nodes to speed computations and hide latency. Hadoop will remain to be one of the widely used solutions to process large data. Therefore, the performance of Hadoop cluster is an important topic of research. Let us quickly understand how hadoop works through an analogy. Historically, ox was used to carry the load. Then, when load increased, we did not consider to grow the ox large, but instead used several ox put together to pull the heavy load. This same idea is applied while analysing big data. When this concept is applied to computing world, it is termed as distributed computing. Client specifications on completion time of jobs like deadline are a significant way to calculate the utility which the clients obtain from cloud. A utility value of a job denotes the quality of service requirements between client and service provider. One of the several ways by which performance of Hadoop can be improved is through proper scheduling of jobs. Also, job scheduling which is based on job size helps in improving the average job response time. Therefore, job scheduler should consider the utility value of job to meet client specifications and also, the job size to reduce the average job response time. In computer science, one of the most studied problems is scheduling. Abstractly, the scheduling problem is defined as follows: given the sets of jobs and resources, where jobs require subset of the resources to progress, how to allocate resources to the jobs

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

optimally for optimizing a metric while jobs get completed. System's performance differs based on the choice made on the scheduling policy. Various performance metrics like the average completion time, the response time, the fairness or the mean waiting time, influence the scheduling policy greatly. A policy that is based on size of a job is called a size-based scheduling policy. The First- In-First-Out (FIFO) scheduler schedules jobs based on their arrival time, while the Shortest Remaining Processing Time (SRPT) policy schedules the job with the smallest remaining processing time first. Among the scheduler families, the best performance is provided by the scheduling policy which is pre-emptive and based on size. This work focuses on scheduling policy for Data-Intensive Scalable Computing (DISC) systems like Hadoop. It is fascinating to study the influence of the scheduling policies on the performances of the parallel and distributed systems as these systems are widely adopted and growing.

## II. LITERATURE REVIEW

This section briefly describes the relevant research work done by different researchers on scheduling algorithm for improving the scheduling policy of Hadoop system. It also enlists the existing approaches for finding the estimated job size and their advantages and drawbacks [1-6].

### 9.1 Scheduling Policy for DISC Systems

Many research works have been followed by the rising of DISC systems such as Hadoop focusing on scheduling in these systems. Chang et al. [5] put forward a scheduler which is based on the analysis by Schulz et al. [6] which shows that the problem to reduce the total completion time of job is NPhard. Chang et al. [5] creates a 2-approximation algorithm to convert the problem in a multi-processor system. When the job sizes are known a-priori, the presented scheduler performs well in terms of job completion times. The paper shows that the scheduler based on estimated job sizes can be implemented in a real-world system. A scheduler which is based on job size should be designed to deal with estimation error to get better performance [5].

### 9.2 Estimating Job Size in MapReduce

In paper HFSP: Bringing Size-Based Scheduling to Hadoop [7], it is shown that in practical scenarios the scheduling based on the job size can work well and paper proposes a way to achieve fairness and optimal response times of system which is based on size based scheduling with aging. HFSP predicts job size by doing online estimation during job execution. "PSBS: Practical Size-Based Scheduling" [8], is enhancement of work in paper [7], deals with error in job size estimations. With this background, it can be noted that scheduling protocols based on size are well known to have a desirable property of achieving optimal mean response time as these policies focus on jobs which are nearest to completion.

### 9.3 Summary

A significant fact inferred from the previous works is the need for short response time of jobs. Size based schedulers can provide better job response time to improve performance in hadoop system. While scheduling with reliability is a well-studied problem [2], the ideas are not particularly designed for systems like Hadoop. To optimize the completion time in the computing clusters it is unclear on how the proposals can be used due to scheduling complexities. The online estimation of jobs runtime using machine learning technique like linear regression method [3,4] is also a work done in recent years. However, more research work on job scheduling in Hadoop is needed which considers the scheduling of the jobs based on the job requirement specified by client. Therefore, in the proposed solution, scheduling of job is done on the basis of job utility value which considers the job requirement specified by client and also, the job size to get improved average job response time.

## III. PROBLEM STATEMENT

To design a scheduling policy based on job utility and job size for Map Reduce jobs in Hadoop environment that focuses on meeting the client's job requirements and also, improving the average job response time so that service quality, average job response time and overall performance of Hadoop cluster can be improved.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

## IV. PROPOSED APPROACH

The proposed approach provides a solution for job scheduling by considering the clients job specification and job size. The calculation of the job utility based on job deadline and priority will help to determine which job has higher value of utility. Jobs which have higher utility will get scheduled first. Jobs with same utility value will be scheduled on the basis of job size. Those jobs are chosen for scheduling on nodes whose job size is smaller when compared to other jobs with same utility value, so that this scheduling reduces the average job response time.

### 3.1 Job Utility and Size Calculation

In the proposed solution, the client specifies the job priority and deadline value (to compute the utility value of job) for each submitted job. For a client, job priority is used to pick a job from multiple jobs submitted by this client, job with higher priority is picked first. Runtime distribution estimator captures the job workload and runtime distribution of resources (also known as containers). The runtime distribution information of containers is reported using the actual container runtime. For satisfying the client's job requirements, utility of each job is calculated. Job size is calculated using the runtime of containers and number of job's tasks. Then, using this calculated job size and utility value of job, the algorithm gives preference to that job whose utility value is more. When utility is same, job with shorter job size is given preference so that the average response time of system gets improved.

## V. ORGANIZATION OF THE PAPER

This section provides a brief summary about the subsequent sections. Section 5 consists of analysis of the proposed solution. Section 6 describes the proposed scheduler architecture. Later section describes the experimentation performed and results discussion. Then, the next section describes literature review and summarizes the various solution approaches for the specified problem and work done by different researchers in the field of job scheduling in hadoop. It also briefly describes the problems in existing approaches proposed by different researchers. Last section of the paper concludes with the achievements and the future work is described.

## VI. ANALYSIS OF PROPOSED SOLUTION

For improving performance of Hadoop system and meeting the client job requirements, an efficient job utility and size-based scheduling policy is proposed. Utility of job is calculated using the parameters specified by the client through an interface. Job whose utility value is more is chosen for scheduling before other jobs. When utility value is same, then job whose size is small is chosen for scheduling so that average job response time is improved. Job size information is required for scheduling using a size-based scheduler, but such information is not present in Hadoop. Therefore, scheduler has to estimate the size of job.

The analysis of proposed solution is as follows:

- The system architecture of scheduler is proposed to include a module to estimate job utility and job size.
- The proposed algorithm calculates the workload which is approximated according to the information of resources (i.e., containers mean runtime distribution) and in order to satisfy the client's job requirements, scheduler uses job utility and job size to perform resource allocation.

## VII. PROPOSED SCHEDULER ARCHITECTURE

This section describes the architecture of the proposed job utility and job size based scheduling algorithm for Hadoop framework. Three main components of scheduler are: (i) a configuration interface for user jobs, (ii) a mean estimator unit for distribution of jobs (MED) and (iii) a containers assigning unit (CAU). Job requirements by clients are submitted using a configuration interface. The CAU is called whenever container gets free. The total estimated workload of each job is provided by MED units to CAU, so that assignment of the containers to job can be done. The

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

CAU repeats this in a cycle as task finish. The scheduler architecture is illustrated in Figure 1 and detailed explanation is provided in the following paragraphs.

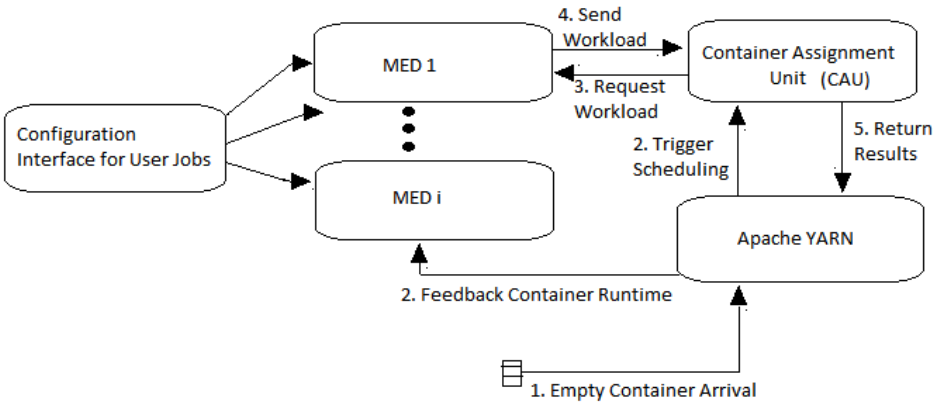


**Figure 1: Proposed scheduler architecture.**

### A. Configuration Interface for User Jobs

The interface is used to get the requirements of job specified by the client like time budget T, priority value P and sensitivity S using an XML file. This XML file is submitted along with the job submission to Hadoop cluster. These parameters are used to calculate the utility of job using a utility (linear) class which gives a value of max(S(T - Ec)+P, 0), where Ec is estimated completion time obtained using job size calculation.

### B. Mean Estimator of Distribution (MED)

The MED unit estimates the workload of jobs to produce a distribution $\eta_i$. The mean time estimator class is used for estimating distribution of workload. This class gives a mean distribution equal to the multiple of the number of tasks pending and the mean of container runtime.

### C. Containers Assigning Unit (CAU)

The CAU decides the assignment of containers to jobs by applying the job size calculation algorithm to obtain the job schedule. As soon as a container gets free in the system, for every job CAU gets the estimate of workloads,i, from the Distribution Estimation unit. Then, the job size calculation algorithm (Algorithm 1) is used to approximate the job runtime.

Algorithm 1 Algorithm for Job Size Calculation

1: Initialize J = 1, 2, ... j as set of user jobs and $H_t = 0$, $\forall t \in T$

2: Assign $Gf = \min_{i;y}(Ui(y))$

3: while!(J is empty) do (apply bisection method)

4: Assign $Gi = \max_{i;y}(Ui(y)\ \forall\ i\ 2 \in J)$.

5: while Gi - Gf ¿ Δ do

6: Select $g = 1/2\ (G_i + G_f)$

7: if $\sum_{i \in J_z} \eta_i + H_{U_z^{-1}(g)}\ \mathrm{CU}_z^{-1}(g)$

$\forall_z \in J$ then

8: Assign $G_f = g$

9: else

10: Assign $G_i = g$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

11: end if
12: end while
13: Assign Ti = $Ui^{1}$(Gf ) to be estimated size
14: Remove i from J, set of jobs
15: end while

Algorithm 2 Queue Occupation Algorithm
1: Input: Target completion time T1, ...TN
2: Input: Estimated workload 1, ...N
3: Input: Average container runtime R1; :::RN
4: Initialize the queues $O_q = 0$, $\forall q \in 1, ...Q$
5: for each $j \in J$, J is set of jobs do
6: Set q = 1
7: while $\eta_j \neq 0$ do
8: $Assign\ A = \min\left(\left\lfloor \frac{T_i - O_q}{R_j} \right\rfloor, \eta_j\right)$ to queue q
9: Update $\eta_j = \eta_j - A$ and $O_q = O_q + A$
10: Find another free queue, compute q++
11: end while
12: end for

## VIII. EXPERIMENTATION AND RESULTS

The performance of the scheduler, proposed in this paper, is assessed by creating an in-house cluster of Hadoop nodes. This section presents the evaluation results starting with the description on the environment used in the experimentation and then, the results are discussed.

### 7.1 Experimental Setup

To evaluate the implementation of the scheduler, a multimode Hadoop cluster is configured. The cluster consists of three virtual machines, created using VMWare Workstation 9. One VM was created as the master node (named hmaster) which ran services namely Name Node, Resource Manager and Secondary Name Node. The remaining two VMs (named hslave1, hslave2) were slave nodes which ran Node Manager and Data Node. Mater node was assigned 3 cores and configured with a RAM of 3 GB; other two VMs were assigned 2 cores and a RAM of 2 GB each. All VMs had a hard disk with capacity 100 GB and CPU 3.10 GHz. All the nodes had Linux Ubuntu (14.04) and JAVA 1.8.2. For noting the runtime of 7 different jobs, we have used a single-node cluster with 5 cores, CPU 3.10 GHz, RAM of 5 GB and har disk capacity of 100 GB on Ubuntu Linux (16.04). Table 1 describes the main properties and parameters of Hadoop used for experiments and all other parameters were used with default values. The modified version of Hadoop 2.6.5 is used for this evaluation.

| Parameter and Properties | Value |
|---|---|
| Replication | 2 |
| Block Size (HDFS) | 128 MB |
| Heartbeat interval | 2 seconds |
| $HADOOP | /usr/local/hadoop/ |
| yarn.resource manager. scheduler.class | org.apache.hadoop.yarn. server.resourcemanager. scheduler.Scheduler |
| mapreduce.framework.name | yarn |

**Table 1: Hadoop parameter and properties setting made in experimentation.**

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

### 7.2 Test Workload

The benchmark job - WordCount, MapReduce job provided in Hadoop, are run on datasets of size 108 MB, 218 MB, 534 MB, 1.2 GB, 2.38 GB, 5 GB and 10 GB. These jobs are considered to be of different type as the data sizes the job will process are different. Firstly, all 7 jobs individually are run on a single-node cluster for 10 times and the total runtime of individual job are noted which is shown in Table 2 (this is the benchmarked runtime of jobs). Then, these 7 jobs are used to create a set of 100 jobs which are run on the multi-node cluster with different submission times in different scenarios, each scenario was tested for 20 times. The jobs priority P is given from 1 to 5 randomly. Time critical jobs are 25% of total jobs, time sensitive jobs are 55% and 20% jobs are time in-sensitive. The experiments are performed for the scenario where the budget of jobs (given in time) is equal to 2 times the benchmarked run-times of wordcount jobs.

| Job | Input Size | Map Tasks | Runtime |
|---|---|---|---|
| Job1 | 108 MB | 1 | 32 sec |
| Job2 | 218 MB | 2 | 76 sec |
| Job3 | 534 MB | 5 | 150 sec |
| Job4 | 1.2 GB | 11 | 254 sec |
| Job5 | 2.38 GB | 24 | 640 sec |
| Job6 | 5 GB | 57 | 1100 sec |
| Job7 | 10 GB | 112 | 1840 sec |

**Table 2: Overview of benchmark jobs.**

### 7.3 Benchmark Results

Comparison experiments among the proposed scheduler and Capacity scheduler are designed in this section, such as the comparison on completion time and response time. To assess the performance, the runtime data for word count benchmark jobs is acquired. Benchmark jobs are run for different scenarios to evaluate the performance. The following scenarios are considered: (i) All jobs are submitted with different intervals like 5, 10, 30, 60 seconds, (ii) New job is submitted after completion of previous job, (iii) All jobs are submitted in the descending order of data size to process, (iv) All jobs are submitted in the ascending order of data size to process, (v) All jobs are submitted randomly with random interval.

## IX. RESULTS AND DISCUSSION

There is improvement in overall average response time of jobs by a percent of 8.41 when compared the proposed scheduler to Capacity scheduler. Two test scenarios *viz.* Test case (a) and Test case (b) are important for discussion. In Test case (a), where jobs are submitted in the descending order of the data size to be processed, we observe that Capacity Scheduler schedules jobs according to the jobs time of arrival where job which arrived first gets executed first as there is no consideration of jobs utility value. While, in the proposed scheduler, scheduler chooses that job whose utility is more. In Test case (b), jobs are submitted in random order with random interval. At a particular instance, we observe that Capacity Scheduler schedules the job in following order: on queue of datanode-1 - Job3, Job5, Job2, Job6 and on queue of datanode-2 - Job4, Job1, and Job7. While using the proposed scheduler, we observe the following schedule of the jobs: on queue of datanode-1 - Job3, Job5, Job2, Job7 and on queue of datanode-2 - Job4, Job1, Job6. In this test scenario, the utility value of Job6 is more than the utility value of Job7. As Job7 arrives before Job6, in Capacity Scheduler, Job7 gets scheduled before Job6 as no utility value of job is considered. While using the proposed scheduler, Job6 is prioritized over Job7 as the former job has higher utility value than the latter job. Also, in this test case, we observe an overall decrease in response time of jobs in the proposed scheduler by a 13% when compared to Capacity Scheduler. There are cases where we see degradation in runtime value of the jobs running under the proposed approach. This might be due to the reason that the VM sometimes stuck in computation and hangs for few seconds as running on limited resources. More acknowledgeable results can be seen when we run our proposed scheduler on a real hadoop cluster with physical machines.

## X. CONCLUSION AND FUTURE WORK

This section concludes the work done followed by the limitations and suggestions for the future work.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

### 10.1Achievements

In the recent years, the unparalleled proliferation in technologies and services of data centers opens new options for data intensive applications in the Cloud. MapReduce frameworks such as Hadoop have been introduced as services. MapReduce provides an effective way for smaller business to take advantages of this powerful programming paradigm in the Cloud. Hadoop performance is an important factor. Improper job scheduling can cause inefficiency in Hadoop performance. Thus, scheduling in MapReduce is one of the most interesting topics for both industries and academic institutes. This project work addresses the problem by scheduling the jobs in Hadoop. We found that current design of Hadoop scheduling and resource management is not optimized for cluster computing applications since many prominent features of the applications from the view point of clients like deadline, priority are not considered. Here, the new proposed scheduler considers the clients job requirements for scheduling. In this work, a job utility-based scheduler is developed for Hadoop MapReduce to meet client's job requirements and to further improve the average job response time. Finally, the effectiveness of the proposed algorithm is evaluated by considering metrics such as execution time, response time. Then, a comparison is done between the original Capacity scheduler and the proposed job utility-based scheduler. Experiments show that the proposed scheduler improves the response time of Hadoop jobs while considering the job requirement specified by clients.

The main contributions of this project work are:

- Design of a utility based and size based job scheduling algorithm for meeting client's job requirements and also for efficient job scheduling in big data processing environment like Hadoop.
- Reduction in response time of hadoop jobs.

The result of experiments show that the proposed algorithm meets the job requirements by 14.36% more and reduces the response time of jobs by an average value of 8.41%, when compared to an existing scheduler viz., Capacity Scheduler, which does not at all consider the requirement of the jobs, like deadline, priority and job size while scheduling jobs to meet clients requirements and to achieve better average job response time.

### 10.2Limitations and Future Directions

Experimentation is performed on an in-house small cluster consisting of virtual machine nodes. It would be interesting to evaluate the proposed scheduler on a cluster of real distributed machines. Further, the effect of network parameters can be huge. This can be seen as a future enhancement. Another work can be to apply machine learning techniques to estimate the job size. While multiple scheduling policies and resource management schemes are designed for tackling different performance issues in Hadoop, we assume that this work is a significant effort in this direction. We hope that the designs proposed here can offer a useful reference point for improving efficiency of cluster computing platforms by exploiting features of cluster computing applications.

## XI. REFERENCES

1. A AuYoung, L Grit, et al. Service contracts and aggregate utility functions. IEEE International Symposium on High Performance Distributed Computing. 2006.
2. A Christian L Roel et al. Robust optimization for resource-constrained project scheduling with uncertain activity durations. Flexible Services and Manufacturing Journal 2013; 25: 175-205.
3. K Mukhtaj, J Yong, et al. Hadoop performance modelling for job estimation and resource provisioning. IEEE Transactions on Parallel and Distributed Systems 2015; 27: 441-454.
4. V Abhishek, C Ludmila, et al. Resource provisioning framework for MapReduce jobs with performance goals. International Conference on Distributed Systems Platforms and Open Distributed Processing 2011: 165-186.
5. C Hyunseok, K Murali et al. Scheduling in MapReduce-like systems for fast completion time. IEEE Infocom 2011.
6. SS Andreas, Scheduling to minimize total weighted completion time: Performance guarantees of lp-based heuristics and lower bounds. Integer Programming and Combinatorial Optimization 1996: 301-315.
7. P Mario, C Damiano, et al. HFSP: size-based scheduling for Hadoop, IEEE Transactions on Cloud Computing 2013; 5: 43-56.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

**Vol. 6, Issue 11, November 2017**

8. DA Matteo, C Damiano, et al. PSBS: Practical size-based scheduling. IEEE Transactions on Computers 2015; 65: 2199-2212.