International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 9, Issue 02, February-2024, http://ijmec.com/, ISSN: 2456-4265

JOB SCHEDULING ALGORITHMS FOR EFFICIENT TASK EXECUTION

[1]Gadala Suhasini
Assistant Professor,
IT DeptMJCET, Hyderabad,
Telangana, India
Suhasini.gadala@gmail.com

[2]Dr.A.S.Gousia Banu,
Assistant Professor
IT Department, MJCET,
Hyderabad, Telangana India
gousia.banu@mjcollege.ac.in
gbanuzia@gmail.com

[3]Samatha Konda
Assistant Professor
IT Department, MJCET,
Hyderabad, Telangana India
samatha.phd22@gmail.com

Abstract-Map Reduce is a programming approach that allows enormous scalability on hundreds or thousands of computers in a cluster like Hadoop. In fact, the word Map Reduce refers to two different operations performed by Hadoop algorithms. The first one is the map task, which receives a collection of information and transforms it into another data set that splits each item into two keys. We need to improvise the algorithms for optimum use of resources with dead line by setting the capacity and priority of queues.

Key words : Map Reduce,Fair Scheduler,Capacity Scheduler

I. INTRODUCTION

Map Reduce is utilized primarily for analyzing large volumes of data saved in the Hadoop cluster concurrently. This is a Google-designed hypothesis for parallelism, allocation of information, and defect tolerance. Data analysis Map Reduced in the form of two keys. The mapping component between two related information items is two keys. For the analysis of huge data sets, MR is shown. The programmers may build Map Reduce programs that can be adapted to specific business situations. The MR workflow traverses several stages and the final output is saved with replicas in HDFS. The Job Tracker serves an important part in scheduled work and monitors the whole map and reduces tasks. A Definition of issue Can any job translating into MapReduce J & needing to analyze length N information during a period D, be executed on a MapReduce clusters with map task spaces for M node functions, reduce task spaces, and potentially perform k tasks only at time [1].

Table 1.1 Effect of data size on Turnaround Time (sec) of grep job for FIFO, Fair, and Capacity schedule.

Data Size	FIFO Scheduler	Fair Scheduler	Capacity Scheduler
818.5 MB	69	56	85
1.2 GB	73	68	92
2.4 GB	159	122	143

DOI: https://doi-ds.org/doilink/02.2024-13498221

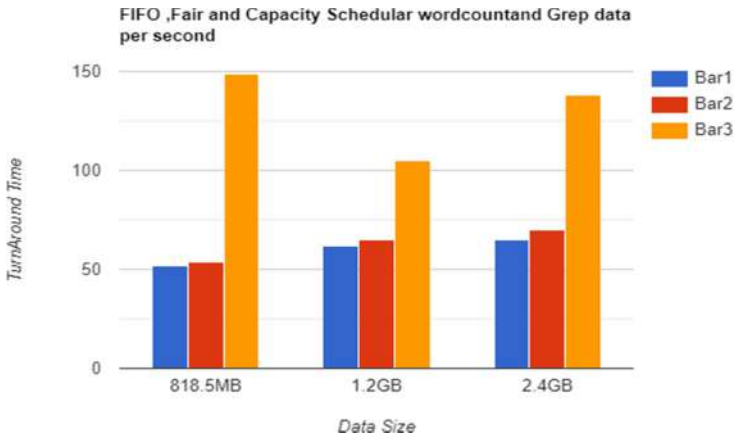


Figure 1.1 Effect of data size on grep operationturn- around time (sec).

To assess how quickly a task can be finished, the number of data objects supplied to the three schedules is adjusted. An intelligent technique to determine system performance is to look at the latency[2]. The faster turnaround time provided by the other two schedulers pales in comparison to what Fair Scheduler can do to assist with task reaction times. The most useful aspect of this scheduling technique is that it cuts system costs. On average, when there is a large amount of information, such as

2.4 GB, the Fair and Capacity methodologies perform fewer calculations during task processing when comparing to the time that Turnaround takes to finish in the above picture illustrated in Figure 1.1 and comparisons specified in Table 1.1.

II. CUMMULATIVE ANALYSIS BASED ON INFORMATION PROCESSED.

By changing the data size used by the three schedulers, you may compare data processed per second. In the illustration below, the Fair and Capacity schedulers handle more bytes as opposed to the FIFO scheduler for input data volumes of 1.2 GB and 2.4 GB. By using the Capacity and Fair schedulers, you may enhance the overall system usage, which impacts data analysis. Capacity scheduling gives capacity assurances to queues, which is to say the total amount of Map - Reduce slots that the queue is allowed to use. When there are unused capacity, they can be put to use by other queues. Consequently, if the cluster usage rises, the runtime of a job will reduce, as will the processing capability. Jobs in the pools are submitted via the Fair scheduler. You may think of pools as lines, too. A limited amount of Map/Reduce slots is assigned to each pool. The pool should assign at least the minimal number of slots when there are pending jobs. The pool itself does not have any positions, howeverif the pool has had no jobs, the slots could be used by another pools[3]. Using these empty Map - Reduce slots can boost the overall data processing capability, as well as reducing execution time.

Table 1.2 : Data handled per second for FIFO,Fair, and Capacity schedulers.

DOI: https://doi-ds.org/doilink/02.2024-13498221

Data Size	FIFO Scheduler	Fair Scheduler	Capacity Scheduler
818.5 MB	2.57	2.52	2.59
1.2 GB	2.48	2.55	2.52
2.4 GB	2.48	2.5	2.49

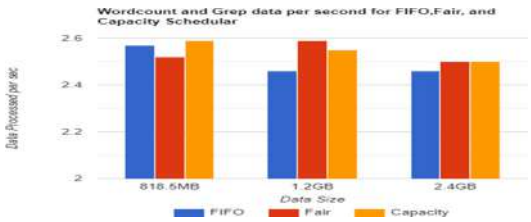
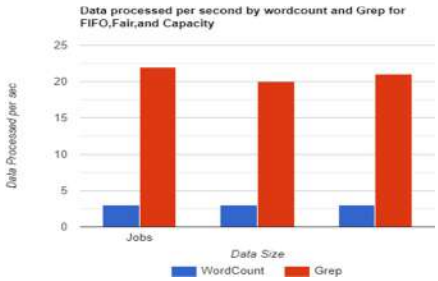


Figure 1.2 : Comparative of FIFO, Fair,and Capacity schedulers' data handled per second.

The amount of data handled per second varies by application. Word limit analyzes less information per second than Grep. Figure shows that Grep processes 8 petabytes of data per second than Wordcount when both processes 2.4 GB of data on FIFO, Fair, and Capacity schedulers. This illustrates that schedulers' effectiveness is job-dependent[4]. Wordcount is a CPU heavy operation, thus it takes longer to process, while Grep requires less time, therefore it boosts data analysis per second illustrated in Figure 1.2 and comparisions observed in Table 1.2.

	FIFO Scheduler	Fair Scheduler	Capacity Scheduler
Word count	2.48	2.46	2.48
Grep	21.96	20.51	20.7

Table 1.3 : FIFO, Fair, and Capacity schedulersdata processed per second by Wordcount and Grep.



DOI: https://doi-ds.org/doilink/02.2024-13498221

Figure 1.3 : FIFO, Fair, and Capacity schedulers'data processing per second for 2.4 GB of data.

	Cap-FIFO	Fair-FIFO	Fair-Fair	Fair-DRF
25th percentile	50	52	52	52
Median	63	70	70	70
75th percentile	140	135	139	140

Table 1.4: Average total delays of four SPCs inone queue.

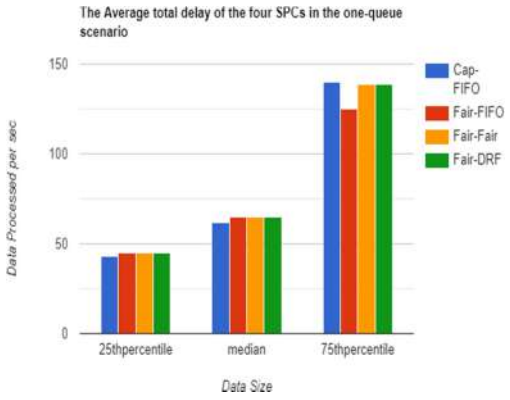


Figure 1.4: The four SPCs' overall averagelatencies in one queue.

On the basis of the information revealed here, we can conclude the following in Figure 1.4 and Table 1.4 The best SPC for the one-queue situation is Fair-DRF due to the good application performance metrics in terms of completion percentage, turnaround time, and streaming system performance[5]. We can only really make a recommendation if we only look at the amount of resources used, and hence Cap-FIFO should be used if we care about resource utilisation efficiency and Average comparisons are specified Table 1.5 and Figure 1.5.

Table 1.5: Average system burden while usingfour SPCs in one queue.

	Average System Load
Cap-FIFO	1105
Fair-FIFO	1110
Fair-Fair	1090
Fair-DRF	1075

DOI: https://doi-ds.org/doilink/02.2024-13498221

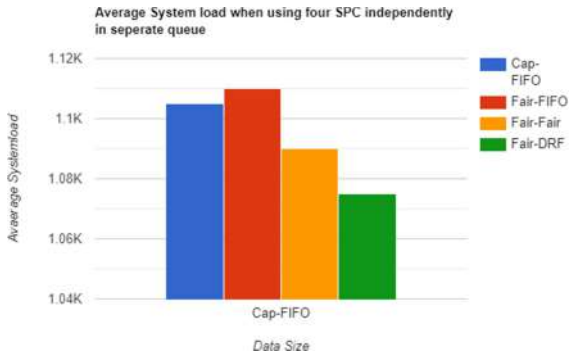


Figure 1.5: Assuming one queue, the average system load when using four SPCs.

The above graphic displays the system load of the four SPCs, as shown in the load average chart. The group, on average, created 1112 containers when running the test workload that involved Cap-FIFO. Because Cap-FIFO contains more containers than the other two SPCs, its average value is lower illustrated in Figure 1.6 and Table 1.6.

Table 1.6: The four SPCs' average work completion rates and turnaround times in the one-queue scenario.

	Average Workload completion time	**Average workload turnaround time**
Cap-FIFO	2963.85	0.9681
Fair-FIFO	2938.47	0.9681
Fair-Fair	2935.68	0.966
Fair-DRF	2940.24	0.9596

DOI: https://doi-ds.org/doilink/02.2024-13498221

Table 1.7: Analysis of FIFO and CapacityScheduler execution times for various files.

	FIFO	Capacity
100	1	0.7
200	1.5	1
300	2.7	2.4
400	3.4	3
500	4.5	3.7
600	5	4.7

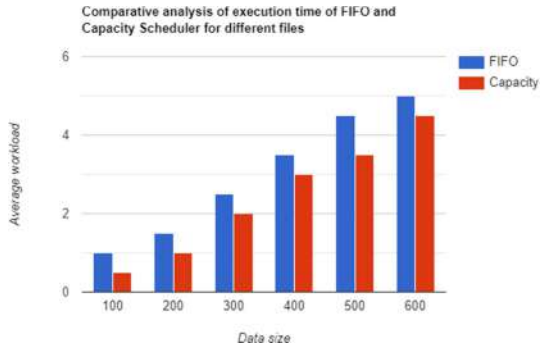


Figure 1.6: The four SPCs' average work completion rates and turnaround times in the one-queue scenario.

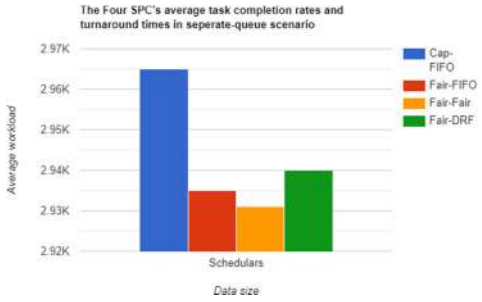


Figure 1.7 :Comparitive analysis of execution timeof FIFO and Capacity Scheduler for different files.

Models Estimate Expiration We are developing an early estimate model depend on a number of assumptions The network comprises of heterogeneous base station such that the expense sequencing of every map is equivalent or reduces the server; The main allocation of the data input is consistent such that every reduction nodes is processed with the same quantity of reduction information; Reduce tasks after completion of all map tasks; The input data already are in HDFS. In order to obtain the terms for the least amount of cartographic jobs and to decrease tasks, we modify the approach utilized for Equal Load Partitioning [7]. To assess the work length, we evaluate completing the

DOI: https://doi-ds.org/doilink/02.2024-13498221

map, reducing runtime and transfer of data throughout copy reduction. Hadoop provides pluggable scheduling algorithms and uses minimal task planning criterion to build the restricted scheduling. It is created utilizing Hadoop version 1.1.2 raw data as a contrib component. To utilize the scheduler, the Hadoop config file has to be updated. We also have an internet application that enables the clients to define the time limit for a particular task illustrated in Figure 1.7 and Table 1.7.

Works The Job Submitter job posting procedure is as follows: The work is submitted by the client. (Schedule 1) The planner calculates the minimal map size and reduces the spaces needed to complete a task(step 2). If the makespan problem exists, the client is informed that a different date number is entered. When approved, work is planned[6].

Constraint Scheduler's Model objectives were: (1) to provide clients instant response as to whether or not the task can be performed inside the specified period and carry out this if the deadlines is fulfilled. Alternatively, clients have the opportunity to re-send amended deadlines. 2) Optimize the amount of jobs in the clusters and meet the time requirements of all tasks.

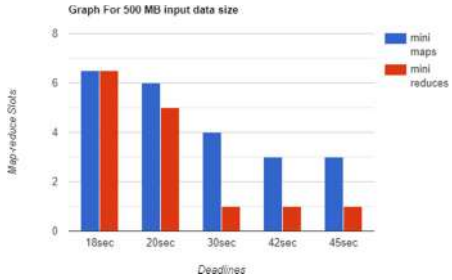


Figure 1.8: Execution of task with deadline using Map-Reduce data of 500MB.

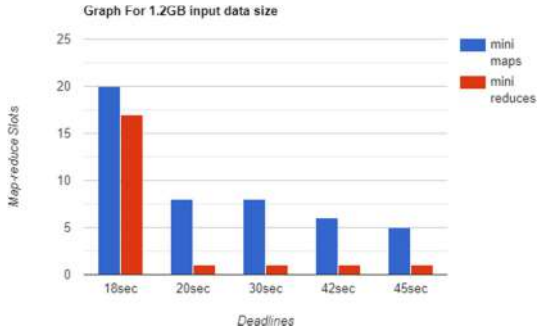


Figure 1.9: Execution of task with deadline using Map-Reduce data of 1.2GB MB.

The picture shown illustrates the given input length of 500 MB with the x-axis time limits and a minimal map amount reducing spaces on the y-axis. The configuration consisted of 6 free maps and 6 free reduction slots. Thus, the graph represents that whenever the timeframe is 20 seconds or greater, the minimal map requirement is fulfilled and work is planned.

DOI: https://doi-ds.org/doilink/02.2024-13498221

The Figure 1.9 displays observations for the data input length of 1.2 GB considering the x-axis deadlines and the minimal map amount reducing the necessary Y-axis spaces. The configuration consisted of 6 free maps and 6 free reduction slots. The chart shows that when 48 secs or more is provided, the minimal need for the map is to decrease spaces and work is planned.

We have enhanced the actual time clustered scheduling method to draw the minimal map and reduced criterion for conducting job scheduling in Hadoop with deadlines. We calculated the number of resources needed to accomplish a task for a certain period of time. To this end, we suggested the concept of estimating parameter values: filtering ratios, units processing fees in mapping tasks, processing unit data costs in order to decrease tasks, communications costs incurred in the transmission of unit information. Our observation indicates that resource consumption will reduce for a given data length with increased timeframes. Furthermore, if data amount grows and the time limit remains unchanged, the need for resources will rise. When the need for resources grows, we may satisfy the requirement by extending physical or virtual nodes dynamically to the current cluster or by providing a workable timeframe[8].

III. YARN SCHEDULER

The YARN schedulers that each distributionrecommends, and whether each distribution supports node labeling.

A YARN scheduler determines how resources are allocated on a Hadoop cluster. When you configure a scheduler, you define the scheduler and the queues that the scheduler manages. The scheduler allocates cluster resources to the queues. The resources that are allocated to a queue determine the resources that are available to jobs that are submitted to the queue. By integrating queues within schedules, multiple tenants can share a Hadoop cluster. Depending on your organization's policies, you might use a fair scheduleror a capacity scheduler. For example, the users in an organization might share a single Hadoop cluster. The organization can allocate an equal number of resources to the jobs that users submit. To allocate resources evenly, the organization can employ a fair scheduler. On the other hand, an association of organizations might collectively fund a Hadoopcluster. Each organization in the association has different computing needs and invests a certain stake in the cluster. Based on an organization's investment, the organization is guaranteed a certain amount of cluster resources. To divide and allocate cluster resources between the organizations in the association, the association can employ a capacityscheduler.

Fair Scheduler A fair scheduler allows the users in an organization to share resources evenly on a Hadoop cluster. The scheduler distributes resources using weights. The weights determine the amount of clusterresources that are available to a queue. The available resources in a queue are shared evenly between jobs in the queue. When you use a fair scheduler, smaller jobs that require less time can access resources that are being used by larger jobs. As a result, users do not have to wait for larger jobs to finish before smaller jobs can run.

Example. Using a **Fair Scheduler** :

You are a Hadoop administrator for a healthcare company that owns a Hadoop cluster where the company runs different data processes. You want to allocate resources to each of the processes. The healthcare company might have two data processes that run simultaneously. One process might handle inventories while another process might

DOI: https://doi-ds.org/doilink/02.2024-13498221

handleprescriptions.

IV. DEADLINE CONSTRAINT SCHEDULER

In this scheduling strategy user specified deadline constraints at time of scheduling the jobs, ensures that jobs scheduled for execution will met deadline. It focuses on issues of deadlines and increase system utilization. Its deal with deadline requirement by cost model of job execution which consider parameters such as input size of data, data distribution, map and reduce run time etc. whenever any job is scheduled it is checked by scheduler weather it will completed within time specified by the deadline or not. Advantages

Deadline scheduler more focus on Optimization of Hadoop implementation.

This scheduling technique also increases system utilization.

Disadvantages

1. There is restriction on nodes which incurs cost,should be uniform in nature
2. Some restrictions or issues of deadline which isspecified by user with jobs.

The following is the proposed algorithm approach to run multiple jobs using fair scheduler with deadlines:

Algorithm : Fair-FIFO:Fair Scheduling with deadlines for multiple jobs

Input: job1,job2...job$_n$ *stored (pool) in HDFS under directory*
output : Job scheduling using Fair scheduler with resource provisioning

```
ScheduleJob(Clusterc, job1, job2, ...job_n(pool))
mapslot ⟵ Availablemapslot
reduceslot ⟵ Availablereduceslot
for each job injoblist job is selected in FIFO order
    i ⟵ nextjob(joblist)
    schedulervalue ⟵ fair.FairScheduler
    recources ⟵ availableresources(mapslots, reduceslots)
set preemptiontime
allotedtime ⟵ deadline
if preemtiontime ⟵ allotedtime
remove job from pool
Kills and execute the next job from pool if not equally
    shared(which utilizes available resources)
remaining resouces are alloted to next job in pool
repeat
```

In case of fair scheduling jobs are given equal shareof resources if during the period of execution if any process is not given access during the sharing time then the job that is running is preempted (forcibly) killed and the other job uses the resources.

At a point in time, the queue Inventory might be running 2 jobs, while the queue Prescriptions mightbe running 3 jobs. The following Figure shows the proportion of cluster resources that each job can use:

DOI: https://doi-ds.org/doilink/02.2024-13498221

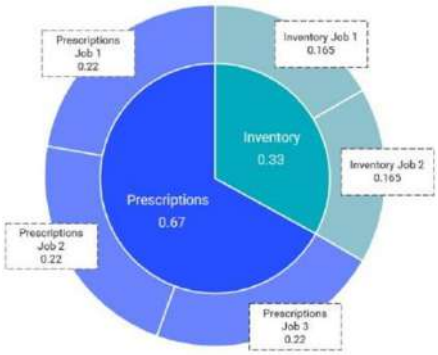


Figure 1.10: Default scheduler set the queue capacity for various jobs.

Note that the resources allocated to the queue Inventory are divided in half. Each halve is allocated to one of the two jobs in the queue. The resources allocated to the queue Prescriptions are divided into three parts. Each part is allocated to one of the three jobs in the queue.

The first job in the queue Inventory completes. The following Figure 1.11 shows the proportion of cluster resources that are available to the remaining jobs:

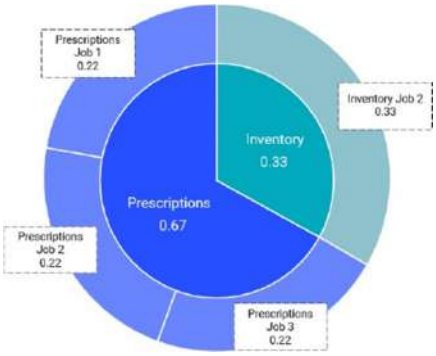


Figure 1.11: Cluster resources available to execute the jobs.

Capacity Scheduler A capacity scheduler allows multiple organizations or multiple environments in a single organization to share a large Hadoop cluster. The scheduler distributes resources using capacities that are allocated to each organization or environment. The capacities determine the percentage of cluster resources that are guaranteed to each organization or environment. The scheduler distributes any excess capacity that is underutilized. For example, a single organization can use a capacity scheduler to assign resources to test and production environments to guarantee each environment a certain percentage of cluster resources. When the production environment is not fully utilizing its allocated resources, the test environment can use the production environment's excess cluster resources.

Similarly, when the test environment is not fully utilizing its allocated resources, the production environment can use the test environment's excess cluster resources. Additionally, the organization does not have to create and

DOI: https://doi-ds.org/doilink/02.2024-13498221

maintain different Hadoop clusters for each environment. Example. Using a Capacity Scheduler You are the Hadoop administrator for an organization, QuickSecurity. Your organization owns a Hadoop cluster that you share with another organization, NextDoorHack. Due to each organization's processing requirements and stake in the Hadoop cluster, you collaborate and determine that QuickSecurity should be guaranteed 80% of the Hadoop cluster's resources, and NextDoorHack should be guaranteed 20% of the cluster's resources. QuickSecurity uses the Hadoop cluster to run processes for two departments, HR and Sales. HR requires 45% of the organization's resources, while Sales requires 55% of the organization's resources. NextDoorHack uses the Hadoop cluster specific queue for internal data processing

The following Figure 1.12 shows the percentage of cluster resources that are guaranteed to jobs in the queues:

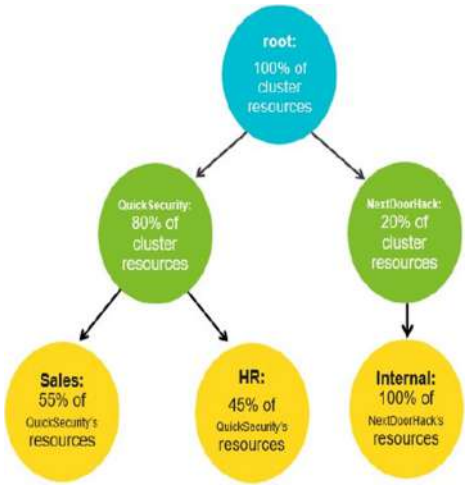


Figure 1.12: Capacity scheduler set the queuecapacity for various jobs.

The following Figure 1.13 shows the percentage of cluster resources that are guaranteed to jobs in the queues:

DOI: https://doi-ds.org/doilink/02.2024-13498221

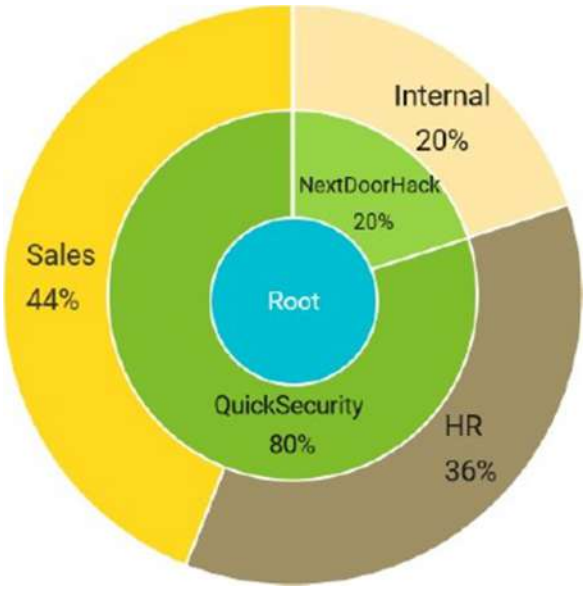


Figure 1.13: Resources available for job execution.

The queue Sales is running two large jobs and the queue Internal is running a small job. The job in the queue Internal does not require all of the queue's resources [9]. The following image shows the resources that might be available to each job:

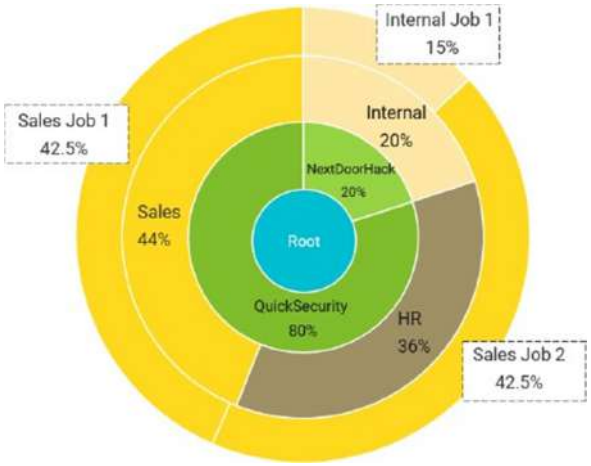


Figure 1.14: Resources available to run each job.

DOI: https://doi-ds.org/doilink/02.2024-13498221

The job in the queue Internal completes. The following Figure 1.13 shows resource available to run each job and 1.14 shows how the capacity scheduler might reallocate the excess resources to the remaining jobs

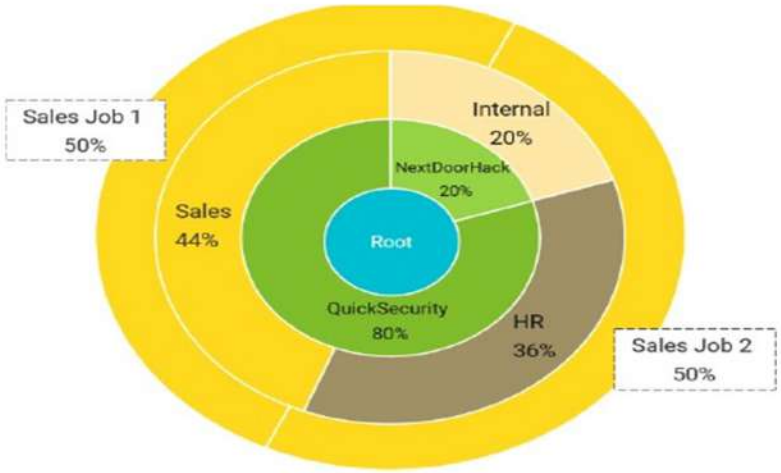


Figure 1.15 Excess resources allocated to remaining jobs

Example. Directing Spark Jobs to a Queue You work at an organization that runs a majority of data processing jobs on the Spark engine. To ensure that Spark jobs have access to cluster resources, you direct Spark jobs to the queue Spark_only. To set the YARN queue for Spark jobs, you can configure the following property in the Hadoop connection

As Shown in the Figure 1.15 The job submission process implemented by Job Submitter does the following: User submits the job. (Step 1) .Scheduler will compute the minimum number of map and reduce slots required for the job completion (step 2). If schedulability test fails user will be notified to enter another deadline value. If passed, job will be scheduled. (Step 4).

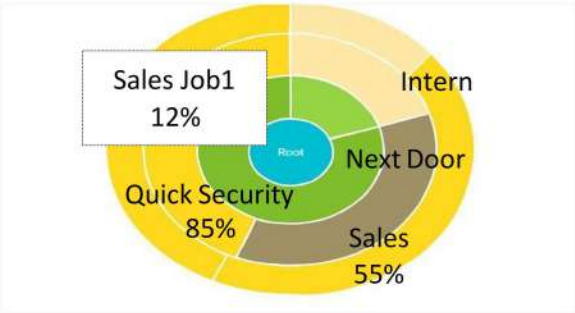


Figure 1.16 : Capacity Scheduler with deadline

RDS determines the resource allocations to individual jobs based on the estimations of job completion time and the predictions on future resource availability. To realize fine-grained resource control, RDS divides job execution into

DOI: https://doi-ds.org/doilink/02.2024-13498221

control intervals and builds performance models for estimating job progress, from which overall job completion time can be inferred. Based on the job completion time estimation and resource availability prediction, RDS derives the optimal job scheduling via an online receding horizon control.

The following is the proposed algorithm that is improving the efficiency and performance of executing the multiple jobs by making use of the queues with different queue sizes and the jobs with more priority are allotted more space and low priority jobs can be allotted the less space by which we can make the resource utilization in more efficient way than the default scheduler.

Algorithm : Cap-FIFO:Capacity Scheduling with deadlines for multiple jobs

Input: Q1⟵ $\{job1, job2...job_n\}$
$Q2 \longleftarrow \{job1, job2...job_n\}$
output : Job scheduling using Capacity scheduler with resource provisioning

1 : $ScheduleJob(Cluster\ c, Q1, Q2)$
2 : $mapslot\ Q1 \longleftarrow Availablemapslotqueue\ capacity$
3 : $reduceslot\ Q1 \longleftarrow Availablereduceslot Queue\ capacity$
4 : $mapslot\ Q2 \longleftarrow Availablemapslotqueue\ capacity$
5 : $reduceslot\ Q2 \longleftarrow Availablereduceslot Queue\ capacity$
6 : $schedularvalue \longleftarrow Capacity.CapacitySchedular$
7 : $for\ each\ job\ \{i, j\}\ in\ \{Q1, Q2\}\ \ is\ selected\ in\ FIFO\ order$
8 : $i \longleftarrow 0$
9 : $j \longleftarrow 0$
10 : $recources \longleftarrow availableresources(MapslotsQ1, Reduceslots Q1, MapslotsQ2, ReduceslotsQ2)$
11 : $set\ preemptiontime\ Q1\ Q2$
12 : $allotedtimeQ1 \longleftarrow deadline1$
13 : $allotedtimeQ2 \longleftarrow deadline2$
14 : $if\ preemtiontimeQ1 \longleftarrow allotedtimeQ1$
15 : $if\ preemtiontimeQ2 \longleftarrow allotedtimeQ2$
16 : $remove\ \ job\ from\ Q1, Q2$
17 : $i \longleftarrow nextjob(Q1)$
18 : $j \longleftarrow nextjob(Q2)$
19 : *repeat till n*

V. CONCLUSION

Here in this paper we provided information of various schedulers and how efficiently resources can be utilized with jobs with high priority and within dead line. Futher these algorithms may be still improvised by using high end processors by parallel execution on multiple nodes.

References

[1] Yuan Tian, Scott Klasky, Hasan Abbasi, Jay F. Lofstead, Ray W. Grout, Norbert Podhorszki, Qing Liu, Yandong Wang, and Weikuan Yu. Edo: Improving read performance for scientific applications through elastic data organization. In CLUSTER, pages 93–102. IEEE, 2011. 109

[2] Weina Wang, Kai Zhu, Lei Ying, Jian Tan, and Li Zhang. A throughput optimal algorithm for map task scheduling in Map-Reduce with data locality. SIGMETRICS Performance Evaluation Review, 40(4):33–42, 2013.

[3] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Aria: Automatic resource inference and allocation for Map-Reduce environments. In Proceedings of the 8th ACM International Conference on Autonomic Computing, ICAC '11, pages 235–244, New York, NY,

DOI: https://doi-ds.org/doilink/02.2024-13498221

USA, 2011. ACM.

[4] Yandong Wang, Yizheng Jiao, Cong Xu, Xiaobing Li, Teng Wang, Xinyu Que, Cristian Cira, Bin Wang, Zhuo Liu, Bliss Bailey, and Weikuan Yu. Assessing the performance impact of high-speed interconnects on Map-Reduce. In TilmannRabl, MeikelPoess, Chaitanya K. Baru, and Hans-Arno Jacobsen, editors, WBDB, volume 8163 of Lecture Notes in Computer Science, pages 148–163.Springer, 2012.

[5] MoMohammed Razia Alangir Banu, Arpita Gupta. MRI Brain Tissue Segmentation and Tumour Localization Using Hybrid Deep Learning Techniques. JCHR (2023) 13(3), 1264-1270| ISSN:2251-6727.

[6] https://jchr.org/index.php/JCHR/article/view/673/676.

[7] 1Ms. Md. Razia Alangir Banu , 2Dr. Arpita Gupta. HYBRID DEEP LEARNING ENHANCES MRI BRAIN TISSUE SEGMENTATION AND TUMOR LOCALIZATION. Eur. Chem. Bull. 2023,12(issue 8), 9050 - 9057. Doi : 10.48047/ecb/2023.12.8.737 ISSN 2063-5346 https://https://www.eurchembull.com/uploads/paper/9047d4ea63367043d1533829209ed2fe.pdf

[8] Deep Spectral Time-Variant Feature Analytic Model for Cardiac Disease Prediction Using Soft Max Recurrent Neural Network in WSN-IoT M. Safa, A. Pandian, Gouse Baig Mohammad, Sadda Bharath Reddy, K. Satish Kumar, A. S. Gousia Banu, K. Srihari & S. Chandragandhi

[9] Journal of Electrical Engineering & Technology https://link.springer.com/article/10.1007/s42835-023-01748-w

DOI: https://doi-ds.org/doilink/02.2024-13498221