DESIGN OF COMPREHENSIVE FRAMEWORK ON OPTIMIZATION METHODS IN DISTRIBUTED CLUSTERS

Dr. Kiran Kumar Pulamolu[1,*], Dr. D.Venkata Subramanian[2], Dr Krishnaraj[3]

[1,3]Professor, Sasi Institute of Technology and Engineering
[2]Professor, School of Computer Science, Hindustan Institute of Technology & Science, Chennai

Abstract

MapReduce is a popular, open source programming paradigm to handle big data which is an industry standard large scale data processing system used by many companies like Yahoo, Google, Face book, etc. The YARN framework uses low resource fairness algorithms such as FIFO, Capacity, Fair, DRF scheduler, whereas these schedulers are not suitable for heterogeneous Hadoop clusters. Therefore, an Enhanced Combined Regression Ranking (eCRRYARN) algorithm was proposed to enhance resource fairness. The proposed algorithm uses linear regression model to estimate the expected resources to be availed by the tenants. The order ranking is given to the estimated resource and the resources shared as per the ranking provided. Hence, the Hierarchical Hadoop Cluster Resource Sharing (HHCRS) algorithm has been adopted for hierarchical distributed cluster aiming to design a cost effective cluster for organization which is spread across the globe.

Keywords: Distributed Cluster, Resource Fairness, Resource Sharing, Hierarchical Cluster, MapReduce.

Received on 06 July 2018, accepted on 21 August 2018, published on 12 September 2018

3rd International Conference on Green, Intelligent Computing and Communication Systems - ICGICCS 2018, 18.5 - 19.5.2018, Hindusthan College of Engineering and Technology, India

doi: 10.4108/eai.12-9-2018.155745

[*]Corresponding author. Email: kiran@sasi.ac.in

1. Introduction

Nowadays, the growing demand for the functionalization of technology has been found in a wide spectrum of modern technology. In recent years, research is progressing in the areas of Cloud, and Big data, especially in distributed clusters. Resource Scheduling and utilization play a prominent role in getting numerous benefits in a big data environment. Many of the organizations are equipped with technological infrastructure through scattered and limited servers, storage devices, networking capability and computer systems. But the processing and storage capabilities of desktops and servers are too limited to support big data applications and distributed clusters. Redundant data in multiple desktops may lead to confusion and wastage of resources primarily in storage.

Very often, the high-end servers are idle and sometimes the servers are running under high peak load. A similar kind of problem does exist in distributed and storage cluster. Many corporate companies and government agencies are spending a lot of money to migrate from a non-cloud to a cloud based environment. The main challenge lies in determining the impact of investments in the light of infrastructure and resource usage for high availability and higher performance. There exists a need to move from proprietary software to open source technologies. The popular open source technologies include Linux, Hadoop and MongoDB, etc. which are used in supporting the migration from non-cloud to cloud based application and data.

A scalable Linux based distributed cluster is one of the low-cost methodologies to address these issues and to keep track of the usage of the resources, applications and Web services. A distributed hierarchical cluster helps in connecting different nodes and locations across the globe at multiple levels. However, some of the major challenges in the Hierarchical distributed clusters are Resource Fairness, Tenants' Truthfulness, Poor Resource Utilization, and Unfair Scheduling. This research work is carried out to address these key challenges and to avoid resource contention, and scarcity through scalable and fair resource sharing optimization frameworks and algorithms.

2. REVIEW OF LITERATURE

Cheng-Zhong Xu *et al.* (2015) studied a structure called CoTuner, to coordinate the configurations of VMs and resident applications for resource sharing and tuning. Georgios L. Stavrinides *et al.* (2017) recommended two synchronization models to establish communication through networks and files, required for synchronizing the nodes. Yanfei Guo *et al.* (2015) Suggested FT-MRMPI, fault tolerant MapReduce framework for HPC Clusters. The experimental results generated by applying the proposed model on a 256-nodes HPC cluster determined that FT-MRMPI succeeded in detecting the faults and in reducing the job completion time by 39%. This is a good base line for any performance measure but may not be directly applicable for a big data Hadoop cluster. There are differences in terms of performance and scalability between an open source and proprietary architecture. Yanfei Guo *et al.* (2014) presented a mechanism Flex slot, to identify the map stragglers automatically and alter the size of the corresponding slots to speed up the execution time. This technique change the memory size of slots of a slave node and resulted in 46% reduction in the completion time of a job over stock Hadoop and 22% over Skew Tune techniques.

Dazhao Cheng *et al.* (2013) proposed an approach Ant, to search the individual tasks' optimal configurations running over multiple nodes automatically. This method achieves this by dividing the nodes into a number of homogeneous sub-clusters and applies a self-tuning algorithm on individual sub-clusters. The experimental results by Dazhao Cheng *et al.* (2017) show that their flex slot technique reduces the job completion time by 46% compared to the stock Hadoop cluster and by 22% compared to Skew Tune. Shanjiang Tang *et al.* (2016) proposed a new fair resource allocation mechanism, the Long-Term Resource Fairness (LTRF). LTRF allows one to share the unused resources among other clients. It ensures that the clients should pay only for the resources they used. Though there are many research works carried out in optimal resource utilization in the Hadoop cluster, there exists a need to further improve fairness and thereby decrease the job completion time and improve the performance of CPU and Memory.

2.1 Research Gap

Based on the extensive literature review on the issues, challenges and opportunities, the research gap is summarized as below:

- Unpredictable truthfulness in resource sharing among tenants with non-trivial workloads.
- Lack of efficiency in resource sharing among tenants in heterogeneous distributed clusters.
- The resources like CPU, Memory and Disk IO are underutilized and more than one resource were not shared effectively in the distributed cluster.
- There is a lack guarantee for the reversal of shared resources in inter tenant and intra tenant clusters.

3. OBJECTIVE

The design and implementation of a cost effective multi-level distributed cluster, is primarily based on the study and analysis of various frameworks and schedulers. The most popular schedulers such as the First in First out (FIFO), Fair, DRF, LTYARN, and others are considered for the comparative study and benchmarking. The objective of the present research is to ensure the performance and fairness that could be achieved in the Heterogeneous distributed cluster for all the tenants. The research objective is summarized as below:

- To incorporate ranking and linear regression with multi variables to improve performance and fairness in both intra and inter node clusters.
- To develop a hybrid algorithm to include heuristics and Weighted Arithmetic Mean (WAM) in Yet Another Resource Negotiator (YARN) to improve utilization and performance including resource fairness.
- To build a Scalable and Comprehensive Hierarchical Distributed Cluster with the proposed algorithms for inter and intra node clusters
- To validate the improvement in performance and fairness through experiments and case studies.

4. MATERIALS AND METHODS

Many recent research studies highlighted the gaps in the problems of addressing resource management and issues in cloud based servers. The present research work was framed basing on the evaluation of the Performance and Fairness of Distributed clusters by adopting different methodologies like experimental evaluation, comparative study and load testing. Multiple environments with existing algorithms have been simulated in an open source platform with a test bed which includes inter-tenant and intra-tenant users. In most cases, low resource utilization was due to the existing design and resource allocation policies in Distributed Cluster configurations and pluggable schedulers.

Therefore, the resource utilization and performance of YARN, the pluggable schedulers should be analyzed and updated properly. In this study, the proposed algorithms have been tested in the distributed cluster environment for validating the behavior and performance with different workloads and the results were evaluated along with comparisons of its performance. The regression and ranking techniques were adopted to predict and rank the expected resources to be reserved. Depending on the order ranking of these workloads, resource sharing among tenants is done.

To further enhance the accuracy and precision of resource fairness, both Heuristics and a popular statistical technique WAM based resource sharing with fairness was implemented. In order to collect more performance data, a Scalable and Hierarchical Distributed cluster with heuristics is constructed as part of the case study for educational institutions. A quantitative analysis was done over the performance metrics to find the correlation between the key parameters and improvements in CPU and Memory usage with the proposed algorithms compared to the existing ones through multiple experiments with varying loads.

4.1 Establishment of YARN Framework

The architecture of YARN is shown in figure 1. The different daemon processes of YARN are described as follows, suppose two clients submitted their jobs at the same time: then these are assigned to the Resource Manager (RM). The RM, depending upon on the assignments

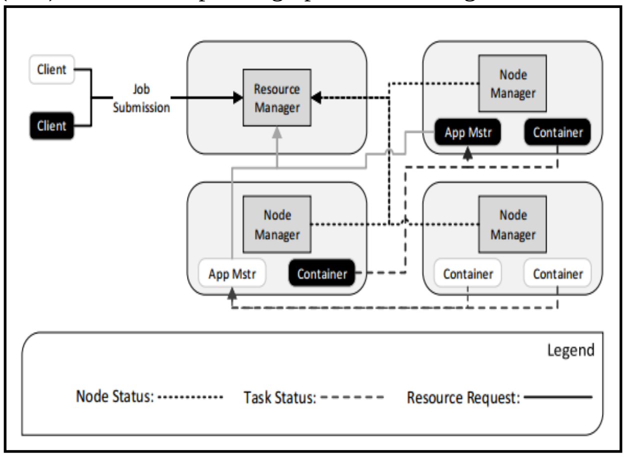


Figure 1: YARN Framework

negotiates the available resource information from the node manager and then submitted it to the AM. In turn, each AM submits a request to containers which allocate the job to the available resources depending on the NM analysis.

The container retains its vouch by sending continuous reports to the AM about the progress of the job assigned to it. The node managers monitor the application manager for utilization of the resource and execution of job status, reflecting to the RM about the status of the job, by sending periodic reports. Similarly, in runtime, the Application Master uses the interface RPC to request containers from the resource manager and to request the Node Managers to launch the scheduling algorithm.

4.2 Dynamic Heterogeneous Priority Based Flow Shop(DHPFS) Algorithm

The allocation of resources is referred to as scheduling, which means that the resource (for example, CPU of a server) is used to perform a particular task (for example, sorting job), giving a guarantee for the completion of the task within a specified time. The appropriate scheduling job is selected by following different kinds of approaches. The main task is to recognize the optimal solution for the scheduling job on different processors or machines. It is critical to note that the number of jobs or processors may differ from job to job. In this work Dynamic Heterogeneity Priority based Flow shop scheduling algorithm (DHPFS) has been proposed with all the key considerations such as scheduling job's work location used to perform different types of jobs.

DHPFS is a special case of scheduling process and client job operations have to be performed in some strict order. The DHPFS enables the flow control in a specific sequence for each job through the set of resources 1,2,….m and other sets of machines. This process can maintain the process's constant flow, with minimum waiting time and minimum idle time.

4.2.1 DHFSS ALGORITHM

The allocation of resources is referred to as scheduling, which means that the resource (for example, CPU of a server) is used to perform a particular task (for example, sorting job), giving a guarantee for the completion of the task within a specified time. The appropriate scheduling job is selected by following different kinds of approaches. The main task is to recognize the optimal solution for the scheduling job on different processors or machines. It is critical to note that the number of jobs or processors may differ from job to job. In this work Dynamic Heterogeneity Priority based Flow shop scheduling algorithm (DHPFS) has been proposed with all the key considerations such as scheduling job's work location used to perform different types of jobs. The flow shop process is as shown in Fig. 2.

DHPFS is a special case of scheduling process and client job operations have to be performed in some strict order. The DHPFS enables the flow control in a specific sequence for each job through the set of resources 1,2,….m and other sets of machines. This process can maintain the process's constant flow, with minimum waiting time and minimum idle time.

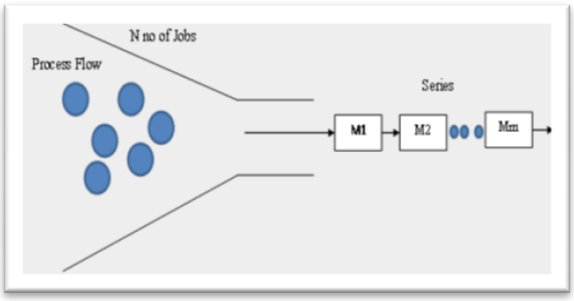


Figure 2.Flow Shop Process

This proposed DHPFS is taken into account by considering the following assumptions and are summarized as

The particular operation of jobs in any kind of machine may not be preempted. Similarly, each and every machine processes on only one job; in turn each and every job is processed on only one machine.

Set-up time is contained both the time of processing and self-regulating job position in the job sequence.

The process time regarding the scheduling operation is defined on the machine as fixed, but in case of no work on machine it is zero.

Scheduling is the method by which processes or data flows are given access to system resources (e.g. processor time, communications bandwidth).The need of a scheduling algorithm is essentially generated as a requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously).

The scheduler is mainly governed by the following performance metrics concerning the above assumptions:

Throughput – is a metric of efficiency that the total No. of processes could be completed in the time of execution per unit.

Processor Utilization or CPU: The scheduler has to keep the CPU or processor utilization as busy as possible, specifically in Latency

Response time –The total time taken from the initial action (submitted user request) to the generation of the first response.

Turnaround time – The total time needed from submission to completion of user process. Alternatively, it is the sum of time periods waiting to get into memory.

The following parameters are consider for minimize the processing time to avoid the memory less, resource contention, scarcity of resources, over-provisioning and resource fragmentation

Table.1 Parameters

Parameters	Details
S_{ij}	Starting time of job i on machine j
R_j	Ready time of job j
C'	Optimal time value
C_{max}	Maximum time value
C_{im}	Completion time of job i on machine m
C_{ij}	Completion time of job i on machine j
PT_{ij}	Processing time of job j on machine i
i	Index for machines
j	Index for jobs
m	Number of machines
n	Number of jobs

Minimize:C_{max}

Subject to:

$C_{max} \geq C_{im}$ $for\ all\ i,$

$C_{ij} = S_{ij} + PT_{ij}$ $for\ all\ i\ and\ j$

$S_{ij} \geq R_j$ $for\ all\ i$

$C_{ij} \geq C_{i},j - 1 + PT_{ij}$ $for\ all\ i,$

$C_{ij} \geq 0$ $for\ all\ i,j$

In the YARN infrastructure based Hadoop framework use the DHPFS process to avoid some of issue. The final schedule depends on their order in all two-machine problems implanted in the problem statement

Let three finite sets J, M, O where

$J$is defined as a set of jobs $1, \ldots, n$,

$M$is defined as a set of machines $1, \ldots, m$, and

O is defined as a set of operations 1, … , m.

Denote

$J_i$the$i - th\ job$ in the permutation of jobs

p_{ik} Processing time of the job $J_i \in J$ on machine k.

$(\forall i \in J)\ (\forall k \in M)$: v_{ik} = $waiting\ time\ (idle\ time)$ on machine $k$before the start of the $job\ J$

$(\forall i \in J)\ (\forall k \in M)$: w_{ik} = $waiting\ time\ (idle\ time)$ of the job $J_i$after ruining processing on $machine\ k$, while waiting for machine $k + 1$ to become free

Define the decision variables is as following

$$\forall i, j \in J: x_i j = \begin{cases} 1 & if\ job\ j\ is\ allocated\ to\ jth\ position\ in\ the\ permutation\ \ i.e\ j_i = j \\ 0 & otherwise \end{cases} \quad (1)$$

The following for mutation use the permutation DHPFS scheduling process

$$\forall i \in J: \sum_{j=1}^{n} x_{ij} = 1 \quad (2)$$

$$\forall j \in J: \sum_{i=1}^{n} x_{ij} = 1 \quad (3)$$

$$\forall k \in M - \{m\}: w_{1k} = 0 \quad (4)$$

$$\forall k \in M - \{1\}: v_{1k} = \sum_{r=1}^{k-1} \sum_{i=1}^{n} p_{i,r} x_{1,i} \quad (5)$$

$$(\forall i \in J - \{n\})\ (\forall k \in M - \{m\}): v_{i+1} + \sum_{j=1}^{n} P_{jk} x_{i+1,j} + w_{i+1,k} = w_{ik} + \sum_{j=1}^{n} P_j, k + 1 x_{ij} + v_{j+1,k+1} \quad (6)$$

$$C_{max} = \sum_{i=1}^{n} (v_{i,m} + \sum_{j=1}^{n} p_{jm} x_{ij}) \quad (7)$$

The above formulation can help to avoid the memory less, resource contention, scarcity of resources, over-provisioning and resource fragmentation in term of using Heterogeneity Priority based Flow shop scheduling algorithm (DHPFS) in YARN infrastructure based Hadoop framework.

4.3 eCRR-YARN to share resources in YARN

A novel resource sharing model, enhanced Combined Regression and Ranking (eCRR) in YARN is implemented to bring resource fairness in sharing among multiple tenants, based on the rank assigned to them. The eCRR-YARN scheduler enables tenants to share multiple resources with fairness in fully distributed cloud clusters. The main goal of the implemented system is to share resources among multi tenants based on the ranking given to workloads by considering the resources preempted by tenants early. The Multiple Regression model gives you the expected resource reserved, based on the resources demanded, resources allocated, and resources preempted.

With the help of the expected resources reserved, the Order Rankings algorithm is used to assign ranks to the workloads submitted. In turn, these rankings were used to calculate the accurate resource reservation. This model shares resources with good fairness among tenants. To predict estimated resources required, a simple linear equation has been adopted, which is given below:

$$y = a + b1x1 + b2x2 + \ldots\ldots\ldots bnxn \qquad (4.1)$$

In the equation 4.1, y represents the estimated resources reserved. a,b1, b2.. are constants, x1 represents the resources allocated, x2 represents the resources demanded and x3 represents the resources preempted. By taking the y value of all the tenants into consideration, ranks will be assigned to each tenant. Finally, the ranks thus generated are used to calculate the actual resources to be allocated to each tenant using the formula.

$$R_{ri} = R_{di} - (R-1)*C \qquad (4.2)$$

In the equation 4.2, R represents the Rank of the given job. R_{di} represents resource demanded by the tenant and R_{ri} is the resource reserved for that tenant and C is the constant, which can be fixed based on the average resource for tenants.

4.3.1 PSEUDO CODE

```
Enhanced Combined Regression Ranking algorithm

$R_a$: Available Resources in Cluster.
$R_a = (R_{a1} \ldots R_{an})$ Resources Allocation. Rai denotes resource allocation for client i.
$R_d = (R_{d1} \ldots R_{dn})$ resources demanded by tenants.    Rdi denotes resources demanded by client i.
if $R_{di}$ less than $R_{ai}$ then
        $U_{ai} \leftarrow R_{di}$            #Allocate demanded resources
        $U_{pi} \leftarrow R_{ai} - R_{di}$     #Resources Preempted
Else
        $R_{eg} \leftarrow$ LinearRegression($R_{ai}$,$R_{di}$)
        R <- Rank($R_{eg}$)
        $R_{ri} \leftarrow R_{di} - R*C$
        Update Client i.
End
```

Let's consider that 50 shares are allocated to each tenant among the four tenants designed as A, B, C and D. For suppose A demands for 30 shares, immediately the request will be processed as the demand is less than the allocated resources. Following that, if B requests for 70, C for 60 and D for 80 resources, then the number of resources to be allocated to these three tenants will be calculated using eCRRYARN algorithm. The order of ranking was given basing expected demand of the resource requirement, for example consider the case of B,C and D order ranks were given as 2, 1 and 3 respectively. Then the actual resources to be allocated is calculated using $R_{ri} = R_{di} - (R-1)*C$, Where C is a constant depends on Standard Deviation of resources allocation to tenants.

A) $R_{rb} = 70 - (2-1)*10$
$R_{rb} = 40$, so 40 resources will be allocated to tenant A

B) $R_{rb} = 60 - (1-1)*10$
$R_{rb} = 60$, so 60 resources will be allocated to tenant B

C) $R_{rd} = 80 - (3-1)*10$
$R_{rb} = 60$, so 60 resources will be allocated to tenant C

D) $R_{rd} = 100 - (4-1)*10$
$R_{rb} = 60$, so 60 resources will be allocated to tenant D

4.4 Heuristics based Resource Sharing YARN: HRS-YARN

The HRS-YARN scheduler enables the tenants to share resources with fairness in Multi node cloud clusters. The primary advantage of the newly implemented system is to share resources between multi tenants based on the resources available in the cluster by updating the heuristic table to efficient fairness. The heuristic table maintains the information regarding the resources that a tenant lent and borrowed to and from other tenants. This interest free loan lending model helps in attracting tenants towards sharing resources. The WAM is used to calculate the average resource requirement of all the individual tenants belonging to a cluster before sharing the resources among them. WAM is a measure of the fundamental affinity of a set of quantitative observations with different degrees of importance for these observations. Each observation is weighted depending on its importance relative to other observations. The weighted arithmetic mean is calculated by dividing the summation of the products of observations and their weights with the total weight.

The HRS-YARN scheduler enables the tenants to share resources with fairness in Multi node cloud clusters. The primary advantage of the newly implemented system is to share resources between multi tenants based on the resources available in the cluster by updating the heuristic table to efficient fairness. The heuristic table maintains the information regarding the resources that a tenant lent and borrowed to and from other tenants. This interest free loan lending model helps in attracting tenants towards sharing resources. The WAM is used to calculate the average resource requirement of all the individual tenants belonging to a cluster before sharing the resources among them. WAM is a measure of the fundamental affinity of a set of quantitative observations with different degrees of importance for these observations. Each observation is weighted depending on its importance relative to other observations. The weighted arithmetic mean is calculated by dividing the summation of the products of observations and their weights with the total weight.

4.4.1 WEIGHTED ARITHMETIC MEAN:

The weighted arithmetic mean is a measure of fundamental affinity of a set of quantitative observations with different importance for these observations. Each observation is weighted depending on its importance relative to other observations. The weighted arithmetic mean is calculated by dividing the summation of the products of observations and their weights with total weight.

Mathematical Definition:

Formally, the weighted mean of a non-empty set of data which means:

$$w_1 + w_2 + \cdots + w_n$$

4.4.2 HRSYARN Pseudo code

```
$R_a$: Available Resources in Cluster.
$R_{a1}$=($R_{a1}$ ...$R_{an}$) Resources Allocation.
#$R_{ai}$ denotes resource allocation for client i.
$R_d$ =($R_{d1}$...$R_{dn}$) resources demanded by tenants.
#$R_{di}$ denotes resources demanded by client i
if $R_{di}$ less than $R_{ai}$ then
    $U_{ai}$ <- $R_{di}$        #Allocate demanded resources
    $U_{pi}$ <- $R_{ai} - R_{di}$   #Resources Preempted & Update heuristic table
else
    W <- $\sum R_i * R_{di} / \sum R_i$
    while : execute pending tasks
        if W < $R_a$ && $U_p$ >0 then
            $U_{ai}$ <- $U_p + R_{di}$
        if W ≈ $R_a$ && $U_p$ >0 then
            $U_{ai}$ <- $R_{di}$+$U_p$%50
        if W > $R_a$ && $U_p$ >0 then
            $U_{ai}$ <- $R_{di}$+$U_p$%25
        else
            Wait until there is a released resource ri from client I
    end
Update heuristic table for client i.
```

4.4.3 RESOURCE SHARING IN HRSYARN

The above algorithm demonstrates the Pseudo code of HRSYARN. In this, R_a represents the set of resources available with the cluster and R_{ai} denotes the resources allocated to tenant i. Let R_d be the resources demanded by the tenants at time t whereas, R_{di} represents the resources demanded by Tenant i. The main aim of HRSYARN is to share resources with fairness among multiple tenants of a cluster. This technique uses the Weighted Arithmetic Mean (WAM) to bring out fairness in resource sharing. To understand the working of the algorithm, let’s consider case of two Tenants A and B having 50 shares each and at time t0, if A requires only 20shares whereas B requires 70. Then B will borrow the unused shares from A. Later on, if A requires excess resources than its share, in turn it will take resources back from B. At this moment, unused resources available with B will be return to A. Under circumstances of insufficient shares at B, it calculates the number of resources to be returned to A based on Weighted Arithmetic Mean (WAM) and U_P value (U_P: Resources preempted by a resource or tenant). Weighed arithmetic mean (WAM) is calculated by dividing the summation of product of each individual tenant’s weights and resources demanded with summation of weights of all the tenants. Total resources demanded by A will be allocated to it if WAM calculated is less than available resources and U_P is greater than zero. Durable extract of the resources demanded will be assigned to A if WAM is nearly equal to available resources and U_P is greater than zero. Minimum resources from demanded resources will be allocated to A if WAM is greater than available resources and U_P is greater than zero. If Zero resources are available, then the tenant has to wait till some tenant preempt its resources. Heuristic tale should be updated with every transition done.

4.5 Hierarchical Hadoop Cluster Resource Sharing (HHCRS)

The Hierarchical Hadoop Cluster depicted in Figure 2, data must be managed in such a way that the information should be accessed from all the master nodes. The meta-data such as data size, file format, access path and so on, of all the HDFS files obtained from the name nodes are recorded and maintained by the global name node. When a file is newly generated or existing file is modified, the respective name node sends the updated metadata to its parent name node. The Global name node stands at the top of the hierarchy. It consists of metadata; data that holds the details of the sub name nodes and access logs used to access the data stored in name nodes. Users can query and obtain the information based on metadata in the global name node. The user query is further automatically converted into more detailed address for performing operations on files, which includes cluster address, HDFS directory and block number. The location where the program is being executed is identified, using the cluster address, paths regarding the data nodes are located using the HDFS directory, and data are identified with their block numbers.

4.5.1 ED-MEDIA USING COST EFFECTIVE HIERARCHICAL HADOOP CLUSTER

The hierarchical Hadoop cluster can be found its use to process big data stored in multiple levels in an organization shown in Figure 3. The structural design of the hierarchical methodology is as shown in Figure.4. The architecture shows a master-slave relationship between global name node and data nodes which are coined as master node and slave node. Every intermediate node also acts as name node. Global name node is the vital node which processes user requests. It splits the user tasks into sub-tasks and share out them to the corresponding name nodes. Global node also takes care about the metadata stored in global name node which is used to process user queries. The function of global name node is also includes to manage the metadata of all the files available in every data node. The data node is installed on each cluster and is enabled to run sub-jobs allocated by parent name node. Hierarchical Hadoop cluster provides a global view of all files stored at different clusters(26).

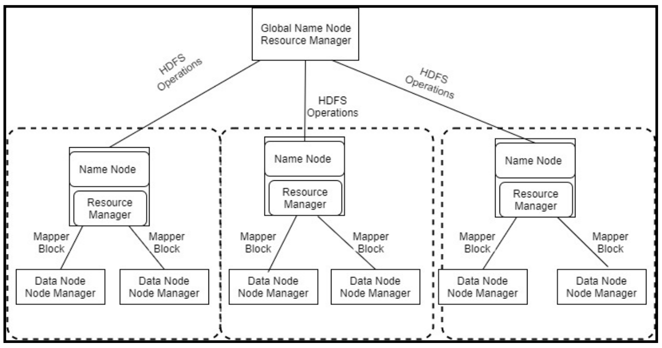


Fig. 3: Block diagram Hierarchical Hadoop Cluster

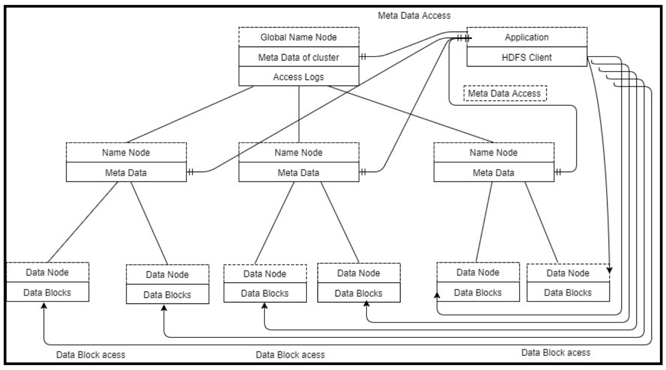


Fig. 4: Hierarchical Hadoop Distributed File System

4.5.2 ALGORITHM

1: HHC RESOURCE SHARING PSEUDO CODE

```
$C_a$ : ($c_{a1}$; $c_{a2}$; : : : ; $c_{an}$) Child clusters available
$c_{ai}$ :  choosen cluster from available clusters to process
$R_a$: Available Resources in Cluster.
$R_{a1}$= ($R_{a1}$ ...$R_{an}$) Resources Allocation. $R_{ai}$ denotes resource allocation for client i.
  $R_d$ =($R_{d1}$...$R_{dn}$) resources demanded by tenants.    #$R_{di}$ denotes resources demanded  by client i
  if $R_{di}$ less than $R_{ai}$ then
      $U_{ai}$ <- $R_{di}$            #Allocate demanded resources
      $U_{pi}$ <-  $R_{ai}$ – $R_{di}$       #Resources Preempted & Update heuristic table
  else
      WAM <- $\sum R_i * R_{di} / \sum R_i$
      while : execute pending tasks
            if WAM < $R_a$ && $U_p$ >0 then
                  $U_{ai}$ <- $R_{di}$
            if WAM ≈ $R_a$ && $U_p$ >0 then
                  $U_{ai}$ <- $R_{di}$%50
            if WAM >  $R_a$ && $U_p$ >0 then
                  $U_{ai}$ <- $R_{di}$%25
            else
  Wait until there is a released resource ri from  client I
  Update heuristic table for client i.
```

The maintenance of fairness in resource allocation among all the tenants in Hierarchical Hadoop cluster stands as one of the major challenge. To probe the resource fairness, a novel algorithm named Hierarchical Hadoop cluster Resource Sharing (HHCRSYARN) is adopted. The steps involved in this algorithm are as shown in the above algorithm. Let us consider n number of child clusters in Hierarchical Hadoop Cluster and then the Global Name node chooses the child cluster based on the resources and data availability. In that the total number of resources available in cluster is R_a. R_a defines the set of individual resources allocated to each tenant. R_d determines the total number of resources demanded by all the tenants at a particular instance of time where R_{di} represents the resources demanded by client i. W describes the set of weight of workloads of all the tenants and W_i represents the weight of individual tenant i. When a tenant sends a request for resources, the demand will be immediately granted if the demand for resources (R_{di}) is less than resources available (R_{ai}) with it and user resource preempted (U_o) will be updated. Whenever the tenant demands more resources than its availability, the request will be granted by taking WAM into consideration. Weighed arithmetic mean (WAM) is calculated by dividing the summation of product of each individual tenants weights and resources demanded with summation of weights of all the tenants. If WAM is less than resources available and U_p is greater than zero, then the total resources demanded will be allocated. If WAM is approximately equals to resources available and U_p is greater than zero, durable extract of the resources demanded will be assigned to tenant. Suppose WAM is nearly equal to available resources and U_P is greater than zero and then minimum resources from demanded resources will be allocated to tenant. Otherwise WAM is greater than available resources and U_P is greater than zero, no resource available for allocation. Then the tenant needs to wait until some other tenant preempts its resources. All the transitions regarding resource allocation and preemption of every tenant should be updated in heuristic table. An efficient Hierarchical Hadoop cluster can be designed using HCC resource sharing algorithm adopted. Hierarchical Hadoop cluster framework can be employed to empower educational institutions to alter the way of using Information Communication Technology (ICT).

5. RESULT AND DISCUSSION

Table 2. Throughput

Number of System	**PFSS**	**DDEP**	**DHPFS**
20	53	78	93
40	58	81	95
60	68	83	96
80	73	88	95
100	78	91	95

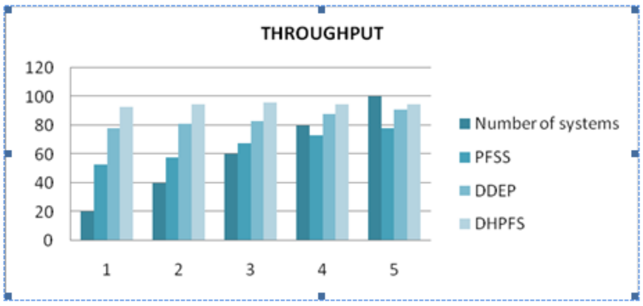


Figure 5. Throughput

Figure 5 shows the comparative results for the average throughput time. According to the contrast analysis of the experimental results, the performance of DHPFS is high and it increases the throughput by 20% compared to the remaining two algorithms.

Table 3. Completion Time

Number of System	PFSS	DDEP	DHPFS
20	10.56	12.67	2.5
40	30.67	50.56	7.5
60	32.89	60.75	10.54
80	50.9	90.87	30.56
100	50.78	92.54	40.67

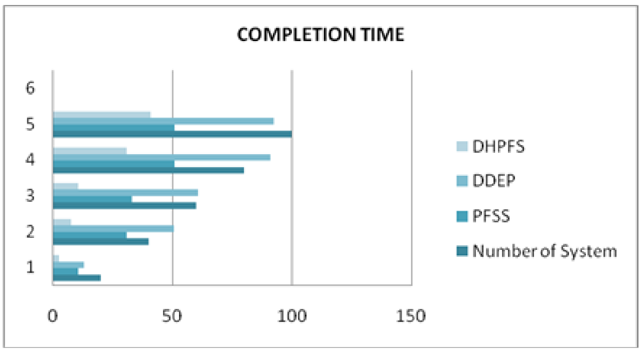


Figure 6. - Completion Time

Table 4. scalability

Number of System	PF SS	DD EP	DHP FS
20	12	10	15
40	18	20	25
60	30	24	35
80	35	28	45
100	32	30	50

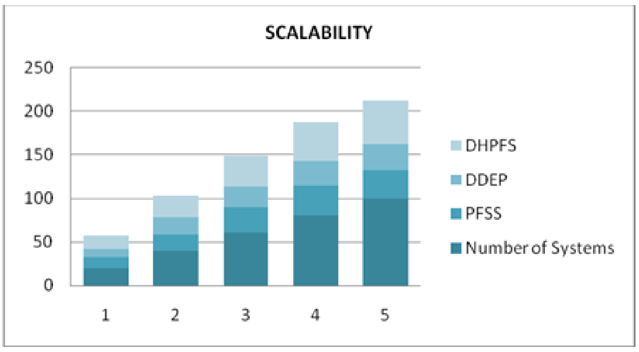


Figure 7 shows the results for the average Scalability.

. Based on the contrast results of all the three algorithms, PFSS, DDEP and proposed DHPFS algorithms, the proposed algorithm shows the promising results in terms of processing time. It reduces the processing time by 30% compared with the other two algorithms. In Hadoop YARN, Dynamic Heterogeneity Priority based Flow shop scheduling algorithm had succeeded in overcoming the issues like memory less, resource contention, scarcity of resources, over-provisioning and resource fragmentation. Compared to the other algorithms, Permutation Flow Shop Scheduling and Drivers for Dynamic Essential Path, DHPFS algorithm is providing promising results in terms of throughput, processing time and scalability. In Cloud Environment, DHPFS provides an efficient resource allocation and it also maximizes the utilization of physical resources as it deals with supplying the resources on the basis of user demand. It improves throughput by 20% and reduces the response time by 30%.Various levels of tests have been conducted to validate the resource sharing performance, fairness and job completion time, using the proposed algorithms and also comparing them with the other algorithms. To summarize the key findings, a scalable Hierarchical Distributed Cluster was setup with 30 nodes using Virtualization. Each node is created with the Intel dual core processor, 2 GB RAM, 50 GB HDD specification. These nodes are installed with the Linux/CentOS 6.8 operating system, Java 1.8 and Hadoop 2.7.3 to establish a hierarchical Distributed Cluster. One node acts as the Global Name node, which runs the Name Node and Resource

Manager Services to communicate with the slave clusters. Five different Java Based Map Reduce workloads have been submitted to Seven Tenants to Test the performance of the Capacity, Fair, LTYARN and HHCRS Algorithms. The experimental results show that the HHCRS Algorithm has succeeded in using the resources effectively and sharing the resources with fairness. The Performance measures for CPU utilization is shown in Tables 5, 6 and the figures 8, 9 represents the comparison of CPU and Memory utilization between the existing and the proposed fair scheduling algorithms.

Table 5: Performance Measure - CPU Utilization

SCHEDULING ALGORITHM	1	2	3	4	5	6	7	8	Average Time CPU Time
Capacity Scheduler	2972	760	6852	4470	6853	4470	760	9912	157047
Fair Scheduler	1555	343	5435	3053	5435	3053	343	8495	145710
LTYARN Scheduler	0835	623	4715	2333	4715	2333	623	7775	139950
HHCRS Scheduler	0498	285	4378	1995	4378	1995	285	7438	137250

Table 6: Performance Measure - Memory Utilization

SCHEDULING ALGORITHM	1	2	3	4	5	6	7	8	Average Memory Usage
Capacity Scheduler	546	546	416	416	416	416	546	483	42782
Fair Scheduler	033	033	903	903	903	903	033	970	38678
LTYARN Scheduler	169	169	039	039	039	039	169	106	31766
HHCRS Scheduler	224	224	094	094	094	094	224	161	24206

Fig 8: Comparison of Scheduling Algorithms – CPU Utilization

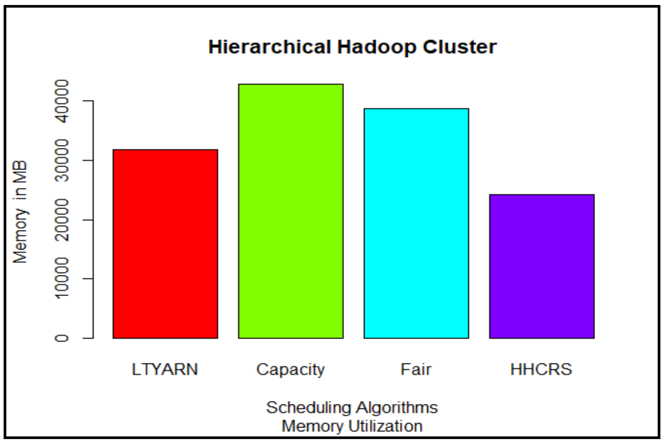
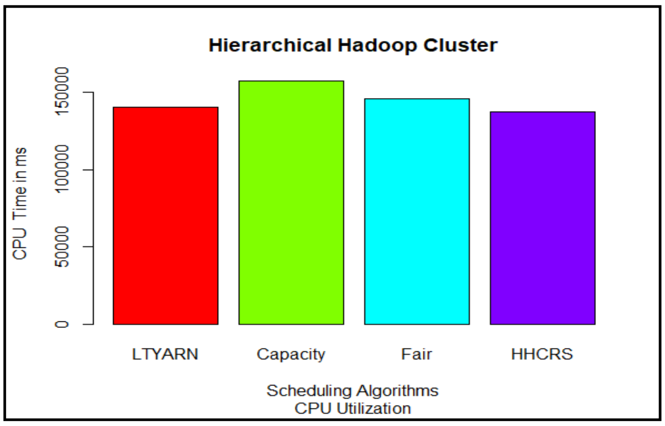


Fig 9: Comparison of Scheduling Algorithms – Memory Utilization



6. SUMMARY

This research compared the resource fairness for YARN in a heterogeneous environment for various methods, their pros and cons. The proposed algorithm HRS-YARN succeeded in attaining fairness by resource sharing and utilization of resources effectively. It produced 25% better results over LTYARN in executing map reduce programs in terms of the average time and memory usage. In eCRR-YARN, the resource sharing is based on the loan with interest free rank based discount share model. The Enhanced Combined Regression Ranking YARN improves the efficiency of the Intra Tenant Distributed cluster with multiple resources sharing with improved fairness. It can predict the ranking of different workloads based on amount resource utilization factors like CPU Intensive, Memory intensive, etc. Based on this ranking the resource sharing is done among the workloads in the Intra Tenants Distributed cluster of a client. In order to improve the performance of this model, HRS-YARN is proposed. HRSYARN uses the heuristics table to store the user preempted resources and the weighted arithmetic mean (WAM) were used to find out the average resource demanded by the tenants at a particular time interval.

On the basis of heuristic and WAM information, resources are reserved for the tenants. In this model, 30% improvement in the performance of the cluster is observed, compared to the existing models. The experimental results demonstrate the improved performance of the HHCRS algorithm over other scheduling algorithms with respect to CPU and memory utilization. It improves the resource utilization by 30% and succeeded in acquiring resource fairness in cloud based clusters. Whenever the fairness percentage increases in terms of resource sharing the performance gradually improves. Therefore, this work confirms that there exists a strong correlation between performance and resource fairness is accepted. The experiments also proved that there are very marginal differences in CPU response and memory utilization in intra and inter tenants' resource utilization.

7. FUTURE DIRECTIONS

The HDFS stores data using the n-fold technique. The n value can be configured by the administrators. In the name of replication, a large volume of storage is wasted and therefore, the performance of the storage system decreases and also the amount of bandwidth used is increased. Another issue with the distributed Cluster is the name node which is a single point of failure. Though the Secondary name node exists in the cluster, this Secondary name node and name node are configured in the Active-Passive mode. The distributed cluster may be studied further with commodity hardware with multiple name nodes in active-active configurations. Future studies are required to include Security, Privacy and Confidentiality in the e-Learning deployed over the Hierarchical distributed Cluster. The e-learning framework and appropriate data mining tools can be integrated to further enhance the proposed algorithms.

8. REFERENCES

1. K. Govindarajan, T. S. Somasundaram, V. S. Kumar, et al., Continuous clustering in big data learning analytics, in: Technology for Education (T4E), 2013 IEEE Fifth International Conference on, IEEE, 2013, pp. 61–64.
2. Di Jian and Yanfeng Peng, Research of performance of distributed platforms based on clustering algorithm., JCP 11 (2016), no. 3, 195–200.
3. X. Bu, J. Rao, C.-Z. Xu, Coordinated self-configuration of virtual machines and appliances using a model-free learning approach, IEEE transactions on parallel and distributed systems 24 (4) (2013) 681–690.
4. G. L. Stavrinides, F. R. Duro, H. D. Karatza, J. G. Blas, J. Carretero, Different aspects of workflow scheduling in large-scale distributed systems, Simulation Modelling Practice and Theory 70 (2017) 120–134.
5. Y. Guo, W. Bland, P. Balaji, X. Zhou, Fault tolerant mapreduce - mpi for hpc clusters, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ACM, 2015, p. 34.
6. Y. Guo, J. Rao, C. Jiang, X. Zhou, Flexslot: Moving hadoop into the cloud with flexible slot management, in: High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for, IEEE, 2014, pp. 959–969.
7. D. Cheng, J. Rao,Y. Guo, C. Jiang, X. Zhou, Improving performance of heterogeneous mapreduce clusters with adaptive task tuning, IEEE Transactions on Parallel and Distributed Systems 28 (3) (2017) 774–786.
8. S. Tang, B.-S. Lee, B. He, H. Liu, Long-term resource fairness: Towards economic fairness on pay-as-you-use computing systems, in: Proceedings of the 28th ACM international conference on Supercomputing, ACM, 2014, pp. 251–260.
9. J. Lin, F. Liang, X. Lu, L. Zha, Z. Xu, Modeling and designing fault-tolerance mechanisms for mpi-based mapreduce data computing framework, in: Big Data Computing Service and Applications (Big Data Service), 2015 IEEE First International Conference on, IEEE, 2015, pp. 176–183.
10. I. A. Moschakis, H. D. Karatza, Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing, Journal of Systems and Software 101 (2015) 1–14.
11. K.Wang, N. Liu, I. Sadooghi, X. Yang, X. Zhou, T. Li, M. Lang, X.-H. Sun, I. Raicu, Overcoming hadoop scaling limitations through distributed task execution, in: Cluster Computing (CLUSTER), 2015 IEEE International Conference on, IEEE, 2015, pp.236–245.
12. D. E. Difallah, G. Demartini, P. Cudré-Mauroux, Scheduling human intelligence tasks in multi-tenant crowd-powered systems, in: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 855–865.
13. G. L. Stavrinides, H. D. Karatza, Scheduling real-time parallel applications in SaaS clouds in the presence of transient software failures, in: Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2016 International Symposium on, IEEE, 2016, pp. 1–8.
14. Y. Yao, J. Wang, B. Sheng, C. C. Tan, N. Mi, Self-adjusting slot configurations for homogeneous and heterogeneous hadoop clusters, IEEE Transactions on Cloud Computing 5 (2) (2017) 344–357.
15. D. Cheng,Y. Guo, X. Zhou, Self-tuning batching with dvfs for improving performance and energy efficiency in servers, in: Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on, IEEE, 2013, pp. 40–49.
16. F. Zhou, H. Pham, J. Yue, H. Zou, W. Yu, Sfmapreduce: An optimized mapreduce framework for small files, in: Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on, IEEE, 2015, pp. 23–32.
17. K. Kambatla, A. Pathak, H. Pucha, Towards optimizing hadoop provisioning in the cloud., HotCloud 9 (2009) 12.

18. B. Sharma, T. Wood, C. R. Das, Hybridmr: A hierarchical mapreduce scheduler for hybrid data centers, in: Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on, IEEE, 2013, pp. 102–111.
19. S. Nair, J. Mehta, Clustering with apache hadoop, in: Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ACM, 2011, pp. 505– 509.
20. D. V. Subramanian, K. P. Kumar, Fuzzy based modeling for an effective it security policy management, in: SAI Computing Conference (SAI), 2016, IEEE, 2016, pp. 173–181.
21. A. H. Ibrahim, H. E.-D. M. Faheem, Y. B. Mahdy, A.-R. Hedar, Resource allocation algorithm for gpus in a private cloud, International Journal of Cloud Computing 5 (1-2) (2016) 45–56.
22. M. Al-Ayyoub, M. Daraghmeh, Y. Jararweh, Q. Althebyan, Towards improving resource management in cloud systems using a multi-agent framework, International Journal of Cloud Computing 5 (1-2) (2016) 112–133.
23. G. L. Stavrinides, H. D. Karatza, Scheduling different types of applications in a saas cloud, in: Proceedings of the 6th International Symposium on Business Modeling and Software Design (BMSD16), 2016, pp. 144–151.
24. Bhavin J Mathiya and Vinodkumar L Desai, Apache hadoop yarn mapreduce job classifi- cation based on cpu utilization and performance evaluation on multi-cluster heterogeneous environment, Proceedings of International Conference on ICT for Sustainable Development, Springer, 2016, pp. 35–44.
25. Taesup Moon, Alex Smola, Yi Chang, and Zhaohui Zheng, Intervalrank: isotonic regression with listwise and pairwise constraints, Proceedings of the third ACM international conference on Web search and data mining, ACM, 2010, pp. 151–160.