International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8, Issue-1, May 2019

An Effective Resource Management in Hadoop Cluster using Optimized Algorithm

bs Vidhyasagar, J Rajapaulperinbam, m Krishnamurthy, j Arunnehru

Abstract: ***Hadoop advances in executing the massive resources required by applications in a parallel and distributed computing environment, which uses the map-reduce framework to process the large dataset. In Hadoop we use two types of schedulers with YARN capabilities to run the application in big data environment namely Fair Scheduler and Capacitive Scheduler. Each scheduler has it is own queues and resource manager to allocate the resources to run the particular application. In this paper, introduction of PSO based centralized job queuing scheduler is used to manage and monitor the resources that will tune up the existing schedulers which gives the optimized resource utilization, speeds-up the execution and provides more active and dynamic execution of jobs in the big data environment.***

Keywords: Hadoop, PSO, Central Queuing, Parallel and distributed computing, Slots and YARN.

I. INTRODUCTION

In the current cyber world, big data and cloud have became an unavoidable technology evidencing in several domains such as social media, e-health and retails. Big data has the caliber to handle petabytes or even more amount of data. Big data is capable of performing various operation such as gathering, handling, perceiving and sharing vast amount data from different sources. This data is available in three different formats viz., structured, unstructured and semi-structured. Apache project introduced an open source tool called Hadoop to handle the large volume of the data in a cost-efficient and effective way. In this tool, multiple jobs are executed in parallel to archive the valid information. To obtain the efficiency by executing these jobs using jobs queue schedulers in appropriate manner. MapReduce concept was initially introduced by Google[10] for processing the vast amount of data..

Revised Manuscript Received on 30 May 2019.
***** Correspondence Author
Vidhya Sagar Bs*. Assistant Professor in the department of Computer Science and Engineering at SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India.
DR. J. Raja Paul Perinbam, Professor In Department Of Electronics And Communication Engineering. Kings Engineering College, Chennai, India,
Dr. M Krishnamurthy, Professor in Department of Computer Science and Engineering, KCG College of Technology. Chennai India.
Dr. J Arunnehru , Assistant Professor in the department of Computer Science and Engineering at SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India.

The open source Hadoop map-reduce framework is widely accepted by the education as well as the commercial domain[11]. It is widely accepted due to its reliability and rapid scalability making its presence in all the big data environment very helpful in processing the enormous amount of information.Mining the valuable information from these raw data precarious jobs because of non availability of resources in the cluster to execute the job. Several jobs are running parallel in the Hadoop cluster with the constrained computing resources, this affects the execution of jobs. To perform these bunch of jobs batch by batch, the MapReduce frame work employs the scheduler, and these schedulers supply the available resource to the process based on the availability of resource in the Hadoop cluster. Improper resource management may lead to low resource utilization and result in extending the job execution time, this issue can be solved by scheduling these available resource efficiently using schedulers. The resource administration and scheduling of the jobs in the Hadoop framework is managed by YARN ,one of the core components in Hadoop, YARN is responsible for allocating system resources to the various applications running in a Hadoop cluster and scheduled tasks are executed on different cluster nodes.

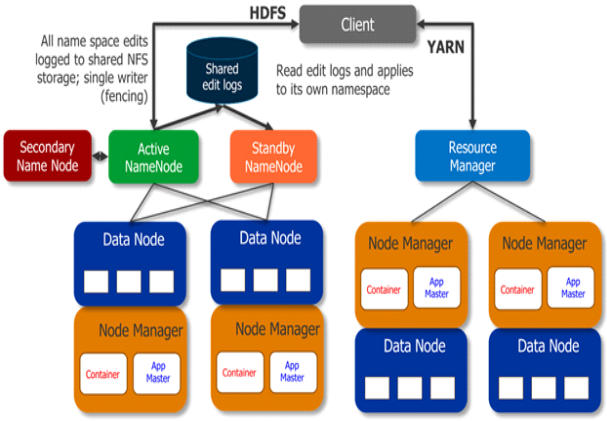


Figure 1: Architecture of YARN

YARN stands for Yet Another Resource Negotiator, the main components of YARN are Resource Manager, NodeManager component and application master component, are showed in the Figure[1]. The Namenode keep track all the active DataNodes in every two seconds. The DataNodes which has blocks size 64MB or 128MB based on the configuration and it contains replicated data in it by setting replica factor in the configuration. Which ensures the data is always available in the cluster for processing[4].

Resource Manager Component: This component is the negotiator of all the resource in the cluster.

Resource Manager is further categorized into an Application Manager that will manage all the user jobs with the cluster and pluggable scheduler. This resource manager is designed for receiving and running the applications in the Hadoop Cluster. In Hadoop 2.0, a MapReduce job will be considered as an application. ***Node Manager Component:*** This node contains job history server which will furnish the information about all the metrics of completed jobs. The NodeManager keeps track of all the users' jobs and their workflow on any, particularly given node. ***Application Master Component***: This is the component where the job resides and responsible for managing every Map-Reduce job and it concluded once execution job completes its processing.

In this paper, we consider various production workloads with different workflow jobs either dependent or independent data to process the job. For these kind of jobs we requires adequate resources to process the jobs in each scheduling queue for efficient memory management in the Hadoop cluster.

Job scheduling plays a vital role in determining the performance of any computing clustering. Especially in Hadoop clusters, this is even more critical because each job requires an enormous amount of resources concurrently. Among these significant challenges related to the job, scheduling is discussed below.

Data volume/storage, the primary challenge for the big data comes from the vast amount of raw data. This is being the thought-provoking task for the enterprise to analyze the meaningful data [12]. There is a massive volume of the data present in structure as well as unstructured format. Storing of the unstructured data is a hard-hitting job.

The format of data sources, both the homogeneous and heterogeneous resources produce the data. Specifically, the different resource cause several issues such as the format of the data and resources when compared to the homogeneous resources [13].

Data velocity, the primary goal of any computing system is, to respond immediately. The same is expected in the big data also but due to the several constraints like querying the vast data, streaming and stored data, interlinking of data storage and data extraction problem play a key role in determining the velocity of the big data [14].

Security and privacy is the primary concern in the cyber world. Most of the time the big data follow the distributed nature and geographically spread one. In this case, whenever we store the data, we have to compile all the privacy and data security laws as per that particular environment [15].

Data sharing [16] is the primary purpose of the big data and connectivity among the different users and application are still challenging one. Due to heterogeneous environment data sharing required a common standard and middlewares be required with the proper authorization and data exchange protocols.

Cost [17], is a significant function in deciding the operation. Building a new infrastructure for the big data environment requires more cost which includes the cost for upgrading the master and slave nodes, networking and acquiring other necessary services.

The main objectives of this paper are listed below:

• Choose optimized node for the task assignment in the current availability of the DataNodes in the Hadoop cluster.

• A significant amount of reduction in the cost of the node improves the job execution performance in the Hadoop cluster.

• Reduce the frequent configuration changes in the virtual cores of the data nodes.

The rest of the paper is organized as follow,§ II present the various job schedulers available in the Hadoop job scheduling. § III briefly discuss the
several existing works in the job scheduler optimizing.§ IV present the architecture of the proposed novel particle swarm optimization (PSO) based centralized job queuing system for Hadoop clusters. § V explains the integration of the PSO into the job scheduling in the YARN resource management. §VI analyzes the performance of the proposed system regarding four different parameters and results are compared with the three different job schedulers, and finally § VII concludes the paper with the future direction.

II. JOB SCHEDULERS

The Hadoop job scheduler [1] primarily classified into YAQ-C, Capacitive scheduler and fair schedulers based on efficient resource utilization, priority and time slice. The main scope of active Hadoop scheduler is to minimize the delay and maximize the throughput of allocating jobs to the cores in the processor [2-3]. Map and Reduce operations are fit into the core to process the job, Sometimes these Core are also called a slot. The mapped operation takes more slots than reduce operation. Commercially two schedulers are available in Hadoop [4] namely Fair and Capacitive scheduler.

• **Fair scheduler**

Fair scheduler [6] was developed by Facebook. Multiple applications that are all allowed to execute based on configurable attributes in fairscheduler.xml. The fair scheduler allocates an equivalent amount share of available resources to respective jobs to make sure the fairness among the distributing the resources between the different jobs. If anyone of the jobs is completed earlier and not being used, then the available resource will be used by other application.

• **Capacity Scheduler**

The Capacity Scheduler was initially developed by Yahoo and adopted by Apache foundation [7-8]. Multiple queues can be assigned based on the adjustment of configuration attributes in capacitivescheduler.xml. Each queue assigns a variable resource allocation which can be made for a different set of applications. In each queue and its child queue assigns a percentage of resources which can be Hadoop allocated in the cluster for executing MapReduce jobs shown in the figure [1-2].

```
yarn.scheduler.capacity.root.capacity=100
yarn.scheduler.capacity.root.queues=Engineering, Humanities
yarn.scheduler.capacity.root.Engineering.capacity=60
yarn.scheduler.capacity.root.Engineering.CSE.capacity=20
yarn.scheduler.capacity.root.Engineering.MECH.capacity=20
yarn.scheduler.capacity.root.Engineering.ECE.capacity=20
yarn.scheduler.capacity.root.Engineering.EEE.capacity=20
yarn.scheduler.capacity.root.Engineering.ADMIN.capacity=20
yarn.scheduler.capacity.root.Humanities.capacity=40
yarn.scheduler.capacity.root.Humanities.academics.capacity=40
yarn.scheduler.capacity.root.Humanities.admin.capacity=40
yarn.scheduler.capacity.root.Humanities.sports.capacity=20
```

Figure 2: Configuration of Capacity Scheduler

- **YAQ**

Jeff Rasleyet al. [30] proposed method called Yaq. This yaq estimate the approximate time for each task has to wait before beginning it was processing when it placed in a queue which is running a local node. This time estimation helps in ordering the tasks in the node's queue for processing. Authors proposed two different favors of YAQ viz centralizeYAQ (YAQ-C) and distributed YAQ (YAQ-D).

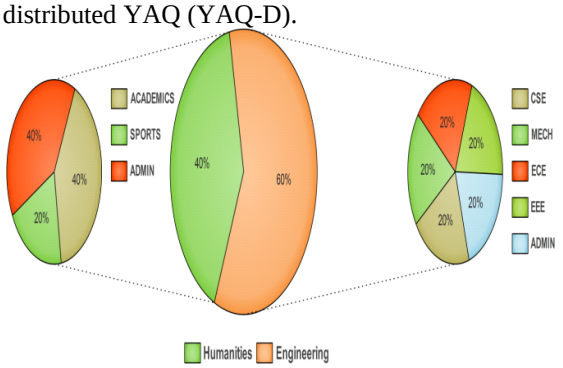


Figure 3: Capacitive scheduler workloads

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>


<?xml version="1.0"?>
<allocations>
  <queue name="queue">
    <minResources> 8192mb,0vcores</minResources>
    <maxResources> 17384mb,0vcores</maxResources>
    <maxRunningApps>10</maxRunningApps>
    <maxAMShare>0.1</maxAMShare>
    <weight>2.0</weight>
    <schedulingPolicy>fair</schedulingPolicy>
    <queue name="sample">
      <aclSubmitApps>WordAPP</aclSubmitApps>
      <minResources>4096 mb,0vcores</minResources>
    </queue>
<allocations>
```

Figure 4: Configuration of Fair Scheduler

III. RELATED WORKS

Sudha Sadasivam et al., [23] has proposed PSO implement Genetic Algorithm(GA) which includes the main features of the GA viz mutation and crossover. This helps the PSO algorithm to advance usages of the resources which resulting in finishing the tasks before the deadline. Sandholm and Lai [24] proposed dynamic proportional scheduling for the multi-user Hadoop environment. The authors claim that their technique provides efficient job prioritization and allocation in the cluster resources and offers isolation among the jobs[24]. Polo et al. proposed method called RAS[25], this helps in the escalation of the asset usages along with the closely monitoring the process ending time in the multi-job Hadoop cluster workloads. Zho et al.[26] proposed a Resource Attribute Selection based job scheduling algorithm. This proposed technique assesses its allocated resources and transmits the same to the master in the cluster. The implementation node uses these details and schedules the future jobs based on the requirement of the resources and cap between the already allocated resources and the upcoming jobs. This transmitted information will be stored in the job information history for the further assessments. Multi-Objective Scheduling Algorithm of Many-Task in Hadoop called as MOMTH [27] was proposed by Nita et al. Author performs the scheduling by resources concerning their unbiased operations and the processed allocated with the time-bounded constraints such as cost and deadline. Tang et al. [28] proposed an algorithm to find out the dual deadline, i.e., deadline for map job and the deadline for reducing job. Each job finishing time will be equal to the deadline of the Reduce job. Pop et al. [29] proposed a periodic job scheduling algorithm because the primary goal of the scheduler is based on estimating the number of resources needed to schedule a job in the Hadoop cluster. This technique used the periodic role and job deadline along with the transmitted information among the cluster will use to schedule the job.

IV. ARCHITECTURE OF THE PSO-CG JOB SCHEDULER

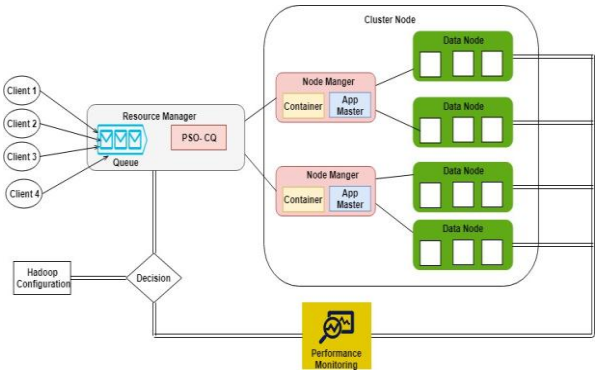


Figure 5: PSO based centralized job queue for YARN in Hadoop cluster

PSO-CQ finds the adaptive configuration to process the jobs submitted by different users. Each user submitted job are stored into the queue for further processing into the DataNode. In Hadoop, there are twenty two configuration parameters available to set the configuration to process the jobs in the minimal DataNode. Each DataNode is monitored to analyze the performance of DataNode.

V. PARTICLE SWARM OPTIMIZATION

In 1995, Keenedy and Eberhart proposed natural inspired algorithm [19] called Particle Swarm Optimization (PSO). This primary evolution of this algorithm has brought the swarm intelligence into the computing. These PSO techniques have proven several time that to solve a different range of optimization issues containing training of the artificial neural network [18] and minimizing the function when compared to other algorithms like Genetic Algorithm, Ant Colony Optimization(ACO) [20]. In a standard PSO structure, the position of the particle is portrayed by the vector called as X. The velocity of the vector is denoted as V. The initial population of the optimization explorations start from side to side of a multi-dimensional solution space. The position of each particle will be persistently adjusted by concerning the previously learned knowledge and other particles in the same population. These particles progressed

Step1: Generate an initial population of scheduling lists by topology sort set the initial value to X_{id};
Step2: Evaluate the scheduling length (Schedule each scheduling list), find out X_j;
Step3: Update the particle position (scheduling list) according to the evolution equation (1, 2) to get a new feasible solution;
Step4: Schedule each scheduling list, if scheduling length of current p_i is better than pi of the particle, update the p_{id}, as do p_{gd};
Step5: Termination criterion is met? No, step3; otherwise step 6;
Step6: Output the configuration.

concurrence with the formula below Equation[1-2].

$$v_i = v_i + c_1 r_1 (p_i - x_i) + c_2 r_2 (p_{gd} - x_i) \text{-- (1)}$$

$$x_i = x_i + v_i \text{ -- (2)}$$

Acceleration constant is represented as c1 and c2. Two different random number which range from 0 to 1 are represented as r1 and r2. In equation 1 there are three different parts. First one represents the inertia of the previous velocity, the second one denotes the "cognition," which has private knowledge of its own. The final one is "social," which denotes cooperation among the aggregate particles.

If the summation of the speeding up leads to the velocity v_i to overdo $v_{max,d}$ in that dimension. In this case, the v_i is bounded to $v_{min,d}$,and $v_{max,d.}$ This province ranges from the current position to the objective position where exploration would take place.

Since 1995, in the areas of scheduling and continuous problems, the PSO has proven several times that delivered better result in discrete space [18]. There will be a direct correlation between the element vector and optimization path which will exist whenever the PSO technique in the discrete problem is deployed. The same PSO technique also modified the velocity and particle position for the centralized queue in the YARN job scheduling in the following section.

VI. SCHEDULING USING PSO

The proven accomplishments of the PSO algorithm in discrete space and optimizing the scheduling problem, has led us to propose PSO-Central Queue algorithm as shown in Figure 6. In this example we illustrated how the PSO particles are mapped into the resources and then how these scheduled. The objective is to minimize the configuration changes in the nodes and minimizing the execution time. Let's assume that there will be X tasks and we plan to distribute them among the Y nodes for processing then PSO particles can be represented as [X Y] matrix. In this situation, Y is the number of available nodes at scheduling time t and X is number of jobs. Position vector of each PSO particle is denoted by either 0 or 1 and each vector of position in the matrix will be one with remaining elements as 0. The sample matrix is shown below.

		JOB									
		1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***
NODE	***I***	1	0	0	1	0	0	0	1	0	0
	II	0	1	1	0	0	0	0	0	0	0
	III	0	0	0	0	1	0	1	0	0	0
	IV	0	0	0	0	0	1	0	0	1	1

Figure 7: Ten independent jobs and four different nodes

In the matrix as shown in the figure 7, it represent that each column and every element depicts the machine that executes the task. The second condition ensures that each task should be executed only on one machine. For example in the above position vector, task1 is executed on machine 1, task2 is executed on machine 3, task3 on machine 1 and so on.

The proposed particle swarm optimization centralized queuing (PSO-CG) algorithm can be summarized as follows:

1. The preliminary population of the scheduling is listed by the topology sort and the random values are initialized.

2. Each job in the list is initialized.

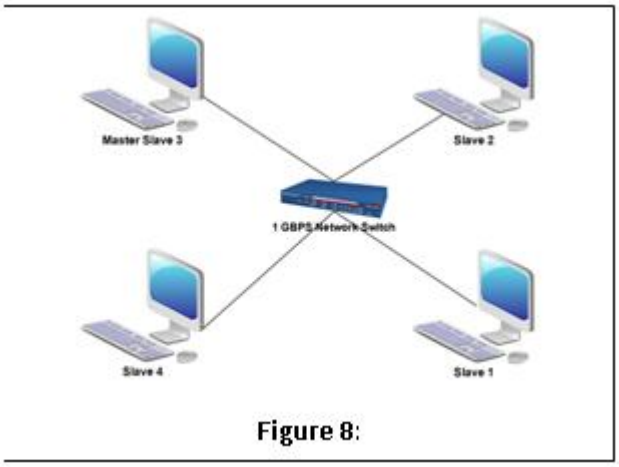


Figure 8:

3. Each job is optimized based on the current P_{id} and P_{gd}.

This step begins with calculating the fitness value of each job based on the configuration.

4. If the fitness value is best as compared to the previous value, then the present value will be updated in the local best called P_{id}.

$$v_i = v_i + c_1 r_1 (p_{id} - x_i) + c_2 r_2 (p_{gd} - x_i) \text{ -- (3)}$$

5. These steps will be repeated for all the queued job in the list.

6. Among the best local fit values, the best global value is derived.

7. This global value is represented as P_{gd}.

8. This P_{id} is kept in the resource manager,and this will be the default configuration for all the name nodes.

9. The P_{id} denotes the local best configuration for that particular name node.

10. P_{id} and P_{gd} are updated as new configuration arrives.

VII. EXPERIMENTAL RESULTS

The deployment of the Apache Hadoop distributed framework is stress-free because it is designed for all kind of hardware and developed using the java which is most reliable in all common hardware ranging from small level to significant level of the dataset. The critical feature of java is it's platform independency as it allows the same features to be enabled to Hadoop and to be installed in any operating system like MAC, Linux, Windows and even more. In this experimental setup, we have installed the Hadoop in the Ubuntu operating system in four different systems to evaluate the real-time performance of the scheduler. This arrangement contains three numbers of slave machines and one master machine. These machines are connected in the same network via a network switch. The configuration is as shown in Figure 8 and the configuration of this machine is as follows:

- Operating system: Ubuntu OS 18.04
- 64 bit I7 processor
- RAM: 8 GB
- 1-TB HDD

As we have elaborated our experimental scenario that consists of four machines in the ratio of 1 master and three slave setup. All these machines are connected via one GBPS Ethernet connection. The Job tracer and Name Node are executing the master node side, also the Task tracking and Data node are being executed in the slave node side.

Ubuntu and Open JDK were installed in these nodes. The Apache Hadoop is the common execution platform for these nodes. T able 1 presented the Hadoop configuration parameters.

Table 1: Hadoop Configuration

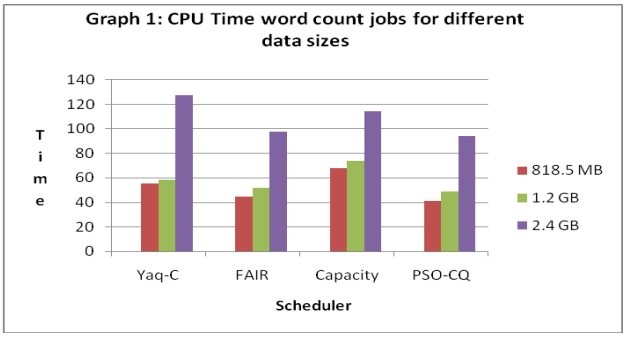
Constraints	Value
HDFS block size	64 MB
Speculative execution	Enabled
Heartbeat interval	3 S
No of map tasks per	Node 2
No of reduce tasks per	Node 1
Replication	Factor 2

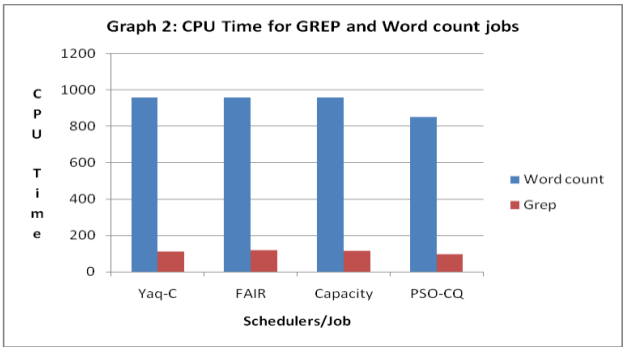
The Grep and Wordcount applications were used to evaluate the performance of the proposed job scheduler. In common these two applications are considered as the benchmark in measuring the MapReduce jobs. In this scenario, different size of datasets were given to Grep and WordCount application. The role of these applications is reading the text files and total the number of words in that particular files. In this experiment, we have used three different sizes of text files such as 818.5 MB, 1.2 GB and 2.4 GB. These size difference will help in evaluating the performance of these different job schedulers. The evaluation of the scheduler's performance is determined using the following factors.

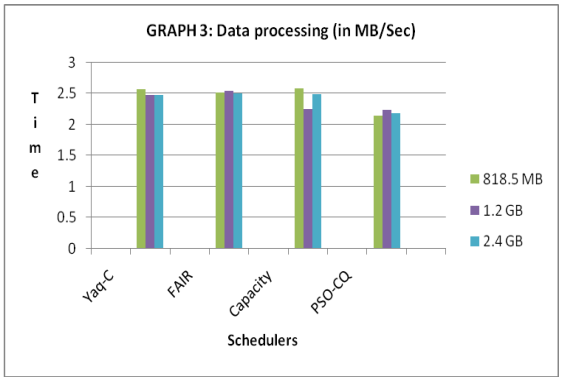
a. CPU time

The standard method to determine the CPU time is the actual time taken by the CPU to process the set of instructions. Figure 8 presents the CPU time of PSO-CQ scheduling algorithm, and the same has been compared with various other job scheduling algorithms.

PSO-CQ uses the optimized configuration in the data node and schedules these job based on the existing configurations that results in less CPU time when compared to the other job schedulers such as YAQ-C, FAIR and Capacity. The experimental values are shown in Table 2 and Table 3 show the comparison CPU time of the Wordcount application and Grep application respectively the PSO-CQ and other scheduling algorithms. from the Graph[1-3] represent the PSO-CQ cpu time is very lesser than other schedulers and data processing.







b. Data processing per second

The entire input has been separated into the small chunks and the speed of the system will be determined by some chunks at the moment. This result will be obtained from the job trackers in the Hadoop framework. The web interface allows to monitor this processing information. The performance of .the job schedulers is measured using various volume of the data.

Table 2: CPU Time word count jobs for different data sizes

Size/Schedulers	Yaq-C	FAIR	Capacity	PSO-CQ
818.5 MB	55.2	44.8	68	41.3
1.2 GB	58.4	52	73.6	48.68
2.4 GB	127.2	97.6	114.4	93.9

Table 4 and Table 5 elaborate about PSO-CQ taking very less time to process data when compared to other scheduling algorithms. The primary cause for this improvement in the proposed method is due to the less time taken for the configuration changes. This directly results in reducing the process waiting time.

Table 3: Data processing (in MB/Sec)

Size/Schedulers	Yaq-C	FAIR	Capacity	PSO-CQ
818.5 MB	2.57	2.52	2.59	2.15
1.2 GB	2.48	2.55	2.25	2.24
2.4 GB	2.48	2.5	2.49	2.19

Table 4: Data processing per in Sec (MB/Sec)

Schedulers/Job	Yaq-C	FAIR	Capacity	PSO-CQ
Word count	2.48	2.46	2.48	2.12
Grep	21.96	20.51	20.7	17.94

VIII. CONCLUSION

This paper summarizes the various challenges in designing of job scheduler and discusses different exiting job schedulers in YARN Viz., YAQ-C, FAIR and Capacity Scheduler.

This paper proposed PSO based central queuing job scheduling which updates the job scheduling based on the currently available jobs and finds the local optimum configuration for each data node. The global optimum configuration is set in the resource manager based on the local optimum. The performance of the proposed PSO-CQ was implemented and the experimental results were compared with other job schedulers like YAQ-C, fair and capacity scheduler. The performance of PSO-CQ regarding it's CPU time and data processing per sec clearly shows that the proposed PSO-CQ provides optimized resource utilization, speeds-up the execution and provides more efficient and dynamic execution of jobs among the Hadoop cluster. In future, this configuration optimization can be enhanced by deciding the changes using the deep neural networks.

REFERENCE

1. S. Suresh, N.P. Gopalan, An optimal task selection scheme for Hadoop scheduling IERI Procedia, 10 (2014), pp. 70-75.
2. Lisia S. Dias, Marianthi.G. IerapetritouIntegration of scheduling and control under uncertainties: review and challenges Chem. Eng. Res. Des., 116 (December 2016), pp. 98-113.
3. Shanjiang Tang, Bu-Sung Lee, and Bingsheng " Dynamic Job Ordering and Slot Configurations for MapReduce Workloads" IEEE TRANSACTIONS ON SERVICES COMPUTING, VOL. 9, NO. 1, JANUARY/FEBRUARY 2016 https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site
4. MohdUsama, Mengchen Liu, Min Chen,Job schedulers for Big data processing in Hadoop environment: testing real-life schedulers using benchmark programs,Digital Communications and Networks,Volume 3, Issue 4,2017,Pages 260-273,ISSN 2352-8648,
5. M. Brahmwar, M. Kumar, G. SikkaTolhit – a scheduling algorithm for hadoop cluster Orig. Res. Article ProcediaComput. Sci., 89 (2016), pp. 203-208
6. D. Yoo, K.M. SimA comparative review of job scheduling for MapReduce
7. Cloud Computing and Intel. Syst. (CCIS), IEEE Int. Conf. on. IEEE (2011)
8. J.V. Gautam, H.B. Prajapati, V.K. Dabhi, S. Chaudhary, A survey on job scheduling algorithms in big data processing IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT'15), Coimbatore (2015), pp. 1-11
9. J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

10. T. White, Hadoop: The definitive guide. " O'Reilly Media, Inc.", 2012.
11. Y. Liu, M. Qiu, C. Liu, et al., Big data challenges in ocean observation: a survey, Personal Ubiquitous Comput. 21 (2017) 55, http://dx.doi.org/10.1007/s00779-016-0980-2.
12. L. Rodríguez-Mazahua, C.A. Rodríguez-Enríquez, J.L. Sanchez-Cervantes, et al., A general perspective of Big Data: applications, tools, challenges and trends, J. Super Comput. 72 (2016) 3073, http://dx.doi.org/10.1007/s11227-015-1501-1.
13. Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung, Big Data: Related Technologies, Challenges Future Prospects, Springer Cham Heidelberg New York Dordrecht London, 2014, ISBN 978-3-319-06244-0, pp. 16–18.
14. I. Anagnostopoulos, S. Zeadally, E. Exposito, Handling big data: research challenges and future directions, J. Super Comput. 72 (2016) 1494, http://dx.doi.org/10.1007/s11227-016-1677-z.
15. S. Cheng, B. Liu, Y. Shi, Y. Jin, B. Li, Evolutionary computation and big data: key challenges and future directions, in: Y. Tan, Y. Shi (Eds.), Data Mining and Big Data. DMBD 2016. Lecture Notes in Computer Science vol. 9714, Springer, Cham, 2016.
16. UthayasankarSivarajah, Muhammad Mustafa Kamal, ZahirIrani, VishanthWeerakkody, Critical analysis of Big Data challenges and analytical methods, J. Bus. Res. 70 (January 2017) 263–286.
17. Jun Sun, WenboXu, Bin Feng, Scheduling Tasks onto Multiprocessors Using Particle Swarm Optimization, DCABES 2005 Proceedings,pp.109-113.
18. Kennedy, J., Eberhart, R.C, "Particle Swarm Optimization." Proc. IEEE Int. Conf. Neural Networks. Piscataway, NJ (1995) pp.1942-1948
19. F. van den Bergh and AP Engelbrecht. "Cooperative Learning in Neural Networks using Particle Swarm Optimizers," SACJ /SART, No 26, 2000
20. R. C. Eberhart and Y. Shi, "Comparison between Genetic Algorithm and Particle Swarm Optimization," Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447,pp 611-616, Springer.
21. P. Pant, R. Tanwar, An overview of big data opportunity and challenges, in: A. Unal, M. Nayak, D. Mishra, D. Singh, A. Joshi (Eds.), Smart Trends in Information Technology and Computer Communications. SmartCom 2016. Communications in Computer and Information Science vol. 628, Springer, Singapore, 2016.
22. G. SudhaSadasivam and DhariniSelvaraj,"A Novel Parallel Hybrid PSO-GA using MapReduce to Schedule Jobs in Hadoop Data Grids", in 2010 Second World Congress on Nature and Biologically Inspired Computing Dec. 15-17,2010 in Kitakyushu, Fukuoka, Japan, PP: 377-383.
23. Wan C, Wang C, Yuan Y, Wang H (2013) Game-based scheduling algorithm to achieve optimize profit in mapreduce environment. ICIC, LNCS 7995: 234-240.
24. Chen H, Shen Y, Chen Q, Guo M (2013) HMHS: Hybrid multistage heuristics scheduling algorithm for heterogeneous MapReduce system ICA3PP, Part I, LNCS 8285: 196-205.
25. Chen Y, Alspaugh S, Katz RH (2012) Interactive analytical processing in big data systems: A cross-industry study of MapReduce workloads proceeding of the VLDB endowment 5: 12.
26. Nita MC, Pop F, Voicu C, Dobre C, Xhafa P (2016) F- MOMTH: Multi-object scheduling algorithm of many task in hadoop. Cluster Computing 19: 1011-1024.
27. Tang Z, Zhou J, Li K, Li R (2012) A MapReduce task scheduling algorithm for deadline constraints cluster computing 16: 651-662
28. Pop F, Dobre C, Cristea V, Bessis N, Xhafa F, et al. (2014) Deadline scheduling for a periodic tasks in inter-cloud environments: A new approach to resource management. Journal of Super Computing 71: 1754-1765.
29. Jeff Rasley, Konstantinos Karanasos, Srikanth Kandula, Rodrigo Fonseca, Milan Vojnovic and Sriram Rao,"Efficient Queue Management for Cluster Scheduling", ACM, DOI: http://dx.doi.org/10.1145/2901318.2901354

AUTHORS PROFILE

Vidhyasagar BS. is currently working as an Assistant Professor in the department of Computer Science and Engineering at SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India. He is pursuing PhD as Part-Time in the Faculty of Information and Communication Engineering at Anna University, India. He received his B.E., and M.E. degree in the faculty of Computer Science and Engineering from Anna University, India. His research interests include Cloud Computing and Big Data.

DR. J. Raja Paul Perinbam received his BE and M.Sc (Eng.) degrees from Madras University, India. He got his doctorate degree from IIT, Madras, India, in the year 1984. He was with Anna University, Chennai, India, as professor in the Department of Electronics and Communication Engineering and head of the Department. of Media Sciences for more than 30 years. Currently, he is working in Kings Engineering College, Chennai, India, as professor in the Department of Electronics and Communication Engineering. His field of research is VLSI design and embedded systems. He has successfully guided more than 12 research scholars in this area.

Dr. M Krishnamurthy is currently working as a Professor and Head in the Department of Computer Science and Engineering, KCG College of Technology. He leads a research group of Computer Science and Engineering faculty in KCG. He has published more than 30 papers in refereed international conferences and journals. His primary work and research interests include large database mining techniques, incremental data mining techniques, and social network analysis and mining. He also received prestigious award of "Best Java Architect" from Cognizant Technology Solutions, Chennai in the year 2000.

Dr. J Arunnehru . is currently working as an Assistant Professor in the department of Computer Science and Engineering at SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India He received his B.E. and M.E. in Computer Science and Engineering from Annamalai University, Tamilnadu, India in 2008 and 2010 respectively. And he completed his Ph.D. in Computer Science and Engineering at Annamalai University, India. He has published more than 30 research papers in international journals and conferences. He is associated with the professional bodies ISTE and IAENG. His current research interests include computer vision, video processing, pattern recognition and machine learning.