Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IMPACT FACTOR: 6.017

IJCSMC, Vol. 6, Issue. 10, October 2017, pg.11 – 17

An Efficient Bandwidth Aware Scheduling Algorithm in Hadoop

V. Thanga Sharmila[1], S. Raja Ratna[2]

[1, 2]Department of Computer Science and Engineering & VV college of Engineering, Tamil Nadu, India

[1] sharmi1695@gmail.com; [2] gracelinrr@yahoo.com

Abstract— Hadoop is an open source software framework used for distributed storage and processing of dataset of big data using MapReduce programming model. One main issue that affects the overall performance of the Hadoop system is maximum completion time problem. The solution to this problem is to assign tasks to data local nodes so that wastage of bandwidth resource can be reduced. Thus the job completion time will be minimized. Several scheduling algorithm have been developed for improving data locality, but all of them either ignore to allocate the task to data local nodes or waste the available bandwidth. To overcome this problem, an efficient bandwidth aware scheduling algorithm in Hadoop was proposed. It is not only able to guarantee data locality but also can efficiently assign tasks in an optimized way.

Keywords— Hadoop, MapReduce, Bandwidth, Scheduling, Data locality, HDFS

I. Introduction

Big data describes data sets which are so large and complex and they are impossible to manage with traditional software tools. Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. The main component of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part known as MapReduce programming model. The Hadoop Distributed File System (HDFS) provides scalable, fault-tolerant, cost-efficient storage for your big data. MapReduce programming model contains Map tasks and Reduce tasks. The input to the Map tasks is key-value pairs and generates a set of key-value pairs as intermediate output. Map tasks can further be classified as local map tasks and nonlocal map tasks. The intermediate key and its list of values are given as input to the Reduce tasks and produce a reduced set of values as output [2]. Hadoop splits files into large blocks and distributes them across nodes in a cluster. It then transfers packaged code into nodes to process the data in parallel. It improves the data locality [1].

A small Hadoop cluster consists of a single master node and multiple worker nodes. The master node consists of a Job Tracker, Task Tracker, NameNode, Secondary NameNode, and DataNode. A slave or worker node acts as both a DataNode and TaskTracker, though it is possible to have data-only and compute-only worker nodes. These are normally used only in nonstandard applications.

In a larger cluster, HDFS nodes are managed by a NameNode server to host the file system index, and a Secondary NameNode gives snapshots of the namenode's memory structures, thereby preventing loss of data and file-system corruption. Similarly, a standalone JobTracker server can manage job scheduling across nodes.

One main issue that affects the overall performance of the Hadoop system is maximum completion time problem. The solution to this problem is to assign tasks to data local nodes so that wastage of bandwidth resource can be reduced. Thus the job completion time will be minimized. Several scheduling algorithm have been developed for improving data locality [3] – [7], but all of them either ignore to allocate the task to data local nodes or waste the available bandwidth since bandwidth is a scare resource.

In this paper, we propose an efficient bandwidth aware scheduling algorithm in hadoop was proposed. It uses OpenFlow controller to divides the bandwidth into equal time slots. Based on the measurement made on the job completion time, the proposed system assigns the task to data nodes locally or remotely. In case of assigning task to the remote data node, OpenFlow controller assigns needed time slots to the data movement. Thus it is not only able to guarantee data locality but also can efficiently assign tasks in an optimized way.

II. Related Work

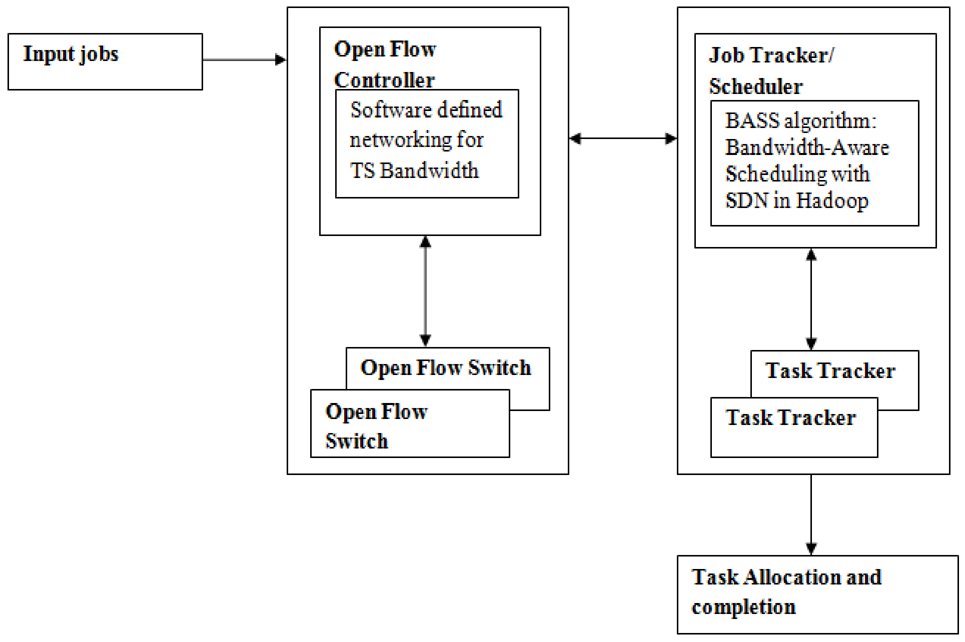
The default Hadoop FIFO scheduler schedules jobs on first come first serve basis [8]. But, it does not provide fairness. Fair scheduler provides a fair sharing of resources among multiple users [9]. It requires better resource utilization. Yahoo's Capacity scheduler provides sharing of cluster capacity among the jobs using hierarchical queues [6]. However, job pre-emption is not possible. Job Aware Scheduling for Heterogeneous Cluster [3] schedules non-local map tasks based on three criteria: job execution time, workload of the job and earliest deadline first. It increases the resource utilization and reduces the average waiting time. But if the number of non-local map tasks is lower, the average waiting time will be higher.

Job Scheduling for Multi-User MapReduce Clusters [4] allocates task slots among pools and each pool allocates its slots among multiple jobs in the pool. It provides data locality and interdependence between map and reduce tasks but reduces the throughput. Resource and Deadline-aware Job Scheduling [11] allocates resources to individual jobs based on job completion time and future resource availability and temporarily delays low priority jobs or jobs with distant deadlines. It takes future resource availability into account, minimize job deadline misses and provide job priority support but lowers the response time. CASH (Context Aware Scheduler for Hadoop) [12] uses node classifier to classify the nodes according to computation or disk capability. Scheduler searches for the jobs with requirement match and data locality. It improves performance and execution time but increase network traffic before assigning a task to a node.

Round robin algorithm [13] scans each and every server in a round robin fashion and assigns the task to the server to exploit data locality. Depends on the number of servers and the remote cost function it assigns the tasks but it is infeasible to find the optimal assignment. Delay Scheduling [14] assign free slots to the job that has less number of tasks and also it searches for local task in the job to achieve data locality. It achieves nearly optimal data locality and increase throughput but if the number of node increases, efficiency gets reduced. A Self-Adaptive Scheduling Algorithm for Reduce Start Time [15] in which reduce task reads the map output data in a copy phase and copy operations completed at a concentrated duration, that can decrease the waiting time of the reduce tasks. But it is not enable to work in shared (heterogeneous) environment.

III.Proposed System

In this paper, we use an efficient bandwidth aware scheduling algorithm in hadoop assign the task to the data local nodes to improve data locality and reduces the job completion time. Bandwidth is the scare resource in the hadoop cluster. Before the task scheduling begins, OpenFlow controller divides the residue bandwidth BW into equal time slots say, T_1, T2 … Tn. In case of assigning task to the remote data node, OpenFlow controller assigns needed time slots to the data movement. Thus all the links on the path are reserved for the particular task. Bandwidth aware scheduling schedules the task to the data nodes based on availability of the bandwidth. Figure 1 depicts the task assignment framework.



Input jobs
Open Flow Controller
Software defined networking for TS Bandwidth
Open Flow Switch
Open Flow Switch
Job Tracker/ Scheduler
BASS algorithm: Bandwidth-Aware Scheduling with SDN in Hadoop
Task Tracker
Task Tracker
Task Allocation and completion

Fig. 1 Task Assignment Framework

The proposed algorithm works as follows: In the N node hadoop cluster, assign job with M tasks. For the task Mi, it finds data local node LN and remote node RN. If LN is found and it is optimal then assign the task to LN. If LN is found and not optimal then calculate the job completion time of RN as in [21].

$$\text{Data Movement time from LN to RN} = \frac{\text{Size of the input split}}{\text{Bandwidth between LN to RN}}$$

Task Execution time = Task Computation time + Data Movement time from LN to RN

Task Completion time = Task Execution time + RN idle time

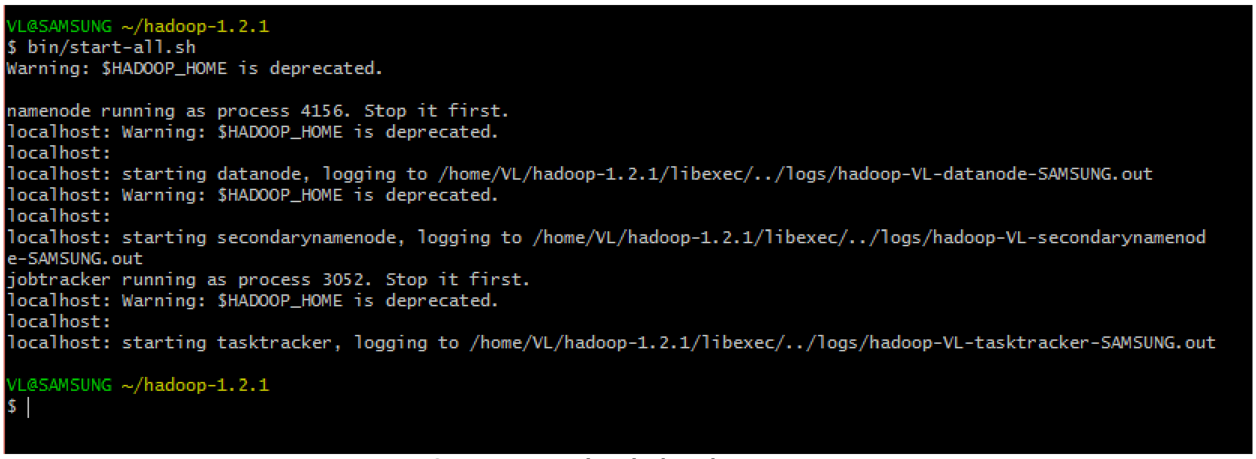
If the job completion time of RN is less than LN and the needed bandwidth is available, the assign the task to RN and also assign the required time slots. If the needed bandwidth is not available, then assign the task to LN. If LN is not found then assign task to RN and assign the required time slots.

A. Algorithm: Efficient Bandwidth Aware Scheduling in Hadoop

```
1.  Start procedure
2.  Input: Job with M tasks and N nodes
3.  Output: Assignment of tasks to nodes
4.  For Mi task, find the data local node LN and remote node RN
    4.1. If LN is found and optimal then assign the task to the LN
    4.2. If the LN is found and not optimal then
        4.2.1. Calculate the job completion time of RN and needed bandwidth for data movement
        4.2.2. If job completion time of  RN < LN and the required bandwidth is available then
            4.2.2.1. Assign the task to RN
            4.2.2.2. Assign the needed time slots
        4.2.3. If the available bandwidth is not enough then assign the task to LN
    4.3. If the LN is not found then
        4.3.1. Assign the task to RN
        4.3.2. Assign the needed time slots
5.  End procedure
```

B. *Experimental Results*

Hadoop was used in the Apache release 1.2.1, Java in the version jdk1.8.0_144 and the operating system was Windows 8.1. A cluster has been setup for the experimental purposes. The cluster is made up of six nodes interconnected by Gigabit Ethernet. One of the nodes will act as a JobTracker and Namenode while the remaining five nodes will each act as an instance of the TaskTracker and Datanode. Each TaskTracker is configured to host one map slot and one reduce slot. To improve the performance of the Bandwidth aware scheduling algorithm, we applied a efficient bandwidth aware scheduling algorithm. Hadoop has been successfully installed in windows 8.1 using cygwin.



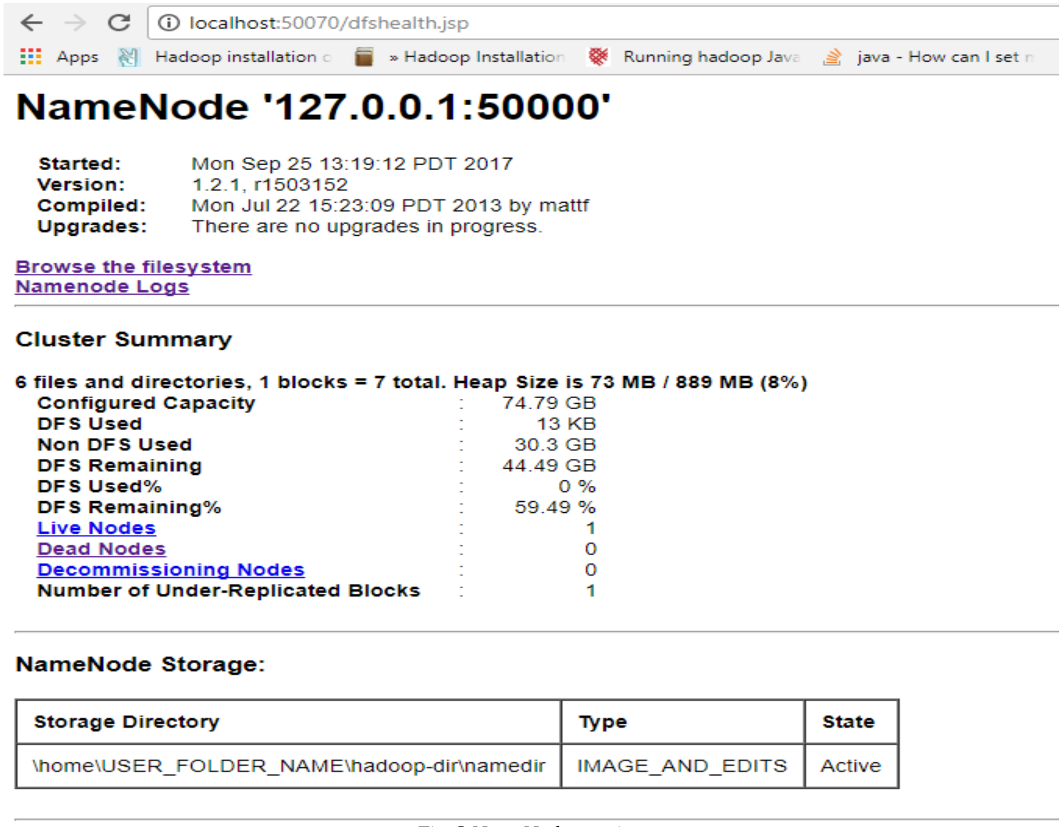
```
VL@SAMSUNG ~/hadoop-1.2.1
$ bin/start-all.sh
Warning: $HADOOP_HOME is deprecated.

namenode running as process 4156. Stop it first.
localhost: Warning: $HADOOP_HOME is deprecated.
localhost:
localhost: starting datanode, logging to /home/VL/hadoop-1.2.1/libexec/../logs/hadoop-VL-datanode-SAMSUNG.out
localhost: Warning: $HADOOP_HOME is deprecated.
localhost:
localhost: starting secondarynamenode, logging to /home/VL/hadoop-1.2.1/libexec/../logs/hadoop-VL-secondarynamenod
e-SAMSUNG.out
jobtracker running as process 3052. Stop it first.
localhost: Warning: $HADOOP_HOME is deprecated.
localhost:
localhost: starting tasktracker, logging to /home/VL/hadoop-1.2.1/libexec/../logs/hadoop-VL-tasktracker-SAMSUNG.out

VL@SAMSUNG ~/hadoop-1.2.1
$ |
```

Fig. 2 Starting NameNode and JobTracker

Once the hadoop has been installed in the windows 8.1, we first need to format the NameNode to create a Hadoop Distributed File System (HDFS). Open the cygwin terminal and execute the command "bin/hadoop namenode –format". This command will run for some time. We should be able to see message "Storage Directory has been successfully formatted". Next step is to start the namenode, datanode, secondary namenode, job tracker and task tracker. It can be done by executing the command "start-all.sh" in cygwin. This command will start all the services in Cluster and now we have Hadoop Cluster running. Figure 2 depicts the execution output of the start-all.sh. To stop all the daemons, we can execute the command "bin/stop-all.sh" in cygwin.



NameNode '127.0.0.1:50000'

Started:	Mon Sep 25 13:19:12 PDT 2017
Version:	1.2.1, r1503152
Compiled:	Mon Jul 22 15:23:09 PDT 2013 by mattf
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

6 files and directories, 1 blocks = 7 total. Heap Size is 73 MB / 889 MB (8%)

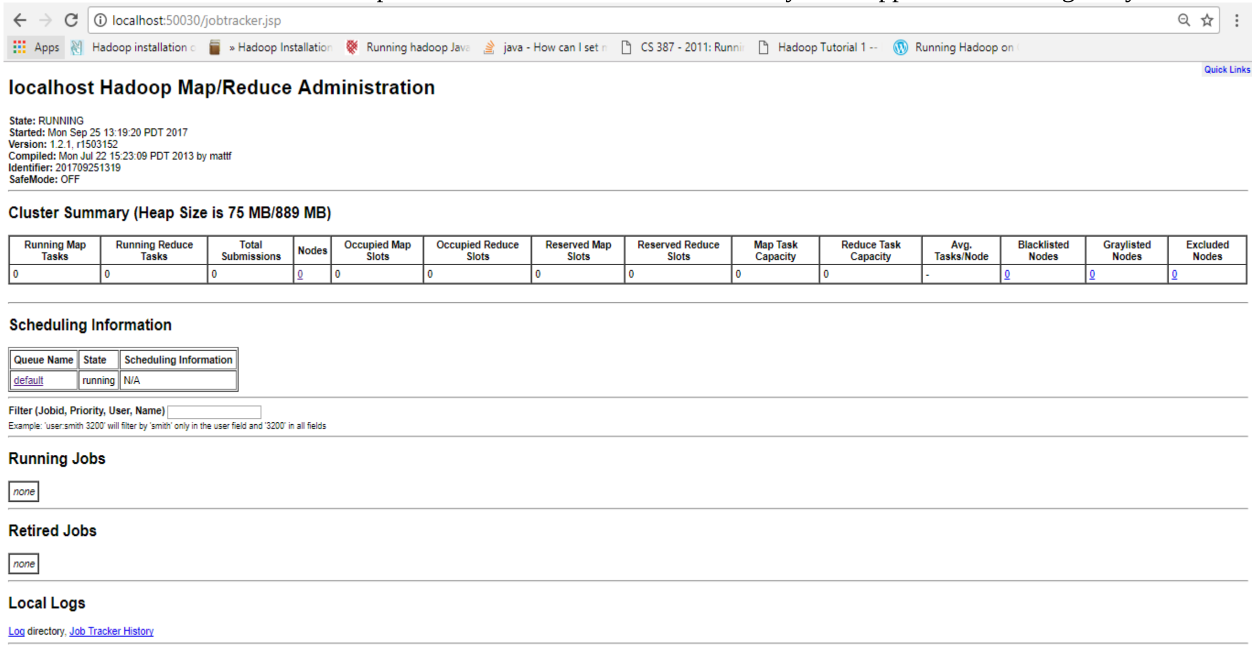
Configured Capacity	:	74.79 GB
DFS Used	:	13 KB
Non DFS Used	:	30.3 GB
DFS Remaining	:	44.49 GB
DFS Used%	:	0 %
DFS Remaining%	:	59.49 %
Live Nodes	:	1
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	1

NameNode Storage:

Storage Directory	Type	State
\home\USER_FOLDER_NAME\hadoop-dir\namedir	IMAGE_AND_EDITS	Active

Fig. 3 NameNode creation

When start-all.sh command is executed, namenode will be created. The NameNode web interface can be accessed via the URL "http://localhost:50070" in browser. Figure 3 depicts the output of the namenode creation displayed on the browser. The first section of the web page displays the name of the server running the NameNode which is 127.0.0.1 and the port 9100, when it was started, version information. In the Next section we can see Cluster Summary which represents a high view of the state of the cluster.
Files and directories, blocks: Each File system metadata item consumes 83 MB memory.
Configured Capacity: It represents total capacity of HDFS.
DFS Used: It represent space used in HDFS.
Non DFS Used: It tells about the space used for non-HDFS items like any other application running on system.


localhost:50030/jobtracker.jsp

Apps | Hadoop installation | » Hadoop Installation | Running hadoop Java | java - How can I set | CS 387 - 2011: Runni | Hadoop Tutorial 1 -- | Running Hadoop on

Quick Links

localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Mon Sep 25 13:19:20 PDT 2017
Version: 1.2.1, r1503152
Compiled: Mon Jul 22 15:23:09 PDT 2013 by mattf
Identifier: 201709251319
SafeMode: OFF

Cluster Summary (Heap Size is 75 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	0	0	0	0	0	0	0	0	-	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Retired Jobs

none

Local Logs

Log directory, Job Tracker History

Fig. 4 JobTracker creation

When start-all.sh command is executed, job tracker will be created. The JobTracker web interface can be accessed via the URL "http://localhost:50030" in browser. Figure 3 depicts the output of the job tracker creation displayed on the browser. It represents the state of the hadoop cluster and its version, cluster summary, scheduling information, running jobs and retired jobs. The cluster summary tells the information about running map tasks, running reduce tasks, total submission, occupied map slots, occupied reduce slots, reserved map slots, reserved reduce slots, map task capacity, reduce task capacity etc.

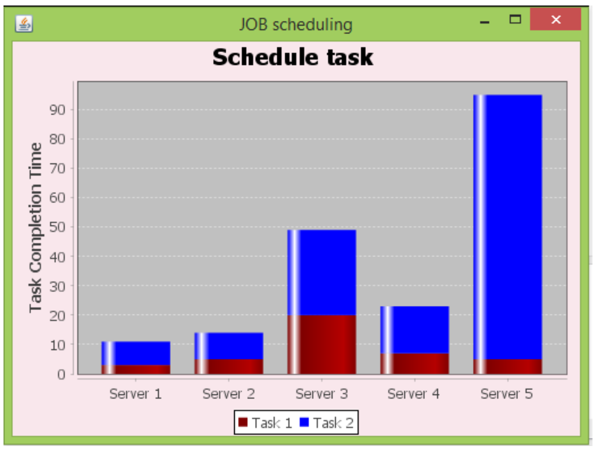


Fig. 5 Task Allocation Result

The proposed system allocates the task among five nodes based on data locality and available bandwidth. By adjusting the bandwidth of each link the performance of the algorithm is measured. The bandwidth does not affect the job completion time. Figure 5 depicts the output of the task allocation to the nodes. Initially all the nodes are free, so the tasks are assigned in First In First Out (FIFO). When new task arrives, the task is assigned to data local nodes based on minimum completion time. Thus the makespan problem can be reduced. If the data local node is not optimal for particular task, then it is assigned to remote node by allocating the required bandwidth.

IV. CONCLUSIONS

In this paper, we proposed a method to efficiently assign task in an optimized way and to effectively utilize the bandwidth. It works better than the existing algorithm by taking less time to compute and gives high accuracy. It reduces the execution time and improves the performance. However, this technique works well in homogeneous cluster. In the future, we have decided to improve the efficiency by working among heterogeneous clusters.

ACKNOWLEDGEMENT

This work was supported in part by Anna University recognized research center lab at VV College of Engineering, Tisaiyanvilai, Tamil Nadu, India.

REFERENCES

[1] Jiazhen Han, Zhengheng Yuan, Yiheng Han, Cheng Peng, Jing Liu and Guangli Li, ”An Adaptive Scheduling Algorithm for Heterogeneous Hadoop Systems”, IEEE ICIS 2017,Pages: 845 - 850, Wuhan, China, may 2017.

[2] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Google, Inc. 2004.

[3] Xiaofei Huang, Hui Zhou, Wei Wu,”Hadoop Job Scheduling Based on Hybrid Ant-Genetic Algorithm, IEEE conference 2015, Pages:226 – 229, china, 2015.

[4] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, Ion Stoica, “Job Scheduling for Multi-User MapReduce Clusters”, Electrical Engineering and Computer Sciences, University of California at Berkeley, Pages: 1 – 16, April 30, 2009.

[5] Peng zhang, Chunlin Li, Yahui Zhao,”An improved task scheduling algorithm based on cache locality and data locality in Hadoop”,17th IEEE Conference on Parallel and Distributed Computing, Applications and Technologies,China, pp. 244 – 249,Dec 2016.

[6] Ruiqi Sun, Jie Yang, Zhan Gao, Zhiqiang He, “A virtual machine based task scheduling approach to improving data locality for virtualized Hadoop”, IEEE ICIS, Taiyuan, China, June 4-6, 2014.

[7] Jian Tan, Xiaoqiao Meng, Li Zhang, “Coupling Task Progress for MapReduce Resource-Aware Scheduling”, IEEE conference, New York, 2015.

[8] B. Thirumala Rao and Dr. L.S.S.Reddy, “Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments”, International Journal of Computer Applications (0975 – 8887), November 2011.

[9] “Hadoop MapReduce Next Generation – Fair Scheduler [Online].” Available: http://hadoop.apache.org/docs/current/Hadoop-yarn/Hadoopyarn- site/FairScheduler.html [Last accessed: November, 2014].

[10] “Hadoop MapReduce Next Generation – Capacity Scheduler [Online].” Available: http://hadoop.apache.org/docs/current/Hadoop-yarn/Hadoopyarn- site/CapacityScheduler.html [Last accessed: November, 2014].

[11] Dazhao Cheng, Jia Rao, Changjun Jiang and Xiaobo Zhou,” Resource and Deadline-aware Job Scheduling in Dynamic Hadoop Clusters”, IEEE International Parallel and Distributed Processing Symposium, Pages: 956 – 965, Shanghai, China, 2015.

[12] Arun Kumar K, Vamshi Krishna Konishetty, Kaladhar Voruganti, G V Prabhakara Rao, “CASH: Context Aware Scheduler for Hadoop”, Association for Computing Machinery, Pages: 52 – 61, USA, 2012.

[13] Michael J. Fischer, Xueyuan Su and Yitong Yin, “Assigning Tasks for Efficiency in Hadoop”,SPAA’10, Thira, Santorini, Greece, June 13–15, 2010.

[14] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker and Ion Stoica, “Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling”, Association for Computing Machinery, Paris, France, April 13–16, 2010.

[15] Zhuo Tang, Lingang Jiang, Junqing Zhou, Kenli Li, Keqin Li, “A Self-Adaptive Scheduling Algorithm for Reduce Start Time”, Future Generation Computer Systems, March 26, 2014.

[16] Dan Tao, Zhaowen Lin, and Bingxu Wang, “Load Feedback-Based Resource Scheduling and Dynamic Migration-Based Data Locality for Virtual Hadoop Clusters in OpenStack-Based Clouds”, Tsinghua science and technology, pp: 149 – 159, Volume 22, Number 2, April 2017.

[17] Ya-Wen Cheng, Shou-Chih Lo," Improving Fair Scheduling Performance on Hadoop", IEEE conference, Hualien, Taiwan, 2017.

[18] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar and Andrew Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters", Microsoft Research, Silicon Valley, Pages: 1 – 20, USA.

[19] Jian Tan, Xiaoqiao Meng, Li Zhang, "Delay Tails in MapReduce Scheduling", Association for Computing Machinery, Pages: 5 – 16, London, England, UK, June 11–15, 2012.

[20] Xiaoyu Sun, Chen He and Ying Lu, "ESAMR: An Enhanced Self-Adaptive MapReduce Scheduling Algorithm", IEEE International Conference on Parallel and Distributed Systems, Pages 148 – 155, 2012, U.S.A.

[21] Peng Qin, Bin Dai, Benxiong Huang, and Guan Xu, "Bandwidth-Aware Scheduling With SDN in Hadoop: A New Trend for Big Data", IEEE SYSTEMS JOURNAL, Pages: 1- 8, 2015, china.