International Journal of Advanced Technology in Engineering and Science
Volume No.06, Issue No. 04, April 2018
www.ijates.com
ijates
ISSN 2348 - 7550

RESEARCH ON TUNING HADOOP PARAMETERS FOR HETEROGENEOUS MULTI-NODE CLUSTER

Ms.Baljinder Kaur[1], Ms.Rimpi[2]

[1,2]Guru Kashi University, Talwandi Sabo

Abstract

Hadoop, an open source MapReduce source, is becoming a de-facto platform dedicated to data storage on distributed and local machines to analyze and process large amounts of information on asset hardware. Provides a variety of parameters with default and standard single-node configuration and collections with multiple locations and applications. If it allows the user to change the configuration according to needs by editing xml files. Tuning Hadoop parameters is a challenging task as performing even a simple program requires modification of different parameters. Therefore, good border configurations can improve the Data Location, the amount of data processed and improve Network, Processor and Output / Output. This paper seeks to shed light on the literature associated with customization parameters for better optimization and efficient use of resources by proposing a framework to elevate and modify parameters to improve Hadoop performance in a multi-node multi-node collection.

Keywords: **Hadoop, HDFS, MapReduce, Parameters.**

I. INTRODUCTION

In this modern world the flooded and massive data is growing in structured, semi-structured and unstructured form consisting of audio, video, text, numbers, images, photographs, stagnant data, radar data, social media data and streaming data [1]. This data is collected from huge datasets repeatedly for immediate exploration with the help of complex applications and tools to visualize, store, route, and analyze the facts and figures from different perspectives for various sources. Organizations ranging from small to large, utilizes this Big Data as supreme fragment in the process of decision making [2]. Big Data can be categorized, as per the Volume, Velocity, Variety, Volatility, Variability, Value, Validity, and Veracity, by eight V's [3][4][5].

Doug Cutting and Michael J. Cafarella, created Hadoop in context to be data intensive to support Nutch search engine project [6]. Hadoop is designed on the basis of master-slave architecture as shown in Fig.1. It offers easy solution for distributed and parallel computing with an ability of skipping the description related to communication recovery program [7]. The master JobTracker is responsible for management of resources of cluster, job scheduling, handling fault-tolerance and monitoring the progress. The TaskTracker module, present on each of the slave nodes, is accountable for throwing parallel tasks along with task status to the JobTracker. Responsibility of slave node here is to run as well as execute one or the other Map or Reduce tasks, and is bifurcated into static computing slots [8].

International Journal of Advanced Technology in Engineering and Science
Volume No.06, Issue No. 04, April 2018
www.ijates.com
ijates
ISSN 2348 - 7550

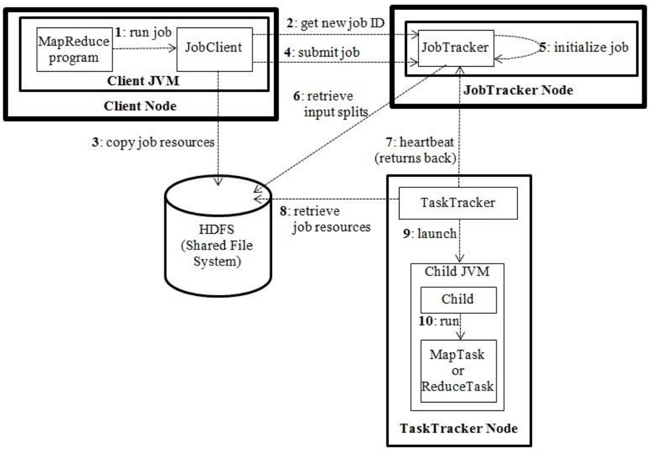


Fig. 1. Apache Hadoop Architecture

Being motivated by GFS, Hadoop Distributed File System (HDFS) is used for storage of huge data (terabytes or even petabytes) and files on several computers [4]. By replicating data on geographically diverse nodes and different servers it attains reliability. These nodes dialogues to: rebalance scattered data, create and transport replicas, and preserve high data replication rate. HDFS contains: NameNode and DataNode where the NameNode acts as master in order to manage namespace and the DataNode is slave node used to store blocks of data nearby and remote locations following distributed policy to perform read/write requests [5].

Map Reduce the model that the Hadoop soul provides high-density highways for large servers in the Hadoop collection. It consists of a static pipeline of two individual tasks: map and reduce, where map task is responsible for converting the input set of data into a different dataset by splitting each element into key-value pairs and reduce task chains the key-value pairs obtained from a map task to form set of pairs for generating output[6]. The map function performs phases: read and sort, and then store the output file to node's local storage. The reduce function performs shuffle, sort and reduce phases [7]. Data locality and Amount of data processed by Hadoop plays an important role in improving the performance of job execution in MapReduce [8].

II. HADOOP PARAMETERS CONCERNS AND RECOMMENDATIONS

A. HDFS associated parameters

Hadoops low performance in heterogeneous environment motivates to introduce a strategy for data placement to place data crossways the nodes in a way so that each node has stable load of data processing. [12] throw a light on the problems like tuning number of map/reduce tasks, cluster configuration, locality of data, application logic, blockages in system, low resource utilization, block reports and replication that degrades performance of heterogeneous Hadoop cluster along with suggestions to improve it.

B. MapReduce associated parameters

To ease the overall execution time of jobs in Hadoop [1] provides a dynamic slot scheduling approach for managing intensive I/O workloads on Task Tracker nodes in clusters by effectively leveraging CPU resources.

According to [7] Hadoop offers different ways to configure parameters in its variants while deploying and this involves vast knowledge of hardware and application for appropriate modification in configuration of a parameter. Configuring parameters by assigning wrong values result in degraded system performance and low utilization of resources at disposal. Sailfish which is one of the improved variations of Hadoop provides auto-tuning and minimized disk i/o operations to establish number of reducers and supervising intermediary data skewness dynamically.

Focusing on configuration of slot and complications of scheduling tasks, [13] proposed novel approach FRESH, for minimizing the makespan of job and enhancing fairness to support both static and dynamic slot configurations by undertaking the decisions regarding number of map/reduce slots required and allocating map/reduce jobs to available slots.

To measure the degree of CPU deployment for individual map task and IO throughput two counters for Hadoop are introduced to forecast optimum Map Slot Value using the proposed low-overhead technique [14]. Map Slot Value, which limits on the number of total map tasks that can run at the same time on single node, remains among essential parameters which directly affect the way resources are allocated and furthermore influences Hadoop performance.

To overcome the problem of delay in completion time of job and lower rate of resource utilization [15] proposes a scheme for scheduling the slots for map-reduce tasks to minimize I/O wait during job implementation and improve resource utilizations in order to strengthen overall performance.

[16] proposes a structure for evaluating the performance to ease the user efforts in MapReduce for fine-tune the settings of reduce task (shuffle, reduce and write) and map task (read, map, collect, spill, merge) with help of performance models: workflow model along with platform model to optimize the performance.

For the efficient use of available resources on the basis of load of each node, [17] proposes a method for which can take decision about number of tasks to be execute.

In order to significantly lower the cost of system [19] proposed a function for MapReduce using clustering algorithm for mean shift to execute the jobs in better way with optimum values for parameters along with analysing the data sets for minimizing energy usage, increased system performance and complexity management.

C. Parameters associated commonly with HDFS and MapReduce

In case of processing batch tasks Hadoop's default configuration results in low utilization of resources which in turn delays in execution time [13] . Furthermore [13] proposes a dynamic effective slot configuration to allocate appropriate tasks to slots while processing batch of map/reduce jobs to provide enhanced fairness and make span.

One of the major issues in MapReduce framework is optimum utilization of resources as it requires configuring various parameters with impeccable balance which is time consuming and challenging practice. [20] performs an analysis to explore various parameters of Hadoop under varying configurations and settings to attain better throughput with an emphasis on execution time and throughput for scheduling jobs. By conducting experiments compare default scheduling methods and to study the behaviour of configuration of parameters [20] recommends optimum value for individual cases.

To overcome the time consuming process in Hadoop to configure the parameters of MapReduce jobs having non-linear and multi-dimensional structures [23] propose predators for 23 parameters as a capable directed optimizer for configuration by utilizing execution time of job and categorizing the parameters with aim of reducing search time by controlling the rate of visiting un-favourable blocks.

Hadoop provides enormous distinct configuration properties which affect its performance and keeping this in view [24] discuss few methods used for tuning hardware and software components on TeraSort dataset on two different clusters with different configurations which shows an increase in processing up to 4.2x on one cluster and 2.1x on another cluster.

[26] presents a detailed study on energy efficacy in MapReduce for different loads which results in pinpointing the factors: replication factor of block size along with distributed file system, CPU intensive and I/O intensive to conclude that a noble tuning of parameters results in enhanced performance along with better utilization of resources for energy saving.

[27] assimilates on going practices in semantic search and machine learning on the basis of ontologies to propose a new approach with an aim to enhance performance of applications in Hadoop to tune the parameters by categorizing them according to influence on system performance, Hadoop phases and workloads characteristics.

III. TAXONOMY OF TUNING HADOOP PARAMETERS

When Hadoop is installed it provides number of configurations for setting up the parameters having default values in xml file. The parameters may be of cluster level or job level. Furthermore based on influence behaviour the parameters can be classified as: Map, Reduce and intermediary phases where intermediary phases consist of shuffling and merging. The default values of parameters in Hadoop are further configurable and can be customised through Coding, updating xml files and passing values at execution time [21] . The parameters in XML files: conf/hdfs-site, core-site, and mapred-site in Hadoop can be customised by user if they are not protected using keyword *final.* With help of methods hadoop –D and hadoop -conf the default configuration value of parameter can be changed at run time using:

hadoop jar examples.jar example_name –D name_of_property(key)= new_value

Hadoop offers users to configure value of parameter using Configuration Class through coding. To create an object of class the syntax is:

ReflectionUtils.newInstance(Class<T> theClass, Configuration conf)

Parameters configuration in Hadoop can be classified on the basis of workload characteristics like I/O, CPU, memory, network and number of mappers as depicted in Table-I, Table-II and Table-III.

Table I: Memory associated parameters

Phase	Parameter	Initial value	Function
Sort/ Shuffle	mapreduce.task .io.sort.factor	10	Choose number of streams to be merged at one time while sorting the files and determines handling the number of open file.
	mapreduce.task .io.sort.mb	512	Decides on the size of memory requisite at time of sort.
Map	mapreduce.map .memory.mb	1536	Decides on how much memory to limit for map task.
	mapreduce.map .java.opts	Xmx1024M	Decide on size of heap memory for maps child java virtual machines.
Reduce	mapreduce.redu ce.memory.mb	3072	Choose amount of memory for reduce task.

	mapreduce.reduce.java. opts	Xmx2560M	Decide on size of heap memory for reduce task child java virtual machines.

Table II: I/O associated parameters

Phase	Parameter	Initial value	Function
Cluster level/ Merge/Shuffle	dfs.blocksize	128 MB 134217728 bytes	Responsible for choosing size of block for a file.
	dfs.replication	3	Decide about replication factor of a block.
	dfs.replication.interval	3	Decides period at which replication takes place in datanodes
	dfs.data.dir	${hadoop.tmp.dir}/dfs/data	Decides where a data node can store the blocks on its local filesystem.
	fs.default.name	file:///	Universal Resource Identifier that decides the FileSystem execution structure as well as authority.
	dfs.default.name	--	It holds NameNodes location. It is requisite of HDFS and MapReduce.
	io.sort.record.percent	0.05	Agree on the fraction for io.sort.mb to acquire at time of sorting the file.
	io.sort.spill.percent	0.80	Choose the proportion of spill while sorting operation.
	io.sort.factor	10	Choose number of total streams to merge at one time during sort operation of files.
	io.file.buffer.size	4096	Decides on amount of data to buffer at time of read plus write processes.
	mapred.min.split.size	64MB	Require each map to process 2 hdfs blocks (1-block = 64MB)
	io.sort.mb	100	Decide on memory of buffer mandatory while performing file sorting.
Job Level/ Core Job	mapred.output.compression.type	RECORD	Choose type of compression for output.
	mapred.output.compression.codec	org.apache.hadoop.io.compress.DefaultCodec	Accountable to codec while compressing the job output.
Map	mapred.compress. map.output	False	Results in deciding the map output compressed or else?
	mapred.map.output.compression.codec	org.apache.hadoop.io.compress.DefaultCodec	Choose codec during compressing of job outputs for map phase.

Table III: CPU associated parameter

Phase	Parameter	Initial value	Function
Map	mapred.map.tasks	2	No. of mappers tasks per job
	mapred.TaskTracker.map.tasks .maximum	2	No. of mapper tasks for job to be executed by a task tracker simultaneously
	mapred.map.tasks.speculative.	True	No. of multi-instances of mappers for parallel execution.

	execution		
Reduce	mapred.reduce.tasks	1	No. of reducers tasks required per job
	mapred.TaskTracker.reduce.tasks.maximum	2	No. of reducer tasks for job to be executed by a task tracker simultaneously
	mapred.reduce.tasks.speculative.execution	True	No. of multi-instances of reducers for parallel execution.
Core Job	mapred.output.compress	False	Required output of job to be compressed or not?
	mapred.output.compression.type	BLOCK	Whether job outputs to be compressed as SequenceFiles? Must be NONE, RECORD or BLOCK.
	mapred.reduce.slowstart.completed.maps	0.0 0.5 1.0	Value 0.0 starts the reducers immediately, 0.5 start the reducers while about half of the mappers' tasks are done, and value of 1.00 wait until mappers finished the job.
	mapreduce.map.output.compress	False	Whether to compress map Outputs or not?

IV. FRAMEWORK

Proposed framework given below will enhance the overall performance of jobs in Hadoop on the basis of workload of job and modified parameter values in heterogeneous environment.

Algorithm:

1. Run Hadoop MapReduce job(s) along with default values of parameters to analyse the performance and store these results on basis of workload characteristics like I/O, CPU, memory and time taken.

2. Apply changes to modify default values and then again execute the job(s) to analyse the performance and store these results.

3. Compare results of both situations i.e. with default parameter values and with modified parameter values to analyse whether performance is tuned or not?

4. If results show improvement in performance then repeat step 2 till results are in better tuning than default values, else go to step 5.

5. Exit

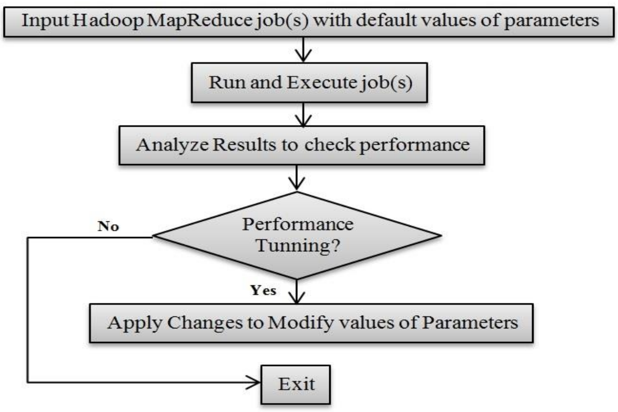


Fig. 2: Framework to modify values

Following set of modified values for parameters given in Table-IV are used for running different Hadoop jobs.

Table- IV: Modified values

Parameter	Default Value	Modified value of parameter		
		1st	*2nd*	*3rd*
dfs.blocksize	64	128	256	512
dfs.replication	3	5	7	9
io.sort.factor	10	20	30	50
io.sort.mb	100	120	150	170
Execution time of job	1000	550	450	355
Improvement over baseline (%)	N.A	30	55	64.5

V. EXPERIMENTS AND RESULTS

Using modified values experiments were carried on Hadoop 1.2.1 multi-cluster nodes using Ubuntu 12.04(LTS) with one master and five slave nodes. The master node is Intel(R) Core(TM) i7-2630 QM CPU @ 2.00 GHz, and 8 GB of RAM.

Table- V: Experimental setup configuration

Node	Processor	RAM
Master	Intel(R) Core(TM) i7-2630 QM CPU @ 2.00 GHz	8GB
Slave-1	Pentium(R) Dual-Core CPU E5800 @ 3.20 GHz	3GB
Slave-2	Intel(R) Pentium(R) Dual CPU E2160 @ 1.80 GHz	1GB
Slave-3	Pentium(R) Dual-Core CPU E5800 @ 3.20 GHz	2GB
Slave-4	Intel(R) Pentium(R) D CPU 2.80 GHZ	1GB
Slave-5	Pentium(R) Dual-Core CPU E5800 @ 3.20 GHz	2GB

Results of experiments to calculate execution time, Total CPU time along with CPU utilization by jobs TeraSort, WordCount and Pi are shown in Table-VI. CPU utilization by these jobs with default values is shown in Fig.3.
CPU utilization = (Total CPU time/Execution time) x 100

Table- VI: Performance on Single-cluster node

Parameter	TeraSort	WordCount	Pi
Execution time	2050	1170	14
Total CPU time	398.18	219.25	1.65
CPU utilization	19.42	18.73	11.78

Furthemore to anlyze the performance of suggested and modified values of parameters (Table-IV) for different jobs on multi-cluster heterogeneous environment experimental results are shown in Table-VII.

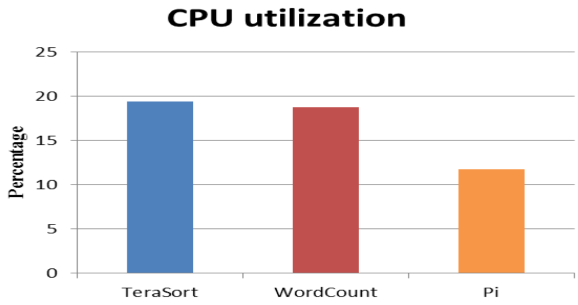
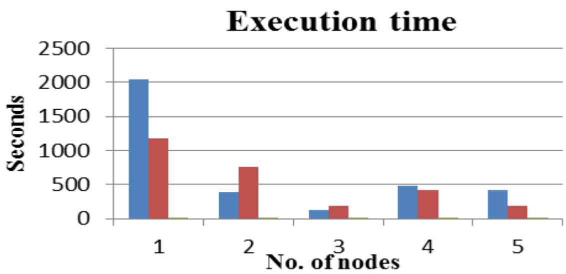
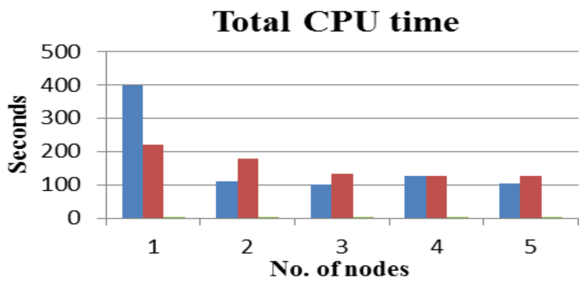


Fig. 3: Evaluation of CPU utilization single-cluster node

Table-VII: Performance on Multi-cluster nodes

Parameter	Job	No. of Nodes				
		1	2	3	4	5
Execution time	TeraSort	2050	388	131	473	426
	WordCount	1170	762	190	426	182.4
	Pi	14	12	19	17	23
Total CPU time	TeraSort	398.18	111.50	100.35	125.45	102.86
	WordCount	219.25	177.29	132.75	27.44	126.9
	Pi	1.65	1.32	1.45	1.06	1.35
CPU utilization	TeraSort	19.42	28.73	76.60	26.52	65.93
	WordCount	18.73	23.27	69.87	29.91	69.57
	Pi	11.78	11	7.63	6.23	5.87





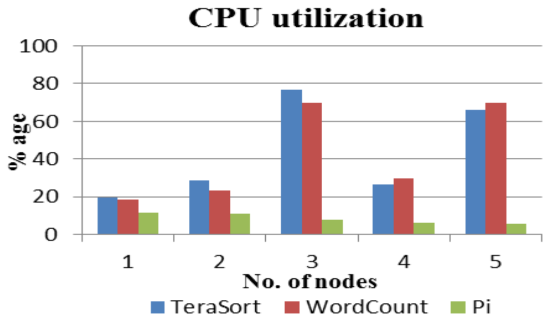


Fig. 3: Execution time, CPU time and CPU utilization of muti-node cluster

CONCLUSION AND FUTURE WORK

The default parameter configuration of Hadoop is not appropriate for all type of clusters especially heterogeneous. Fine tuning Hadoop Parameters in right manner can enhance data locality and amount of data processed to improve the performance of resources. In future, further to improve data locality and amount of data processed there is need to design a frame that offers better arrangements of parameter configuration setting via executing different Hadoop jobs with varying parameter sets. There is need to design a novel scheduling framework to offer enhanced data locality as well as enriched amount of data processing to improve overall performance of Hadoop in Heterogeneous multi node cluster. Proposed framework along with better combination of values for different parameters can enhance the performance of Hadoop in heterogeneous environment.

REFERENCES

[1] S. Kurazumi, T. Tsumura, S. Saito, and H. Matsuo, "Dynamic processing slots scheduling for I / O intensive jobs of Hadoop MapReduce," in *2012 Third International Conference on Networking and Computing*, 2012, pp. 288–292.

[2] C. A. Hansen, "Optimizing Hadoop for the cluster," 2012.

[3] T. White, *Hadoop: The Definitive Guide*, 4th ed., vol. 4. O'Reilly Media, 2015.

[4] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010*, 2010, pp. 1–10.

[5] M. Vaidya, "Parallel Processing of cluster by Map Reduce," *International Journal of Distributed and Parallel Systems*, vol. 3, no. 1, pp. 167–179, 2012.

[6] X. Zhao, L. Liu, Q. Zhang, and X. Dong, "Improving MapReduce Performance in a Heterogeneous Cloud : A Measurement Study," in *2014 IEEE 7th International Conference on Cloud Computing*, 2016, no. June 2014, p. 8.

[7] V. Kalavri and V. Vlassov, "MapReduce: Limitations, optimizations and open issues," in *Proceedings - 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2013*, 2013, pp. 1031–1038.

[8] A. Sharma and G. Singh, "A Review on Data locality in Hadoop MapReduce," in *5th IEEE International Conference on Parallel, Distributed and Grid Computing(PDGC-2018)*, 2018, pp. 723–728.

[9] T. L. S. R. Krishna, D. T. Ragunathan, and S. K. Battula, "Performance Evaluation of Read and Write Operations in Hadoop Distributed File System," in *2014 Sixth International Symposium on Parallel Architectures, Algorithms and Programming*, 2014, pp. 110–113.

[10] A. Pal, P. Agrawal, K. Jain, and S. Agrawal, "A Performance Analysis of MapReduce Task with Large Number of Files Dataset in Big Data Using Hadoop," in *2014 Fourth International Conference on Communication Systems and Network Technologies*, 2014, pp. 587–591.

[11] J. Xie *et al.*, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters," in *2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW)*, 2010, p. 9.

[12] B. T. Rao, N. V Sridevi, V. K. Reddy, and L. S. S. Reddy, "Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing,"

Global Journal of Computer Science and Technology, vol. 11, no. 81–87, 2011.

[13] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi, "FRESH : Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters," in *2014 IEEE 7th International Conference on Cloud Computing*, 2014, no. June, p. 9.

[14] K. Kc and V. W. Freeh, "Tuning Hadoop map slot value using CPU and IO metrics," in *Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 4th and 5th Workshops, BPOE 2014*, J. Zhan, R. Han, and Chuliang Weng, Eds. Springer, 2013, pp. 141–153.

[15] Y. Yao, J. Wang, B. Sheng, and N. Mi, "Using a Tunable Knob for Reducing Makespan of MapReduce Jobs in a Hadoop Cluster," in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, p. 8.

[16] Z. Zhang, L. Cherkasova, and B. T. Loo, "Getting More for Less in Optimized MapReduce Workflows," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, 2014, no. May, p. 8.

[17] K. Yamazaki, R. Kawashima, S. Saito, and H. Matsuo, "Implementation and Evaluation of The JobTracker Initiative Task Scheduling on Hadoop," in *2013 First International Symposium on Computing and Networking*, 2013, pp. 622–626.

[18] Y. Chen, A. Ganapathi, and R. H. Katz, "To Compress or Not To Compress Compute vs . IO tradeoffs for MapReduce Energy Efficiency," in *Green Networking '10 Proceedings of the first ACM SIGCOMM workshop on Green networking*, 2010, pp. 23–28.

[19] G. Sasiniveda and R. N, "Performance Tuning and Scheduling of Large Data Set Analysis in Map Reduce Paradigm by Optimal Configuration using Hadoop," *International Journal of Computer Applications*, vol. 70, no. 21, pp. 37–41, 2013.

[20] G. Bansal, A. Gupta, U. Pyne, M. Singhal, and S. Banerjee, "A Framework for Performance Analysis and Tuning in Hadoop Based Clusters," in *Smarter Planet and Big Data Analytics Workshop (SPBDA 2014), held in conjunction with International Conference on Distributed Computing and Networking (ICDCN 2014), Coimbatore, INDIA*, 2014, p. 6.

[21] C. Li *et al.*, "An Adaptive Auto-Configuration Tool for Hadoop," in *2014 19th International Conference on Engineering of Complex Computer Systems*, 2014, pp. 69–72.

[22] D. Wu and A. Gokhale, "A Self-Tuning System based on Application Profiling and Performance Analysis for Optimizing Hadoop MapReduce Cluster Configuration," in *20th Annual International Conference on High Performance Computing*, 2013, pp. 89–98.

[23] K. Wang, X. Lin, and W. Tang, "Predator - An Experience Guided Configuration Optimizer for Hadoop MapReduce," in *2012 IEEE 4th International Conference on Cloud Computing Technology and Science Predator*, 2012, pp. 419–426.

[24] S. B. Joshi, "Apache Hadoop Performance-Tuning Methodologies and Best Practices," in *ICPE '12 Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, 2012, pp. 241–242.

[25] J. Yan, X. Yang, R. Gu, C. Yuan, and Y. Huang, "Performance Optimization for Short MapReduce Job Execution in Hadoop," in *2012 Second International Conference on Cloud and Green Computing*, 2012, pp. 688–694.

[26] B. Feng, J. Lu, Y. Zhou, and N. Yang, "Energy Efficiency for MapReduce Workloads : An In-depth Study," in *ADC '12 Proceedings of the Twenty-Third Australasian Database Conference - Volume 124*, 2015, no. April, pp. 61–70.

[27] A. Bonifacio, A. Menolli, and F. Silva, "Towards an Ontology-Based Semantic Approach to Tuning Parameters to Improve Hadoop Application Performance," *IT in Industry*, vol. 2, no. 2, pp. 56–61, 2014.

[28] H. Yang, Z. Luan, W. Li, D. Qian, and G. Guan, "Statistics-based Workload Modeling for MapReduce," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, 2012, pp. 2043–2051.

[29] S. Babu, "Towards Automatic Optimization of MapReduce Programs," in *SoCC '10 Proceedings of the 1st ACM symposium on Cloud computing*, 2010, pp. 137–142.

AUTHORS PROFILE

Gurwinder Singh MCA, M.Phil, M.Tech(IT) working as Assistant Professor in Sikh National College Banga and doing research in School of Computer Applications, Lovely Professional University Punjab, India. He is having more than 13-years of teaching experience. He is a member of CSTA.

Publications:

1. "A Review of Scheduling Algorithms in Hadoop" published in *Proceedings of ICRIC 2019 by Springer*, Lecture Notes in Electrical Engineering-597.
2. "A Review on Data locality in Hadoop MapReduce" published in IEEExplore.
3. " A Review of Big Data Challenges and Preserving Privacy in Big Data" published in proceedings of 2nd International Conference on Data and Information Sciences (ICDIS 2019) by Springer "Advances in Data and Information Sciences".
4. "A Study on Big Data Tools and Applications" published in IJRECE VOL. 6 ISSUE 2 APR.-JUNE 2018 ISSN: 2393-9028 (PRINT) | ISSN: 2348-2281 (ONLINE)
5. "Big Data Applications" published in International Journal of Advance Research in Science and Engineering (IJARSE) Vol. Issue 10 October 2017.

Dr. Anil Sharma MCA, Ph.D. working as Professor in School of Computer Applications, Lovely Professional University Punjab, India. He received his doctorate from HPU Shimla. He has over 18 years of Academic experience. He has published more than 10 research papers in reputed journals. His area of interest includes Databases, Cloud Computing, and IOT.