*I.J. Modern Education and Computer Science*, **2015, 12, 51-56**
Published Online December 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.12.07

# Performance Analysis of Schedulers to Handle Multi Jobs in Hadoop Cluster

**Guru Prasad M S**
SDMIT/CSE, Ujire, 574240, India
Email: guru0927@gmail.com

**Nagesh H R and Swathi Prabhu**
MITE/CSE, Moodabidri, 574227, India
SMVITM/CSE.Udupi, 576115, India
Email: nageshhrcs@rediffmail.com, prabhuswathi2@gmail.com

***Abstract*—**MapReduce is programming model to process the large set of data. Apache Hadoop an implementation of MapReduce has been developed to process the Big Data. Hadoop Cluster sharing introduces few challenges such as scheduling the jobs, processing data locality, efficient resource usage, fair usage of resources, fault tolerance. Accordingly, we focused on a job scheduling system in Hadoop in order to achieve efficiency. Schedulers are responsible for doing task assignment. When a user submits a job, it will move to a job queue. From the job queue, the job will be divided into tasks and distributed to different nodes. By the proper assignment of tasks, job completion time will reduce. This can ensure better performance of the jobs. By default, Hadoop uses the FIFO scheduler. In our experiment, we are discussing and comparing FIFO scheduler with Fair scheduler and Capacity scheduler job execution time.

***Index Terms*—**BigData, Apache Hadoop, MapReduce Framework, Hadoop Schedulers, Job Execution Time, Ganglia tool.

## I. Introduction

As the Internet usage keeps increasing, Data generated during the day to day life is more than terabytes. This needs to be processed in many Internet Service Providers. MapReduce [1] framework now has the solution for this, which is used for large-scale data processing, i.e. thousands of nodes by using commodity hardware. Hadoop, an popular open source framework implementation of MapReduce model and maintained by Apache Software Foundation. Hadoop [2], is already used for processing hundred terabytes of data on at least 10,000 cores. In this environment, people may share the same cluster for many purposes, so that the cluster needs to run for different kinds of workloads (heterogeneous workload) on the same data center. Here we can see the problem of job scheduling in the cluster.

The default scheduler in Apache Hadoop is single queue, schedules the jobs in FIFO order which called as FIFO scheduler. According to this scheduler tasks executes based on the arrival time. Some of the schedulers like capacity scheduler [4] which is multi user scheduler as well as a fair scheduler [3] which uses multiple queues and utilizes different resources in the cluster. Using these schedulers, we could assign jobs to queues which manually guarantee their specific resource share among the homogeneous and heterogeneous workload.

The performance of Hadoop framework mainly depends on the cluster size, the hardware configuration in each node and the scheduling methods for the jobs. The scheduling algorithms in distributed systems usually have the goals of spreading the load on processors and maximizing their utilization of resources which minimizes the total task execution time internally it effects on job execution time.

In our work, we concentrated on the problem that, how can we improve the hardware utilization rate when different kinds of workloads (i.e. homogeneous and heterogeneous data) run on the clusters in MapReduce framework. In practical, different type of jobs often simultaneously run in the Cluster, the scheduler is having the main role in assigning the jobs and is closely tied to the performance of the MapReduce framework. Mainly two kinds of scheduling policy one is Job level i.e. scheduling jobs submitted into Hadoop cluster and another one is task level i.e. scheduling the tasks from a specific job. In this paper, we focus on different Hadoop schedulers such as FIFO, Fair, Capacity scheduler and their job execution time analysis. Our main aim is to compare the job execution time of above mentioned schedulers with default Hadoop scheduler's job execution time (FIFO). We considered a job execution time as performance analysis parameter for our entire journey we used Ganglia software [17] which provides execution environment and complete real-time monitoring, which gives the clear idea about the resource usage. Ganglia were developed at the California University in Berkeley Computer Science Division to link clusters across logical way in the Berkeley campus. It is completely open-source because it was developed at a university and no proprietary components are needed..

The rest of the paper is organized as follows. We

describe the background of Hadoop and related work of this paper in Section 2. Section 3 gives experiment results and an analysis of job execution time with different schedulers and in Section 4 final conclusion, Future work.

## II. Background & Related Work

Over the last few years, data stored in the world has increased exponentially. Data sources are everywhere, from user-generated content to large scientific experiments and Web 2.0, from social networks to wireless sensor networks. This large amount of data is a valuable asset in our information society this called the BigData. The problem here is to process the data in order to extract the useful information. As we mentioned it is petabyte of data, by using traditional methods for data analysis is not possible. Because the data analysis tools are unable to keep up with the increase in size, diversity and rate of change of data size. For this a new tools and approaches are required, currently the most used tool is definitely Hadoop.

In this section, we briefly discuss about Hadoop architecture and related scheduling mechanisms. Hadoop framework running on main two components, Hadoop Distributed File System (HDFS) and MapReduce framework, both are Master-Slave architecture. One master node and number of slave nodes. Hadoop Distributed File System (HDFS) stores and manages the file which is of fixed size data blocks (64MB by default). The master node called Namenode will assigns the data blocks to slave nodes called Datanodes for processing. Namenode contains metadata i.e. information about the each data blocks which is in datanodes and meta data of meta is saved in Secondary Namenode for every regular interval of time.

Each jobs assigned to Hadoop framework divides it into number of subtasks i.e. map tasks and reduce tasks, which will be handled with a Map function and a Reduce function in the MapReduce framework. Each Map task deals with a small part of the input data. After Map task processing, it will generates the results those are intermediate state key-value pairs, this output will be the input for Reduce function. Reduce function will merge based on a specific key, then generate and output as value-keys. Every few seconds master node receives heartbeats from slave nodes.

MapReduce framework, running on HDFS, is used for data processing in parallel and in distributed pattern. It consists of a JobTracker which runs on the master node and several TaskTracker runs on slave nodes. Namenode and JobTracker, the core of Hadoop, are running on the master node, and a TaskTracker is running on a slave node together with one Datanode. Thus one master node and multiple nodes constitute a distributed storing and processing clusters for large-scale data. JobTracker is in charge of scheduling and monitoring all the tasks, including map tasks and reduce tasks, of a MapReduce job. Tasks are distributed to TaskTrackers who execute the map function and reduce function defined by user's MapReduce application which uses the data saved in Datanodes. Fig.1 shows overall architecture of Hadoop.

According to cluster resource utilization, current Hadoop schedulers classified into two types: (a) full utilization of resources to maximize the use of it. (b) Partial utilization of resources to do concurrent processing. Hadoop's default scheduler works based on first type. In below section we are going discuss about Hadoop schedulers

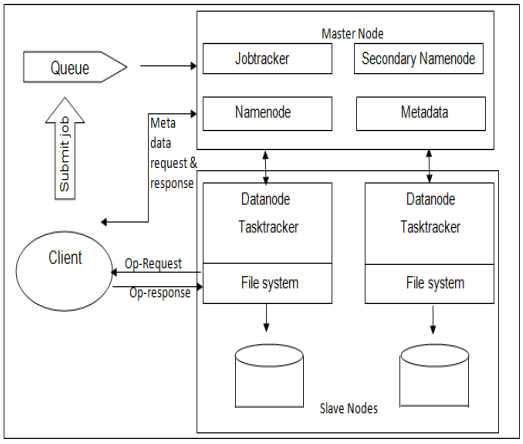



Fig.1. Hadoop Architecture.

### A. Default Scheduler (FIFO)

The default Hadoop scheduler works using a FIFO queue. Jobs are submitted to cluster, divide them into tasks, and then tasks are moved into the queue and processed by available slots in the nodes. Although there is a support for jobs priorities, but it is not turned on by default. In FIFO scheduler each job would use the entire cluster, so jobs which are submitted need to wait for their turn. Here the main problem is sharing resources fairly among users and all so among jobs, to solve this problem we need a better scheduler. Once a task slots become free, then first waiting job in the queue will be assigned for execution. From this scheduler we can say that one job will take all task slots within the cluster. Also, jobs those arrived at a later stage or with a less priority will be blocked behind the higher priority jobs. Recently few alternative schedulers have been developed such as Yahoo!'s Capacity scheduler and Facebook's Fair scheduler.

### B. Fair Scheduler

The fair scheduler [3] in Hadoop was developed by Facebook. The main intention of this scheduler is to assign the resource on and average to all jobs and all users so that each job gets equal share of available resources. Therefore the jobs waiting time will reduce and CPU will be used totally among all jobs. In Fair Scheduler actually organizes jobs by resource pool with each pool holds minimum number of map and reduce slots. By default, each user there is a separate pool. Free slots in idle pools can be taken to other pools and resources are fairly shared among all jobs but by default slots are fixed. The Fair Scheduler can pre-empt the jobs

if pool has not received its fair share. Fair scheduler will kill tasks in pools running to lower the resource consumption. Also we can set the job priority as like FIFO schedule. From this scheduler we can give guarantee that longer jobs are not be starved of resources.

### C. Capacity Scheduler

The Capacity Scheduler [4] was originally developed by Yahoo which is defined for large clusters or different organization. In capacity scheduling queues are created which are used for multiple users. For each queue we can allocate number of map and reduce slots. During execution queues are monitored, if entire capacity of one queue is not used its excess capacity used for another queue temporarily. As like FIFO and Fair scheduler Capacity scheduling has the ability to prioritize the jobs within a queue. There is strict access control on queues (queues are tied to a organization or person).Both capacity scheduler & fair scheduler have similarities:

- Good for multi users because support multiple queues or pools.
- Each queue or pool supports for FIFO or different job priorities.
- In both schedulers, they can share idle slots to other queues or pools.

And following are its differences:

- Different strategies
- Different memory constraints

Since Hadoop cluster is connected by networks, data transportation and then execution of jobs is major issues. Some researches focused on optimizing MapReduce processes automatically and managing resources allocation with different job. Our focus is on job schedulers which will helps to schedule different kind of workloads. We considered few authors survey results that has mentioned about schedulers, then we implemented it and analyzed the results by using Weblog data and Amazon data. Weblog data is semi structured data and Amazon data is record format data

## III. Results and Discussion

### A. Experimental Environment

For the performance evaluation, we considered Hadoop three nodes cluster with homogeneous hardware property, i.e. Each node in the cluster has a 3.8 GB RAM, Intel® Core i5 3470 CPU @3.20GHz * 4 processor. We setup cluster on Ubuntu 14.04 with Hadoop 1.2.1 stable release used oracle jdk 1.7 and ssh configuration to manage Hadoop daemons. Our cluster setup is having 1 NameNode and 3 DataNodes for the purpose of an experiment. Configuration files such as mapred-site.xml, core-site.xml, hdfs-site.xml are setup by default values with replication factor 2 and default block size 64MB. We used Web Log data of 2.1 GB [5] and Amazon data [6] of 990 MB for our analysis, collected this data from different sites. We performed a set of experiments to evaluate the performance .by using different schedulers on heterogeneous workload and homogeneous environment.

In this paper, we are comparing default FIFO scheduler's execution time with Fair and Capacity scheduler's execution time. Execution time includes submission to completion of the particular job. For the analysis, we are using Amazon data which is of a record format, our application invokes mapper for each record which is passed as input for our application. The mapper extracts the customer ID and generates values for each customer ID this output will be given to reducer, where reducer will take the customer ID and value, then sum up and gives the output as the number of items the customer has ordered and Web Log data which is semi-structured data. We had written application which shows the output of hits by hour of the day. In application Column 1 is the hours (24 hour format), column2 is the number of page access for each hour. From using these two applications with the homogeneous and heterogeneous workload, we started our experiment. Initially we analyzed this data in our application; we can check this result in Fig.2 and Fig. 3 respectively.

After this analysis of Bigdata we started work on different Hadoop schedulers. Initially we executed these application in default scheduler, then started to compare the results with different schedulers for same applications. Below section we are discussing the results from our analysis.

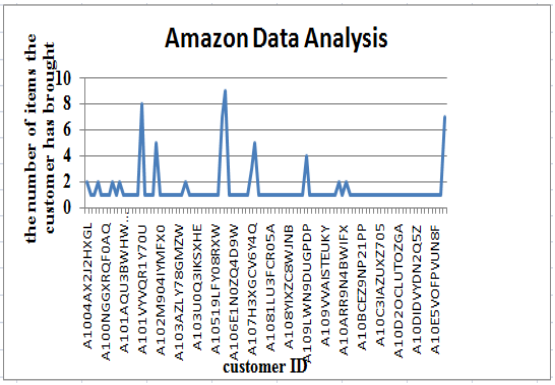



Fig.2. Amazon Data Analysis

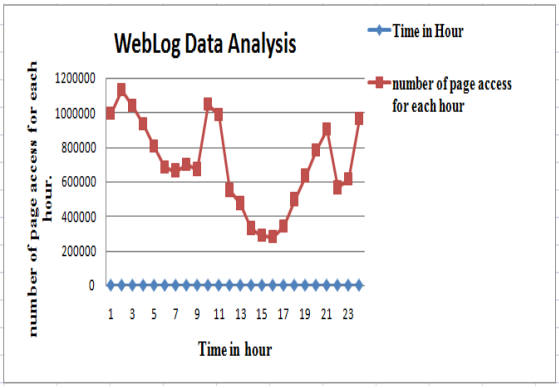



Fig.3. Weblog Data Analysis

In the FIFO scheduler (default scheduler), while executing the first job, second and third jobs need to wait. From our experiment we can tell that total execution time increases as the number of jobs increases, it effects on overall performance of jobs in system Fig.4 shows the execution time of three jobs using different inputs of Web Log data and Amazon data. With homogeneous workload (Data size same) and heterogeneous workload (different data size) From this we can say that as the number of jobs increases job waiting time also increases and small jobs will be struck behind the large jobs in case of heterogeneous workload so the performance of overall job execution time decreases.

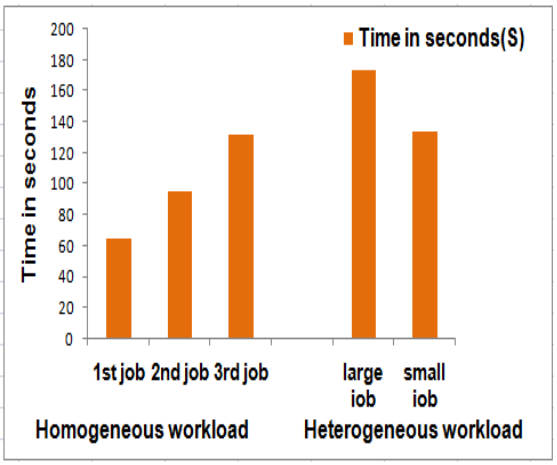



Fig.4. FIFO Job Execution Analysis

We considered Fair scheduler, as we already discussed above, main intention of this scheduler is to fair resource usage, i.e. all jobs should have Fair share of resources either it may be large or small and if we assign the pools to different users then each pool should share the resource fairly among the jobs. The results from our analysis are that Fair scheduler takes less job execution time than the FIFO which showed in Fig.5.In Fig.6, we are giving homogeneous and heterogeneous workload to different pools in Fair scheduler. In this figures we can see the job execution time comparison. Here we considered two pools for our experiment

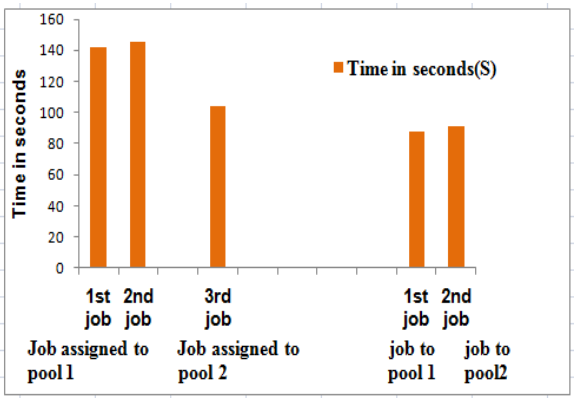



Fig.5. Fair Scheduler Job Execution Time Comparison in Homogeneous Workload with Different Pools

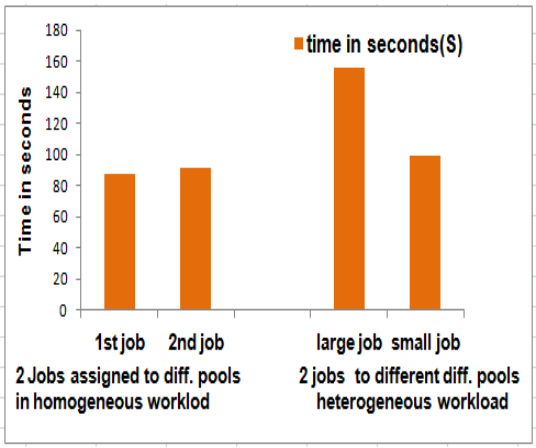



Fig.6. Fair Scheduler Job Execution Time Comparison in Heterogeneous Workload

After the fair scheduler, we considered one more Hadoop supported scheduler is Capacity scheduler. Here we created queues to assign the jobs. Depends on the hardware resources we can increase the number of queues. As the number of queue increase job execution time also increases. We considered 2 queues assigned the jobs to them parallel and the 3 queues and assigned the jobs. We can see the results in Fig.7.

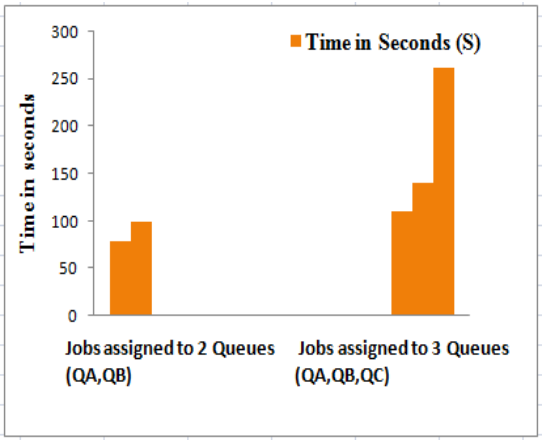



Fig.7. Capacity Scheduler Job Execution Time by Increasing Number of Queues

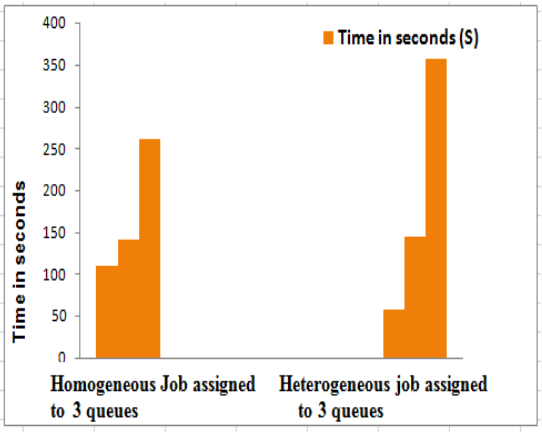



Fig.8. Capacity Scheduler Job Execution Time for Homogeneous and Heterogeneous Workloads

Then we compared capacity scheduler job execution time in homogeneous and heterogeneous environment. Fig. 8 shows the comparison of it.

In this scheduler there will be many parameters those can be tuned and get better job execution time such as QueueCapacity or MaximumCapacity etc. Compare to Fair scheduler capacity scheduler will take more job execution time. From our experiment we can tell that Fair is better among other schedulers

We used Ganglia tool to analyze the resource usage. of entire cluster. It will clearly show the resource usage by the cluster We can observe it in Fig.9. Graphs in the figure shows last one hour load of Hadoop cluster, memory usage, CPU usage and network usage respectively From this tool we can analyze the resource usage of entire cluster very easily.

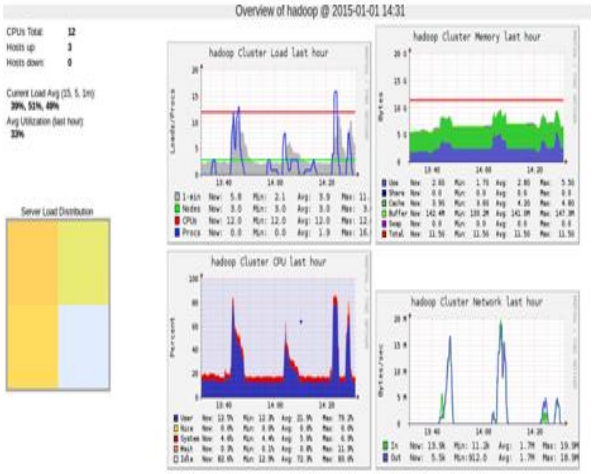



Fig.9. Resource usage by the Cluster

## IV. Conclusion and Future Work

We considered homogeneous and heterogeneous workloads for our experiment and total job execution time as the main parameter for analysis which includes job execution time and waiting time of jobs. From our analysis, we can tell that FIFO scheduler is takes more execution time as the number of jobs increases but for small jobs it is good scheduler. The job execution time is less for Fair scheduler in multi jobs with heterogeneous workload because resources are fairly shared among all jobs. And Capacity schedulers also take more execution time compare to Fair scheduler. In future work, we planned to consider the capacity scheduler parameter to analyze the job execution time in the heterogeneous environment and to implement our own scheduler to solve data locality problem.

## Acknowledgment

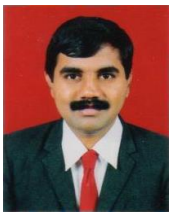
We would like to thank every member of the faculty at SDMIT, MITE and SMVITM for their guidance and support, which has helped us, complete this research project successfully.

## References

[1] V. Kalavri, V. Vlassov, “MapReduce: Limitations, Optimizations and Open Issues”, 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, DOI10.1109/TrustCom.2013.126, pp.1031-1038.
[2] Apache Hadoop. http://hadoop.apache.org.(2012, Aug).
[3] Fair scheduler [online] http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html.
[4] Capacity scheduler http://hadoop.apache.org/core/docs/current/capacity_scheduler.html.
[5] NASA weblog dataset http://ita.ee.lbl.gov/html/contrib/NASAHTTP.html.
[6] Amazon data http://snap.stanford.edu/data/#amazon.
[7] Tom White. 2010. “Hadoop: The Definitive Guide” (Second edition).O'Reilly Media/Yahoo Press.
[8] V. Kalavri, V. Vlassov, “MapReduce: Limitations, Optimizations and Open Issues”, 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, 2013, DOI10.1109/TrustCom.2013.126, pp.1031-1038.
[9] Aditya B. Patel, Manashvi Birla and Ushma Nair “Addressing Big Data Problem Using Hadoop and Map Reduce”, 2012 nirma university international conference on engineering, nuicone-2012, 06-08december, 2012.
[10] D. Wu, A. Gokhale,” A Self-Tuning System based on Application Profiling and Performance Analysis for Optimizing Hadoop MapReduce Cluster Configuration” IEEE conference,2013, pp. 89-98.
[11] Yao et al., 2013, “Scheduling Heterogeneous MapReduce Jobs for Efficiency Improvement in Enterprise Clusters”.
[12] Deshmukh et al., 2013, “Job Classification for MapReduce Scheduler in Heterogeneous Environment”, 2013 International Conference on Cloud & Ubiquitous Computing & Emerging Technologies.
[13] Chauhan et.al, 2012, “The Impact of Capacity Scheduler Configuration Setting on MapReduce Jobs”, 2012 Second International Conference on Cloud and Green Computing, 978-0-7695-4864-7/12 $26.00 © 2012 IEEE, pp 667-674.
[14] Zhao et al., 2012, “TDWS: a Job Scheduling Algorithm based on MapReduce”, 2012 IEEE Seventh International Conference on Networking, Architecture and Storage.
[15] Shi et al., 2011, “*S3*: An Efficient Shared Scan Scheduler on MapReduce Framework”, 2011 International Conference on Parallel Processing.
[16] Liu et al., 2013, “Evaluating Task Scheduling in Hadoop-based Cloud Systems” 2013 IEEE International Conference on Big Data.
[17] For Ganglia toolhttp://ganglia.sourceforge.net/.
[18] “Big data: the next frontier for innovation, competition, and productivity,” McKinsey Global Insititute, http://www.mckinsey.com/insights/mgi/research/technology and innovation/big data the next frontier for innovation, June 2011.
[19] Intel peer research: Big data analysis, intel’s it manager survey on how organizations are using the big data,” http://www.intel.com/ content/www/us/en/big-data/data-insights-peer-research-report.html, Auguest 2012.
[20] K. Shim, “Mapreduce algorithms for big data analysis,” in Proceedings of the VLDB Endowment. VLDB, pp. 2016–2017, Auguest 2012.
[21] J. Dittrich and J.-A. Quian´e-Ruiz, “Efficient big data processing in hadoop mapreduce,” Proceedings of the VLDB Endowment, vol. 5, no. 12, pp. 2014–2015,

Auguest 2012.
[22] J. Dean and S. Ghemawat, “Mapreduce: A flexible data processing tool,” *Comunications of the ACM*, vol. 53, no. 1, pp. 72–77, January 2010.
[23] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for, in-memory cluster computing,” in Proc. 9th USENIX Conf. Netw. Syst. Des. Implementation, 2012, p. 2.
[24] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S.-H. Bae, J. Qiu, and G. Fox, “Twister: A runtime for iterative mapreduce,” in Proc. 19th ACM Symp. High Performance Distributed Comput., 2010, pp. 810–818.

## Authors’ Profiles



**Dr. Nagesh H.R**, Dean (Academic), Professor & Head, Department of Computer Science & Engineering, Mangalore Institute of Technology & Engineering, Moodbidri, has got his M.Tech and Ph.D (Computer Engineering) from NITK Surathkal. He has published more than 50 research papers in National and International Conferences and journals. He has delivered more than 20 invited talks in topics like 'Component Based Software Development', 'Internet Security', 'Web Security', 'Web Engineering', 'Information Security' ,'Network Management', 'Promoting Global Cyber Security' ,'Security issues in Distributed Systems', 'Digital library and Information Search', 'Information Security Management' ,'Recent Trends in Information Technology' and 'Security issues in Cloud Computing'. He has also chaired many sessions in International and National level technical paper presentations. He has also published one chapter titled 'Proactive models for Mitigating Internet DoS/DDoS Attacks', in 'Selected Topics in Communication Networks and Distributed Systems', World Scientific, London, April 2010. He had also worked as Visiting faculty to NITK Surathkal and NITK-Science and Technology Entrepreneurs Park, Karnataka, Surathkal. Published two books titled 'Fundamentals of CMOS VLSI Design' for V semester Electronics & Communication Engineering students of VTU: Pearson Education & 'VLSI Design' for V semester Electronics & Communication Engineering students of JNTU: Pearson Education. Member of BOS for PG studies in Computer Science at Mangalore University and Manipal Institute of Technology for PG studies in Computer Science & Engineering. Worked as member of BOE and Exam coordinator in VTU Belgaum. Member of BOS in Computer Science & Engineering of VTU Belgaum for year 2013 to 2016.

**Mr. Guru Prasad M S,** Asst .professor, Dept of Computer Science & Engineering. Shri Dharmasthala Manjunatheshwara Institute of Technology, Ujire, Dakshinna Kannada. He got his M.Tech (Computer Engineering) from NMAMIT Nitte. He has published 5 research papers in International Conferences and Journals. He has delivered 7 invited talks on “Big Data Analytics”.

**Ms. Swathi Prabhu**, Asst. Professor , Dept. of Computer Science & Engineering, Shri Madhwa Vadiraja Institute of Technology & Management, Bantakal, Udupi. She got her M.Tech (Computer Engineering) from NMAMIT Nitte. She has published few research papers in National and International Conferences and journals. Her interested area is BigData Analysis using Hadoop.