Distributed Computing Engines for Big Data Analytics

Bh. Prashanthi,Griet, G. Sowjanya,,Griet, D. Krishna Madhuri,Griet

Abstract: ***Technologies like cloud computing paved way for dealing with massive amounts of data. Prior to cloud, it was not possible unless you invest large amounts for computing resources. Now there is ecosystem which is conducive to storing and processing voluminous data that cannot be handled by local computing resources. With such ecosystem, big data technology came into existence. Big data is the data characterized by volume, velocity, veracity and variety. This has enabled enterprises to give more value to every piece of data. This in turn led to the increased usage of cloud for both storage and processing. For processing big data efficient technologies are required. New programming paradigm like MapReduce with Hadoop distributed programming framework is widely used. However, there are other emerging frameworks like Apache Spark and Apache Flink to handle big data more efficiently. In this paper, empirical study is made on the three frameworks like Hadoop, Apache Spark and Apache Flink with different parameters like type of network, block size of HDFS, input data size and other configuration changes. The experimental results revealed that Apache Spark and Apache Flink outperform Hadoop. This is evaluated with different benchmark big data workloads.***

Index Terms –Big data, big data analytics, Hadoop, Apache Spark, Apache Flink, distributed programming frameworks

I. INTRODUCTION

Big data technologies emerged to improve performance of organizations. In other words, organizations can have competitive advantage with business intelligence garnered efficiently with big data analytics. MapReduce is the distributed programming framework that is widely used [1]. However, it has its limitations when compared with the newly emerged frameworks like Apache Flink and Apache Spark. It is understood that Spark has support for Resilient Distributed Dataset (RDD) which makes it perform well. It supports parallel operations, shared variables and abilities that are found in MapReduce counterparts [2]. There are many MapReduce frameworks like Hadoop found in the literature [3]. However, it is understood that with some benchmarks, Hadoop does not perform well. Thus it became essential and open research to evaluate different platforms that are used to analyse big data. It is also important in the context of streaming applications and big data analytics with streaming applications.When literature is reviewed on MapReduce and Spark frameworks, Spark was found to be better in certain benchmarks while MapReduce showed better performance in case of sort based workloads. The performance of Spark is attributed to its RDD caching [5]. Apache Spark, Apache Samza and Apache Storm are found to be useful for big data streaming applications. Performance and fault tolerance are considered for evaluation and found that Storm showed least latency over others while Spark has higher throughput than other frameworks [10]. From the works of [11], [12], [13], [14] and [15], it is understood that big data analytics applications need to consider different aspects like pre-processing, efficient computing resources usage, evaluation of the frameworks, synchronization and fault tolerance of the frameworks. Motivated by the existing work, we understood that it is still an open problem to make empirical study on distributed programming frameworks like Hadoop, Apache Spark and Apache Flink by considering different parameters. Our contributions are as follows.

1. We proposed a methodology for systematic evaluation of the three distributed programming frameworks like Hadoop, Apache Spark and Apache Flink.
2. We built a prototype application to evaluate proof of the concept.
3. We have made generalized conclusions on the evaluation of the three frameworks. The conclusions help in making well informed decisions.

The remainder of the paper is structured as follows. Section 2 provides review of literature on big data analytics and distributed programming paradigms. Section 3 presents the proposed methodology. Section 4 presents the experimental results. Section 5 provides conclusions and directions for future work.

II. 2. RELATED WORK

This section provides review of literature on big data analytics and different distributed computing frameworks. There is unprecedented need for big data applications in the real world. In the words of Veiga et al. [1] it is essential to evaluate big data frameworks for better decision making. They evaluated many MapReduce solutions on a cluster associated with High Performance Computing (HPC). They found that new frameworks like DataMPI outperformed Hadoop. Zaharia et al. [2] proposed Spark framework for handling massive amounts of data. They introduced Resilient Distributed Datasets (RDDs) with Spark for better performance. When Spark is compared with Hadoop, Spark showed impressive performance over its predecessor. Alexandrov et al. [3] on the other hand presented an open source big data platform known as Stratosphere. It showed better performance over Hadoop and Hive. Jakovits and Srirama [4] compared many MapReduce frameworks with scientific computing applications.

Revised Manuscript Received on July 20, 2019.
Bh. Prashanthi,Griet, Assistant Professor,Dept of CSE, GRIET,
G. Sowjanya,,Griet, Assistant Professor,Dept of CSE, GRIET,
D. Krishna Madhuri,Griet, Assistant Professor,Dept of CSE, GRIET,

They evaluated Spark, Hadoop, Haloop, Twister and MPI. They found that MPI and Twitster are two frameworks suitable for scientific applications. Shi et al. [5] studied the frameworks like Spark and MapReduce with large scale data. They found that Spark is much faster than MapReduce. Spangenberg et al. [6] analysed different big data frameworks in terms of new approaches. They compared Apache Flink and Apache Spark. They found that with relational query Apache Flink performed well while Apache Spark did well for WordCount benchmark. With K-Means and PageRank benchmarks Flink was better than that of Spark. Samosir et al. [7] evaluated frameworks that support data stream processing. The frameworks include Apache Spark, Samza and Storm. With empirical study they found that Storm is better candidate for processing big data streams. Chintapalli et al. [8] also focused on streaming computation frameworks like Flink, Spark and Storm. They found that Spark streaming provides higher throughputs while the Flink and Storm showed similar performance. Hafsa and Jemili [9] evaluated big data analysis techniques for the purpose of intrusion detection. They found that Apache Spark performed better with the task of intrusion detection. Qian et al. [10] evaluated differing streaming platforms such as Apache Storm, Apache Spark and Apache Samza. Fault tolerance and computational capabilities are observed. Storm was found to show least latency while the Spark was found to be better with fault tolerance.

Zaharia et al. [11] evaluated RDDs for in-memory computations. They observed that RDDs help in coarse grained transformations, restricted form of shared memory and support wide range of computations. Bal et al. [12] explored Distributed ASCII Super Computer (DAS) with different variations. They found such thing is suitable for processing large volumes of data. Veiga et al. [13] proposed an automatic tool for evaluation of MapReduce platforms. The tool provides computations with MapReduce with different networks like InfiBand and Ethernet. Siddique et al. [14] on the other hand performed empirical study on frameworks meant for bulk synchronous parallel computing such as Apache Hama and Apache Giraph. Both make use of HDFS. They found that Hama showed better performance over Giraph. Rehman et al. [15] proposed methodology for big data reduction in order to improve performance of big data processing and add value to enterprises.

Gardner et al. [16] proposed a framework for predictive modelling that is suitable for big data analytics. They found it to be generic framework to solve a class of computational problems. Venkataraman et al. [17] proposed different performance prediction models for big data analytics. The models are capable of advanced predictions for better use in the real world. Awan et al. [18] used Apache Spark for in-memory data analytics to characterise the same to be useful for big data analytics. They found that in-memory data analytics has some constraints with respect to limitations and performance. Yang et al. [19] studied Apache Spark for data caching optimization with respect to big data analytics. They found that its RDD feature is very useful in this regard. Different big data frameworks are explored in [20]. From the literature it is understood that there are many evaluations of big data frameworks. However, it is an open problem to have performance evaluation of big data frameworks with different parameters. This paper throws light into the evaluation of frameworks like Apache Spark, Apache Flink and Hadoop.

III. 3. PROPOSED METHODOLOGY

This section provides methodology for evaluation of the three frameworks. However, it does not provide the details of the three frameworks. Overview of those frameworks can be found in [1], [2], [5] and [10]. Hadoop supports MapReduce programming model. Spark on the other hand has other features like RDD. Apache Flink is the distributed programming framework supports stateful computations on data streams. The frameworks are configured according to the developer guides provided. The configuration details are described here. All the frameworks have access to HDFS. HDFS block size is set to 128 MB. Replication factor is set to 3 for all the frameworks. Heap size is set to 2.3 GB for Hadoop, 18.8 GB for Spark and Flink. Mappers per node is set to 4 for Hadoop, 1 for Spark and Flink. Reducers per node is set to 4 for Hadoop, 8 worker cores for Spark and 8 task manager cores for Flink. Shuffle parallel copies are set to 20 in Hadoop. IO sort spill percentage is set to 80% in Hadoop and Flink. IO sort MB is set to 600 in Hadoop while the same is configured for others. Number of buffers per node is set to 512 in Flink. Memory pre-allocation for task manager is set to false in Flink.

Different benchmark sources are used for evaluation. They are known as WordCount, Grep, TeraSort, Connected Components, PageRank and K-Means. The first two is CPU bound while the TeraSort is I/O bound. The remaining three are iterative in nature. Input data size considered for these benchmarks are 100 GB, 10 GB, 100 GB, 9 GB, 9 GB and 26 GB respectively. Different input generators like RandomTextWriter, TeraGen, DataGen and GenKMeansDataset are used for generating test data. The network interface is configured to support both IP over InfiniBand and GbE. WordCount benchmark is for big data processing. It counts occurrence of each word in a given document corpus. Grep on the other hand makes a count of matches of regular expressions in the input corpora. TeraSort, as the name implies, sorts key value pairs that are 100 byte-sized. Connected Components is an algorithm that is used to find connected components in a graph. PageRank is also an algorithm for different elements. K-Means on the other hand is used for clustering.

Performance evaluation metric used in this paper are execution time under different workloads and parameters like block size of HDFS, input data size and network interface used. Hadoop, Apache Flink and Apache Spark are evaluated using the aforementioned benchmarks under different parameter settings. This kind of empirical study reveals the performance difference of the frameworks. Section 4 presents experimental results for the three frameworks that led to conclusions provided. This research provides useful insights on the performance differences among the frameworks used for big data analytics.

IV. EXPERIMENTAL RESULTS

Experiments are made with three distributed computing frameworks that are widely used for big data analytics. Big data needs such frameworks to exploit thousands of commodity computers associated with cloud computing. The results show the execution time, impact of HDFS block size on the time taken, impact of input data size on the time taken and the effect of mappers and reducers configuration, task managers and cores configuration, and workers and cores configuration. Apache Spark, Apache Flink and Hadoop are used for empirical study with different benchmarks such as WordCount, Grep, TeraSort, Connected Components, PageRank and K-Means.

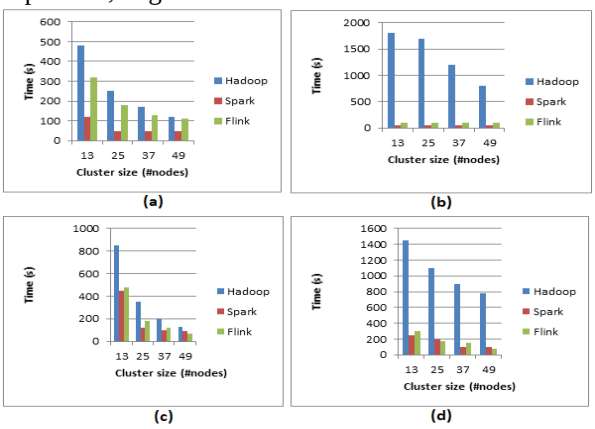

Figure 1: Execution time with WordCount benchmark (a), Grep benchmark (b), TeraSort benchmark (c) and Connected Components benchmark (d)

As shown in Figure 1, cluster size is considered for evaluating performance of the frameworks. It is observed that the cluster size (number of nodes in the cluster) has its impact on the performance in terms of time taken to execute a benchmark. In case of WordCount benchmark, Hadoop framework exhibited least performance. Highest performance is observed with Apache Spark. The performance of Flink is better than that of Hadoop. With respect to Grep benchmark, the trend in results is same as that of WordCount. In case of TeraSort Apache Spark showed highest performance. Flink on the other hand showed highest performance only when number of nodes in cluster is 49. As usual, Hadoop is the least performer. With Connected Components benchmark, Hadoop shows least performance while Spark shows highest performance. When the number of nodes is 25 and 49, Flink showed highest performance.

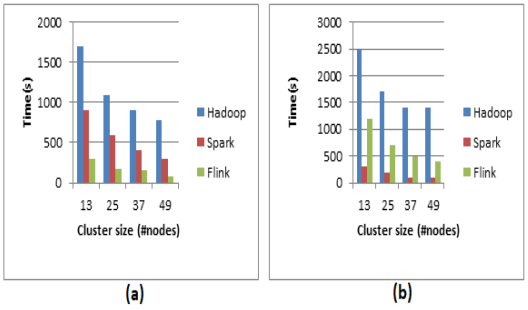

Figure 2: Execution time of PageRank (a) and K-Means algorithms

As presented in Figure 2, PageRank and K-Means benchmarks are used to evaluate the frameworks. Cluster size has its impact on the time taken by the frameworks to execute given benchmark. Apache Flink is the best performer when PageRank benchmark is used. In case of K-Means benchmark, highest performance is shown by Apache Spark.

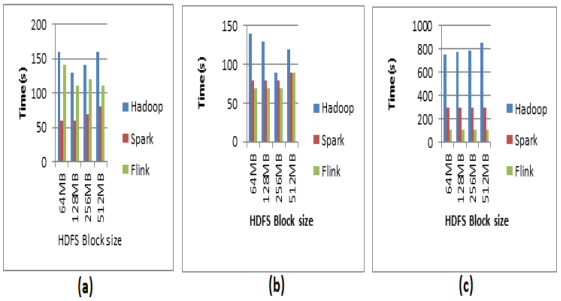

Figure 3: Impact of HDFS block size with WordCount (a), TeraSort (b) and PageRank (c)

Figure 3 shows performance comparison in terms of the execution time when different HDFS block size is used. There is impact of block size on the execution time. In all the three experiments (with benchmarks like WordCount, TeraSort and PageRank) Hadoop framework showed least performance. Apache Spark showed higher performance in case of WordCount benchmark. With TeraSort, interestingly, Apache Flink performed best. In case of PageRank benchmark, the Flink is the highest performer while the Apache Spark shows better performance over Hadoop.

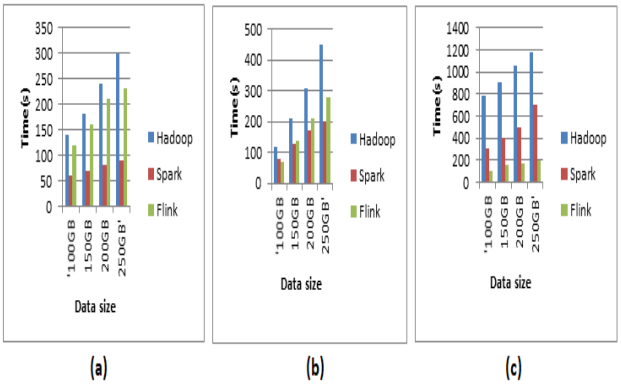

Figure 4: Impact of input data size with WordCount (a), TeraSort (b) and PageRank (c)

As shown in Figure 4, performance comparison in terms of the execution time when different input data size is used. There is impact of input data size on the execution time. In all the three experiments (with benchmarks like WordCount, TeraSort and PageRank), Hadoop framework showed least performance. Apache Spark showed higher performance in case of WordCount and TeraSort benchmarks. With TeraSort, interestingly, Apache Flink performed best in case of 100 GB data size.

In case of PageRank benchmark Flink outperforms other two frameworks.

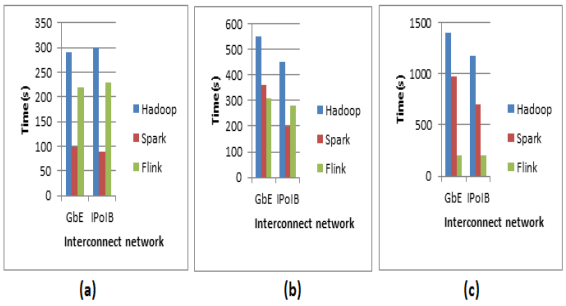

Figure 5: Impact of the network with WordCount (a), TeraSort (b) and PageRank (c)

As shown in Figure 5, performance comparison in terms of the execution time when different interconnect network is used. There is impact of interconnect network on the execution time. In all the three experiments (with benchmarks like WordCount, TeraSort and PageRank) with two interconnect networks, Hadoop framework showed least performance. Apache Spark showed higher performance in case of WordCount benchmark. With TeraSort, interestingly, Apache Flink performed best in case of GbE network and Spark showed best performance in case of IPolB network. In case of PageRank benchmark, the Flink is the highest performer while the Apache Spark shows better performance over Hadoop.

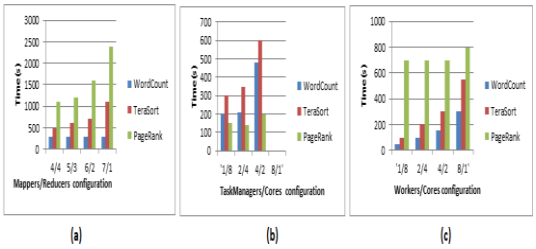

Figure 6: Performance with different thread configurations of Hadoop (a), Apache Spark (b) and Apache Flink (c)

As presented in Figure 6, it is understood that the experiments are made with different configurations. The configurations include mapper/reducer configuration, number of task managers/cores configuration and workers/cores configuration. The results revealed that there is impact of time taken when aforementioned configurations are adjusted. With mappers/reducers configuration there is impact of time taken in order to execute different benchmarks based on configuration. The results revealed that when number of reducers is decreased, the time taken is increased. In the same fashion, when number of cores is decreased, it took more time to execute benchmarks.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a methodology to evaluate different distributed computing frameworks that are used for big data analytics. Since big data is capable of adding value to enterprises that maintain it for discovering accurate business intelligence, this research assumes significance. When there are multiple frameworks available for big data analytics, it is essential to know the performance and scalability of the frameworks to make well informed decisions. Hadoop is the widely used framework that supports a new programming model known as MapReduce. It changed the way programming is made and computing resources are exploited in large scale. In fact, it paved way for parallel processing and dealing with storage and analysis of big data with its associated distributed file system known as HDFS. Afterwards many other distributed computing frameworks came into existence. Unless, they are understood in terms of usage and performance, it is not possible to make expert decisions. Keeping this in mind, we evaluated three frameworks for bit data analytics. They are known as Hadoop, Apache Spark and Apache Flink. Different parameters of these frameworks are considered for evaluation. In the same fashion, different benchmarks for big data analytics are used to evaluate the performance of the frameworks. The empirical study showed that Apache Spark and Apache Flink provide better performance over Hadoop. The performance difference is huge thus helping in making conclusions. In future we intend to continue our research on the architectural decision modelling pertaining to those frameworks for further evaluation.

REFERENCES

1. J. Veiga, R. R. Exp´osito, G. L. Taboada, and J. Touri˜no, “Analysis and evaluation of MapReduce solutions on an HPCcluster,” Computers & Electrical Engineering, vol. 50, pp.200–216, 2016
2. M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets,” in Proc. of the 2nd USENIX Conference on Hot topics in Cloud Computing (HotCloud’10), Boston, USA, 2010, pp. 1–7.
3. A. Alexandrov et al., “The Stratosphere platform for Big Data analytics,” The VLDB Journal, vol. 23, no. 6, pp. 939–964, 2014.
4. P. Jakovits and S. N. Srirama, “Evaluating MapReduce frameworks for iterative scientific computing applications,” in Proc. of the International Conference on High Performance Computing & Simulation (HPCS’14), Bologna, Italy, 2014, pp. 226–233.
5. J. Shi et al., “Clash of the titans: MapReduce vs. Spark for large scale data analytics,” in Proc. of the Very Large Data Bases (VLDB) Endowment, vol. 8, no. 13, 2015, pp. 2110–2121.
6. N. Spangenberg, M. Roth, and B. Franczyk, “Evaluating new approaches of Big Data analytics frameworks,” in Proc. Of the 18th International Conference on Business Information Systems (BIS’15), Pozna´n, Poland, 2015, pp. 28–37.
7. M. Bertoni, S. Ceri, A. Kaitoua, and P. Pinoli, “Evaluating cloud frameworks on genomic applications,” in Proc. of the 2015 IEEE International Conference on Big Data (IEEE BigData 2015), Santa Clara, USA, 2015, pp. 193–202.
8. S. Chintapalli et al., “Benchmarking streaming computation engines: Storm, Flink and Spark streaming,” in Proc. of the 1st IEEE Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware (IPDRM’16), Chicago,USA, 2016.
9. Mounir Hafsa and Farah Jemili. (2018). Comparative Study between Big Data Analysis Techniques in Intrusion Detection, p1-13
10. S. Qian, G. Wu, J. Huang, and T. Das, “Benchmarking Modern Distributed Streaming Platforms,” in Proc. of the 2016 IEEE International Conference on Industrial Technology (ICIT 2016), Taipei, Taiwan, 2016, pp. 592–598.
11. M. Zaharia et al., “Resilient Distributed Datasets: A fault tolerant abstraction for in-memory cluster computing,” in Proc. of the 9th USENIX Conference on Networked Systems Design and Implementation (NSDI’12), San Jose, USA, 2012, pp. 15–18.
12. H. Bal et al., “A medium-scale distributed system for computer science research: Infrastructure for the long term,” Computer, vol. 49, no. 5, pp. 54–63, 2016.

13. J. Veiga, R. R. Exp´osito, G. L. Taboada, and J. Touri˜no, "MREv: An automatic MapReduce Evaluation tool for Big Data workloads," in Proc. of the International Conference on Computational Science (ICCS'15), Reykjav´ık, Iceland, 2015, pp. 80–89.
14. Kamran Siddique, Zahid Akhtar, Edward J. Yoon, Young-Sik Jeong, Dipankar Dasgupta and Yangwoo Kim. (2016). Apache Hama: An Emerging Bulk Synchronous Parallel Computing Framework for Big Data Applications. *IEEE*. 4,p1-9.
15. Muhammad Habib ur Rehman, Victor Chang, Aisha Batool, Teh Ying Wah. (2016). Big Data Reduction Framework for Value Creation in Sustainable Enterprises, p1-23.
16. Josh Gardner, Christopher Brook, Juan Miguel Andres, Ryan S. Baker. (2018). MORF: A Framework for Predictive Modelling and Replication at Scale with Privacy-Restricted MOOC Data, p1-10.
17. Shivaram Venkataraman, Zongheng Yang, Michael Franklin, Benjamin Recht, Ion Stoica. (2016). Ernest: Efficient Performance Prediction for Large-Scale Advanced Analytics. *USENIX Symposium on Networked Systems Design and Implementation*, p1-17.
18. Ahsan Javed Awan, Mats Brorsson, Vladimir Vlassov and Eduard Ayguade. (2015). Performance Characterization of In-Memory Data Analytics on a Modern Cloud Server, p1-8.
19. Zhengyu Yang, Danlin Jia, Stratis Ioannidis, Ningfang Mi, and Bo Sheng. (2018). Intermediate Data Caching Optimization for Multi-Stage and Parallel Big Data Frameworks, p1-10.
20. Wissem Inoubli,Sabeur Aridhi,Haithem Mezni,Mondher Maddouri,Engelbert Mephu Nguifo. (2018). An Experimental Survey on Big Data Frameworks, p1-21.

AUTHORS PROFILE

Bhupathi Prashanthi Received her B. Tech degree in computer Science and engineering Ellenki College of Engineering and technology, JNTUH, Hyderabad, Telangana in 2009, and M. Tech Degree in Computer Science and Engineering from RRS College of Engineering and technology, JNTUH, Hyderabad, Telangana in 2013. She is currently working as an assistant professor in the department of Computer Science and Engineering of Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana India. Her research interest includes : Data Mining, Image Processing and Data Structures.

Ganta Sowjanya Received her B. Tech degree in computer Science and engineering from Andhra Loyolla Institute of Engineering and Technology, JNTUK, Vijayawada, Andhra Pradesh in 2013, and M. Tech Degree in Computer Science and Engineering from DJR College of Engineering and technology, JNTUK, Vijayawada, Andhra Pradesh in2016. Research Interest: Data Mining , Image Processing. Currently she is working as an assistant professor in the department of Computer Science and Engineering of Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana India. Her research interest includes Data Mining , Image Processing and Data Structures.

D. Krishna Madhuri Received her B.Tech degree in Computer Science and Engineering from Swarnandhra College of Engineering & Technology, Narsapur, Andhra Pradesh in 2009, and M.Tech Degree in Computer Science and Engineering from Sridevi Women's Engineering College,JNTUH, Hyderabad, Telangana in 2012. She is currently pursuing her Ph.D degree from Sri Satya Sai University, Sehore, Madhya Pradesh. Currently she is working as an assistant professor in the department of Computer Science and Engineering of Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana India. Her research interest includes Data Bases, Data warehouse and Data Mining and Big Data.