MANTECH Publications

Recent Trends in Computer Science and Software Technology
Volume 9, Issue 1, January-April 2024

On Traffic-Aware Partition and Aggregation in Map Reduce For Big Data Applications

M. Shalima Sulthana[1], Mrs. K. Bhanu Sri[2], Ms. B. Ramya Sri[3]

Assistant Professor[1, 2, 3]

Department of CSE

Vignan's Institute of Management and Technology for Women Ghatkesar Hyderabad [1, 2, 3]

Corresponding Authors' Email: *m.s.sulthana2012@gmail.com[1], juluri.bhanusri@gmail.com[2], ramyasri1832@gmail.com[3]*

Abstract

Map Reduce is a scheme for processing and managing large scale data sets in a distributed cluster, which has been used for applications such as document clustering, generating search indexes, access log analysis, and numerous other forms of data analytic. In the existing system, a hash function is used to partition intermediate data among reduce tasks and most of the previous algorithms proposed concentrated on other parameters like data uploading, time reduction etc. None of them dealt with network traffic. Our proposed system consists of a decomposition-based distributed algorithm to deal with the large-scale optimization problem for large data application and an online algorithm is additionally designed to adjust data partition and aggregation in a dynamic manner. The cost of Network traffic under both offline and on-line cases is significantly reduced as demonstrated by the extensive stimulation results by the various proposals considered and used.

Keywords: *Big Bata, Data Aggregation, Dynamic Decomposition-based Distributed K- means Algorithm, HC Algorithm, Traffic Minimization.*

INTRODUCTION

The most popular computing framework for big data processing due to its simple programming model and automatic management of parallel execution is Map Reduce. The open source implementation Hadoop [4], [5] along with Map Reduce have been adopted by leading companies, such as Yahoo!, Google and Facebook, for various big data applications,

such as machine learning [6], [7], [8], bioinformatics [9], [10], [11], and cyber-security [12], [13].

This divides a computation into two main phases, namely map and reduce which in turn are carried out by several map tasks and reduce tasks, respectively. In the map phase, map tasks are parallel launched to convert the original input splits into intermediate data in a form of key/ value pairs. These pairs are stored on local machine and organized into multiple data partitions, one per reduce task. In the reduce phase, each reduce task fetches its own share of data partitions from all map tasks to generate the final result.

There is a shuffle step between map and reduce phase. In this step, the data produced by the map phase are ordered, partitioned and transferred to the appropriate machines executing the reduce phase. The resulting network traffic pattern from all map tasks to all reduce tasks can cause a great volume of network traffic, imposing a serious constraint on the efficiency of data analytic applications. For example, with tens of thousands of machines, data shuffling accounts for 58.6 percent of the cross-pod traffic and amounts to over 200 pet bytes in total in the analysis of SCOPE jobs [14]. For shuffle-heavy Map Reduce tasks, the high traffic could incur considerable performance overhead up to 30-40 percent as shown in [15]. By default, intermediate data are shuffled according to a hash function [16] in Hadoop, which would lead to large network traffic because it ignores network topology and data size associated with each key. As shown in Fig. 1, we consider a toy example with two map tasks and two reduce tasks, where intermediate data of three keys K1, K2, and K3 are denoted by rectangle bars under each machine. If the hash function assigns data of K1 and K3 to reducer 1, and K2 to reducer 2, a large amount of traffic will go through the top switch. To tackle this problem incurred by the traffic oblivious partition scheme, we take into account of both task locations and data size associated with each key in this paper. By assigning keys with larger data size to reduce tasks closer to map tasks, network traffic can be significantly reduced. In the same example above, if we assign K1 and K3 to reducer 2, and K2 to reducer 1, as shown in Fig. 1, the data transferred through the top switch will be significantly reduced.

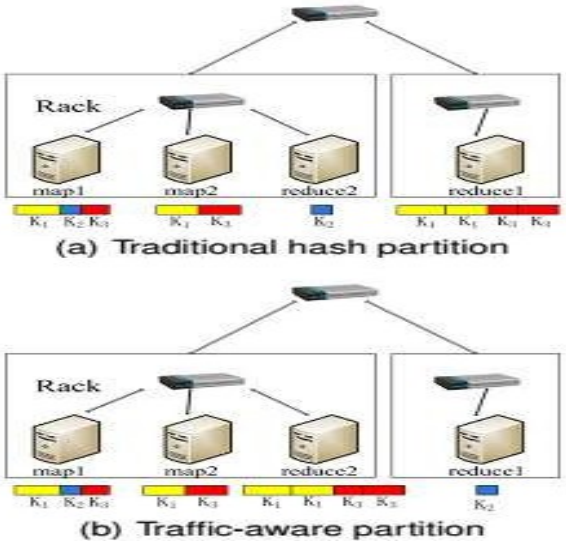


Fig .1:- System Architecture

In this paper, data partition and aggregation are jointly considered for a Map Reduce job with an objective to minimize the total network traffic. In particular, we propose a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. In addition to that an online algorithm is designed to deal with the data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that in both offline and online cases our proposals can significantly reduce network traffic cost.

LITERATURE SURVEY

The existing system proposes a Map Reduce algorithm to solve the ER problem for a huge collection of entities with multiple keys. The algorithm is characterized in the combination-based blocking and the load-balanced matching. The combination-based blocking utilizes the multiple keys to sort out necessary entity pairs for future matching. The load-balanced matching evenly distributes the required computations to all the reducers in the matching step so as to remove the bottleneck of skewed matching computations for a single node in a Map Reduce framework.

Most existing work focuses on Map Reduce performance improvement by optimizing its data transmission. Narayan et al. [18] have explored the use of Open Flow to provide better link bandwidth for shuffle traffic. Palanisamy et al. [19] have presented Purlieus, a Map Reduce

resource allocation system, to enhance the performance of Map Reduce jobs in the cloud by locating intermediate data to the local machines or close-by physical machines. This locality awareness reduces network traffic in the shuffle phase generated in the cloud data center. However, little work has studied to optimize network performance of the shuffle process that generates large amounts of data traffic in Map Reduce jobs. A critical factor to the network performance in the shuffle phase is the intermediate data partition. The default scheme adopted by Hadoop is hash-based partition that would yield unbalanced loads among reduce tasks due to its unawareness of the data size associated with each key. To overcome this shortcoming, Ibrahim et al. [20] has developed a fairness-aware key partition approach that keeps track of the distribution of intermediate keys' frequencies, and guarantees a fair distribution among reduce tasks. Meanwhile, Yan et al. [21] have introduced a sketch-based data structure for capturing Map Reduce key group size statistics and presented an optimal packing algorithm which assigns the key groups to the reducers in a load balancing manner. Hsueh et al. [22] have proposed and evaluated two effective load balancing approaches to data skew handling for Map Reduce-based entity resolution. Unfortunately, all above work focuses on load balance at reduce tasks, ignoring the network traffic during the shuffle phase. In addition to data partition, many efforts have been made on local aggregation, in-mapper combining and in network aggregation to reduce network traffic within Map- Reduce jobs. Condie et al. [23] have introduced a combiner function that reduces the amount of data to be shuffled and merged to reduce tasks. Lin and Dyer [24] have proposed an in-mapper combining scheme by exploiting the fact that mappers can preserve state across the processing of multiple input key/value pairs and defer emission of intermediate data until all input records have been processed. Both proposals are constrained to a single map task, ignoring the data aggregation opportunities from multiple map tasks. Costa et al. [25] have proposed a Map Reduce-like system to decrease the traffic by pushing aggregation from the edge into the network. However, it can be only applied to the network topology with servers directly linked to other servers, which is of limited practical use. Apart from the existing work, we investigate network traffic reduction within Map Reduce jobs by jointly exploiting traffic-aware intermediate data partition and data aggregation among multiple map tasks.

PROPOSED WORK

A distributed algorithm is proposed for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. An online

algorithm is designed whose basic idea is to postpone the migration operation until the cumulative traffic cost exceeds a threshold. The network topology is based on three tier architectures: an access tier, an aggregation tier and a core tier. It offers computers as physical or more often as virtual machines (VMs). A cluster of VMs, virtual cluster, is often requested as a platform for users to run parallel or distributed applications such as Map Reduce and Dryad applications. In order to get high throughput, fast response, load balance, low cost, and low price, many topics on VM configuration, VM placement, VM consolidation, and VM migration are explored. The network topology of a virtual cluster has a significant impact on the execution of applications running on it because the physical nodes where VMs are located can be linked in different ways. For example, some nodes are located in the same rack while others in different racks through a slow link.

A special architecture other than above architecture is the hierarchical network where two physical nodes may lie in different local area networks. Furthermore, the characteristics of different applications have different requirements for the network topology. Some applications create tasks running on different VMs which need to exchange large amount of data frequently, while others create tasks which execute independently or exchange a little data. Map Reduce and Map Reduce like models are widely used to process “Big Data”. Applications based on such models place heavily data dependency or communication on VMs, so network traffic becomes the bottleneck of jobs. The following are three phases of data exchange in the execution process of an application based on Map Reduce model.

The main focus of this paper is that it targets at provisioning a virtual cluster according to the position relationship between VMs so as to decrease the network traffic and improve the performance of Map Reduce and Map Reduce-like applications rather than modifying the job scheduling strategies We solve the shortest distance problem by formulating it into an integer linear programming.

The online heuristic VM placement algorithm and the global sub-optimization algorithm are compared by simulations. The former has lower time complexity while the latter returns shorter average distance for multiple requests. We analyze the performance of our approach through experiments. In the experiment, we adopt different virtual cluster architectures to test

different Map Reduce applications. Two metrics of application runtime and cluster affinity show the efficiency of virtual cluster optimization.

The large-scale data is processed and simplified using the Map Reduce programming model which works very well on the commodity cluster which exploits the processing by processing of map tasks and reduce tasks in parallel. There have been many efforts that have been made towards in order to enhance performance of jobs of Map Reduce; in shuffle phase network traffic that is generated is ignored many a times. It plays a key role in improving the performance. Historically, partition of intermediate data is done using a hash function in reduce tasks. However, data size and network topology for each key are not considered. Hence this is not efficient from traffic perspective. Following are the two issues this project aims to resolve:

Network traffic: Cost of the network traffic has to be decreased using a novel intermediate data partition method.

Aggregator placement problem: An algorithm called the distribution algorithm which is designed to solve the issue of optimize the big-scale decomposition of the big data applications

Here, we develop a distributed algorithm to solve the problem on multiple machines in a parallel manner. Our basic idea is to decompose the original large-scale problem into several distributive solvable sub problems that are coordinated by a high-level master problem. The model is as shown in the

Fig 2. The figure shows the Map Reduce with the aggregators placed for processing of traffic aware scheme

Algorithm 1 Distributed Algorithm

1: set $t = 1$, and $\nu_j^p (j \in A, p \in P)$ to arbitrary nonnegative values;
2: **for** $t < T$ **do**
3: distributively solve the subproblem SUB_DP and SUB_AP on multiple machines in a parallel manner;
4: update the values of ν_j^p with the gradient method (15), and send the results to all subproblems;
5: set $t = t + 1$;
6: **end for**

DISTRIBUTED ALGORITHM

Now, we verify that our distributed algorithm can be applied in practice using real trace in a cluster consisting of 5 virtual machines with 1GB memory and 2GHz CPU. Our network topology is based on three tier architectures: an access tier, an aggregation tier and a core tier (Fig. 2). The access tier is made up of cost effective Ethernet switches connecting rack VMs. The access switches are connected via Ethernet to a set of aggregation switches which in turn are connected to a layer of core switches. An inter-rack link is the most contentious resource as all the VMs hosted on a rack transfer data across the link to the VMs on other racks.

In three different racks the VMs are distributed and the Map-reduce tasks are scheduled as in Fig. 2. For example, rack 1 consists of node 1 and 2; mapper 1 and 2 are scheduled on node 1 and reducer 1 is scheduled on node 2. The intermediate data forwarding between mappers and reducers should be transferred across the network. The hop distances between mappers and reducers are shown in Fig. 4, e.g., mapper 1 and reducer 2 has hop distance 6.data forwarding between mappers and reducers should be transferred across the network. The hop distances between mappers and reducers are shown in Fig. 2, e.g., mapper 1 and reducer 2 has a hop distance 6.

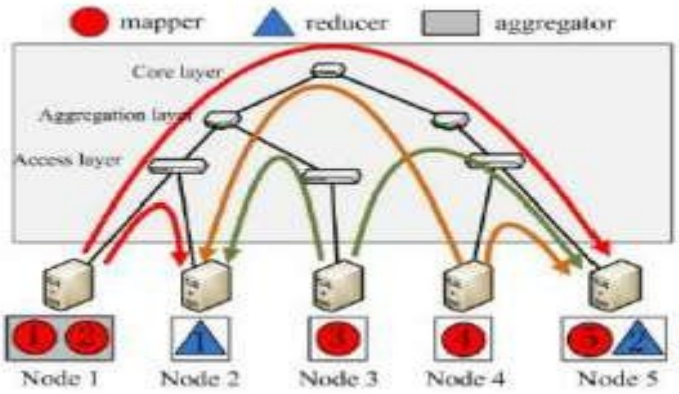


Fig. 2:- Map Reduce with Aggregators

The intermediate data from all mappers is transferred according to the traffic-aware partition scheme. We can get the total network 2690:48 in the real Hadoop environment while the simulated network cost is 2673:49. They turn out to be very close to each other, which indicate that our distributed algorithm can be applied in practice.

RESULTS AND DISCUSSION

The shortest distance problem is solved by formulating it into an integer linear programming concept. The online heuristic VM placement algorithm and the global sub-optimization algorithm are compared by simulations. The former has lower time complexity while the latter returns shorter average distance for multiple requests. We analyze the performance of our approach through experiments. In the experiment, we adopt different virtual cluster architectures to test different Map Reduce applications. Two metrics of application runtime and cluster affinity show the efficiency of virtual cluster optimization. Heuristic distance is mapped to the virtual cluster with the most appropriate central node built by our online VM placement algorithm. Shortest distance with a random central node is mapped to the same virtual cluster, however, the central node is chosen randomly. It is clear that even if we select the same virtual cluster, but not the same position of the central node, the distance difference is also great. For Map Reduce applications, the selection of the central node is the same important with the architecture of the virtual cluster. Each physical node has different capability and each request has different requirement. In this paper we have described a map reduce frame work used for parallel processing large amount of data. Although there are many existing works which have been focused on the traffic reduction, they did not do well in the map reduce paradigm. Those works focused mainly on the map and reduce phases instead of concentrating on shuffle phase. They did not attempt to reduce the data traffic for the improvement of the network traffic and the data cost.

In the below chart we can observe the difference between the length of both Aggregation and No Aggregation time. We can observe that Aggregation length is higher than No Aggregation length. The difference will be shown in terms of Processing Time. We have implemented an efficient system for map reduces jobs by data partition and aggregation for big data applications for improving the network traffic performance and reducing the network cost.

PDS Cloud

PDS Cloud has a hierarchical data model supporting independent tenants whose assets are organized in multiple aggregations based on content and value. Continuous changes to data objects, life-cycle activities, virtual appliances and cloud providers are applied in a manner transparent to the client. PDS Cloud is being developed as an infrastructure component of the

European Union ENSURE project, where it is used for preservation of medical and financial data.

Linked Open Data Aggregation

Information aggregation comes with inherent problems, such as provision of poor quality, inaccurate, irrelevant or fraudulent information. The sharing of this data would result in a more complete bibliographic data landscape for manga; the majority of this data, however, exists in isolation from other institutions. In seeking to combine this data, this paper presents a Linked Data model for the aggregation of bibliographic data for manga. The Europeana Data Model was used as the basis for the aggregation functions

Partition-Based Online Aggregation

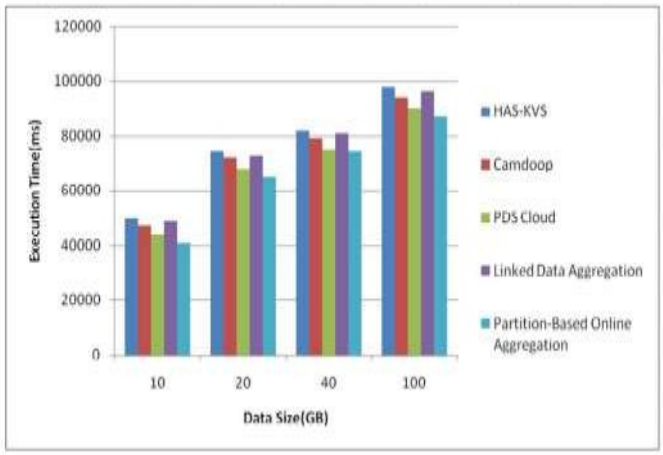
Online aggregation is an attractive sampling-based technology to response aggregation queries by an estimate to the final result, with the confidence interval becoming tighter over time. It has been built into a Map Reduce-based cloud system for big data analytics, which allows users to monitor the query progress, and save money by killing the computation early once sufficient accuracy has been obtained. Content-aware repartition method with a fair-allocation blocks placement strategy to increase the sampling efficiency and guarantee the storage and computation load balancing simultaneously.

HAS-KVS

High Speed aggregation with distributed key value store (KVS). Distributed KVS accomplishes this high performance and availability with distributing data between various servers, however distributing data produces several specific types of processing complex at the same time. A typical instance of this kind of processing is aggregation processing, in which various data are aggregated to generate results. With distributed KVS, where data are allocated between servers, a great several pieces of communication among servers are produced for aggregation, which acquires time. To address this problem developing a method to recognize high-speed aggregation with distributed KVS, and realized a research model presenting about eight times higher speed than that of the previous technique. With the innovative method, several basic operations (such as rekey, map, filter and reduce) that effectively run in distributed KVS have been utilized and combined to recognize aggregation processing.

CAMPDOOP

Camdoop, a Map Reduce-like system running on Cam Cube, a cluster designs that uses a direct-connect network topology with servers directly linked to other servers. Camdoop exploits the property that Cam Cubeservers forward traffic to perform in-network aggregation of data during the shuffle phase. Camdoop supports the same functions used in Map Reduce and is compatible with existing Map Reduce applications.



Graph showing Various Time Costs

CONCLUSION

In this paper, we study the joint optimization of intermediate data partition and aggregation in MapReduce to minimize network traffic cost for big data applications. We propose a three-layer model for this problem and formulate it as a mixed-integer nonlinear problem, which is then transferred into a linear form that can be solved by mathematical tools. To deal with the large-scale formulation due to big data, we design a distributed algorithm to solve the problem on multiple machines. Furthermore, we extend our algorithm to handle the MapReduce job in an online manner when some system parameters are not given. Finally, we conduct extensive simulations to evaluate our proposed algorithm under both offline cases and online cases. The simulation results demonstrate that our proposals can effectively reduce network traffic cost under various network settings. The future scope of the works should be made on complex data partitioning in the database where more intelligent methods have to be employed, which includes analyzing computation cost, skew record etc.

REFERENCES

1. J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
2. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map task scheduling in mapreduce with data locality: Throughput and heavytraffic optimality," in Proc. IEEE INFOCOM, 2013, pp. 1609–1617.
3. F. Chen, M. Kodialam, and T. Lakshman, "Joint scheduling of processing and shuffle phases in mapreduce systems," in Proc. IEEE INFOCOM, 2012, pp. 1143–1151.
4. Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy data uploading approach for the hadoop distributed file system," in Proc. IEEE Int. Conf. Cluster Comput., 2013, pp. 1–5.
5. T. White, Hadoop: The Definitive Guide: The Definitive Guide. Sebastopol, CA, USA: O'Reilly Media, Inc, 2009.
6. S. Chen and S. W. Schlosser, "Map-reduce meets wider varieties of applications," Intel Res., Pittsburgh, PA, USA, Tech. Rep. IRP-TR-08-05, 2008.
7. H. Lv and H. Tang, "Machine learning methods and their application research," IEEE Int. Symp. Intel. Info. Process. Trusted Comput. (IPTC), pp. 108–110, Oct. 2011.
8. S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, "Presto: Distributed machine learning and graph processing with sparse matrices," in Proc. 8th ACM Eur. Conf. Comput. Syst., 2013, pp. 197–210.
9. A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in Proc. IEEE 4th Int. Conf. eScience, 2008, pp. 222–229.
10. J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, "Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study," in Proc. 4th Int. Workshop Data Intensive Comput. Clouds, 2013, pp. 1–5.
11. R. Liao, Y. Zhang, J. Guan, and S. Zhou, "Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for largescale biological datasets," Genomics, Proteomics Bioinformat., vol. 12, no. 1, pp. 48–51, 2014.
12. G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing map-reduce to high end computing," in Proc. 3rdPetascale Data Storage Workshop, 2008, pp. 1–6.

13. W. Yu, G. Xu, Z. Chen, and P. Moulema, “A cloud computing based architecture for cyber security situation awareness,” in Proc. IEEE Conf. Commun. Netw. Security, 2013, pp. 488–492.
14. J. Zhang, H. Zhou, R. Chen, X. Fan, Z. Guo, H. Lin, J. Y. Li, W. Lin, J. Zhou, and L. Zhou, “Optimizing data shuffling in data-parallel computation by understanding user-defined functions,” in Proc. 9th USENIX Conf. Netw. Syst. Des. Implemen. (NSDI ’12), Berkeley, CA, USA: USENIX Association, 2012, pp. 295–308.
15. F. Ahmad, S. Lee, M. Thottethodi, and T. Vijaykumar, “Mapreduce with communication overlap,” J. Parallel Distrib. Comput., vol. 73, pp. 608–620, 2013.
16. H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, “Mapreduce-merge: Simplified relational data processing on large clusters,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2007,pp. 1029–1040.
17. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, “Online aggregation and continuous query support in mapreduce,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2010, pp. 1115–1118.
18. S. Narayan, S. Bailey, and A. Daga, “Hadoop acceleration in an openflow-based cluster,” IEEE SC Companion: High Performance Comput., Netw., Storage Analysis (SCC), pp. 535–538, Nov. 2012.
19. B. Palanisamy, A. Singh, L. Liu, and B. Jain, “Purlieus: Localityaware resource allocation for mapreduce in a cloud,” in Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal., 2011, p. 58.
20. S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, “Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud,” in Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci., 2010, pp. 17–24.
21. W. Yan, Y. Xue, and B. Malin, “Scalable and robust key group size estimation for reducer load balancing in MapReduce,” IEEE Int. Conf. Big Data, pp. 156–162, Oct. 2013.
22. S.-C. Hsueh, M.-Y. Lin, and Y.-C. Chiu, “A load-balanced mapreduce algorithm for blocking-based entity-resolution with multiple keys,” in Proc. 12th Australasian Symp. Parallel Distrib. Comput., 2014, pp. 3–9.

23. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce online," in Proc. 7th USENIX Conf. Netw. Syst. Design Implementation, 2010, vol. 10, no. 4, p. 20.
24. J. Lin and C. Dyer, "Data-intensive text processing with mapreduce," Synthesis Lectures Human Language Technol., vol. 3, no. 1, pp. 1–177, 2010.
25. P. Costa, A. Donnelly, A. I. Rowstron, and G. O'Shea, "Camdoop: Exploiting in-network aggregation for big data applications," in Proc. 7th USENIX Conf. Netw. Syst. Design Implementation, 2012, vol. 12, p. 3.

AUTHOR PROFILE

Mannuru Shalima Sulthana Submitted Thesis in the field of Machine learning under the supervision of C. Nagaraju Professor of Computer Science and Engineering, YSR Engineering College of Yogivemana University, Proddatur, Kadapa Andhra Pradesh. She received her B. Tech in Information Technology at SSITS Rayachoty. M.Tech in Computer Science and Engineering at SSITS Rayachoty. She has six years of teaching experience as an Assistant Professor in the department of Computer and Engineering at IIIT RGUKT RKVALLEY Idupulapaya; she has published 19 Research papers in various national and international journals and attended four Conferences. She has attended 23 Faculty Development Programs and workshops. She has qualified AP-SET and AP-RCET in 2019.She is Permanent member of various professional societies like IAENG (294777) International Association of Engineers and Member in SDIWC (The Society of Digital Information and Wireless Communications)-8714. She has file done patent on - System and method for remote health monitoring using IoT driven using machine learning algorithms.