International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-1, October, 2019

# Optimized and Efficient Computation of Big Data in Heterogonous Internet of Things

**P. S. Rajakumar, S. Vijayanand, G. Sreeram**

***Abstract:*** ***The development of information technology, distributed computing, hardware, wireless communication and intelligent technology has been increased in Internet of Things (IoT) heterogonous filed to improve the limitations of cloud computing in big data processing. Computation of data over wireless communication based distributed computing face different challenges in term of off-loading decision, data delay in heterogonous IoT devices. Optimization of caching, data computation and load maintenance of different edge clouds is still a challenging task in heterogonous IOT for effective processing of big data. So that this paper presents Novel Optimized and Sorted Positional Index List (OSPIL) approach on big data processing to reduce and optimize delay, I/O cost, CPU usage and memory overhead significantly. In this approach sorted index is used to build attributes are (data consists different attributes) arranged in ascending order. This approach consists two phases in data processing, in Phase 1, scan the depth of all the sorted list and schedule the processing data. In phase 2, explore the sorted list and then give results in sequential order on hash table. Experimental results of proposed approach give better and significant data processing results to optimize delay, I/O cost, CPU usage and memory overhead on real world data sets relates wireless communication***

***Keywords: Distributed environment, internet of things, sorted positional index, wireless communication and edge cloud computing.***

## I. INTRODUCTION

Because of continuous changes in user's communication, implementation of Internet of Things, wireless technologies has been brought based on tremendous changes in communication. From first generation of networks to 4 and 5 generation networks, IoT networks further realizes inter communication/connections between different things and between peoples and things based on increasing modern technology in wireless communication. In recent years, IoT act as bridge connection between peoples and physical network world, some of the advanced exhaustive features involved in implementation of IoT. Present improvement of wireless communication technology, 3-5 generation partnership technology consists 3 modes. Enhanced wireless broadband, massive type of communication, reliable latency communication. To process data in heterogonous IoT devices require high amount of energy, cost, high delay and reliable communication. To satisfy above tasks in computing user diverse requests and computing tasks based on scheduling, researchers to make cloud computing achieve solve the limitations of storage and computing based IoT devices. Artificial Intelligence enabled edge joint computation, caching and communication in heterogonous IoT (Smart-edge- CoCaCo) is used to reduce overcast I/O cast and improve efficiency of user service data processing for cloud execution to optimize the computational efficiency. Increase coverage, off-loading delay; optimize the network structure and computing caching and off-loading of heterogonous IoT. Smart-edge- CoCaCo can't process big data efficiently. And also people expect data processing results quickly. Based on these problems, in this paper we propose and present a Novel Optimized and Sorted Positional Index List (OSPIL) approach on big data processing to reduce and optimize delay, I/O cost, CPU usage and memory overhead significantly. In this approach sorted positional index is constructed for each attribute (data consists different attributes) arranged in ascending order. OSPIL approach consists two phases to process data, explore the positional index of each user in phase 1, retrieve the data processing results in phase 2. In Phase 1, OSPIL first retrieves sorted positional index list (L1, L2,....., Ln) arrange round robin scheduling criteria (A1,A2,.....,An), mathematical analysis is discussed to calculate the depth d of overall data set. In Phase 2, explore each sorted positional index list from data store and obtain data processing results, this procedure done in OSPIL is called as pruning operation. Extensive experimental are conducted on different synthetic data sets, OSPIL gives better results when compare to traditional approaches. Basic contributions of proposed approach in this paper as follows

a) Presents novel Optimized and Sorted Positional Index List approach for big data processing, which can be constructed different data structures to reduce I/O cost significantly.
b) Describe procedure of sorted positional index for data processing in distributed edge computing.
c) Discuss about pruning operation to arrange data in positional index with respect to mathematical analysis.
d) Shows significant experimental results compare over

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
**Dr.P.S.Rajakumar***, Computer Science and Engineering , Dr.M.G.R. Education and Research Institute, Chennai, India. Email: suraarus@yahoo.com
**Dr. S. Vijayanand**[2], Computer Science and Engineering., Rajarajeswari College of Engineering, Bangalore , India. Email: kgsvanand@gmail.com
**Dr. G. Sreeram**, department, Name Koneru Lakshmaiah Education Foundation , Guntur, India. Email: sreeram@kluniversity.in

## II. SMART-EDGE-COCACO BASED HETEROGONOUS IOT SERVICES

This section describes heterogonous IOT services with respect to enable smart edge computing services in intelligent devices. To realize the automation, intelligent and humanization in heterogonous IOT services, new architecture is defined to enable smart edge computing service in IOT, design of new algorithm described in figure 1.

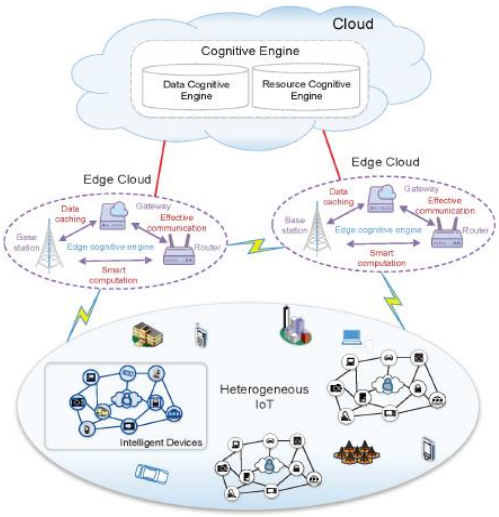


**Figure 1. New architecture to define smart edge computing IoT services.**

As shown in figure 1, it define basic design of heterogonous IoT, this architecture consists edge cloud and remote cloud[16]. Basic design of edge computing and artificial intelligence, it can be dynamically monitor and maintain network design resources and realize efficient processing of compute data in smart applications. In heterogonous IoT, users directly contact with smart device provided by the operators present in edge computing environment. Heterogonous IoT is mainly designed for delay sensitive application services relates to most research subject relates to dynamic network resources for efficient processing computing. In heterogonous IoT, Smart-Edge-CoCaCo algorithm is one of component to improve the efficiency in communication and hit ratio to achieve off-loading. For different data volume computation, smart edge CoCaCo calculation selects the location of complexity off-loading and then balance network load, provide relieve from congestion in network and reduce the network delay of computation off-loading. Still smart edge CoCaCo calculation method faces some challenges i.e. different off-loading data running on server then which of the computation tasks runs on local machine and which computational tasks running on remote cloud. Therefore we need to implement caching, communication and computation of edge computing, optimize the computation off-loading and reduce the overall delay in processing of different data tasks in heterogonous IoT.

## III. REVIEW OF RELATED WORK FORPOSITIONAL INDEX OF DATA

This section describes different methodologies to process big data in edge computing. Recent advances in innovation have enabled associations to gather amazingly a lot of information, envisioning high an incentive in breaking down them. "Huge Data" the executives and handling have been one of the greatest difficulties of our time. Current methodologies comprise of handling frameworks conveyed on a lot of production machines and adventure monstrous parallelism to productively break down colossal datasets [1][17]. The best framework is the Google's Map Reduce structure, which conceals the multifaceted nature of information dispersion, correspondence and undertaking booking and offers a straightforward programming model for composing systematic applications, while likewise giving solid adaptation to non-critical failure ensures[2][3]. Different data processing for Map-Reduce functionality have been proposed with the Apache Hadoop framework with most preferable data received, based on Map Reduce programming model, HDFC executive data relations in data sources In spite of its ubiquity, the Map-Reduce model and its Hadoop execution have likewise been reprimanded and have been contrasted with present-day parallel database management systems (DBMSs), as far as execution and multifaceted nature [4][5]. There have been broad investigations on Map Reduce attributes, recognizing a lot of inadequacies of the model and current executions[18]. The blast of Big Data investigation is a noteworthy driver for datacenter figuring. As the volume, assortment, and speed of the information routinely gathered by business, logical and administrative clients far surpasses the limit of a solitary server, scaling execution in the Big Data period is essentially done through expanding the number of servers[6]. Be that as it may, this methodology of scaling execution leaves Big Data registering presented to a vitality usage issue, and mounting operational overheads identified with the datacenter costs, for example, facilitating space, cooling and labor costs[7]. Associatively with the blast of Big Data, the previous couple of years have seen a tremendous development of the preparing rate of ARM-based cell phones, for example, advanced mobile phones and tablets. Because of the quick developing scene of versatile equipment, and in an offer to decrease the vitality related costs, numerous organizations and research ventures are progressively taking a gander at utilizing non-customary equipment as server stages [10, 9]. For instance, Barcelona Supercomputing Center is seeing utilizing ARM-based frameworks as the reason for their scale stage [9]. Key equipment sellers, for example, Dell, HP, and Applied Micro have propelled server models dependent on ARM processors [11], and plenty of new companies are investigating embracing ARM arrangements in the endeavor registering scene. Indeed, even AMD, which verifiably has just sent server processors dependent on x86/x64 design, focuses to dispatch ARM-based servers [7][8]. Scaling Big Data execution requires different server hubs with great CPU and I/O assets. Naturally, top of the line ARM-based servers could fit this bill well, as they have a moderately decent parity of these two assets [12][13]. Besides, their low vitality utilization, low cost, and little physical size make them appealing for group arrangements. This normally brings up the examination issue of the achievability of low-control ARM servers as contenders for conventional Intel/AMD x64 servers for Big Data handling.

On the off chance that an ARM-based bunch can coordinate the execution of a conventional Intel/AMD group with lower vitality or cost, this could introduce another period of green processing that can help Big Data examination achieve new dimensions of execution and cost-proficiency[14][15].

## IV. NOVEL OPTIMIZED AND SORTED POSITIONAL INDEX LIST IMPLEMENTATION PROCEDURE

In this section, we define novel implementation procedure for sorting data and arrange data in different positional index based on attribute values presents pruning, define the implementation of Optimized and Sorted Positional Index List (OSPIL) performed on computation of data in heterogonous IoT. Basic preliminaries used in our implementation shown in table 1.

**Table 1. Description of used parameters in OSPIL implementation.**

| Used parameter | Description |
|---|---|
| T | Skyline Query table |
| $L_i$ | Attribute based sorted positional index |
| $AS_{skyline}$ | Sorted Skyline |
| N | T with tuple number |
| M | $AS_{skyline}$ size |
| D | Depth of data set attributes |
| HT | PI based Hash table |
| $SET_{num}$ | Sorted positional index |

Symbols used in OSPIL implementation with description show in table 1. Basic things used in our implementation as follows:

**1.1. Sorting data in positional index format:** Let us consider the table T with positional index (PI) of $t \in T$ is i, if and only if t is the $i^{th}$ record of T.

Based on the above preliminaries we denote T(i) be the tuple in T with positional index PI=i and T[i][j], $j^{th}$ attribute with $i^{th}$ tuple in Table T(i), execution of OSPIL requires positional index list for each attribute value. Table T consists different attributes i.e. $T = (A_1, A_2, ....., A_M)$ , maintenance of positional index $L_j$ for each attribute $A_j (1 \le j \le M)$ . $L_j$ stores the information in T and define positional index in ascending order of attribute $A_j$, relation between positional index for each attribute as follows:

$$\forall_{i_2}, i_2 \left(1 \le i_1 \prec i_2 \le n\right), T\left(L_j(i_1)\right)[j] \le T\left(L_j(i_2)\right)[j]$$

Procedure for arrange positional index in ascending order as follows: First Table T keeps set of data in columns i.e. $CS = \{C_1, C_2, ....., C_M\}$ , each column in dataset $C_j$ is represented as $C_j(PI, A_j)(1 \le j \le M)$

PI denotes positional index in table T and $A_j$ be the corresponding attribute T (PI), based on above relation $C_j$ arrange data in ascending order according to different attribute $A_j$ . OSPIL constructs ascending positional index for each attribute $A_j$ only M list needed. OSPIL reduces overhead of data structures used to arrange data in ascending order from exponential linear formation of each attribute $A_j$. This process is continued until arrange all the attributes in positional index with different attributes in big data.

### 1.2. Overview of Proposed OSPIL Algorithm

Based on procedure to arrange attributes in ascending order with prescribed attribute relation. OSPIL consists two phases for processing and analyzing data.

**Phase -1:**

OSPIL retrieve categorized data from Skyline server with existing sorted positional index set. OSPIL preserve categorized data with OSPIL retrieval with observed dataset details. Best positional index allocated for each tuple based on parallel processing of different documents. If tuple in hash table filled with record PI and then go to next record to process remaining data relations embedded with different data processors with allocated parallel positions.

**Phase –II:** In OSPIL, phase 2 is used to contrasting sequential data processing of skyline synthetic data. It describes the tuples partitioning in T with standard data shown in line 2 of algorithm of OSPIL. OSPIL describes best positional index PI in table T with different elements using two basic commands SET and GET. SET is used for accessing data from server based on positional index, GET is used to store data directly in hash table and allocate index for data with respectable CPU usage and input and output cost with different tuple partitioning. OSPIL define computation of synthetic data with different data set processing. Step by step procedure arrange data in positional index to reduce overhead in processing data in terms of data from big data show in algorithm 1.

Implementation of positional index for different attributes shown in algorithm 1, we perform early pruning in phase 1 and then retrieving data with positional index for each attribute, pruning give better and efficient results in high dimensional data to arrange data in ascending positional index with respect to each attribute. We discuss performance of OSPIL in terms of retrieval tuples of data as a measure of CPU overhead, input and output (I/O) cost in processing of data from high dimensional data in heterogonous IoT. Performance evaluation of proposed approach discussed in next section.

**Algorithm 1. Step by step procedure to evaluate data analysis in heterogonous IoT.**

Input: Different synthetic data sets relates to social networks and indexed in T1,T2,....,Tn , categorize the positional index between $L_j (1 \le j \le m)$

1. Initially list_index=0, Boolean log= true /false
2. For i=1 to n
3. Arrange positions (pi1, pi2,....., pin ) from recorded data L=L1,L2,...,Ln
4. Update PI = record_PI+1

5. For i=1 to n, apply early pruning (each positional index), continue
6. Arrange data in HT
7. Apply late_pruning with different data processing and arrange them in ascending_order
8. SET data and process into ascending order data
9. Retrieval all the tuple_data.
10. Execute all the tuple partitioning values into single aspect of positional index data. Store and process PI data.

## V. EXPERIMENTAL EVALUATION

In this section, we discuss about the performance of OSPIL, we use JAVA platform for implementation of user interface in OSPIL with client and server architecture to process data in heterogonous internet of things. Results executed I3 Pentium processor CPU and 4 GB RAM 500 GB hard disk, the data stored in Seagate Barracuda ST32000542AS. We evaluate the performance of OSPIL against LESS[2], Zserch [3] and enable smart CoCaCo [1] approaches.

**Table 2. Parameters used in implementation of OSPIL.**

| Used Parameter | Described value |
|---|---|
| Synthetic/Real data sets | >400 MB |
| Recrods_length | 512 bits for each attribute |
| Attribute features | Arranged in index |
| Hash-key length | 16 bits |

Experiments are executed in this document performed on two data sets i.e. uniform distributed and real datasets. Basic parameters used in this implementation described in table 2.Using these parameters, implementation of OSPIL with comparison of existing approached values shown in table 3 and performance evaluation shown in figure 2.

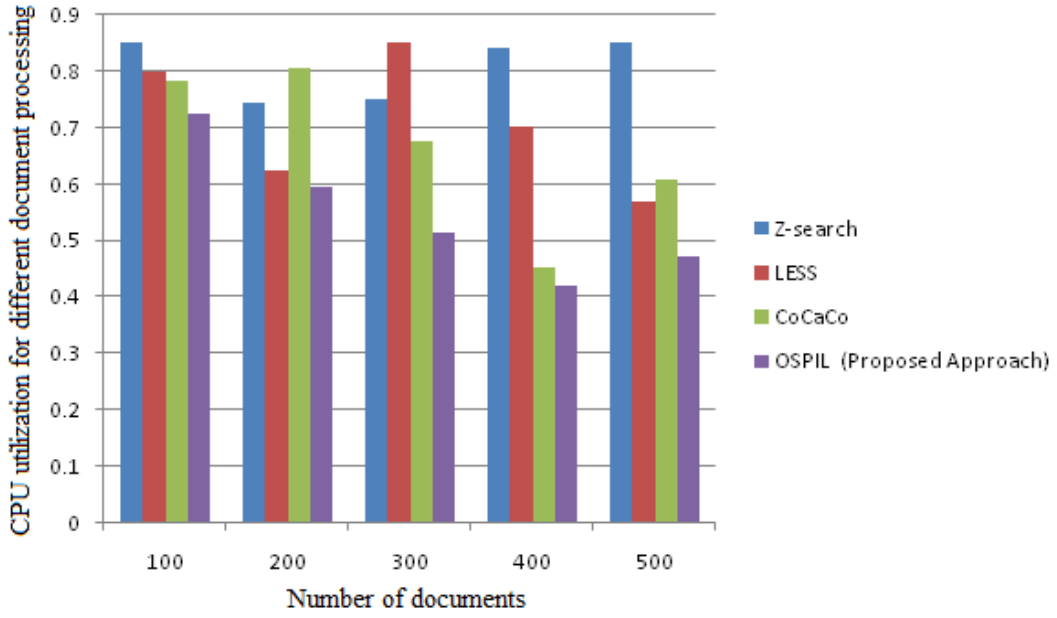


**Figure 2. Performance of CPU utilization in different document processing.**

**Table 3. Evaluation values for different memory usage in different file processing**

| Documents | Z-search | LESS | CoCaCo | OSPIL (Proposed Approach) |
|---|---|---|---|---|
| 100 | 0.56 | 0.342 | 0.4245 | 0.352 |
| 200 | 0.42 | 0.301 | 0.415 | 0.401 |
| 300 | 0.472 | 0.399 | 0.28 | 0.338 |
| 400 | 0.445 | 0.444 | 0.392 | 0.345 |
| 500 | 0.472 | 0.333 | 0.335 | 0.345 |

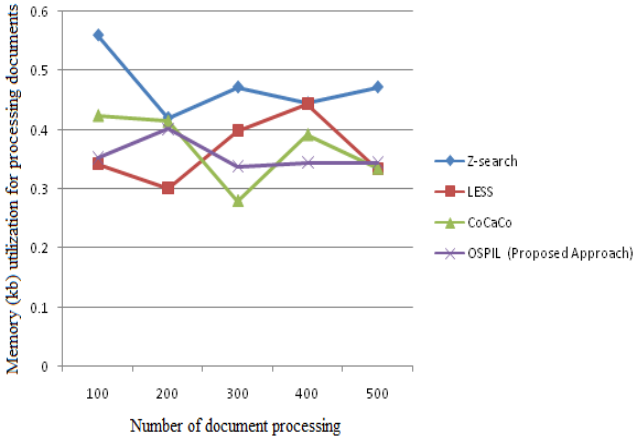


**Figure 3. Performance evaluation of memory utilization.**

Performance evaluation of CPU utilization with respect to processing of documents in heterogonous IoT. Table 3 and figure 2 gives memory utilization to process documents
Performance evaluation of memory utilization in processing of documents in real time heterogonous IoT. Time efficiency values for different documents of proposed approach with existing approaches shown in table 4 and figure 4.

**Table 4: Evaluation values for different documents**

| Documents | Z-search | LESS | CoCaCo | OSPIL (Proposed Approach) |
|---|---|---|---|---|
| 100 | 0.61 | 0.412 | 0.501 | 0.427 |
| 200 | 0.552 | 0.346 | 0.496 | 0.468 |
| 300 | 0.598 | 0.486 | 0.346 | 0.367 |
| 400 | 0.61 | 0.462 | 0.454 | 0.329 |
| 500 | 0.61 | 0.376 | 0.396 | 0.356 |

## VI. CONCLUSION

In this paper, we present and introduce a Novel Optimized and Sorted Positional Index List (OSPIL) approach on big data processing to reduce and optimize delay, I/O cost, CPU usage and memory overhead significantly. This approach consists 2 phases, in first phase, explore different documents based on attribute labels, in second phase, arrange positional index for each attribute in ascending order to evaluate data attributes for real time data sets. Experimental evaluation of proposed approach gives optimized and better results in document processing and reduces over a head in CPU, memory and time utilization. Further improvement of proposed approach developed in 3 interesting scenarios: Local processing, Routing in data processing, merging of results in terms of load based data processing.

## REFERENCES

1. Yixue Hao, Yiming Miao, Long Hu, M. Shamim Hossain, Ghulam Muhammad, and Syed Umar Amin, "Smart-Edge-CoCaCo: AI-Enabled Smart Edge with Joint Computation, Caching, and Communication in Heterogeneous IoT", 0890-8044/19/$25.00 © 2019 IEEE.
2. P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and Analyses for Maximal Vector Computation," The VLDB J., vol. 16, no. 1, pp. 5- 28, 2007.
3. K.C. Lee, W.-C. Lee, B. Zheng, H. Li, and Y. Tian, "Z-Sky: An Efficient Skyline Query Processing Framework Based on ZOrder," The VLDB J., vol. 19, no. 3, pp. 333-362, 2010.
4. Y. Tao, X. Xiao, and J. Pei, "Efficient Skyline and Top-K Retrieval in Subspaces," IEEE Trans. Knowledge Data Eng., vol. 19, no. 8, pp. 1072-1088, Aug. 2007.
5. Xixian Han, Jianzhong Li, "Efficient Skyline Computation on Big Data", IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 11, NOVEMBER 2013.
6. M. Barbera et al., "To Offload or Not to Offload? The Bandwidth and Energy Costs of Mobile Cloud Computing," Proc. IEEE INFOCOM, Turin, Italy, 2013, pp. 1285–93.
7. M. Chen et al., "Edge-CoCaCo: Toward Joint Optimization of Computation, Caching, and Communication on Edge Cloud," IEEE Wireless Commun., vol. 25, no. 3, 2018, pp. 21–27.
8. M. Chen et al., "Wearable Affective Robot," IEEE Access, vol. 6, 2018, pp. 64,766–76.
9. M. Chen et al., "Green and Mobility-Aware Caching in 5G Networks," IEEE Trans. Wireless. Commun., vol. 16, no. 12, 2017, pp. 8347–61.
10. M. Gibas, G. Canahuate, and H. Ferhatosmanoglu, "Online Index Recommendations for High-Dimensional Databases Using Query Workloads," IEEE Trans. Knowledge and Data Eng., vol. 20, no. 2, pp. 246-260, Feb. 2008.
11. P. Godfrey, "Skyline Cardinality for Relational Processing," Foundations of Information and Knowledge Systems, vol. 2942, pp. 78-97, Springer Berlin/Heidelberg, 2004.
12. P. Godfrey, R. Shipley, and J. Gryz, "Algorithms and Analyses for Maximal Vector Computation," The VLDB J., vol. 16, no. 1, pp. 5- 28, 2007.
13. J. Gray and P.J. Shenoy, "Rules of Thumb in Data Engineering," Proc. 16th Int'l Conf. Data Eng. (ICDE '00), pp. 3-12, 2000.
14. K. Hose and A. Vlachou, "A Survey of Skyline Processing in Highly Distributed Environments," The VLDB J., vol. 21, no. 3, pp. 359-384, 2012.
15. L. Tong, Y. Li, and W. Gao, "A Hierarchical Edge Cloud Architecture for Mobile Computing," Proc. IEEE INFOCOM, San Francisco, CA, 2016, pp. 399–400.
16. P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Commun. Surveys & Tutorials., vol. 19, no. 3, 2017, pp. 1628–56.
17. Published an article on "Ensuring an Efficient Access Control Security in Cloud Computing Using Broadcast Group Key Management" in the International Journal - Sylwan Publisher, Poland, 160(6):219-230 • June 2016.
18. Published an article on "Secure data deduplication using efficient Cokm in cloud storage" in the International Journal of Applied Engineering Research 10(7):17553-17561 • January 2015.

## AUTHORS PROFILE

**P.S.Rajakumar** M.Tech degree in Computer Science and Engineering from Dr.M.G.R.Educational and Research Institute University in 2006, and received his Ph.D. degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad, Telangana, Andhra Pradesh, India in 2018. Currently working as a Professor in the department of Computer Science and Engineering at Dr.M.G.R.Educational and Research Institute, Chennai, Tamil Nadu, India.

**S.Vijayanand** received his B.E Degree in Computer Science and Engineering from University of Madras, Chennai, Tamil Nadu, India in 2004, M.Tech degree in Computer Systems and Networks from Dr.M.G.R.Educational and Research Institute University in 2006, and received his Ph.D. degree in Computer Science and Engineering from Sri Krishna Devaraya University, Anantapur, Andhra Pradesh, India in 2017. Currently working as a Associate Professor in the department of Computer Science and Engineering at Rajarajeswari Colleg of Engineering, Bangalore, Karnataka, India.

**Sreeram Gutha** received his B.Tech Degree in Information Technology from University of Madras, Chennai, Tamil Nadu, India in 2004, M.Tech degree in Computer Systems and Networks from Dr.M.G.R.Educational and Research Institute University in 2006, and received his Ph.D. degree in Computer Science and Engineering from Jawaharlal Nehru Technological University, Kakinada, Andhra Pradesh, India in 2018. Currently working as a Associate Professor in the department of Computer Science and Engineering at Koneru Lakshmaiah Education Foundation, Guntur, Andhra Pradesh, India.