

A noble approach to develop dynamically scalable namenode in hadoop distributed file system using secondary storage

Tumpa Rani Shaha¹, Md. Nasim Akhtar², Fatema Tuj Johora³, Md. Zakir Hossain⁴,
Mostafijur Rahman⁵, R. B. Ahmad⁶

^{1,2,4}Department of Computer Science and Engineering, Dhaka University of Engineering and Technology (DUET),
Gazipur, Bangladesh

³Institute of Information Technology, Jahangirnagar University (JU), Bangladesh

⁵Department of Software Engineering, Daffodil International University (DIU), Dhaka, Bangladesh

¹Department of Computer Science and Engineering, Daffodil International University (DIU), Dhaka, Bangladesh

⁶Faculty of Informatics and Computing, University Sultan Zainal Abidin (UniSZ), 22200 Besut, Terengganu, Malaysia

Article Info

Article history:

Received Nov 12, 2018

Revised Dec 13, 2018

Accepted Dec 27, 2018

Keywords:

DataNode

Hadoop

Metadata

NameNode

Secondary Storage

ABSTRACT

For scalable data storage, Hadoop is widely used nowadays. It provides a distributed file system that stores data on the compute nodes. Basically, it represents a master/slave architecture that consists of a NameNode and copious Data Nodes. Data Nodes contain application data and metadata of application data resides in the Main Memory of NameNode. In cached approach, they fragment the metadata depending on the last access time and move the least frequently used data to secondary memory. If the requested data is not found in main memory then the secondary data will be loaded again on the RAM. So when the secondary data reloads to the primary memory then the NameNode main memory limitation arises again. The focus of this research is to reduce the namespace problem of main memory and to make the system dynamically scalable. A new Metadata Fragmentation Algorithm is proposed that separates the metadata list of NameNode dynamically. The NameNode creates Secondary Memory File in perspective of the threshold value and allocates secondary memory location based on the requirement. According to the proposed algorithm the maximum third, out of fourth of main memory is used at the secondary file caching time. The free space aids in faster operation by Dynamically Scalable NameNode approach. This proposed algorithm shows that the space utilization is increased to 17% and time utilization is increased to 0.0005% with the comparison of the existing fragmentation algorithm.

Copyright © 2019 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Tumpa Rani Shaha,

Department of Computer Science and Engineering,

Dhaka University of Engineering and Technology, Gazipur, Bangladesh.

Email: tumpa.cse49@gmail.com

1. INTRODUCTION

In this modern age, it has become the main concern to handle the data that is being generated every day. Approximately 25 quintillion bytes of data are created every day and 90% of the data has been created in the last two years. This data are being generated from everywhere like sensors for gathering climate information, social media sites, transaction records, satellites etc. These data sets are immensely unstructured and as a result to process and estimate these big data is a great concern. As the data size has increased extremely RDBMS has found it challenging. More ever as these data sets are semi-structured and unstructured RDBMS cannot categorize as they are designed to handle structured data. This problem requires a database management system that is capable of analyze these data in an efficient and convenient way.

Apache Hadoop is such kind of DBMS for handling semi-structured and unstructured data that provides an open source, distributed database processing platform across several thousand nodes. More ever it is high speed and has greater tolerance to fault along with cost efficiency as it stocks data in small amount via multiple servers.

File system metadata and application data are stored separately in the existing Hadoop Distributed File System. NameNode contains the metadata of the system and DataNode contain application data. Per cluster about tens of thousands of clients can access the Hadoop storage at a time. DataNode store the block of data in their local file system and NameNode store metadata of all the DataNode in their local file system. So if we try to extend the network or try to add new DataNode then because of NameNode main memory limitation we can't extend the network. This namespace limitation is one of the important problems of existing Hadoop Distributed File System.

This proposed Dynamically Scalable NameNode (DSN) approach introducing a Metadaya Fragmentation Algorithm (MFA) to fragment the metadata frequently and increase the namespace capacity dynamically by making the interaction between main memory and secondary memory of NameNode.

2. LITERATURE REVIEW

In the field of modern technology, the use of Hadoop for handling big data has become an active area of research. Several approaches have been suggested on this field.

In [1], they developed the Hadoop Distributed File System on behalf of Yahoo. They have explained about the Hadoop architecture and showed the result for handling 25 Petabyte of data at Yahoo.

A classification based metadata management system is proposed in [2]. They focused on reducing the bottleneck of the NameNode main memory. They fragment the metadata of NameNode based on the importance factor. They have calculated three (High, Medium and Low) types of importance factor (If). Hash table is used to represent high If, a tree map is used to represent medium If and sequence files are used to represent low If.

A cached approach is proposed in [3] for addressing NameNode scalability in HDFS. Their main focus was to enhance the existing architecture. They fragment the metadata depends on the last access time and moved the least frequently used data to cache. They were able to remove 250MB of data from RAM. But for data searching when the requested data not found in main memory then the secondary data will be loaded again on the RAM. So when the secondary data reload to the primary memory the issue of NameNode main memory limitation arises again.

In paper [4], they analyze the requirement like hardware, software, network environment for improving the performance of cloud computing. They developed a cache system in layered passion where the system has a client library and multiple cache services. Client library can access the files from the shared memory. This distributed cache system can manipulate large number of files with a millisecond level in highly concurrent environment.

In [5], they developed a mechanism to improved Hadoop performance using metadata for handling big data. By assigning jobs to the DataNode, H2Hadoop was extended the ability of NameNode. They were successful for reducing CPU time and number of need operation.

In [6-8], they proposed a system for improving metadata management in HDFS for small files. They focused on the small files in the main memory and provide archival methods for those small files.

Distributed metadata management scheme is proposed in [9]. They proposed a system for distributed metadata management scheme in HDFS to improve the HDFS efficiency.

In [10], the namespace is departed into several fragments. Replicas of each fragment are dispersed among the NN. More time is needed for metadata searching with synchronization because the fragmented namespaces are distributed among different NN

In [11], they proposed a Dynamic Directory Partitioning (DDP) technique where they allowing directory metadata and file metadata in a diverse way. They improved the performance on scalability and adaptability.

An efficient metadata management system is proposed in. They proposed directory level based metadata management which is more efficient than the directory sub tree partitioning and traditional hashing technique.

3. RESEARCH METHOD

The DSN methodology has the following design principle (1) Dynamically Scalable NameNode architecture and (2) working procedure. In this section, the system architecture and the working procedure of the DSN architecture is given.

3.1. Dynamically Scalable NameNode Architecture

The Dynamically scalable NameNode architecture is shown in Figure 1. DNS has master/slave architecture. DNS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients and a number of DataNode, usually one per node in the cluster. In overall, the DSN system consists of one NameNode, a group of DataNode, clients, main memory and secondary memory concept which is discussed in this section.

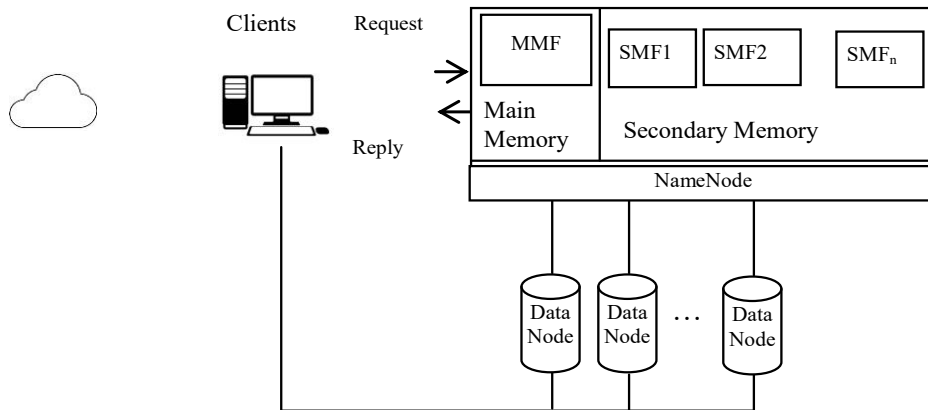


Figure 1. Dynamically Scalable NameNode Architecture

3.2. NameNode

The focal point of HDFS is NameNode. It keeps the track where the file data is kept over the cluster. The directory tree of all file in the system also kept here. When the clients wish to locate a file or they need to add/copy/delete/move a file then client applications send a request to the NameNode. The NameNode replies with corresponding DataNode address.

3.3. Main Memory

Normally the namespace of the Hadoop system is stored in NameNode main memory. In this proposed architecture we introduce Main Memory File (MMF) concept, which stores the high priority metadata of the system.

3.4. Secondary Memory

In this proposed architecture we introduced the secondary memory concept. The fragmented low priority metadata will store in the secondary memory. A lot of files can store in the secondary memory according to the proposed algorithm which is discussed in working procedure section.

3.5. DataNode

DataNode cache the data in the HDFS. DataNode talks to the NameNode to perform modifications of the data commanded by the NameNode and response to the NameNode after a fixed time interval continuously with a list of a chunk that they are storing for file system activity. Clients system can communicate to the DataNode directly if the NameNode has assigned the address of the DataNode.

3.6. Clients

Clients of the proposed system can request to the NameNode for any particular file. NameNode will reply with the address of the requested DataNode to the clients. Then clients directly communicate with the DataNode for reading or writing operation.

3.7. Metadata

HDFS metadata is divided into two categories of files named fsimage and edits log. The complete state of the file processing system at a point in time is content by the fsimage file. A unique increasing transaction id is assigned in every modification of file system. After all modification to that id fsimage files represents the file system state.

3.8. Working Procedure

In this DSN architecture when Hadoop client request to NameNode, it firstly check the available space of MMF (Main Memory Metadata File). If so then the new file is created. And default priority 1 (Lowest Priority) is set for the newly created file. But if there is no available space in MMF then least priority metadata will be moved to SMF (Secondary Memory Metadata File) following the proposed Metadata Fragmentation Algorithm (MFA). That is a priority based dynamic metadata classifier is proposed for the main memory utilization. For assigning priority let us assume the following parameters

T_d =Fixed Time Interval
 H =Number of Hits during T_d
 MMS =Main Memory Size
 M_{th} =Main Memory Threshold
 S_{th} =Secondary Memory Threshold
 MMF =Main Memory Metadata File
 SMF =Secondary Memory Metadata File
 S = Size of each metadata
 x = Number of metadata file for $M_{th} = (MMS/2)/S$
 y = Number of metadata file for $S_{th} = (MMS/4)/S$

Generally, the full fsimage file is stored in the main memory of NameNode. To fragment the fsimage file threshold value (M_{th}) is calculated by $(MMS)/2$. That is half of the main memory size is the threshold for MMF. Secondary Memory Threshold (S_{th}) value is calculated by $(MMS)/4$. So x is the number of metadata file that can be stored on M_{th} and y is the number of metadata file that can be stored on S_{th} . Figure 2 shows the metadata fragmentation algorithm.

1. If $MMF > M_{th}$ then
2. Calculate new Priority value (P)= Average (Old Priority, H)
3. Sort the metadata depending on P in descending order
4. Keep high order x factor of data in MMF
5. Shift rest lowest data to SMF [$i=1 \dots n$]
6. If $SMF[i] > S_{th}$ then
7. Repeat step 2 & 3
8. Keep high order y factor of data in $SMF[i]$
9. Shift rest low factor data to $SMF[i+1]$
10. end if
11. end i

Figure 2. Metadata Fragmentation Algorithm

When the size of the metadata file exceeds the M_{th} then the fragmentation algorithm is triggered. When the threshold value exceeds, then the priority value for each metadata will be updated frequently if needed based on trigger. Newly generated priority values are sorted (higher to lower order) and metadata having higher priority will keep to the MMF. That is x number of metadata has been stored in MMF

Low priority metadata records are separated out and moved into the file created on secondary storage. As low priority metadata frequently moves to the secondary storage so the number of SMF will extend according to the size of metadata. The number of metadata has been stored in each SMF is measured by factor y and they must be stored according their higher to lower priority. Let consider the size of the main memory is 1 GB, then the threshold value (M_{th}) will be 512 MB and the size of each fragmented file in the secondary memory (S_{th}) is $1 \text{ GB}/4=256 \text{ MB}$. If we consider that size of each metadata is 1MB then MMF can contain 512 metadata which is factor x .

When the user searches any particular file, the system will search that data in the main memory first. If it is found, the file will be replied to the user with the DataNode address. But if it is not found in the main

memory then according to the priority value the requested file will be cached to the main memory from the secondary memory through page table which is shown in Figure 3.

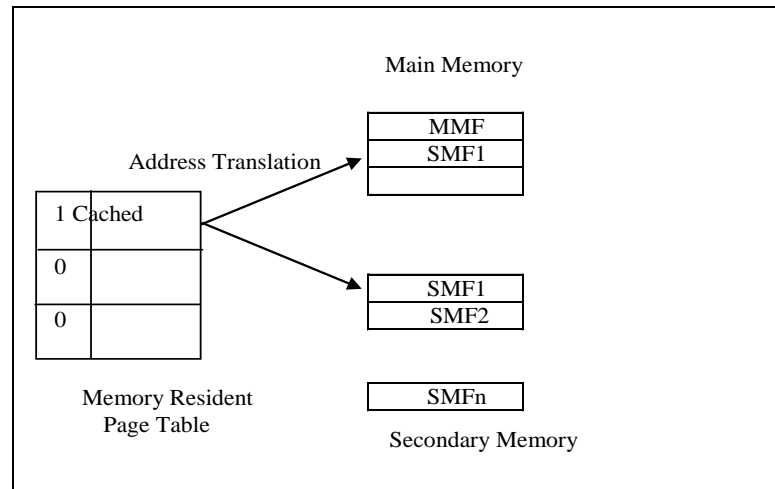


Figure 3. Secondary File Caching

4. RESULTS AND ANALYSIS

To evaluate the performance of the MFA algorithm we have conducted two kinds of test: 1. Performance on main memory usages 2. Performance on average response time. In this section we have demonstrated the performance of the DSN approach and the comparison with the existing cache approach.

4.1. Simulation Platform

We have developed the MFA and existing fragmentation algorithm using C++ language in two different computers. One of those is 4GB RAM with 2.10 GHz Core i3 processor and another one is 8GB RAM with 1.60GHz Core i5 processor.

4.2. Performance on Main Memory Usages

In this section the performances on main memory usages of DSN approach and existing cached approach in terms of size of main memory is discussed. Figure 4 shows the NameNode main memory usage comparison.

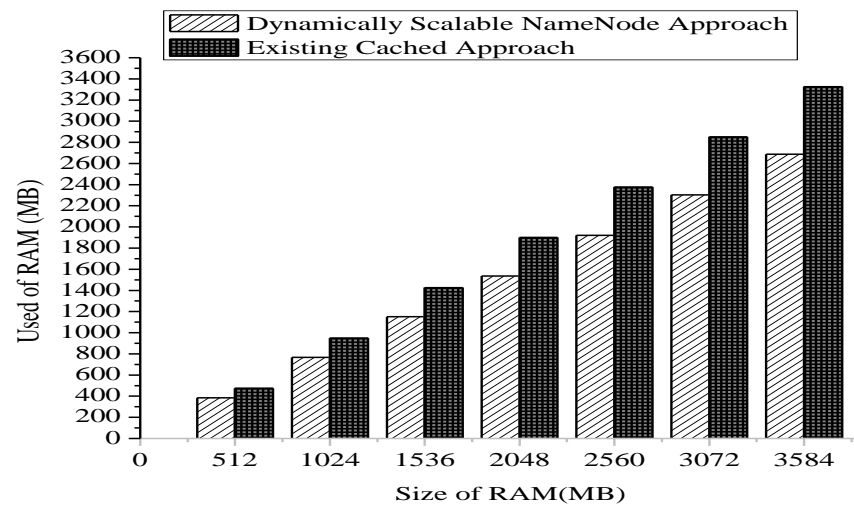


Figure 4. NameNode Main Memory Usage Comparison

According to the MFA the use of RAM for the Dynamically Scalable NameNode approach is calculated by the size of x factor, y factor and the size of each metadata. After the fragmentation of cache approach the main memory can store 700MB metadata and 250MB data in the secondary memory of 1GB RAM [3]. But in DSN system the main memory is able to hold 512MB and 256MB in secondary memory after the metadata fragmentation algorithm trigger. The Secondary Memory can store several files of size 256MB. So the storage capacity has been increased dynamically.

Existing cached approach is used 92% of RAM and the DSN algorithm required maximum 75% of main memory in worst case. So this DSN approach is utilized average 17% of main memory usage. This free space of main memory ensure the overall response time of the NameNode.

4.3. Performance on Response Time

In this section the performances on average response time of DSN approach and existing cached approach is discussed. For analyzing the average response time of the NameNode, we have made a setup to simulate of proposed and existing MFA algorithm in two well configured computers. Setup-1: 4GB RAM with 2.10 GHz Core i3 processor and Setup -2: 8GB RAM with 1.60GHz Core i5 processor. Figure 5 and Figure 6 show the average response time analysis of setup-1 and setup-2.

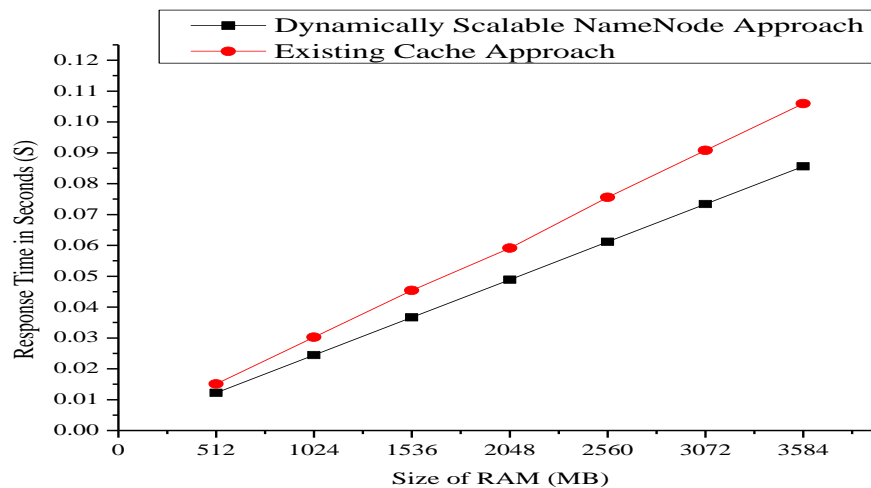


Figure 5. Average Response Time Analysis of Setup-1

Let consider the size of each metadata (S) is 1MB. Then the MMF will contain 512 metadata which is factor X and each SMF can contain 256 metadata which is factor Y.

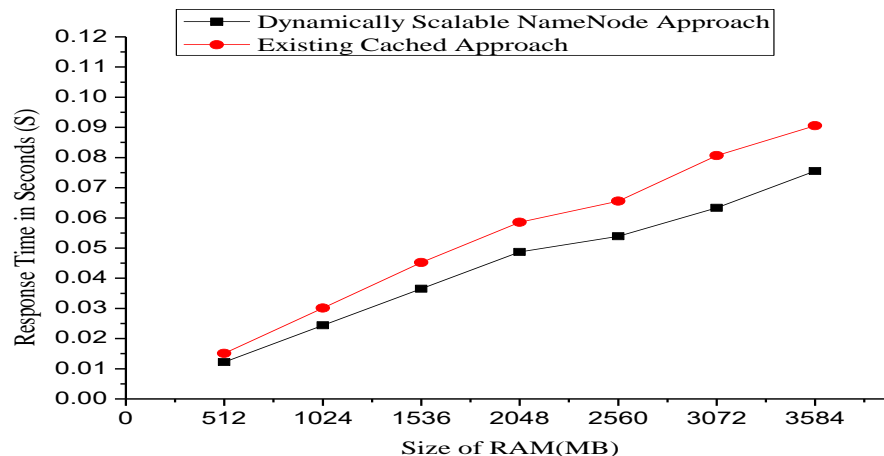


Figure 6. Average Response Time Analysis of setup-2

In simulation, the configuration of setup-2 is higher than setup-1. So we can see that the average response time in setup-2 is less than setup -1. So here it is proved that this proposed system will provide better response time in high configured system.

5. CONCLUSION

In this proposed work we have experimented with a large amount of data efficiently thus the time requirements has been reduced and memory utilization is increased. The proposed system is more efficient than the existing cached approach that is proved by our performance evaluation section. By implementing the concept of secondary storage it has been shown that amount of metadata will not be so high that the NameNode will be irresponsive due to the excessive amount of data. At the same time the client request can be handled more frequently than the existing system. In future work we would like to introduce several parameters and be proved mathematically so that the system can work more efficiently and can be implemented in real time system.

REFERENCES

[1] K. Shvachko, H. Kuang, S. Radia, and R.Chansler, "The Hadoop Distributed File System," *IEEE 26th Symposium*, pp. 1-10, May, 2010.

[2] A.Chandrasekar, K.Chandrasekar, H. Ramasatagopan, and J.Balasubramaniyan, "Classification based Metadata Management for HDFS," *IEEE 14th International Conference on High Performance Computing and Communications, 2012*

[3] Zhang, G. Wu, X. Hu, and X. Wu, "A Distributed Cache for Hadoop Distributed File System in Real-time Cloud Services," *13th International Conference on Grid Computing*, pp. 12-21, 2012.

[4] H. Alshammari, J.Lee, and H. Bajwa, "H2Hadoop: Improving Hadoop Performance using the Metadata of related jobs," *IEEE Transactions on Cloud Computing*, PP. 1-1, 2015.

[5] G. Mackey, S. Sehrish, and J. Wang, "Improving Metadata Management for Small Files in HDFS," *IEEE International Conference on Cluster Computing and Workshops*, 2009.

[6] S. Bende, R. Shedge, "Dealing with Small Files Problem in Hadoop Distributed File System,"*7th International Conference on Communication, Computing and Virtualization, 2016.*

[7] Dr. Raut, S., Phakade, P., "An Innovative Strategy for Improved Processing of Small Files in Hadoop" *International Journal of Application or Innovation in Engineering and Management*, 3, pp. 278-280, July, 2014.



[8] M. Varade, V.Jethani, "Distributed Metadata Management Scheme in HDFS," *International Journal of Scientific and Research Publications*, 3, 2013.

[9] Y. KIM, T. Araragi, J. Nakamura, T. Masuzawa, "A Distributed NameNode Cluster for a Highly- Available Hadoop Distributed File System," *IEEE 33rd International Symposium on Reliable Distributed System*, 2014.

[10] Y. Fu, N. Xiao, and E. Zhou, "A Novel Dynamic Metadata Management Scheme for Large Distributed Storage Systems," *10th IEEE International Conference on High Performance Computing and Communications*, 2008.

[11] L. Ran, and H. Jin, "An Efficient Metadata Management Method in Large Distributed Storage Systems," *International Conference on Human-centric Computing and Embedded and Multimedia Computing*, pp. 375-383, 2011.

BIOGRAPHIES OF AUTHORS

	<p>Tumpa Rani Shaha obtained her B.Sc and M.Sc degrees under department of Computer Science and Engineering from Dhaka University of Engineering and Technology (DUET), Gazipur, Bangladesh. Tumpa Rani shaha research interests are on Data Mining, Big Data, Hadoop Distributed File System, Machine Learning and Deep Learning. Currentlt she is a faculty member at department of Computer Science and Engineering, Daffodil International University.</p>
	<p>Md. Nasim Akhtar received the M.Eng and Ph.D degrees from National Technical University of Ukraine, Kiev, Ukraine and Moscow State Academy of Fine Chemical Technology, Russia, in 1998 and 2010, respectively. Currently, he is a Professor in the Department of Computer Science and Engineering, Dhaka University of Engineering and Technology (DUET), Gazipur, Bangladesh. His research interests include Distributed Data Warehouse System On Large Clusters, Digital Image Processing and Water Marking, Peer to Peer Networking, Cloud Computing, Operating System. He has presented papers at conferences both home and abroad, published articles and papers in various journals.</p>

A noble approach to develop dynamically scalable namenode in hadoop distributed... (Tumpa Rani Shaha)

	<p>Md. Zakir Hossain received the B.Sc Engineering degree in Computer Science and Engineering Department from Dhaka University of Engineering and Technology (DUET), Gazipur, Bangladesh, in 2015 and he is currently pursuing the M.Sc Engineering degree in Computer Science and Engineering Department in Dhaka University of Engineering and Technology (DUET), Gazipur. His research interest includes Data Mining, Big Data, AI, Machine Learning, Cloud Computing, Software Engineering, Computer Network, IoT. He has presented papers at conferences both home and abroad</p>
	<p>Ms. Fatema Tuj Joohora is the Lecturer of a reputed private university in Bangladesh. She has received her B.Sc and M.Sc degree in Information Technology from Jahangirnagar University (JU). Her recent publications include "An Efficient Approach of Training Artificial Neural Network to Recognize Bengali Hand Sign" (2016). Her research interest includes Data Mining, Artificial Neural Network, Image processing, and cloud computing.</p>
	<p>Mostafijur Rahman completed his BSc in Computer Science from National University of Bangladesh (2003). He Pursued his MSc (2009) and PhD (2017) in Computer Engineering, from UNIMAP, Malaysia. He worked as Lecturer since 2009 to September, 2017 for School of Computer and Communication Engineering in UNIMAP. Currently he is serving as Assistant Professor in the Department of Software Engineering at Daffodil International University (DIU), Bangladesh. His research interest in Software Testing, Multimedia and Creativity in Medical Science, Computer Security, Cloud Computing, Algorithm Optimization, Parallel and Distributed System, Device Driver for GNU/Linux based embedded OS. He has presented papers at conferences both home and abroad, published articles and papers in various journals.</p>
	<p>R. Badlishah Ahmad received the M.Sc Engineering and Ph.D degrees from University of Strathclyde, UK in 1995 and 2000, respectively. Currently, he is a Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin (UniSZA). His research interests in Computer and Telecommunication Network Modelling include WSN and Optical Network using discrete event simulators (OMNeT++), Optical Networking and Embedded System based on GNU/Linux. He has presented papers at conferences both home and abroad, published articles and papers in various journals.</p>