Performance Enhancement Evaluation in Database Decompression Using HIRAC Algorithm

Muthukumar Murugesan[1], T. Ravichandran[2]

[1] Research Scholar, Department of Computer Science, Karpagam University, Coimbatore, Tamilnadu-641021, India,

[2] Principal, Hindusthan Institute of Technology, Coimbatore, Tamilnadu-641032, India

Abstract

Storage space ability and computational authority increases, processing and analyzing large volumes of database systems acts as a significant role in several areas of scientific research. Database compression and decompression is a susceptible problem in the database framework. Disk storage systems can often be the most expensive components of a database solution, even a small reduction in the storage subsystem can result in substantial cost savings for the entire database solution. When you have large amounts of data, the cost of the storage subsystem can easily exceed the cost of your data server.

In this paper we proposed a new algorithm for database compression and decompression called HIRAC algorithm. If we compress the database, there is chance of hurting the data in the compressed database systems. Projected algorithm will manipulates each row in compressed database inside the file to extract the original database without losing any data.

Keywords:database compression, database decompression, real-time database, compression performance, database performance

1. Introduction

Data stored in databases keep growing as a result of businesses requirements for more information. A big portion of the cost of keeping large amounts of data is in the cost of disk systems, and the resources utilized in managing that data. Reducing the amount of memory consumed, and freeing it up for other database or system operations can further improve the database performance for queries and other operations.

Generally, in day to day basis lakhs and millions of transactions will be used to process and store the data into real time database system. Global providers like insurance, banking, finance, railway system and others are having a massive number of transactions on daily basis. All these transactions should be handled on the database for further reference. Such databases' size would be raised as MBs and GBs of data through daily actions. There is a huge problem for implementing the backup of data without losing. Global providers are used for processing the backup of the database several times in a day. This research addresses the difficulties of the problem and provides the solution using compressing, decompressing and taking the backup of the database in multiple devices.

Extensive experiments were conducted to estimate and enhance the performance of the proposed **HIRAC**(Hierarchical Iterative Row-Attribute Compression) algorithm for real time database systems. It gives you the ability to transparently compress data on disk in order to decrease disk space and storage infrastructure requirements. Compressed rows are smaller, not only you need fewer disks, but your overall system performance may be improved. Accessing data from the disk is the slowest database operation. By storing compressed data on disk, fewer I/O operations need to be performed to retrieve or store the same amount of data. Therefore, for disk I/O-bound workloads, the query processing time can be noticeably improved.

In a compressed database system, data are accumulated in compressed layout on disk and are decompressed through query processing. Compression develops query presentation for two reasons. First, it saves disk and memory bandwidth because a compressed disk page or cache line packs more data than an uncompressed disk page or cache line. Second, if the compressed data can be kept compressed during query processing, compression reduces disk and memory latency by fitting more compressed data in memory and cache. Nevertheless, compression acquires CPU overhead for the compression and decompression of data.

To enhance our research in this work, we present an efficient approach for decompressing the compressed database to retrieve the information based on users' requests. Before performing the decompression, it is necessary to identify and find the location of database in the parallel multi-storage repository which consists of set of compressed database based on the users' requisite query information. After identifying the exact data location in the compressed database, extract it from the multi storage parallel repository. Decompress the compressed database to retrieve the user needed information from the compressed database. By doing like this, the user database transaction processing achieves high in performance and retrieves the information in fewer intervals of time.

2. Related Work

Several techniques have been presented to compress the database. Among which, the recently used technique is attribute level compression methods. This type of compression methods needs more time to perform large number of transactions. To overcome the issue, our first work describes the process of compressing the real time database systems using HIRAC algorithm [1]. It excellently compressed the database systems in a less interval of time. Our second work presented multi storage parallel process based on compressed database [2]. Using optimal multi storage parallel backup, we efficiently achieved the parallel multi storage backup compressed data for real time database systems. Here we focused on the data storage in real time database systems to store the data more effectively and to reduce the storage requirements. Our third work outlined the techniques to analyse the compression performance for real time database systems [3]. This technique not only reduces space requirements on disk and I/O performance when measured in records per time for permanent and temporary data, they also reduce the requirements of memory, thus reducing the number of buffer faults resulting in I/O.

Many compression schemes are evaluated at present. A greatly proficient and new dictionary-based lossless compression technique is employed and compared beside a prior execution for a reconfigurable scheme. The paper [3] appears at numerous incompatible devise parameters, for instance the compression ratio, latency, silicon area, and power utilization. In [4], we are apprehensive with an efficient construction for chronological data set that does not provide on-line query presentation and is particularly considered for very huge sensor network database. The modeled data are compressed by adapting numerous candidate strategies counting dictionary-base constrict and lossless vector quantization.

The paper [5] originates diverse compression strategies for data mounded up in row inclined besides column-oriented databases. Obviously, data in columns is further compressible than data in rows. Compressions algorithms [6] realize improved on data with little in series entropy (high data value location) i.e are employed for optimization position. The structure of data firmness in the real-time database has been originated beside with the sequential data aspect, the merits and demerits of data compression algorithm [7]. A database compression algorithm [8] realizes well as favoring at characteristic stage with no requirement of the decompression of a better unit.

2.1 Database Compression using HIRAC Algorithm

Compression can significantly reduce disk space storage, increase memory utilization, and better I/O performance; there is a risk of CPU degradation. It optimizes storage of multiple rows in a page, by minimizing the data redundancy. Therefore, one must take into account and consider the trade-off between compression and performance.

The process of real time database compression using HIRAC algorithm is processed under below steps.

- Analyzing the entire database
- Implementing the HIRAC algorithm
- Copying the data from the database files into backup devices at parallel.

Database analysis is apprehensive with the environment and use of data. It engages the classification of the data elements which are desired to sustain the data dealing out system of the organization, the introduction of these elements into rational groups and the description of the relations among the resulting groups. To compress database, an algorithm is presented called HIRAC algorithm which iteratively enhances the collection of selected representative rows. From one step to the next, new representative rows may be chosen, and old ones discarded. Since the endeavor of a compression algorithm is to decrease the storage constraint for a database, HIRAC algorithm utilize this as an optimization principle and guarantee that only patterns which progress the compression are established in each step of its iterations. Though the optimization problem is hard in an existing compression algorithms case as well, the heuristic used in this simple. Using several backup devices used for backup of the compressed database utilized in parallel in terms of enhancing the speed of backup and restores operations.

2.2 Multi Storage Parallel Backup

For backup and restore of data stored on the storage system, a standard method for backup requires the host systems to extract the data from the database to the data storage systems. This method would be incredibly slow and it requires typing up the host systems' time in the form of database access operations and data processing. A better solution is known as direct connect. A high speed direct connection is required between the storage systems and backup management systems thereby allowing fast transfer of data directly to the backup management systems without the intervention of host systems.

The Multi Storage Parallel Backup Compression consists of the below five steps:

- Compressed database using HIRAC algorithms
- Maintain the internal synchronization point
- Creating the transaction log files if they do not exist
- Copying the backup/compress data to the backup devices
- Copying the transaction log with the same backup devices

The backup compressed data is concerned with backup and restore of 1) data files as seen from the database client 2) taking backup copy of the control file system 3) Archived log files. A parallel multi storage database system has several restrictions on a backup system for a data storage system. All data files in the parallel multi storage devices must reside on contiguous portions of storage devices and visible to all parallel server database devices. An optimal approach is used here for storing the items in the compressed database. A compressed database has several rows and columns, but when it stored under different multiple storage devices with different row or column attributes. Through the control and log files, the transaction carried over the compressed database is illustrated and consumes less disk storage requirements since it is compressed efficiently.

In this work, we present an efficient approach for decompressing the compressed database to retrieve the information based on users' requests. Before performing the decompression, it is necessary to identify and find the location of database in the parallel multi storage repository which consists of set of compressed database based on the users' requisite query information.

3. Database Decompression Using HIRAC Algorithm

The proposed work is efficiently designed to perform the decompression process for compressed database which is stored under parallel multi storage repositories. The proposed Performance Enhancement Evaluation in Database Decompression (PEEDD) using HIRAC Algorithm is processed under two different phases. The first phase is to identify the location of data in the compressed database which is stored under multi storage repositories. The second phase describes the process of decompressing the respective compressed database.

Reading and writing compressed data consumes more CPU resources than reading and writing uncompressed data. Consider an uncompressed table with a row size of 800 bytes. Five of this table's rows would fit in a 4K data page/block. Now after data is compressed, assume that the compression routine achieves 30% compression on average. In that case, the 800-byte row will consume only 560 bytes (800 x 0.30 = 560). After compressing the data, seven rows will fit on a 4K page. Because I/O occurs at the page level, a single I/O will retrieve more data, which will optimize the performance of sequential data scans and increase the likelihood of data residing in the cache because more rows fit on a physical page.

The architecture diagram of the proposed Performance Enhancement Evaluation in Database Decompression using HIRAC Algorithm is shown in fig 1.

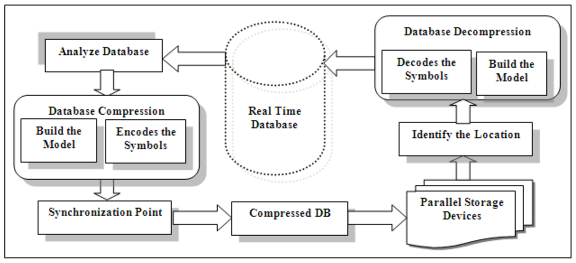


Fig 1: Structure of Database Decompression

From the fig (fig 1), it is being noted that the process of decompression is done only after performing the two major processes. The first process is to compress the database systems using HIRAC algorithm. The second process is to store the compressed database systems in parallel multi storage by taking backup of those compressed database systems. If the user wants to retrieve the items from the database, then it is necessary to decompress the compressed database which is stored under parallel multi storage backup repositories.

The following subsection describes the decompression process for real time database systems.

3.1 Decompressing the Compressed Database

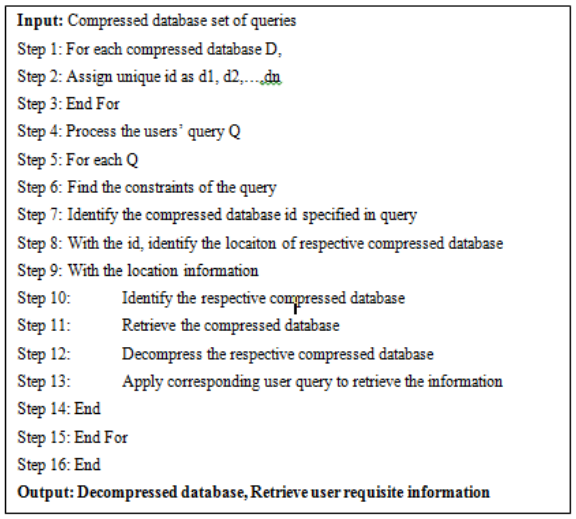
The process of decompressing the compressed database which is stored in multi storage parallel backup repositories involves in below four major steps.

1. Identify the location of database in parallel compressed database repositories
2. Extract the compressed database information from repositories
3. Organize the locations into common place
4. Implement the decompression process to decompress the respective compressed database.

The location of database in the storage repository is identified for accessing the users' query. Decompression process will take place based on the outcome.

Identify the Location
Extract the Information
Implement the Process
Organize the Locations

Fig 2: Elements of Database Decompression



```
Input: Compressed database set of queries
Step 1: For each compressed database D,
Step 2: Assign unique id as d1, d2,…,dn
Step 3: End For
Step 4: Process the users' query Q
Step 5: For each Q
Step 6: Find the constraints of the query
Step 7: Identify the compressed database id specified in query
Step 8: With the id, identify the locaiton of respective compressed database
Step 9: With the location information
Step 10:        Identify the respective compressed database
Step 11:        Retrieve the compressed database
Step 12:        Decompress the respective compressed database
Step 13:        Apply corresponding user query to retrieve the information
Step 14: End
Step 15: End For
Step 16: End
Output: Decompressed database, Retrieve user requisite information
```

Fig 3: Algorithm of the Proposed PEEDD

After compressing the database, it is stored under different storage repositories in parallel. Once the compressed database is stored under different locations, each compressed database is specified with some unique ID. For instance, let the set of compressed database be D. Assign unique id to each compressed database as d1, d2,…, dn. With the unique id, the compressed database is retrieved from the storage repositories. A user will send a query to access the compressed database for user retrieval process. A user query contains an information like compressed database id with retrieval information. With the compressed database id, the respective database is retrieved from the storage repositories and processed for further retrieval schemes.

Once the compressed database is identified, retrieve the specified database from storage repositories. Then implement the decompression algorithm which is the reverse process of compression algorithm. At first, with the given attribute id and constraints, respective information is retrieved and accessed. The process of the proposed PEEDD using HIRAC algorithm is shown in fig 3.

The above figure (fig 3) describes the process of the retrieving the user requisite information from the compressed database which is stored under different storage repositories. With the specific id, identify the location of the compressed database in the storage repositories. With the location information, retrieve the respective compressed database and decompress it. The decompression is based on the reverse process of the compression algorithm. After decompressed the database, the users' query is processed and used to retrieve the information from the real time database environments.

4. Experimental Evaluation

The proposed Performance Enhancement Evaluation in Database Decompression Using HIRAC Algorithm is implemented in SQL Server and Java. We used a real time 1GB sample database for a trialing to examine the efficiency of the decompression performance. In this work, we have seen how the compressed database systems are decompressed with the set of representative rows and columns for real time environments which are stored in multi storage parallel devices. Here we have outlined a series of performance experiments in order to examine the efficiency of the proposed HIRAC algorithm. The experiments were tested using the below configuration.

System Component	Description
Processors	Intel® Core 2 Duo CPU P-IV 2 GHz Processer
Memory	3 GB memory
Operating System	Microsoft Windows XP Professional Version 2002 Service Pack 3
Database Manager	Microsoft SQL Server 2008 Enterprise Edition (64 bit)
Database Size	1 GB

Table 1: System Components

Over the last decades, enhancements in CPU speed have provided developments in disk processing rates by commands of amount. For instance, the CPU speed has been enhanced by on one thousand in 15 years, as disk bandwidth and latency have been enhanced by simply forty fold and three fold correspondingly. This hardware drift enforces a rigorous confronts to the presentation of a database system as database applications are frequently memory-bound or disk-bound.

The proposed HIRAC based decompression model for real time environment is efficiently designed for decompressing the compressed database systems stored in multiple storage systems.
The performance of the PEEDD using HIRAC algorithm is measured in terms of:

1. Transaction time
2. Number of users request
3. Storage space
4. Time consumption for decompression

Transaction Time denotes the time period during which a database fact is/was stored in the database.

$$\text{TPS (Transaction per Second)} = \frac{\text{Total Response Time}}{\text{Expected Time to Complete the Transaction}}$$

Storage Space denotes the area in which the parallel storage repositories provide space to store compressed database in it.

Time Consumption for Decompression denotes the time taken to decompress the compressed database for user information retrieval purpose.

5. Results and Discussion

In this work, we have seen how the compressed database is decompressed with size of 1GBs which is efficiently stored under multi storage parallel devices. The new proposed approach is efficiently done by using HIRAC algorithm.

There are two significant troubles for reconstructing a compressed database system:

- How a compressed database compresses the data
- How query has been executed to reconstructing the database.

These two troubles are inconsequential as of the transaction of disk and memory access provides decompression overhead. For instance, to decide how to constrict the data, the classification must suspiciously weigh the I/O savings a compression technique might take with its possible decompression overhead. A decompression technique producing considerable I/O savings but effecting exorbitant decompression transparency would be unsuitable since the complete performance might not be enhanced. Likewise, through query processing, the method desires to choose when to decompress the data and reside compressed in the production of the present process. Maintaining data compressed in the outcome of the present process might keep more disks I/O since the compressed outcome will be slighter than the uncompressed outcome. Nevertheless, this might acquire more decompression transparency for the later operations.

We used a real time database for an experimentation to examine the efficiency of the decompression of compressed database system. The below table and graph described the proposed PEEDD using HIRAC algorithm with an appropriate results.

No. of Database	Storage Space (MB)	
	Proposed PMBDC	Existing Attribute Level Method
2	9.2	16.3
4	12.5	19.2
6	18.9	23.5
8	21.3	25.2
10	22	29.4
12	24.2	32
14	24.8	35.4

Table 2: No. of Database vs. Storage Space

The above table describes the requirement of storage space needed to store the compressed database in a reliable manner. The storage space of the proposed PEEDD system using HIRAC algorithm is compared with existing attribute level compression methods.

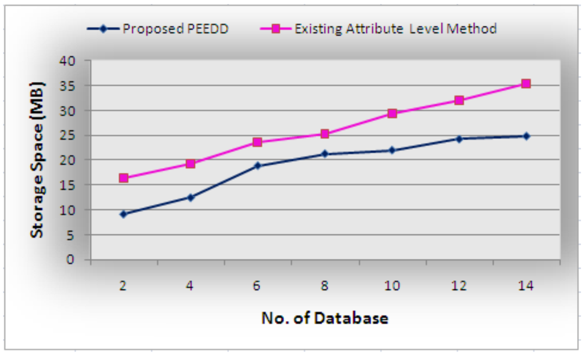


Fig 4: No. of Database vs. Storage Space

Fig 4 describes the requirement of storage space needed to store the compressed database in a reliable manner. Storage space denotes the storage area in which the parallel multi storage repositories provide space to store compressed database in it. The storage space is measured in terms of megabyte (MB). In the proposed PEEDD approach, the process of storage is done by compressing the database using HIRAC algorithm. After performing the compression, the compressed database is multi storage repositories in parallel. Since the database is compressed with respect rows and columns, the storage space required to store the database is less in the proposed PEEDD system. Compared to existing attribute level methods which does not support minimal storage space, the proposed PEEDD suits best to achieve the parallel multi-storage area. The variance in the storage space needed to store the compressed database is 50-60% low in the proposed Performance Enhancement Evaluation in Database Decompression Using HIRAC Algorithm.

No. of Compressed Database	Time Consumption for Decompression (ms)	
	Proposed PMBDC	Existing Attribute Level Method
1	2.4	5.8
2	3.8	9.7
3	5.8	13.5
4	6.7	17.2
5	9.3	20
6	10.2	21.5
7	12.1	23

Table 3: No. of Compressed Database vs. Time Consumption for Decompression

The above table (table 3) describes the time taken to decompress the compressed database based on the retrieved set of compressed database systems. The time consumption for decompression of the proposed parallel multi storage backup decompression for real time database systems is compared with with existing attribute-level compression methods.

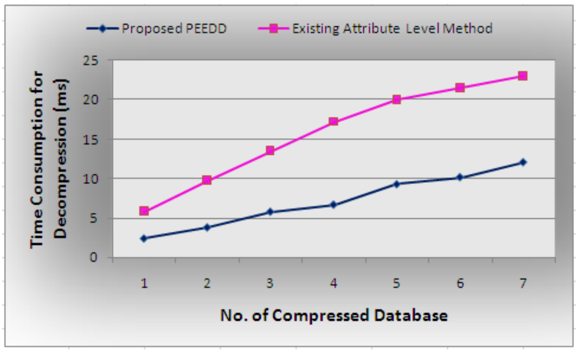


Fig 5: No. of Compressed Database vs. Time Consumption for Decompression

Fig 5 describes the time taken to decompress the compressed database based on the retrieved set of compressed database systems. Time consumption for decompression denotes the time taken to decompress the compressed database for user information retrieval purpose. In the proposed PEEDD, the decompression takes place in minimal interval of time because the exact database is retrieved from the storage repository based on the user query. So, the time take to decompress the database is also being low since the location of database is easily identified. Compared to existing attribute level methods which does not support minimal storage space, the proposed PEEDD system suits best to achieve the decompression process for user retrieval purpose. The variance in the decompression time is 60-70% low in the proposed Performance Enhancement Evaluation in Database Decompression Using HIRAC Algorithm.

The most obvious advantage of database decompression is that of reducing the time consumption. The decompression performance has been tested in this paper, which is compared with the existing attribute level method. The tested data is from simulation data is based on historical data characteristics. Testing includes compressed database file size, decompression timing on existing and proposed decompression approach. The below table described the time consumed for decompression of the proposed decompression method below are the tested results for the same.

Compressed File Size (MB)	Decompression Time (mm:ss)	
	Proposed PEEDD	Existing Attribute Level Method
1024	1:16	1:58
510	0:42	1:03
340	0:33	0:52
250	0.21	0:37

Table 4 Compressed DB Size vs. Time Consume

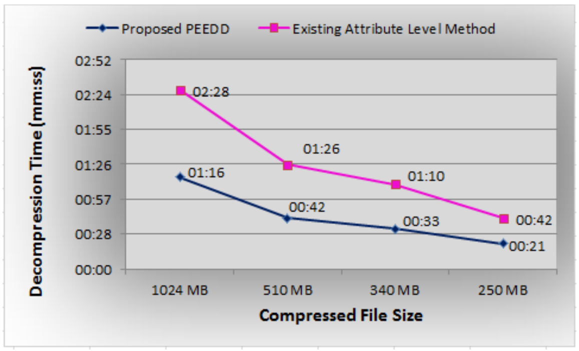


Fig 6 Compressed DB Size vs. Decompression Time

The result of decompression performance test is shown in table 4 and Fig 6. The proposed decompression algorithm achieves good performance. HIRAC compression algorithm can satisfy real-time performance demand of compressed database. The time is measured in terms of minutes and seconds (mm:ss). Table 4 and Fig 6 is depicts a performance test results for existing and proposed algorithms. The decompression time of HI"RAC algorithm is very short. As shown in above table, the new decompression algorithm is superior in all respects to the general decompression algorithm in the real-time database.

1. The decompression rate is nearly doubled than before. Comparing with previous data decompression algorithm, our proposed algorithm provides 50% ~ 60% good results.
2. The decompression time is greatly lower than before. The existing decompression algorithm is adopted in the time code labels and the quality code of new decompression algorithm, so the decompression time is greatly reduced.
3. The decompression time was cut down. So it is better to meet the requirements of real-time database.

Finally it is being observed that the present work provides an efficient approach for decompressing the compressed database which is stored under different storage repositories in parallel. The decompression process is reliably done based on users' query and achieves high efficiency in its performance.

6. Conclusion

In this work, we efficiently decompress the compressed database stored in multi storage system simultaneously based on users' query for real time environment. The compression is done using HIRAC algorithm which followed iterative steps to achieve the better compression rate. Then the compressed data is to be stored under different devices simultaneously with an optimal approach. Then for decompression, it first identifies the location of the database in storage repositories and then extracted it for decompression. The merits of using the proposed PEEDD are 1) It increases the user database transaction time for retrieval process 2) It efficiently achive the fast retrieval of information from database 3) Greatly reduced the decompression time by identifying the location of database in storage repositories. Experimental results have shown that the proposed Performance Enhancement Evaluation in Database Decompression Using HIRAC Algorithm provides 70% transaction time and achieves 75% in efficiency compared to an existing attribute level method for storing compressed data. As an overall result HIRAC algorithm enables more granular, enhances the compression and decompression performance.

References

[1] M.Muthukumar and T.Ravichandran, Optimizing and enhancing parallel multi storage backup compression for real-time database systems, IJECTA, ISSN: 2229-6093, Volume3 Issues4, Pages 1406-1417, July 2012.
[2] M.Muthukumar and T.Ravichandran, Optimizing multi storage parallel backup for real time database systems, IJESAT, ISSN: 2250-3676: Volume2 Issues5, Pages 1515-1521, Sep 2012.
[3]M.Muthukumar and T.Ravichandran, Analyzing compression performance for real time database systems, International Conference on Advanced Computer Engineering and Applications (ICACEA), 2012.
[4] Aslam, N. et. Al., "Code Compression and Decompression for Coarse-Grain Reconfigurable Architectures", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008
[5] Pei-Lun Suei et. Al., "Data compression and query for large scale sensor data on COTS DBMS" IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2010
[6] Gupta, A. et. Al., "Nonlinear Sparse-Graph Codes for Lossy Compression", Information Theory, IEEE Transactions on, Volume: 55, Issue: 5, May 2009
[7] Veluchandhar et. Al., "A backup mechanism with concurrency control for multilevel secure distributed database systems", Third International Conference on Digital Information Management, 2008.ICDIM 2008.
[8] Aghav, S. et. Al., "Database compression techniques for performance optimization", 2010 2nd International Conference on Computer Engineering and Technology (ICCET).
[9] Changing Zhen et. Al., "Design and Realization of Data Compression in Real-TimeDatabase", International Conference on Computational Intelligence and Software Engineering, 2009.CiSE 2009.
[10] Kuruppu, S. et. Al., "Iterative Dictionary Construction for Compression of Large DNA DataSets", IEEE/ACM Transactions on Computational Biology and Bioinformatics, feb 2012
[11] Hongyuan Ma et. Al., "Experiences with Hierarchical Storage Management Support in Blue Whale File System", 2010 International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT),
[12] Hongfei Yin et. Al., ""Verification-Based Multi-backup Firmware Architecture, an Assurance of Trusted Boot Process for the Embedded Systems", 2011 IEEE 10th International

Conference on Trust, Security and Privacy in Computing and Communications (TrustCom).
[13] SeongHoon Kim et. Al., "A configuration management system for multi-hop zigbeenetworks", ICCE '09. Digest of Technical Papers International Conference on Consumer Electronics, 2009.

First Author: Muthukumar Murugesan received the B.Sc, M.C.A and M.Phil degrees in Computer Science from the Madras University, Chennai, Tamilnadu, India, Bharathidasan University, Tamilnadu, India and Manonmaniam Sundaranar University, Tamilnadu, India in year 1997, 2000 and 2003 respectively. He was working as IT Analyst, in Department of Application and Software Development, IBM India Private Limited, Bangalore, Karnadaka, India. Currently he is a research scholar in Department of Karpagam University, Coimbatore, Tamilnadu, India. His fields of interest are database, compression, storage, and network.

Second Author: Dr.T.Ravichandran received the B.E degrees from Bharathiar University, Tamilnadu, India and M.E degrees from Madurai Kamaraj University, Tamilnadu, India in 1994 and 1997, respectively, and PhD degree from the Periyar University, Salem, India, in 2007. He is currently the Principal of Hindustan Institute of Technology, Coimbatore Tamilnadu, India. Before joining Hindustan Institute of Technology, Professor Ravichandran has been a Professor and Vice Principal in Vellalar College of Engineering & Technology, Erode, Tamilnadu, India. His research interests include theory and practical issues of building distributed systems, Internet computing and security, mobile computing, performance evaluation, and fault tolerant computing. Professor Ravichandran is a member of the IEEE, CSI and ISTE. Professor Ravichandran has published more than 120 papers in refereed international journals and refereed international conferences proceedings.