OPTIMIZING MULTI STORAGE PARALLEL BACKUP FOR REAL TIME DATABASE SYSTEMS

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

[1] *Research Scholar, Karpagam University, Coimbatore, Tamilnadu-641021, India,* **amgmuthu@yahoo.com**

[2] *Principal, Hindusthan Institute of Technology, Coimbatore, Tamilnadu-641032, India,***dr.t.ravichandran@gmail.com**

Abstract

Real time databases are processing millions and billions of records on daily basis. Processing and taking backups of those real time databases are highly expensive. Traditional backup methods are progressively more difficult to back up huge volume of data's. To overcome this issue, in previous work we addressed the difficulties of the issues and provided a solution for compressing the real time database systems using HIRAC algorithm. Present research focusing on optimal multi storage parallel backup ***(MSPB)*** *for real time database systems by using* ***Parallel Multithreaded Pipeline (PMP)*** *Algorithm to store compressed database at multiple devices in parallel***.** *PMP allows backups to be written to multiple devices in parallel. Using multiple devices can increase throughput in proportion to the number of devices used.*

Index Terms: *parallel backup, multi storage server, backup multi devices, and data storage*

---***---

1. INTRODUCTION

The demand for data storage capacity is constantly increasing. While the capacities available from data storage devices are also increasing, many applications have data storage requirements that exceed the capacity available from a single data storage device. There are two main problems that traditional backup methods suffer from:

- Capacity of the backup device
- Time taken to do the backup - also known as the "backup window"

One solution for these applications is a data storage system using an array of data storage devices. Data storage device arrays increase storage capacity by dividing the data to be stored among the devices in the array. For example, conventional systems typically divide the data by file or sector, with different files or sectors being stored in different devices in the array. While this arrangement results in improved storage capacity, the performance of the array as a whole is typically limited to the level of performance of the individual data storage devices. This limitation exists because conventional systems typically transfer the data one file at a time or one sector at a time to the individual data storage devices, which usually causes the data storage devices not receiving data to sit idle while they wait for data to be transferred to them.

Using multiple backup devices for backup and restore operations enables to use parallel I/O to increase the speed of backup and restore operations because each backup device can be written to or read from at the same time as other backup devices. For enterprises with large databases, using many backup devices can greatly reduce the time taken for backup and restore operations.

Over the previous decades, the amount of data collected in databases has developed at a very elevated rate.. Alternatively, developments in speed of RAM memories and CPUs have outpaced developments in substantial storage devices by guidelines of magnitude. This scientific development led to the employment of data firmness, dealing some implementation transparency (to squeeze and decompress data) for the diminution of space engaged by data. In a compressed database, data is hoarded in condensed format on disk and is also decompressed instantly when convert from disk or through query dispensation. In databases, and mainly in data warehouses, the diminution in the size of the data acquired by compression symbolizes usually a expand in speed, as the second cost in finishing time (to compress and decompress the data) is remunerated by the diminution in amount of the data that have to be examine/stored in the disks.

Computer systems permit the dispensation of enormous quantities of data for an assortment of purposes. As the capability to practice the data enhances, the necessitate for data storage systems which present enormous data storage capabilities united with fast admission for horde systems. Another feature essential by many business and industries is continuous availability. Many businesses operate on a worldwide basis, and have a requirement for round the clock contact to the databases accumulated in one or more data storage systems. The data accumulated in these data storage systems is demanding at an implausible rate, for instance, with business processing, stipulation systems and data mining, the data is varying and modifying many times per second. Another prerequisite of data storage system is episodic backup of data together for archival purposes and for data revival in crate of a system failure. Consequently, system backup must be achieved on a common basis.

After compressing the real time database systems, it is necessary to take back up of those databases adapt for ease retrieval from archive. While a backup is being written to multiple backup devices, several internal synchronization points occur. The most important such point occurs when all the data in the database has been backed up and the transaction log is about to be backed up. The present work gives a simple and an optimal approach for backup the compressed database in parallel multi storage servers.

2. LITERATURE REVIEW

The construction of data compression in the real-time database has been commenced along with the chronological data feature, the advantages and disadvantages of data compression algorithm [11]. A reliable datasets are commonly vast sufficient to stipulate data compression. A usual data compression approaches heed for the table as a enormous byte string and dart at the byte level. A database compression algorithm [10] called accomplishes well density while approving contact still at attribute stage with no need of the decompression of a greater unit.

Data compression has been a much trendy subject in the study literature and there is a vast extent of work on this subject. The most apparent inspiration to consider compression in a database background is to weaken the space essential in the disk. The paper [3] proposed the firmness of data in Relational Database Management Systems (RDBMS) using open substance compression algorithms. Information theory consistently densest with "usual data," be it textual data, image, or video data. Nevertheless, databases of different sorts have appear into survival in current years for accumulating "alternative data" counting social data, biological data, topographical maps, web data, and medical data. In compressing such data, one must reflect on two types of information [4] with lossy compression techniques [7]. A system of lossy compression of divergent memory less sources for contracting the sources carried on bounded distortion measure [5].

Scheduling at enhancing compression production, an iterative algorithm [6] is designed to determine the best partition of a frame with calculation of compression ratio [2]. Data hoarded in databases keep increasing consequently of businesses necessities for more information. A small section of the cost of charging huge amounts of data is in the price of disk systems, and the resources operated in running that data. The paper [8] commences different compression techniques for data piled up in row slanting in addition to column-oriented databases. Observing data in this compressed design as it is activated upon has been revealed to progress query performance by up to an order of magnitude. Naturally, data in columns is more Compressible than data in rows. Compressions algorithms [9] achieve better on data with low in sequence entropy (high data value position) i.e are worked for optimization point.

3. PROPOSED MULTI STORAGE PARALLEL BACKUP (MSPB) FOR REAL TIME DATABASE SYSTEMS

The proposed work is efficiently designed for implementing the multi storage parallel backup systems for real time database systems. The real time database systems are compressed with the **HIRAC** algorithm (explained briefly in the previous work), then the process of storing the backup compressed data in multiple storage devices takes place.

Since the work provided a solution for allowing the database to backup at multiple devices in parallel, it reduces the disk storage requirements and increases the speed of backup and restores operations. It is mainly used for enterprises with real time large databases. The major purpose of this technique is to permit the diminution of the space engaged by aspect database tables with large number of rows, dropping the total space engaged by the data warehouse. The architecture diagram of proposed multi storage parallel backup for real time database systems (MSPB) is shown in fig 3.1.

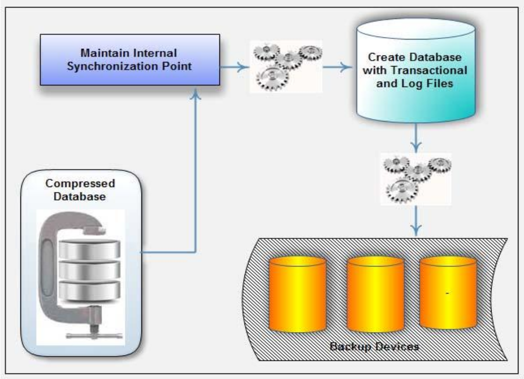

Maintain Internal Synchronization Point
Create Database with Transactional and Log Files
Compressed Database
Backup Devices

Fig 3.1 Architecture Diagram of Proposed MSPB

From the fig 3.1, it is being observed that the Multi Storage Parallel Backup consists of 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 above steps have been followed to perform the process of multi storage parallel backup of compressed database systems.

3.1 Overview of Backup Systems

Overviews of parallel database backup systems major components are shown in fig 3.2. One or more host systems are prepared to process, access and accumulate data from a data storage systems. The host system has some parallel processors, Symmetric multi-processor are integrated to the data storage systems over an interface (which might be any of the interface such as network or some other device). The host systems are also integrated to backup systems which recognizes data backup and restore to suitable storage device. The interface among the host and backup systems are of several types of interface such as TCP/IP connection. An optimal approach is used here for storing the items in the compressed database.

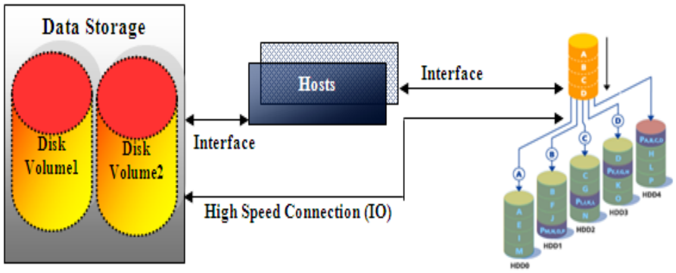


Fig 3.2 Overview of Components of Backup Systems

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 (I/O) 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.

3.2 Optimal Multi Storage Parallel Backup Compressed Data

If the database and transaction log files do not already exist, they must be created before data can be restored to them. The database and transaction log files are created and the file contents initialized to zero. Separate worker threads create and initialize the files in parallel. The database and transaction log files are sorted by disk device, and a separate worker thread is assigned to each disk device. Because creating and initializing files requires very high throughput, spreading the files evenly across the available logical drives yields the highest performance. The work presented here will be designed (Fig 3.3) to handle the backup of a compressed database efficiently if it has only one instance. An instance is termed as a multiple processes running on a computer system. A log information is also be kept to track the database changes made by the instances.

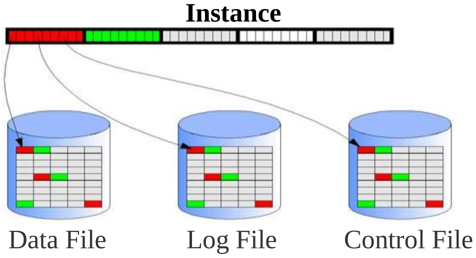


Fig 3.3 Block Diagram of Backup Storage Single Instance Model

The above diagram explains the process of log and transaction files maintained if an instance happens. It includes several data files which has control file, a small administrative file required by every database, necessary to start and run the compressed data system. Several identical control files are linked with a single file for data security. A log files consists of a sequential log actions that are reapplied to the database, if they did not written into the disk. The log usually consists of two files; one is optionally spooled to archived database systems, while the other is being written online. The online log files consist of set of log files which might not have been compressed. Finally, there is a parameter files which maintains instance specific information, for instance, buffer sizes, or other routing information.

In the Fig 3.3, 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 multi storage parallel database system has several restrictions on a backup system for a data storage system. All data files in the multi storage parallel devices must reside on contiguous portions of storage devices and visible to all parallel server database devices.

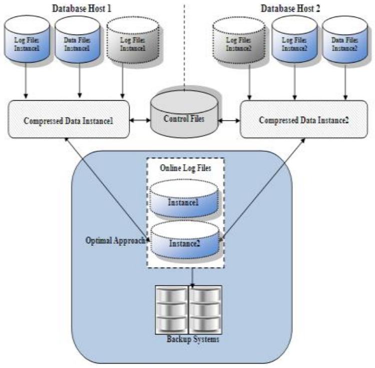
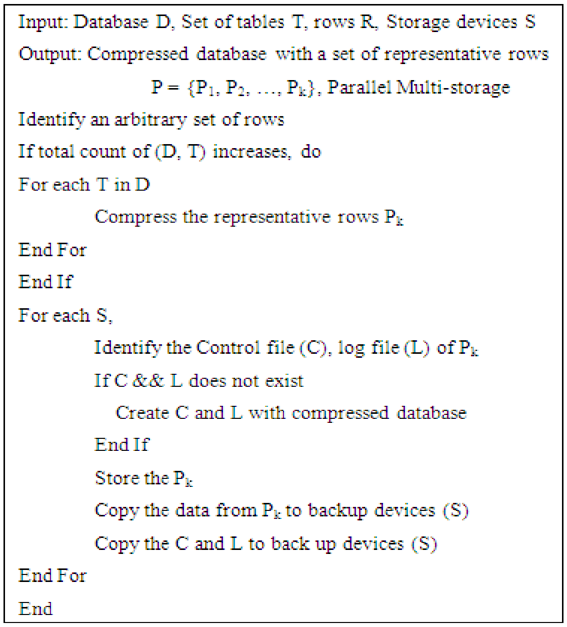


Fig 3.4 Multi Storage Parallel Server Model for Logs and Control Information

Multi storage parallel database architecture is depicted in fig 3.4. One database is serviced by several instances. An instance is one or more process along with memory servicing a database by independently accessing the database. Normally, the database is first compressed using HIRAC algorithm. Then the compressed database is stored with multiple storage devices by dividing as parts of compressed data. 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.



```
Input: Database D, Set of tables T, rows R, Storage devices S
Output: Compressed database with a set of representative rows
            $P = \{P_1, P_2, \ldots, P_k\}$, Parallel Multi-storage
Identify an arbitrary set of rows
If total count of (D, T) increases, do
For each T in D
        Compress the representative rows $P_k$
End For
End If
For each S,
        Identify the Control file (C), log file (L) of $P_k$
        If C && L does not exist
          Create C and L with compressed database
        End If
        Store the $P_k$
        Copy the data from $P_k$ to backup devices (S)
        Copy the C and L to back up devices (S)
End For
End
```

Fig 3.4 Algorithm for Multi Storage Parallel Backup

The algorithm (Fig 3.4) begins by identifying the arbitrary set of rows from the table. It then repeatedly enhances with the aim of enhancing the total coverage over the table to be compressed. Based on the control and log files of the compressed database of its storage devices, the restore operation of compressed database takes place according to the respective storage device in an optimal manner.

4. EXPERIMENTAL EVALAUTION

In this work, we have how the compressed housing database systems with the set of representative rows and columns for real time environments are stored in parallel multi storage devices. The implementation of the proposed optimal multi storage parallel backup for real time database systems (MSPB) is done in Java and used data set derived from UCI repository. It allowed out a series of performance experiments in order to examine the efficiency of the proposed multi storage parallel process for real-time database systems explained in a fig 3.2, 3.3, 3.4. The experiments were run on an Intel P-IV machine with 4 GB memory and 2GHz dual processor CPU. The proposed MSPB for real time environment is successfully designed for taking compressed database backup and stored with multiple storage systems. The performance of the proposed optimal multi storage parallel backup for real time database systems is measured in terms of

- CPU Overhead
- Time to Backup
- Disk Storage Requirements

CPU Overhead: measures the quantity of work a computer's giving out unit can achieve for a particular task. In the proposed MSPB, the percentage of the capability of CPU used for restoring the compressed backup data is low and some of these tasks engage underneath the utilities of the computer's operating system.

Time to Backup: is the time taken to restore the compressed data across multiple storage devices simultaneously. The proposed MSPB consumes less time to restore the backup compressed data since it preceded the restore operation in an optimal manner.

5. RESULTS AND DISCUSSION

In this work, we have seen how the compressed database with size of 5GBs is efficiently stored under multi storage parallel devices by adapting the optimal parallel server method and compared the results with an existing simple model for storing database at multiple devices in parallel written in mainstream languages such as Java. We used a real time database for an experimentation to examine the efficiency of the proposed optimal multi storage parallel backup for real time database systems. The below table and graph described the performance evaluation of the proposed optimal multi storage parallel backup for real time database systems with an appropriate results.

No. of Devices	CPU Overhead	
	Proposed MSPB	Existing SMSD
2	10	25
4	24	39
6	18	21
8	30	36
10	46	28

Table 5.1 No. of Devices vs CPU Overhead

The above table (table 5.1) described the CPU overhead based on number of parallel devices present for storing the compressed database. The efficiency of multi storage in parallel using the proposed optimal multi storage parallel backup for real time database systems is compared with an existing simple model for storing database at multiple devices in parallel [SMSD].

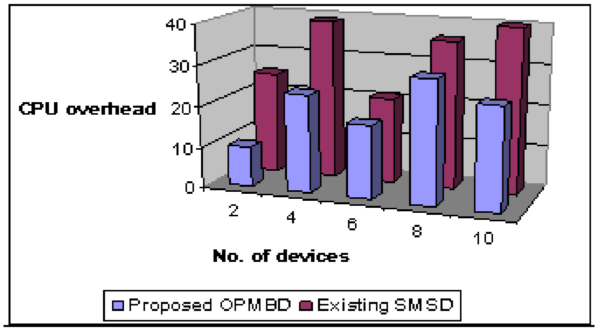


Fig 5.1 No. of Devices vs CPU Overhead

Fig 5.1 describes the CPU overhead raise when number of devices used for storing the compressed data increases. CPU Overhead measures the quantity of work a computer's giving out unit can achieve for a particular task. In the proposed MSPB, the percentage of the capability of CPU used for restoring the compressed backup data is low and some of these tasks engage underneath the utilities of the computer's operating system. Normally, with the proposed MSPB, the database are at first compressed using HIRAC algorithm, then the compressed database is stored under different storage devices simultaneously. So, even the number of storage devices with the CPU increases, the CPU overhead in the proposed MSPB is low. Because the devices attached with the CPU has compressed database which is spitted in an optimal manner. The device which has an optimal number of parallel compressed data are taken as to retrieve the items from the compressed database. Compared to an existing simple model for storing database at multiple devices in parallel, the proposed optimal multi storage parallel backup data compression for real time database systems provides less CPU overhead and the variance is 20-30% low.

No. of Data(MBs)	Time to Backup (sec)	
	Proposed MSPB	Existing SMSD
25	5	15
50	13	20
75	9	29
100	24	25
125	20	33

Table 5.2 No. of Data vs Time to Backup

The above table (table 5.2) described the time taken to restore the data in multiple store age devices simultaneously. The efficiency of multi storage in parallel using the proposed optimal multi storage parallel backup data compression for real time database systems is compared with an existing simple model for storing database at multiple devices in parallel [SMSD].

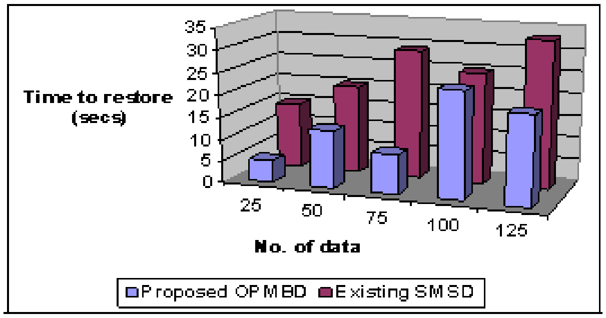


Fig 5.2 No. of Data vs. Time to Backup

Fig 5.2 described the processing time to take to restore the compressed data into multiple storage devices in parallel. Time to restore is the time taken to restore the compressed data across multiple storage devices simultaneously. The proposed MSPB consumes less time to restore the backup compressed data since it preceded the restore operation in an optimal manner. In the proposed MSPB, the restoring of the compressed data with the devices are done probably with optimizing the parallel multi storage devices. If the number of data in the database increases, the time to restore all the compressed in the storage devices simultaneously is less. Since the optimization technique is used with the parallel multi storage devices, the proposed MSPB consumes less time to restore the compressed data. The restoring time is measured in terms of seconds (secs). Compared to an existing simple model for storing database at multiple devices in parallel, the proposed optimal multi storage parallel backup data compression for real time database systems consumes less restoring time and the variance is 25-35% low.

Techniques	Disk Storage Requirements (%)
Proposed MSPB	24
Existing SMSD	56

Table 5.3 Methods vs. Disk Storage Requirements

The above table (table 5.3) described the requirements of disk to store the compressed database simultaneously. The efficiency of multi storage in parallel using the proposed optimal multi storage parallel backup data compression for real time database systems is compared with an existing simple model for storing database at multiple devices in parallel [SMSD].

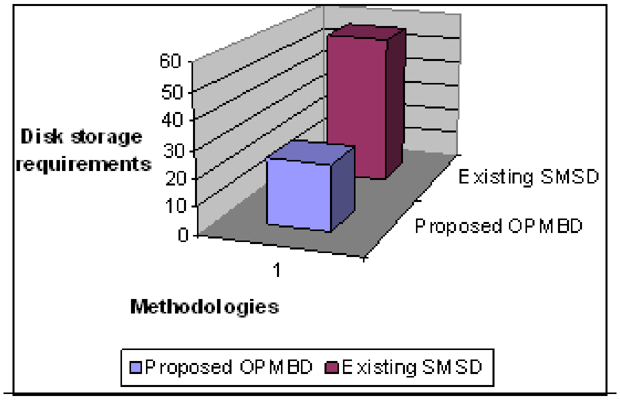


Fig 5.3 Methods vs Disk Storage Requirements

Fig 5.3 describes the requirements of disk to store the compressed database simultaneously based on the methodology applied for compression technique. Since the proposed MSPB stored under multiple devices in parallel, it does not require more disks to store.

Finally it is being depicted that the present work gives a simple and an optimal approach for restoring the compressed backup data in parallel server database stored in data storage systems. Parallel server database allowed multiple users to access the database simultaneously by taking online or offline backup data with proper access to all control files.

6. CONCLUSION

In this work, we efficiently stored the compressed database in multiple devices simultaneously for the real time environment. Then the compressed data is to be stored under different devices simultaneously with an optimal approach. Parallel read operations can be increased by spreading the database files among more logical drives. Similarly, parallel write operations can be increased by using more backup devices. However, careful configuration of the hardware is likely to be needed to obtain optimal performance.

Merits of using the proposed MSPB are:

- It increases the speed of backup and restore operations
- It efficiently allowed the backup data to be stored at multiple devices at parallel
- Greatly reduced the time taken for backup operations with less disk storage requirements

Experimental results have shown that the proposed optimal parallel multi-storage backup data compression for real time database systems are efficient in terms of backup time, efficiency compared to an existing simple model for storing compressed data.

REFERENCES

[1]. Limin Liu et. Al., “A low-complexity iterative mode selection algorithm Forwyner-Ziv video compression”, ICIP 2008. 15th IEEE International Conference on Image Processing, 2008.

[2]. Matsushita, R. et. Al., “Critical compression ratio of iterative re-weighted l1 minimization for compressed sensing”, IEEE on Information Theory Workshop (ITW), 2011.

[3]. Jorge Vieira, Jorge Bernardino, Henrique Madeira , “Efficient compression of text attributes of data warehouse dimensions”, Proceeding on the 7th international conference on Data warehousing and Knowledge discovery, 2005.

[4]. Yongwook Choi et. Al., “Compression of Graphical Structures: Fundamental Limits, Algorithms, and Experiments”, Information Theory, IEEE Transactions on, Feb. 2012.

[5]. Yoon Kim et. Al., “Locally-separated vertical channel SONOS flash memory (LSVC SONOS) for multi-storage and multi-level operation”, IEEE Silicon Nanoelectronics Workshop, 2008. SNW 2008.

[6]. Yazhong Ren et. Al., “Data storage in digital library using multi-storage technology”, IEEE 3rd International Conference on Communication Software and Networks (ICCSN), 2011.

[7]. Wainwright, M.J. et. Al., “Lossy Source Compression Using Low-Density Generator Matrix Codes: Analysis and Algorithms”, Information Theory, IEEE Transactions on, **Volume:** 56, Issue: 3 , March 2010.

[8]. Gupta, A. et. Al., “Nonlinear Sparse-Graph Codes for Lossy Compression”, Information Theory, IEEE Transactions on, Volume: 55, Issue: 5, May 2009.

[9]. Veluchandhar et. Al., “A backup mechanism with concurrency control for multilevel securedistributeddatabasesystems”, Third International Conference on Digital Information Management, 2008. ICDIM 2008.

[10]. Aghav, S. et. Al., “Database compression techniques for performance optimization”, 2010 2nd International Conference on Computer Engineering and Technology (ICCET).

[11]. Chenggang Zhen et. Al., “Design and Realization of Data Compression in Real-TimeDatabase”, International Conference on Computational Intelligence and Software Engineering, 2009.

BIOGRAPHIES

M.Muthukumar 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.

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 100 papers in refereed international journals and refereed international conferences proceedings.