ISSN (Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

An ISO 3297: 2007 Certified Organization Volume 5, Special Issue 11, May 2016

International Conference on Innovations in Engineering and Technology (ICIET'16)

6th & 7th April 2016

Organized by

K.L.N. College of Engineering and Technology, Madurai, Tamil Nadu, India

# A Database Driven Framework for Non-Intrusive Load Monitoring

Dr. S. Srinivasan, S. JothiSankar

Professor, Dept. of Computer Science and Engineering, Anna University, Madurai Campus, India

PG Scholar, Dept. of Computer Science and Engineering, Anna University, Madurai Campus, India

**ABSTRACT:** A Database-Driven Framework for Non-Intrusive Load Monitoring is the effort to study and tounderstand the Big Data Storage and Analysis. And also to derive some useful solutions and betterments over existing frameworks in place using the latest powerful technologies and tools. Big Data is considered as a promising technology for increasing segment of IT industry and business applications includes names health care, banking etc. Since the extent and volume of big data is large, the fixed amount of storage is not suitable for storing such dynamic generation of data. Therefore recent technology uses cloud based storage and storage networking therewith improved data storage and mining techniques. There are the requirements for the flexible distributed architecture for the storage, transfer, manipulation and analysis of time-series data, the network-transparent with facilitation of remote viewing and management of large data sets, and best use of efficient data Reduction and indexing techniques. The requirements led to search for the better solutions using the latest technology, which are the objective of the proposed paper.

**KEYWORDS:** Bigdata, Non-Intrusive, Time-series, Data-Reduction, Indexing, Load Monitoring.

## 1. INTRODUCTION

A database driven framework is the comprehensive method designed to solve the Big Data problem of non-intrusive load monitoring. The structured Time-series data (Non-intrusive load) need to be stored as compressed data to its maximum and the best indexing method adopted for decompress and retrieval of Big Data. The framework meant to provide the central component of flexible, distributed architecture for storage, transfer and manipulation of data using NilmDB functionalities, which is network transparent and with remote viewing of large data sets. Thus, Big Data includes huge volume, high velocity and extensible variety of data, and the data in it will be of different types as, Structured data (Relational Data), Semi Structured Data (XML Data) and Unstructured Data (Word, PDF, Text, Media Logs etc.).

The Big data is really critical to our life and is emerging as one of the most important technologies in modern world. Some of the benefits which are very much known to all of us are, using the information kept in the social network like Facebook, the marketing agencies are learning about the response for their campaigns, promotions and other advertising mediums. Using the information in the social media like preferences and product perception of their consumers, product companies and retail organizations are planning their production.

### *1.1 Technologies Proposed for Big Data*

The technologies in the Big Data are important in providing more accurate analysis, which may lead to more concrete decision-making resulting in greater operational efficiencies, cost reductions and reduced risks for the business. To harness the power of big data, you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real-time and can protect data privacy and security.There are various technologies in the market from different vendors including Amazon, IBM, Microsoft etc., to handle big data. While looking into the technologies that handle big data, we may examine the two classes of technology as Operational Big Data and Analytical Big Data. Were Operational Big Data include the systems like MongoDB that provide operational capabilities for real-time, interactive workloads were data is primarily captured and stored, NoSQL Big data systems are designed to take advantage of new cloud computing architectures that have emerged over the past decade to allow

massive computations to be run inexpensively and efficiently, this makes operational big data workloads much easier to manage, cheaper and faster to implement.

*1.2 Impact of NILM on Real Time Applications*

As stated by the NREL, U.S. Department of Energy (Technical Report, NREL/TP-5500-55292 November 2012) NILM technologies promise to deliver valuable data on what, when and how end-users are consuming energy and related data on it. To create killer apps based on these data, we must first determine who will use them and why. The potential customers for a residential NILM application include:

- The homeowner or building manager, who is using energy and has various motivations for seeking out a NILM application;
- The technology sector, which will incorporate NILM technology into an existing or new product;
- The service sector, which will use NILM tools to deliver home maintenance, retrofits, advertising, or security services; and
- The utility, which sells the energy and receives regulatory incentives to cost-effectively implement efficiency.

*1.3 Database Management Tools*

Some NoSQL systems can provide insights into patterns and trends based on real time data with minimal coding and without the need for data scientists and additional infrastructure. In Analytical Big Data, which includes systems like Massively Parallel Processing (MPP) data base systems like Hadoop, NILM etc., and MapReduce technique that provide analytical capabilities for retrospective and complex analysis that may touch most or all of the data.MapReduce provides a new method of analyzing data that is complementary to the capabilities provided by SQL, and a system based on MapReduce that can be scaled up from single servers to thousands of high and low end machines. The major challenges associated with big data are, Capturing data, Curation or Refinement, Storage, Indexing, Searching, Sharing, Transfer, Analysis and Presentation. To fulfill the above challenges, the organizations normally take the help of higher end Enterprise Servers.

## II. **OBJECTIVES**

The main objective of the paper is to discuss and provide best and optimal solution for data storage and management of data access methods using the data handling techniques to improve the efficiency of Non-Intrusive load monitoring and database management system, which comprise of,

1) Formulation of Data Storage Structure for the purified data storage into the portable multiplatform supported data files.
2) Indexed storage of data elements for the quick and efficient data storage and retrieval over the network.
3) Using best tools for the optimized data encryption / decryption, data compress and security.

The survey of Ahmed Zoha et.al (2012), A significant reduction in the energy wastage can be achieved through fine-grained monitoring of energy consumption and relaying of this information back to the consumers. Motivated by this, we see a large scale deployment of smart meters in the residential environment by the governments of UK and USA. While it is envisioned that the smart meters will charge consumers based on peak or off peak timings, traditional smart meters are only able to measure energy consumption data at house level granularity. In order to implement a precise demand –response functionality, a much finer granularity of information is required. To achieve this, research efforts have led to the development of Appliance Load Monitoring (ALM) methodds.

There are two major approaches to ALM, namely Intrusive Load Monitoring (ILM) and Non-Intrusive Load Monitoring (NILM). In the literature, ILM and NILM are alternatively referred to as distributed sensing and single point sensing methods respectively. This is because the ILM apporaches require one or more than one sensor per appliance to perform ALM, whereas NILM just requires only a single meter per house or a building that is to be monitored.As most of the work in NILM is based on supervised learning, which requires each appliance to be profiled during the training phase, a subtle change by an energy supplying company (*i.e.*, power factor correction) at the main circuit line can cause a mismatch of appliance profile ( presented by E. Budish et.al. (2013), D. J. Leeds (2013), J. Han et.al. (2011) and PraywinEbnezer et.al (2014) ). Another closely related challenge for NILM is the update of the appliance signature database. We can easily conclude from our literature review that most of the NILM solutions for load disaggregation require off-line training of the algorithms. It is impractical to include all the appliances of different characteristics (make, model, size, *etc*.) in the database and hence algorithms would not be able to recognize any such device that is not in the appliance signature database. One possible solution is to make use of interactive technology in which a system can interact with the user using different interaction channels such as WEB and through mobile phone based short message service. In the presence of an unknown load the system makes an intelligent guess or the closest match will be presented to the user for verification. The user input can further be used to label the device pattern and consequently the appliance signature library will be updated.

Future work should focus on unsupervised learning algorithms for NILM that do not require human labeling of data. In order to facilitate the grouping and detection of appliances in an unsupervised fashion, the contextual cues from the environment and device usage patterns can further be exploited. The algorithms including KNN, ISODATA, Self-Organizing Trees do not require a pre-learning phase and they have been successfully employed in other research

domains for the task of unsupervised pattern classification. The effectiveness of these learning methods for unsupervised load disaggregation can further be explored.

## III. EXISTING SYSTEM STUDY

*3.1 System Architecture*

The NILM database framework is structured as a client-server architecture. The server process, called simply NilmDB, runs on a single system and stores data on that system. Client programs can run on the same system or any other system, and connect to this server to perform actions such as inserting new data and extracting existing data.

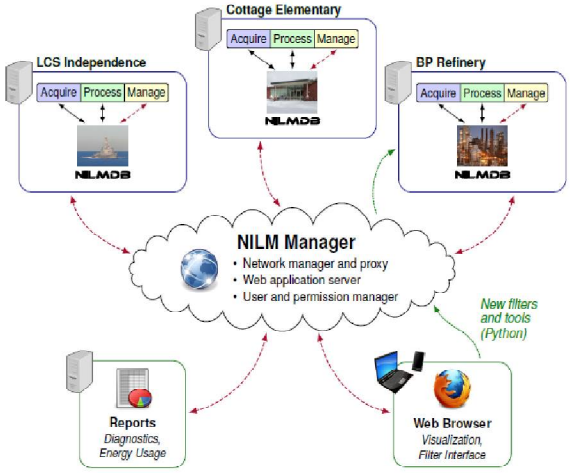


*Figure 3.1 NILM System Architecture*

The organizational architecture of NilmDB is outlined in the Figure. 3.1.All data in NilmDB is organized within streams. Streams contain homogeneous time-series data from a particular source. This data can represent physical quantities, computed values, or any other time stamped information. Examples include voltage, current, temperature, vibration, spectral envelopes, system run-time and health metrics, error and event indicators etc. Streams are organized and identified in the database using a tree like path structure that mirrors an arrangement of files and folders.

Paris, J et.al. (Sep 2014) presents with NilmDB as, the streams can be viewed conceptually as large tables of data. Each row contains a unique time stamp and data that matches the stream's layout, which is determined when the stream is created and indicates the number of columns and their data type.The NilmDB database layer is responsible for managing the low-level storage and retrieval of stream. There are two primary data stores, as A SQL backend, based on the SQLite embedded database utility and a custom bulk data storage backend used for stream data contents.

Binary search algorithm to enumerate unique preprocessor outputs as the data element value varies. Algorithm used to find the data sets for the time range in NilmDB :

*3.1.2 Algorithm*

1. Outputs ← [ ] // Output array
2. Function ENUMERATE (f, A_low, A_high)
3. P_low← PREPROCESSOR(SAMPLE(f(A_low)))
4. P_high← PREPROCESSOR(SAMPLE(f(A_high)))
5. If P_low<>P_high then // More than one output in this interval ?
6. A_mid← (A_high + A_low) / 2
7. ENUMERATE(f, A_low, A_mid) // search 1st half
8. ENUMERATE ( f, A_mid, A_high) // Search 2nd half
9. Else if P_low not in outputs then
10. Outputs ←outputs + P_low // One new unique output ; store it
11. ENUMERATE(f, 0,1)
12. Return SORT(outputs)
13. Stop

*3.1.3 Disadvantages*

- The framework is Platform dependent
- No Indexing and Insecurity in the data storage.

- Data fetch by the Binary search on each row for the time range.
- No data compression is out of scope of framework.

## IV. THE PROPOSED SYSTEM

*4.1 System Architecture*

The proposed system is designed to solve the disadvantages of existing system, specifically focussed to design of the data input structure and automated indexing using the unique identities as sequential numbers or rowids of each record item for the quick search and retrival of indexed rows from the data files. And in turn, making the database and data segments accessible by any of the platforms used by the clients or data sources. The extensive SQL features of SQLite has been used to solve the data structuring, Indexing, compressed storage and retrieval over the network and clients in the request of data streams with optimized data access speed and storage management.

Were the filters can take role of controlling data input streams, the database tools and features will manage the storage and and data access methods using builtin search algorithms and SQL querys. When the querys to search and fetch the row data executed, which results best and minimized search time of about 40 % over the indexed data sets in the tables of the database files. The data compress (ZIP) mechanism of the database tool provides best compressed savings on the stoage media space hence saves considerable amount of disk space, minimizes the over burden of Operating System on managing big data files on disks. The advantages over the existing system summerizes the benefits aspectes of the new system and meeting the requirements of latest Big Data handling challenges.

*4.1.1 Advantages*

- The structured and Indexed data storage
- Minimized time to search any data sets from the database file.
- Compress and Decompress of data for reduced storage area on disk.
- Secured and Non Intrusive storage mechanism.
- Platform independent Database Tool.

*4.2SYSTEM DESIGNS AND ARCHITECTURE*

*4.2.1 Flow Diagram*

The architecture of the proposed system is described in the Figure. 4.2.1, each component states the unique operaton and process stages of the system.The central component manages the component activities for the regulated data streams and data storage and fetch activities of the system.

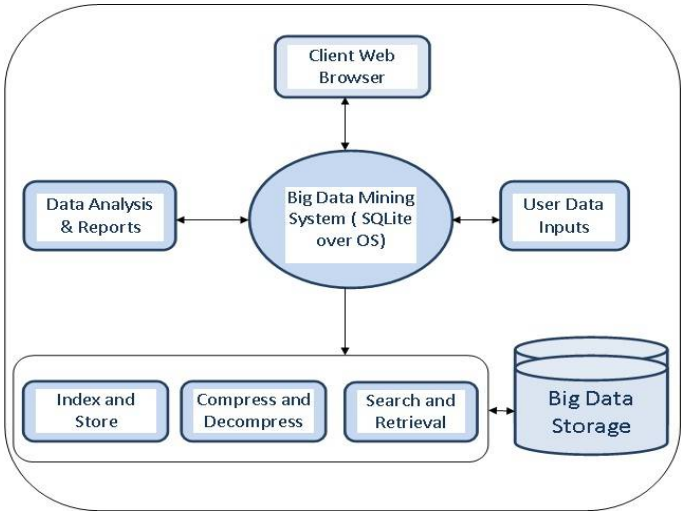


*Figure 4.2.1 Flow Diagram*

*4.2.2. UML Diagram*

The Unified Modeling Language (UML) Diagram used to describe the process flow of each stages of system components, is described in the Figure. 4.2.2.

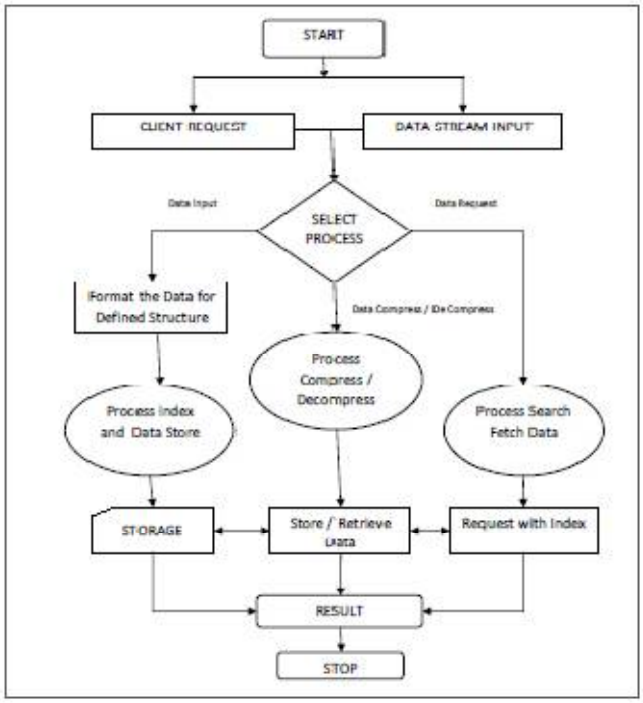


*Figure 4.2.2 UML Diagram*

*4.3 ALGORITHM DESCRIPTION*

*4.3.1 Indexing and Encryption Algorithm*

The Indexing of data items from each stream with the timestamp which is sent by the client devices, data sources are structured with a format such that, which can be inserted into the table as Row Id of each row is sequential and incremental, so that the first prime key has been fixed with the Row Id data value along with data input stream one by one. By assigning the Row Id of each row, the table becomes indexed automatically by the database table, hence there is not necessary to create separate index entity to maintain the table data for quick search purpose.The compress and decompress (Encrypt / Decrypt) is achieved by the use of built-in functions of SQLite, which performs the compress and decompress operation on table data. At the data insert process, performs compress and stores data to the table data location and preserves the space considerably. And also encrypts the data during the compress process (ZIP). The decompress (decryption) occurs during the data read, update and delete operations.

*Data Access Algorithm :*

*// Design SQL Queries for Insert column p1//*
*// intpk, c1,c2,c3..cn, tn as timestamp to Table Tb1 //*
1. Create Table Tb1(p1 int pk, c1 text, c2 text, cn int, tn date) – Table Creation
*// Table created with p1 as pk //*
2. rowid ← p1 *// Assigned automatically by DB system //*
3. Auto increment (rowid) *// Along with primary key//*
*// Optional explicit Index creation//*
4. Create index Tb1_ndx on (p1,cn,tn)
*// The table data indexed implicitly on rowid or p1 //*
*// Database Encrypt / Decrypt//*
5. dbfile ← sqlitecrypt (database / table)
*// Encrypts / Decripts specific db or table Decrypt //*
*// with SQLiteCrypt of AES 256 Standard //*
*// Search row data of p1 on tn timespace //*
6. Search (Tb1 (p1,tn) )
*// Takes rowid and p1 values //*
7. Internal Index (Tb1) ← [rowid, p1, tn]
*// Internally takes values of rowid and related p1, tn //*
8. Tb1 (Values) ← Table Row = rowid / p1
*// Directly takes the Tb1 row values //*
*// through indexed rowid and p1 //*

*// SQLite ZIPVFS database compress//*
9. DB_comp ← SQLiteZIPVFS ( DB)
*// DB Compress Extension Stores compressed DB file //*
*// into DB_comp / DB_Comp.zip //*
10. Return Tb1, DB, DB_comp
11. STOP
*// Returns resultant objects for the SQL Queries //*

The data access algorithm is to simplify the search task at database level using built in functions / procedures in the simplified method.

*4.4 The Database Engine SQLite*

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the mos widely deployed database in the world with more applications than we count including several high-profile projects like Adobe, AIRBUS, Apple, BOSCH, Firefox, Thunderbird Email Reader, Google Gears for Android etc.An understanding of SQLite is a prerequisite for understanding ZIPVFS. The architecture of the SQLite is described in the Figure. 4.4.1provides a high-level overview of the internal organization of SQLite. The Tokenizer, Parser, and Code Generator components are used to process SQL statements and convert them into executable programs in a virtual machine language. The Virtual Machine module is responsible for running the SQL statement byte code.

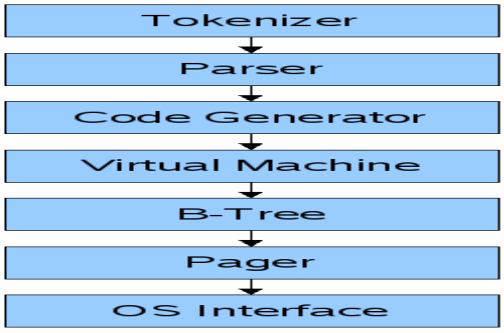


*Figure.4.4.1 Architecture of SQLite*

The components that interact with ZIPVFS are the bottom two: the Pager and the OS Interface. Hence, we will provide additional details on those two components. The OS Interface is really an abstract class called "sqlite3_vfs" that can be subclassed to provide an operating system interface for the various machines upon which SQLite runs. We call each subclass of sqlite3_vfs a "VFS" which is an acronym "Virtual File System". The VFS objects provide methods such as the following:

- Check for the existence of a file on disk.
- Delete a file.
- Open a connection to a file.
- Read or writes a specific number of bytes from a specified location within an open file.
- Truncate a file.
- Close a file connection.
- Return the size of a file in bytes.
- Flush buffered file content to the disk surface.
- Lock or unlock a range of bytes within a file.

The SQLite core never invokes any operating-system interface calls itself. Instead, when SQLite needs to manipulate files (or take advantage of other operating system services) it invokes methods in a VFS object and that VFS object translates the request into system calls appropriate for the underlying operating system. SQLite works on both Windows and Ubuntu (Linux extension) were the real servers can be deployed. The communication system established with data emitting devices and sources and for handling client requests and are managed by the server side components of Widows with Wamp (PHP) and with NilmDB on Ubuntu. DB Browser for SQLite is used to import and export the database files and to perform the operations of the modules on the data collected.

*4.4.1 DB Browser for SQLite*

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain as is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications. Figure 4.4.1 showsDB

Browser for SQLite (sqlite browser) is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

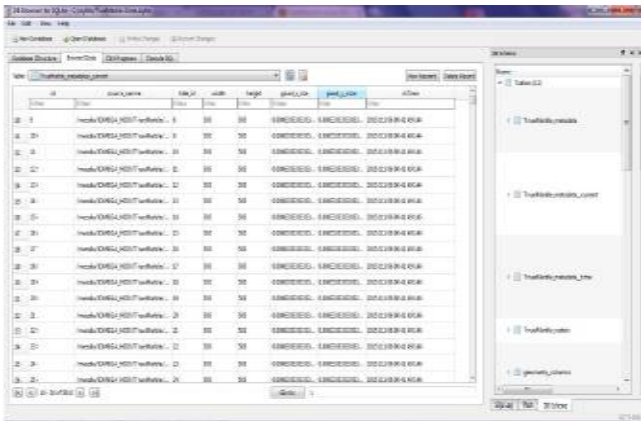

Figure 4.4.1 SQLite DB Browser

It is for users and developers wanting to create databases, search, and edit data. It uses a familiar spreadsheet-like interface.

## 5. DATA STORAGE AND ACCESS SPEED

The graph shown in the Figure.5.1b, result of Comparison of the NilmDB with SQLite and other database frameworks as per the data sets displayed in Table5.1a in terms of data storage and access speeds per second.

| DB Engines | KB / Second | MB / Hour | GB / Day |
|---|---|---|---|
| NilmDB | 160 | 576 | 13.8 |
| CoreData | 182 | 639 | 14.9 |
| SQLite | 304 | 1068 | 25 |

Table5.1a Data Storage / Retrieval Comparison

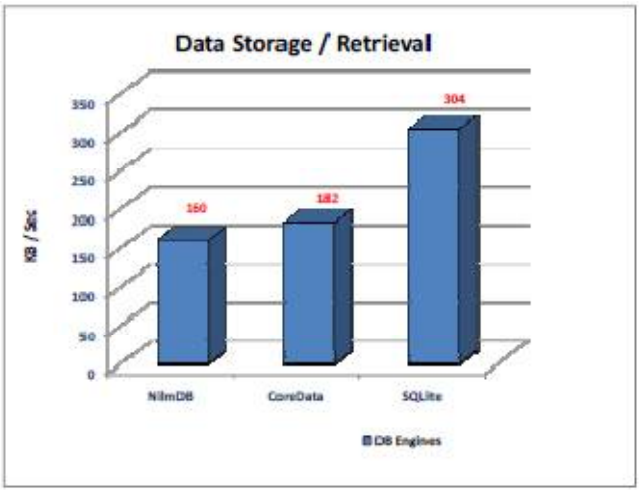


*Figure.5.1b : Comparison Graph of Access Speed*

SQLite is supported with the API for database encryption and decryption using SQLiteCrypt, which is transparent AES 256 encryption support for SQLite. Which implements a self-contained, embeddable, zero-configuration encrypted SQL database engine, that allows you to protect data in SQLite database. Embedded into the application using the passpharase in the original database with simpler zero configuration method. AES offer very good performance compared to other encryption alternatives ie. DES.SQLiteCrypt encrypt/decrypts data as block. It means that only required piece of data being encrypted or decrypted, not process whole database every time user executes SQL Commands or formulated function calls. It allows to change passphrase easily with one SQL command and works everywhere original SQLite works at ie., Windows, Linux, Mac OS, Android, iOS etc., and the database are fully cross platform.

## VI. CONCLUSION

The presented database driven non-intrusive load monintoring system defines the formulation of data design and structue of entities to store and data access methods. This work has pictured a new, comprehensive framework for improving the quality and capabilities of NILM, while introducing a powerful new system architechture to manage and analyze the vast quantities of data it generates and supports to manage the Big Data access methods using the framework designed.The data management tools used are fully support the objective features as, Platform Independent, Data Portability, Optimal data access speed, best Indexing and easy to implement at any kind of Server proposed for the implementation of Big Data Management system. The components of each moduls are funtioning independently with well defined purposes and results the expected output. When comparing the existing NILM system and its features, the new framework provides best optimized resuls on Access speed, Managebility, Less complex, Data Compress and Encryption and essy to deployment.

In Future, we propose A Database-driven framework for Big Data Management (Non-Intrusive Load) and Monitoring system with enhanced features to support the cloud based web database across different geographical locations to manage the Big Data and Data Mining using the advanced platforms and tools like Hadoop, MongoDB, Oracle etc., with the support of the efficient and highly available Web Servers (Apahche / Wamp / Xamp). The user frinendly interactive front-end environments development using Python / PHP / JSP with JScripts / ASP. NET. The framework can be developed to best suit the concurrent access to the data in the database segments with Big Data Management solutions.

## REFERENCES

[1] Paris, J.; Donnal, J.S.; Leeb, S.B., "NilmDB: The Non-Intrusive Load Monitor Database," *Smart Grid, IEEE Transactions on* , vol.5, no.5, pp.2459,2467, Sept. 2014 doi: 10.1109/TSG.2014.2321582

[2] D. J. Leeds *The Soft Grid 2013-2020:Big Data & Utility Analytics For Smart Grid,TechRep.*, 2012 [online] Available: http://www.greentechmedia.com/research/report/the-soft-grid-2013

[3] E. Budish , P. Cramton and J. Shim *The High Frequency Trading Arms Race:Frequency Batch Auctions as a Market Design Response,* 2013 [online] Available:http://home.uchicago.edu/~shim/Papers/HFT-FrequentBatchAuctions.pdf

[4] *An Update on Google Health and Google PowerMeter*,, 2013 :Google, Inc. [online] Available: http://googleblog.blogspot.com/2011/06/update-on-google-health-and-google.html

[5] J. Han , E. Haihong , G. Le and J. Du "Survey on NoSQL database", *Proc.ICPCA*, pp.363 -366 2011

[6] J. Paris *A Framework for Non-Intrusive LoadMonitoring and Diagnostics*, 2006

[7] PraywinEbnezer et.al (2004) : A Survey on Big Data Storage in Cloud

[8] Ahmed Zoha et.al (2012) : Non-Intrusive Load Monitoring approaches for Disaggregated Energy Sensing : A Survey