International Conference on Applied Energy
ICAE 2013, Jul 1-4, 2013, Pretoria, South Africa
Paper ID: ICAE2013-552

TOWARDS SMART DATA COMPRESSION FOR FUTURE ENERGY MANAGEMENT SYSTEM

Muhammad Nabeel, Fahad Javed, Naveed Arshad

Department of Computer Science
School of Science and Engineering
Lahore University of Management Sciences
Lahore, PAKISTAN
{muhammad.nabeel, fahadjaved, naveedarshad} aat lums.edu.pk

ABSTRACT

The smart grid is the next generation of electricity system in which information technology will aid in providing more efficient, secure and robust energy management. To facilitate this, automated metering infrastructure is being deployed to capture household energy consumption. The state of the art meters can collect this data at the granularity of a second or even lower. To transmit, store and manage this fast streaming large quantities of data is a challenging problem. Although different algorithms exist for compaction and compression of data, managing time series data of this magnitude and peculiarity is a less explored problem so far. In this paper we present three different data management techniques that can be applied to the smart metering data. We explore different algorithms that can be used and the issues that arise from their usage on the smart meter data. We also look at some peculiarities of the smart metering data which can be harnessed for more efficient data management. In the end we provide an analysis of the techniques and a guideline for selection of algorithms for smart meters data management infrastructure.

Keywords: Energy efficient data storage and management, supervisory control and data acquisition (SCADA) for smart grids.

NONMENCLATURE

Symbols	
Sec	Second
Min	Minute
Hr	Hour
Dev	Device
CR	Compression ratio

Abbreviation	
SCADA	Supervisory control and data acquisition
EMS	Energy management system
IEA	International energy agency
DBMS	Database management system
DCC	Data compression and compaction

1. INTRODUCTION

The smart grid is the next generation of electricity system in which information technology will aid in providing more efficient, secure and robust energy management. To achieve this goal, it is envisioned that home of consumers will be fitted with automated metering infrastructure capable of providing near real-time energy consumption data to the service provider. Various applications are being developed for non-intrusive load monitoring, energy forecasting, fault detection, demand side management etc, with the assumption that this data will be available for various applications. Needless to say that collection and transmission of this data will be of great importance to the viability of smart grids.

The research on how to collect and use this data is an active area. However, how to manage this data, to our knowledge, has not been studied in great detail. The problem of data management is significant and as we will see in section 2, even with a small set of 10,000 houses, the data could become unmanageable. The question is not just limited to storage spaces but also the fact that a larger database takes longer to query important information. As we will show, a simple query on a few months' data can take a very long time to execute thus making the system cumbersome. That is why we foresee that this data-management as the bottleneck in delivering better services in future smart grids.

To resolve this issue, in this paper we provide a survey of compression and compaction techniques used in other systems, for example, in databases and data mining communities. We weigh in different factors in the application of the algorithm to smart grid data and provide results of applying three related compression techniques on typical smart grids data.

2. DATA MANAGEMENT IN SMART GRIDS

Smart grids of the future are expected to be very large interconnected information systems where the consumers and the utility providers or their computing agents will communicate to deliver various services. As we will see in data sections, these applications have different data requirements ranging from high frequency data in the range of 16 KHz for non-intrusive [6] load monitoring to maintenance of gigabytes of historical data for load forecasting [5]. Managing these varied data requirements is a very big challenge.

The data management problem can be decomposed into two components. First is the data compression in networks for effective transmission of data. Transferring data at 16 KHz for all the houses in a region can be very costly. However, management of high speed data has received sufficient attention from sensor networks community where various protocols have been proposed for transferring high frequency sensor data [1].

The second problem is management of large datasets on the server repository. As we will see in section 2, even at 1 KHz, data sizes can grow very fast for even a 10,000 homes data. There are two aspects to this server side data management. First is storing in hardware and the second is its efficient structuring for quick retrieval and processing. As we shall see, there are various limitations in size for storage and large datasets can be very slow to query.

In this paper we will provide a survey and analysis memory optimization and query optimization for data management on servers which will increase performance and reduce storage. We will first develop an understanding of data flow in energy management system. Then we will discuss the challenges faced by energy management systems, analyze the data flow in smart grids and compare different algorithms on smart grid to determine how much storage savings we can achieve.

2.1 ENERGY MANAGEMENT SYSTEM

Energy management system is a computer aided system used by operators or background running services of electric grids. Basic purpose of EMS is to monitor, control and optimize the performance of a generation and/ or transmission system. These monitor and control functions are known as supervisory control and data acquisition. Energy management system can also provide metering and monitoring functions that allow operators to gather data and insight gives them that allow them to make more informed decisions about energy activities across their respective area. [2]

2.2 DATA FLOW IN ENERGY MANAGEMENT SYSTEM

Figure 1 shows the flow of data in an EMS. Smart grid monitors real time power generation, transmission grid and distributions grid. This comprehensive view of the power grid is achieved through sensors providing information in real time (seconds to milliseconds delay). Real time power grid controls and solves problems of prediction and detection and repairs the power system by integrated SCADA which also finds and heals the faults to avoid costly power failure and ensure reliability of power. The information integrated layer transmits data and other information between the data collection layer and the application layer and establishes enterprise service bus. [3].

The smart grids applications are deployed at the application layer. Some of the applications that have been discussed and deployed are advanced visualization [4], forecasting [5], load disaggregation [6] and fault tolerance [3].

The data needs for each of the application vary greatly. The load disaggregation algorithm for instance requires a 16 KHz data for accurate results. On the other hand 1Hz data is sufficient for visualization. The rate of propagation of this data is subject to the application but the data flow channels are expected to follow the data flows in figure 1. One thing to note here is that since the energy data changes its semantics on every layer, it is an important task to make sure that the system does not lose information.

2.3 DATA MANAGEMENT CHALLENGES IN EMS

The major component of data in smart grids is time series data [7] which continuously comes from different meters or devices at granularities ranging from readings collected after an hour to thousands of reading in a second. This variation is due to the different needs of the applications in smart grid. Each record of smart data contains meter id, a valid timestamp and meter reading.

For such high-speed data two major problems occur: Firstly, storage of database becomes a major issue because data comes at a rate of one second or lower. This causes problem in data loss as data can easily overflow from storage capacity. Secondly, the performance of the queries on this large data is slow. This means that if we run a query on a large amount of data then the performance of queries like aggregation slows down proportionally with size of data. So there is a need to compress and compact table as much as possible without losing the necessary information.

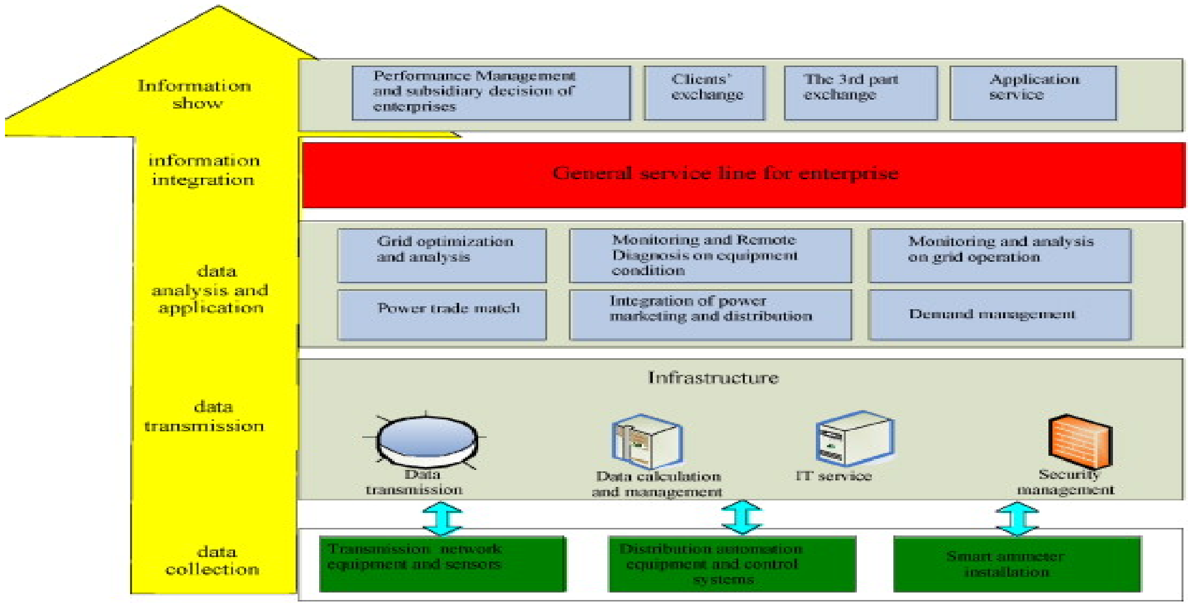
Information show
information integration
data analysis and application
data transmission
data collection
Performance Management and subsidiary decision of enterprises
Clients' exchange
The 3rd part exchange
Application service
General service line for enterprise
Grid optimization and analysis
Monitoring and Remote Diagnosis on equipment condition
Monitoring and analysis on grid operation
Power trade match
Integration of power marketing and distribution
Demand management
Infrastructure
Data transmission
Data calculation and management
IT service
Security management
Transmission network equipment and sensors
Distribution automation equipment and control systems
Smart ammeter installation

Figure 1 Data Flow in EMS [3]

To show the increase in the size of data in a smart grid we present an analysis here. If we have 10,000 meters connected which provide us with data after every one second, where floating point and integer takes 4 bytes and date and time take 8 bytes then we need storage space as shown in table 1.

Table 1 Growth rate of smart grid data

Meter Reading	No. of Rows	Size
1 month	25,920,000,000	386GB Approx
6 Month	155,520,000,000	2.2 TB Approx
1 year	311,040,000,000	4.5 TB Approx
2 year	622,080,000,000	9.0 TB Approx
3 year	933,120,000,000	13.5 TB Approx
4 year	1,244,160,000,000	18.0 TB Approx
5 year	1,555,200,000,000	22.6 TB Approx

For 10,000 meters every month 386 GB of smart data is needed on server and in one year the data size grows up to 5 TB and after 5 years the size of data extends up to 23 TB, as the size of the data grows the performance of queries decrease. The following graph shows how size of the smart data increases for 5 years assuming 386 GB of records added at every month. On x-axis represent time and y-axis represent growth of data.

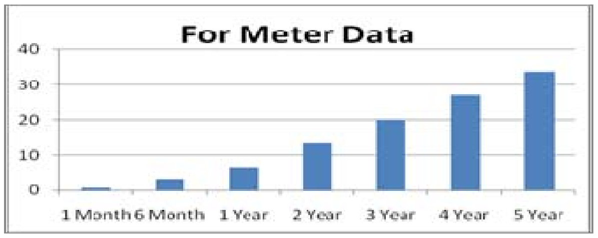

Figure 2 Smart Data Growth Rate. Y-axis represents size of data in terabytes for each year

Let us now discuss the decrease in performance of queries. The SQL statement benchmark for performance includes database configuration, disk space and memory [8]. Most frequent SQL queries in the smart grid are aggregation, insertion and selection. The maximum total amount of memory consumed by an object instance is 4 GB but when an object instance exists in memory, there is no fixed limit on the number of attributes in the object. When an object instance is inserted into the table, attributes are expanded into separate columns in the table [9]. If 'select' and 'aggregate' fetched data, memory exceeds more than 4 GB than the traditional database systems are unable to store that information? And if multiple queries are run by multiple users fetching large numbers of smart grid data from a centralized server then memory will be filled up very quickly. Moreover, as performance is proportional to data storage the data size increases the query time. For example for a 1.3 TB data the load time is 7 hours and to run required report on this 1.3 TB of data takes between 2-7 hours [10].

One can easily argue that the size of the table depends upon the operating system constraint on the file size; not by the DBMS [1]; but operating systems have some limitation on the file system as well. Following list of operating system describes limitation on table in MYSQL 5.6 release. [11]

Table 2 File size limit

Operating System	File-size Limit
Win32 w/ FAT/FAT32	2GB/4GB
Win32 w/ NTFS	2TB (possibly larger)
Linux 2.2-Intel 32-bit	2GB (LFS: 4GB)
Linux 2.4+	(using ext3 file system) 4TB
Solaris 9/10	16TB
MacOS X w/ HFS+	2TB

If we have 386 GB of table only Linux 2.4+ using ext3 file system can support such a large volume speed data up to ten months and a Solaris 9/10 can handle up to 30 months. After ten or thirty months management has to shift or perform rollover operations on existing datasets which is a costly operation. This costly operation used at regular interval to keep data flow working, so there is a strong need to compress the dataset to keep the shifting or rollover operation cost to a minimum which helps in memory and performance optimization. Every traditional DBMS have limitation on tables depending upon operating system constraint and internal DBMS working constraint, the following table provides us with an idea of how much data can be maintained by popular DBMS [8] [12] [13].

Table 3 Traditional DBMS comparison

DBMS	Windows Server	Linux Server
MSSQL	16 Terabytes	16 terabytes
Oracle	16 terabytes	32 terabytes
MYSQL	2 terabytes	4 terabytes

In the future, the problem of managing data in energy management systems could be more serious because of the popularity of EMS across the globe and as more and more houses, industry, offices are attached with EMS more storage is required for each addition of new device and meters. Roughly, if one meter is added and is sending data at the rate of one second, then it will take 40 MB of storage per month and if 10,000 new meters added it will take 386 GB approx data storage every month.

Compression is one of the solutions to reduce the size of data to save storage and increase I/O operations by which query processing performance could be enhanced. Derivative of datasets also compress data by changing its semantics according to smart grid functionality like forecasting, demand side management, load management, peak leveling, sustainability, demand response support. There are some off-the-shelf products like IBM Informix [14] which provide solution for time series data. The goal of this paper is to develop an open source compression strategy which is application specific to energy management system and can achieve more compression optimized for smart grid-related data.

3. SURVEY OF DATA MANAGEMENT TECHNIQUES

In this section we survey various techniques used for data management in different fields. The major techniques used to save space to boast performance is data compression which is reversible and data compaction which is irreversible.

3.1 DATA COMPRESSION AND COMPACTION (DCC)

There are two core techniques for data large management. First, Data compression is used to reduce redundancy in stored or communicated data, to increase the density of important data set. Compressing a file to half of its original size is equal to double the capacity of storage medium, so, it may become feasible to store data at higher level of storage. Doing also reduces the load on the input/output of the computer system [12]. Data compression is also known as reversible compression because decompression yields original data. Following are few highly used compression techniques from different fields.

Second, the second technique to reduce the data size is data compaction. Due to its very nature data compaction is also known as non-reversible compression because we get relevant set of information from the original data but we cannot reconstruct it. For example, we may generalize a value and specific variations of instances can be lost due to this generalization.

3.2 SEMANTIC INDEPENDENT AND DEPENDENT

The DCC technique can be semantic dependent, that is, it can use the inherent information within the domain to reduce the data size. In comparison domain independent DCC techniques only look at the statistics of data to perform their task [15].

3.3 DCC TECHNIQUES

In this section we will discuss some of popular DCC techniques.

Abbreviations

In abbreviations, instead of storing full words its abbreviated value is store. Conversion takes place in two ways, either when database is loaded from secondary memory to primary memory or during data reorganization. This scheme can cause reduction of size in range of 50-90% depending upon data distribution and repetition of words. A disadvantage of abbreviation is the look up cost associated with the encoding and decoding phase.

Pattern substitution

In pattern substitution, an algorithm runs on the entire database and looks for common patterns of two or more characters occurring frequently and replaces them with short codes. The Cost of a look up tables is added to this scheme because every new code assigned must be saved to the look up table which helps in decoding strings into original format. In some cases pattern substitution gives better compression ratio than null suppression encoding because pattern substitution contributes to compression with zeros and blanks.

Differential compression

Differential compression looks for a relationship between previous character and new character sequence whether or not is has any similarity with present. This technique is useful when we have telemetry data with a uniform size vary less. In ideal situation one can achieve compression up to 98% depending upon type of data [16].

Statistical or variable length encoding

Variable length encoding is also termed as Huffman encoding. Huffman encoding is based on the assignments of codeword's to upcoming stream of messages and assigning is based upon the probabilities with which the source message appears in the message ensemble. An interesting aspect of Huffman encoding is the assigning the codeword's to the similar message stream, assigning smaller codeword's to the frequently occurring messages and longer codeword's to smaller probabilities [17].

Static huffman encoding

Static Huffman encoding has two passes. By one pass it computes probabilities and assigns mapping to commonly occurring sub messages and in the second pass it encodes. In the first pass of static coding codeword's messages are passed by encoder to decoder [18].

Dynamic huffman encoding

Dynamic Huffman encoding maps from a set of messages to the set of code-words where the set of code-words change over time. The assignment of code-words to messages is based on the values of relative frequencies of occurrence at each point in time. Suppose we have a message X in the source message that is assign a short codeword in the initial transmission due to frequently occurrence in the message at the start of the transmission. Later when more high probability messages appears to occur with higher frequency that short codeword's message X will be assigned to another high probability messages and X will assigned with a longer codeword. Dynamic keyword can be replaced with adaptive keywords because they adapt to change over time. All of the adaptive methods are one-pass methods; only one scan of the ensemble is required. During transmission one pass methods encoder defines and redefines the mapping dynamically, during transmission [21].

Hybrid huffman encoding

As the name refers, hybrid is in between the static and the dynamic Huffman encoding, being neither fully static nor completely dynamic. In a hybrid model, the coder and the decoder maintain identical codebooks of k static codes. A sender must send previously agreed upon codes and to the receiver [18].

Cormack scheme

Cormack scheme is designed to compress database file. This scheme is a part of IBM as Information Management System (IMS), which compresses each and every individual record and then expansion is performed each time a record is retrieved. Context information used by compression routine is limited to a single record as records may be retrieved in any order. For this routine to be applicable for any database, it must be able to adapt to any arrangement of the record. The compression routine in IMS follows the hybrid model because of redundancy by using different code-words scheme for different record. Database records are usually spread as heterogeneous collections of small fields indicating local properties of information are more worthy than global characteristics [16].

Universal codes and representations of the integers

Universal codes are defined as follows: if map source messages to codeword and the resulting codeword length is bounded by c1 (H+c2), which is given a source with nonzero entropy, a universal code achieves the average codeword length which is at most a constant time the optimal possible for that source. Possible compression achieved by this scheme depends upon the magnitudes of the above mentioned constant that is c1 and c2. As the average codeword length approaches entropy, the universal code with c1=2 is asymptotically optimal [22].

LZW compression

LZW compression converts variable length strings into fixed length codes. The symbol strings are selected so that they all have an almost equal probability of occurrence [23].

Run length encoding

Run length encoding is used in business data compression. Types of redundancy present in business data includes runs of zero in numeric fields, sequence of blanks in alphanumeric fields and fields having some records in one attribute and null in other in same rows. By run length encoding sequence of zeros or null can be compressed and this can be achieved through the use of presence bits [24].

Dictionary substitution

Dictionary substitution is used in situations where limited set of attribute value exists. Dictionary substitution replace alphanumeric representations of information (like month, gender etc) which the few bits necessary to represent the limited number of possible attribute values [25].

4. COMPRESSION EVALUATION

Compression evaluation measures the reduction in data size. Most useful method for measuring compression in terms of reversible and irreversible performance is the compression ratio [26].

$$\mathrm{CR} = \frac{Size\ of\ data\ without\ compression}{Size\ of\ data\ after\ compression}$$

Since the unit for data varies from application to application the average data size reduction is usually measured by a compression ratio.

$$Reduction = \left(1 - \frac{1}{CR}\right) *\ 100\%$$

Another important factor that is always considered when performing a compression operation is complexity of compression algorithm. Different compression algorithms achieve different compression ratios depending upon different datasets.

5. DATA COMPRESSION ACCEPTANCE FACTORS

In spite of proven facts that compression reduces data storage and provides other benefits, data compression is still not given the appreciation and this might be because of following factors [15].

Misunderstood compression limits

Designer of databases misunderstand compression because of a lack of knowledge on how much compression is possible for given set of data.

System complexity

There is no doubt that compression adds complexity because of encoding and decoding techniques which need to be used on compressed data. Designer hesitates to accept additional complexity without upfront benefits.

Mathematical mystique

Much of available literature on data compression is about mathematical mystique. Designers will usually avoid those areas in which they feel uncomfortable.

6. PROPOSED DATA COMPRESSION IN EMS

To evaluate the compression techniques we applied DCC on two datasets: Consumption data from authors lab collected through eguage meters [27] and Reference Energy Disaggregation Data Set (REDD) [6]. Eguage is smart meter infrastructure which provides aggregated consumption up to granularity of 1 Hz. Egauge meters were deployed in the authors lab to measure consumption. This data forms the first dataset for evaluation. The second data set was collected by MIT CSAIL group as a reference for energy load disaggregation (REDD). The data was collected from six houses at granularity of 16 KHz.

6.1 ANALYSIS

As discussed above the compressed form of data generally achieves better compression for text around 50-60% and it takes less time to compute. Arithmetic coding has been reported to reduce file size between 12% - 73% of its original size. Compression through Huffman coding reduces the size of student record database by about 42% [21]. Above all the technique shows that if we are able to reduce size of storage then processor required doing less work to fetch data from storage disk.

However, compression is smart grids is a very atypical problem with very specific load profiles. The load variation over time is mostly constant with some strong variations when devices transition between states.

Abbreviations scheme work best where we have strings of full words so we store its abbreviated values, but in smart grid every record contain integers and floating points due to number factors. Therefore, an abbreviation scheme is not suitable to compress smart meter data.

Huffman encoding assigns code-words to symbol by looking at probabilities of occurring. The look up cost for a code value table become high if we have different values of set, smart data contains number which increment on every seconds so to maintain code value to incremental number is a drawback because look up table cost become higher than the compression ratio.

One of property of smart data is that the data chunks are of uniform size and tend to vary relatively slowly for each meters and devices, due to the slow change for each device record differential compression can be run to compress data on smart data center. To achieve better compression in smart grid data we propose the following compression algorithm for smart grid data based on the unique features and attributes of smart grid data. The algorithm is adaptation of differential compression to store fast stream smart data in compact form:

```
Algorithm 1:
Step 1: Get incoming stream of data from a device.
Step 2: Fetch the last save record of same device by comparing device ID.
Step 3: Result= previous_reading XOR upcoming_reading.
Step 4: if result return true do not save new record.
Step 5: Repeat step 1 for new stream of data from device.
```

Differential compression works in environments which have uniform data that vary relatively less, the same feature is present in smart data; advantage of using differential compression is to have reversible compression. Only a new record with a change in information to previous record with same meter id will be saved, in between same record which is duplicated with different timestamp is compressed. Differential compression achieved on REDD aggregated data is

Table 4 Differential compression on aggregate dataset

	House 1	House 2	House 3
No of rows without compression	152032	111671	66423
No of rows after compression	70059	7611	40868
Compressed ratio	2.17	14.67	1.62
Reduction in size	54%	93.2%	39%

If we compact the data on basis of granularity by changing the data semantics, on basis of granularity compression ratio achieved on different Eguage meter [27] is

Table 5 Compression Ratio (CR) on basis of granularity

Conversion		Dev 1	Dev 2	Dev 3	Dev 4
From	To				
Sec	Min	2.7%	2.7%	2.6%	2.7%
Min	Hr	36.3%	36.4%	36.2%	36.3%
Hr	Day	24.0%	24.0%	23.8%	23.7%

Changing the semantics on the basis of granularity provides a decent compression ratio. Average reduction size achieved by compressing Eguage smart meter data on basis of granularity is

Table 6 Reduction of size on basis of granularity

Conversion		Reduction Size
From	To	
Sec	Min	63%
Min	Hr	97.3%
Hr	Day	95.8%

So, by converting per second data into different aggregations yields good compression ratio which saves storage and increases performance, like a home user who wants to see the report on a daily basis can receive a report from per day aggregation data instead of a dataset from per second repository. Additionally, home consumers also save run time report generation algorithm cost for converting per second data into per day data.

As shown in figure 3 every stakeholder requires different granularity of data for different properties of smart grid. To find fault in a network topology different granularity of dataset is required. To show a report to the management so that they can take decision, a different granularity of dataset is required and so on, so one of solution is to have a curated database [28] for different features of smart grid which can achieve single responsibility principle. Through the single responsibility, principle load from primary database shifts, each stakeholder query to related dataset instead of query from centralized smart data repository.

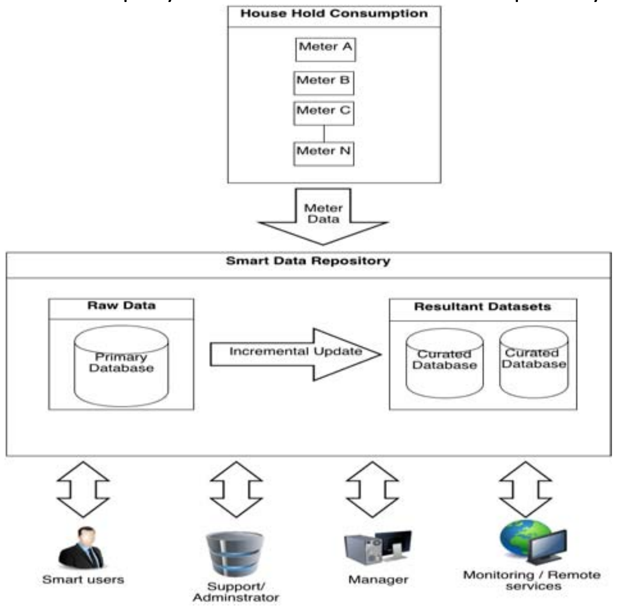

Figure 3 Derivative Approach

There is a background service which runs at regular interval and convert data according to feature set of smart grid and store it into curated data repository by which load of queries distributed and performance of query increase.

7. CONCLUSION AND FUTURE WORK

In this paper we have provided a comprehensive survey of data management techniques. We have provided an analysis for application of different techniques for data management in smart grids and have discussed the applicability of different algorithms for smart grids data. We have also shown the benefits that can be gained by applying a variation of differential compression on the energy load data. We have shown how energy community can effectively store large flow of data, by removing redundant data to save storage and increase performance. By selecting right compression algorithm and compaction techniques we can achieve more than 50% reduction in size of smart grid data can. Our next step is to apply other related compression algorithms to improve the results as well as expand the evaluation datasets to arrive at a more concrete result.

ACKNOWLEDGEMENT

This work is in part supported by grants from the Department of Computer Science at LUMS and ICT Research and Development Fund of Pakistan.

REFERENCE

[1] Berman, P., Calinescu, G., Shah, C., & Zelikovsky, A. Efficient energy management in sensor networks. Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing 2005; 2:71-90.
[2] Massoud Amin, S., & Wollenberg, B. F. Toward a smart grid: power delivery for the 21st century. Power and Energy Magazine IEEE 2005; 3(5):34-41.
[3] Wu, Y. N., Chen, J., & Liu, L. R. Construction of China's smart grid information system analysis. Renewable and Sustainable Energy Reviews 2011; 15(9):4236-4241.
[4] Ali, U., Rana, Z. A., Javed, F., & Awais, M. M. EnerPlan: smart energy management planning for home users. In Neural Information Processing. Springer Berlin Heidelberg 2012; 543-550.
[5] Javed, F., Arshad, N., Wallin, F., Vassileva, I., & Dahlquist, E. Forecasting for demand response in smart grids: An analysis on use of anthropologic and structural data and short term multiple loads forecasting.Applied Energy 2012.
[6] J. Zico Kolter, Matthew J. Johnson. REDD: A Public Data Set for Energy Disaggregation Research. In proceedings of the SustKDD workshop on Data Mining Applications in Sustainability 2011; 1:6.
[7] E. Michael Azoff. Neural network time series forecasting of financial markets. John Wiley & Sons, Inc 1994.
[8]http://docs.oracle.com/cd/B19306_01/server.102/b14237/limits002.htm#i287915, 2013-01-05
[9]http://docs.oracle.com/cd/B19306_01/server.102/b14237/limits003.htm, 2013-01-05
[10] Michael J. Knieser, Francis G. Wolff, Chris A. Papachristou. A technique for high ratio LZW compression. In Proceedings of the conference on Design, Automation and Test IEEE Computer Society 2003;1-10116.
[11]http://dev.mysql.com/doc/refman/5.6/en/table-size-limit.html, 2013-01-01
[12] Debra A. Lelewer and Daniel S. Hirschber. Data compression. ACM Computing Surveys (CSUR) 1987;19(3): 261-296.
[13]http://msdn.microsoft.com/enus/library/ms143432.aspx, 2013-01-01
[14]http://www.dbta.com/Articles/Editorial/Trends-and-Applications/Managing-Time-Series-Data-with-Informix-76454.aspx, 2013-01-05
[15] Mark A. Roth, Scott J. Van Horn. Database Compression. ACM Sigmod Record 1993; 22(3):31-39.
[16] G.V Cormack, R.N.S Horspool. Data Compression Using Dynamic Markov Modelling. The Computer Journal 1987, 30(6):541-550.
[17] Gilbert, E. N., and Moore, E. F. Variable-Length Binary Encodings. Bell System Tech 1994;38:933-967
[18] Ross williams. Adaptive data compression. Vol. 110: Springer; 1991.
[19] Mohamed elbaradei. Tackling the global energy crisis. Indian Journal of Power and River Valley Development 2009; 59(1): 1.
[20] Hassan farhangi. The path of the smart grid. Power and Energy Magazine, IEEE 2010; 8(1):18-28.
[21]Knuth, D. E. 1985. Dynamic Huffman Coding. J. Algorithms 1985; 6(2):163-180.
[22] Elias. Universal Codeword Sets and Representations of the Integers. Information Theory, IEEE Transactions on 1975; 21(2):194-203.
[23] H.K Reghbati. An overview of data compression techniques. Computer 1981;14(4):71-75.
[24] David salomon. Data compression by david salomon. 3rd edition: Verlag New York Incorporated; 2004.
[25] Miklos ajtai, Randal burns, Ronald fagin, Darrell D.E. Long. Compactly encoding unstructured inputs with differential compression. Journal of the ACM (JACM) 2002; 49(3):318-367.
[26] Goetz Graefe, Leonard d. Shapiro. Data compression and database performance. In Applied Computing 1991;22-27.
[27] http://eguage.net/, 2012-12-21
[28] Peter buneman, james cheney, wang-chiew tan. Curated databases. In Proceedings of the twenty-seventh ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems 2008; 1-12.