# Hierarchical Cloud Storage Engine

Andrei Dobrin, Cristian Dragana, Grigore Stamatescu, Valentin Sgarciu
Department of Automatic Control and Industrial Informatics
University "Politehnica" of Bucharest, Romania
Email: {andrei, cdragana}@imtt.pub.ro, grigore.stamatescu@upb.ro, vsgarciu@aii.pub.ro

***Abstract*—Cloud computing and cloud storage systems are being used more and more in a variety of domains, from everyday user applications like healthcare monitoring systems and intelligent buildings to military devices. The deployment of these frameworks also enables modern control and automation paradigms found in cyber-physical systems and Industry 4.0. Driven by exponential decreases in computing and storage costs along with high bandwidth, low-latency communication networks, cloud-based infrastructures have increasingly been adopted in large-scale industrial applications. Our paper proposes hierarchical or key-value databases for systems that gather a very big number of data items from a variety of sensors, with focus on smart building and smart city scenarios. We discuss GT.M as a key-value database engine, optimized for transaction processing with a very high throughput. Results of a simulation-based study comparing hierarchical and relational database performance for several types of operations are presented.**

**Keywords: Cloud engine system; GT.M; key-value database; hierarchical database; NoSQL database; big data; data mining**

## I. Introduction

In this article we continute the work of [1] where we studied the challanges of storing data in Cloud platforms and propose a new solution for storing big amounts of data from WSNs in the cloud with the idea of using hierarchical databases, more concretely, GT.M as the engine for the database. To test if this is a viable solution, we compared the said engine with PostgreSQL. The objective of comparing GT.M with PostgreSQL is to test if a hierarchical model is better suited to store data from a large number of sensors. A key-value database can be better than a relational one because it is specially optimized for a high number of transactions. The data used for this experiment was from a building management system.

Of course, after storing the data we must interpret and understand its characteristics. Big data is a relatively new concept that is having a big impact in multiple research areas and in multiple industries. This domain is important since it can reveal totally new aspects of systems from customer buying trends (how to group products in shops to increase selling of products) to sentiment analysis on Twitter and even the decoding of the human genome. With data mining we can gather a lot of information that previously we didn't even knew was there in the first place and we can make systems more efficient, more precise and more robust since we can detect current challenges, where potential challenges might arise and of course where we can improve the system. If one wants to understand a system, one needs to gather as much data as it can from it, in all sorts of stages and from all types of "areas". To do all this we need sensors, a lot of them in order to be able to replicate the state of the system as accurately as we can.

In this paper section II gives an overview of what other researchers have done in the domain of Key-Value databases, section III gives an overview of the two main types of databases (hierarchical and relational databases), each type's characteristics and the advantages and disadvantages of one over the other. Section IV studies how the information can be analysed with the help of big data and data mining concepts. Section V shows the experimental results of using GT.M versus PostgreSQL as a database where the benefits, but also the potential challenges of GT.M are presented. The last section presents the conclusions and details future steps to evaluate GT.M as viable solution for cloud storage in a smart building environment.

## II. Related work

As [2] strongly emphasizes, an important factor, that can make this type of project successful or not, is how the database is designed. To handle the information on a database, researchers created a high-level programming model called MapReduce with which high volumes of data can be processed by using parallel and distributed computing on large clusters or groups of nodes. This can be used in GT.M with the idea to try to store the information on multiple databases at the same time. With clustering of information processing times can be reduced if the architecture is well designed.

Hierarchical databases are used in a variety of domains for their advantages. In [3] such an hierarchical database is used to store location data for personal communication services. Their scheme allows to dynamically adjust user location information distribution based on the patterns of the mobile terminals (MT). A unique distribution strategy is determined for each MT in a way that both assures a complete coverage and an optimization of resources. Another important aspect in the scheme is that location pointers are set up at determined locations to indicate the current location of the MTs. This was a necessary step to effectively reduce the signaling and database access overhead for location registration and call delivery. Moreover, the required processing is computed by a distributed network of registers which, in the end, offers the benefit of removing the necessity to use centralized coordination.

## III. Hierarchical databases

Most mainstream database systems like SQL Server, MySQL, DB2, Oracle and others are of a type that is called relational system. This means that we have tables of data and between them we have relationships with the use of keys. This is the prefered type of system because it is sometimes easier for people to understand its concept and implement it. This is done by creating a database with all the information split between tables and then inserting small pieces into each table. Hierarchical databases on the other hand are a different type of system because they store the information, for a particular segment of the data that needs to be stored, or better said for a chunk of a system in one table. This stored information can be separated by keys, but in a different way than in the relational model, their structure is like the one of a tree. With the main key as the root, other keys as leaves and the information stored beneath each leaf.

### A. Trees in data structures

A tree is a set of elements connected by straight lines in a non-linear manner and with no closed loops. If a tree has n nodes then, it contains n-1 graph edges. The points of connection are known as children nodes and the segments as branches. The last nodes being called leaves. A tree can either have a central node if it has the structure of a centered graph or have a root node if each other nodes are one graph edge further away from the one considered as the root.

To generate a rooted tree we have this mathematical function:

$$T(x) = \sum_{n=1}^{\infty} T_n x^n = xexp \sum_{r=1}^{\infty} \frac{1}{r} T(x^r)$$

This function is related to the one used for generating a number of unrooted trees by:

$$t(x) = \sum_{n=1}^{\infty} t_n x^n = T(x) - \frac{1}{2}[T^2(x) - T(x^2)]$$

From [4] we know that:

$\lim_{n\to\infty} \frac{t_n n^{5/2}}{\alpha^n} = \beta$

Where the two constants are given by:

$\alpha = \lim_{n\to\infty} \frac{T_n}{T_{n-1}} = 2.955765$

$\beta = \frac{1}{\sqrt{2\pi}}[1 + \sum_{k=2}^{\infty} T(\frac{1}{\alpha^k})\frac{1}{\alpha^k}]^{3/2} = 0.5349485$

### B. Hierarchical database vs relational database

Even though key-value databases are less common, they are used in a variety of systems in which they work better than relational databases for those types of applications. For example, one type of system in which relational databases do not work very well, even if there exist these types of systems from Oracle, are core banking systems. In core banking systems, there is a need for a database engine that supports multiple transactions per second for each client and moreover, in some periods of the year like quarter interest calculation and financial year end, the number of transactions posted is of a very big order. This can affect a lot how the system behaves in such an event and a lot of things can happen that can temporarily disrupt its functionality or even stop it which is of course unacceptable.

A good system design can prevent this, but in the case of a key-value database it is easier to support this kind of transactions because the database can be implemented in such a way that all the information which is needed for these operations can be stored in one single table or in the worst case in a few tables that relate to each other with exact pointers. So, if the database has a good design and all operations which affect its state or the query run on it does not need a lot of additional operations which have a big impact on the processor, this type of database is much faster when it comes to processing multiple transactions per second from multiple sources.

A visual representation of the main difference between the two types of databases can be seen in the below figure 1:

As it can be seen in the above figure, a key-value database looks like an index in a relational database and it does have a lot of the features of an index. At a first glance, the first type of database can be easier to understand and to grasp the different pieces of information which it stores then the second type of database, but this is just at the beginning, before we understand the hierarchical model since this model can be constructed in a way that resembles the real system more clearly. In time, this can be much easier to work with.

### C. Advantages and disadvantages of hierarchical databases

At a first glance, Relational databases are superior to NoSQL ones and this is because of three reasons:

- Difficult or impossible to extend the data model if the applications that uses it are constructed in a very tight way since they will also need to be modified;
- It is not easy to force consistency between all logical entities;
- Queries can take a lot of time if they are complex and no keys or indexes are used or they do not follow the model of the database.

Even with these disadvantages, hierarchical databases are faster than relational databases. If they are designed for a single type of application they are a better choice, meaning we do not intend to use the same database for other applications which were not designed to work with this model. Moreover, in the case of multiple transaction per second relational databases have a hard time coping with all the data whereas key-value databases can process them instantly.

### D. GT.M

GT.M is a high-throughput key-value database engine optimized for transaction processing, it is a NoSQL database type. It is an implementation of ANSI standard M for several UNIX systems and Linux, but now a version for Windows has been developed. Since 2000 the database is open source and is maintained by FIS Global, a company that offers financial applications and data storage solutions, since it is being used for the FIS Profile banking application which powers many banks on a world-wide coverage, the biggest being ING DIRECT in Spain, Romania, France, Italy, Holland, India, UK.

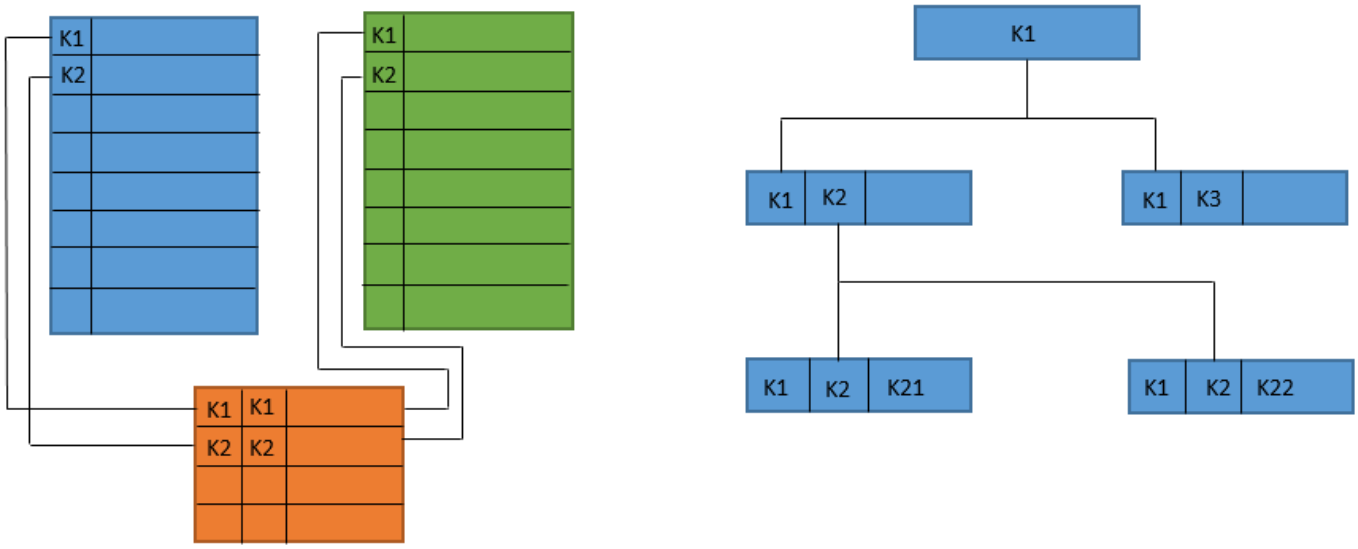


Fig. 1. Relational (left) vs. hierarchical (right) databases

Also, the database is used in other industries like healthcare, transportation, manufacturing and others. [5]

GT.M is used mainly as a core banking system and as it is noted in [6], the quality and availability of service is very important for such a system. If this is not the case, the system cannot perform well and customer satisfaction is out of the question. In today's era customers have great expectations from their banks and they demand 24/7 100% availability of service. Since GT.M is almost 50 years old and it is used in some of the biggest banks, shop chains and insurance companies it proves its capabilities and advantages. This makes it a possible candidate for a cloud engine server that is used to store sensor data since it can process transactions at a very high speed.

In hierarchical databases, natively, you can't use relational queries. Instead an interface has to be created in order to support this type of language. This is not a simple task since the queries might be very complex or even if the query looks simple, the actual operations on the database can be very hard on the disk and processor if the interface is not correctly designed. Therefore, the interface to a hierarchical DBMS poses a difficult translation challenge [7]. When creating such an interface these guidelines have to be considered:

- Include all SQL features that can be translated to DL/1 features to make use of efficient IMS capabilities;
- Include SQL features that are frequently used in applications so that most of the applications can be covered by the defined SQL subset;
- Exclude SQL features that are seldom utilized to avoid the implementation of a relational DBMS and the run time overhead of a large system. [8]

## IV. Results

To test if GT.M is faster and if it is with how much, a comparison with PostgreSQL has been done with a set of data from the temperature sensors used in a building management system. The building is the new research facility, built in 2016 in the Polytechnic University of Bucharest campus. The data is represented by two columns: one with the date-time value when the data was recorded and the second with the actual value read by the sensor.

The simulation was done on the same machine, in the same conditions, at the same time by importing in PostgreSQL a file and then in GT.M the same file. The machine had Ubuntu 14.04 since it is the best version of Ubuntu that can run GT.M at the current moment. The import in both databases was done from command line to minimise the differences.

As it can be seen in figure 2, as the record number per seconds increases so does the gap between the two databases types. For example, GT.M is 12 times faster than PostgreSQL for a file with more than 4.000.000 records. This is a very important gap, but it is also a very important observation: such a database has benefits only for very large sets of data. If the dataset per second that needs to be recorded in the database is relatively small, the benefits of a hierarchical database are not easily seen.

Testing reveals a another challenge with the hierarchical database: the algorithm must assure that the received data has different values for the first column because if they do not (the sensors send data very fast and they cannot include milliseconds in the date-time value for example, only the last value of the sensor will be stored in the database.

One of the benefits of GT.M is that the code is "embedded" in the database so we can program everything in the database, even every transaction type individually. Adding a programmed count as an additional column in the database (to allow duplicates the first file column) increases the transactioning time by a big coefficient as it can be seen in Figure 3

Even though this might seem like a disadvantage of a

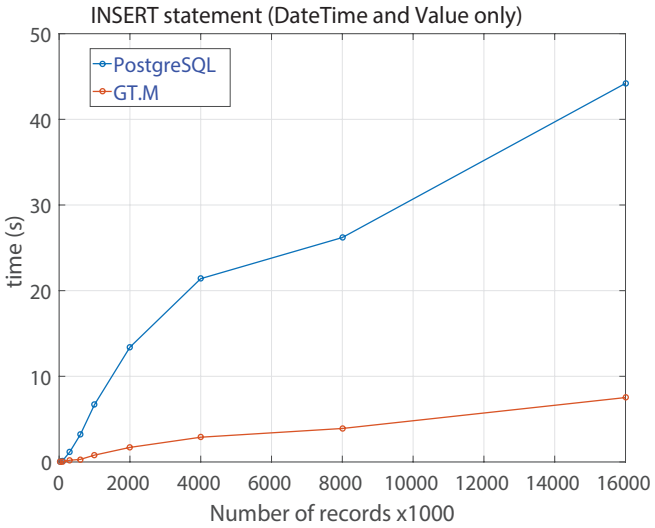


Fig. 2. PostgreSQL vs GT.M transactions time

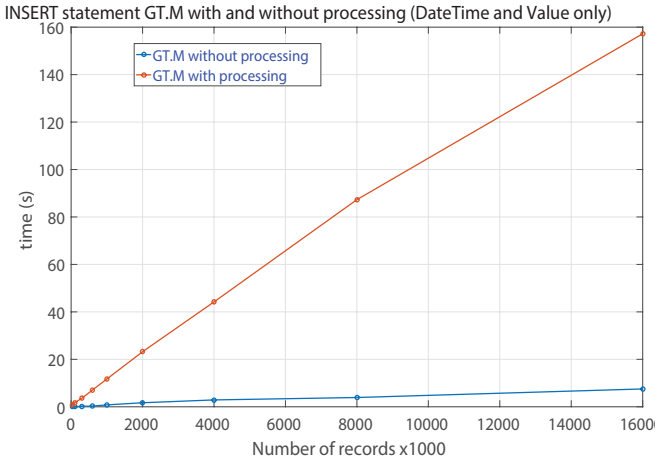


Fig. 3. GT.M with and without transaction processing

hierarchical database, it can be used in as an advantage for the end users by making a lot of the processing of the incoming information before saving the data in the database and not after the data is recorded, reducing the time needed to do some of the operations when we retrieve the information to present it to the user. Operations on the same dataset, under the same keys (for example, the data stored for the same sensor) is much faster in a hierarchical database since retrieving all previous data is very fast.

In an hierarchical database the data can either be saved as separate columns or in the same value with a chosen separator. This is useful since saving a column to the hierarchical database is actually an operation. For example, if we want to save four columns:

- Identifier: an ID or other type of unique identifier;
- Moment: Date-time value of the time the sensor read the data from the environment;
- Value: Actual data value;

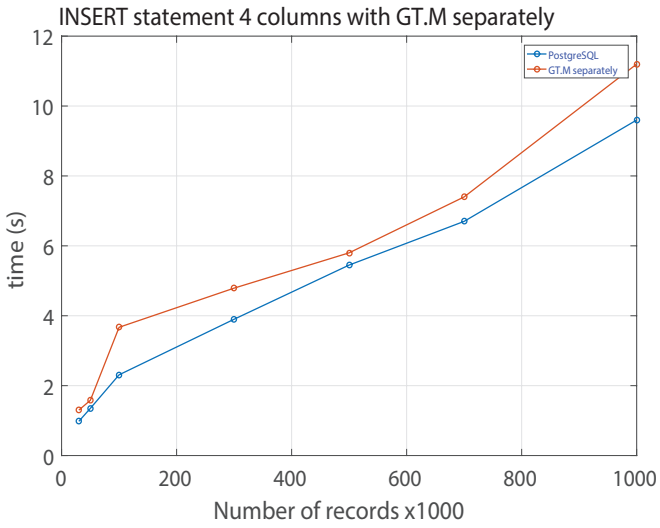


Fig. 4. INSERT statement 4 columns with GT.M values separately

- Description: Other relevant data that we might want to save.

We could do it in the form of:

- SensorData(SensorName,Identifier,"Moment") = Moment;
- SensorData(SensorName,Identifier,"Value") = Value;
- SensorData(SensorName,Identifier,"Description") = Description;

In which case we would have to do three saving operations on the GT.M table. On the other hand if we would save the data like this:

- SensorData(SensorName,Identifier) = Moment—Value—Description;

We would gain a lot of transaction time as it can be seen in Figures 4 and 5:

As it can be seen in the first Figure, the difference has dropped significantly between GT.M and PostgreSQL. That is because even saving the data in the save field with a chosen separator a little processing of the transaction is required and an additional if statement is mainly responsible for the big difference.

To better observe the differences between the two database engines a test for the "SELECT" and "DELETE" queries has been done with the "WHERE" clause "Value¿60.0". The results can be seen in Figures 6 and 7:

All the results can be better seen in the below table with the average improvements of GT.M over PostgreSQL:

## V. Application to Big Data and Data Mining

The amount and types of digital data is increasing almost exponentially comparing with a few years back. All intended users need advanced data analysis tools and services and scalable architectures to be able to extract useful information from big data repositories. Cloud computing systems offer such support for addressing both the computational and data

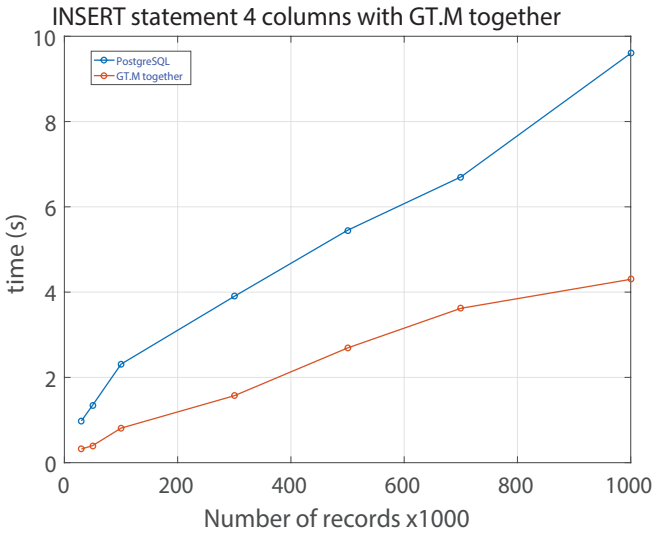


Fig. 5. INSERT statement 4 columns with GT.M values together

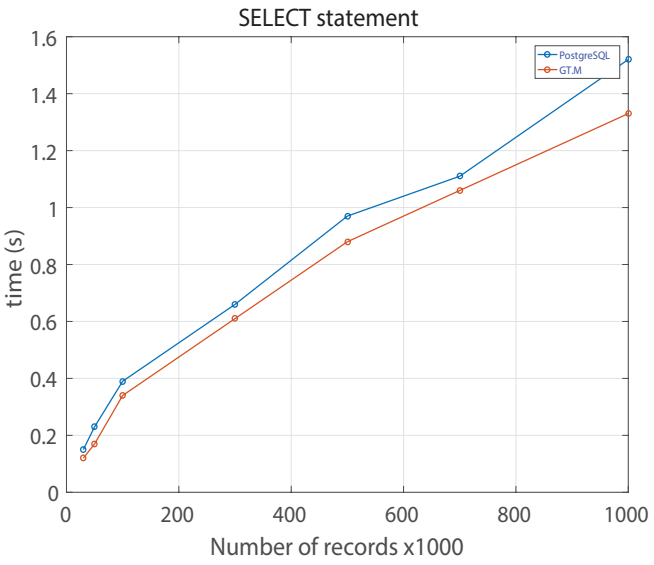


Fig. 6. SELECT statement

TABLE I
ZONES

| Operation | Average improvement coeficient |
|---|---|
| Simple INSERT | 8.08 |
| INSERT with GT.M stored separately | 0.8 |
| INSERT with GT.M stored together | 2.55 |
| SELECT | 1.16 |
| DELETE | 0.53 |

storage needs of big data mining and parallel knowledge discovery applications. [9]

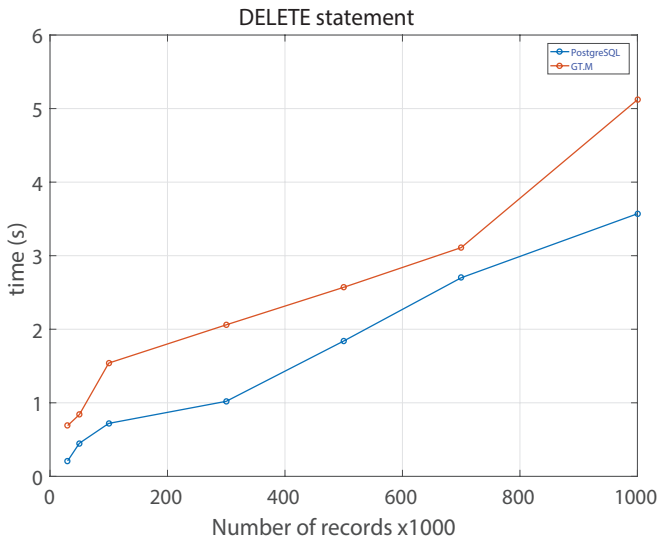


Fig. 7. DELETE statement

### A. Big data

Big data means looking into information that is already at your disposal, but looking at it from multiple, different perspectives. This gives us the opportunity to understand things that previously were not even thought of. The concept can be implemented in multiple domains like banking, financial, building management systems, marketing, telecommunication, medical, life science, healthcare, social data and many more.

One implication of big data is that humans, especially non-scientific people, have a really new and totally different concept of what the data that is around them every day means. Where formerly everything was signal, now 99% is noise, which can lead some to be overwhelmed and make it non-cost-effective to keep analyzing the said data, especially if the filter of the information is not adequate. [10]

One important aspect that big data can be used in the domain of databases which store sensor data is that not only real-time data is stored, but also a lot of information prior, during and after a fault in the whole system [11]. With Big Data techniques we can better understand what went wrong and how it went wrong by observing all the components of the system. This brings the advantage of faster locating the challenge since it could have started in a place we would not think to look in the beginning.

As [12] states, big data is characterized by 3 Vs: Volume, Velocity, and Variety. GT.M is a very good candidate to store information since it can support all these 3 important characteristics. It can deal with a very big and varied amount of data in a short period of time.

### B. Data mining

Data mining has given impressive results in almost every domain (e.g. healthcare, wireless sensor networks, social networks, etc) with the development of various algorithms. The first limitation of data mining is that each process needs its

own algorithm. Even though some parts of an algorithm can be used in others every one has its inherent limitations. The application domain and the actual data influence very much the choice as well as the performance of any data mining, machine learning or statistical algorithm. [13]

Since GT.M is also a compiler for the M language which was developed once with the database engine and it retrieves data very fast, algorithms should be implemented in two ways, both on the system itself and outside in the UI application, in a general programming language (JAVA, C#) which brings other challenges [14]. There different types of data mining algorithms like:

- Classification algorithms: Based on the characteristics of some already known variables, others can be approximated;
- Regression algorithms: Based on known data, some continuous numerical variables, such as profit or loss, can be predicted;
- Segmentation algorithms: By analysing the data, we can group different components that have similar properties into groups, or clusters;
- Association algorithms: Correlations between different attributes in a dataset can be found with this type of algorithm.
- Sequence analysis algorithms: summarize frequent sequences or episodes in data.

## VI. Conclusions

Using key-value databases to store a large number of sensor data can be a better option than using a traditional database if the hierarchical database is well designed since key-value databases are much better at processing transactions. So, when it comes of a very big number of nodes, that send a big quantity of data, multiple times per second, a hierarchical database is a better option since the commit time is smaller an aspect that is noticeable when there is a very big number of elements that want to send and store their data in a cloud database. GT.M is an open source, high-throughput key-value database engine optimized for transaction processing. Even if it has not been used in these kind of scenarios, looking at its characteristics and advantages it is clearly a good candidate for a cloud storage system since it is being used with great success in some of the world's most important banks, hospitals and shops. There are still a lot of aspects that need to be researched by implementing such a system and testing it with real sensor data. Some of the aspects identified until now are what types of sensor data can it store with great accuracy, how easy it is to retrieve the data in an ergonomic manner, what calculations can be done in the system, since it has an own language and others.

The next steps that will be taken in investigating the possibility of using GT.M as a cloud storage engine and designing a database that will give the opportunity of storing data from a large number of sensors in the real world are to use a server accessible from anywhere, install GT.M on virtual machine with a distribution of Linux on it, create the necessary interface for clients to be able to insert data and then query it. Finally, connect the database to a real life building management system and feed it sensor data in parallel with other database engines to analyse the results.

Different types of data mining algorithms will have to be implemented and the results analysed in order to determine the best choice for different types of applications in GT.M.

An important aspect in data mining is privacy. Since the great popularity that this domain has started to get in the last years has also attracted an unwanted attention from people that might want to exploit the information stored on such cloud platforms. An emerging research topic in data mining, known as privacy preserving data mining (PPDM) [15]. In the future, different aspects of how retrieval of data has to be made in a way that assures sensitive data is not compromised.

## References


[1] Andrei Dobrin, Grigore Stamatescu, Cristian Dragana and Valentin Sgarciu, *Cloud Challenges for Networked Embedded Systems: a Review*, 2016 20TH INTERNATIONAL CONFERENCE ON SYSTEM THEORY, CONTROL AND COMPUTING (ICSTCC), 2016

[2] Carson Kai-Sang Leung, Richard Kyle MacKinnon, Fan Jiang *Reducing the Search Space for Big Data Mining for Interesting Patterns from Uncertain Data* 2014 IEEE International Congress on Big Data, 2014

[3] Joseph S. M. Ho and Ian F. Akyildiz*Dynamic Hierarchical Database Architecture for Location Management in PCS Networks*, EEE/ACM TRANSACTIONS ON NETWORKING, 1997

[4] Knuth, Donald E.; Saitou, Hiroaki; Nagao, Takahiro; Matui, Shougo; Matui, Takao; Yamauchi, Hitoshi, *The Art of Computer Programming. - Volume 2, Seminumerical Algorithms*

[5] *FIS Profile documentation*

[6] M. Mutingi, H. Mapfaira, N. P. K. Moakofi, S. A. Moeng, C. Mbohwa *Simulation and Analysis of a Bank Queuing System* 2015 International Conference on Industrial Engineering and Operations Management (IEOM), Dubai, 2015

[7] Chin-Wan Chung and Kenneth E. McCloskey *Access to Indexed Hierarchical Databases Using a Relational Query Language*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 1993

[8] Chin-Wan Chung Kenneth E. McCloskey *A RELATIONAL QUERY LANGUAGE INTERFACE TO A HIERABCBICAL DATABASE MANAGEMENT SYSTEM*, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 1989

[9] Domenico Talia DIMES *MakingKnowledge Discovery Services Scalable on Clouds for Big Data Mining* ,University of Calabria & DtoK Lab Srl Rende (CS), Italy, 2015

[10] Melanie Swan Contemporary Philosophy MA Candidate *Philosophy of Big Data Expanding the Human-Data Relation with Big Data Science Services* 2015 IEEE First International Conference on Big Data Computing Service and Applications, UK, 2015

[11] Jinxin Huang, Lin Niu, Jie Zhan, Xiaosheng Peng, Junyang Bai, Shijie Cheng *Technical Aspects and Case Study of Big Data based Condition Monitoring of Power Apparatuses* , 2014 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC), China, 2014

[12] Kyounghyun Park, Minh Chau Nguyen, Heesun Won *Web-based Collaborative Big Data Analytics on Big Data as a Service Platform* Big Data SW Research Department, Electronics and Telecommunication Research Institute, 2015 17th International Conference on Advanced Communication Technology (ICACT), South Korea, 2015

[13] Archana Purwar and Sandeep Kumar Singh *Issues in Data mining: A comprehensive survey*, Department of Computer Science /Information technology Jaypee Institute of Information Technology, Noida India, 2014

[14] Oana Chenaru, Grigore Stamatescu, Iulia Stamatescu and Dan Popescu, *Towards cloud integration for industrial wireless sensor network systems*, Advanced Topics in Electrical Engineering (ATEE), 2015 9th International Symposium, 2015

[15] Lei Xu, Chunxiao Jiang, Jian Wang, Jian Yuan AND Yong Ren*Information Security in Big Data: Privacy and Data Mining*,Department of Electronic Engineering, Tsinghua University, Beijing 100084, China, 2014