Measurement 151 (2020) 107218

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

Information management in IoT cloud-based tele-rehabilitation as a service for smart cities: Comparison of NoSQL approaches

Antonio Celesti [a,g,*], Aimé Lay-Ekuakille [b], Jiafu Wan [c], Maria Fazio [a,e], Fabrizio Celesti [d], Agata Romano [f], Placido Bramanti [e], Massimo Villari [a,e]

[a] Department of Mathematical, Computer, Physic and Earth Sciences (MIFT), University of Messina, Messina, Italy
[b] Department of Innovation Engineering, University of Salento, Lecce, Italy
[c] School of Mechanical and Automotive Engineering, South China University of Technology, Guangzhou, China
[d] Department of Medicine and Surgery, University of Insubria, Italy
[e] IRCCS Centro Neurolesi "Bonino Pulejo", Messina, Italy
[f] Provincial Health Agency (ASP) of Messina, Messina, Italy
[g] On behalf of CINI Big Data Laboratory, Italy

ARTICLE INFO

Article history:
Received 28 December 2018
Received in revised form 14 October 2019
Accepted 25 October 2019
Available online 6 November 2019

Keywords:
Smart cities
Healthcare
Tele-rehabilitation
Sensor and actuators
Remote monitoring
Cloud computing
Internet of Thing
Big data
NoSQL DBMS
Measurement

ABSTRACT

Nowadays, recent advancements in ICT have sped up the development of new services for smart cities in different application domains. One of these is definitely healthcare. In this context, remote patient monitoring and rehabilitation activities can take place either in satellite hospital centres or directly in citizens' homes. Specifically, using a combination of Cloud computing, Internet of Things (IoT) and big data analytics technologies, patients with motor disabilities can be remotely assisted avoiding stressful waiting times and overcoming geographical barriers. This paper focuses on the Tele-Rehabilitation as a Service (TRaaS) concept. Such a service generates healthcare big data coming from remote rehabilitation devices used by patients that need to be processed in the hospital Cloud. Specifically, after a feasibility analysis, by using a Lokomat dataset as sample, we measured and compared the performances of four of the major NoSQL DBMS(s) demonstrating that the document approach well suits our case study.

1. Introduction

In recent years, governments have shown the need to create sustainable and technological advanced health systems. For example the European Commission have invested about 500 millions of euros in cooperation projects and actions, as outlined within the Europe 2020 strategy [1]. Tele-medicine is one of the major applications that positively can impact people's lives in a smart city. In this context, home assistance is a key area of great interest for saving financial resources dedicated to traditional hospitalization. In particular, the remote control and monitoring of rehabilitation facilities is becoming easier than in the past thanks to recent Information and Communication Technologies (ICT). Furthermore, an increasing interest is growing about remote sensing and big data processing thank to the advent of new cutting-edge Cloud computing, Internet of Things (IoT), and big data analytics technologies. Advanced tele-rehabilitation techniques applied in a smart city context allow the medical personnel, for example, checking up children suffering from either partially or fully palsy [2] as well as triggering nano-drugs within the human body [3]. Moreover, a smart city context makes easier post-treatment for special categories of patients, for example, those suffering from either temporary or permanently motor and cognitive disability who require to be continuously supported and guided [4,5].

This scientific work falls within the mission of IRCCS "Bonino Pulejo", a clinical and research center located in Messina (Italy) specialized in neuroscience for the prevention, recovery and treatment of serious acquired neuro-lesions. Among its objectives, IRCCS "Bonino Pulejo" aims at providing a Tele-Rehabilitation (TR) post-discharge service to its patients for motor and cognitive recovery.

* Corresponding author.
E-mail addresses: acelesti@unime.it (A. Celesti), aime.lay.ekuakille@unisalento.it (A. Lay-Ekuakille), mejwan@scut.edu.cn (J. Wan), mfazio@unime.it (M. Fazio), fabrizio.celesti@studenti.unime.it (F. Celesti), placido.bramanti@unime.it (P. Bramanti), mvillari@unime.it, mvillari@irccsme.it (M. Villari).

https://doi.org/10.1016/j.measurement.2019.107218
0263-2241/

Since providing rehabilitation in medical centers is limited by high costs and/or geographical barriers, a possible alternative is represented by the adoption of remote exercise-oriented treatments through home-based medical devices connected with the Hospital system by means of Cloud-based services. Also, many managers and experts believe that Cloud computing can improve e-health big data management by reducing capital expenditure (e.g., hardware, software, networking, licensing fees, technicians, etc) and therefore they encourage its adoption [6]. In fact, considering the new emerging Cloud computing technology combined with IoT and big data analytics solutions, this is possible by means of the development of Hospital Cloud providers offering different Tele-Health (TH) services including Tele-Rehabilitation as a Service (TRaaS). The benefit of TRaaS is twofold: on one hand pushing down clinical costs and on the other hand improving the quality of life of both patients and their families who are not constrained to long trips anymore. Moreover, TRaaS paves the way toward a new kind of healthcare services for citizens in a smart city.

According to such a scenario, patients who suffer of neurological diseases, can experience treatments directly in their own homes by means of home-rehabilitation devices equipped with sensors that send tele-monitoring big data over the Internet to a Cloud Hospital system which processes them. The "big" term is used in this context because such pieces of data are characterized by the well-known three "V(s)": Volume (a huge amount of health-care information have to be managed, stored and processed), Velocity (health-care data are produced quickly and transmitted in real-time) and Variety (health-care data come from heterogeneous remote medical devices in different formats).Therefore, clinical operators can receive tele-monitoring big data from remote patients and send them back real-time feedbacks regarding rehabilitation exercises, hence, dynamically improving therapies if required. In particular, tele-monitoring big data related to patients can describe their real-time health status, personal records, history of diseases, treatments, information on the surrounding environment, etc. Therefore, there is a concrete need of more flexible and efficient DBMS solutions. Despite of the popularity of relational databases founded on the well known Structured Query Language (SQL), scalable schemeless NoSQL solutions appears to be a promising alternative for an effective management of clinical big data in general [7].

In this paper, we specifically analyze different NoSQL solutions for the management of the tele-monitoring big data generated by a TRaaS supplied by a Hospital Cloud provider. In particular, in order to motivate the technological feasibility of TRaaS, firstly we provide an overview of the major IoT solutions for home tele-monitoring consisting of low-cost non-intrusive medical devices (e.g., bracelets, balances, or any other connected sensing device monitoring either the patient or the surrounding environment) equipped with an active broadband connection (e.g., mobile, Wi-Fi, Bluetooth, etc) [8] that allow patients to send tele-monitoring big data (e.g., real-time patient's speed, heart rate, respiratory rate, training load, single-lead ECG, etc) to a Hospital Cloud system. Thus, healthcare operators can remotely provide patients real-time feedbacks during exercises and "on-fly" adjust their therapies if required. Moreover, we discuss a tele-monitoring big data model starting from i) the analysis of hybrid storage solutions [9,10] that exploit both SQL and NoSQL strategies; ii) the analysis of the information format provided by Lokomat, i.e., one of the major medical devices used in many neuro-rehabilitation centers. We stress that a tele-monitoring big data model is strongly required for the following reasons:

- a TRaaS, due to its scalable nature, produces a huge amount of tele-monitoring raw data that rapidly grow up;
- due to the heterogeneous nature of home rehabilitation devices, tele-monitoring big data can frequently change: this means that new attributes can be suddenly created, modified or deleted;
- moreover, since rehabilitation home devices are heterogeneous, it is not possible to conceive in advance a static data model.

In the end, in order to understand which NoSQL DBMS better fits the requirements of our reference tele-monitoring big data model, we tested in the IRCCS "Bonino Pulejo" data center both NoSQL document, column, and graph approaches.

Hereby, we clarify that other well-known Cloud computing challenges that have been widely discussed in literature, such data security, data transmission, system scalability, etc are out of the scope of this paper.

The reminder of the paper is organized as follows. Section 2 presents a reference TRaaS scenario motivating its technological feasibility. Section 3 presents the research methodology and the proposed system. Section 4 shows the tele-monitoring big data storage experience and research results. In the end, Section 5 presents a discussion on experimental results and conclusions.

2. Tele-Rehabilitation as a Service (TRaaS)

In this Section, we provide an analysis of the state of the art in e-health and tele-monitoring big data management and present a reference TRaaS scenario, motivating the technological feasibility of home-rehabilitation devices.

2.1. Background and related work

Telemedicine is a widely debated topic [11]. Currently, we are observing an increasing interest of the scientific and industrial communities about possible Cloud computing and IoT applications in healthcare, especially focusing on tele-monitoring and health big data management.

A discussion about how to combine Cloud computing and IoT technologies in a medical monitoring and management environment is discussed in [12]. In particular, a remote monitoring Cloud Platform of Healthcare Information (RMCPHI) is presented and analysed. Moreover, an efficient Particle Swarm Optimization Combined With Simulated Annealing Algorithm (PSOSAA) is proposed for the medical monitoring and management Cloud-based applications. A Cloud Based Intelligent Health Care Service (CBIHCS) that performs real-time monitoring of users' health data for diagnosis of chronic illness such as diabetes is discussed in [13]. Specifically, advanced body sensor components are used to gather pieces of user specific health data and to store them in Cloud-based storage repositories for subsequent analysis and classification. In addition, infrastructure level mechanisms are proposed to provide dynamic resource elasticity. A simple portable kit that may be easily interfaced/integrated with the most common mechanical tools used in motion rehabilitation, with feedback to both patient for self-monitoring and trainer/therapist for clinical reporting, is proposed in [14]. The system consists of: one step-counter; three couples of photo-emitter detectors; one central unit for collecting and processing the telemetrically transmitted data; a software interface on a dedicated personal computer; and a network adapter. A wave-based bilateral teleoperation scheme for rehabilitation therapies assisted by robot manipulators is discussed in [15]. The main feature of this bilateral teleoperator is that both robot manipulators, master and slave, are controlled by impedance. Thus, a pair of motion-based adaptive impedance controllers are integrated into a wave-based configuration, in order to guarantee a stable human-robot interaction and to compensate the position drift.

An improved behavior is observed compared to implementations of the classical wave-based approach.

Cloud Computing can be the enabler for data sharing and integration on a large scale. In [16], High Performance Computing (HPC) solutions for bioinformatics, analytical paradigms of big data for computational biology, and issues that are still open in the biomedical and health sectors are discussed. In particular, it is pointed out that Cloud computing can solve the issues of storing and analyzing big data in many fields of bioinformatics. In particular, as far as telemedicine, in order to support a monitoring and automated analysis of patients, it is specified that it is very important to consider an efficient scalable Cloud infrastructure. Cloud computing well suits the medical imaging field [17]. Today, thanks to high-resolution imaging tools, the data volume can reach petabytes for a national healthcare system on yearly basis.The Cloud computing model is destined to provide a decisive contribution to meet high-performance processing needs for the reconstruction and analysis of medical images and will allow a broad sharing of imaging data, as well as advanced remote analysis. A Cloud-based mobile system aimed at improving respiratory therapy services at home is proposed in [18]. The proposed platform uses vital signs monitoring as a way for sharing data between hospitals, caregivers and patients. Specifically, by using an iterative research approach and users' direct feedbacks, it is shown how mobile technologies can improve a respiratory therapies.

The volume of healthcare data including different and variable text types, sounds, and images is increasing day by day. Therefore, the storage and processing of these data is a necessary and challenging issue. Generally, relational databases are used for storing health data which are not able to handle the massive and diverse nature of them. In this context, several NoSQL solutions have been investigate to manage big data in specific e-health application domain. In [19], issues related to data management in multimodal neuroimaging studies are discussed. In particular, an application based on MongoDB is used to recognize files to be stored through the use of a standardized nomenclature. In [7], three database approaches, i.e., NoSQL, XML-enabled and native XML, are investigated to evaluate their suitability for structured clinical data. Results show that NoSQL database is the best choice for query speed, whereas XML databases are convenient in terms of scalability, flexibility and extensibility, which are essential characteristics for clinical data. In [20], a study aimed at presenting the model based on NoSQL databases for the storage of healthcare data is presented. Among the different types of NoSQL DBMS solutions, the document-based ones were selected according to the nature of health data. The considered model was implemented in a Cloud environment for leveraging distribution properties.In particular, data were shored on the database by applying a "sharding" configuration that allows configuring the database as a distributed system. The efficiency of the model was evaluated in comparison with a relational data model, considering query time, data preparation, flexibility, and extensibility parameters. Moreover, it was demonstrated that the performance of MongoDB was better than the traditional SQL server in terms of flexibility, data preparation and extensibility. In [21], clinical dimensional model design which can be used for the development of a clinical data market is discussed. The model were designed considering temporal storage of patient's data with respect to all possible clinical parameters which can include both textual and image based data. Moreover, patients' data were used as input in data mining application in order to find correlations between individuals and a population.

Differently from the aforementioned related works, in this paper, we focus on studying the performance of different NoSQL database solutions supporting a hypothetical TRaaS scenario.

2.2. Towards Tele-Rehabilitation as a Service (TRaaS)

In this Section, we present a reference scenario including a Hospital Cloud provider supplying a TRaaS to patients. Fig. 1 shows a hospital that provides, by means of its own Cloud, a TRaaS to its patients who are placed in their homes. A TRaaS enables medical centers to manage the whole rehabilitation cycle including assessment, intervention, consultation, and education. Tele-Rehabilitation has recently emerged as an effective tool providing rehabilitation care to patients directly in their own home, also increasing clinical outcomes, positively enhancing patients' Quality of Life (QoL) and fostering their reintegration into the society. Moreover, it allows the hospital personnel to supervise the rehabilitation process of a patient when he/she is at his/her home. In this way, all monitored vital parameters are sent to the hospital Cloud through the Internet. TRaaS can potentially also enable a greater improvement of the patients' cognitive and psychological status, as well as an improvement in caregivers' QoL. TRaaS raises two major issues: technological feasibility and big data management of monitored health data over the hospital Cloud. In fact, a set of smart medical devices for health monitoring are strongly required in order to replicate at patients' home, a rehabilitation treatment environment similar to that available in a medical center. Moreover, since the hospital Cloud can receive a huge amount of tele-rehabilitation information coming from several patients spread over a wide geographical area, a big data management system is strongly required.

2.3. Home rehabilitation technology feasibility

In this Section, we motivate how a TRaaS can be achieved by adopting low-cost smart home devices enabling tele-monitoring. During the treatment at home, patients can use wearable sensing devices in order to monitor their status (e.g., real-time speed, heart rate, respiratory rate, training load, single-lead ECG and so on) along with other sensing devices monitoring the surrounding environment. All generated big tele-monitoring data have to be sent to the Hospital Cloud. Thus, healthcare operators can remotely provide them real-time feedback during exercises and on-fly adjust the therapy is required. So as to make a TRaaS costly effective, low costs monitoring equipments with high reliability in data sensing and processing should be adopted. Until now the market in such a domain has been dominated by proprietary, vendor lock-in and expensive solutions that make difficult their integration with third-party software systems. However, the current trend

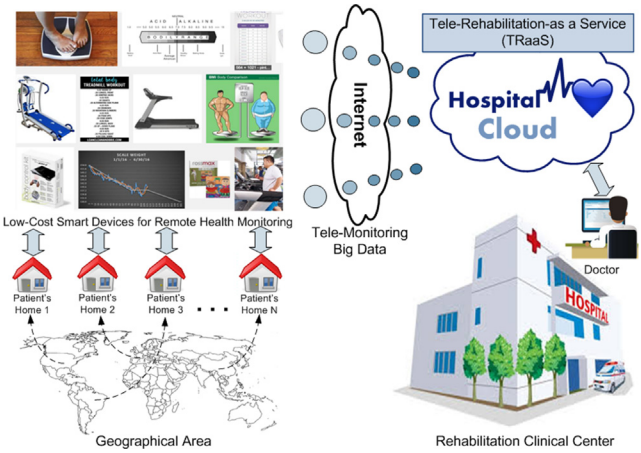


Fig. 1. Example of hospital Cloud providing a Tele-Rehabilitation as a Service (TRaaS).

of creating very low-cost IoT components is opening new possibilities in the domain of remote healthcare tele-monitoring. Indeed, in the last period we are witnessing to a real battle among chip makers that are cutting down costs of chips [22], like Realtek (e.g., RTL8710 costs 1.5$) and Espressif (e.g., ESP8266 costs 2.0$ [23]). Moreover Espressif recently released another challenging device named ESP32 [24], which is much more powerful with respect to the well-known ESP8266 [23]. Even Gartner has recently defined Espressif as *Cool Vendor in IoT 2016* [25] for two reasons: low cost and small size. ESP32 devices are all characterized not only by a remarkable low cost, but even by their standalone capabilities, such as computation resources (micro-controller based products are very powerful), RAM (at least 1 MByte of memory), communications&protocols (e.g., WiFI, Bluetooth, Ethernet, IP, UDP, TCP, CoAP) and many GPI/Os (e.g., PWM, PCM, ADC, DAC etc.) for interfacing them with any external physical/electrical transducer. In acquiring and converting analogue physical/electrical data (ADC), the GPIOs of such devices present a considerable sampling rate in terms of frequency (up to 100khz, that is good also for ultrasound scans) along with a good depth of bits per sample (at least 10 bits). With a few dollars, it is possible to benefit of low-cost smart board devices offering wireless, computation and storage capabilities, I/O facilities for electrical transducers, security capabilities for guaranteeing security and privacy.

In the context of human health monitoring, smart devices allow us to transform any simple tool for rehabilitation in an advanced IoT device that can be remotely controlled and monitored. Moreover, in the market, new cheap complex systems able to monitor even more than human health parameters are appearing that could be easily integrated with ESP32 for example. Differently from existing Commercial Off-The-Shelf (COTS) human health monitoring devices, ESP32 may allow the development of a plethora of very cheap customized tele-monitoring devices for a TRaaS scenario. Examples are scales able to measure apart from human weight other parameters such as fat mass, lean mass, and body mass index, by exploiting FDA-cleared bioelectrical impedance analysis. Often these scales are connected through Bluetooth to remote computers to perform periodic analysis of measured data.

In fact, patients may have a plethora of wireless systems able to monitor and assist them at home, such as: low cost treadmill, exercise bike, adult walker aid, bluetooth scale, blood monitor (for pressure, glucose, oxygen, etc.), step counter (pedometer), bluetooth heart rate monitor, game-based interaction with SmartPhones, game-based interaction with smart TVs and remote controllers and so on. In some cases, cheap smart sensing devices measure physical parameters interfacing them with physical/electrical transducer (e.g., pedalling electronic measurement, running electronic measurement, indoor positions in home, etc.). In other cases, they may represent the sinks conveying all the data generated by wireless systems (e.g., smart health monitoring devices with remote controllers) [26]. Each device can be customized for one or more specific purposes. Game-based interactions allow care givers to have feedbacks on the psychomotor conditions of a patient during the rehabilitation at home. Indeed, simple games allow to know how patients interact with their environments, hence how much the rehabilitation is effective.

3. Method and material

3.1. Tele-monitoring data model

As previously discussed, in a TRaaS we assume that patients are equipped with low-cost tele-monitoring (TM) devices able to i) collect data on their health status and on the surrounding environment; ii) send such collected data to the hospital Cloud for processing. In this context, a data model that allows the Hospital Cloud to efficiently manage together both conventional personal and big tele-monitoring data coming from patients' homes if required. For this reason a traditional SQL-like technology considered alone is not suitable, but it must be integrated with a NoSQL technology able to efficiently manage tele-monitoring data in a flexible and scalable fashion. Fig. 2, shows a hybrid Entity-Relationship (ER) diagram that describes our tele-monitoring data model considering both "small" and "big" data entities [10,27]. It allows organizing data into a database, but it is independent from any specific DataBase Management System (DBMS) technology. The main components of the ER model are *entities* and *relationships* among entities. An *entity* may be defined as an independent subject or object. It can be either physical or logical, and it is usually rendered as a rectangle. Entities are represented by means of their properties, called *attributes*, and all attributes have values. For example, in a e-health domain, a patient entity may have name, surname and age as attributes. A *relationship* captures how entities are related each others. For example, a patient has a relationship with the doctor. Relationships are portrayed as rhombus and lines connect entities and relationship blocks in an ER diagram. *Cardinality* defines the number of elements of the same entity that can be associated with the number of elements of another entity via the same relationship. The ER diagram depicted in Fig. 2 shows how the *Health State* of a patient is strictly related to the personal data of the *Patient* itself, to the *Doctor* that nurses him/her, to the *Observation* of physical, biological, and neurological *Parameters* measured with clinical *Devices*, and, hence, to the specific *Therapy* provided to the Patient. The model describes a complex "patient centric" health care system, in which the patient status is observed during the time by several instruments and medical devices located at the care center or/and at home. Observations and measurements are evaluated by one or several doctors considering also personal factors of the patient and his/her health status, in order to provide the best suitable therapy.

In such a TRaaS scenario two different types of data need to be managed:

1. *Personal data*. Personal information of patients and doctors, that includes identity records and personal details. This type of data are memorized in the storage system one time and generally does not change frequently during the time;
2. *Big tele-monitoring data*. Parameters generated from monitoring activities on the patient. They are independent measurements or composed phenomena collected by sensing devices or medical instruments. Observations can be expressed by tuples $(key, value)$ and stored in text files that are sent to the hospital Cloud. Such data can also include multimedia contents (e.g., audio, image, video and animation) recorded by information content processing devices. The volume of tele-monitoring big data can rapidly grows up during the time and, thus, a big data Cloud-based storage system is required.

In the following, we are going to describe how to manage such a complex data system, considering the above different types of data.

Due to the heterogeneous nature of involved data, different technologies must be exploited to implement an efficient Cloud-based storage system. Decoupling personal data and big tele-monitoring data allows us to consider pseudo-anonymized healthcare information: who accesses personal data is not able to know the healthcare status of the patient, and who accesses tele-monitoring big data is not able to know the identity of the related patient. Only users that have credentials to use the storage system can merge and correlate different types of information and build the e-health status and history of a patient. Data anonymization

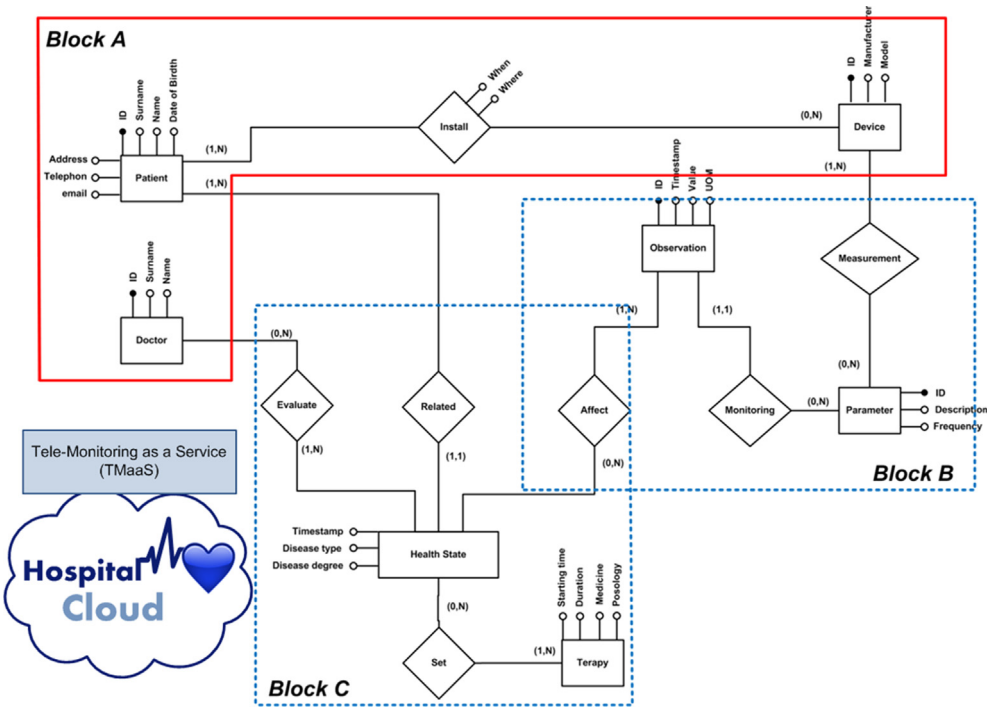


Fig. 2. TMaaS Cloud ER Data model for tele-rehabilitation data management.

also allows us to exploit benefits of a hybrid/public hospital Cloud storage system, thus to adopt the most useful, reliable and scalable storage service. Indeed, keeping personal data in a local and private storage system, tele-monitoring big data can be pushed on external private or commercial Cloud storage providers without any risk for the patients' privacy. Personal patients' data can be stored in a traditional SQL-like database (e.g., mySQL, Postgrees, MariaSQL,...) thus to optimize data storage ad retrieval of structured data and to be compliant with existing legacy systems. Indeed, personal data identified in the Patient and Doctor entities of the above discussed ER model includes Name, Surname and other possible personal information. Instead, tele-monitoring big data deal with very large unstructured data sets that need a rapid analytics with answers provided in seconds. However, strategies to manage big data strongly depend on the specific data model. In the ER model shown in Fig. 2, observations collected during the time for each patient via several heterogeneous devices and instruments can generate big data because analysis activities can produce several tuples in a short time interval. Thus, in long periods (days, months, years) a huge amount of data need to be stored and processed. Observations can be made available through documents in text format that encapsulate tele-monitoring data in a scheme-less fashion.

Starting from the ER model, we identified the portions (blocks) of the data model necessary to implement personal data and big tele-monitoring data in the hospital Cloud provider. Red lines identify personal data entities that can be stored in a traditional SQL-like database (*Block A*) because they are related to static information of patients and doctors. Blu dotted lines identify instead blocks including tele-monitoring big data entities that need to be stored in a NoSQL database. In particular, *Block B* collects all unstructured tele-monitoring big data coming from heterogeneous health monitoring devices used by patients in their homes, and provides information about vital parameters, besides patients feedbacks. Due to the heterogeneity of home monitoring devices, generated big data are unstructured. In fact, building a specific structured static data model as well as for relational databases is quite difficult because data can frequently change: new parameters with can suddenly appear, whereas others can change their structure or also disappear. *Block C* collects evaluations of doctors on the rehabilitation process and on the current health status of patients. In this paper, we focus our the attention on *Block B* that is the most critical for the implementation of near future hospital Cloud TRaaS.

3.2. Management of tele-monitoring big data

Considering a traditional SQL-like database the same data model can be implemented by means of different DBMS solutions. In this case, the performance of the system typically depends on the adopted DBMS solution. Instead, considering NoSQL databases things are different. In fact, the performance of NoSQL DBMS solutions, besides the degree of the adopted data distribution, replica, and parallelization, depends on the nature of the data model itself. Thus, for a particular data model a NoSQL solution can be better than another. Considering Block B of Fig. 2 we remark that:

- the involved entities are characterized by a huge amount of tele-monitoring raw data that rapidly grow up due to the scalable nature of the TRaaS;
- due to the heterogeneous nature of home rehabilitation devices, big tele-monitoring data can frequently change. This means that new fields can be suddenly created, modified or deleted;
- moreover, since rehabilitation home devices are heterogeneous it is not possible to conceive in advance a static data model.

Since NoSQL solutions typically address the aforementioned requirements, in our opinion, they represent the best solution to store and manage tele-monitoring big data of a TRaaS supplied by a hospital Cloud provider. NoSQL DBMS solutions can be classified according to the data model nature. In order to identify the best NoSQL DBMS where to store tele-monitoring big data, in this paper we consider four of the major current commercial solutions implementing the database entities of Block B, that are, MongoDB (adopting a document-based data model), Cassandra (adopting a

column-based data model), Hbase (adopting a column-based data model), and Neo4J (adopting a graph-based data model). In the following, we provide a brief overview of such NoSQL DBMS solutions.

3.2.1. MongoDB

MongoDB is an open source distributed parallel document NoSQL database that natively supports a huge amount of unstructured data and for this feature it is also defined scheme-less or scheme-free. In MongoDB, a database is organize in *collections*, each one storing data as *documents* according to a JSON-like format called BSON. MongoDB can create databases, collections and documents "on fly". This makes MongoDB very flexible and able to grow up, if necessary, during the time. The MongoDb data model is generally defined "denormalized" and takes the advantage of rich documents. Embedded data models allows creating/updating and querying nasted in a single atomic write operation. MongoDb manages data distribution and replica by means of sharding and replica-set mechanisms.

Considering the ER scheme depicted in Fig. 2, *Block B* entities are implemented using different collections according to a normalized data model.

3.2.2. Cassandra

Apache Cassandra is an open source distributed parallel shemaless column NoSQL DBMS designed to handle large amounts of data across distributed servers, providing high availability with no single point of failure. The main features of Cassandra includes decentralization, replication, scalability, fault-tolerance, tunable consistency, and map-reduce. Moreover, Cassandra introduces a native Cassandra Query Language (CQL), i.e., providing native SQL-like syntaxes. It is essentially a hybrid solution between a key-value and a column-oriented DBMS. Its data model is a partitioned as a row store with tunable consistency. A column family (called "table" since CQL 3) contains rows and columns. Each row is uniquely identified by a row key. Each row has multiple columns including a name, a value, and a timestamp. Unlike a table in an RDBMS, different rows in the same column family do not have to share the same set of columns, and a column may be added to one or multiple rows at any time. Therefore, a table in Cassandra can be considered as distributed multi-dimensional map indexed by a key.

Considering the ER scheme depicted in Fig. 2, *Block B* entities are implemented using different column families according to a normalized data model that specifies relationships between columns.

3.2.3. Hbase

HBase is an alternative open source distributed NoSQL column database written in Java. It was developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of Hadoop Distributed File System (HDFS), providing BigTable-like capabilities for Hadoop. HBase allows data compression, in-memory operations, and bloom filters on a per-column basis. Tables in HBase can serve as the input and output for MapReduce jobs to be run in Hadoop, and may be accessed through either Java or REST APIs. HBase enables faster read and write operations on large datasets with high throughput and low input/output latency. The Hbase architecture includes region and master servers. Region servers are activated to store and retrieve data. When a table grows up beyond a target size it is slit in two region serves in order to balance the workload by means of an automatic sharding process. Each object, stored in a column family, has a reading cache called BlockCache and a writing cache called MemCache. Data are organized in hfiles inside a MemStore. In order to speed up the reading process, at regular intervals hfiles are combined in a bigger hfile by means of a compacting mechanism. Master servers monitor region servers and manage load balancing in a Hbase cluster. For each master server there is a backup server. Hbase services are synchronized by means of an additional piece of middleware called Zookepeer.

Considering the ER scheme depicted in Fig. 2, *Block B* entities are implemented using different column families according to a normalized data model that specifies relationships between columns.

3.2.4. Neo4j

Neo4j is a highly scalable open source NoSQL graph database designed to leverage not only data but also its relationships. The Neo4j's native graph storage and processing engine deliver constant real-time performance, helping enterprises build intelligent applications in order to meet today's evolving data analytics challenges. Relationships provide directed, named semantically relevant connections between two node-entities. A relationship always has a direction, a type, a start node, and an end node. Like nodes, relationships can have any properties. In most cases, relationships have quantitative properties, such as weights, costs, distances, ratings, time intervals, or strengths. Since relationships are stored efficiently, two nodes can share any number or type of relationships without sacrificing performance. Neo4j efficiently implements the Property Graph Model (PGM) directly at the storage level. Data reliability is a key design consideration for Neo4j. In fact, as opposed to graph processing or in-memory libraries, Neo4j provides full database characteristics including ACID transaction compliance, cluster support, and run-time fail-over, allowing to leverage the advantages of graph data in production scenarios. It also supports replication with master re-election and failover mechanisms keeping data safe and reliable.

Considering the ER scheme depicted in Fig. 2, *Block B* patients and tele-monitoring observations are implemented by means of nodes connected by observe edges. Patients are related to 0..N tele-monitoring observations via observe relationship.

4. Experiments

The objective of this Section is to understand which NoSQL solution well suits the requirements of a TRaaS in terms of tele-monitoring big data storage. For this reason, as previously mentioned, we tested and compared four among the major NoSQL DBMS implementations available on the market, that are Mongodb, Cassandra, Hbase, and Neo4j. Moreover, in order to simulate a tele-monitoring big data model, we considered the Lokomat data format.

Listing 1: Example of native Lokomat data format.

```
[Bob Brown]
Patientname = Bob_Brown
ID = Null
Weight = 65.0
Height = 1.70
l_shank = 0.52
l_thigh = 0.40
Lokomat shank = 0.00
lokomat thigh = 0.00
Recorded = 11:36:39 19.10.2015
Version = Lokocontrol 6_x
LegType = V5
[L-WALK]
Trainingduration = 00:08:36
```

```
Distance= 268.848 m
[Per step data]
Body Weight Support [kg];
patient coefficient;
Range of motion HL [%];
Range of motion KL [%];
Range of motion HR [%];
Range of motion KR [%];
Offset ROM HL [o];
Offset ROM KL [o];
Offset ROM HR [o];
Offset ROM KR [o];
Guidance Force L [%];
Guidance Force R [%];
Speed [km/h];
Energy Hip Left;
Energy Knee Left;
Energy Hip Right;
Energy Knee Right;
#Step; bio_HL_st; bio_HL_sw;
bio_KL_st; bio_KL_sw; bio_HR_st; bio_HR_sw;
bio_KR_st; bio_KR_sw;
Position deviation stance HL;
Position deviation stance HR;
Light curtain left; Light curtain right;
Unloading Left;Unloading Right

33.000; 0.520; 1.000; 1.000;
1.000; 1.000; 0.000; 0.000;
0.000; 0.000; 100.000; 100.000;
1.300; Null; Null; Null;
Null; 1.000; −45.169; −110.564;
302.388; −54.345; 33.579; −57.469;
64.833; −31.883; Null; Null;
Null; Null; Null; Null;

33.000; 0.520; 1.000; 1.000;
1.000; 1.000; 0.000; 0.000;
0.000; 0.000; 100.000; 100.000;
1.600; Null; Null; Null;
Null; 2.000; −55.742; 179.035;
304.360; −175.878; 86.271; −78.211;
134.951; 9.940; Null; Null;
Null; Null; Null; Null;

33.000; 0.520; 1.000; 1.000;
1.000; 1.000; 0.000; 0.000;
0.000; 0.000; 100.000; 100.000;
1.600; Null; Null; Null;
Null; 3.000; −57.466; 153.848;
332.553; −127.601; 196.073; −120.732;
261.467; 41.980; Null; Null;
Null; Null; Null; Null;

...
```

The Lokomat system is widely used in many neuro-rehabilitation centers. Lokomat offers the most physiological gait pattern with constant feedback and therapy assessment. It improves patient outcomes by increasing the therapy volume and intensity, providing task-specific training and enhanced patient engagement. The Lokomat device, we are considering is placed at IRCSS "Bonino Pulejo" healthcare center. Fig. 3 shows the structure of the considered device. It consists of a support system and an esoskeleton for patient with movement impairments; a tipis roulant; and a video/audio system reproducing virtual reality multimedia contents. We assume, that how motivated in Section 2.3, in the TRaaS scenario, a similar environment can e replicated at patients' homes interacting with a Hospital Cloud. Listing 1 shows an example of Lokomat data format. Unstructured data are organized in two parts. In the first one it reports "generic data" regarding the patient (e.g., name, weight, height, etc) and generic data regarding the device and the particular observation (e.g., Lokomat version, shank, thin, walk distance, etc). Instead, the second part, "observed data" (e.g., patient coefficient, range of motion, speed, etc) are reported. Further recent details about Lokomat observation values are available in [28]. In our experiments, we refer with the term "observation" a document compliant with the Lokomat data format and including, apart from general data, 220 records. An example of record is reported from line 27 to line 34 of Listing 1. Since a standard format for representing tele-monitoring data over the Cloud does not exist at the moment of writing this paper, in our experiments we considered the Lokomat data model. We simulated datasets respectively containing 10, 100, 1000, 10000 observations and we stored them in Mongodb, Cassandra, Hbase, and Neo4j. Experiments were conducted in the IRCSS "Bonino Pulejo" data center, considering databases configured into a single server with the following hardware configuration: blade with CPU Dual-Core AMD Opteron(tm) Processor 2218 HE, RAM 6 GB, OS ubuntu server 12.04.2 LTS 64 Bit. For a performance comparison with a relational SQL-like DBMS, i.e., Oracle [27], we populated the different NoSQL database implementations with the same datasets, and we executed query tests so as to assess the time spent to extract data related to a specific patient, considering the time of query submission and the time for query response. In particular, we performed three queries with an increasing degree of complexity:

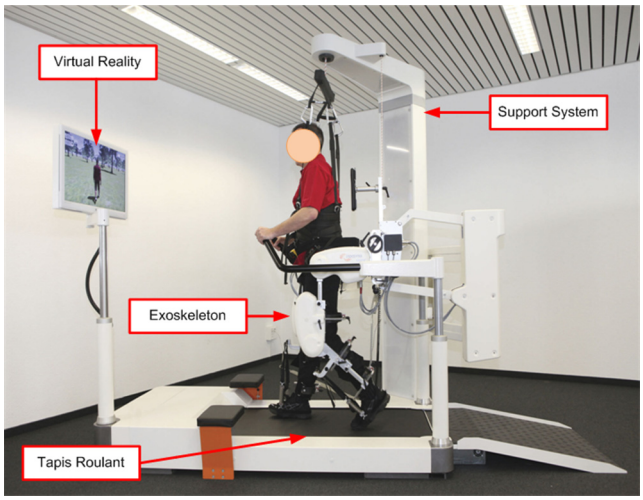


Fig. 3. Lokomat v6 rehabilitation device.

- *Query 1* retrieves the tele-monitoring data of all patients.
- *Query 2* retrieves the tele-monitoring data of a specific patient.
- *Query 3* retrieves the tele-monitoring data characterized by a "lwalk_training_duration" less than *a* specific value and with "width" not equal to a specific value *b*, for patients with at least 5 measurements.

Furthermore, we ran the three queries on MongoDB (Listings 2, 3, and 4), Cassandra (Listings 5, 6, and 7), Hbase (Listings 8, 9, and 10), and Neo4J (Listings 11, 12, and 13) respectively using pymongo, cassandra-cluster, happybase, and neo4j-driver drivers/

libraries. For each dataset, we respectively executed the same query 30 times on Cassandra and 31 times on Mongodb, Hbase, and Neo4J, because, for performance optimization, when the same query is executed on the same dataset, these last ones process all actual data only the first time, putting results in their cache memories. As a consequence from the second query on, the information extraction is performed on cached data. Thus, we obtained mean execution times and corresponding confidence intervals at 95%.

Listing 2: syntax of Query 1 using the Python "pymongo" driver/library.

```
def Query1(db):
  collection = db.Patients
  patients = collection.find({})
  return patients
```

Listing 3: syntax of Query 2 using the Python "pymongo" driver/library.

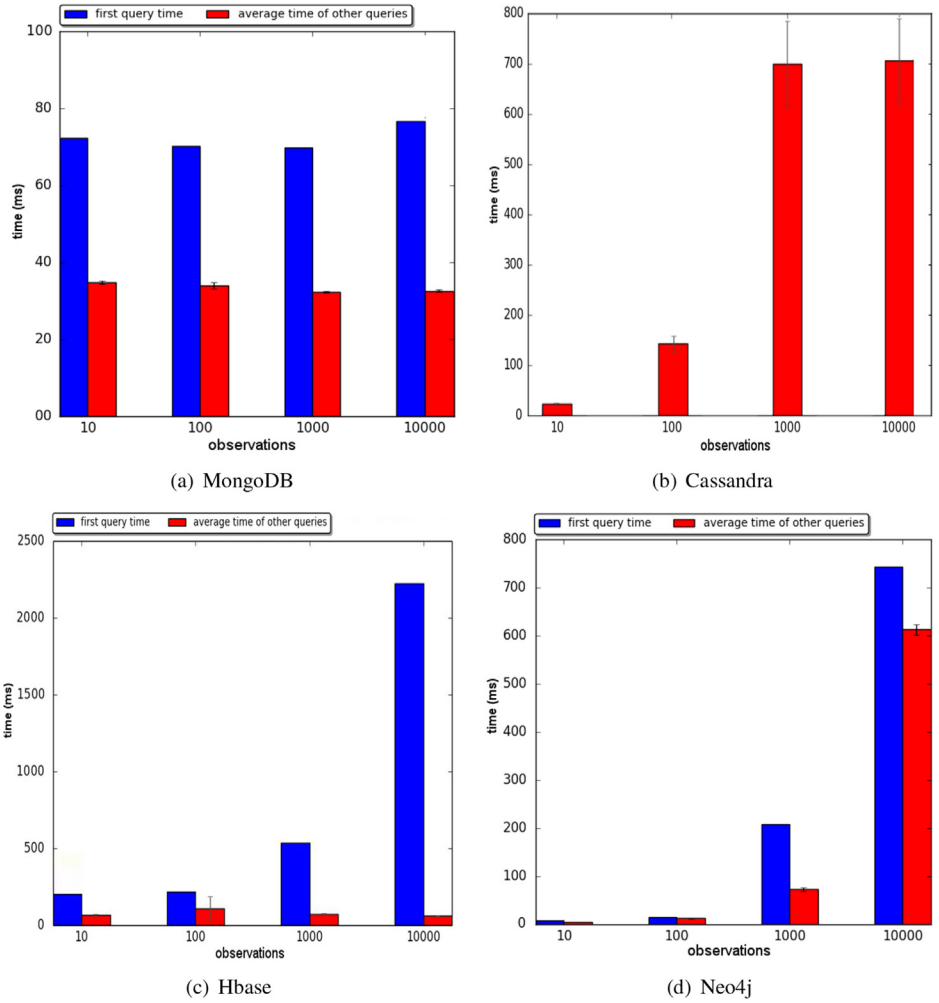
```
def Query2(db):
  collection = db.Patients
  patients = collection.find('name': 'KRVYRSKX7')
  return patients
```

Listing 4: syntax of Query 3 using the Python "pymongo" driver/library.

```
def Query3(db):
  collection = db.Patients
  patients = collection.find({
    'lwalk_training_duration': {'$lt': 2400},
    'width': {'$ne': 0.80},
    'step_datas': {'$size': 5}
    })
  return patients
```

Listing 5: syntax of Query 3 using the Python "cassandra-cluster" driver/library.

```
def query1():
  return query("select * from patient;")
```



(a) MongoDB (b) Cassandra

(c) Hbase (d) Neo4j

Fig. 4. Query 1. Retrieving tele-monitoring data of all patients.

Listing 6: syntax of Query 3 using the Python "cassandra-cluster" driver/library.

```
def query2():
  return query("select *
    from patient
    where name = 'KRVYRSKX7'
    allow filtering;")
```

Listing 7: syntax of Query 3 using the Python "cassandra-cluster" driver/library.

```
def query3():
  x = []
  results = query("select *
         from patient
         where lwalk_training_duration < 2400
         allow filtering;")
  for a in results:
    if (a[ll]!= None):
      if ((a[l3]!= 0.80) and (len(a[ll]) > 4)):
        x.append(a)
  return results
```

Listing 8: syntax of Query 3 using the Python "happybase" driver/library.

```
def get_info_patients(connection, dataset):
  time = datetime.datetime.now()
  patients_table = connection.table('patient')
  return patients_table.scan(columns=
    ['analysis:width',
    'analysis:lokomat_shank',
    'analysis:lokomat_thigh',
    'analysis:lwalk_distance',
    'analysis:height',
    'analysis:id',
    'analysis:legtype',
    'analysis:lwalk_training_duration',
    'analysis:name',
    'analysis:l_shank',
    'analysis:l_thigh',
    'analysis:lokomat_recorded',
    'analysis:step_datas'])
```

Listing 9: syntax of Query 2 using the Python "happybase" driver/library.

```
def get_patient_with_name(connection,dataset):
  time = datetime.datetime.now()
  patients_table = connection.table('patient')
  return patients_table.scan(filter=
  "SingleColumnValueFilter
  ('analysis','name',=,'regexstring:
  'KRVYRSKX7')")
```

Listing 10: syntax of Query 3 using the Python "happybase" driver/library.

```
def get_misuration__patient(connection,dataset):
  time = datetime.datetime.now()
  patients_table = connection.table('patient')
  return patients_table.scan(filter=
  "SingleColumnValueFilter('analysis',
  'lwalk_training_duration',<,2400) AND
  ('analysis','width',!=,0.80) AND
  ('analysis','step_datas',>,5)")
```

Listing 11: syntax of Query 2 using the Python "neo4j-rest-client" driver/library.

```
Benchmark queries
bench_query('query0','''
  MATCH (n:patient)
  RETURN n''')
```

Listing 12: syntax of Query 2 using the Python "neo4j-rest-client" driver/library.

```
bench_query('queryl','''
  MATCH (n:patient)
  WHERE n.n = ''SIVV33WO"
  RETURN n''')
```

Listing 13: syntax of Query 2 using the Python "neo4j-rest-client" driver/library.

```
bench_query('query2','''
  MATCH (p:patient)-[r:measure]->(m:measurement)
  WITH p, m, count(m) as relcount
  WHERE p.lwalk_td < 5000 AND p.w
  <> 5000 AND relcount > 4
  RETURN p"')
```

Fig. 4 shows the response times respectively obtained with MongoDB, Cassandra, Hbase, and Neo4j executing query 1. On the x-axis we reported the different dataset sizes, whereas on the y-axis we reported the time expressed in milliseconds (ms). How can be observed in the histogram of Fig. 4a, the fastest NoSQL database implementation is MongoDB. In fact, considering the first execution of query 1, it takes roughly 75*ms* on 10, 100, 1000 observations and roughly 80*ms* on 10000 observations. Accordingly, a constant value of roughly 35*ms* can be appreciated considering the average query execution time on cached data. Cassandra, whose response times are represented in the histogram depicted in Fig. 4b, shows a good performance for 10 observations, that is roughly 20*ms*, but increasing the number of observations we can observe a performance degradation. In fact, we can observe, for both 1000 and 10000 observations, a similar query response time of roughly 700*ms*. Fig. 4c shows the response times of the Hbase implementation. Here response times linearly increases from

200*ms* up to roughly 2250*ms*. However, the response time on cached data appears quite constant being roughly 80*ms*. Whereas, considering Neo4j, whose histogram is shown in Fig. 4d, we observe slightly better response times considering cached data up to 1000 observations, but considering 10000 observations we notice a response time of roughly 600*ms*. In general, considering 10000 observations, the best solution for is represented by Mongodb.

Fig. 5 shows response times respectively obtained in MongoDB, Cassandra, Hbase, and Neo4j executing query 2. On the x-axis we reported the different dataset sizes, whereas on the y-axis we reported the time expressed in ms. How can be observed in the histogram of Fig. 5a, the fastest NoSQL implementation is still MongoDB. In fact, both the first query and average execution times are roughly 35 μs. Also Hbase, whose histogram is depicted in Fig. 5c, shows good response times. In fact, increasing the number of observations the average response time on cached data appears quite constant being roughly 120*ms* for 10000 observations. Instead, considering the first execution time, we observe a constant value of roughly 210*ms* up to 1000 observations and a value of roughly 850μs considering 10000 observations. Whereas, considering Cassandra and Neo4j, whose histograms are respectively depicted in Fig. 5b and 5c, we can observe a meaningful performance degradation. As far as Cassandra, we can observe, for 1000 and 10000 observations, a similar query response time of roughly 1200*ms*. In the end, regarding Neo4j, we can observe mean response time under 100*ms* up to 1000 observations considering cached data and a meaningful performance degradation considering 10000 observations with roughly 410*ms*. Thus, in this case Cassandra is the worst implementation in terms of performance. Fig. 6 shows the response time respectively obtained on MongoDB, Cassandra, Hbase, and Neo4j executing query 3. On the x-axis we reported the different dataset sizes, whereas on the y-axis we reported the time expressed in ms. How can be observed in the histogram of Fig. 6a, even though in this case the fastest NoSQL implementation is MongoDB. Both the first query and the average execution times are roughly 35*ms*. Whereas, considering Cassandra and Neo4j, whose histograms are respectively depicted in Fig. 6b, we observe mean response times under 50 ms up to 100 observations and a mean value of roughly 290 ms for 1000 and 10,000 observations. Considering Hbase, according to the histogram depicted in Fig. 6c, c, up to 1000 observations we observe constant mean response times for both the fist execution (roughly 250 ms) and the execution on cached data (80 ms). Instead, for 10,000 observation the first execution takes 2200 ms whereas the execution on cached data remains constant to roughly 80 ms. In the end, considering Neo4j, whose means response times are shown in Fig. 6c, we observe good mean response times up to 1000 observations an a meaningful performance degradation for 10,000 observations with an executions time of roughly a 4500 ms for both the first execution query and for the query on cached data.

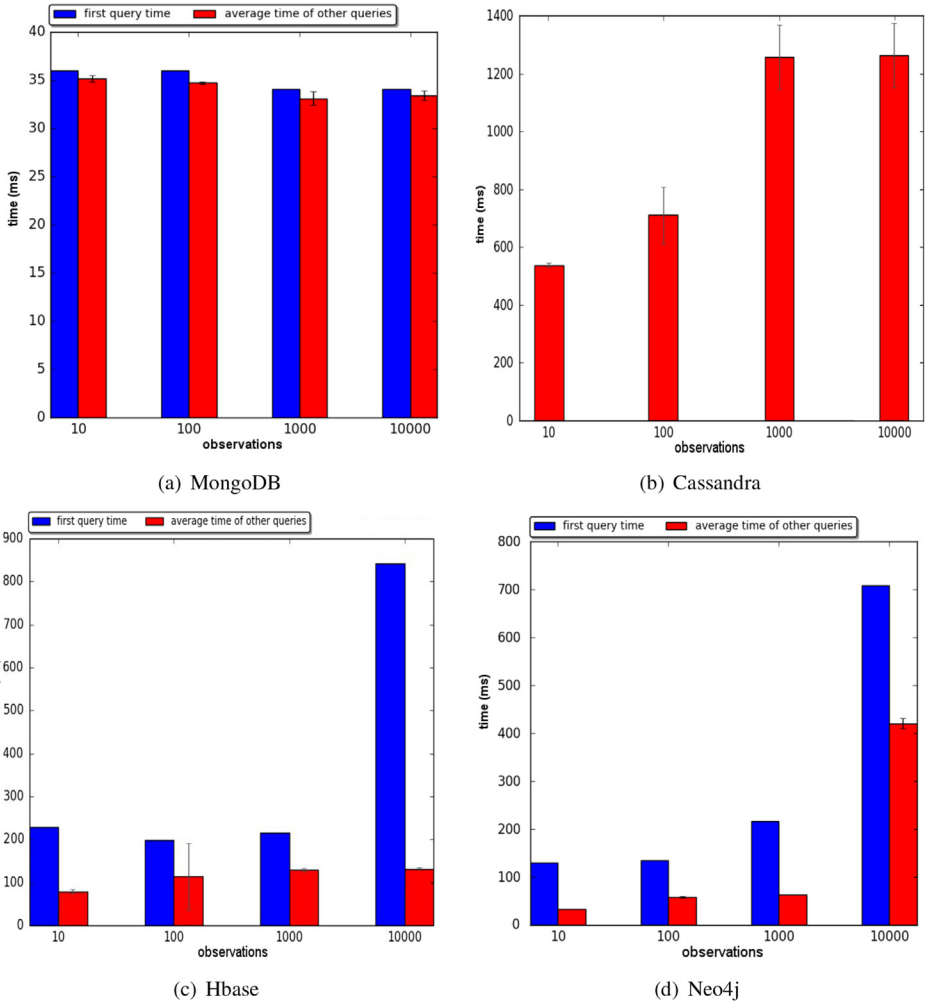


Fig. 5. Query 2. Retrieving tele-monitoring data of a specific patient.

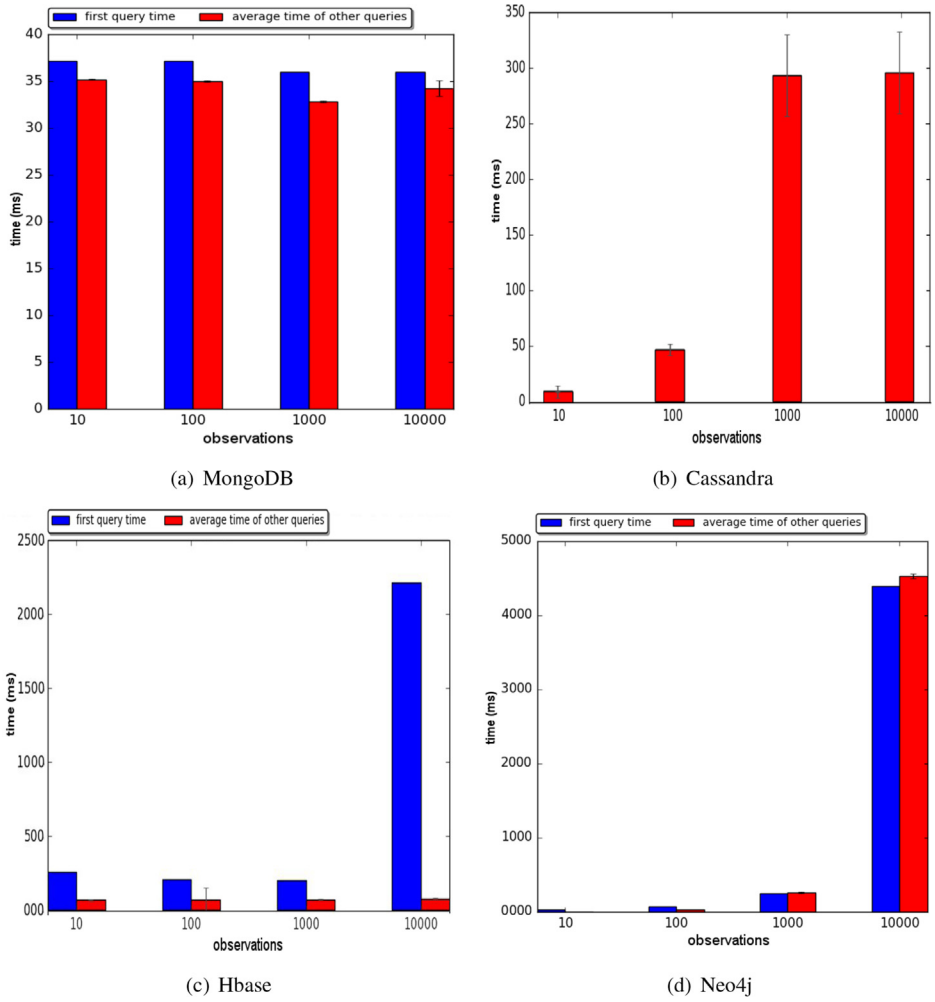


Fig. 6. Query 3. Retrieving tele-monitoring data characterized by a "lwalk_training_duration" less than *a* specific value and with "width" not equal to a specific value *b*, for patients with at least 5 measurements.

5. Discussion and Conclusion

An efficient big tele-monitoring data management is on the basis of future Internet TRaaS offered by Hospital Cloud providers. From our experiments conducted in the IRCCS "Bonino Pulejo" datacenter, the NoSQL document-based approach implemented by means of MongoDB resulted the best solution, in terms of performance, for the management of big tele-monitoring data. It was also interesting to study the response time of the different alternative NoSQL database implementations considering both first time query execution and the query execution on cached data.

For all considered NoSQL DBMS implementations, except for Cassandra, we considered both the first time execution on the whole dataset and the average execution time on cached data. The first one, is interesting for real-time applications accessing data that are frequently updated, whereas the query execution on cached data is interesting for applications that process batch data in background. In our opinion, the first query execution time is more critical for TRaaS because it can be of vital importance for the patient. From this point of view MongoDB also resulted the best NoSQL implementation. In fact, it provides the best response times considering all queries. What it is suprising is instead the behavior of Neo4j: although being conceived to manage graph data, considered the adopted datasets it provides good performances: in fact, in query 1 it has a similar response time compared to Cassandra and it is faster than Hbase. Furthermore, in query 2, it is even faster than Hbase. Instead, considering query 3, it provides the worst response time due to the complexity of the query itself. This paves the way toward a possible adoption of Neo4j expecially if graph data are included in the produced robotic rehabilitation dataset. Hbase, instead results the worst solution for both qury 1 and 3. This negative result can be motivated by the fact that it depends from both the Hadoop framework and Zookeeper communication system to properly work, causing a meaningful overead when the DBMS is installed on a single server. In general, as far as applications processing batch tele-monitoring big data in background, apart MongoDB, also Hbase, Cassandra (both adopting a NoSQL column-based approach) represent good alternatives. We stress that these results were obtained considering for each DBMS a standalone configuration on a single server and that such results can be improved considering alternative distributed configurations for all considered NoSQL solutions even though Hbase requires a greater number of nodes than MongoDB and Cassandara in order to offer comparable response times.

Our experiments focused on tele-monitoring big data collected by the Lokomat device in order to consider a robotic rehabilitation scenario. In future work, we plan to explore more complex TRaaS scenarios including integrated data coming from various "home" tele-monitoring devices. Furthermore, in order to improve the TRaaS, we will focus on tele-monitoring big data analytics

techniques based on Machine Learning able to allow clinical operators to adjust, if required, in real-time the rehabilitation therapy of patients.

With this paper, we hope we succeeded in stimulating the scientific community interested in Cloud based tele-health services.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the Italian Healthcare Ministry founded project Young Researcher (under 40 years) entitled "Do Severe acquired brain injury patients benefit from Telerehabilitation? A Cost-effectiveness analysis study" – GR-2016-02361306. Jiafu Wan's work was supported by Guangdong Province Key Areas R&D Program (Nos. 2019B010150002, 2019B090919002).

References

[1] Europe 2020 – for a healthier EU.https://ec.europa.eu/health/europe_2020_en (accessed 01.06.2019)..
[2] A. Lay-Ekuakille, P. Vergallo, A. Trabacca, M.D. Rinaldis, F. Angelillo, F. Conversano, S. Casciaro, Low-frequency detection in ecg signals and joint eeg-ergospirometric measurements for precautionary diagnosis, Measurement 46 (1) (2013) 97–107, https://doi.org/10.1016/j.measurement.2012.05.024, URL:http://www.sciencedirect.com/science/article/pii/S0263224112002333.
[3] C. Chiffi, A. Lay-Ekuakille, Physiotherapeutic movements by advanced robotic beds: perspectives in triggering nanodrugs, in: 2016 Nanotechnology for Instrumentation and Measurement (NANOfIM), 2016, pp. 5–16, https://doi.org/10.1109/NANOFIM.2016.8521420.
[4] P. Mostarac, R. Malarić, M. Jurčević, H. Hegeduš, A. Lay-Ekuakille, P. Vergallo, System for monitoring and fall detection of patients using mobile 3-axis accelerometers sensors, in: 2011 IEEE International Symposium on Medical Measurements and Applications, 2011, pp. 456–459, https://doi.org/10.1109/MeMeA.2011.5966724.
[5] R. Velázquez, E. Pissaloux, P. Rodrigo, M. Carrasco, N.I. Giannoccaro, A. Lay-Ekuakille, An outdoor navigation system for blind pedestrians using gps and tactile-foot feedback, Appl. Sci. 8 (4) (2018), https://doi.org/10.3390/app8040578, URL:http://www.mdpi.com/2076-3417/8/4/578.
[6] A.M. Kuo, Opportunities and challenges of cloud computing to improve health care services, J. Med. Internet Res. 13 (3) (2011).
[7] K.K.-Y. Lee, W.-C. Tang, K.-S. Choi, Alternatives to relational database: comparison of nosql and xml approaches for clinical data storage, Comput. Methods Programs Biomed. 110 (1) (2013) 99–109.
[8] M. Fazio, M. Paone, A. Puliafito, M. Villari, Heterogeneous sensors become homogeneous things in smart cities, in: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on, 2012, pp. 775–780.
[9] Y.-T. Liao, J. Zhou, C.-H. Lu, S.-C. Chen, C.-H. Hsu, W. Chen, M.-F. Jiang, Y.-C. Chung, Data adapter for querying and transformation between sql and nosql database, Future Gener. Comput. Syst..
[10] M. Fazio, A. Celesti, A. Puliafito, M. Villari, Big data storage in the cloud for smart environment monitoring, Proc. Comput. Sci. 52 (2015) 500–506.
[11] S. AlDossary, M. Martin-Khan, N. Bradford, A. Smith, A systematic review of the methodologies used to evaluate telemedicine service initiatives in hospital facilities, Int. J. Med. Inf. 97 (2017) 171–194.
[12] S. Luo, B. Ren, The monitoring and managing application of cloud computing based on internet of things, Comput. Methods Programs Biomed. 130 (2016) 154–161, https://doi.org/10.1016/j.cmpb.2016.03.024, URL:http://www.sciencedirect.com/science/article/pii/S0169260715303291.
[13] P.D. Kaur, I. Chana, Cloud based intelligent system for delivering health care as a service, Comput. Methods Programs Biomed. 113 (1) (2014) 346–359, https://doi.org/10.1016/j.cmpb.2013.09.013, URL:http://www.sciencedirect.com/science/article/pii/S0169260713003209.
[14] D. Giansanti, S. Morelli, G. Maccioni, M. Brocco, Design, construction and validation of a portable care system for the daily telerehabiliatation of gait, Comput. Methods Programs Biomed. 112 (1) (2013) 146–155, https://doi.org/10.1016/j.cmpb.2013.06.001, URL:http://www.sciencedirect.com/science/article/pii/S0169260713001843.
[15] M. Mendoza, I. Bonilla, E. González-Galván, F. Reyes, Impedance control in a wave-based teleoperator for rehabilitation motor therapies assisted by robots, Comput. Methods Programs Biomed. 123 (2016) 54–67, https://doi.org/10.1016/j.cmpb.2015.09.016, URL:http://www.sciencedirect.com/science/article/pii/S0169260715002436.
[16] I. Merelli, H. Pérez-Sánchez, S. Gesing, D. D'Agostino, Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives, BioMed Research International, Hindawi..
[17] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, R. Buyya, An autonomic cloud environment for hosting ECG data analysis services, Future Gener. Comput. Syst. 28 (1) (2012) 147–154.
[18] N.A. Risso, A. Neyem, J.I. Benedetto, M.J. Carrillo, A. Farías, M.J. Gajardo, O. Loyola, A cloud-based mobile system to improve respiratory therapy services at home, J. Biomed. Inf. 63 (2016) 45–53, https://doi.org/10.1016/j.jbi.2016.07.006, URL:http://www.sciencedirect.com/science/article/pii/S1532046416300600.
[19] F. Fernandes, P. Marques, R. Magalhães, N. Sousa, V. Alves, New Advances in Information Systems and Technologies: Volume 2, Springer International Publishing, Cham, 2016, Ch. Enabling Data Storage and Availability of Multimodal Neuroimaging Studies—A NoSQL Based Solution, pp. 107–116.
[20] Z. Goli-Malekabadi, M. Sargolzaei-Javan, M.K. Akbari, An effective model for store and retrieve big health data in cloud computing, Comput. Methods Programs Biomed. 132 (2016) 75–82, https://doi.org/10.1016/j.cmpb.2016.04.016, URL:http://www.sciencedirect.com/science/article/pii/S0169260715301632.
[21] D. Sengupta, P. Arora, S. Pant, P. Naik, Design of dimensional model for clinical data storage and analysis, Appl. Med. Inf. 32 (2) (2013) 47–53.
[22] New Chip Alert: RTL8710, A Cheaper ESP8266 Competitor.http://hackaday.com/2016/07/28/new-chip-alert-rtl8710-a-cheaper-esp8266-competitor, 2016..
[23] ESP8266EX is among the most integrated Wi-Fi chips in the industry, by Espressif –https://espressif.com/en/products/hardware/esp8266ex, 2016..
[24] ESP32 Coming Soon, by Espressifhttps://espressif.com/en/support/explore/get-start/esp32, 2016..
[25] Espressif Named Cool Vendor in IoT 2016 by Gartnerhttps://www.gartner.com/doc/3311335/cool-vendors-%20iot-thingi%20cation..
[26] M. Fazio, M. Villari, A. Puliafito, IP address autoconfiguration in ad hoc networks: design, implementation and measurements, Comput. Netw. 50 (7) (2006) 898–920.
[27] M. Fazio, A. Bramanti, A. Celesti, P. Bramanti, M. Villari, A hybrid storage service for the management of big e-health data: a tele-rehabilitation case of study, in: Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet '16, ACM, New York, NY, USA, 2016, pp. 1–8, https://doi.org/10.1145/2988272.2988276.
[28] H. van Hedel, A. Meyer-Heim, C. Rüsch-Bohtz, Robot-assisted gait training might be beneficial for more severely affected children with cerebral palsy, Develop. Neurorehab. 19 (6) (2016) 410–415.