Introducing a Conceptual Framework for Architecting Healthcare 4.0 Systems

Aleksandar Novakovic[*, 1], Adele H Marshall[*, 1] and Carolyn McGregor[2, 3]

[*] Sharing First Authorship
[1] School of Mathematics and Physics, Queen's University Belfast, United Kingdom
{a.novakovic, a.h.marshall}@qub.ac.uk
[2] Faculty of Business and IT, Ontario Tech University, Oshawa, Canada
[3] Faculty of Engineering and IT, University of Technology, Sydney, Australia
c.mcgregor@ieee.org

Abstract. There is an enormous number of healthcare analytics applications in existence which have been embedded into healthcare systems with varying degrees of success. One of the key challenges is their need for access to sensitive patient data in a healthcare system that has a multitude of healthcare applications. This paper introduces a new theoretical framework as an architecture in which Healthcare 4.0 applications can operate. The framework proposes using Apache Kafka as the core technology for creating data integration pipelines with the goal being to bring standardisation into the healthcare systems. The architecture offers a safe and secure environment in which multiple applications and algorithms from different organisations can seamlessly co-exist.

Keywords: Healthcare 4.0, Apache Kafka, big data, data pipelines, real-time analytics.

1 Introduction

We live in the era of Big Data, where enormous amounts of information is collected each second, in both structured and unstructured formats across a number of different platforms and devices. The data underpins all modern enterprises nowadays, and the healthcare industry is no different. When presenting at the Doctor's surgery with symptoms, it is the data about the patient that the Doctor uses to make an informed diagnosis of their condition and likewise it is data that informs the treatments and medications that should be administered and the follow up during recovery.

This data revolution is impacting significantly on the healthcare industry. The ever evolving health sector consists of a number of inter-related processes whose change not only has an impact on the overall healthcare delivery of care and services but also impacts on the clinicians, healthcare providers and ultimately, the patient. This complexity of co-existing multiple processes can benefit from big data analytics. In fact, a health sector that fully integrates big data analytics is essential.

The past seven years has seen the introduction, expansion and maturing of big data analytics in healthcare research and practice. In particular it offers data tools that can

collate, manage and analyse vast amounts of data, structured and unstructured in nature [1]. There are many sources from which healthcare data can be generated, from clinical, to genomic, or pharmacological to behavioural hence emphasizing just some of the variety in healthcare data available. The data is quite often collected and stored across multiple systems which may well be placed in a number of different physical locations and organisations such as healthcare centres, hospitals, government departments, and research labs. Each one of these organisations is being overwhelmed by the continuous increase in overall data volume and speed at which it is being generated, illustrating the velocity of big data in healthcare. Data repositories are also experiencing growth in size and complexity so, not only by the variety, volume, and velocity, but also by veracity that exists due to data inconsistency. Such characteristics are commonly known, well versed features of big data and well accepted concepts of any modern day system.

The extra consideration in the healthcare domain, is the sensitivity of personal data and the need for a platform that encompasses vast amounts of personal data both from patient records and from medical devices, and transforms it into intelligent healthcare systems. The smart environment needs to inform, personalise and support diagnosis and treatment pathways. In this context, any Healthcare 4.0 system shall take one or both of the following two forms: (1) the real-time analysis helps to find out irregularities in the collected data and act as fast as possible to prevent undesired consequences on the patient's health, or (2) the long-term analysis uses the massive data collected from Internet of Things (IoT) devices to uncover insights and identify trends and opportunities.

Consider a hospital setting where there is a central system that records the patient information for a busy Intensive Care Unit. Data will be recorded by the clinical staff regarding the patient condition and treatment alongside data streaming onto the system from medical devices such as a ventilator recording critical information on the ventilation being administered to the patient. It is impossible for a clinician to view all the data for a specific patient however, real-time analysis of the data can create early alarms to alert the clinician of a change in condition and additionally the data can be used to identify any underlying trend or gradual change in the patient condition. This can act as a decision support mechanism for the clinician and potentially flag up certain characteristics that would otherwise go unnoticed. Likewise, in the community, patients currently diagnosed with type I diabetes can have data recorded from their mobile devices such as their insulin pumps that can calculate the required dose of insulin which is monitored in real-time but also can be used in long term analysis to consider the long term risk of developing one of the possible multiple complications associated with the condition.

It has previously been predicted that quality of care of the patient and the overall efficiency of the system will be vastly increased with the full implementation of Electronic Health/Care Records (EHRs/ECRs) along with the systematic collection of physiological data by healthcare providers [2]. However, this is not yet the reality, despite advances in data collection and storage [3]. The key challenges are within the implementation of new approaches to inform decisions based on vast amounts of data and the ability to embed such new algorithms into the healthcare system. Even to this day, clinical decision support systems are not being used to their full potential and are being restricted to draw from just one data set or have predefined rules embedded within.

Our previous research considered real-time clinical decision support systems (CDSS) identifying the lack of current relevant metrics and clinical feedback as the biggest hindrance to the development of real-time CDSS [4] [5]. This motivated us to develop a set of new performance analytics techniques, with particular emphasis on CDSS for improving the quality of care of mechanically ventilated patients. This resulted in new suitable metrics to evaluate a CDSS working as decision support to the clinicians. But, what happens when there is more than one CDSS, or when there are several different algorithms performing different functions in the one healthcare system? Another challenge is how the algorithms potentially share and use knowledge from one another while protecting patient data, confidentiality and intellectual property.

Although there are many studies, proposing architectural solutions for various use cases, such as mechanical ventilation [5] [6], and neonatal care [7], all of these solutions lack flexibility as they are tightly coupled to existing clinical systems. Furthermore, the systems are very complex in nature, each using a different blend of technologies, and cloud services which makes practical implementation of proposed systems and collaboration between research teams difficult, if not impossible. For example, the published proposal may rely on using Python programming language and cloud-based technologies and services, while employees in the IT department of the hospital are specialized in .NET technologies, and may not have access to the cloud applications. This results in a system that cannot be accessed or implemented by the hospital team and so the proposed solution and benefits of improved quality of care are not realised.

In order to reap the full benefits of new healthcare 4.0 systems, such hurdles need to be overcome to release the users and researchers from the ongoing challenges of technology differences, and dealing with sensitive personal data. To the best of our knowledge there is no paper that proposes a solution that would enable easier access to the healthcare data and collaboration between medical practitioners and researchers. We propose a unique approach that utilizes the Apache Kafka data streaming platform as the underpinning technology to build a conceptual framework with the goal to provide a set of guidelines for overcoming such challenges.

This paper is organised as follows. Section 2 provides a brief overview of Apache Kafka discussing the challenges related to architecting and building big data pipelines in general. Section 3 addresses these challenges by introducing a framework as a set of guidelines for building scalable, secure and fault tolerant data pipelines particularly for the Healthcare 4.0 industry. Section 4 provides an overview of related work and describes how our architectural solution differs from anything that has been done previously and Section 5 concludes the paper and offers our thoughts on future research.

2 Relevant Theoretical Concepts

2.1 Apache Kafka

Apache Kafka [8] is an open-source distributed messaging platform [9] built for collecting and distributing large volumes of data, at high velocities. The entire Kafka's ecosystem is based on the Producer-Consumer messaging pattern [10] characterised by five key components: message, topic, producer, consumer and broker.

The *messages* are basic data units within the ecosystem, which are generated by the processes called *producers*. In order to achieve the efficiency in the performance, the producers send messages in batches to the centralised *cluster* consisting of single or multiple servers (also known as *broker/s*) where they get organised into categories called *topics*. Using the RDBMS terminology, the closest analogy to explain the concepts of messages and topics would be rows and tables in the database, respectively.

For achieving high scalability and redundancy, each topic can furthermore be segregated into multiple *partitions* and each partition replicated across multiple brokers within the cluster to obtain performance far superior to the ability of a single server. It is important to highlight that all of the messages belonging to a single batch are published to the same topic and partition in an append-only manner.

Opposite to the producers, the *consumers* are the processes that are used for reading messages from single or multiple topics, in the same order as they are being produced (i.e. first in first out – FIFO principle).

Customised producers and consumers in Apache Kafka can be defined either by using low-level Producer-Consumer API or by using the higher level Connect framework. The former approach is used in instances where researchers have full access to the underlying systems' programming logic and are able to modify the code of the application they want to connect an application to, so that they can either push data into or pull data from Kafka. Alternatively, the Connect framework is used for the scalable and reliable streaming data between Apache Kafka and other externally managed datastores that are not necessarily written by the researchers for which the code cannot be modified [11].

Data streaming between Kafka and other external datastores is performed using processes called *connectors*, which can be either *Source* or *Sink Connectors*. Source connectors are used for ingesting entire data sources (e.g. relational and non-relational databases, key-value stores, file systems, search indexes, etc) and streaming the updates to Kafka topics when these occur, while sink connectors are used for delivering data from Kafka topics into destination data sources (e.g. warehouses, data lakes, etc.) for batch analysis [12]. A plethora of source and sink connectors that can be used for connecting Kafka with various data sources can be found on the Confluent Hub portal [13].

Features that differentiate Kafka from other similar Producer-Consumer messaging systems (e.g. RabbitMQ, ActiveMQ, etc.) are the way in which the messages are immediately purged upon consuming, and its unique capability to persist the topics and their messages on disk for some configurable amount of time or until the designated storage space is filled. This feature enables consumers to replay the messages when needed which is crucial for the fault tolerance of the downstream systems and can facilitate longer term analytics. All persistence settings can be tuned for each topic separately, and upon exceeding either the allowed disk's quota or the retention time, the messages are automatically deleted from the system [14].

2.2 Challenges in Building Data Integration Systems

Narkhede et al. [11] state that the most important characteristics to take into consideration when designing data pipelines with a focus on integrating multiple systems, are:

timeliness, security, reliability and scalability. Apache Kafka meets all of these requirements as the technology for the implementation of these data integration systems:

1. **Timeliness –** Generally speaking, the timeliness characteristic refers to the capability of the data integration systems to provide support for the variable consumption needs of different consumer systems. For instance, some consumers might expect to receive their data within just a few milliseconds of its generation, while others may wish to receive it weekly in bulk. Bearing in mind that Apache Kafka is a distributed messaging platform with scalable and reliable storage capabilities, it can act as a huge buffer for received messages enabling decoupling time-sensitivity requirements between the producers and consumers. This allows producers to write to the Kafka cluster as frequently or infrequently as required, and consumers to read and deliver the messages either as they arrive, or to work in batches and read the messages that were accumulated in the cluster over time or all at once [11].
2. **Security –** When it comes to the data integration pipelines, the main security considerations are those related to the: (i) encryption, (ii) authentication, and (iii) authorisation. Kafka provides support for SSL encryption of the data as it is transferred from the data sources to Kafka topics or from the topics to sinks, which is of huge importance especially when the data cross data centre boundaries. To prevent unauthorised and unauthenticated access to the data, Kafka supports the implementation of role-based access control (RBAC) authorisation and SASL authentication mechanisms. Additionally, Kafka also provides audit logs to track access [11].
3. **Reliability –** The main reliability concern is the design of data integration pipelines that avoid single points of failure and permit fast and automatic recovery from all kinds of failure events. In Kafka the data delivery reliability can be ensured through two different delivery mechanisms: *at-least-once* and *exactly-once delivery* [11].
4. **Scalability –** The requirement for the data integration pipelines is to support very high throughputs and to be able to scale out in order to support increased messaging loads, when it is needed. By acting as a buffer between producers and consumers, Kafka does not require the coupling of consumer throughput to producer throughput, as is the case with many other messaging brokers. Instead because of its capabilities to accumulate received messages on disk, Kafka has the ability to scale either side of the pipeline by adding consumers or producers independently and thus matching the changing throughput requirements [11].

3 A Conceptual Framework for Architecting Healthcare 4.0 Applications

In this section we introduce the conceptual framework with the primary goal of bringing in the standardisation and ease that is required for the development of Healthcare 4.0 applications. Our proposal extends and enhances the current development of healthcare applications which follows a three-tiered architecture consisting of the Data Emitting Layer (DEL), the Healthcare Gateway Layer (HGL) and the Application Layer (APL).

As the name suggests, DEL includes any possible healthcare data emitting mechanism which includes, but is not limited to, devices for collecting medical images (X-Rays, CAT Scans, Magnetic Resonance, etc.), devices for collecting sensory readings such as those necessary for vital signs monitoring (e.g. pulse rate, respiration rate, blood pressure, body temperature, etc.), or data from the healthcare professionals who, having collected information about the patient, are in charge of writing the diagnosis, prescriptions or admitting and discharging patients to and from the hospital. Depending on the source and type of the collected data, using various protocols (such as TCP, HTTP, MQTT, Bluetooth, etc.) this information is then transported to HGL where it is usually persisted in the electronic medical records (EMR), Hadoop File System (HDFS), AWS S3, or similar storage solutions for further processing and analysis. The healthcare providers are the key stakeholders so have to maintain both DEL and HGL, due to the sensitivity of the patient information. The researchers and academic collaborators outside of these organisations need to pass rigorous security checks and obtain special data access rights to be able to utilise just a small portion of the information for the purpose of developing clinical decision support systems and other advanced applications in the APL. Each healthcare organisation has its own set of internal policies hence making the requesting of these permissions and eventual access to the data, a very lengthy process which generally can take several months. If we take into account that every healthcare organisation will utilise a different combination of technologies and programming languages for running their internal IT infrastructure (.NET vs JVM vs other programming stacks, cloud vs on-premise deployment, different cloud providers if the cloud is used, relational vs non-relational databases for storing information and different vendors of these technologies, etc.), research organisations need to be prepared for a huge degree of flexibility and adaptability in order to start using the data from these systems. Cumulatively this all causes a serious negative impact to the speed of development of Healthcare 4.0 applications, thus raising the need for creating a standardised framework for easier collaboration between the healthcare providers and research organisations.

To overcome these issues, we propose adding an additional layer to decouple communication between HGL and APL. We name this extra layer the Data Pipeline Layer (DPL) and due to the reasons previously described, we selected Apache Kafka as its underpinning technology. Fig. 1 provides an overview of the proposed architecture.

To facilitate data transfer between Kafka and existing data sources in HGL and APL we propose using Kafka Connect. The main motivation for proposing this framework is the fact that it provides out-of-the-box features like configuration management, offset storage, parallelization, error handling, support for different data types, and most importantly it is extremely flexible as it can be used by non-developers who would only need to configure the connectors for communication with the data sources [11].

The healthcare providers (data owners) should be in charge of maintaining the Kafka Cluster and source connectors. Due to the advanced security capabilities which enable data encryption and protection of data stored in Kafka, from unauthorised and unauthenticated access, healthcare providers are able to pre-plan what datasets they are willing to share with potential research collaborators and have a fine grain control over adequate data access rights. Research organisations, with the right data access permis-

sion would be able to start using healthcare data by attaching their preferred sink connector to the Kafka cluster provided by the healthcare organisation. Similar to how data exchanges between HGL and APL, if the data sharing agreement permits, Apache Kafka could be used for establishing the connection, hence fostering the collaboration between the research organisation in the Research Collaboration Layer (Fig. 1).

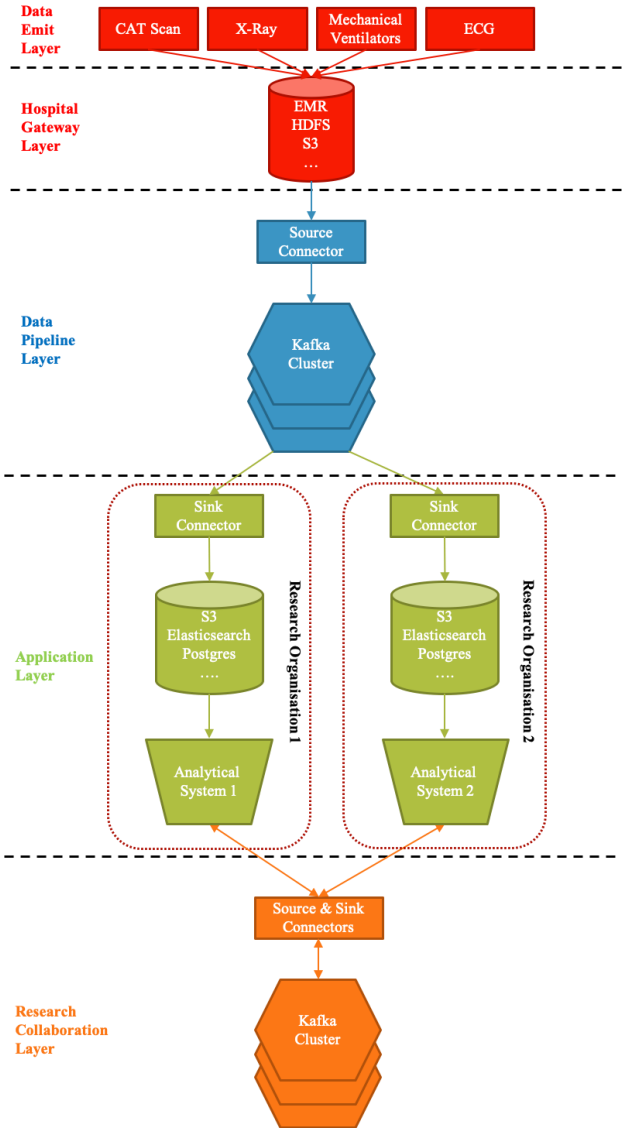

Fig. 1. A conceptual framework for architecting Healthcare 4.0 Applications

4 Related Work

The Apache Kafka has been increasingly used as a distributed streaming platform in real-time processing of IoT events [15], Industry 4.0 [16], and smart cities [9].

Gokalp et al. [17] created a real-time patient monitoring system which collected patients' vital sign parameters heart rate, blood pressure, respiration, skin temperature, and blood oxygen level SPO2 using IoT devices. Kafka was used as a message broker to distribute the data from IoT devices to Apache Storm [18] which processed the data in real time and warned clinicians if collected values crossed predefined thresholds.

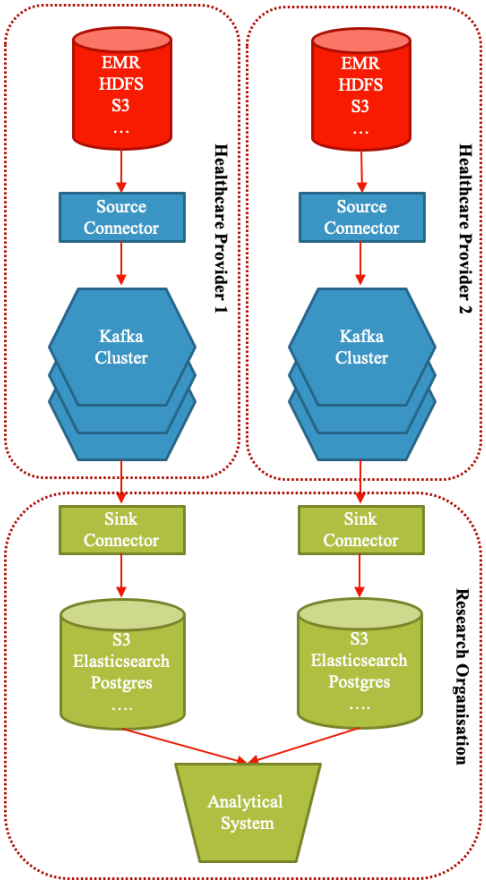

Fig. 2. A research organisation utilising data from two different healthcare providers

There are two fundamental differences between their work and the conceptual framework proposed in this paper. The first difference is in the way that Kafka communicates with the DEL. Gokalp et al [17] authors have full access to the underlying systems' programming logic and are able to modify the application code they want to connect to so that they can push data into Kafka directly from the IoT devices. The majority of

Healthcare 4.0 projects do not have this condition as the healthcare providers are the data owners. Hence, the Kafka Connect framework is a better alternative.

The second difference is in the transformations that the pipelines perform. Gokalp et al. [17] use the ETL (extract-transform-load) approach where the data pipeline, or in their case Apache Storm [18], makes modifications to the data as it passes through. The main disadvantage of this method is that valuable information is getting lost in the process which automatically creates restrictions on those who want to process the data further down the pipeline [11]. Hence, in our paper we propose using the ELT (extract-load-transform) process instead providing maximum flexibility to users of the target system, since they have access to all the data in the original format [11]. Fig. 2 provides an illustration of the proposed architecture for a research organisation utilising data for their healthcare analytics applications from two different healthcare providers.

The Artemis platform makes use of the Vines device connectivity software to enable Data Acquisition from medical devices within neonatal ICUs [19]. Vines utilises RabbitMQ. Vines was chosen due to the proprietary messaging protocol for the output signals from many medical devices including those used within the collaborating NICUs for the Artemis deployments. Decoupling of the Data Collection and Data Acquisition components from the remaining analytics, data storage and visualisation components of Artemis was proposed in [20]. Both Vines and Python scripts were assessed for their applicability for the Data Acquisition function within the context of low resource settings with a case study context of the NICU within Belgaum Children's Hospital, India. The component nature of Artemis enables the Vines component to be easily replaced by the architecture proposed in this paper.

A key challenge for different healthcare providers is ensuring that new medical device procurement procedures ensure procured support output of data streams and utilise standardised and well documented messaging protocols rather than proprietary messaging approaches.

5 Conclusions and Further Directions

This paper introduces a novel conceptual framework as a consistent architecture for Healthcare 4.0 applications. The framework proposes Apache Kafka as the core technology for creating data integration pipelines bringing standardisation to the healthcare systems enabling easier communication between healthcare providers and researchers.

The architecture offers a safe and secure environment in which multiple applications and algorithms from different organisations and providers can seamlessly "plug" into the healthcare providers Kafka "socket" to utilise sensitive data without any issue with data access, or security. Each healthcare provider has their "socket" created for each organisation and likewise each organisation may have multiple "sockets" in which they can "plug" into to access the relevant healthcare data set and provide their service.

Further directions of the research is to deploy previously designed healthcare analytics applications to the new architecture in a hospital setting.

Acknowledgment

This work has been supported by Queen's University Belfast, United Kingdom and Ontario Tech University, Canada.

References

1. A. Belle, R. Thiagarajan, S. M. R. Soroushmehr, F. Navidi, D. A. Beard, K. Najarian, "Big Data Analytics in Healthcare," *Hindawi Publishing Corporation*, vol. 2015, pp. 1-16, 2015.
2. T. D. Gunter and N. P. Terry, "The Emergence of National Electronic Health Record Architectures in the United States and Australia: Models, Costs, and Questions," *Journal of Medical Internet Research*, vol. 7, no. 1, p. e3, 2005.
3. A. S. Kellerman, "What it will take to achieve the as-yet-unfulfilled promises of health information technology," *Health Affairs*, vol. 32, no. 1, pp. 63-68, 2013.
4. A. H. Marshall and A. Novakovic, "Analysing the Performance of a Real-Time Healthcare 4.0 System using Shared Frailty Time to Event Models," in *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, Cordoba, Spain, 2019.
5. A. Novakovic and A. H. Marshall, "Introducing the DM-P approach for analysing the performances of real-time clinical decision support systems," *Knowledge-Based Systems*, vol. 198, *pp*. 105877, 2020.
6. C. J. Gillan, A. Novakovic, A. H. Marshall, M. Shyamsundar and D. S. Nikolopoulos, "Expediting assessments of database performance for streams of respiratory parameters," *Computers in Biology and Medicine*, vol. 100, pp. 186-195, 2018.
7. S. Balaji, M. Patil and C. McGregor, "A Cloud Based Big Data Based Online Health Analytics for Rural NICUs and PICUs in India: Opportunities and Challenges," in *2017 IEEE 30th International Symposium on Computer-Based Medical Systems (CBMS)*, Thessaloniki, Greece, 2017.
8. "Apache Kafka," [Online]. Available: https://kafka.apache.org. [Accessed 28/02/20].
9. J. Y. Fernandez-Rodriguez, et al., "Benchmarking real-time vehicle data streaming models for a smart city," *Information Systems*, vol. 72, pp. 62-76, 2017.
10. C. Barba-Gonzales, et al., "On the design of a framework integrating an optimization engine with streaming technologies," *Future Generation Computer Systems*, vol. 107, pp. 538-550, 2020.
11. N. Narkhede, G. Shapira and T. Palino, Kafka: The Definitive Guide, O'Reilly Media, Inc, 2017.
12. "Kafka Connect," [Online]. Available: https://bit.ly/2SSpLEH. [Accessed 28/02/20].
13. "Confluent Hub," [Online]. Available: www.confluent.io/hub/. [Accessed 28/02/20].
14. P. Dobbelaere and K. S. Esmaili, "Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper," in *DEBS '17: Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, Barcelona, Spain, 2017.
15. D. Plaza-Corral, I. Medina-Bulo, G. Ortiz and J. Boubeta-Puig, "A stream processing architecture for heterogeneous data sources in the Internet of Things," *Computer Standards & Interfaces*, vol. 70, pp. 1-13, 2020.

16. R. Sahal, J. G. Breslin and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *Journal of Manufacturing Systems*, vol. 54, pp. 138-151, 2020.
17. M. O. Gokalp, A. Kocyigit and E. Eren, "A visual programming framework for distributed Internet of Things centric complex event processing," *Computers and Electrical Engineering*, vol. 74, pp. 581-604, 2019.
18. "Apache Storm," [Online]. Available: https://storm.apache.org. [Accessed 28/02/20].
19. C. McGregor, C. Inibhunu, J. Glass, I. Doyle, A. Gates, J. Madill and J. E. Pugh, "Health Analytics as a Service with Artemis Cloud: Service Availability," in *Proceedings of the 42nd IEEE Engineering in Medicine and Biology conference*, Montreal, Canada 2020.
20. M. Bastwadkar, C. McGregor, S. Balaji, "A Cloud Based Big Data Health-Analytics-As-a-Service Framework to Support Low Resource Setting Neonatal Intensive Care Unit", in *Proceedings of the 4th International Conference on Medical and Health Informatics (ICMHI 2020)*, Kamakura City, Japan, 2020