Towards a Generic Cloud-based Sensor Data Management Platform: A Survey and Conceptual Architecture

Vincent C. Emeakaroha, Kaniz Fatema, Philip Healy, John P. Morrison
Irish Centre for Cloud Computing and Commerce (IC4)
University College Cork, Ireland
Email: {vc.emeakaroha, k.fatema, p.healy, j.morrison}@cs.ucc.ie

Abstract—**With the increasing numbers of sensors and smart devices being deployed worldwide, the volume of data they generate is becoming difficult to store and process on local platforms. Cloud computing provides scalable resources that could address these issues. However, platform-independent methods of gathering and transmitting sensor data to Clouds are not widely available. This paper presents a survey of Cloud-based sensor monitoring and data gathering platforms. It reviews the state-of-the-art and discusses their strengths and weaknesses. Informed by the survey, the paper further proposes a generic conceptual architecture for achieving a platform-neutral Cloud-based sensor monitoring and data gathering platform. We also discuss the objectives, design decisions and the implementation considerations for the conceptual architecture.**

Keywords*–*Sensor Monitoring Platforms; Sensor Data Gathering; Cloud Computing; Generic Sensor Cloud Platform; Interoperable Communication

I. Introduction

Data gathering IT techniques such as those underlying entity tracking and control systems, for example, have shown the commercial value of real-time control of real-world devices. Based on this, more generalised applications for sensor devices are becoming significant in real world usages. Sensors enable access to remote objects and environmental information providing the raw materials from simple monitoring through to next-generation applications such as smart cities. The gathering, storage and processing of sensor data locally is becoming very costly due to the increase in the number of sensors and the volume of data they generate.

In a similar manner, the Internet of Things (IoT) [1], which promise to connect objects, devices and humans, generate large volumes of data. Harnessing these data by organisations can be a complex process due to heterogeneous operating systems, varying connectivity protocols and legacy application compatibility. Furthermore, the ability to draw meaningful insights from the volume of data unleashed by these technologies is another challenge.

Cloud platform offers scalable compute and storage resources to support the management of these data [2]. However, solutions [3]–[6] available today are mostly customized for particular usages. To unlock the business potential in this area, generic solutions are required to address the core challenges such as communication bottle-neck, data interchange formats, security and interoperability.

This paper presents a survey reviewing the state-of-the-art of Cloud sensor monitoring platforms and data gathering techniques. Based on the survey findings, it further proposes a platform-neutral architecture providing a generic Cloud interface that addresses the identified challenges. The main contributions are (i) a survey of industrial, commercial and open source sensor monitoring and data management platforms; (ii) analysis of the platforms to identify trends, issues and challenges; and (iii) the proposal of a generic architecture utilising standardised data interchange formats and interoperable communications.

The rest of the paper is organised as follows: Section II presents the related work. The state-of-the-art analysis is performed in Section III. Section IV discusses the identified challenges and objectives for a platform-neutral architecture while Section V presents the proposed architecture and its implementation considerations. In Section VI, we conclude the paper and discuss the follow-up work.

II. Related Work

In this area, previous efforts have been focused mostly on particular issues like the virtualisation of physical sensors on Clouds, data privacy and security or the provision of efficient platforms for sensor data storage and processing. Catrein *et al.* [7] propose a Cloud design for user-controlled storage and processing of sensor data to ensure privacy. They identified the importance of using the Cloud for sensor data processing. However, their approach focuses on security-related issues such as data privacy and access control. Aoki *et al.* [8] present a Cloud architecture to enable fast response to real world applications in spite of the flood of sensor data. The authors used the strategy of reducing network latency to achieve this goal but they did not consider creating a generic interface for the diverse sensor data gathering.

Piyare *et al.* [9] propose an architecture for integrating Wireless Sensor Networks (WSN) into Cloud services for real time data collection. In their approach, WSN is considered an important paradigm for Internet of Things since they consist of smart sensing nodes with embedded Central Processing Units (CPU) and sensors for monitoring different environments. This work concentrated on connecting WSNs to Clouds and does not consider other sensor types. The authors in [3] [4] discuss particular approaches for gathering medical sensor data. Alamri *et al.* [10] present a survey on sensor-Cloud, architecture, applications and approaches. The survey analyses the current efforts and challenges in this area. It shows that many of the existing efforts are geared towards creating virtual sensors from physical ones on Clouds.

Salehi *et al.* [11] present Global Sensor Network (GSN), which is a middleware to interconnect diverse sensor network technologies. This work focus on providing a mechanism for easy integration of existing sensor networks. With this approach, the management of sensor networks are simplified. For example, changing or updating components within a sensor network does not interfere or hinder communications with other sensor networks. However, it does not consider the gathering of single sensor data. Other efforts like the Sensor Observation Service (SOS) [12] provide standards in accordance with the Open Geospatial Consortium (OGC) for discovery and retrieval of real-time data from diverse sensors in the context of geospatial data processing and sharing.

Thus, to the best of our knowledge, none of the existing work provide a generic Cloud-based monitoring platform for gathering sensor data. In the next section, we present our analysis of the state of the art.

III. State-of-the-Art Analysis

This section details the survey of the existing tools and platforms for monitoring, gathering and processing sensor data. It highlights their characteristic features and practical use cases.

A. *IBM Mote Runner*

Mote Runner is IBM's infrastructure for WSNs [13]. It is based on a virtual machine targeted to resource-constrained hardware environments and consists of two parts: a run time for mote-class hardware, such as Libelium Waspmote or MEMSIC Iris motes, and a development environment for WSN applications.

At its core, Mote Runner is designed to run on very small, standard, embedded controllers, including low-power 8-bit processors, thereby reducing both initial investments as well as post-deployment and maintenance costs. It has been shown to be light in energy consumption [14]. It provides a high-level, language-friendly, resource-efficient and high-performance Virtual Machine (VM) that shields portable applications from hardware specifics. In addition, Mote Runner allows programmers to use object-oriented programming languages such as Java and a development environment based on Eclipse to develop portable WSN applications that may be dynamically distributed. Its features include [15]:

- Low-power,
- Support for harvesting solar power as a source of energy,
- Wirelessly connected embedded systems,
- Provides Software Development Kits (SDK) for developers,
- Supports multiple high-level languages such as Java and C#,
- Runs on 8 bit micro-controllers with as little as 4 KB of Random Access Memory (RAM) and 32 KB of flash memory,
- Leverages integrated development environments such as Eclipse, Visual Studio and MonoDevelop.

This tool focuses on WSN related applications and mote actuators. However, little or no information was available about its support for wired or other sensor types. The details about the used data interchange formats are not publicly available.

B. *SensorCloud Platform*

MicroStrains SensorCloud offers a sensor data storage, visualisation and remote management platform that leverages Cloud computing technologies to provide data scalability and rapid visualisation [16]. It was initially designed to support long-term deployments of MicroStrain wireless sensors. However, it now supports any web-connected third party devices, sensors, or sensor networks through a simple OpenData Application Programming Interface (API). It aims to provide virtually unlimited storage for the sensor data [17]. Its features include:

- Presumedly unlimited data storage with triple-redundant reliability,
- A time series visualisation and graphing tool,
- A MathEngine analytic tool facilitating quick user application development using their data on the Cloud,
- Provides Short Message Service (SMS) and Electronic mail (E-mail) alerting capabilities,
- Provides OpenData API and Representational State Transfer (REST) API for data transport,
- Supports only eXternal Data Representation (XDR) and Comma Separated Value (CSV) data interchange formats.

Some of its usage scenarios include structural health monitoring and condition based monitoring of high value assets where commonly available data tools are often not capable in terms of accessibility, data scalability, programmability, or performance. All communication with the SensorCloud is performed over Secure HyperText Transfer Protocol (HTTPS), which is secure. It restricts however, other forms of interactions such as low-level communication thereby forcing the use of HyperText Transfer Protocol (HTTP)-capable middleware to connect the devices and the platform. Furthermore, XDR and CSV are the only data interchange format types currently supported.

C. *Ostia Portus Platform*

Ostia Portus is designed to mediate between multiple vendor technologies where each vendor has particular platforms, databases and programming languages. It achieves this by taking the isolated data sets from individual systems and packaging them into a standard service [18]. This platform connects a variety of devices including sensors, networks and platforms. Its features include:

- Support for relational databases,
- Uses Secure Socket Layer (SSL) over HTTP to achieve security,
- Supports Simple Object Access Protocol (SOAP), REST, Publication/Subscription (PUB/SUB) protocols,
- Supports Java Message Service (JMS), RabbitMQ, Transmission Control Protocol (TCP)/Internet Protocol (IP), Message Queue (MQ) transport protocols,
- Easy installation and use.

Portus is built using open technologies and exploits open standards in accessing organisations data and presenting them using business defined views. Its core components are: (i) server written in C and C++ and hosted by Apache; (ii) control centre for administration that is written in Java and built to run with the Eclipse Framework and (ii) front-end providing web services.

D. TempoDB Platform

TempoDB is a database as a service aiming to store and analyse time series data from sensors, smart meters, servers and automotive telematics. It is a commercial tool consisting of the following features [19]:

- Simple REST API for data storage and retrieval,
- Allows data storage at full resolution (no downsampling),
- Guarantees data availability with its three times data replication,
- Offers SSL encryption for all data transfer,
- API clients available in multiple programming languages like Java, .Net, Python, Ruby,
- Supports Internet of things.

The primary aim of TempoDB is the management of time series data sets with timestamps in ISO8601 format [20]. In querying data from TempoDB storage using a client API, the returned data is formatted only as JavaScript Object Notation (JSON). Other data interchange formats are currently not available.

This platform is currently expanded and renamed into TempoIQ [21]. It now offers flexible sensor data monitoring and alerting mechanism. The alerts are based on thresholds and informs the user about the status of its applications that are using the sensor data.It offers also analytic tools to support user applications.

E. FreshTemp Temperature Monitor

FreshTemp is a Cloud-based monitoring system for perishable goods [5]. It automates temperature collection during production, transportation and storage of any perishable product by providing the capability of integrating with the different temperature sensors monitoring such products. It offers real-time data logs, configurable alerts and online Dashboard. It is a commercial tool aiming to provide solutions for food services, transportation, health care and industrial usages. Its core features include:

- Wireless temperature monitoring,
- Bluetooth food probes,
- Bluetooth data loggers,
- Real time data logs,
- Alerting mechanism with SMS E-mail and phone,
- Online dashboard.

This tool focuses solely on temperature sensors, which limits its usability for managing and controlling other sensor device types. Furthermore, it does not offer any programming API for application development or any means of accessing the data stored on the platform by developers.

F. SensaTrack Monitor

SensaTrack is a multi-platform independent monitoring service that is ideal for Machine-to-Machine (M2M) sensor monitoring. It is developed by Cannon Water Technology Inc. and released in November 2012 [6]. The SensaTrack Cloud-based monitoring software solution allows users to visualize sensor data from any web enabled device. It is designed for enterprises with distributed assets like chemical storage facilities, bulk storage tanks, bins and silos. Its features include:

- Ability to quickly scan multiple locations,
- Secure data servers,
- Wireless data gateways,
- Track trends and find problems early,
- Easy installation by non-technical personnel.

SensaTrack uses wireless communications equipment based on Zigbee protocols developed by the Digi Corporation. It also supports hybrid networks of wired and wireless sensors. This tool has no support for application developments and does not provide an API for external access.

G. Bluwired S-Cloud Platform

Bluwired S-Cloud provides a platform for sensor data exploration, interaction and analysis [22]. It facilitates the management of sensor and device data from any web enabled location in the world, and to deploy data processing and analysis applications that rely on gathered data on the Cloud. Its features include:

- A platform for sensor data exploration, interaction and analysis,
- Management of wireless sensor data from any web enabled location in the world,
- Support for user development and deployment of data processing and analytic applications using their data on the cloud,
- Facilitates real-time data storage and retrieval,
- Provides alerting mechanism for abnormal events,
- Provides visual management interface.

Bluwired S-Cloud promises to offer reliable storage, tracking and analysis of sensor data that comes from monitoring and control solutions for most applications, including: Factory Automation, Process Control, Agriculture and Irrigation monitoring, Patient Monitoring Systems, Oil and Gas. It is a commercial tool and does not offer an open source API for accessing the platform.

H. Xively Platform

Xively is a Cloud Service platform that harnesses the power of IoT to quickly and easily transform connected product vision into market reality. It was formerly known as “pachube” and later as “COSM” [23]. Xively is currently a division of LogMeln Inc. and strives to provide business solutions through the IoT. It offers a platform that connects devices and products with applications to provide real-time control, management and storage. Its features include:

- Secure real-time messaging,
- Time series data storage,

- Selective data sharing,
- Provides easy connection to external Cloud services like Twitter and Facebook,
- Encryption with Transport Layer Security (TLS) and SSL,
- Real-time message bus based on Message Queue Telemetry Transport (MQTT).

An example use case for Xively is the “Park-A-Lot” project [24], which is designed to support an automated parking management system. Although it is a commercial product, but, it provides open APIs and libraries for easy usage by developers to create smart applications and interacts with the platform. Xively provides effective means to reason about devices and actuators at high level but fails to provide detailed context information within which all these devices are being placed, especially when it comes to small scale setups such as individual houses.

I. Nimbits Platform

Nimbits is a platform as a service (PaaS) for developing software and hardware solutions that seamlessly connect to the Cloud and each other. It has the ability to record and share sensor data on the Cloud [25]. Within Nimbits, sensor data are stored as data points using textual, JSON or eXtensible Markup Language (XML) formats. It provides REST web services for logging and retrieving time and geo stamped data (such as a reading from a temperature sensor). Nimbits servers can run on both powerful Cloud platforms like Google App Engine and on the smallest Raspberry Pi device. Its graphic user interface is tree structured having parent and child structures, which allow user content to be organised according to a parent-child structure and could be dragged and dropped as desired [26]. Its features include:

- Ability for recording and sharing data,
- Data storage as data points,
- Easy connection of data to analytic tools,
- Graphic user interface for visualisation,
- Ability to generate alerts based on defined thresholds or events.

Nimbits is an open source platform for the Internet of things. It is freely available and provides libraries, APIs and documentations for different programming languages.

J. ThingSpeak Platform

ThingSpeak is an open source Internet of things platform that provides API to store and retrieve data from things using HTTP over the Internet or via a Local Area Network [25]. It has the ability to facilitate the creation of sensor logging applications, location tracking applications, or a social network of things providing status updates.

In excess to its ability to store and retrieve numeric data and alphanumeric data, its API allows for numeric data processing such as time scaling, averaging, median, summing and rounding. ThingSpeak is organised in Channels, which is where a user application can store and retrieve data. Each Channel supports data entries of up to 8 data fields. The channel feeds support JSON, XML, and CSV data formats for integration into applications. Its features include [27]:

- Open API for developers,
- Real-time data collection,
- Geolocation data gathering,
- Data processing and analytic tools,
- Data visualisations on web and mobile devices,
- Device status messages and event alerting,
- Supports diverse programming languages like Java, JavaScript, .Net, Ruby,
- Allows easy plugins integrations.

ThingSpeak offers also a hosted service that is different to the open source version. Open.Sen.se [9] is another Internet of Things tool that is very similar in characteristics and features to ThingSpeak.

K. Microsoft Azure Intelligent System Service

The Microsoft Azure Intelligent Systems Service aims to securely connect, manage and capture machine-generated data from industry devices, sensors and other line-of-business (LoB) assets across a range of operating system platforms. The intelligent service represents the efforts of Microsoft to address the challenges of IoT and to help businesses utilise its potentials.

This tool promises to offer enterprises the ability to extend their Microsoft Azure Cloud across connected devices and sensors in order to capture vital data, analyze them with familiar Microsoft tools like HD Insight and Power BI for Office 365 in order to facilitate taking quick and appropriate actions that drive impact. Its features include [28]:

- Secure connection and management of devices and data,
- Support for real-time control of heterogeneous environments and accelerated implementations,
- Supports efficient capture, store, join, analyse, visualise and share data,
- Provide a trusted platform that can be extended easily for industrial specific requirements,
- Improve operations and unlock new business opportunities by harnessing machine-generated data from connected sensors and actuators.

The intelligent service is a fully commercial software as the other Microsoft services however, it is not yet production ready. A limited preview version was released in April 2014.

L. Paho Platform

The Paho project aims to provide scalable open-source implementations of open and standard messaging protocols to facilitate new, existing, and emerging applications to enable machine-to-machine and Internet of things usage scenarios. it is a part of the Eclipse foundation and its features include:

- Enables levels decoupling between devices and applications,
- Encourages the growth of scalable web and enterprise middleware and applications,
- Supports resource constrained embedded platforms,
- Based on MQTT,

- Provide MQTT client implementations in Java, Python, C, C++,
- Provide open libraries, API and client implementations,
- Enables integration of wide range of middleware, programming languages and messaging models.

Paho strives to be a software for constrained networks, devices with limited processing resources and embedded platforms [29].

M. Platform Summary and Comparison

This section presents a summary of the identified platform features in a table, which enables a quick comparison. The analysis offers an opportunity to identify trends and the gaps in technological advancements in this area. A goal of this analysis is to organise our thoughts and to see what is available in order to design a conceptual architecture.

The described and analysed platforms represent the current efforts towards addressing the challenges of Cloud-based sensor monitoring and data gathering, and their real-time analysis in order to facilitate informed decision making and smart applications. This information is summarised in Table I.

As shown in Table I, the analysed platforms exhibit some similar characteristics in terms of data storage, support for web technologies and availability of a REST API. The commercial platforms are closed source, thus, details regarding programming languages, API, data interchange formats were not accessible. It can be observed that the platforms have poor support for resource constrained deployments, energy efficiency, communication protocol implementation, software development environments and data interchange formats. These issues represent the challenges facing the existing platforms.

The commercial tools promise many features and functionalities, but details of their implementations are completely hidden from the public, which slows down technical know-how establishment and thereby makes it difficult for developers to leverage such functionalities when creating applications. This problem obstructs market growth and poses challenges to technology adoption. Furthermore, it hinders the realisation of portable solutions facilitating interoperability among different platforms. This shows the need for generic solutions with open implementations. Openness is a means to mitigate this problem. Providing open source APIs and interfaces enables easy creation of interoperable software and assures adherence to defined standards.

Another interesting point is that most of the open source tools are developed in the context of Internet of Things. This means that they could be easily used in designing generic solutions. We therefore view the open source tools as providing a good basis for the development of open solutions supporting standardised interfaces and data interchange formats.

According to Table I, most of the open source analysed tools are using textual data interchange formats, which have the advantage of being human readable and easy to understand. However, the serialised data in those formats are not very compact in size for efficient transportation without consuming large amounts of bandwidth. This identifies the need for integrating standardised binary data interchange formats that have the ability to achieve compact serialisation of data.

In general, the commercial solutions tend to surface many features. But, they are mainly closed source and implement proprietary technologies instead of the standardised ones. Most of them also do not provide open-source API for the general public, which could enable quick adoption and development of generic solutions. On the side of the open source solutions, they provide basic features and are not quite advanced. But they offer basis for further developments and possible realisation of generic solutions.

IV. Issues and Objectives

In this section, we discuss the challenges informed from the state-of-the-art analysis and present our objectives for the proposed architecture.

A. Challenges and API Requirements

As can be observed in Table I, none of the analysed platform provides a complete set of features representing a generic and open solution. The implementation of such generic platform is complex and difficult. Many challenges exist in this spectrum especially regarding the heterogeneity of the targeted hardware and software components. At the lower levels, there is a plethora of sensor devices out there, which gather data in different formats. To design a generic interface for these data types requires a comprehensive study of the existing data types and how they could be aggregated or adapted/converted to a standard format.

At the higher levels, another challenging factor is the security and data access control on the Cloud platform. The authentication of a gateway server before allowing the forwarding of the sensor data to the Cloud platform is not enough. Many customers worry more about the privacy of their data and what the Cloud providers might do with them on the Cloud. To address these issues, different levels of security assurances are required for securing the data on the Cloud and controlling their usage.

Furthermore, the communication mechanism for transferring the sensor data and the actuator control information is challenging as shown in Table I. Such a mechanism must be simple, pluggable and reliable in order to ensure, robust and reliable communications.

B. Objectives

Currently, the ability to gather data from different sensor devices and feed them into Cloud platforms in a unified manner is lacking. This hinders the availability of raw data on computational capable platforms that can quickly and efficiently analyse the data to derive knowledge, which could support numerous usage scenarios such as real-time critical applications in health and medicine. Therefore, the goals of the generic Cloud-based sensor monitoring and data processing platform are to provide standard and open mechanism for collecting diverse remote sensor data and processing them irrespective of the sensor devices or usage platform. Furthermore, we aim to achieve:

1) Easy setup and portability,
2) Platform-neutral data formatting and serialisation,
3) Interoperability among heterogeneous Cloud platforms and
4) Simple configurations and ease of usage.

In the next section, we describe our proposed architecture.

TABLE I. SURVEYED PLATFORM FEATURES

Capability/ Features	**IBM Mote Runner**	**SensorCloud**	**Portus**	**TempoDB**	**FreshTemp**	**SensaTrack**	**Bluwired S-Cloud**	**Xively**	**Nimbits**	**ThingSpeak**	**MS Intelligent Service**	**Paho**
API/Library	SDK for implementations	OpenData API, REST API	No	REST API, client API	No	No	No	RESTful API, client libraries	REST API, libraries	Open API and libraries	No	Open API, client libraries
Data Formats	N/A	XDR, CSV	JSON, XML, IDoc	JSON	N/A	N/A	N/A	JSON, XML, CSV	Textual, JSON, XML	JSON, XML, CSV	N/A	N/A
Programming Language	Java, C#	Python, Java, C#, C++	C, C++, Java, PHP	Java, .Net, Ruby, Python	N/A	N/A	N/A	Objective C, C, Java, JavaScript, Ruby	Java, JavaScript	Java, JavaScript, Python, Ruby, .Net, node.js	.Net	Python, Java, JavaScript, C, C++
Open Source / Commercial	Commercial with open SDK	Commercial solution	Commercial Solution	Commercial with open source client	Commercial Solution	Commercial Solution	Commercial Solution	Commercial with Open API libraries	Open source solution	Open source with hosted version	Commercial solution	Open source solution
Visualisation	No	Yes and graphing tool	No, but with external web clients	Yes	Online dashboard	Web enabled	Visual management interface	Management console	Graphic user interface	Web capable devices	Yes	N/A
Analytic Tool	No	MathEngine	No	Yes, for time series data	No	No	Blu Automation Studio	Yes	No	Yes	HD Insight, Power BI	No
Messaging Protocol	N/A	No	RabbitMQ, JMS, TCP/IP, IBM MQ	No	No	No	No	MQTT	N/A	N/A	N/A	MQTT
Notification / Alert	No	SMS and email	No	No	SMS, email, phone	Text message, email	Custom messages	Yes	Yes	Yes	N/A	N/A
Energy Efficiency	Low power, harvest solar power	No	No	No	No	No	No	No	No	No	No	N/A
Connection Type	wireless	Http	Http	Http	Wireless	Wireless gateway, Zigbee, Wired	Wireless	N/A	Http	Http, wireless, Zigbee	N/A	Http
Platform Resource Requirement	Resource constrained devices, embedded controllers	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Resource constrained devices, embedded platforms
Software Development Tool	Eclipse, Visual Studio, MonoDevelop	N/A	Eclipse	N/A	No	No	No	Developer workbench	Arduino, Java compatible tools	Arduino, Java compatible tools	N/A	Eclipse
Security	N/A	Https, SSL	SSL over http	SSL encryption	N/A	Secure data servers	N/A	Encryption with TLS and SSL	Keys, oAuth	Write API Keys	Enterprise-grade security	Authentication, SSL
Database Type	N/A	N/A	Relational databases	Relational databases	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Storage Space	Yes, limited	Seemingly unlimited with triple redundant	Yes	Yes, with 3 times replications	Yes	Yes	Yes	Yes, time series archiving	Yes	Yes	Yes	Yes, limited

V. GENERIC CLOUD-BASED SENSOR MONITORING PLATFORM

The details of our proposed generic Cloud-based sensor monitoring platform is presented in this section. We discuss the conceptual architecture and its implementation considerations.

A. *Conceptual Architecture*

The conceptual architecture is designed to address the identified challenges posed by the existing platforms. Ease of use, portability, openness and interoperability are the key factors driving this design. This architecture is further motivated by what we think could be of value to customers with a system of distributed sensors and actuators.

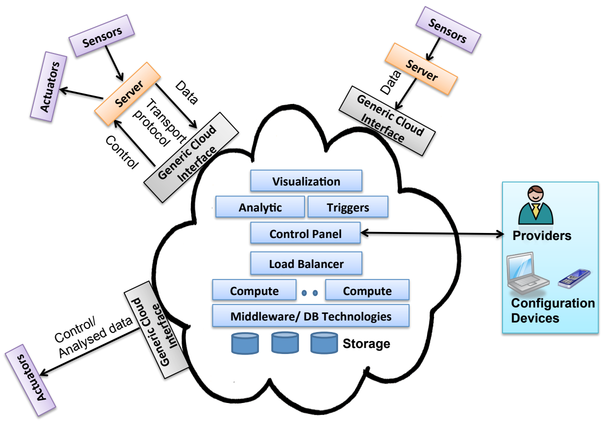

Figure 1. Conceptual Architecture

Figure 1 presents the proposed architecture. As shown on that figure, it is a distributed environment designed to accommodate efficient sensor monitoring, data gathering and sending of control information based on analysed data to actuators. Different usage scenarios are considered in this conceptual architecture such as:

Dual interactions: In this case, a local or public server is attached to both sensors and actuators thereby making it capable of forwarding the captured sensor data and as well receiving the control information to be passed on to the actuators.

Data gathering: This scenario includes situations where the aim is to store and analyse the sensor captured data in the Cloud. This could also be to make the data public for applications to use or to use the analysed data results to derive control decisions for actuators.

Controller actuating: This represents the cases where the stored sensor data are directly being used by applications or to control an actuator.

As shown in Figure 1, the conceptual architecture consist of different components. However, we focus on the key ones due to space constraints.

Database Technologies

The storage of the sensor data on a Cloud platform requires efficient management for inputting and querying the data. This process is supported by appropriate database technologies. In our design, we aim to make this component pluggable so as to support diverse DB types depending on the usage platform. This enables the use of both SQL and NoSQL databases. The relational SQL database types would provide easy compatibility since most of the existing applications, according to our review and experiences, are based on this technology. The NoSQL databases are aimed for quick and less complex scaling of the data storage system.

Generic Cloud Interface
The generic Cloud interface shown in Figure 2 is responsible for mediating between the sensor servers and the Cloud platform. It provides a simple interface and supports numerous formats for time series and alphanumeric data coming from the sensor servers. On the side of the Cloud platforms, it uses platform-neutral data interchange formats to achieve interoperability among heterogeneous Cloud environments.

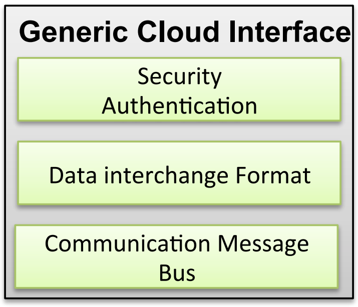

Figure 2. Generic Cloud Interface

As shown in Figure 2 the generic Cloud interface includes three essential components - i) security, ii) data interchange format and iii) communication message bus. These components provide the core functionalities of this interface and therefore deserve more explanations.

Security
This component ensures the privacy of the data interactions and guarantees their authenticity. It provides authentication mechanism to validate the access credentials of the sensor servers to access and transfer data to the Cloud platform. It also enforces data location constraints specified as a policy by the sensor data owners. This policy controls the location of the sensor data storage in the Cloud. It also informs the data owner whenever their data are being copied to some other location.

Data Interchange Format
The format of the sensor data impacts how they can be transported and analysed. This component plays the role of formatting the sensor data into a platform-neutral data interchange format enabling portability, interoperability and efficient transportation. The data interchange formats can be categorised into two groups: (i) self-describing data interchange formats such as XML and JSON and (ii) binary (schema-based) data interchange formats like MessagePack & Protocol Buffers. Based on our previous findings [30], the two format groups have their advantages and drawbacks. The self-describing data interchange format group has the strength of being human readable and easy to understand. But, from the transmission perspective, they contain redundant components, which affects the size of data transfers. The binary data interchange format group is not human readable. But, they are more efficient for transmission according to the performance results in [30]. For this generic interface, we propose a hybrid data interchange format combining the strength of self-describing and binary data interchange formats.

Communication Message Bus
This component provides the communication mechanism for transferring the sensor data to the Cloud platform and also for sending control information to the actuators. It uses the data interchange formats for formatting and serialising the data. The message bus consist of three internal components: (1) Producer, (2) Messaging infrastructure and (3) Consumer

The producer feeds in the data to be transmitted by connecting to the external data producing devices. It integrates the data interchange formats to appropriately prepare the data for transmission. The messaging infrastructure provides the functions of a message broker by asynchronously delivering messages from the producer to the consumers (synchronisation decoupling). The producer does not need to know the nature or location of a consumer. It simply delivers its messages to the broker, which in turn routes them to the appropriate consumer (space decoupling). The broker therefore enables space, time and synchronisation decoupling [31]. This feature facilitates the necessarily loose relationship between a producer and a consumer, which is essential in distributed systems like Clouds. To realise the messaging infrastructure, we base our implementation on the well established Advanced Message Queuing Protocol (AMQP) [32]. The consumer connects the receiving end of the communication. It ensures that the data is appropriately deserialised for the end target platform.

B. Implementation Considerations

For a prototype implementation of this proposed architecture, we suppose the following considerations to be important.

Software artifacts: We envision the use of well established open source software components and standardised technologies as the basis for the implementation. As shown by our survey, there are some promising open source projects that could be considered. This would support the vision for an open and generic solution. Furthermore, it would avoid unnecessary re-implementations.

Component interactions: The generic interface is designed to interact with sensors, actuators and servers mediating between remote sensors and actuators in different fashions. In implementing these interactions, care has to be taken in choosing the data interchange format and communication mechanism since these two factors are very relevant in achieving a wide portability and interoperability of a solution.

Openness: Efforts should be made to make any implemented solution accessible to the general public. This would support quick adoptions and provide demos to the industry to encourage uptake among startup enterprises.

VI. Conclusion and Future Work

This paper presented a survey of the current Cloud-based sensor monitoring and data gathering platforms. It analysed the state-of-the-art and identified open challenges. Both open source and commercial solutions were reviewed to provide a comprehensive survey. We used a table to compile all the features of the analysed platforms, which created a basis for quick comparisons and identification of trends.

According to our survey, the commercial solutions are generally closed source and implement proprietary technologies, which hinders portability and interoperability of their solutions. Furthermore, there is poor support for messaging protocols, energy efficiency and resource constrained environments by the analysed platforms. Also, the platforms use mostly textual data interchange formats, which are not very compact in size for efficient transportation. To address these identified challenges, we proposed a generic Cloud-based sensor monitoring and data gathering platform. The goal is to provide an open solution implementing standardised technologies to promote fast adoptions.

In our next steps, we intend to implement a prototype of our proposed conceptual architecture as a proof-of-concept. We aim to provide it to the general public as a open source software to enable fast testing, deployment and adoption by the market. This contributes to our vision of facilitating interoperable data management in Clouds.

ACKNOWLEDGMENT

This work is supported by the Irish Centre for Cloud Computing and Commerce (IC4), an Irish National Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer Networks, vol. 54, no. 15, 2010, pp. 2787 – 2805.

[2] B. Rao, P. Saluia, N. Sharma, A. Mittal, and S. Sharma, “Cloud computing for internet of things and sensing based applications,” in Sensing Technology (ICST), 2012 Sixth International Conference on, Dec 2012, pp. 374–380.

[3] J.-C. Liu, K.-Y. Chuang, and Y.-L. Wen, “An efficient data gathering system for home medical treatment,” in Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on, Aug 2012, pp. 460–463.

[4] C. Rolim, F. Koch, C. Westphall, J. Werner, A. Fracalossi, and G. Salvador, “A cloud computing solution for patient’s data collection in health care institutions,” in eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED ’10. Second International Conference on, Feb 2010, pp. 95–99.

[5] FreshTemp, “Cloud-based temperature monitoring tool,” https://freshtemp.com/ [retrieved: 23-09-2014].

[6] SensaTrack, “Online sensor monitoring platform,” http://www.sensatrack.com/index.html [retrieved: 23-09-2014].

[7] D. Catrein, M. Henze, K. Wehrle, and R. Hummen, “A cloud design for user-controlled storage and processing of sensor data,” in Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), ser. CLOUDCOM ’12, 2012, pp. 232–240.

[8] Aoki, H. et al., “Cloud architecture for tight interaction with the real world and deep sensor-data aggregation mechanism,” in Software, Telecommunications and Computer Networks (SoftCOM), 2010 International Conference on, September 2010, pp. 280–284.

[9] Piyare, R. et al., “Integrating wireless sensor network into cloud services for real-time data collection,” in ICT Convergence (ICTC), 2013 International Conference on, Oct 2013, pp. 752–756.

[10] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, “A survey on sensor-cloud: Architecture, applications, and approaches,” International Journal of Distributed Sensor Networks, 2013, http://dx.doi.org/10.1155/2013/917923, In-press.

[11] A. Salehi and K. Aberer, “GSN, quick and simple sensor network deployment,” european conference on Wireless Sensor Networks (EWSN), Netherlands, January 2007.

[12] Open Geospatial Consortium, “Sensor observation service,” http://www.ogcnetwork.net/SOS_Intro [retrieved: 23-09-2014].

[13] IBM, “Ibm mote runner for wireless sensor platform,” http://www.zurich.ibm.com/moterunner/ [retrieved: 23-09-2014].

[14] Caracas, A. et al., “Energy-efficiency through micro-managing communication and optimizing sleep,” in Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2011 8th Annual IEEE Communications Society Conference on, June 2011, pp. 55–63.

[15] A. Caracas, T. Kramp, M. Baentsch, M. Oestreicher, T. Eirich, and I. Romanov, “Mote runner: A multi-language virtual machine for small embedded devices,” in Sensor Technologies and Applications, 2009. SENSORCOMM ’09. Third International Conference on, June 2009, pp. 117–125.

[16] MicroStrain, “Microstrain sensorcloud platform,” http://www.sensorcloud.com/ [retrieved: 23-09-2014].

[17] V. K. Sehgal, A. Patrick, and L. Rajpoot, “A comparative study of cyber physical cloud, cloud of sensors and internet of things: Their ideology, similarities and differences,” in Advance Computing Conference (IACC), 2014 IEEE International. IEEE, 2014, pp. 708–716.

[18] Ostia, “Portus Platform,” http://www.ostiasolutions.com/index.php/product/platform-overview [retrieved: 23-09-2014].

[19] TempoDB, “The Time Series Database Service,” https://tempo-db.com/ [retrieved: 15-08-2014].

[20] Zhang, Jia et al., “Sensor data as a service–a federated platform for mobile data-centric service development and sharing,” in Services Computing (SCC), 2013 IEEE International Conference on. IEEE, 2013, pp. 446–453.

[21] TempoIQ, “Sensor Analytics for the Measured World,” https://www.tempoiq.com/?r=1 [retrieved: 23-09-2014].

[22] Bluwird, “BLUWIRED S-CLOUD Platform,” http://bluwired.com/Pages/Discover/BluwiredCloud.aspx [retrieved: 23-09-2014].

[23] J. Hong and M. Baker, “Interaction platforms, energy conservation, behavior change research, and more,” IEEE Pervasive Computing, vol. 12, no. 3, 2013, pp. 10–13.

[24] Yang, Kuo-pao et al., “Park-a-lot: An automated parking management system,” Computer Science and Information Technology, vol. 1, no. 4, 2013, pp. 276–279.

[25] C. Doukas and I. Maglogiannis, “Bringing IoT and cloud computing towards pervasive healthcare,” in 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS). IEEE, 2012, pp. 922–926.

[26] A. A. Chandra, Y. Lee, B. M. Kim, S. Y. Maeng, S. H. Park, and S. R. Lee, “Review on sensor cloud and its integration with arduino based sensor network,” in IT Convergence and Security (ICITCS), 2013 International Conference on. IEEE, 2013, pp. 1–4.

[27] ThingSpeak, “ThingSpeak Platform,” https://thingspeak.com/ [retrieved: 23-09-2014].

[28] Microsoft, “Microsoft azure intelligent system service,” http://www.microsoft.com/windowsembedded/en-us/intelligent-systems-service.aspx [retrieved: 23-09-2014].

[29] M. Prihodko, “Energy consumption in location sharing protocols for android applications,” in The Institute of Technology, Linkoeping University, 2012, Master Thesis.

[30] V. C. Emeakaroha, P. Healy, K. Fatema, and J. P. Morrison, “Analysis of data interchange formats for interoperable and efficient data communication in clouds,” in Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, ser. UCC ’13, 2013, pp. 393–398.

[31] N.-L. Tran, S. Skhiri, and E. Zimanyi, “EQS: An elastic and scalable message queue for the cloud,” in 2011 IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), 2011, pp. 391–398.

[32] S. Vinoski, “Advanced message queuing protocol,” IEEE Internet Computing, vol. 10, no. 6, 2006, pp. 87–89.