2

IoT, Cloud and BigData Integration for IoT Analytics

Abdur Rahim Biswas[1], Corentin Dupont[1] and Congduc Pham[2]

[1]CREATE-NET, Italy
[2]University of Pau, France

2.1 Introduction

Over the last years, the Internet of Things (IoT) has moved from being a futuristic vision to market reality. It is not a question any more whether IoT will be surpassing the hype, it is already there and the race between IoT industry stakeholders has already begun. The IoT revolution comes with trillions of connected devices; however the real value of IoT is in the advanced processing of the collected data. By nature, IoT data is more dynamic, heterogeneous and unstructured than typical business data. It demands more sophisticated, IoT-specific analytics to make it meaningful. The exploitation in the Cloud of data obtained in real time from sensors is therefore very much a necessity. This data processing leads to advanced proactive and intelligent applications and services. The connection of IoT and BigData can offer: i) deep understanding of the context and situation; ii) real-time actionable insight; iii) performance optimization; and iv) proactive and predictive knowledge. Cloud technologies offer decentralized and scalable information processing and analytics, and data management capabilities. This chapter describes a Cloud based IoT and BigData platform, together with their requirements. This includes multiple sensors and devices, BigData analytics, cloud data management, edge-heavy computing, machine learning and virtualization.

In this chapter, Section 2.2 introduces the characteristics of an online Cloud IoT platform. Section 2.3 shows the challenge posed by the huge amount of data to be processed, from the point of view of the quality and quantity of data. It gives an overview of the technologies able to address those challenges.

Section 2.4 presents LoRa, a key enabler for the collection of the data. The chapter includes also initial results of two EU-funded projects on IoT BigData: WAZIUP in Section 2.5; and iKaaS in Section 2.6.

2.2 Cloud-based IoT Platform

According to the NIST definition, Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. The Cloud paradigm can be delivered using essentially three different service models. These are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

A Cloud-based IoT platform is then a dynamic and flexible resource sharing platform delivering IoT services. It offers scalable resources and services management. The exploitation of IoT data depends on massive resources, which should be available when needed and scaled back when not needed.

2.2.1 IaaS, PaaS and SaaS Paradigms

A Cloud based IoT platform needs usually to select one from the three different service models: IaaS, PaaS or SaaS. IaaS allows delivering computer infrastructure on an outsourced basis in order to support enterprise operations. This service model is based on the paradigm of virtualization of resources. The initial success of the Cloud is due to the possibility to embed practically any legacy applications within Virtual Machines (VMs), which are managed by an external stakeholder. This permits to relieve the application owner from managing physical infrastructures. PaaS, on the other hand, provides a platform allowing customers to develop, run, and manage applications. It removes the complexity of building and maintaining the infrastructure typically associated with developing and deploying an application. Typically, a PaaS framework will compile an application from its source code, and then deploy it inside lightweight VMs, or containers. Furthermore, PaaS environments offer an interface to scale up or down applications, or to schedule various tasks within the applications. Finally, SaaS is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted. It is sometimes referred to as “on-demand software”. SaaS is typically accessed by users using a thin client via a web browser.

Cloud-based IoT platforms are usually based on the SaaS paradigm. They provide IoT-related services using a web interface on a pay-per-use basis. For example, a service such as Xively[1] provides a web service with a database able to store sensors data points. This data is then processed and displayed in various graphics.

However, SaaS IoT platforms are limited to the possibility of their web interface. They will not permit the developers to create complex and custom applications. Extensibility mechanisms are sometime offered, allowing extending the web services offered with user-provided callbacks. However the resulting application will not be homogeneous and will be difficult to maintain. Instead, we present in Section 2.5 a concept of IoT Cloud platform based on the PaaS paradigm. Developing an IoT BigData application is a complex task. A lot of services need to be installed and configured, such as databases, message broker and big data processing engines. With the PaaS paradigm, we abstract some of this work. The idea is to let the developer specify the requirements of his application in a specification file called the "manifest". This specification will be read by the PaaS framework and the application will be compiled and instantiated in the Cloud environment, together with its required services.

2.2.2 Requirements of IoT BigData Analytics Platform

An IoT BigData analytic platform should be able to dynamically manage IoT data and provide connectivity with the diverse heterogeneous objects, considering the interoperability issues. It is able to derive useful information and knowledge from large volume of IoT data. The platform shall offer ubiquitous accessibility and connectivity of the diverse objects, services and users, in a mobile context. It shall allow dynamic management and orchestration of users, a huge amount of connected devices as well as massive amount of data produced by those devices. Finally it shall allow personalization of users and services, providing services based on users preference and requirements including real-world context.

Intelligent and Dynamic

The platform should include intelligent and autonomic features in order to dynamically manage the platform functions, components and applications. The platform should also be capable to make proactive decisions, dynamic deployment, and intelligent decisions based on the understanding of the

[1]https://xively.com

context of the environments, users and applications requirements. The platform provides dynamic resources management for IoT, considering performance targets and constraints. This includes offloading workload from clients/hosts to the Cloud and dynamic resources and service migration, as presented in Section 2.6.

Distributed

The platform includes distributed information processing and computing capabilities, distributed storage, distributed intelligence, and distributed data management capabilities. These capabilities should be distributed across smart devices, gateway/server and multiple cloud environments. The processing capability needs to be migrated closer to users, to save bandwidth.

Scalable

The platform needs to be scalable in order to address the needs of a variable number of the devices, services and users. The data management, storage and processing services need to be dimensioned dynamically.

Real-Time

The platform need to be able to process data in real-time, i.e. providing a fast analysis and responses for situations of urgency. A real-time data analysis platform needs to be able to prioritize urgent traffic and processing from non-urgent ones.

Programmable

The platform shall support programmable capabilities of IoT business and service logics, data warehouse scheme, template of data and service model.

Interoperable

The platform provides interoperability between the different IoT services and infrastructure. The APIs need to follow the existing standards. The components are published and maintained as Open Source software. The target is to deliver a common data model able to exploit both structured and unstructured data. In order to create multimodal and cross-domain smart applications, it is necessary to move from raw data to linked data and adopt unambiguous description of relevant information.

Secure

The platform shall include security and privacy by design. This includes different features like data integrity, localization, confidentiality, SLA. Holistic approaches are required to address privacy & security issues across value chains.

2.2.3 Functional Architecture

Our IoT platform solves key problems in IoT analytics, data management and visualization that have traditionally been developed within each application. Developers can easily embed the platform components into their applications saving the time, expertise and expense of building the components themselves. This enables application that would have been too costly and time-consuming to develop. The platform integrates easily with existing sensors, network infrastructure and end-user applications.

Figure 2.1 displays the functional overview of the BigData IoT platform. The topmost block represents the Cloud platform, the middle one is the network connectivity while the bottom one is the local deployment, including gateway and sensors. The following functional domains have been identified:

- The “Smart Applications” domain is the IoT application itself.
- The “Users Management” allows the management of the identification, roles and connections of users.
- The “Interoperable Service and Dynamic Workflow” domain allows application writing, deploying, hosting and execution.
- The “Processing and Analytic Engine”, provides services of stream processing and data analytics.
- The “Network communication” domain provides the IoT connectivity.
- The “Embedded software” and Hardware domains represent the IoT gateway and sensors themselves.

2.3 Data Analytics for the IoT

The amount of IoT data coming from real-world smart objects with sensing, actuating, computing and communication capabilities is exploding. The sensors and devices are more and more deployed, within more applications and across industries. This section first explores the characteristics of this data. It then presents several data analytics techniques able process this data.

2.3.1 Characteristics of IoT Generated Data

The volume and quality of the data generated by IoT devices is very different from the traditional transaction-oriented business data. Coming from millions of sensors and sensor-enabled devices, IoT data is more dynamic,

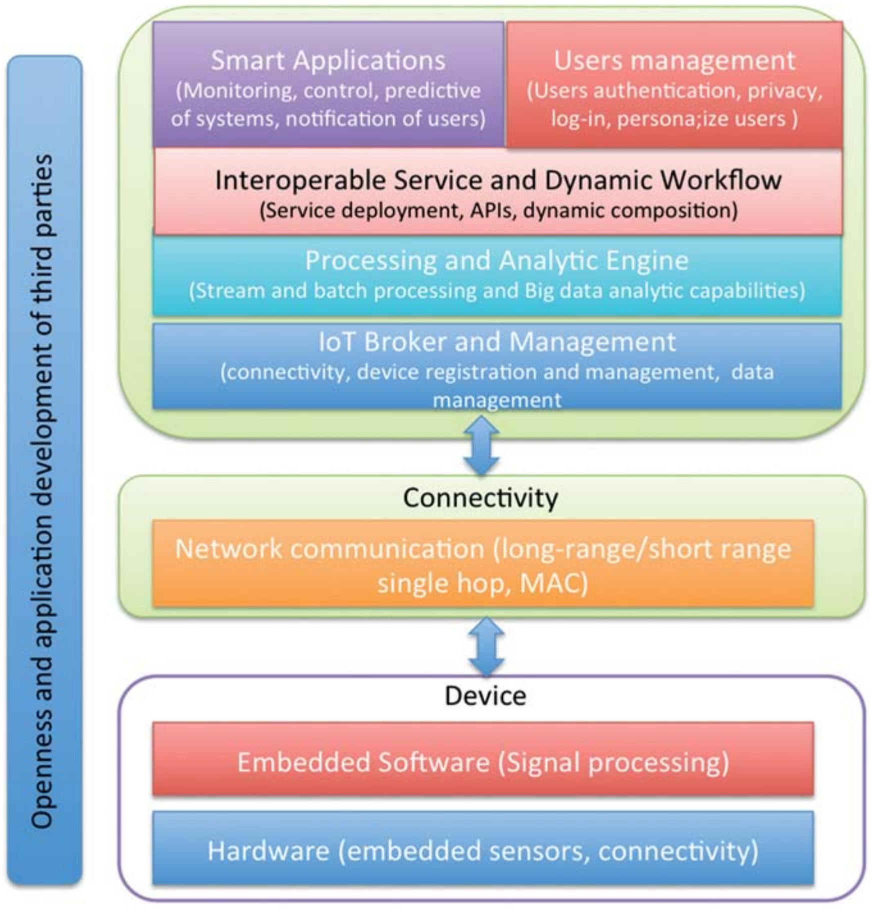
Openness and application development of third parties
Smart Applications
(Monitoring, control, predictive of systems, notification of users)
Users management
(Users authentication, privacy, log-in, persona;ize users)
Interoperable Service and Dynamic Workflow
(Service deployment, APIs, dynamic composition)
Processing and Analytic Engine
(Stream and batch processing and Big data analytic capabilities)
IoT Broker and Management
(connectivity, device registration and management, data management
Connectivity
Network communication (long-range/short range single hop, MAC)
Device
Embedded Software (Signal processing)
Hardware (embedded sensors, connectivity)

Figure 2.1 Functional Architecture of IoT and Bigdata platform.

heterogeneous, imperfect, unprocessed, unstructured and real-time than typical business data. It demands more sophisticated, IoT-specific analytics to make it meaningful.

As illustrated in Figure 2.2, the BigData is defined by 4 “Vs”, which are Volume, Velocity, Variety and Veracity. The first V is for a large volume of data, not gigabytes but rather thousands of terabytes. The second V is referencing data streams and real-time processing. The third V is referencing the heterogeneity of the data: structure and unstructured, diverse data models, query language, and data sources. The fourth V is defining the data uncertainty,

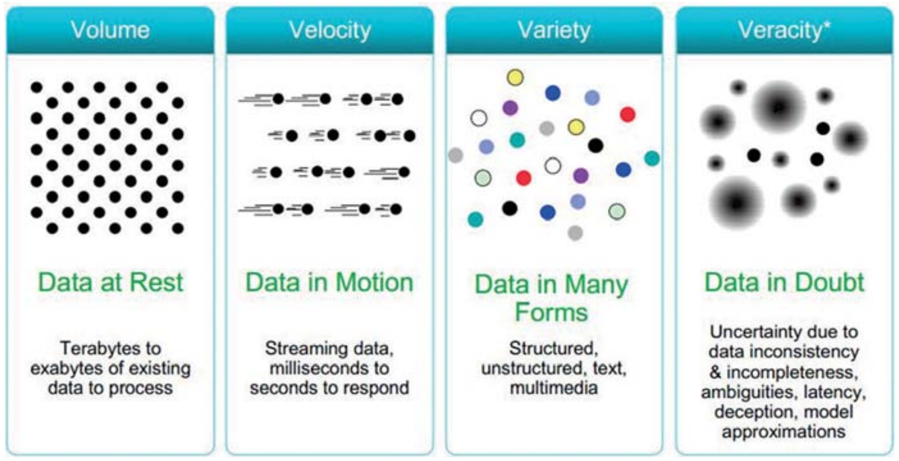
Volume
Velocity
Variety
Veracity*
Data at Rest
Terabytes to exabytes of existing data to process
Data in Motion
Streaming data, milliseconds to seconds to respond
Data in Many Forms
Structured, unstructured, text, multimedia
Data in Doubt
Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

Figure 2.2 BigData properties [4].

which can be due to data inconsistency, incompleteness, ambiguities, latency and lack of precise model.

The IoT faces all 4 Vs of the BigData challenges. However the velocity is the main challenge: we need to process in real-time the data coming from IoT devices. For example, medical wearable such as Electro Cardio Graphic sensors produce up to 1000 events per second, which is a challenge for real-time processing. The volume of data is another important challenge. For example General Electric gathers each day 50 million pieces of data from 10 million sensors. A wearable sensor produces about 55 million data points per day. In addition, IoT also faces verity and veracity BigData challenges.

2.3.2 Data Analytic Techniques and Technologies

A cloud-based IoT analytics platform provides IoT-specific analytics that reduce the time, cost and required expertise to develop analytics-rich, vertical IoT applications. Platform's IoT-specific analytics uncover insights, create new information, monitor complex environments, make accurate predictions, and optimize business processes and operations. The applications of the IoT BigData Platform can be classified into four main categories i) deep understanding and insight knowledge ii) Real time actionable insight iii) Performance optimization and iv) proactive and predictive applications.

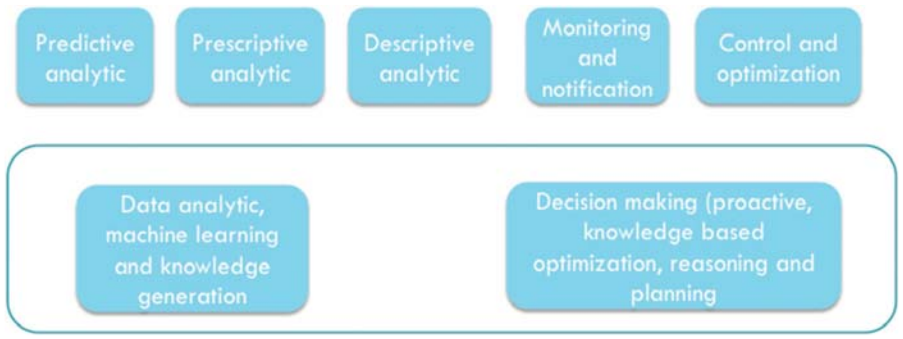
Predictive analytic
Prescriptive analytic
Descriptive analytic
Monitoring and notification
Control and optimization
Data analytic, machine learning and knowledge generation
Decision making (proactive, knowledge based optimization, reasoning and planning

Figure 2.3 IoT BigData applications.

In the following we provide various technologies allowing building such an IoT analytics platform.

Batch Processing

Batch processing supposes that the data to be treated is present in a database. The most widely used tool for the case is **Hadoop MapReduce**. MapReduce is a programming model and Hadoop an implementation, allowing processing large data sets with a parallel, distributed algorithm on a cluster. It can run on inexpensive hardware, lowering the cost of a computing cluster. The latest version of MapReduce is YARN, called also MapReduce 2.0. **Pig** provides a higher level of programming, on top of MapReduce. It has its own language, PigLatin, similar to SQL. Pig Engine parses, optimizes and automatically executes PigLatin scripts as a series of MapReduce jobs on a Hadoop cluster. Apache **Spark** is a fast and general-purpose cluster computing system. It provides high-level APIs in Java, Scala, Python and R, and an optimized engine that supports general execution graphs. It can be up to a hundred times faster than MapReduce with its capacity to work in-memory, allowing keeping large working datasets in memory between jobs, reducing considerably the latency. It supports batch and stream processing.

Stream Processing

Stream processing is a computer programming paradigm, equivalent to dataflow programming and reactive programming, which allows some applications to more easily exploit a limited form of parallel processing. **Flink** is a streaming dataflow engine that provides data distribution, communication and fault tolerance. It has almost no latency as the data are streamed in real-time (row by row). It runs on YARN and works with its own extended version of MapReduce.

Machine Learning

Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. It is especially useful in the context of IoT when some properties of the data collected need to be discovered automatically. Apache Spark comes with its own machine learning library, called **MLib**. It consists of common learning algorithms and utilities, including classification, regression, clustering, collaborative filtering, dimensionality reduction. Algorithms can be grouped in 3 domains of actions: Classification, association and clustering. To choose an algorithm, different parameters must be considered: scalability, robustness, transparency and proportionality. **KNIME** is an analytic platform that allows the user to process the data in a user-friendly graphical interface. It allows training of models and evaluation of different machine learning algorithms rapidly. If the workflow is already deployed on Hadoop, **Mahout,** a machine learning library can be used. Spark also has his own machine learning library called MLib.

H20 is a software dedicated to machine-learning, which can be deployed on Hadoop and Spark. It has an easy to use Web interface, which makes possible to combine BigData analytics easily with machine learning algorithm to train models.

Data Visualisation

Freeboard offers simple dashboards, which are readily useable sets of widgets able to display data. There is a direct Orion Fiware connector. Freeboard offers a REST API allowing controlling of the displays. **Tableau Public** is a free service that lets anyone publish interactive data to the web. Once on the web, anyone can interact with the data, download it, or create their own visualizations of it. No programming skills are required. Tableau allows the upload of analysed data from .csv format, for instance. The visualisation tool is very powerful and allows a deep exploration the data. **Kibana** is an open source analytics and visualization platform designed to work with Elasticsearch. Kibana allows searching, viewing, and interacting with data stored in Elasticsearch indices. It can perform advanced data analysis and visualize data in a variety of charts, tables, and maps. **Elasticsearch** is a highly scalable open-source full-text search and analytics engine. It allows to store, search, and analyze big volumes of data quickly and in near real time.

It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. It is really designed for real-time analytics, most commonly used with Flink or Spark streaming.

2.4 Data Collection Using Low-power, Long-range Radios

Regarding the deployment of IoT devices in a large scale, it is still held back by technical challenges such as short communication distances. Using the traditional mobile telecommunications infrastructure is still very expensive (e.g., GSM/GPRS, 3G/4G) and not energy efficient for autonomous devices that must run on battery for months. During the last decade, low-power but short-range radio such as IEEE 802.15.4 radio have been considered by the WSN community with multi-hop routing to overcome the limited transmission range. While such short-range communications can eventually be realized on smart cities infrastructures where high node density with powering facility can be achieved, it can hardly be generalized for the large majority of surveillance applications that need to be deployed in isolated or rural environments. Future 5G/LTE standards do have the IoT orientation but these technologies and standards are not ready yet while the demand is already high.

Recent so-called Low-Power Wide Area Networks (LPWAN) such as those based on Sigfox^{TM} or Semtech's LoRa^{TM} [1] technology definitely provide a better connectivity answer for IoT as several kilometers can be achieved without relay nodes to reach a central gateway or base station. Most of long-range technologies can achieve 20 km or higher range in LOS condition and about 2 km in urban NLOS [2]. With cost and network availability constraints, LoRa technology, which can be privately deployed in a given area without any operator, has a clear advantage over Sigfox which coverage is entirely operator-managed. These low-power, long-range radio technologies will definitely allow a huge amount of sensors to be installed in remote area, thus augmenting the amount of data to be treated in the IoT Cloud platform.

2.4.1 Architecture and Deployment

The deployment of LPWAN (both operator-based and privately-owned scenarios) is centred on gateways that usually have Internet connectivity as shown in Figure 2.4. Although direct communications between devices are possible, most of IoT applications follow the gateway-centric approach with mainly

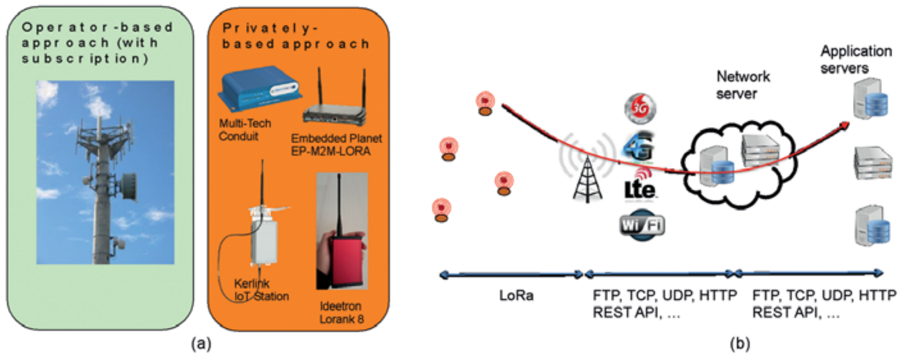
Operator-based approach (with subscription)
Privately-based approach
Multi-Tech Conduit
Embedded Planet EP-M2M-LORA
Kerlink IoT Station
Ideetron Lorank 8
(a)
Network server
Application servers
LoRa
FTP, TCP, UDP, HTTP REST API, ...
FTP, TCP, UDP, HTTP REST API, ...
(b)

Figure 2.4 Gateway-centric deployment.

uplink traffic patterns. In this typical architecture data captured by end-devices are sent to a gateway which will push data to well identified network servers. Then application servers managed by end-users could retrieve data from the network server. If encryption is used for confidentiality, the application server can be the place where data could be decrypted and presented to end-users.

The LoRa Alliance has issued the LoRaWAN specification [3] in a tentative for standardization of public, large-scale LoRa LPWAN infrastructures featuring multi-gateways and full network/application servers' architecture as previously depicted in Figure 2.4. This specification also defines the set of common channels for communications, the packet format, Medium Access Control (MAC) commands that must be provided and 3 end-devices classes depending on communication requirements. This architecture can however be greatly simplified for small, ad-hoc deployment scenarios where the gateway can directly push data to some servers or IoT-specific cloud platforms if properly configured.

2.4.2 Low-cost LoRa Implementation

The implementation of the full LoRaWAN specification requires gateways to be able to listen on several channels and LoRa settings simultaneously. Commercial gateways therefore use advanced concentrators chips capable of scanning up to 8 different channels: the SX1301 concentrator is typically used instead of the SX127x chip which is designed for end-devices. Commercial gateways cost several hundredth euros with the cost of the SX1301-capable board alone to be more than a hundred euro.

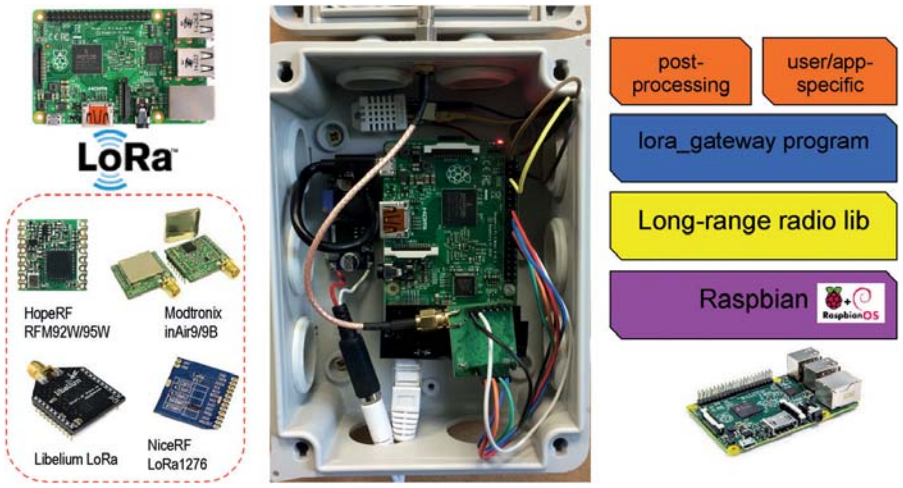

Figure 2.5 Low cost gateway from off-the-sheves components.

For many adhoc applications, it is however more important to keep the cost of the gateway low and to target small to medium size deployment scenario for various specific use cases instead of the large-scale, multi-purpose deployment scenarios defined by LoRaWAN. Note that even though several gateways can be deployed to serve several channel settings if needed. In many cases, this solution presents the advantage of being more optimal in terms of cost as incremental deployment can be realized and also offer a higher level of redundancy that can be an important requirement in developing countries for instance.

Our LoRa gateway could be qualified as "single connection" as it is built around an SX1272/76, much like an end-device would be. The cost argument, along with the statement that too integrated components are difficult to repair and/or replace in the context of developing countries, also made the "off-the-shelves" design orientation an obvious choice. Our low-cost gateway is based on a Raspberry PI (1B/1B+/2B/3B) which is both a low-cost (less than 30 euro) and a reliable embedded Linux platform. Our long-range communication library supports a large number of LoRa radio modules (most of SPI-based radio modules). The total cost of the gateway can be as low as 45 euro.

Together with the "off-the-shelves" component approach, the software stack is completely open-source: (a) the Raspberry runs a regular Raspian distribution; (b) our long range communication library is based on the SX1272 library written initially by Libelium and (c) the lora_gateway program is kept as simple as possible. We improved the original SX1272 library in various ways to provide enhanced radio channel access (CSMA-like with SIFS/DIFS) and support for both SX1272 and SX1276 chips. We believe the whole architecture and software stack are both robust and simple for either "out-of-the-box" utilization or quick customization by third parties.

We tested the gateway in various conditions for several months with a DHT22 sensor to monitor the temperature and humidity level inside the case. Our tests show that the low-cost gateway can be deployed in outdoor conditions with the appropriate casing. Although the gateway should be powered, its consumption is about 350 mA for an RPIv3B with both WiFi and Bluetooth activated.

2.5 WAZIUP Software Platform

The WAZIUP project, namely the Open Innovation Platform for IoT-BigData in Sub-Saharan Africa is a collaborative research project using cutting edge technology applying IoT and BigData to improve the working conditions in the rural ecosystem of Sub-Saharan Africa. First, WAZIUP operates by involving farmers and breeders in order to define the platform specifications in focused validation cases. Second, while tackling challenges which are specific to the rural ecosystem, it also engages the flourishing ICT ecosystem in those countries by fostering new tools and good practices, entrepreneurship and start-ups. Aimed at boosting the ICT sector, WAZIUP proposes solutions aiming at long term sustainability.

The consortium of WAZIUP involves 7 partners from 4 African countries and partners from 5 EU countries combining business developers, technology experts and local Africa companies operating in agriculture and ICT. The project involves also regional hubs with the aim to promote the results to the widest base in the region.

2.5.1 Main Challenges

The WAZIUP Cloud platform needs to face a number of challenges. Those challenges are related to the specific environment in which the platform will be deployed, and the need of its end users. First of all, we identified that farmers in Sub-Saharan Africa are lacking data on culture status. For instance, parameters such as potassium and nitrogen levels are very useful for precision farming. Secondly, farmers are lacking actionable information on the condition of the farm. This actionable information can be displayed in the form of alerts, forecasts and recommendations. An example of such a service is a recommendation on the water levels needed for irrigation, taking into accounts the weather forecasts. On a larger scale, governments and institutions are lacking information and statistics on their territory. An example is geographical statistics on the spreading of a disease in a country.

On a more technical level, we noticed that most rural African users have mobile phones, but not always smart phones. Furthermore, 3G is not always present in rural areas. Internet and grid connection can also be intermittent. Lastly, a huge challenge that the WAZIUP platform should address is the cost of IoT devices, application development and application hosting.

2.5.2 PaaS for IoT

As introduced before, PaaS framework will compile an application from its source code, and then deploy it inside lightweight virtual machines, or containers. This compilation and deployment is done with the help of a file called the manifest, which allows the developer to describe the configuration and resource needs for his application. The manifest file will also describe the services that the application requires and that the platform will need to provision.

The idea of WAZIUP is to extend the paradigm of the PaaS to IoT. Indeed, developing an IoT BigData application is a complex task. A lot of services need to be installed and configured, such as databases and complex event processing engines. Furthermore, it requires an advanced knowledge and skills in programing of embedded devices, of data stream processors, of advanced data analytics, and finally of GUIs and user interactions. We propose to abstract those skills using the PaaS paradigm.

Figure 2.6 shows the PaaS deployment in WAZIUP. Traditional PaaS environment are usually installed on top of IaaS (in blue in the picture). The blue boxes are physical servers, respectively the Cloud Controller and one Compute node. The PaaS environment is then installed inside the IaaS V Ms, in green in the picture. We use Cloud Foundry as a PaaS framework. It comes with a certain number of build packs, which and programming languages compilers and run time environments. It also provides a certain number of preinstalled services such as MongoBD or Apache Tomcat. The manifest file, showed on the right hand side, provide a high-level language that allows describing which services to instantiate. We propose to extend this language to IoT and BigData services:

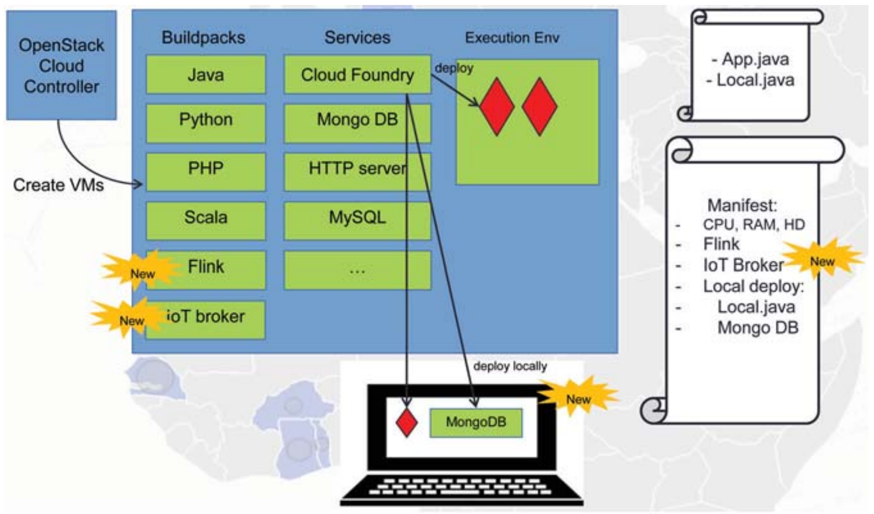

Figure 2.6 PaaS deployment extended for IoT in WAZIUP.

- Data stream and message broker
- CEP engines
- Batch processing engines
- Data visualization engines

Furthermore, we propose to include in the manifest a description of the IoT sensors that are required by the application. This query includes data such as the sensor type, location and owner. The manifest also includes the configuration of the sensors. The application will then be deployed both in the global Cloud and in the local Cloud.

2.5.3 Architecture

Figure 2.7 presents the full WAZIUP architecture. There are 4 silos (from left to right): Application development, BigData platform, IoT platform, Sensors and data sources. The first silo involves the development of the application itself. A rapid application development (RAD) tool can be used, such as Node-Red. The user provides the code source of the application, together with the manifest. As a reminder, the manifest describes the requirements of the application in terms of:

- Computation needs (i.e. RAM, CPU, disk).
- Reference to data sources (i.e. sensors, internet – sources . . .).
- BigData engines needed (i.e. Flink, Hadoop . . .).

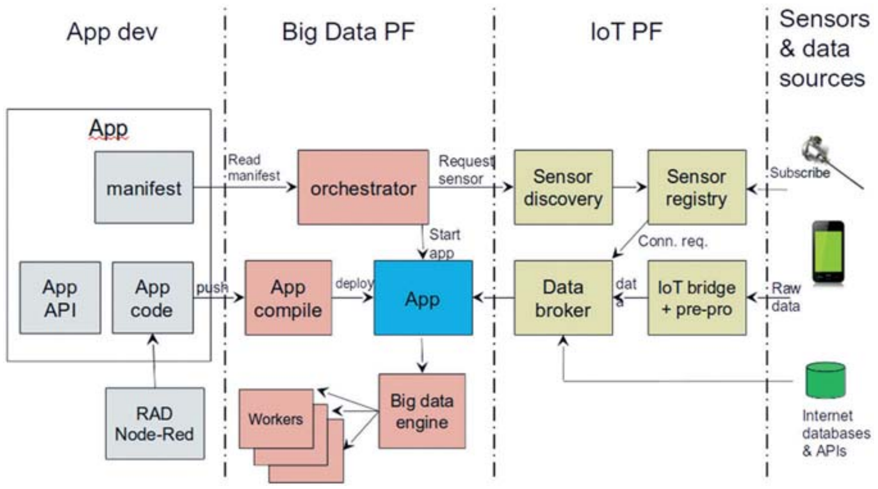

Figure 2.7 WAZIUP architecture.

- Configuration of sensors (i.e. sampling rate).
- Local and global application deployment.

The application source code, together with the manifest, is pushed to the WAZIUP Cloud platform by the user. The orchestrator component will read the manifest and trigger the compilation of the application. It will then deploy the application in the Cloud execution environment. It will also instantiate the services needed by the application, as described in the manifest. The last task of the orchestrator is to request the sensor and data sources connections from the IoT components of the architecture. The sensor discovery module will be in charge of retrieving a list of sensors that matches the manifest description.

On the left side of the diagram, the sensor owners can register their sensors with the platform. External data sources such as Internet APIs can also be connected directly to the data broker. The sensors selected for each application will deliver their data to the data broker, through the IoT bridge and preprocessor. This last component is in charge of managing the connection and configuration of the sensors. Furthermore, it will contain the routines for pre-processing the data transmitted, such as cleaning, extrapolating, aggregating and averaging data points.

2.5.4 Deployment

WAZIUP will be deployed and accessed in an African context, where internet access is sometime scarce. WAZIUP therefore has a very strong constraint regarding low internet connectivity. To fulfil this requirement, we propose a Cloud structure in two parts: the *global Cloud* and the *local Cloud*. The global Cloud corresponds to the Cloud in the traditional sense. The local Cloud corresponds to the gateway and an optional connected computer. The idea of WAZIUP is to extend the PaaS concept to the local Cloud.

A typical WAZIUP deployment is illustrated in Figure 2.8. On the left hand side of the picture, the application is designed by the developer, together with the manifest file. It is pushed on the WAZIUP Cloud platform. The orchestrator then takes care of compiling and deploying the application in the various Cloud execution environments. Furthermore, the orchestrator drives the instantiation of the services in the Cloud, according to the manifest. The manifest is also describing which part of the application need to be installed locally, together with corresponding services. The local application can then connect to the gateway and collect data from the sensors.

2.6 iKaaS Software Platform

The iKaaS platform combines ubiquitous and heterogeneous sensing, BigData and cloud computing technologies in a platform enabling the Internet of Things process consisting of continuous iterations on data ingestion, data storage, analytics, knowledge generation and knowledge sharing phases, as foundation service provision.

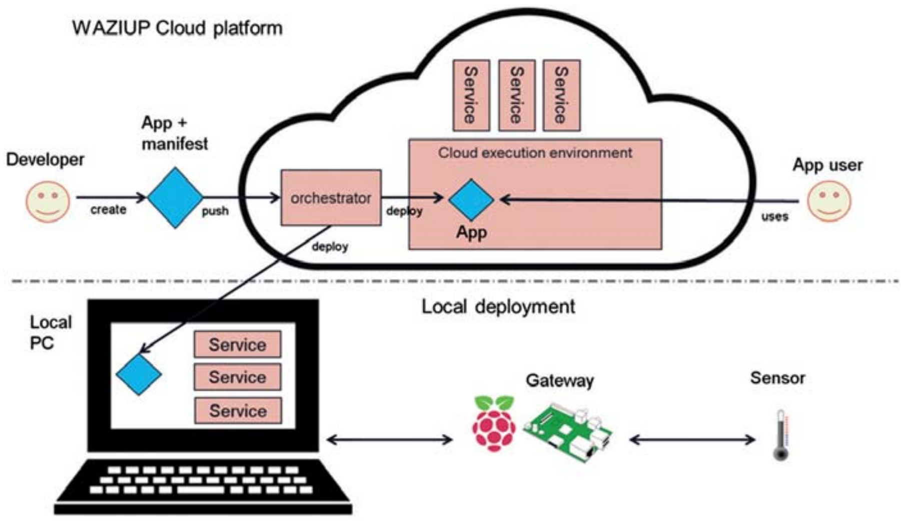

Figure 2.8 WAZIUP local and global deployment.

The iKaaS platform consists of two distinct Cloud ecosystems: the *Local Cloud* and the *Global Cloud*. More specifically:

- A Local Cloud provides requested services to users in a limited geographical area. It offers additional processing and storage capability to services. It is created on-demand, and comprises appropriate computing, storage and networking capabilities.
- The Global Cloud is seen in the "traditional" sense, as a construct with on-demand and elastic processing power and storage capability. It is a "backbone infrastructure", which increases the business opportunities for service providers, the ubiquity, reliability and scalability of service provision.

Local Clouds can involve an arbitrarily large number of nodes (sensors, actuators, smartphones, etc.). The aggregation of resources comprises sufficient processing power and storage space. The goal is to serve users of a certain area. In this respect, a Local Cloud is a virtualised processing, storage and networking environment, which comprises IoT devices in the vicinity of the users. Users will exploit the various services composed of the Local Cloud's devices' capabilities. For example, a sensor and its gateway equipped with the iKaaS platform.

The Global Cloud allows IoT service providers to exploit larger scale services without owning actual IoT infrastructure.

The iKaaS Cloud ecosystem will encompass the following essential functionality:

- Consolidated service-logic, resource descriptions and registries will be parts of the Global Cloud. These will enable the reuse of services. Practically, a set of registries will be developed and pooling of service logic and resources will be enabled.
- Autonomic service management will be part, firstly, of the Global Cloud, and, then, in the Local Clouds. This functionality will be in charge of (i) dynamically understanding the requirements, decomposing the service (finding the components that are needed); (ii) finding the best service configuration and migration (service component deployment) pattern; (iii) during the service execution, reconfiguring the service, i.e., conducting dynamic additions, cessations, substitutions of components.
- Distributed data storage and processing is anticipated for the structure of global and local clouds. This means capabilities for efficiently

communicating, processing and storing massive amounts of, quickly-emerging, versatile data (i.e., "BigData"), produced by a huge number of diverse IoT devices. Another important capability will be the derivation of information and knowledge (e.g., on device behaviour, service provision, user aspects, etc.), while ensuring security and privacy, which are top concerns.

- Knowledge as a service (KaaS) will be primarily part of the Global Cloud. This area will cover: (i) device behaviour aspects; (ii) the way services have been provided (e.g., through which IoT resources) and the respective quality levels; (iii) user preferences.

As can be seen the iKaaS functionality will determine the optimal way to offer a service. For instance service components may need to be migrated as close as possible to the required (IoT) data sources. IoT services may need generic service support functionality that is offered within the Cloud, and, at the same time, they do rely on local information (e.g., streams of data collected by sensors in a given geographic area), therefore, the migration of components close to the data sources will help in the reduction of the data traffic.

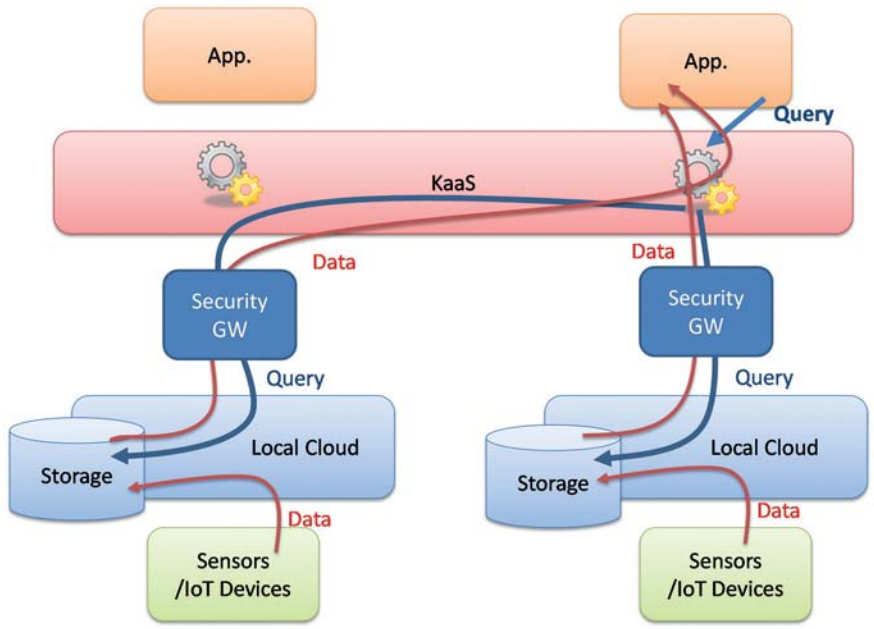

Figure 2.9 iKaaS platform.

2.6.1 Service Orchestration and Resources Provisioning

The platform offers mechanisms that autonomously analyse application requirements, user preferences and Cloud resources and accordingly decide upon the most appropriate deployment of services. The most appropriate deployment must achieve the best balance between system performance, quality of service and cost. In this context, services may be decomposed into smaller components, based on the current situation and information on data sources, in order to be migrated and executed in a "Local Cloud", near the data sources, following the Hadoop maxim that "*Moving Computation is Cheaper than Moving Data*". Alternatively, services may be deployed and executed in the Global Cloud. Furthermore, this mechanism will facilitate the notion of "Everything as a Service", and attached gateway to host and process services on-demand, by means of service migration instead of being limited to predefined services. The local IoT Gateway may act as part of a "Local Cloud" on an on-demand basis in coordination with the Global Cloud, provided that the Local Cloud has sufficient resources to process and execute the service.

The platform uses a model that allows the service to be analysed and decomposed into a certain number of sub-components according to application requirements, user preferences including privacy constraints, policies, system state and data sources location. The service sub-components are then migrated to either Local Clouds, to be computed near the data sources (e.g., sensors) or into the Global Cloud, to take advantage of the extensive computing power and storage available. The optimal distribution is decided with the aim of achieving the best balance between overall system performance (network traffic, computing load), quality of services (prompt and accurate delivery of service result) and service costs.

2.6.2 Advanced Data Processing and Analytics

Information stream processing algorithms and mechanisms offer on-the-fly analysis of volatile data coming from the distributed sensing infrastructure. In addition, the platform includes off-line BigData analytics over persistent data capable of uncovering hidden patterns and unknown correlations. This will allow feeding with contents the envisaged knowledge service platform.

The iKaaS platform includes the analysis of the requirements and challenges posed by those information stream processing and knowledge acquisition scenarios to the provision of a set of IoT and BigData services over cloud and network infrastructures.

The provision of those services comprises processing capabilities, covering the knowledge acquisition lifecycle. This lifecycle goes from the aggregation of heterogeneous data, through information stream processing services and visualization services, to the derivation of knowledge and experience. Special attention will be paid to the consolidation of existing approaches and to the design of complementary solutions able to address the technological challenges:

- Information Stream Processing, information extraction and visualization, mechanisms enabling the usage of smart virtual objects as a multi-cloud cloud based resource.
- Distributed and scalable storage mechanisms for smart virtual objects that supports service decomposition, migration and corresponding resource allocation aspects within the iKaaS local and Global Cloud environments.
- Analytics engines and mechanism for assessment and processing of data over a large number of smart objects. The objective is to derive reliable information and to provide knowledge that can be provided as a service to facilitate situation aware applications.

Given that processing and storage may take place in either the Global or the Local Cloud or both, in support of real-time autonomic and flexible service execution, the mechanisms defined in this task shall support flexible and fast discovery of smart virtual objects and allocation of data sources so that efficient and cost-effective service and resource migration can be realized.

Hence, the platform offers the mechanisms and techniques for handling smart objects and processing of their data to satisfy real-time service execution requirements in Cloud environments and also to derive useful "contextual" information and knowledge to serve cost-effective, low latency resources migration and allocation needs.

The scalable and distributed storage mechanism for smart virtual objects and aggregated and anonymised data will also need to be managed dynamically in order to deal with the large number of data sources.

2.6.3 Service Composition and Decomposition

Principle of Service Composition and Decomposition

IoT and BigData applications are complex large-scale applications, including a combination of multiple sources, functionalities and composed by many small functional services across multiple sectors/domains. For example, an active and healthy living of ageing people application includes many small services like monitoring the blood pressure, monitoring the heart rate, weight monitoring, location awareness, smart lighting, utility metering, notification and reminders, etc., across health, well-being, security and home automation domains. Additionally, for IoT and BigData in a given application as the service is evolving, more and more services added to the applications/systems. Therefore, it is important to design the iKaaS services as small and autonomous as possible, with well-defined APIs to operate them individually.

iKaaS functional decomposition of an application/complex service (as defined in the previous sub-section) allows to achieve loose coupling and high cohesion of multiple services. Alternatively multiple simple services can be composed into complex services for the purposes of various applications. In Figure 2.10, the basic logic of service decomposition and composition are shown.

Functional decomposition of services gives the agility, flexibility, scalability of individual services to operate autonomously. Each of the simple services is running in its own process and communicating with lightweight mechanisms. The overall high-level service logics (e.g. software module) are decomposed to multiple service logics or software modules which can be delivered as independent runtime services. These services are built around business capabilities and are independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management

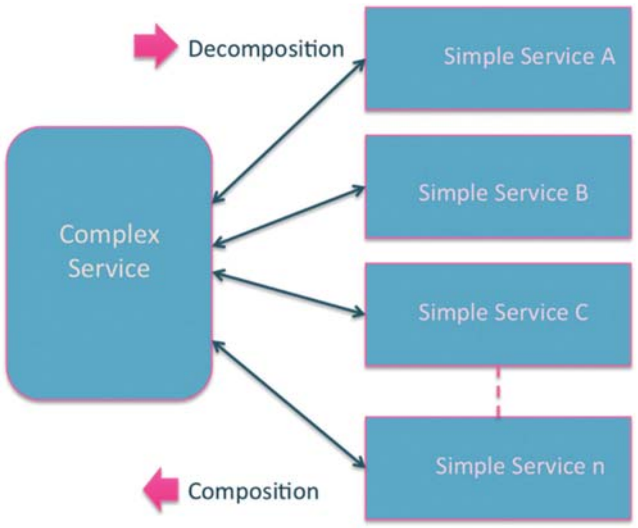

Figure 2.10 Service composition and decomposition.

of these services, which may be written in different programming languages and use different data storage technologies.

Pattern of Service Composition

The iKaaS service design pattern significantly impacts how the services will be composed and decomposed. One of the main concepts is to design the services as independent as possible. In the design pattern the service replication and reliability should also be considered. One individual complex service can be composed by multiple isolated end-users or system level services. The relationship between the services and datasets, whether each of the services is using its own dataset or sharing a single dataset with other services can vary. However each iKaaS simple service is associated with a relevant dataset in order to make the service fully independently designed and deployable.

At runtime, one iKaaS service may consist of multiple service instances. Each service instance is a runtime (e.g., Docker container). In order to be highly available, the containers are running on multiple Cloud VMs. In this case, the Service Manager acts as a load balancer that distributes requests across the service instances.

2.6.4 Migration and Portability in Multi-cloud Environment

Service migration is a concept used in cloud computing implementation models that ensures that an individual or organization can easily shift services

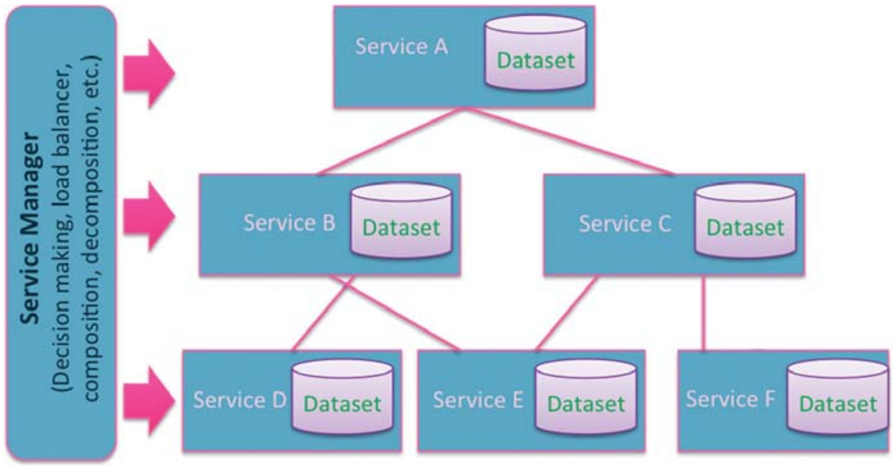

Figure 2.11 Patten for composition and decomposition.

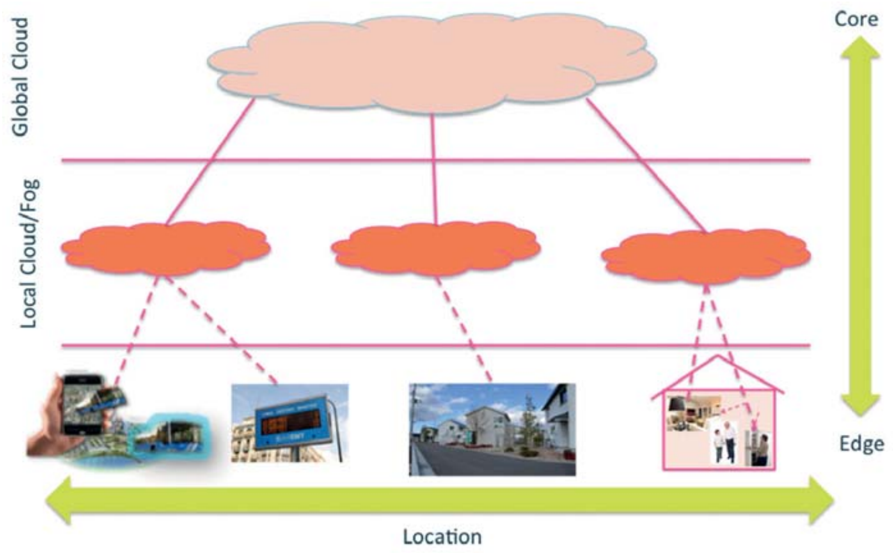

Figure 2.12 iKaaS distributed local and global cloud with service migration.

between different cloud vendors without encountering implementation, integration, and compatibility and interoperability issues. The concept is defined the process and framework by which these applications can be deployed on another cloud vendor or supported private cloud architecture.

iKaaS runtime puts services and all of its dependencies into a container which is portable among different platforms, desktop distributions and clouds. One can build, move and run distributed services with containers. By automating deployment inside containers, developers and system administrators can run the same application on laptops, virtual machines, bare-metal servers, and the cloud.

The concept of service migration is applicable in multi-cloud and distributed computing environment, where the processing capabilities are moved to near the data sources or a simple service is run near data sources. iKaaS is a fully distributed architecture, in which the overall platform functionalities and capabilities are distributed between local and global could. The concept of local could can be seen as an edge or fog computing, where pre-processing are done at the origin level.

Fog computing and computing near the data source provide a promising new approach to significantly reduce network operation cost by moving the computation or early pre-processing close to the data sources. A key challenge in such systems is to decide where and when services should be migrated with respect to users mobility, overall situation and environment context.

Edge/local computing can provide elastic resources to large scale data process system without suffering from the drawback of cloud, high latency. In cloud computing paradigm, event or data will be transmitted to the data centre inside core network and result will be sent back to end user after a series of processing. A federation of fog and cloud can handle the BigData acquisition, aggregation and pre-processing, reducing the data transportation and storage, balancing computation power on data processing. For example, in a large-scale environment monitoring system, local and regional data can be aggregated and mined at fog nodes providing timely feedback especially for emergency case such as toxic pollution alert. Detailed and thorough analysis as computational-intensive tasks can be scheduled in the cloud side.

2.6.5 Cost Function of Service Migration

One of the main challenges for the services migration is to define the strategy for the service migration. There is a tradeoff between the service migration cost and the transmission cost (such as communication delay and network overhead) between the user and the cloud. It is challenging to find the optimal decision also because of the uncertainty in user mobility as well as possible non-linearity of the migration and transmission costs. The service migration offers the benefits of reduction in networks overhead and latency over changing the location of the users. It is often challenging to make the optimal decision in an optimal manager, which can optimize the cost functions based on the situation and user's preferences.

iKaaS will propose a framework for dynamic, cost-minimizing migration of distribution services into a hybrid cloud infrastructure that spans geographically distributed data centers. We will propose an algorithm which optimally places services in different clouds to minimize overall operational cost over time, subject to service response time constraints. The framework will be designed based on the Markov-Decision-Process (MDP), to study service migration in iKaaS Cloud environment.

2.6.6 Dynamic Selection of Devices in Multi-cloud Environment

The end-devices are expected to play a key role in iKaaS not only for they data they can provide for the optimization of the iKaaS provided services, but also for the data they can provide with respect to end-device suitability and social relationships.

That is because in the case of end-device suitability, end-devices can be viewed as the end-points of an end-to-end service chain over a multi-cloud infrastructure. As such, this device suitability identification can be seen as "fixing" the end-points of the service provisioning chains, allowing as such (once the end-points are fixed) to then "fix" the location/placement of service provisioning functions in a way that optimizes both the service and the overall cloud performance. For example, if a device is identified as suitable, which is attached at a certain local cloud, then it would make sense for the other service functionalities needed to be instantiated at that local cloud so as to be close to the data source.

Some of the key factors that can be taken under consideration when defining device suitability are:

- Location/mobility pattern of a device; so as to define where the owner of the device is or is predicted to be so as to appropriately instantiate service functions close to the corresponding locations.
- Battery levels and evolution of battery levels; so as to be able to deduce whether a device can be relied upon for providing/receiving data, therefore corresponding service functions will need to be appropriately instantiated.
- Availability of sensors; how often a user has its device sensors exposed and is ready to be a potential match for inclusion in a service delivery chain.
- User away and reaction times; to make sure the user carries the device with them and is able to see an alert on time and react to it.
- Data quality: The quality of users inputs, without false-positives or misleading measurements.

All these factors will be further and better thought of and appropriate knowledge building mechanisms based on the nature and granularity of data will be considered. The scope is to eventually produce and store the knowledge about device suitability so that functionalities that decide on the placement of service functionalities in an end-to-end delivery chain, can take this into account when performing their joint service and cloud platform optimization processes.

Acknowledgement

This work has been produced in the context of the H2020 WAZIUP as well as H2020 iKaaS projects. The WAZIUP/iKaaS project consortium would like to acknowledge that the research leading to these results has received funding from the European Union's H2020 Research and Innovation Program (H2020-ICT-2015/H2020-ICT-2015).

References

[1] Semtech, "SX1276/77/78/79 – 137 MHz to 1020 MHz Low Power Long Range Transceiver. rev.4-03/2015," 2015.

[2] S. Jeff McKeown, "LoRa – a communications solution for emerging LPWAN, LPHAN and industrial sensing & IoT applications. http://cwbackoffice.co.uk/docs/jeff∼20mckeown.pdf," accessed 13/01/2016.

[3] LoRa Alliance, "LoRaWAN specification, v1.01," Oct. 2015.

[4] Anders Quitzau, "Transforming Energy and Utilities through Big Data & Analytics" Big Data & Analytic, IBM, http://www.slideshare.net/Anders QuitzauIbm/big-data-analyticsin-energy-utilities

[5] www.waziup.eu

[6] www.ikaas.com