Low-cost IoT, Big Data, and Cloud Platform for Developing Countries

Corentin Dupont[1], Tomas Bures, Mehdi Sheikhalishahi[2], Congduc Pham[3], Abdur Rahim[1]

[1] FBK/Create-Net
cdupont@fbk.eu,arahim@fbk.eu
[2] Innotec21 GmbH
mehdi.sheikhalishahi@innotec21.de tomas.bures@innotec21.de
[3] University of Pau
congduc.pham@univ-pau.fr

Abstract. Gartner forecasts that 6.4 billion connected things will be in use worldwide in 2016, up 30 percent from 2015, and will reach 20.8 billion by 2020. In 2016, 5.5 million new things will get connected every day. Furthermore, the current research and marker trends shows the convergence between IoT and Big Data. On the other hand, Africa as a continent has seen very little of this activity. In this paper we present WAZIUP, a project aiming at building an open innovation platform able to accelerate innovation in rural Africa. The WAZIUP platform will allow to develop IoT applications coupled with Big Data capacities. The platform is taylored to the specific requirements and constraints of African users. We will give an overview of the WAZIUP IoT and Big Data platform, detail its technical aspects and finally introduce four use case deployements.

1 Introduction

TODO ►*change this intro to fit paper objectives: low cost HW review, Security/AuthN/AuthZ, data analytics using Elastic search, market opportunities*◄

TODO ►*C. Pham: the idea is to say that there will be a huge uptake of IoT devices that will be deployed worldwide for a large variety of applications; and that needs suitable platforms. This uptake will definitely come from the possibility and the availability of low-cost IoT devices that can be deployed in a very simple manner thanks to new radio technologies.*◄

ICT developments in Africa has already enabled significant modernizations across traditional sectors. Notable examples are the micro-health insurance accessible through mobile devices, index-based crop insurance and crowd-sourced management of public services. These innovative applications recognize and leverage commonalities between sectors, blur traditional lines, and open up a new field of opportunities.

The opportunity for ICT in Africa is huge especially for IoT and big data: those technologies are promising a big wave of innovation for our daily lives. The promise of IoT is to connect billions of sensors, devices, equipment and systems.

In turn, the challenge is to drive business outcomes, consumer benefits, and to create new value. The new mantras for the IoT Era is the collection, convergence and exploitation of data. The information is collected from sensors, devices, gateways, edge equipment and networks and stored in their respective IoT platforms. This information is processed in order to increase business efficiency through automation while reducing downtime and improving people productivity.

While developed countries are discussing about massive deployment of IoT, countries in Sub-Saharan Africa are still far from being ready to enjoy the full benefit of Iot. They face many challenges, such as the lack of infrastructure and the high cost platforms complexity in deployment. At the same time, it is urgent to promote IoT worldwide : WAZIUP will contribute by reducing part of the technology gap between EU and Africa. The goal of WAZIUP is to deploy an IoT and big data platform for African needs and validate it through several Sub Saharan Africa real-life use cases.

WAZIUP targets the rural community in Sub-Saharan Africa: about 64% of the population is living outside cities. The region will be predominantly rural for at least another generation. The pace of urbanization here is slower compared to other continents, and the rural population is expected to grow until 2045. The majority of rural residents live on less than few euros per day. Rural development is particularly imperative in sub-Saharan Africa, where half of the rural people depends on the agriculture/micro and small farm business, other half faces rare formal employment and pervasive unemployment. For rural development, technologies have to support several key application sectors such as living quality, health, agriculture and climate changes.

The biggest challenge of WAZIUP is to reduce costs and power consumption while increasing the robustness of the hardware. Hardware has to be robust enough so as to require lower maintenance and handle environmental and deployment treats as well. WAZIUP will present an innovative design of the IoT platform dedicated to the rural ecosystem. To achieve that, low-cost, generic building blocks will be deployed for maximum adaptation to end-applications in the context of the rural economy in developing countries. Another challenge of WAZIUP is to be able to manage the network deployment and to facilitate IoT communication. Lower cost solutions has to be used : privilege price and single hop dedicated communication networks, energy autonomous, with low maintenance costs and long lasting operations. Dynamic management of long range connectivity has to be taken into account (e.g., cope with network & service fluctuations), such as devices identification, abstraction/virtualization of devices, communication and network resources optimization. Finally, WAZIUP aims to exploit the potential of big-data applications in the specific rural context. Data will be collected from the IoT sensors themselves, but WAZIUP will also collect open data from other sources to build predictive models and enrich the platform.

From a technical standpoint, WAZIUP will pay attention to all related privacy and security aspects with specifics addressing the involved communities such as farmers, and developers.

Continued Openness will be ensured through the release of open specification and open software components and/or algorithms. Low-cost and low-energy consumption will be possible through the design of innovation hardware (sensors/actuators), and of IoT communication & network infrastructure.

The challenges outlined above will be tackled using an open IoT-Big Data Platform with affordable sensors connected through an Iot-Cloud open platform. The technical functionalities encompassed by the platform will be a cloud-based real-time data collection combined with analytics and automation software, an intelligent analytics of sensor and device data, an integration to third parties platforms and a Platform-as-a-Service (PaaS) provider.

The PaaS will provide to business clientele with independently maintained platform upon which their web application, services and mobile applications can be built.

The rest of this paper is structured as follows: Section **??** presents the architecture of the WAZIUP platform. Section **??** shows the implementation chosen. Section **??** details the deployement of WAZIUP, contrained by its environment. Section **??** presents four use cases that will be used to validate the WAZIUP concepts. Section 4 presents a survey of the literature on the topic, together with a survey of the Big Data Open Source tools. Finally we conclude the chapter in Section 5.

2 The IoT era with low-cost IoT for all

2.1 IoT connectivity made easy

Recent Low-Power Wide Area Networks (LPWAN) technologies for Internet-of-Things (IoT) introduced by Sigfox and Semtech's (LoRa^{TM}) are currently gaining incredible interest and are under intense deployment campaigns worldwide. They definitely initiated a new innovation cycle as they obviously provide a much better connectivity answer for IoT (most of IoT devices have small amount to data to send and very limited battery power) compared to traditional cellular-based connectivity (e.g. GSM/GPRS/3G) or short-range technologies such as IEEE 802.15.4. They offer several kilometers range without relay nodes to reach a central gateway, thus greatly simplifying large-scale deployment of IoT devices as opposed to the complex multi-hop approach needed by short-range radio technologies. Fig. 1 shows a typical extreme long-range 1-hop connectivity scenario to a long-range gateway, which is the single interface to Internet servers, using low-cost LoRa radio modules available from many vendors. Most of these long-range technologies can achieve 20km or higher range in line-of-sight (LOS) condition and about 2km-4km in non-LOS conditions [**?**, **?**] such as in dense urban/city environments.

2.2 Low-cost DIY IoT hardware

Commercial IoT devices are getting mature but they are definitely too expensive for very low-income countries. In addition, these highly integrated devices are

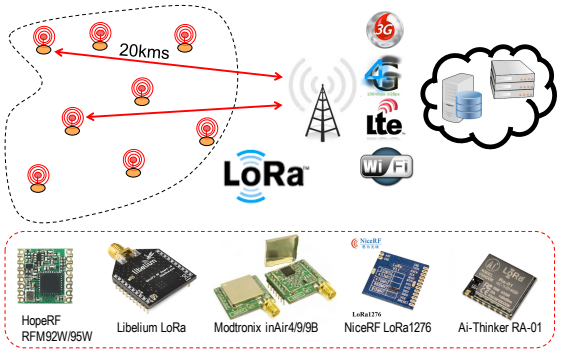

Fig. 1. Extreme long-range application with new radio technologies

difficult to repair with their parts being hardly locally replaced. The availability of low-cost, open-source hardware platforms such as Arduino boards definitely pushes for a Do-It-Yourself (DIY) and "off-the-shelves" design approach for a large variety of IoT applications. The Arduino ecosystem is large and proposes various board models, from large and powerful prototyping boards to smaller and less energy-consuming boards for final integration purposes as illustrated in Figure 2. For instance, the small form factor Arduino Pro Mini board based on an ATmega328 microcontroller has a high performance/price tradeoff and can be used to build a low-cost generic sensing IoT platform with LoRa long-range transmission capability for about 7 euro: 2 euro for the Arduino and 5 euro for the radio module!

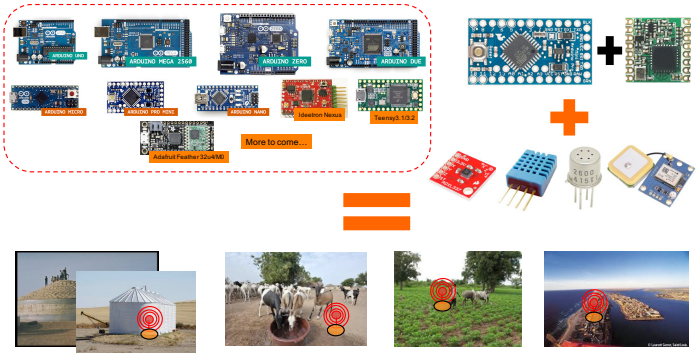

Fig. 2. Generic low-cost IoT hardware

Integration of these generic IoT becomes straightforward and the Arduino Pro Mini is available in the 3.3v & 8MHz version for much lower power con-

sumption, offering the possibility of running for more than a year on 4 AA regular batteries as illustrated in Fig. 3.

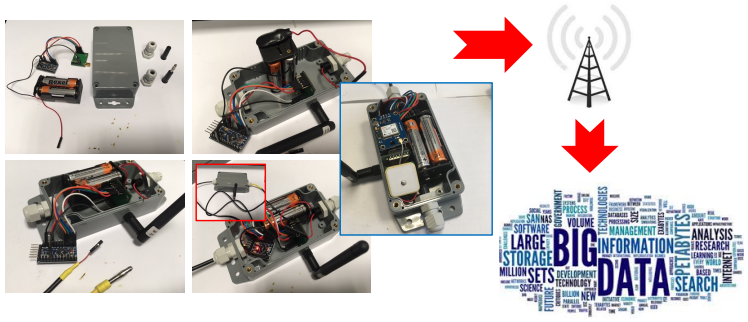

Fig. 3. Easy integration with DIY approach for maximum appropriation

It is expected that this availability of low-cost DIY IoT will create a tremendous uptake of the technology on a large-scale, for a large variety of applications, including those from developing countries as even a limited deployment of IoT devices can have huge impacts.

2.3 Hardware and Software Security

In a large scale IoT platform, that consists of local and global clouds, providing security is a challenge. A local cloud is connected to one or several IoT gateways that are close to local cloud.

An IoT gateway is a wireless based on 4G/3G communications; thus, we cannot build a local/private network space with Local Cloud even. Somehow we need to have a public endpoint to access Orion from the Internet. Therefore, even in local cloud model we need a public endpoint. Even in local cloud model, we need a public IP or the IoT gateway would have to be in the local network. In our platform, global cloud provides the main data broker (global Orion context broker) which receives data from distributed IoT gateways. IoT gateway communications with global Orion need to pass through the Internet. If Orion is exposed to the outside world using a public IP address of platform, then all access to Orion have to be controled in such a way to allow only authorized accesses, and deny non-authorized ones.

One way to provide security for such a communication is to use FIREWALL, and allow IP addresses of IoT gateways to be able to access Orion endpoint (e.g. public IP of platform: Orion Port). And then, deny all other accesses to Orion. Since we know that which IoT gateways are connected, or would like to connect to our platform this is a solution. At the end, we are enforcing access control to Orion by design with this solution. However, one issue can come from gateway with Internet access through a 3G dongle where the IP address is not always the same, and it may change.

Generally, we can distinguish two scenarios for local cloud:

- 1) Connected with limited bandwidth (typically charged by transmitted data): Here we use local cloud to optimize for data usage because connecting to internet is expensive and slow. Data are transferred to local cloud from internet, but that's low amount compared to what would be needed to access the dashboard remotely. Here the local cloud need a public IP or we will have to implement a method to circumvent NATs. This would mean the local cloud periodically pull the data from global cloud with static public IP or it keeps an open connection to a cloud with static public IP. Data go from the IoT gateway to the global cloud and from there to the local cloud.
- 2) Disconnected IoT gateway: In this case, the gateway must be connected with some local communication means (e.g. wifi, ethernet or LoRA) to the local network of the local cloud.

Therefore, it is important to have a co-design from hardware and software experts in place in order to address different scenarios.

Relying on IP address is not really good in some cases. There is another way in which we can rely in defining gateway id, and accepting only upload from registered gateways. It is easy to add a field in the message, to have the gateway id; thus, we can define access policies based on registered Gateway ids. For that, at platform, we need to receive the message, decode it and see if it is from a registered gateway id to let the communication happen. For this case, Gateway ID plus secure token sounds good for simple security. Alternatively, we could go for certificates. This would go nicely with securing the communication to go over SSL. To be sufficiently secure, we should communicate over HTTPS and use certificates. Since an IoT gateway is a Linux machine, we can certainly do the certificates and HTTPS.

This would mean that we have our own Certification Authority (CA). From this, we would generate certificates for the Orion and the gateway. Then we would setup both the parties to require that the opposite party presents a certificate signed by the CA.

This would have two benefits: 1) proper authentication, and 2) securing data during transport.

3 WAZIUP Platform

Figure 4 presents a global overview of Waziup cloud platform stack, and its connection to IoT gateway through data broker (Orion Context Broker). The role of each component is presented, together with the technology selected in parenthesis. The WAZIUP platform uses three distinct Cloud layers (in blue in the picture):

1. Infrastructure as a Service (IaaS),
2. Container as a Service (CaaS),
3. and finally Platform as a Service (PaaS).

The first layer is provided by OpenStack. Its main role is to provide Virtual Machines (VMs), in which we run the full platform. This layer is fundamental because most of Cloud vendors (Amazon, Rackspace) use VMs as basic selling units. The second layer is provided by Kubernetes. The role of this layer is to provide and serve containers, such as Docker containers, to WAZIUP services and applications. The containers provide light-weight and ultra-fast virtualization for applications and micro-services. The containers themselves are running inside the VMs. The third and final Cloud layer is provided by Deis. It provides services to developers, such as compiling and deploying of an application. All the applications pushed by the users will be compiled with Deis and hosted in containers on Kubernetes.

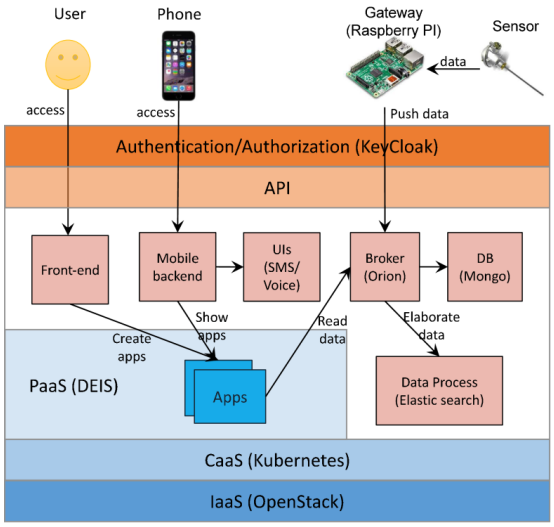

Fig. 4. Global overview of WAZIUP Cloud platform, and services

The Gateway pushes sensors' data to the data broker, which is FIWARE Orion. The data is distributed to the applications requesting it. Orion also interfaces with the database and the data processing (Elastic Search), for historical data analysis.

Keycloak provides both Authentication and Authorization management. For instance, when accessing the dashboard, the user need to provide a username and password. This is provided by Keycloak authentication layer. To access the platform, the users (Web, mobile) and external components (IoT gateway) need to go through the API server. The API server exposes a common API for all the services of the Waziup platform. Each of the endpoints of the API server is secured with Keycloak.

In addition, through the dashboard and APIs the user can access only sensors that are authorized for him. This is enforced by Keycloak authorization layer.

Mobile phones are used to interfaces with the SMS and voice commands component. This component allows Waziup applications to send SMS and voice notifications to the users.

In next sections, we present WAZIUP platform components in more details on security and data analytics.

3.1 Platform Security Architecture

IoT security has several dimensions; one is from communication between IoT devices and software platform; the other side is related to the whole software platform managing data, and services.

Figure illustrates WAZIUP services architecture in more details. The access to different WAZIUP services is performed by the WAZIUP APIs server (Dashboard API Server in Figure . The APIs server acts as a gateway placed in the demilitarized zone between the Internet and the internal network with WAZIUP services. WAZIUP APIs server provides public endpoints for the internally hosted services and proxies to them. All these services will not provide any Ingress to outside world. APIs Server can communicate with all Kubernetes internal services. And can respond to Client queries about different services. Thus, APIs Server will act as a proxy to provide security for WAZIUP services.

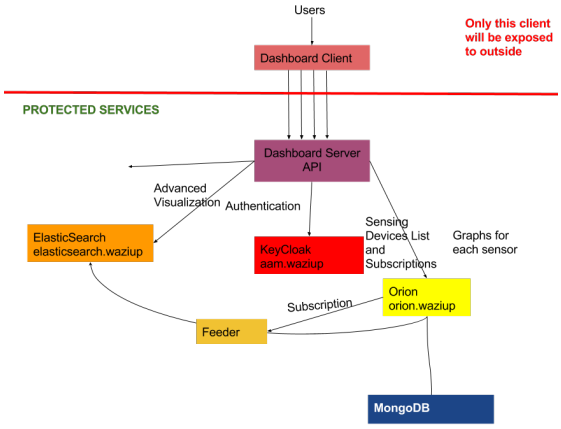

Fig. 5. Detailed Picture of WAZIUP Cloud platform, and services architecture

Authentication is the first requirement in implementing security. Users should be first identified by WAZIUP platform, then they can request access to different resources; that we call this step as authorization. WAZIUP APIs server takes care of the authentication and authorization of incoming requests from Internet. The authentication is performed with the help of Keycloak server. The authorization is done by the WAZIUP APIs server directly. Depending on whether a service is accessed in an interactive manner (from the web-browser) or in a programmatic

manner (directly from another service), WAZIUP APIs server provides two basic flows:

- Authentication for interactive use (e.g. from a Dashboard web client): A user accesses a page (e.g. through a Dashboard web client). The browser sends a request to the APIs server. The server finds out that client has not authenticated yet and redirects the clients browser to a login screen provided by Keycloak
  User enters his/her credentials. Keycloak validates the credentials, sends a request to APIs server notifying it that the user is authenticated and redirects users web-browser back to the original page.
  When the WAZIUP APIs serves the web page now, it creates a session for the user and includes the access token to the session. The response to the client then includes a cookie with the session id.
  Any subsequent request by the user includes the session id, which enables the WAZIUP APIs server to lookup the access token and validate the access.
- Authentication for programmatic use (e.g. from a sensor gateway): A user generates (typically through some web-based client) an offline access refresh token. The token is given to a device (e.g. the sensor gateway) as a configuration parameter. When the device wants to make a request, it contacts Keycloak and requests an access token to be generated based on the refresh token. Keycloak server provides an access token. (As opposed to the refresh token, the access token has rather limited duration of validity - typically in order of minutes.) The device makes a request to a service endpoint (on the WAZIUP APIs server). As part of the request, it includes the following header (where access_token is to be replaced by the actual access token received from Keycloak): Authorization: bearer ACCESS_TOKEN

Once the authentication is successful completed, the APIs server uses the access token to validate if the user/device has access to a particular resource.

For the sake of the authorization, every user has an attribute permissions (this is maintained by Keycloak). The attribute permissions attaches the the role of the user (admin, advisor or farmer) to resources (sensors, etc.) under a particular service path in Orion.

Examples of values of the permissions attribute:

- "admin": admin access to anything regardless of the service path
- "advisor: /FARM1; advisor: /FARM2" - advisor (i.e. read-write access) to anything under /FARM1 and /FARM2 service paths
- "farmer: /FARM1" - farmer at /FARM1
- The roles can be combined, such that a user gets different roles at different service paths: "advisor: /FARM1; advisor: /FARM2; farmer: /FARM2; farmer: /FARM3"

Keycloak setup The approach described above requires that every user has an attribute permissions in Keycloak. Moreover, Keycloak has to be set up to include the attribute permissions into tokens it provides (in particular the access token). This is because the WAZIUP APIs server inspects the access token

and reads the permissions attribute from it. Then, it interprets the permissions attribute to authorize a user w.r.t. to a particular resource.

The particular steps to obey in Keycloak setup are the following:

```
Required setup for a Client in Keycloak:
Configure Mappers. Add there a mapper as follows:
Name: permissions
Mapper Type: User Attribute
User Attribute: permissions
Token claim name: permissions
Claim JSON Type: String
Add to ID token: ON
Add to access token: ON
Add to userinfo: ON
Download config under Clients/dashboard/Installation
Select "Keycloak OIDC JSON"
Copy the text to "keycloak.json" in the WAZIUP APIs project
Required setup a User in Keycloak:
Create a user as usual
Create a new attribute "permissions" in Attributes and fill it with the permissions specificat
```

3.2 Data Management and Analytics Architecture

WAZIUP architecture integrates Orion for providing current sensor values and Elasticsearch to keep the history (timeseries) of sensor values. The Elasticsearch Feeder service is responsible for transferring data (current sensor values) from Orion to Elasticsearch (for historical record keeping). Figure presents the place of Feeder between Orion and Elasticsearch.

Elasticsearch Feeder is written in NodeJS. It accesses Orion and Elasticsearch through their REST APIs. It allows transferring data from Orion to Elasticsearch either in periodic manner (e.g. every 5 minutes) or by subscription when data are recorded in Elasticsearch only if they change in Orion.

The Elasticsearch Feeder is configured by specifying a set of tasks, where each task defines the Orions service path along with sensors their attributes and maps it to an Elasticsearchs index. A commented example of the configuration is below. The configuration is provided in YAML:

```
# Defines the HTTP endpoint for subscriptions
# Has to be present, even if no subscription-triggerred tasks are defined below.
endpoint:
  # Unique ID of the Feeder instance
  id: feeder1
  # URL at which Feeder endpoint can be reached
  url: http://feeder
  # Host/port to which Feeder should bind
  host: 0.0.0.0
```

```
  port: 8000

# Defines the default ElasticSearch connection settings.
# Can be specialized in the same section in a task
elasticsearch:
  host: elasticsearch
  port: 9200

# Defines the default Orion connection settings. Can be specialized in the same section
# in a task
orion:
  # MUST NOT end with slash
  uri: http://broker

# Defines (multiple) tasks. Each task defines sensors and target index
# in ElasticSearch
tasks:
- trigger: subscription

#  Trigger types are: "time", "subscription"
#  If it is "time", Feeder periodically polls Orion with the period specified below
#  If it is "subscription", Feeder listens to notifications and recreates the
#    subscription every period below.
#  Renewal of the subscription includes new listing of sensors, which means that
#    after the "period" time
#  new sensors will be automatically included in the subscription.
#  Note that Orion sends a notification every time a subscription is recreated.

#  period: 5000

# Specifies the throttling (in seconds) when subscription is used. If not specified, no throtti
# throttling: 5

#  Specifies task-level Orion settings
#  Entries from the global "orion" are used here as defaults. They can be overridden
#  here. The same way, the fields here can be moved to the global "orion".

  orion:
    service: waziup
    servicePath: /FARM1/TESTS

#  Specifies task-level Elasticsearch settings
#  Entries from the global "elasticsearch" are used here as defaults. They can be
#  overridden here. The same way, the fields here can be moved to global
#  "elasticsearch".
```

```
  elasticsearch:
    index: test

#  Specifies the filter over the sensors discovered.
  filter:

#  If set, only entities listed below will be considered. If missing, no filtering by
#  IDs is done
    ids:
    - WS_FARM1_Sensor2
    - WS_FARM1_Sensor3
    - WS_FARM1_Sensor4

#  If set, only attributes listed below will be considered. If missing, no filtering by
#  attributes is done.
    attributes:
    - SM1
    - SM2
```

3.3 Applications Platform Architecture

WAZIUP vision is to make it easy for IoT developers to develop new applications, dashboards and services. With WAZIUP APIs Server model, we will develop some generic proxy APIs services such as for Security (KeyCloak proxy API), Data Management (Orion proxy API), Data Analytics (ElasticSearch proxy API), etc. These APIs may be developed as part of WAZIUP APIs Server, or separate Service APIs. These are just proxy APIs calling the respective service APIs. Figure 6 presents this concept for realization of application templates.

We will move toward this architecture design at the Dashboard level. With that we can realize Application Development templates for IoT developers much simpler. We give them Development APIs and templates in order to develop their own customized Application. These APIs will be developed as a NodeJS server.

With that, instead of contacting Orion in Dashboard Client, it will call Orion API of WAZIUP Server API with some parameters, and will get for example the list of sensors, sensors data, subscriptions, etc.

This approach will make it easier for developers to develop new dashboards having in place these already well developed and tested APIs on top of Waziup Services.

4 Related Work

In this section, we survey the related work. We first analyse similar work from the literature. Regarding the IoT big data aspect, the research have already been

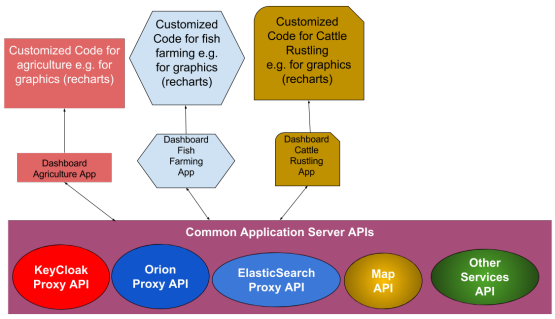

Fig. 6. A global view of WAZIUP Application Template platform

largely instantiated inside Open Source frameworks. We present a selection of the most interesting Open Source contributions.

4.1 Review of literature

IoT in Africa: Their is very little penetration of IoT in Africa, as evidentiated in [1] and [2]. The authors of [1] provide a survey, country by country, of the undertaking of IoT. They also document some of the challenges affecting adoption of IoT in the continent. Africa has only 7% of her households on the Internet; this is far behind the worlds figure of 41%. Given this lag in the baseline technology needed to implement Internet of Things, the author of [2] advocate for a technological leap and an African-centric approaches to IoT. Taking the case of a drought early warning and assets tracking systems, the author demonstrates that by innovatively incorporating the realities such as the prevalence of African indigenous knowledge on weather, unreliable communication, low-end mobile phone handsets, among others, a home-grown Internet of Things flavour has higher chance of succeeding. An extensive report from Cisco [3] provides also many insights on the current use and potential of Internet of Things (IoT) technologies in tackling global development challenges, highlighting a number of specific instances where IoT interventions are helping to solve some of the worlds most pressing issues.

A deployment of a Wireless Sensor Network for precise irrigation in Malawi is presented in [4]. For the system to be self-sustained in terms of power, the study used solar photovoltaic and rechargeable batteries to power all electrical devices. The system incorporated a remote monitoring mechanism through a General Packet Radio Service modem to report soil temperature, soil moisture, WSN link performance, and photovoltaic power levels. Irrigation valves were activated to water the field. The paper give insights to develop a robust, fully automated, solar-powered, and low-cost irrigation system to suit the socioeconomic conditions of small scale farmers in developing countries.

The authors of [5] provides a survey of possible IoT applications in South Africa and Zambia. In particular, they identify examples of IoTs to mitigate

the agricultural needs of these communities for the domains of crop farming, weather forecasting, wildlife management, forestry, livestock farming, market identification and rural financing.

IoT in Agriculture: There is not a lot of literature on the specific topic of applying IoT in agriculture, and practically none went it comes to rural Africa. In [6], the author presents a work supporting the transition from traditional agriculture to modern agriculture in China. They propose an agriculture intelligent system based on IOT for organic melon and fruit production. A number of new technologies are used, such as RFID and sensors. They monitor temperature, humidity, light and CO_2 around the crops and use a small model on the fruits of growth process. Always in China, [7] uses Internet of things and RFID technologies to realize automatic control production of agriculture.

In [8], the authors perform temperature control in greenhouse using Zigbee. In [9], the authors elaborate a crop growth model. The model is then embbeded in their IOT application system. This allows them to make the system more intelligent and adaptive for the facility agriculture. The authors of [10] and [11] proposes remote agriculture monitoring and process automation. They are both based on gateway infrastructure and wireless connection. The work in [12] shows a semantically enhanced digital agriculture use case built with the OpenIoT platform.

In [13], the authors uses IoT to check electronically on the vital signs of the cattles. Their tool facilitates the day to day management of dairy activities. It also provides forecastings allowing to handle weather related issues, cattle health and emergencies.

IoT is also deployed within the product supply chain, another key area of agriculture. In [14], the authors builds a quantitative trust model to describe the trustworthiness of foods delivered in supply chains. The Internet of Agricultural Things (AIoT), where the technologies of the Internet of Things are widely used in all of the phases in the agriculture industry, is proposed to resolve the food safety problem. In order to provide a common model to describe and transmit the data in agriculture Internet of Things, [15] proposes a specific ontology, while [16] uses a naming service to identify products.

With difference with the literature surveyed in this section, the proposed Waziup platform is a full IoT platform taylored entierly for African need and constraints. In particular, it as application hosting capacities based on the PaaS paradigm, is resilient to disconnections and provides big data capacities.

4.2 Review of Big Data tools

Far from an exhaustive list, this paragraph describes the most used Open Source Big Data tools and compares them in order to give a better understanding of the Big Data ecosystem. Moreover, this review gives an indication on the best tools fitted for WAZIUP platform.

Databases and data warehouses HDFS, developed by Apache, is a distributed, scalable and portable file-system written in Java for the Hadoop Framework[4]. It has been designed for large dataset analysis and by its structure has high fault tolerance. It is the basis upon which everything works in the Hadoop Ecosystem. Build on top of HDFS, Apache HBase is a distributed, non-relational column oriented datastore[5]. HBase is designed to efficiently address random access and fast record lookup. It has the capability to handle extremely large tables of data with low latency. Though, this data storage tool should be used when random and real-time read/write access to data is needed and when many thousands of operation per seconds need to be performed on large datasets (up to petabytes).

Apache Hive[6] is a data warehouse infrastructure that can manage and query unstructured data as if it were structured. As a full component of Hadoop Ecosystem, it uses MapReduce for execution and HDFS for storage. It has its own language SQL-like (HiveQL) that brings expressiveness to the queries. This storage mode should be used for SQL-like queries and when higher language than MapReduce is needed. Used by big companies who cant afford to lose data (Apple, Netflix, Spotify), Apache Cassandra[7] is a column oriented database of structured data. The data are highly available thru column indexes and are automatically replicated thru multiple nodes for fault tolerance. Cassandra has a unique masterless ring design that is easy to setup and to maintain[8].This tools should be use when losing data is not the critical point and not affordable.

First considered as an outsider, getting rid of traditional table-based relational database, mongoDB[9] quickly became a must-have tool: a NoSQL, relational, document oriented database. Document are shared in JSON format with dynamic schemas (called BSON) and makes the integration of data sometimes easier. This database is very useful when you need to consume your data in many applications, as many connectors have been developed.

Data publication and subscription Apache Flume[10] is a distributed, reliable and available service for efficiently collecting, aggregating and moving large amounts of streaming event data. Flume should be used if the data is designed for Hadoop as it can move them to HDFS. It has many built-in sources and sinks and can process data in-flight using interceptors, which is useful for data masking or filtering. It is composed of agents and data collectors (and interceptors if needed).

More general purpose, Apache Kafka[11] is a high-throughput, distributed, publish-subscribe messaging system. It can replicate events, has low latency and is capable of data partitioning. Kafka is also easily scalable and this tool is very useful when the data is to be consumed by multiple applications. It is

[4] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
[5] https://hbase.apache.org/
[6] https://hive.apache.org/
[7] http://cassandra.apache.org/
[8] http://www.planetcassandra.org/what-is-apache-cassandra/
[9] https://www.mongodb.com/
[10] https://flume.apache.org
[11] http://kafka.apache.org

composed of producer, consumers and topics. Actually, we might not have to choose between Kafka and Flume, as both can work quite well together. If the workflow design requires streaming data from Kafka to Hadoop, using a Flume agent with Kafka source to read the data makes sense. This association is quite common and is called Flafka[12]. If a Data/Context Scenario is developed, we may need to use a context broker. Orion Context Broker[13] (FIWARE platform) is a publish/subscribe platform that is able to register context elements and manage them through updates and queries. It is possible to subscribe to context information when some conditions occurs (e.g. an interval of time passed or the context elements have changed). Orion is a C++ implementation of the NGSI9/10 REST API binding developed as a part of the FIWARE platform.

Data processing Obviously, choosing a data processing tool depends mostly on the outcome expected from the data. The most common tool for BigData analysis, and what we probably think at first, is Hadoop MapReduce[14]. It has proven its efficiency in many ways and is an incredible tool. But if we want to step a bit aside of Hadoop workflow or if we have specific needs, other tools exists and some are becoming more and more powerful.

But first, lets talk about this milestone Hadoop MapReduce. MapReduce programming model contributed to the amazing progress of BigData processing this past decade. By breaking down the work and recombining it in series of parallelizable operations, it is simple but incredibly efficient and scalable to ten thousands of machines if needed. 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.

If a higher level of programming on top of MapReduce is needed, Apache Pig[15] is the one. Pig has its own language (PigLatin) similar to SQL and works on top of MapReduce. Pig Engine parses, optimizes and automatically executes PigLatin scripts as a series of MapReduce jobs on a Hadoop cluster. Its easy to learn and opens Hadoop to data professionals who may not be software engineers.

First designed to work with HDFS on top of YARN, Apache Spark[16] is a different system for processing data and can work out of Hadoop ecosystem with other data managements systems. It does not work with MapReduce and it can be up to a hundred times faster than MapReduce with its capacity to work in-memory, allowing to keep large working datasets in-memory between jobs, reducing considerably the latency. What makes it more and more attractive to many users worldwide, is its wide range of applications: batch and stream processing (micro-batch processing with 0.5s latency), machine learning (MLib), SQL

[12] http://blog.cloudera.com/blog/2014/11/flafka-apache-flume-meets-apache-kafka-for-event-processing/

[13] http://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker

[14] http://hadoop.apache.org

[15] http://pig.apache.org

[16] http://spark.apache.org

(with Hive), graph Analytics (graphX). Language supported are Java, Python and Scala.

Demand for stream processing becoming more and more important in Big Data analysis, Apache Flink[17] has been recently developed and is growing very fast. 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. Language supported are Java and Scala.

Machine learning Machine learning is the union between statistics and artificial intelligence. It blends AI heuristics with advanced statistical analysis. We let the machine learn about the data, make decisions, and then apply statistics. Algorithms used for this tasks 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. Overlearning (or overfitting) of the model must be carefully checked.

Without any math or programming requirement, KNIME[18] is an analytic platform that allow the user to proceed the data in a user-friendly graphical interface. It is a good tool to train your model and evaluate different machine learning algorithms rapidly. If the workflow is already deployed on Hadoop, a machine learning library exists and is called Mahout[19]. Spark also has his own machine learning library called MLib[20]. H20[21] is a software dedicated to machine-learning, which can be deployed on Hadoop or Spark (Flink in development). It has an easy to use Web interface, which makes possible to combine big data analytics easily with machine learning algorithm to train models.

Data visualisation and exploration To visualise the data in real time, Freeboard provides a simple, real-time dashboard, commonly used in IoT world[22]. There is a direct Orion Fiware connector. To connect with streaming engines, a JSON connector can be used. Design is simple and customisation is not possible, but it is a very good dashboard to visualise easily raw data coming from sensors, before data analysis.

Tableau Public[23] offers a good visualisation and exploration tool on batch data. Tableau is a software where you can upload your analysed data (previously extracted in .csv format). The visualisation tool is very powerful and allow a deep exploration the data. However it is not designed for really Big Data with large datasets and the open Source version of Tableau (Public) does not offer the data streaming capacities (e.g. Spark connectors). Nevertheless, Tableau Public is a

[17] http://flink.apache.org
[18] http://www.knime.org
[19] http://mahout.apache.org
[20] http://spark.apache.org/mllib/
[21] http://www.h2o.ai/
[22] https://freeboard.io/
[23] https://public.tableau.com/s/

highly customisable, user-friendly and intuitive exploration tool for data that have already been processed and analysed.

To visualise data in real-time, after analysis (filtering, aggregating, correlating), one of the best tool is probably Kibana[24]. It is the visualisation tool coming with ElasticSearch. Elasticsearch is a search server based on Apache Lucene that 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.

5 Conclusion

With ICT technologies, Africa can dramatically improve its agricultural productivity by enabling the rapid and cost-effective deployment of advanced and real time monitoring. The immediate effect is to improve coordination and logistics, by reducing time and investment horizons for Research and Development and new product development, and by allowing for the enhanced analysis of historical and ongoing data. With respect to the water sector, such systems can also dramatically improve water use efficiency, allow for the growth of water provider SMEs by providing practical and cost effective new payment, monitoring and management systems. This technology can also offer a new cost-effective alternative for integrated watershed management by networking real-time water quality and flow data. Furthermore, given the fundamental roles which agriculture and water play in the African economic and social development and more generally onto environmental sustainability, WAZIUP can both directly and indirectly bring a much wider range of benefits related to food security, gender equality, poverty reduction and resource use efficiency.

Acknowledgements

This work has been carried out within the European project WAZIUP (H2020-ICT-687607).

References

1. Nashon Onyalo, Hosea Kandie, and Josiah Njuki. The internet of things, progress report for africa: A survey. *International Journal of Computer Science and Software Engineering*, 2015.
2. M. Masinde. Iot applications that work for the african continent: Innovation or adoption? In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 633–638, July 2014.
3. ITU. Harnessing the internet of things for global development. Technical report, ITU, 2015.

[24] https://www.elastic.co/fr/products/kibana

4. Million Mafuta, Marco Zennaro, Antoine Bagula, Graham Ault, Harry Gombachika, and Timothy Chadza. Successful deployment of a wireless sensor network for precision agriculture in malawi. *International Journal of Distributed Sensor Networks*, 2013.
5. N. Dlodlo and J. Kalezhi. The internet of things in agriculture for sustainable rural development. In *Emerging Trends in Networks and Computer Communications (ETNCC), 2015 International Conference on*, pages 13–18, May 2015.
6. Fu Bing. Research on the agriculture intelligent system based on iot. In *2012 International Conference on Image Analysis and Signal Processing*, pages 1–4, Nov 2012.
7. Fan TongKe. Smart agriculture based on cloud computing and iot. *Journal of Convergence Information Technology (JCIT)*, 2013.
8. L. Dan, C. Xin, H. Chongwei, and J. Liangliang. Intelligent agriculture greenhouse environment monitoring system based on iot technology. In *Intelligent Transportation, Big Data and Smart City (ICITBS), 2015 International Conference on*, pages 487–490, Dec 2015.
9. Xiangyu Hu and Songrong Qian. Iot application system with crop growth models in facility agriculture. In *Computer Sciences and Convergence Information Technology (ICCIT), 2011 6th International Conference on*, pages 129–133, Nov 2011.
10. Nakutis, V. Deksnys, I. Jaruevicius, E. Marcinkevicius, A. Ronkainen, P. Soumi, J. Nikander, T. Blaszczyk, and B. Andersen. Remote agriculture automation using wireless link and iot gateway infrastructure. In *2015 26th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 99–103, Sept 2015.
11. Prosanjeet J. Sarkar and Satyanarayana Chanagala. A survey on iot based digital agriculture monitoring system and their impact on optimal utilization of resources. *Journal of Electronics and Communication Engineering (IOSR-JECE)*, 2016.
12. P. P. Jayaraman, D. Palmer, A. Zaslavsky, and D. Georgakopoulos. Do-it-yourself digital agriculture applications with semantically enhanced iot platform. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6, April 2015.
13. A. Ilapakurti and C. Vuppalapati. Building an iot framework for connected dairy. In *Big Data Computing Service and Applications (BigDataService), 2015 IEEE First International Conference on*, pages 275–285, March 2015.
14. W. Han, Y. Gu, Y. Zhang, and L. Zheng. Data driven quantitative trust model for the internet of agricultural things. In *Internet of Things (IOT), 2014 International Conference on the*, pages 31–36, Oct 2014.
15. Siquan Hu, Haiou Wang, Chundong She, and Junfeng Wang. *AgOnt: Ontology for Agriculture Internet of Things*, chapter AgOnt: Ontology for Agriculture Internet of Things, pages 131–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
16. Y. Liu, H. Wang, J. Wang, K. Qian, N. Kong, K. Wang, Y. Shi, and L. Zheng. Enterprise-oriented iot name service for agriculture product supply chain management. In *Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on*, pages 237–241, Oct 2014.