AGISIT_18_19_GROUP17_PROJECT_REPORT

The Infrastructure of a “Smart” Factory

Management and Administration of IT Infrastrutures and Services

Team nr.: 17

91872: Jakub Syrek

91718: Michal Tomaszewski

91930: Roy De Prins

71003: Carlos Manuel Mendes Branco

Information Systems and Computer Engineering

IST-ALAMEDA

2018/2019

Contents

List of Figures

Chapter 1

Introduction

The purpose of the project is to design an architecture of a 'Smart Factory'. The factory is concerned with the production, commercialization and distribution of industrial chemical products, fertilizers and nanomaterials. The Industrial Complex consists of three factories, located in Barreiro, Alverca do Ribatejo and Estarreja. These three factories must deliver a large number of bags with different products to all 18 district capitals in Portugal, plus Madeira and Azores. The Industrial Complex must be extremely innovative: The transformation to a Connected 'Smart' Factory is the ultimate goal of the project.
Besides this architecture, also a Proof Of Concept for an Analytic Tool in a Cloud environment to be used in the Factory Production System must be deployed.

Chapter 2

Design of an Architecture

2.1 Functional Design

The Functional Design is used to gather the Assumptions, Requirements, Constraints and Risks of the project. This phase is important to indicate the approach to the design that will be taken.

2.1.1 Assumptions

The Assumptions for the System are:

A01 Use of a cloud.

A02 The same exterior/geographical faults (i.e. earthquakes, forest fires, ...) do not occur at multiple factories at the same time.

A03 Work flow, upkeep and management of the machines can be done remotely from anywhere in the world.

A04 All machines are able to connect with each other, no hardware limitations prevent this.

A05 All processes are able to be automated, being it easy or hard.

2.1.2 Requirements

The Requirements for the system are:

R01 [General] Shop floor machines should report metrics on speed, uptime, number of products created/moved, queue times and possible upcoming malfunctions (ie: X-sensor went mute; Y-sensor detected a delay on the conveior belt, Z-sensor detected value above normal range).

[R01.1][General] Delays above a given time should be escalated to the shop-floor supervisor.

R02 [General] End tier equipments must have complete wireless communication.

R03 [Availability] Communications between all equipments and systems must be continuous.

R04 [Security] Each equipment may only communicate within a single factory, apart from delivery equipments.

R05 [Transparency] Factories must send compressed daily data of the shop-floor machines usage and statistics to a central server.

R06 [Transparency] All available information for all transactions, per factory, must be stored in a central repository on a daily basis.

R07 [Security] All communications must be secure.

R08 [Backup/Recovery] State for all systems must be recoverable in case of catastrophe.

R09 [Traceability] Ongoing deliveries must be trackable.

R10 [Performance] All computations should be handled in a centralized way.

R11 [Performance] Control feedback from shop-floor statistics should be done in real time and take no less than 10 seconds between information emission and control feedback reception.

R12 [Security] Access to all data should be restricted by group.

R13 [Usability] All systems must be administrateable by the company.

R14 [Integration] When available, open source implementations should be chosen in detriment of others unless no other option can be found or under solid conjecture.

R15 [Security] All platforms and equipments should only be accessible from the company headquarters or from within the factories.

R16 [Scalability] New factories should be easily added to the existing system.

R17 [Maintainability] All equipments should be easily replaceable.

2.1.3 Constraints

The Constraints for the System are:

C01 Budget - the amount of money that can be spent on the development of the project.

C02 Computation power - the amount of computing power that current machines can provide.

C03 Space limit - the amount of space that can be used to store new machines and sensors required for the project.

C04 Energy usage - the limit for energy that can be provided to the factories in order to empower the new infrastructure.

C05 Legislation - the local low and constraints connected with the protection of the data.

C06 Vendor - the limited number of vendors and suppliers that can be contracted in order to provide the required machines and devices.

C07 Training availability for the employees to acquaint them with the new system.

C08 Internet connection speed in order to transfer huge amount of data between data centers.

2.1.4 Risks

The Risks for the System are:

RI01 Software security - risk of system being hacked.

RI02 Hardware security - risk of intentional physical damage of hardware, like sensors, servers, etc.

RI03 System error due to software bug or incompatible components.

RI04 Lost connection between machines.

RI05 No power energy in fabric/data center.

RI06 Intentional/unintentional incorrect using of system by employees.

RI07 Exterior/geographical faults

RI08 Indoor fire (i.e. data centers)

RI09 Exceeding constraints limits

2.2 Conceptual Design

The Conceptual Design considers the high-level design of the architecture of the project. It is used to show the components related to the provided requirements, in an abstract form.

2.2.1 Overview

Inside each factory, all conveyor belts, forklifts, storage rooms are to be fitted with wireless, self-powered, communication devices and sensors in order to provide a continuous information stream to be handled by the central services. Moreover, each factory has to have complete wireless coverage along side with powering devices so as to be able to broadcast information and receive factory specific warnings (namely supervisors via app message for example) for delays and faults currently happening. The same is true for depots but with their respective equipments.

All computation is handled in a de-centralized cloud service, hosting all required services, API's and storages. Factories and depots on the other hand are designed to have very reduced computation necessities as well as all Enterprise level solutions. Continuous reports to stakeholders and office staff are displayed via a front-end API

that is also able to handle minor request such as data statistics visualization, item search, database querying and current statuses for factories.

2.2.2 Smart Factory Logical Levels

Now we set up to design the Logical Levels of the Smart Factory. Figure 2.1 represents 3 main levels:

- Device Level - gathering various hardware devices which are collecting raw data, like statistics of machines in a factory or weight of some products (single bags or whole truck).
- Control Level - here specific, prepared data is send to Information Level (with some API). We have various types of connectivity - wire, wireless and by remote access which ensures mobility of system control.
- Information Level - collecting tools like ERP and MES, which provide BPM software that allows the factory to use a system of integrated applications to manage business assumptions. MES real-time system give us also ability to track and document the transformation of raw materials to finished goods. Of course, beside that, there can be other 3rd party cloud instance.

2.2.3 The Functional Layers of the Architecture

In smart factories, the main base for acquiring data is smart devices and controllers placed on the manufacturing machines. They build the Physical System layer of the architecture. Next, the data is processed in the Data Infrastructure Layer. At the Application layer, data is analyzed and integrated with the system using web services and APIs. Finally, everything is served in the Web Control Panel and Dynamin Production System at the Collaboration and Business Processes Layer. The information is shared between layers using different protocols and standards. These functional layers can be seen in figure 2.2.

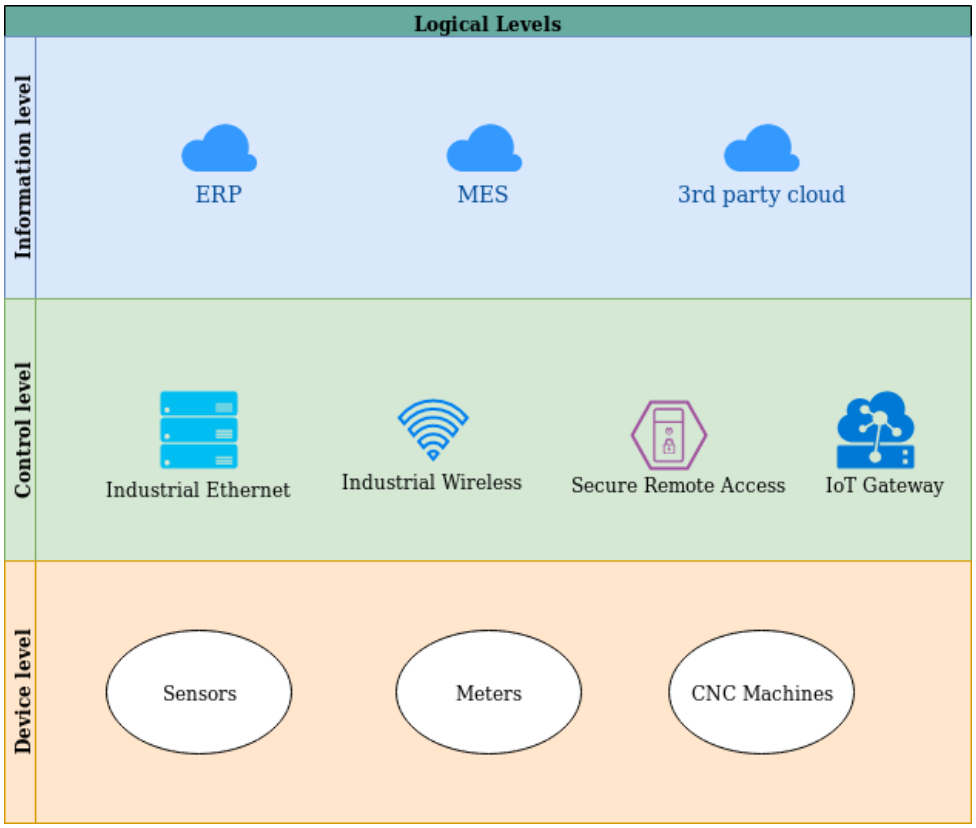

Figure 2.1: Smart Factory Logical Levels

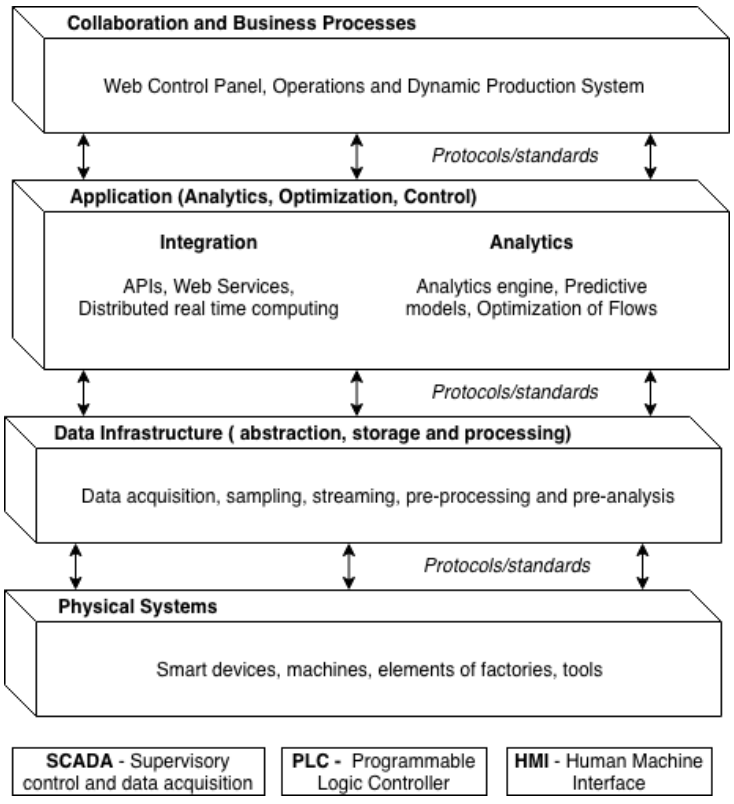

Figure 2.2: Architecture: Functional Layers

2.2.4 The Smart Manufacturing Value Chain

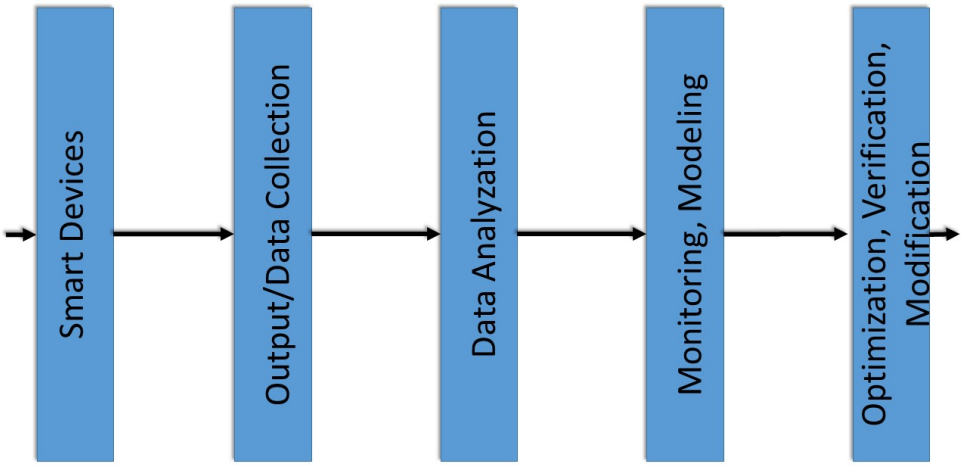

Figure 2.3: The Smart Manufacturing Value Chain.

The smart manifacturing value chain is visualized in figure 2.3. The chain starts with smart devices that have the purpose to deliver data to get insights about the state of the smart factory. Next there is the collection of the outputs that the smart devices deliver. After that the collected data has to be analyzed in order to make some models out of it; you monitor the characteristics of the factory. After the models are made, conclusions can be drawn. It is possible that everything works correctly (verification), that some little adjustments must be made (optimization) or that the whole strategy should be changed (modification).

2.2.5 Smart Factory Tiered Model

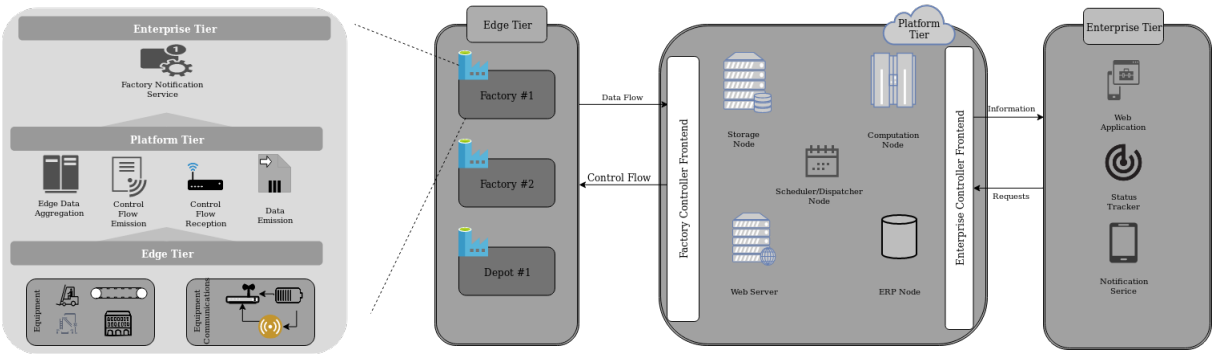

Figure 2.4: Smart Factory Tiered Model

Here we set up to design a tiered view of the Smart Factory. In Figure 2.4 we can see on the left side a structured view of each factory and equipments both in use and needed and how these interconnect in order to report statistics and informations of the workplace. Deposits, even though they are not specified, work similarly to factories apart from the fact that some depot equipments can communicate outside the factory (ie, trucks in delivery). Furthermore we assume that all communications are to be mediated by the central platform where control software, databases, dispatchers, external service providing pages and other programs are located. These aforementioned services are responsible for all information control flows and data articulation and are to be cloudified.

2.3 Logical Design

The Logical Layer of the Architecture consists of (see figure 2.5):

- Smart Factory Backbone link that provides data from smart devices and infrastructure of the smart factory.
- Two Databases instances connected to the switch to store the logs and data.
- Two Application Servers used to preprocess the data gathered in the smart factory.
- Load Balancer connected with the factory backbone that distributes traffic between Application Servers,
- Multiple computers at the Datacenter Office to track the status of the Datacenter and to manage it locally,
- Set of switches to connect all parts of the Datacenter,
- Router that provides connectivity withe the cloud,
- Cloud with the other parts of Smart Factory infrastructure.

Most of the elements consist of doubled or multiplied elements (i.e. two separated Databases that store duplicated data) to provide robustness in case of any problems.

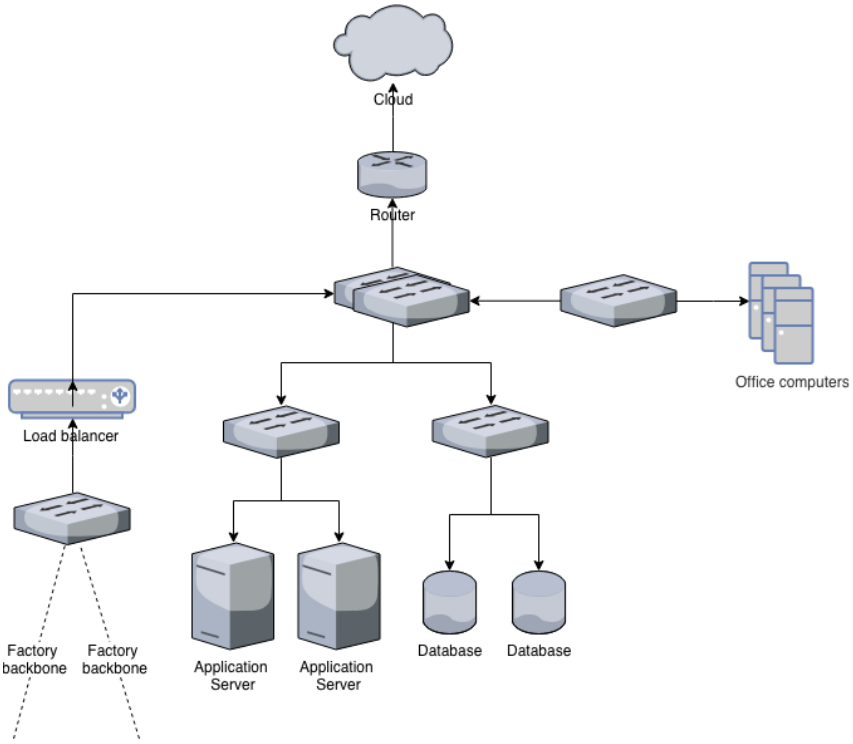

Figure 2.5: A Logical View of a Datacenter infrastructure of the factory

The whole infrastructure of each Datacenter is integrated within a cloud. The Logical Layer of single Datacenter is implemented in each of Smart Factories.

2.3.1 A Logical View of the Cloud-based Analytic Tool components

We set up to design a Logical Logical View of the Cloud-based Analytic Tool components. Figure 2.6 represents how data is processed by cloud components:

- Input Data - data gathered from various hardware devices/sensors.
- Ingest component - streaming data is used for logging, monitoring and our cloud functionalities.
- Pipelines - with Cloud Dataflow we can gather data and transform it for specific purpose - here ”The computation of sigma” occurred in a case of production of fertilizers.
- Storage - as a storage services we use regular Cloud Storage and HDFS - Hadoop Distributed File System, used in MapReduce procedure.

- Analytics - Apache Hadoop used as a service for a computation and processing data with MapReduce method. Specific view in figure 2.7
- Application and Presentation - e.g. website application showing results of processing data.

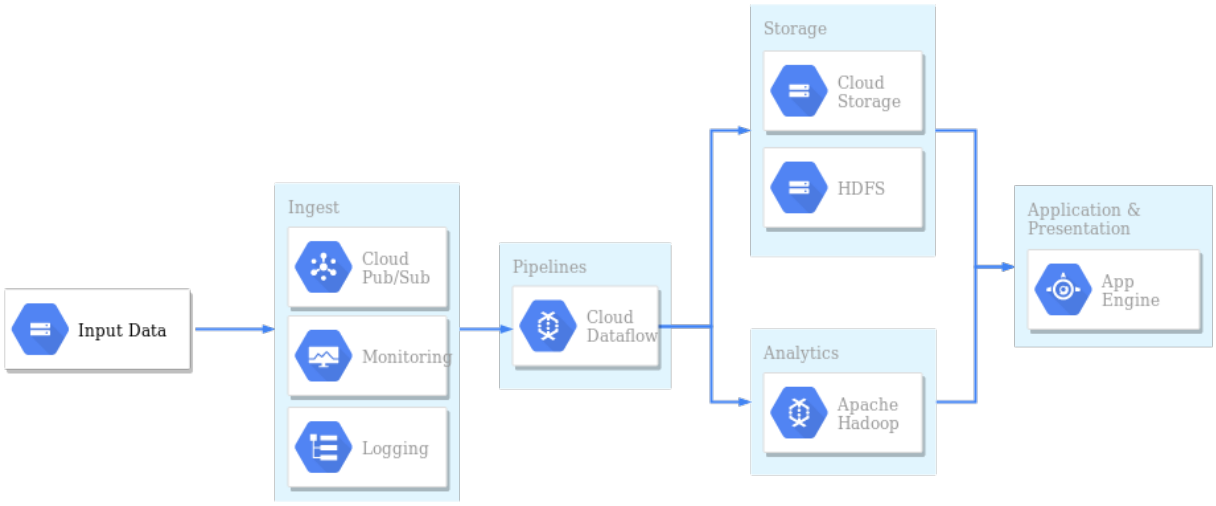

Figure 2.6: Logical View of the Cloud-based Analytic Tool components

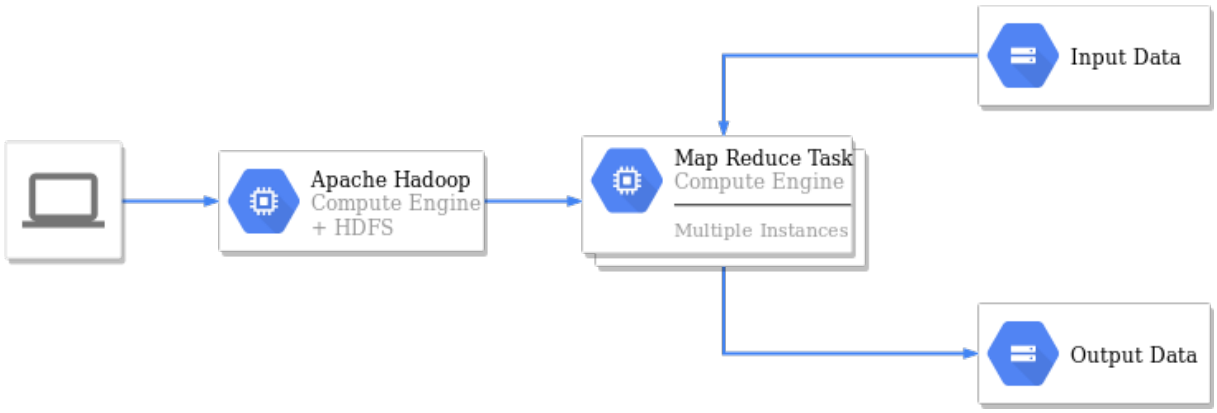

Figure 2.7: Apache Hadoop MapReduce method

The Apache Hadoop framework consists of the MapReduce method and Hadoop Distributed File System (HDFS). The input data that consists of key value pairs is stored on the HDFS. They are split into fixed-size blocks, and allocated to the user-specified maps. The map function is applied on the input block to produce intermediate key-value pairs. The intermediate data is partitioned by the key, and the grouped records are shuffled to the appropriate reduces. Then, the shuffled records are merged and sorted in the node that a reduce task located. Each reduce sequentially processes key-

value pairs by the user-specified reduce function to generate the final output key-value pairs.

Multiple instances perform a series of Monte Carlo runs, the result being a summary of the runs (a vector).

2.4 Physical Design

Now we cover the Physical View of the factory which focuses on the flow of data, and gathering of the same, through different buildings that make up a company's factory grounds. Moreover a detailed view of the equipment and the location within each building will be provided along with limits and upgradeability capacities for the latter.

2.4.1 Shop-floor Overview

The following overview assumes that no more than *10Gbps* throughput will come out of any factory at any given moment.

Beginning with the shop-floors a switching equipment should be installed in the center of the building. In one of the sides (closest) a switching central should be installed having on the outside, the factory grounds, the connection to the backbone leading to the main building. On the inside, this switching central needs to be fitted with no less than 5 switches. 2 For the shop floor, two for administration purposes (1 redundant) and one more for quick substitution.

On the middle of the building and on the roof's inside, a cabinet with 3 switches should be equipped in order provide quick off ground access for repairing, the same process should be repeated for every 'node' we are about to install. A total of 3 *10Gbit 16 port* switches should be installed per node, 2 of them to guarantee at least *10Gbps* (in case 1 of the switches linked to the routers fails and a single maximum length path remains for each wing). The third one should stay for quick replacement in case of failure of any of the previous. All routers and switches in every configuration so far should be linked redundantly.

Following this 3 switch policy linked to 2 *2.4 GHz 802.11ac ready* routers, for each wing of the factory and every 100 meters apart these 'nodes' should be installed in order to cover the entire factory with wireless connection for all devices.

This is visualized in figure 2.8.

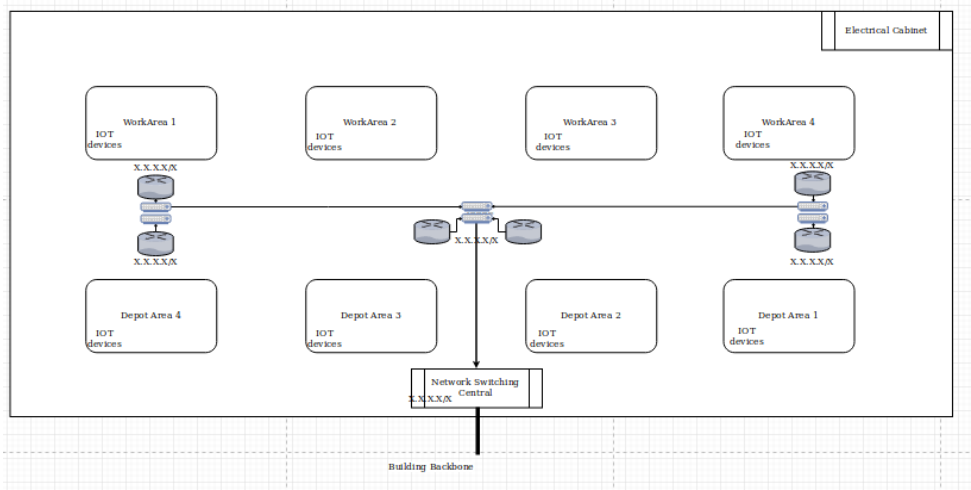

Figure 2.8: Shop-floor schema

2.4.2 Main Building Overview

The local datacenter or factory server/computation room should be located in the main building of the factory. Behind it should sit the control and operations room which will hold all monitoring equipment for both the datacenter and factory wide status. Furthermore and similarly to the shop floors all connections should be redundant. Routers for this building should be *10Gbit* with at least *16 ports* in order to support all office computers connected via ethernet. The same replication schema still applies, for the same distances specified the previous section. Likewise the routers should be *2.4 GHz 802.11ac ready*. The ISP link should be redundant (provided by 2 different ISPs) and both must provide a minimum bandwith of **Yet To Be Decided**.

Looking closer at the server room, pairs of redundant *26 port 46Gbps* routers (up to 1.5 of the maximum in-factory throughput) need to be installed in order to receive the factory data. The same applies for the out-link connecting to the ISP. Inside this room 2 access points should be made available, 1 for the control and operations room and another for the rest of the building. Further information should be presented in order to calculate the required bandwith between server racks and the switches connecting them.

This can be seen in figure 2.9.

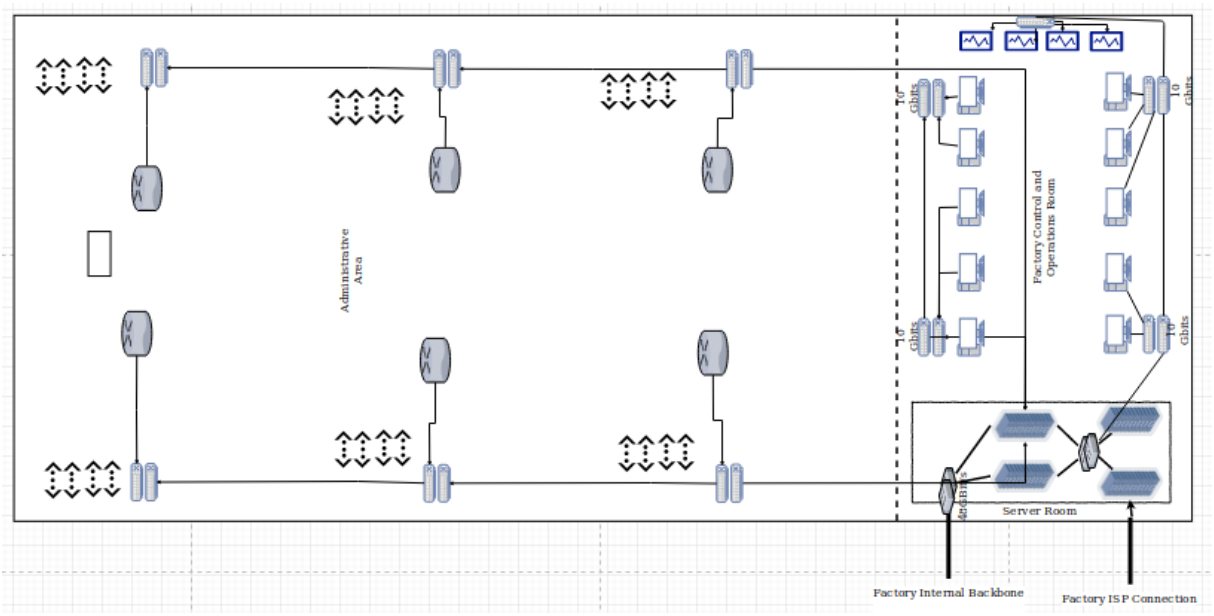

Figure 2.9: Main Building schema

2.4.3 Factory Grounds Overview

Finally we can see an overview of the factory grounds. Abstracting production lines and buildings *2N* multi-mode fibre optic cables should link all buildings to the main one (2N to support N failures). Cables should follow the 10Gbps-500meter 100Mbps-2000meter rule of thumb for the typical optic cable.

See figure 2.10.

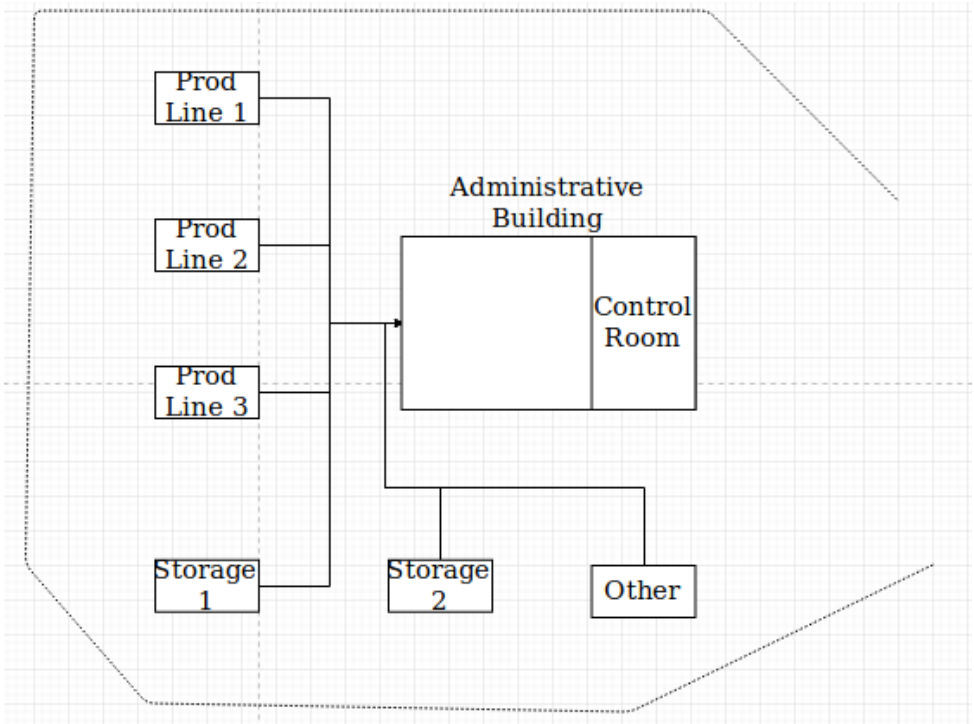

Figure 2.10: Factory Grounds Schema

Chapter 3

Proof Of Concept

Following the instructions provided in the zip file you will be able to deploy the entire network.

For the proof of concept we set do two things. Firstly generate a random dataset matching the requirements introduced by the user in a front end page (such as mean value, standard deviation and number of entries) and secondly, to compute these same parameters but in a cluster via hadoop and map reduce.

To achieve the latter a script is triggered to produce the above results by moving the *".dat"* file with the user input to the cluster's master node, then a dataset is generated and moved to a *gcp bucket* and the job (map-reducing) is submitted to the google cloud platform. The results are later shown back to the user.

Further details on the implementation of the pyspark map reduce can be seen at *"test-tenant/tasks/map_red.py"* file and webscripts are also present at *"test-tenant/task/webxscript.sh* (figure 3.1).

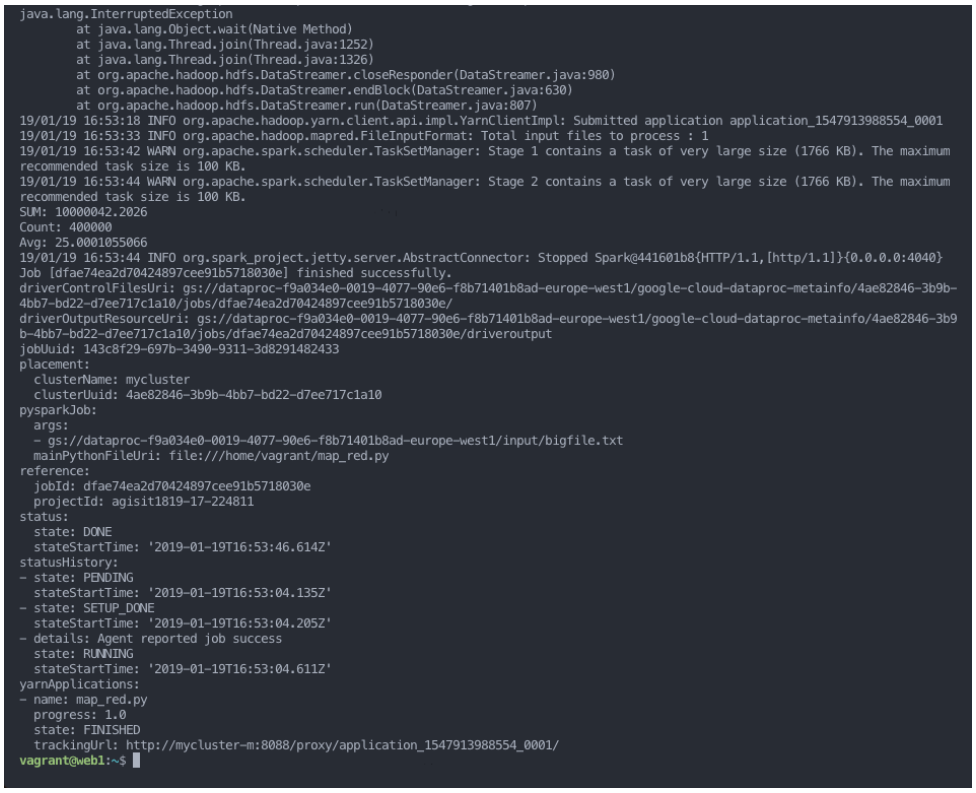

Figure 3.1: Calculating sigma

3.1 Handling input/output

Our idea was to deploy frontend page (we have chosen Angular framework) and back-end server (Python Flask framework) to the webserver.

Frontend page:

- Entering input data
- Sending input data to the server via REST request
- Getting response from the server (google cloud computation result)

Figure 3.2 represents front page of our solution.

Server:

- Transforming input data from frontend page into .dat file (used as a argument of emmiter.exe)
- Running the script with google cloud commands (webxscript.sh)

For deploying these components we used Ansible playbooks.

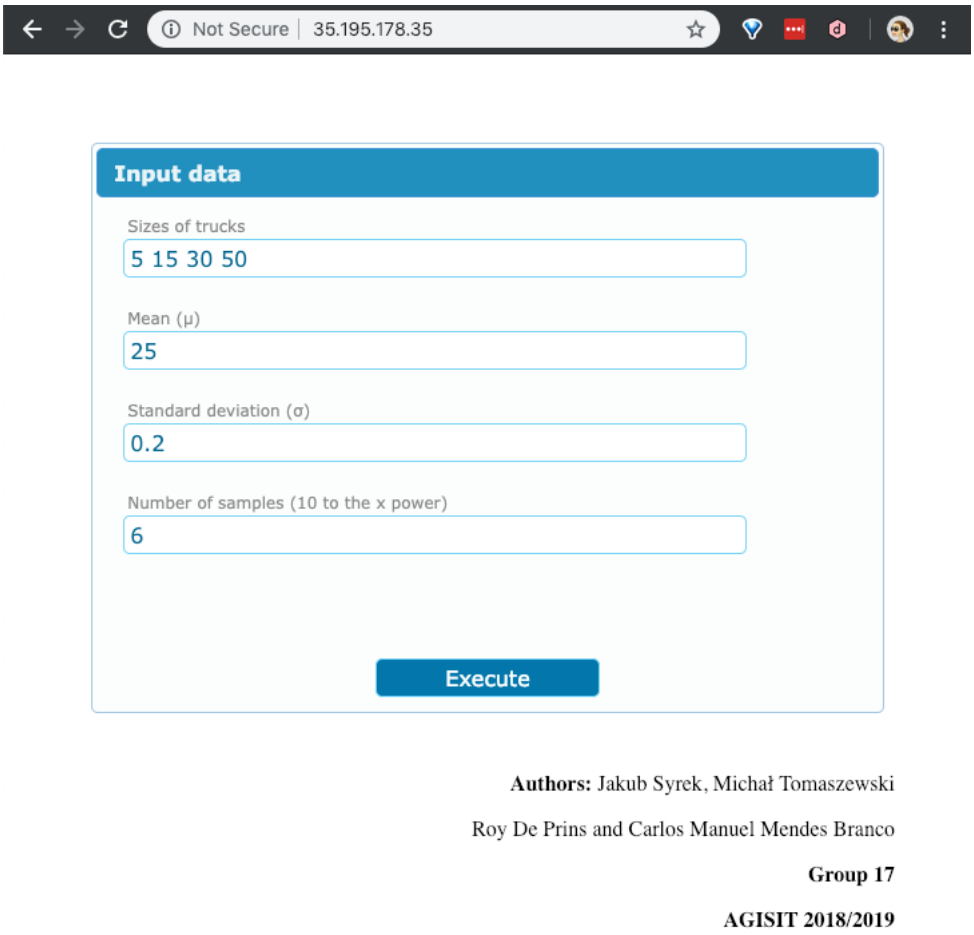

Figure 3.2: The front page

3.2 Problems

The only problem that we occurred and that stopped us from preparing a complete standalone solution was deploying the python server that would handle the frontend requests with calculation parameters into a google cloud host instance. We assume that the problem was connected with exposing the running python server to the host itself using ansible script.

Figure 3.3 represents working server at web1, but due to the problem with exposing it for frontend connections, we couldn't provide fully working system.

The possible solution would be using PHP script in order to handle frontend input, however due to the lack of time and knowledge about PHP itself, we failed to finish this last step.

Nevertheless the solution we provided can be used to solve the stated problem using manual approach (changing the .dat file manually and executing *webxscript.sh*

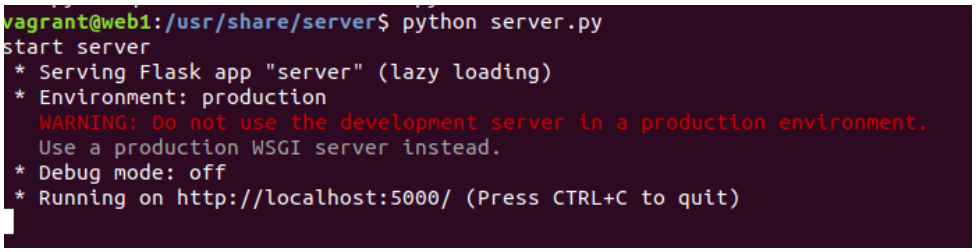

Figure 3.3: Working python server at web1 instance

on the web1 instance).