Group 4

Creation of a workload orchestration solution reducing electrical consumption in Orange datacenters

---

Advanced project S8

Students :
Ali Idrissi
David Renouf
Elie Nowominski
Maryem Koutit
Thomas Houlier
Tom Aucler
Yuntian Ding

Supervisors :
M. Frédéric DENIS
M. Alexis CARREL-BILLIARD

Contents

Introduction **3**

1 Project context **3**
1.1 Origin of the project . . . 3
1.2 Objectives and avenues of work . . . 4

2 Creation of a Kubernetes cluster and Prometheus integration **4**
2.1 First attempt : Physical cluster . . . 4
2.1.1 Cluster creation with Raspberry Pi . . . 4
2.1.2 Kubernetes with kubeadm . . . 5
2.1.3 Prometheus and Grafana . . . 5
2.1.4 Limitation of this method . . . 6
2.2 Second attempt : Virtual cluster . . . 6
2.2.1 Cluster creation with kind . . . 6
2.2.2 Prometheus and Grafana . . . 6

3 Final work and results **7**
3.1 Development of the algorithm . . . 7
3.1.1 Node labeling with Kubernetes . . . 9
3.1.2 Consumption data with Prometheus . . . 10
3.1.3 Production data with weather predictions . . . 10
3.2 Algorithm . . . 11
3.3 Results . . . 11
3.3.1 Grafana . . . 12
3.3.2 Graphs . . . 13

4 Project management **14**
4.1 Agile method . . . 15

Conclusion **15**

Glossary

Api API stands for Application Programming Interface and allows an application to communicate with other services without access to the details of their implementation. 9

Cluster A cluster is a set of machines (nodes) that allow containerised applications to be run. The term cluster was used in this project to refer to the whole: master node and slave nodes. 4–7, 15

Grafana Grafana is open source software for data visualisation which allows creating dashboards and graphs from several sources. It is linked directly to Prometheus in order to visualise collected metrics on relevant dashboards. 4, 6, 7, 12

HypriotOs Hypriot Os is an operating system for Raspberry Pi supporting Docker and Kubernetes technologies. It is initially installed on Raspberry Pis. 4

Kind Kind is a Kubernetes cluster virtualization tool. In the project, it was used to avoid the hardware constraint that would occur in a physical cluster. 6

Kubeadm Kubeadm is a minimalist and light version of Kubernetes that allows starting a minimal Kubernetes cluster. This version is used for the adaptation to the `Raspberry Pi`. 4, 5

Kubernetes Kubernetes is an open-source platform for managing workloads and containerised services (Docker) within an IT system. This was the core technology of the project which managed the orchestration of the worloads. 3, 5, 6, 9, 10, 15

Prometheus Prometheus is an open source software for collecting server metrics. It retrieves data through exporters and can be linked to Grafana to display the different metrics (e.g. CPU load used by a Raspberry Pi). 4–7, 10, 15

Raspberry Pi The Raspberry Pi is a single-board ARM-based nanocomputer which has the size of a credit card. It was used to model a data center in the project. 2–5, 15

Wiremock Wiremock is software that simulates an HTTP API. It allows using a service that is not offered by an existing API. 9, 10, 15

Workload A workload is a task that a computer or server must process at a given time. This term is used to designate the applications that run on the different work nodes. 4, 7–9

Introduction

Global warming and the production of greenhouse gases are both major issues of our time. Every company, especially digital companies concern about ecological issues. Since the start of the century, the amount of data on the internet and the cloud has experienced a significant growth. Nowadays the carbon footprint of data stocked in data centers, clouds and computers amounts to about 4% of the global carbon footprint. It will be one of the main sources of greenhouse gas emissions in the future. To tackle this global issue, the European government asked European companies to transform their datacenters into "green datacenters" by 2040. It is why Orange, one of the most important telecommunication company in France, has already started to consider their ecological transition.

This report consists of 4 parts. The first part gives the context and the framework of the project. The second item highlights the creation of a cluster, which is the core of the project. Then the next section presents the final attempt and explains the results. Finally, the last part shows the project management, which is an important part of the team's work.

1 Project context

1.1 Origin of the project

In this context, Alexis Carrel-Billard and Frederic Denis, two engineers from Orange, proposed a project topic as the beginning of this transition. The goal of the project is to propose a first solution to the Orange strategic team. The main target of the ecological transition is not to produce green energy but to optimize the electrical consumption in the data centers of Orange. Solar panels can bring a significant amount of green energy according to the size of the data centers. But an optimization system has not been put in place yet to consume smartly and sustainably this green energy .

Orange has 3 data centers in France, they are in Rouen, Nice and Val-de-Reuil (Paris). The aim of the project is to optimise the distribution of workloads in different data centers according to the available green energy. In this way, green energy would be optimally consumed.

Two Orange engineers defined the requirements at the beginning of the project. The project guideline was to model the data centers using a `Raspberry Pi` cluster and develop a worload orchestration solution using `Kubernetes`. The complementary technologies, like `Prometheus` and `Grafana`, were naturally integrated into the project subsequently.

Customer needs required the use of new technologies (Kubernetes, Raspberry Pi, Prometheus, Grafana). Before starting the technical part of the project, some bibliographic researches were carried out in order to define feasible schemes and to develop the first avenues of work.

1.2 Objectives and avenues of work

The main goal of the project is to propose a workload orchestration solution that allows reducing the electrical consumption in Orange datacenters. This objective can be completed in the following steps:

- Construction of a cluster composed of 4 `Raspberry Pi`
- Integration of `Kubernetes` into the cluster
- Integration of `Prometheus` and `Grafana` into the cluster
- Development of an intelligent workload orchestration solution

To build the cluster, Orange engineers provided 4 `Raspberry Pis`, a router and several ethernet cables. A Jetson Nano was available in case a `Raspberry Pi` didn't have enough power. The seond step and the third step were to integrate technologies into the cluster.

The first three steps were to build a functional `Kubernetes` cluster supervised by `Prometheus`. These steps were documented so that the work could be continued easily afterwards.

The final step was to develop an algorithm that distributed intelligently workloads in the cluster according to the available green energy. This algorithm must centralize the consumption and production data in each data center. Production data was simulated in term of weather data and consumption data was retrieved using `Prometheus`.

2 Creation of a Kubernetes cluster and Prometheus integration

The creation of a `Cluster` which allows the correct deployment of `Workload`s was the core of the project. Whether virtual or physical, the cluster is composed of a master node that orchestrates the deployment of `Workload`s and several working nodes that represent the data centers.
The cluster information was retrieved via `Prometheus` and then visualised on `Grafana`.

2.1 First attempt : Physical cluster

The first attempt was to consider a physical cluster as a group of data centers. A `Raspberry Pi` represented a datacenter.

2.1.1 Cluster creation with Raspberry Pi

The first step was to configure each `Raspberry Pi` by installing `HypriotOs` and `Kubeadm`.

Once the three `Raspberry Pis` were initialized, the next step was to interconnect them. To achieve this target, a local network composed of a router connected to the internet was created.

[Figure 1] shows the physical `Cluster` with 3 `Raspberry Pi` slaves, a Raspberry Pi master and a `Jetson Nano`. The 4 `Raspberry Pis` and the `Jetson Nano` were interconnected with the router connected to the internet. The role of the 3 `Raspberry Pi` slaves was to run workloads, the role of the `Raspberry Pi` master was to distribute the workloads and the role of the `Jetson Nano` was to host the Prometheus and the Grafana server.

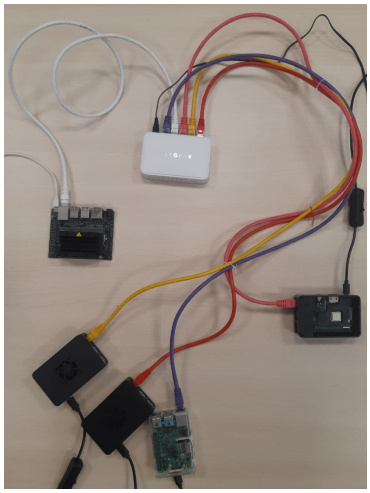

Figure 1: Physical cluster

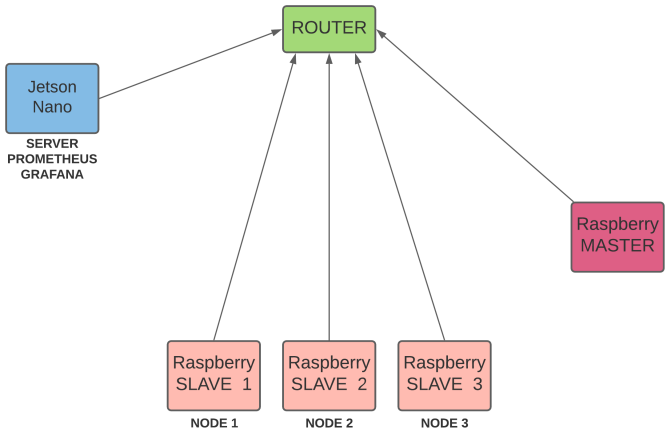

Figure 2: Diagram of the physical cluster

2.1.2 Kubernetes with kubeadm

Installing a `Kubernetes` system on a `Cluster` was a key point of the project. Considering the work with micro computers, there were several solutions for integrating `Kubernetes` to the cluster, for this project, Kubeadm was considered as the adapted one. `kubeadm` is a Kubernetes sub system created for micro-systems.

The installation of `Kubeadm` on each microcomputer is a complex prodecure. Therefore, this important section was documented.

Once the `Raspberry Pis` were configured and `Kubernetes` was integrated into the system, the cluster was ready to be used.
`Kubeadm` is easy to use and allows using `Kubernetes` on the `Raspberry Pis`. However, since `Kubeadm` is simplified relative to `Kubernetes`, manipulating configuration files and adding plugins caused difficulties.

2.1.3 Prometheus and Grafana

Once the cluster was up and running, the goal was to integrate a `Prometheus` system into the cluster. In fact, `Prometheus` allowed retrieving the internal metrics of the cluster that were useful for the project, this tool retrieved cluster metrics using an exporter installed directly on each `Raspberry Pi`.

The installations of the server Prometheus and Grafana were performed on the Jetson Nano which provided more power. Afterwards a problem occured : instead of retrieving the internal metrics from the Kubernetes cluster which include information like CPU load of a workload, number of workload on a Raspberry pi, Prometheus retrieved the metrics from the Raspberry Pi which include information like the power consumption, CPU load of the Raspberry Pi.

2.1.4 Limitation of this method

The lite version of Kubernetes showed its limitations when integrating Prometheus into the cluster. It was necessary to find an alternative in order to continue the project.

The richness of the agile method made it possible to change the trajectory. While the project was planned on the physical hardware, each member used a virtual version of Kubernetes. This virtual version allowed team members to practice this technology individually. In agreement with the two engineers from Orange, the team chose to use a virtual cluster for the rest of the project.

2.2 Second attempt : Virtual cluster

For the second attempt, the solution considered was to use virtual cluster software called Kind. Kind allows having a **Kubernetes virtual cluster** on a personal machine. This solution was chosen for its various advantages:

- No hardware constraint: everyone can manipulate the Kubernetes cluster from their personal machines.
- Easy to install: no need to configure Raspberry Pis

Kind seemed to be the best solution to replace the physical cluster. As a result, the use of Raspberry Pis became unnecessary, nevertheless the work done previously was not useless since the same technologies and concepts were reused.

2.2.1 Cluster creation with kind

The creation of a Kind virtual cluster on a machine is a complex procedure. Hence, this important section was documented as well.

2.2.2 Prometheus and Grafana

With the Kind cluster operational, Prometheus had to be integrated into the cluster. This section was crucial because it was the integration of Prometheus which was problematic in the first attempt.
The installation of Prometheus was achieved based on the Helm installation tool. Concretely, Prometheus was integrated to the Cluster through a workload that was running on the Cluster permanently. An advantage of using the Helm tool was that the Grafana

technology was installed at the same time.

The steps to install `Prometheus` on the `Kind` support were documented by the team.

Once Kubernetes, Prometheus and Grafana were integrated into the cluster, it was possible to observe the internal metrics of the cluster. To do so, a `Grafana` plugin created for Kubernetes data visualization was used. This plugin allowed creating simple and meaningful dashboards. For example, the CPU usage of each workload running on the cluster is shown below [Figure 3] :

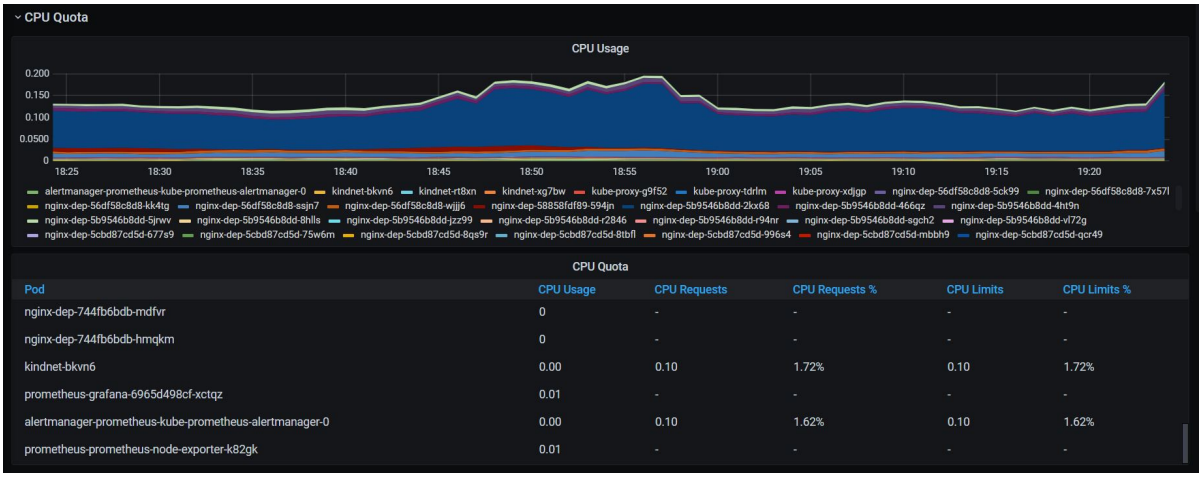

Figure 3: Basic metrics illustrated in Grafana

3 Final work and results

Development started after the virtual cluster was properly initialized and the `Prometheus` system was installed.

3.1 Development of the algorithm

The main idea of the solution is to create a program that retrieves useful information (renewable energy production, workload consumption on each datacenter) and deduces an optimal `Workload` distribution on the `Cluster`.

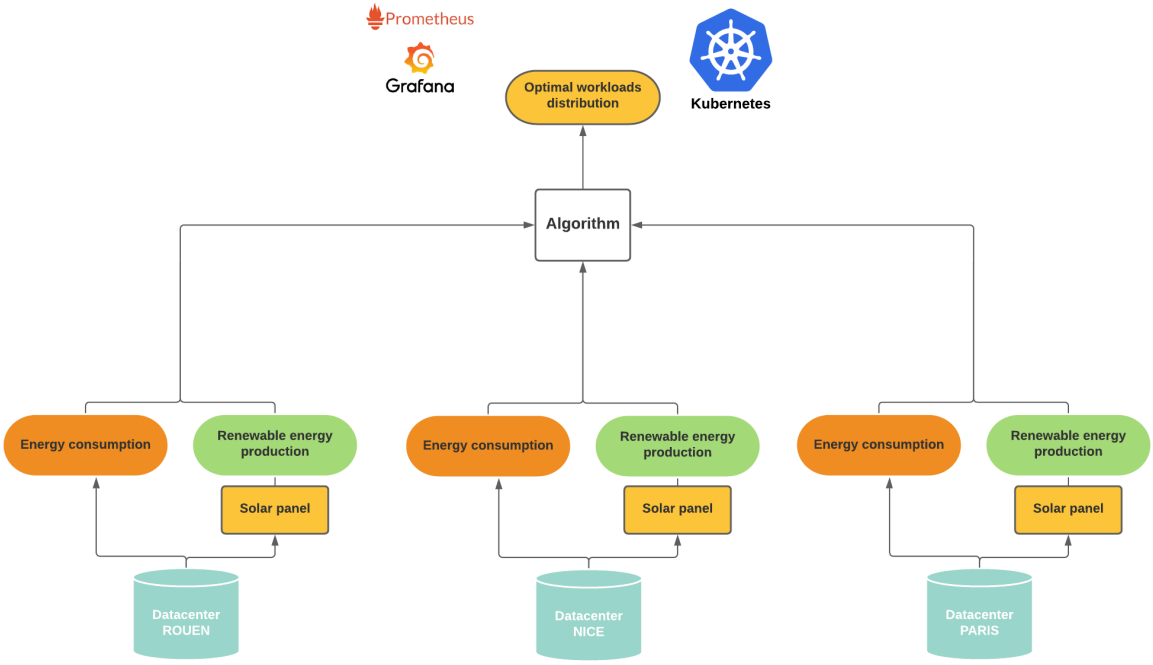

Figure 4: Scheme of the global idea

As illustrated in [Figure 4], the algorithm retrieved real-time consumption and production data for each data center. From this collected data, the algorithm ranked the data centers based on their energy provisions. A score was assigned to each data center, it represented the rate of `Workload` that the data center can support.

Here is an example to illustrate the ranking system :

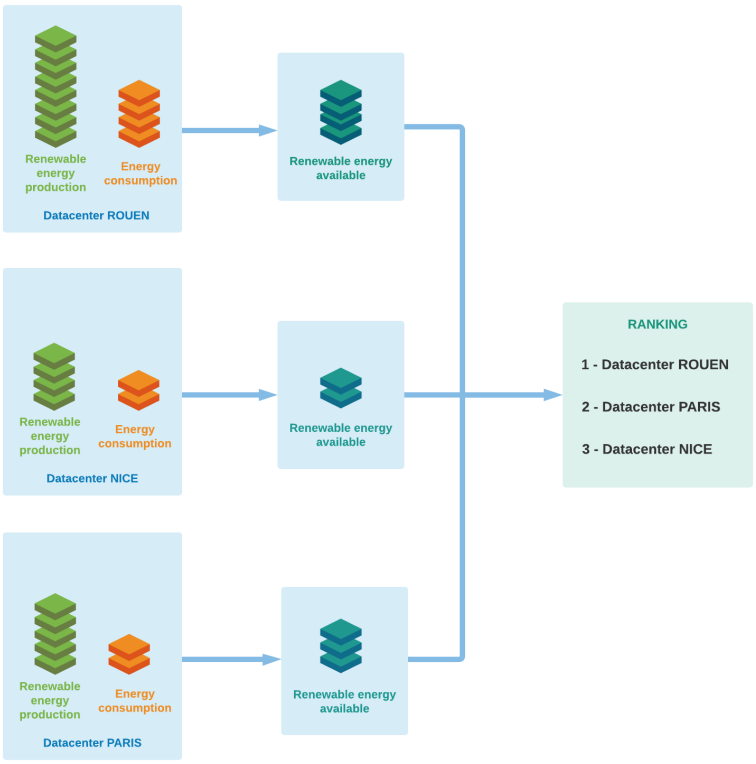

Figure 5: Example of the scoring idea

The example in [Figure 5] illustrates a case where the Rouen data center has the most available renewable energy, the Paris data center is the second and the Nice data center is the last. Then the ranking system assignes scores based on the amount of renewable energy available in each data center. `Workload` deployment is weighted by these scores. In this example, the majority of the workloads are deployed on the Rouen data center.

To summarize, the core of the solution is the algorithm that **centralizes** the relevant data and determines a rank for each data center in real time.

The solution was designed using the functionalities offered by the various technologies :

- **Nodes labelling** proposed by Kubernetes
- Retrieval of **internal metrics from the cluster** accomplished by Prometheus
- **Weather scenario** simulated by a weather `Api` and `Wiremock`

3.1.1 Node labeling with Kubernetes

`Kubernetes` natively offers node labeling functionality. And labeling consists simply in assigning a label to a node and a workload. The concept is to control the deployment of the `Workload` with the labeling. The workload is strictly deployed on a node which owning the same label as it.

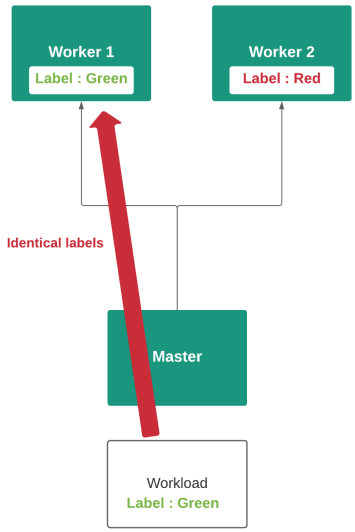

Figure 6: Kubernetes labelling feature

The labelling functionality offered by `Kubernetes` was directly integrated into the algorithm.

3.1.2 Consumption data with Prometheus

The `Prometheus` monitoring tool properly integrated into the `Kubernetes cluster` provides access to internal `cluster metrics`. For example, these metrics can be the CPU load used on each worker node or the number of `workloads` running on a worker node.

3.1.3 Production data with weather predictions

The production of renewable energy in a data center was a key input to the alogorithm's operation. The energy production data in Orange data centers is not accessible to the project team. Therefore, this data was simulated in term of weather data. Based on weather data (sunshine, cloud cover...), the solar energy production was determined.

The technical implementation of this section was based on various weather scenarios. And the creation of the scenarios was realized with the `Wiremock` tool. The first step was to centralize weather data on the three geographical areas where Orange data centers are located. Then, weather data was exposed on an API using `Wiremock`.

Thus, the algorithm was able to directly query the API to retrieve the weather data. Then, the weather data was converted into solar energy production for each data center.

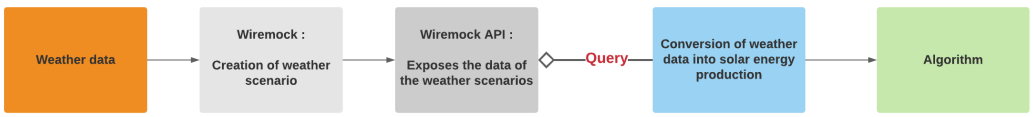

Figure 7: Prediction of production data according to weather scenarios

The calculation of the available energy for a data center took into account all the data provided by the API. Then the produced energy was compared to the energy consumed by the same data center to deduce the available green energy. By comparing the available energy on each data center, a coefficient was deduced and then used to deploy the appropriate number of workloads to the data center.

The following formulas calculate the avaible energy and the coefficent for a data center :

Computation of the available energy for a data center:

$$E_{available}(t+1) = E_{available}(t) + E_{production}(t) - E_{consumption}(t) \tag{1}$$

Computation of the coefficient for a data center:

$$Coefficient(t) = \frac{E_{available}(t)}{\Sigma E_{available}(t)} \tag{2}$$

3.2 Algorithm

The algorithm uses the three concepts described above in a simulated manner. The program allows to centralize the production and consumption data of the datacenters in real time. A score is then assigned to each datacenter, which represents the rate of workload it can support. Finally, the workloads are intelligently distributed to the datacenters using the labeling.

[Figure 8] illustrates the technical aspects combined together :

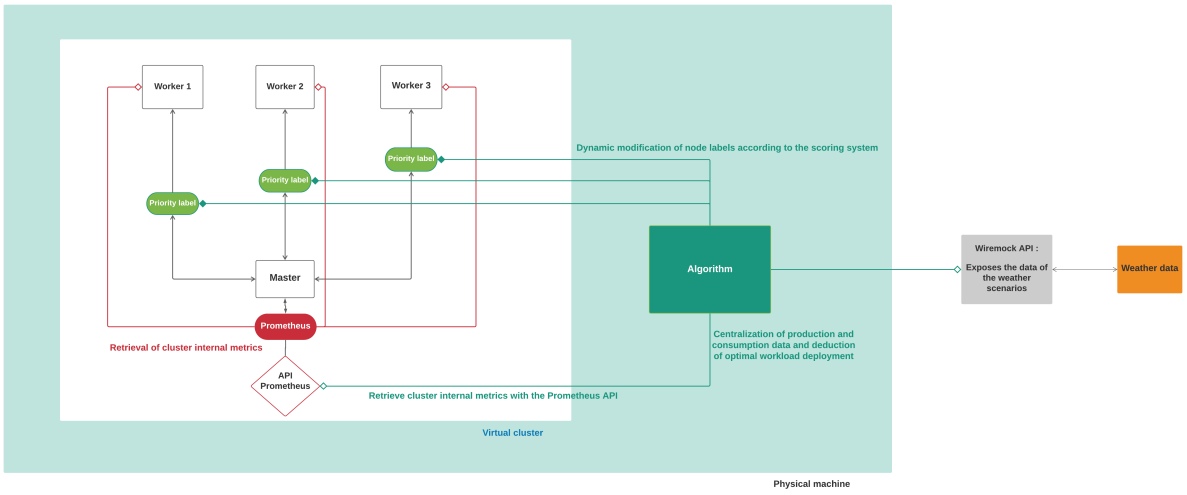

Figure 8: Solution diagram

3.3 Results

The performance and results of the algorithm are observed in two ways:

- Using the `Grafana` dashboards in real time
- Plotting relevant metrics after running the program

Weather scenarios are created in order to visualize the behavior of the cluster when the algorithm is running. In this example [Figure 9], the weather scenario is described by the following weather data:

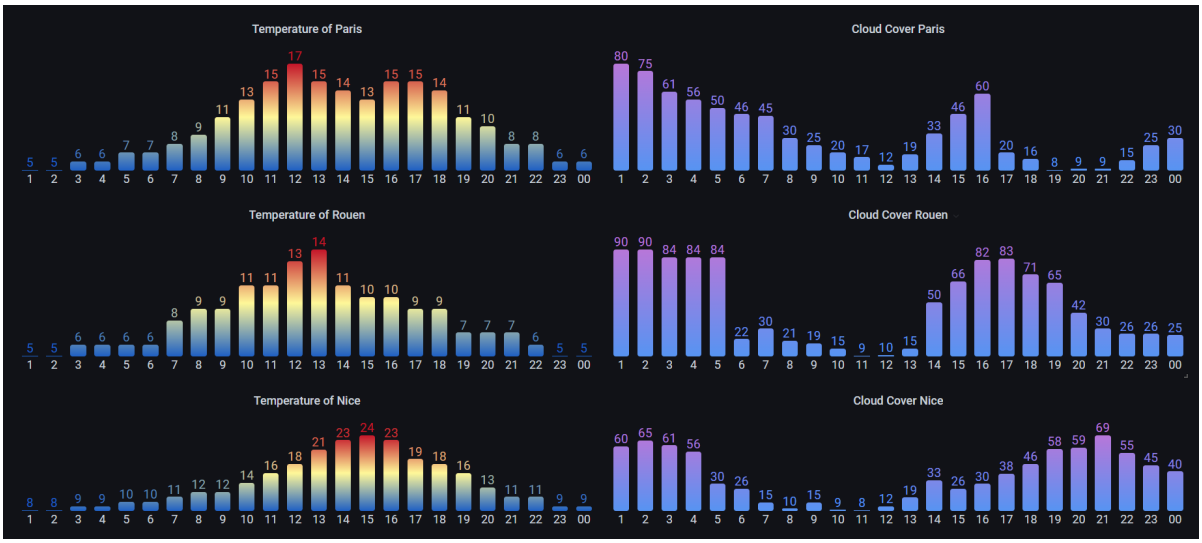

Figure 9: Grafana dashboard - Temperature and cloud cover on each datacenter

3.3.1 Grafana

`Grafana` allows visual representations of internal cluster metrics in real time. The algorithm aims to manipulate intelligently workloads between data centers. This helps to optimize the energy consumption in the data centers. Therefore, it is relevant to observe the distribution of workloads in real time across the data centers. For this purpose, a Grafana plugin that allows the visualization of Kubernetes data was installed.

The [Figure 10] shows a `Grafana` dashboard that represents the distribution of workloads across the three data centers in real time.

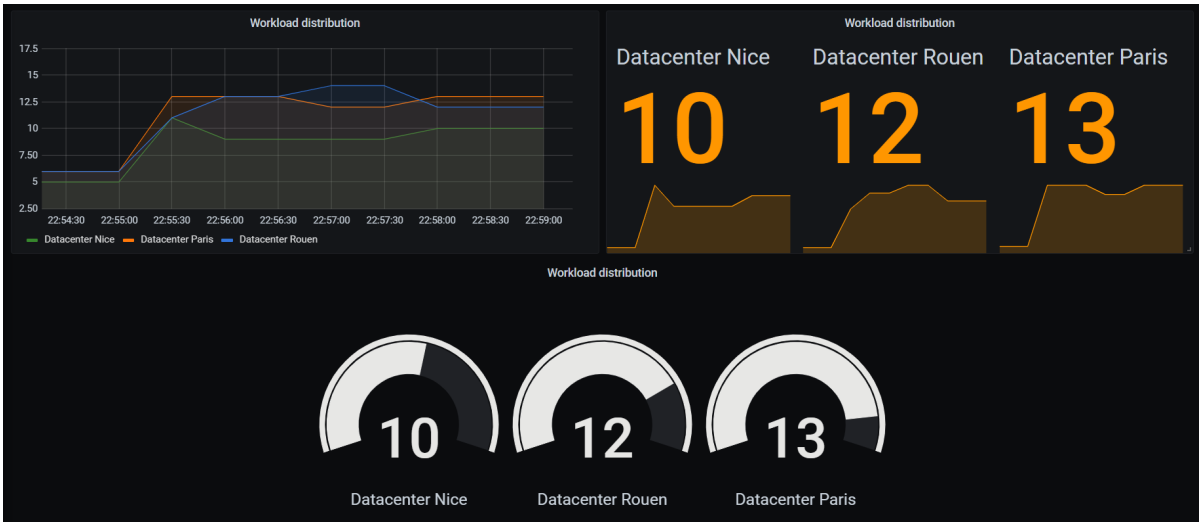

Figure 10: Grafana dashboard - workload distribution between datacenters

3.3.2 Graphs

In order to visualize how the algorithm behaves and see if it performs well, there are two intersting metrics to observe:

- The evolution of the renewable energy available on each datacenter
- The evolution of the number of workloads running on each datacenter

This scenario simulates strong sunshine in Nice, average sunshine in Paris with a rainfall around 3 pm and weak sunshine throughout the day in Rouen. The results obtained with this scenario are illustrated below [Figure 10]:

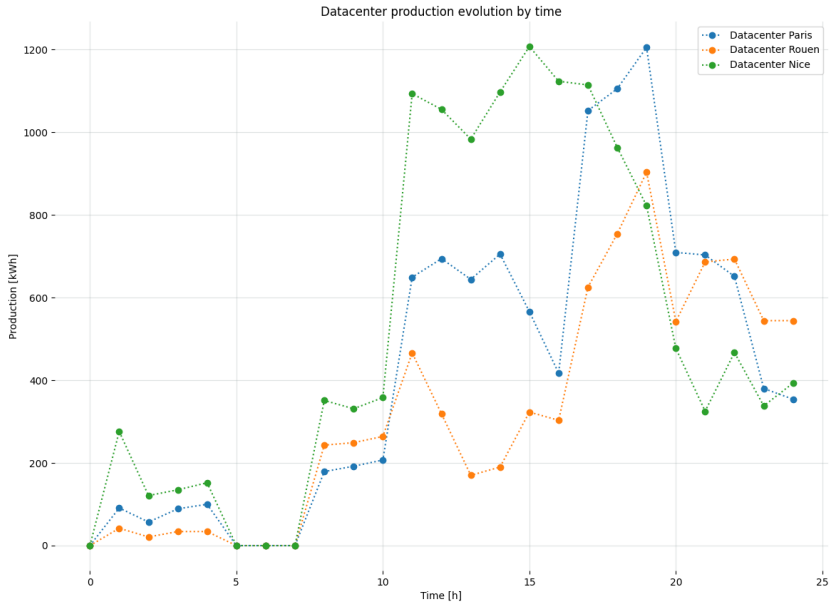

Figure 11: evolution of solar energy production by time

[Figure 11] shows the solar energy output of each data center per hour. This production is directly linked to the chosen scenario: the evolution of the production in Nice is a Gaussian curve with a peak around 3 pm. In Rouen, the production increases very gently because of the low sunshine. And in Paris, the rainfall at around 3 pm makes the curve decreases sharply. The evolution of the quantity of renewable energy is illustrated below [Figure 12]:

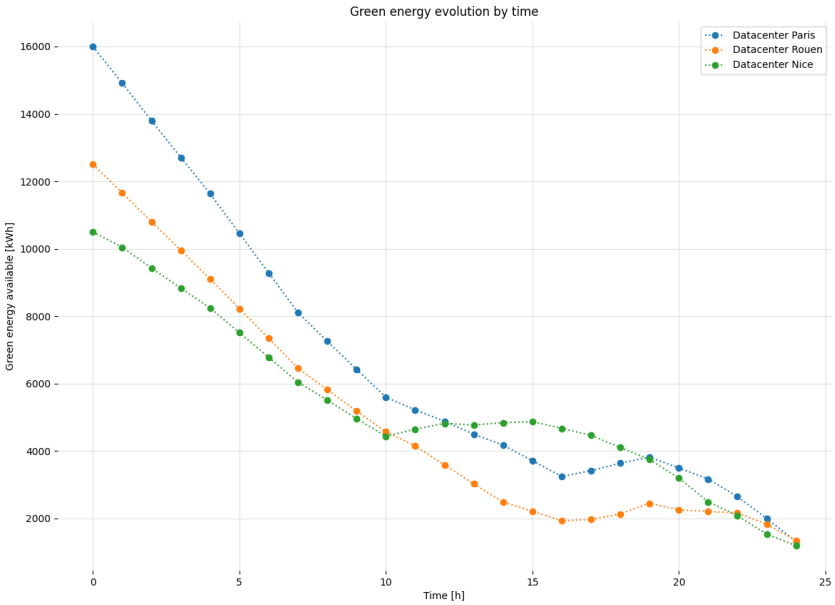

Figure 12: Evolution of the green energy available by time

The amount of renewable energy available is directly related to the production data. This is why when the sun rises around 6 am, the amount of energy decreases less strongly. This phenomenon is also observed when the sun become strong in Nice from 10 am.

The behavior of the cluster is driven by the deployed algorithm: when a data center has a high renewable energy quantity, it is assigned the most workloads. This phenomenon can be observed from 3pm to 8pm when the data center in Nice has the highest amount of green energy among all the data centers. During this period, workloads are moved from the other two data centers to the data center in Nice. This behavior has the effect of increasing the consumption in the data center in Nice and reducing that in the data centers in Rouen and Paris.

4 Project management

Project management is an essential part for a scientific project. Particularly when it is a work with a group of seven.

4.1 Agile method

The agile methodology was chosen in conformity with the method used by Orange engineers. This methodology allows more flexibility and transparency between the managers and the team.

The project was divided into two weeks segment called sprint. For each sprint, some user stories were defined and the end-of-sprint objectives were represented. Then the sprint ended with a demonstration and a sprint retrospective. The end-of-sprint retrospective highlighted the difficulties encountered in both the technical and management aspects.

All the user stories were set up on `Github Project` in order to facilitate project management. Then weekly tasks were distributed on `Discord` and the progress of each sprint was monitored by a tracking table. A rigorous technical bibliography has been written for each important handling or installation in order to facilitate the work and avoid duplication.

Conclusion

The project was based on the search for an innovative solution which led to a considerable quantity of thinkings, researches and tests. Since `Kubernetes` is a new technology and it has not been often used yet in the industry, researches and innovations were the central aspect of the project. The project was initially difficult to grasp, but it allowed the team to familiarise with high demand technologies. Therefore, one significant part of the project was dedicated to develop documentation and learn technologies. An essential step was then to create a `Kubernetes` cluster and to supervise it using `Prometheus`. When `Raspberry Pi` showed its limitations, the team needed to move forward in order to find a new solution.

Agile method made it easier to decide to stop using `Raspberry Pis` and to use a virtual cluster instead. Once the `Kubernetes Cluster` and `Prometheus` were up and running, the development of the final attempt was based on a Python algorithm. This program centralized the useful information of the cluster retrieved by Prometheus and the green energy production information given by `Wiremock` in order to label dynamically the nodes.