Tools for graphical visualization of the energy consumption of a VM

Philippe Roose
Computer science researcher
Anglet, France
Philippe.Roose@iutbayonne.univ-pau.fr

Adel Noureddine
Computer science researcher
Anglet, France
adel.noureddine@univ-pau.fr

Olivier Le Goaer
Computer science researcher
Pau, France
olivier.legoaer@univ-pau.fr

Slim Khiari
Computer science student
Vélizy-Villacoublay, France
slim.khiari@ens.uvsq.fr

Abstract- **Today, the energy management of enterprise IT systems requires the management of heterogeneous and distributed devices. In the past, the IT infrastructure was limited to computers connected by ethernet, and a few local servers. From now on, the computer park has diversified with mobile devices such as laptops and smartphones, internet terminals such as WiFi and routers, local and remote servers such as the cloud, as well as various connected objects. The objective of this article is to highlight two tools for measuring and visualizing the energy consumption of a virtual machine in order to easily manage its energy.**

Keywords: Scaphandre API, Prometheus, energy consumption

INTRODUCTION

The general theme of this article is green IT. It is a set of techniques aimed at limiting the environmental consequences of information and communication technologies. Green IT appeared when the IT infrastructure became one of the causes of the increase in the greenhouse effect due to its relatively high consumption of electrical energy. This article highlights the combination of two fairly effective tools for collecting and then graphically displaying the energy consumption metrics of a virtual machine. The tool used to collect energy consumption data is an API, called Scaphandre, and the data visualization tool in the form of graphs for setting up a final dashboard summarizing, by real time, the data collected for a data virtual machine, is called Grafana. Grafana is part of an ecosystem called Prometheus. The originality of this article is the detailed explanation, the implementation and the display of the result in the form of a dashboard of the combination of the two tools the Scaphandre API and the Prometheus ecosystem. Also, it is important to note here that our data without real data of the energy consumption of our VM. This data will therefore be represented in the form of meaningful metaphors on our final dashboard.

To do this, we will first understand how the Scaphandre API and the Prometheus ecosystem work. Next, we will look at the implementation of these two technologies. Finally, we will see the final result that we obtained thanks to these tools.

BACKGROUND

Scaphandre API

Scaphandre is an API aimed at measuring the energy consumption metrics of technological services. It provides the energy consumed by a single process on a server or virtual machine. In order to easily understand the calculation of the amount of energy consumed by a single process executed on a machine (server or virtual machine), we can imagine a rectangle containing lines each representing the calculation time allocated to each process. For machines working on different processes at the same time (i.e. working on one process for a short interval of time and then another time interval and so on), we call these intervals the jiffies. Each process keeps a running total of the total number of jiffies allocated to it. Thus, to know the quantity of resources of a machine used by a given process, it suffices to know the total number of jiffies used. In order to know the power used by a given process, we first count the jiffies used by this process when it is running. Then, for each jiffy, we check the amount of power consumed at those specific times. By aggregating all the power readings for all the jiffies over a specific time frame, we can arrive at a usable figure for the amount of power used in terms of watt hours.

Without deploying Scaphandre, or an API of the same type, it is of course possible to know the most important part of the resources of a machine in terms of use by a given process. On the other hand, if we want to find the amount of energy used by a process, we must necessarily know how much energy is used by the machine. Energy consumption information is extracted using RAPL technology. It is a technology integrated in Intel processors and AMD processors, having the x86 architecture, produced after the year 2012. The powercap module, located between Scaphandre and this energy consumption data, writes the energy consumption in files, which will then be read by Scaphandre. After being read, Scaphandre API stores this data in buffers, and therefore allows further processing through the various exporters.

The Scaphandre API is divided into two main parts; a sensor and an exporter.

The 1st part, the sensor, is therefore intended to obtain the energy consumption metrics of the host, and make them available to the exporter. For example, PowercapRAPL obtains and transforms metrics from the powercap Linux kernel, which serves as an interface to obtain RAPL function data from x86 processors. The PowercapRAPL sensor first collects energy consumption measurements and then converts them into energy consumption measurements. Each time the exporter, like the prometheus exporter, requests a measurement, for example each time a request arrives, which we call request "i", PowercapRAPL reads the values of the energy meters from powercap. Then this sensor stores them and performs the same operation for CPU usage statistics and for each process running on the machine in real time. Now, between 2 measurement requests, request "i" and request " i+1", we have the possibility to obtain the energy consumption subset related to the PID of a process. So to find out what a service actually consumes, simply join the consumption of all the associated PIDs. It is important to note that this functionality is not available directly for virtual machines. In this case, with the QEMU exporter, we will first have to run Scaphandre on the hypervisor, and then make the VM metrics available.

Regarding the 2nd part, the exporter is intended to request the sensors to obtain new measurements and store them for possible later use. The exporter therefore allows you to export the current metrics. For example, the prometheus exporter exposes metrics on an HTTP endpoint, to be retrieved by a prometheus instance. Then the stdout exporter just exposes the metrics to standard output. Regarding the qemu exporter, it is intended to collect metrics related to the execution of virtual machines on a Qemu/KVM hypervisor. These metrics will be made available to each virtual machine by running the PowercapRAPL sensor with the –vm option. The Qemu

exporter puts virtual machine metrics into files the same way the powercap kernel module does. He imitates this behavior so that the sensor can act the same as it would on a non-virtual machine.

Prometheus ecosystem

Prometheus is a free software application created in 2012 by SoundCloud. Since 2016, this application has been part of the Cloud Native Computing Foundation. It is used to monitor any purely digital time series by recording and processing them. It gathers, organizes, and stores metrics from infrastructure platforms, applications, and services, including HTTP endpoint metrics. Prometheus can also collect his own health metrics and monitor them. This ecosystem is therefore a solution that encompasses both the management of a multidimensional data model and the collection of scalable metrics while maintaining a certain operational simplicity.

It is developed with the Go programming language and has its own query language, promQL, which will facilitate the manipulation and analysis of the collected data. Prometheus is an ecosystem with several components working together for the purpose of generating reports on the performance of a system. Thus, Prometheus collects and stores metrics from applications that expose metrics in a plain text format through HTTP endpoints.
The diagrams below are complementary. They allow us to better understand the architecture of the prometheus ecosystem that we used to build our dashboard.

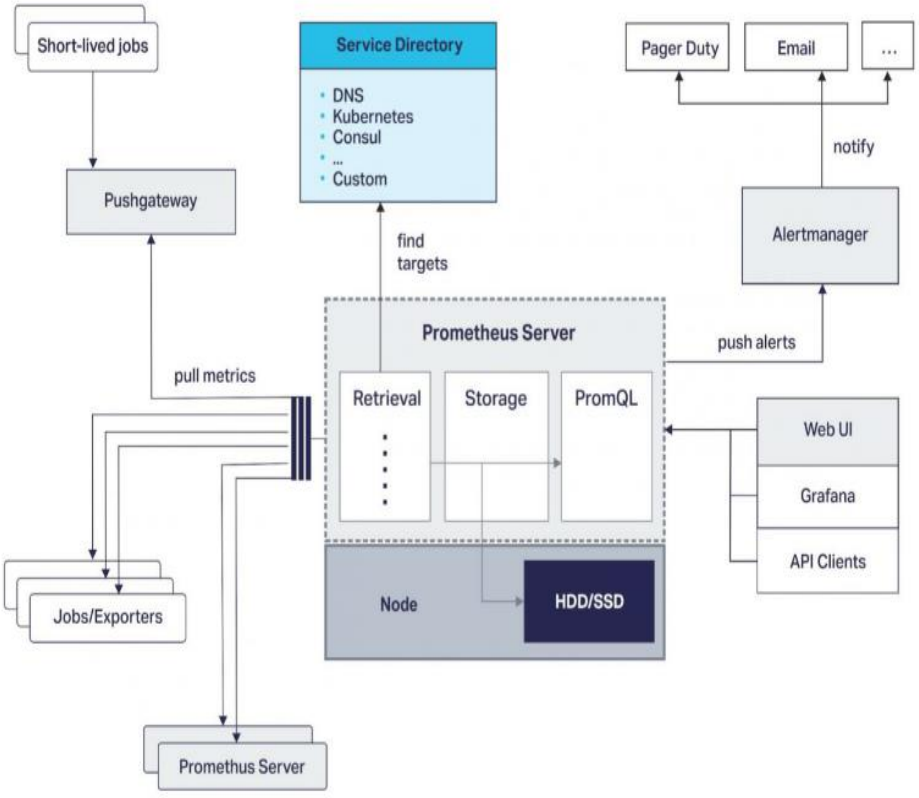


Figure 1 - Diagram of the Prometheus ecosystem

The first component of this ecosystem is the Prometheus server. He takes care of both storage of metrics, and planning of monitoring tasks by querying the data sources at a predefined polling frequency. Monitoring tasks are managed through the YAML configuration file, and configured using the directive “scrape config”.

In the diagram above we have an element that is in the server Prometheus who is called SD. Indeed, in order to know the targets from which the collection data will be made, Prometheus relies on several SD mechanisms. For instance, we can have file based SD than custom implementations SD can use. This is therefore possible by directly managing a YAML type file containing a whole list of targets. In addition, Prometheus also provides other SD implementations like Kubernetes or Amazon Elastic Compute Cloud.
The second component of this ecosystem is the Prometheus exporter. Exporters allow collecting metrics from a specific third-party system and rendering them by suite available for Prometheus servers to retrieve. The client libraries, used by applications, enable an HTTP endpoint where internal metrics are exposed and then collected by Prometheus servers.
The third component is the client libraries. In fact, applications cannot provide metrics only after adding instrumentation to their codes using the Prometheus client libraries directly.
The fourth component is Alertmanager. The main purpose of this element is to manage alerts sent by the Prometheus server by sending notifications through several means.
The fifth component is the pushgateway. This component allows you to retrieve the short-lived external services metrics. Finally, we have the Web UI component. It is a web application intended to view the result of a query in graph or table form, in real time promQL.
Our objective is therefore to recover the energy consumption of a VM using the 2 tools presented previously. This will allow us, in the end, to create an easily understandable dashboard through the different metaphors that we will insert. The main difficulty at this stage is to properly target the metaphor that we are going to use because of the order of magnitude of the value of the energy consumption received.

PROPOSED METHOD

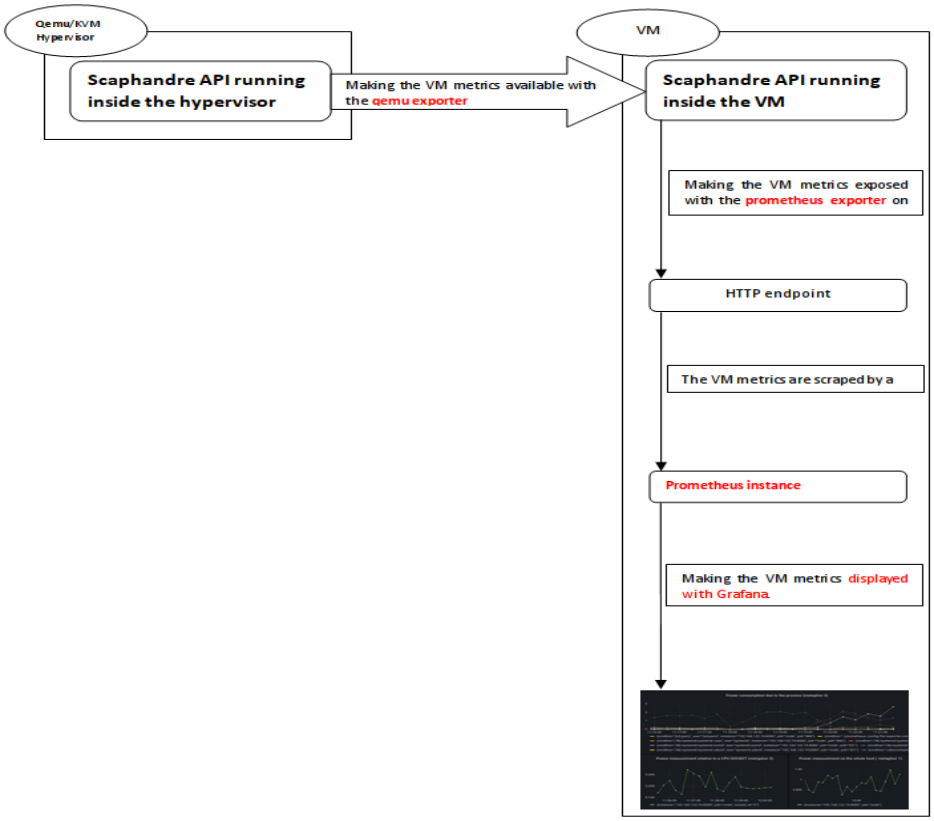


Figure 2 - Diagram summarizing our architecture

In the previous diagram, the sensor runs, by running scaphandre at first on the hypervisor, and making the VM metrics available with the qemu exporter to scaphandre running inside the VM. The main goals of the sensors are to get metrics of energy consumption from a machine and to make them available to an exporter. The qemu exporter is a specific exporter because it is only intended to collect metrics related to running virtual machines on a Qemu/KVM hypervisor. Now to display these metrics with the Grafana tool and retrieve them, we use the prometheus exporter. This exporter exposes the metrics on an HTTP endpoint in order to be scraped by a prometheus instance.
In this section, we will talk about the main steps in order to set up the tools presented previously.
Note that the major problem in measuring energy consumption is to
do inside a VM because in general the VM does not have access to the
power measurements. The advantage of Scaphandre is that it solves this problem by
allowing communication between a Scaphandre instance on the hypervisor and another
instance running on the VM. Thus, the Scaphandre agent on the hypervisor will calculate the
metrics for this VM and the one on the VM will subsequently access these metrics.

Installing a KVM Hypervisor

It is important to mention that this method only works on Qemu/KVM hypervisors. It will therefore be necessary to install a KVM hypervisor and configure a VM. After downloading the Linux Ubuntu .iso file, we therefore created a virtual machine. One of the difficulties encountered during this part is to realize the non-operation of the Scaphandre API with more than one vCPU. So we have configured a virtual machine with a single vCPU.

Launching the API on the hypervisor

After cloning the “Scaphandre” directory on the bare-metal machine and loading the "intel_rapl_common" module for measuring energy consumption under
Ubuntu, we created the binary file, which allows us to launch the Scaphandre API.
We launched Scaphandre with the exporter QEMU. Now we can give access to the vm created to its metrics measured from the hypervisor. For this we have
created a tmpfs mount point in order to mount the VM directory "/var/lib/libvirt/scaphandre/ubuntu22.04".

Launching the API on the VM

Regarding the VM, as for the hypervisor, we cloned the “Scaphandre” directory,
and we have created the binary file which allows us to launch the Scaphandre API. In the “var” directory, we have created a directory named “scaphandre”. Thus, we were able to launch the Scaphandre API on this VM by exporting the metrics with the Prometheus exporter.
Metrics are now collectable, in a non-graphical way, through the browser by directly typing the IP address of the VM followed by the port number 8080.
Once the metrics have been obtained, we now seek to display them graphically in real time.
For this, we need 3 important elements: the Scaphandre API working with the exporter Prometheus, in order to measure energy consumption metrics by real-time VM, Prometheus to collect them, and Grafana to visualize them graphically based on a data source connection to the server Prometheus.

Graphic display of metrics

After installing Prometheus, we created a configuration file to make our VM recognized by Prometheus, by adding in "targets" the IP address of the VM followed by the port number 8080 for metrics collected by the Scaphandre API. This will then allow the metrics of the energy consumption of the VM to be stored in the Prometheus server. After installing and launching Grafana, we are finally able to start building the dashboard with Grafana and view graphs of VM power consumption metrics. This

visualization tool is now accessible via port 3000. After defining the

Prometheus data source, and entered the IP address of the Prometheus server that is running on port 9090, we graphically obtain the metrics of the energy consumption of the VM provided by the Scaphandre API. This

visualization tool is now accessible via port 3000. After defining the Prometheus data source, and entered the IP address of the Prometheus server that is running on port 9090, we graphically obtain the metrics of the energy consumption of the VM provided by the Scaphandre API.

EVALUATION

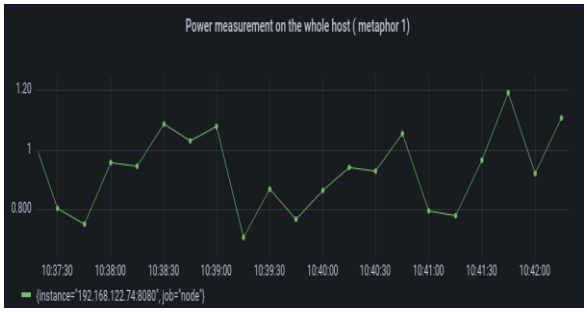
Before starting this part, it is important to note that we have performed energy consumption measurements for only 5 minutes for simplicity and efficiency. To set up a dashboard easily understandable by any user, we are going to set up metaphors to translate the energy consumption data obtained into more practical data. To better represent the conversion metaphors of energy consumption we used in our dashboard, we have made a small table, easy to understand;

Metaphor	**Device**	**Power consumption**
1	*Sony PS5 console in sleep mode*	1.3 W
2	A charger left plugged in without a phone (on average)	0.3 W
3	In standby mode most printers will draw around (on average)	4 W
4	Entering a phrase into its search engine and waiting for the results will cost about	0.3 Wh

The metaphors have been chosen in such a way that the difference in the conversion from Watt to a certain metaphor is not large in order to obtain results that are easy enough to understand for any user of the final dashboard. We have selected 3 main metrics thanks to the Scaphandre API.The first is "scaph_host_power_microwatts". This metric represents power measurement on the whole host, in microwatts. In order to simplify the understanding of this metric, we have chosen this metaphor; *Sony PS5 console in sleep mode : 1.3 W*. Here is the final PromQL query allowing us to insert this metaphor in this metric;

```
scaph_host_power_microwatts{instance="192.168.122.74:8080"}/(1000000*1.3)
```

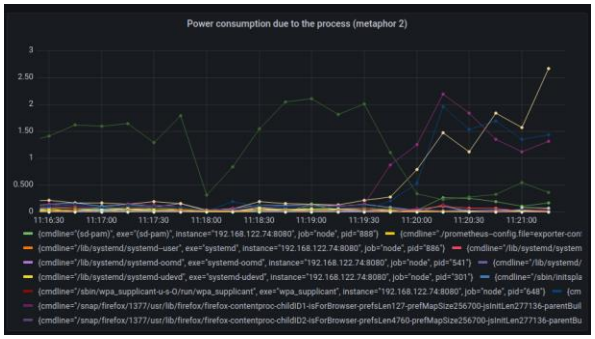
We obviously divided the result of the query by 1000000 in order to convert the result into Watt. Here is the real-time graph resulting from this query;



The same work is done but with 2 other metrics and 2 other metaphors. The second metric chosen is "scaph process power consumption microwatts". This is the power consumption due to the process in microwatts. In order to better visualize this metric, we used this metaphor; "A charger left plugged in without a phone (on average): 0.3 W".
Here is the promQL query for this metric;

```
(scaph_process_power_consumption_microwatts{instance="192.168.122.74:8080"}/
1000000)/0.3
```

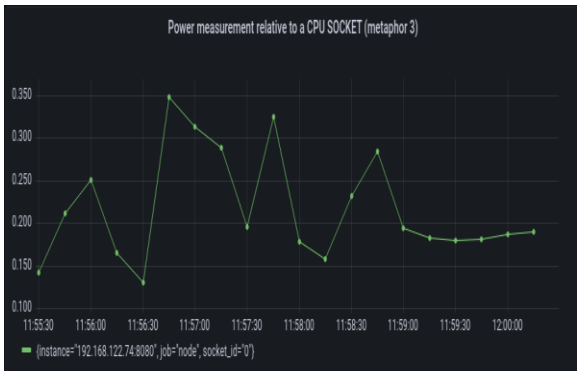
And here is the corresponding graph;



Note that each graph in the previous figure represents a particular process. We can also compare this graph with metaphor 4 (see table of metaphors above). Indeed, metaphor 2 and 4 are similar in terms of power consumption. The 2 are worth 0.3 W.

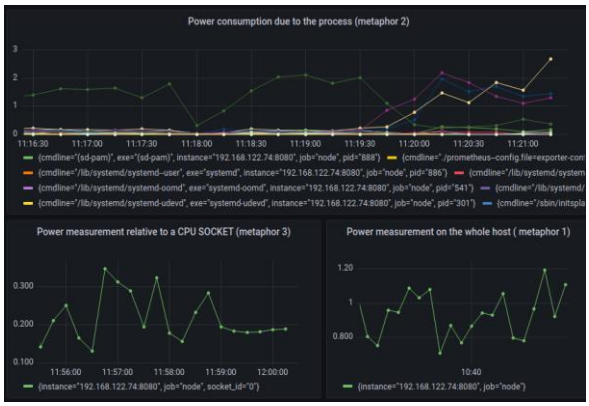
Finally, let's move on to the third metric which is "scaph_socket_power_microwatts". This metric represents the power measurement relative to a CPU socket, in microwatts.For this metric, we chose this metaphor;"In standby mode most printers will draw around (on average): 4 W".

```
(scaph_socket_power_microwatts{instance="192.168.122.74:8080"}/1000000)/4
```



Finally, we notice, through the 3 previous graphs of the 3 metrics, that the energy consumption values are quite low. This is explained by the fact that there are not many programs running on our VM.

Here is our final dashboard;



Of course, the Scaphandre API provides other metrics. But the other metrics do not perfectly meet our needs for energy consumption research. Hence our focus on these 3 metrics.

CONCLUSION AND FUTURE WORK

In this work, a dashboard containing energy consumption metrics was produced. For this, we have proposed a set of 2 effective tools; measurement of energy consumption, and graphic display. This set of tools is composed of the Scaphandre API and the Prometheus ecosystem. However, despite the advantages of the Scaphandre API such as efficiency in terms of use, it has some shortcomings. Indeed, it does not make it possible to measure the energy consumption of a VM without going through the hypervisor. This sometimes poses difficulties, especially when we do not have direct access to the hypervisor. This point is therefore one of the areas on which the developers of this API can improve.

REFERENCES

https://hubblo-org.github.io/scaphandre-documentation/
https://www.devopsschool.com/blog/what-is-prometheus-and-how-it-works/
https://www.monpetitforfait.com/energie/aides/consommation-console-jeux
https://energyusecalculator.com/electricity_cellphone.htm
https://energyusecalculator.com/electricity_printer.htm
https://business.directenergy.com/blog/2017/november/powering-a-google-search