FedE-ator : A framework for energy consumption analysis of federated learning in distributed systems

Mai Huong Do, Millian Poquet, Georges da Costa

FedE-ator : A framework for energy consumption analysis of federated learning in distributed systems

Mai Huong Do, Millian Poquet, and Georges Da Costa

Université de Toulouse,
Laboratoire IRIT - Cr Rose Dieng-Kuntz
31400 Toulouse - France
huong.mai-do@irit.fr, millian.poquet@irit.fr, georges.da-costa@irit.fr

Résumé
The development of AI is driven by advanced Machine Learning (ML) models and large data sets. The computational demands of state-of-the-art machine learning models have surged by 300,000 times in recent years [7]. Federated Learning (FL), an ML technique, might also be computationally intensive, leading to significant energy consumption. This work focuses on developing FedE-ator, a framework designed to measure the actual energy consumption of FL processes in real-world distributed systems. As a first step in validating the framework's usability, we conducted an experiment on the CIFAR10 dataset, systematically varying FL settings (server strategies, client models) and hardware configurations (CPU frequencies), with multiple repetitions. The experiment was carried out on the Grid'5000/SLICES-FR platform.

Mots-clés : Energy Consumption, Federated Learning, Grid'5000, SLICES-FR.

1. Introduction

ML has developed to handle many complex problems and brings conveniences. However, the red alarm was raised for energy usage of ML tasks. There have been studies discussing the tradeoff between ML performance and energy consumption, but still minorities compared to studies focusing on performance [7]. FL is an ML setting where many clients collaboratively train a model under the orchestration of a central server while keeping the training data decentralized [5]. Although it does not place heavy computation on the server, FL distributes computation to clients and requires frequent updates from client models. The energy concern in FL becomes the total consumption across multiple clients and the communication between the clients and the server. Understanding energy consumption in FL is a logical step towards the sustainable advancement of this technology.
Energy measurement techniques can be broadly categorized into two main approaches :

— Hardware solutions : physical tools used to directly measure the total energy consumption of a device (*e.g.*, Wattmeters).
— Software solutions : rely on software applications that interface with system sensors to monitor energy usage without requiring external hardware (*e.g.*, MojitO/S[1], Perfetto[2], PowerAPI[3], Alumet[4])

1. https://gitlab.irit.fr/sepia-pub/mojitos
2. https://perfetto.dev/
3. https://powerapi.org/
4. https://alumet.dev/

While many tools exist for energy measurement, our focus is on the energy of the FL process. We aim to develop a specialized energy monitoring tool tailored for FL frameworks. Additionally, we seek to integrate both hardware and software approaches : software enables domain-specific energy insights (CPU, GPU, RAM, *etc.*), whereas hardware-based monitoring ensures accurate total energy measurement of entire device without being domain-specific (including all components such as processors, memory, main board, storage, fan, *etc.*). Combining these methods can achieve more precision and granularity in energy analysis for FL environments. Our contribution is developing a framework with the functions below :

— Measures energy consumption for FL in distributed systems (Grid'5000/SLICES-FR), including synchronization between different measurement methods.
— Allows configuring CPU/GPU frequencies (DVFS).
— Allows running multiple repetitions for each configuration to improve results reliability.

In Section 2 we provide a detailed description of the FedE-ator framework. We also conduct an experiment as a first validation step of the framework's usability (on a CPU machine) presented in Section 3. Finally, we conclude our work and future plan in Section 4.

2. Framework description

FedE-ator was developed for use in Grid'5000[5] platform. This platform is a large-scale and flexible testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing including Cloud, HPC and Big Data and AI.

2.1. Components and functions

The components and functions are presented below :

Configuration module (json) : configures a list of instances by users, each instance includes :

— Hardware (e.g., CPU, GPU specifications)
— FL (e.g., server/clients configuration, commands and IP)

Controller module : reads the **Configuration** and orchestrates the experiment. An experiment involves executing through each setting of all **instances**, and turn with a number of repetitions. Each **instance** can be executed at different frequencies as defined, a process referred to as **running**. Each **running** consists of the following sub-processes :

— FL process : initiated by the **FL module**.
— Monitoring processes : initiated by **Monitoring module**, continuously monitoring the FL process from begin to finish.

FL module : uses **oarsh** in Grid'5000 to go through nodes assigned as servers and clients and active the FL commands, records log to output directory.

Monitoring module : uses Expetator[6], a tool for running HPC benchmarks in Grid'5000 and supports both MojitO/S and Kwollect[7] - two energy measurement components. MojitO/S monitor processor and memory power consumption (using RAPL), system and network values, and is deployed on each node while Kwollect collects power consumption data from the Grid'5000 wattmeters.

2.2. Energy consumption measurement

We used both software profiling and hardware-based solutions to measure energy. Two measurement solution are synchronized by timestamps, from begin to finish of FL process.

Figure 1 provides details of the energy monitoring module, including all measurement methods within Kwollect and MojitO/S.

5. https://www.grid5000.fr/
6. https://expetator.readthedocs.io/
7. https://www.grid5000.fr/w/Monitoring_Using_Kwollect

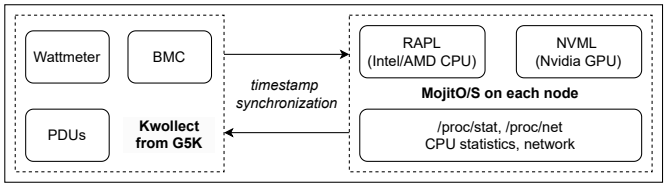

FIGURE 1 – Energy monitoring module

2.2.1. Software profiling solution : MojitO/S

MojitO/S is an energy and network monitoring software at the O/S level, runs on GNU/Linux [2]. The unit of energy is in micro-Joules and the network packet size is in bytes. MojitO/S monitoring processes are deployed on all nodes and launched using the **mpirun** command from the Open MPI library [8]. MojitO/S supports extracting performance information from **/proc** directory, Running Average Power Limit (**RAPL**) and NVIDIA Management Library (**NVML**).

- **/proc/stat**, holds various pieces of information about the kernel activity and is available on every Linux system. From this file we get CPU usage statistics of Linux Kernel (*e.g.*, idle : time spent in vacations twiddling thumbs).
- **/proc/net**, provides a comprehensive look at various networking parameters and statistics. It describes aspects of the system's network configuration. MojitO/S allows automatic detecting the type of network configuration and gets network information (*e.g.*, number/size of packets received/transmitted).
- **RAPL**, is a feature which allows monitoring energy consumption across different domains of the CPU chip, attached DRAM and on-chip GPU [4]. This feature is integrated in modern x86 CPUs from Intel and AMD.
- **NVML**, is a C-based programmatic interface for monitoring and managing various states within NVIDIA GPUs (*i.e.*, dram, GPU core), is also the underlying library for the NVIDIA-supported nvidia-smi tool [9].

2.2.2. Hardware-based solution : Kwollect

On Grid'5000, energy consumption is measured using plug-in devices such as Wattmeters, Board Management Controllers (BMCs), and Power Distribution Units (PDUs). The recorded power values are expressed in Watts. Kwollect stores the measured data, and Expetator collects it through the Grid'5000 API.

2.3. Configuration functions

2.3.1. DVFS

Our framework includes a DVFS adjustment function, allowing control over CPU/GPU frequencies to analyze their impact on FL performance and energy consumption. Before executing each instance, **Monitoring module** sets the frequency of all CPUs/GPUs as specified in **Configuration**. After execution (when the FL process is complete), it restores the frequencies to their default values.

This function detects configured frequencies and go through all of them. It supports by 3 options in configuration :

- $f = f_{max}$: Maximum performance mode (set frequency at maximum and start **running**).
- $f = \{f_{min}, f_{max}\}$: Minimum and maximum performance modes (set the frequency to the minimum and start the **running**, then set it to the maximum and run it again)
- $f = list[f_i]$: Custom frequencies selection (set the frequency in each value in selection

8. https://www.open-mpi.org/
9. https://docs.nvidia.com/deploy/nvml-api/index.html

and start the **running**, change to next value and repeat).
This function enables frequency tuning within the system's available configurations.

2.3.2. Multiple instances and repetition

One **configuration** includes one or multiple **instances** with different setting of FL and hardware (DVFS). Each **instance** is identified by ID, name and output folder for easy results management. This framework also allows setting the number of repetition, enables multiple runs within a single experiment without needing to restart separate experiments.

3. Experiment

As a first validation step of the framework's usability, we experimented with the CIFAR10 dataset, establishing 4 FL settings (2 server strategies, 2 local models of clients) and 4 CPU frequencies, with each instance running in 2 repetitions. Our experiment was designed with reproducibility in mind, the code and data of our work are available here [3].

3.1. Experimental setup

3.1.1. Hardware infrastructure

We executed all experiments on 4 machines (1 server, 3 clients) from the **taurus** cluster on the Grid'5000 testbed. These clusters were selected because it contain Intel CPUs with an feature of RAPL and the external power meters. Detailed specifications of these machines are as follows :

- Manufacturing/arrival date : 2012-07-16/2012-09-14.
- CPU : Intel Xeon E5-2630 (Sandy Bridge), x86_64, 2.30GHz, 2 CPUs/node, 6 cores/CPU.
- Memory : 32 GB.
- Storage : 299 GB HDD RAID-0 Dell PERC H710.

3.1.2. Software dependencies

The FedE-ator framework requires Python v3.9 or higher and Expetator v0.35. Our experiments utilize Flower v1.13, a FL framework. The required dependencies for running example benchmark include flwr-dataset, TensorFlow, NumPy, and Scikit-learn.

3.1.3. Benchmark

FL setting : A FL strategy is a method used to combine ML models from multiple clients into a central server. In our work, we use the Flower framework [1] and implement 2 FL strategies : FedAvg [6] and FedAvgClients. FedAvg is a common approach that averages model updates from clients, while FedAvgClients is a variant where only 67% of clients are selected in each training round.
On each client device, training occurs in multiple small steps called epochs, which represent how many times the model goes through the local dataset. For each round of FL training, a client trains its model for a number of epochs, chosen from the set $1, 2$. Other details are provided in Table 1. Others are shown in Table 1.
Hardware setting : based on the hardware conditions of **Taurus**, the list of chosen CPU frequencies in this experiment is [120000, 160000, 200000, 230000], which includes f_{min}, f_{max}, and two available frequencies in between.

3.2. Results analysis

We provide the first version of the results analysis. Figure 3 visualizes the performance and resource usage of different instances at 4 frequencies. The subplots analyze relationship between CPU frequencies and various performance (accuracy, execution time) and energy metrics (mean power, energy consumption and CO_2). The carbon intensity in France (31 gCO_2eq/kWh), which is measured in grams of CO_2 equivalent per kilowatt-hour, is taken from the Electricity

ML task	image classification
Model	MobileNetV2, batch size=32
Stopping round	15
Dataset	CIFAR10 (32×32 RGB images of vehicles and animals, 10 classes)
Data distributed	train 80%, test 20%
Data partition	dividing equally by number of clients from full dataset (60000 samples / number of clients)
Number of server	1 (taurus-1)
Number of clients	3 (taurus-8, taurus-9, taurus-13)

TABLE 1 – FL setting.

Map website[10]. The accuracy values shown are not intended for comparison but to illustrate the ability of experiment (we do not set random seeds in the ML training process, the results are not statistically stable or reproducible). The trends of the energy measured by both methods (Kwollect and MojitO/S) are consistent. However, the energy consumption measured by Kwollect are higher than those from MojitO/S because Kwollect measures the entire device, including all components, whereas MojitO/S only count the energy of package (CPU + iGPU), dram (RAM).

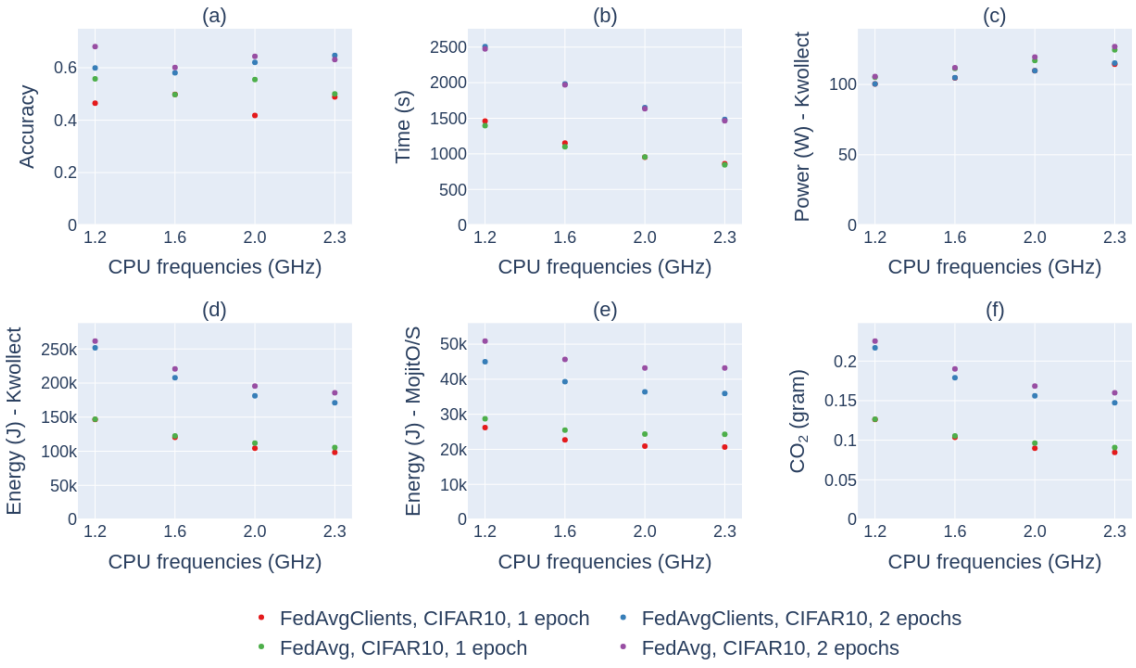

FIGURE 2 – FL performance and resource consumption at different CPU frequency settings
(a) Accuracy after 15 rounds, (b) Processing time of FL, (c) Mean power consumption of entire devices, measured by Kwollect, (d) Total energy consumption of entire devices, calculated by multiplying Mean power (Kwollect) by Processing time, (e) Total energy consumption of 2 domains (package + dram), measured by MojitO/S, (f) CO_2 emission, calculated by multiplying Energy consumption of entire devices by carbon intensity in France (31 gCO_2eq/kWh) (each point in the figure is the mean values of 2 repetitions).

10. https://app.electricitymaps.com/zone/FR/72h/hourly

4. Conclusion

This work introduced FedE-ator, a energy measuring framework for FL process in distributed computing systems (Grid'5000). Our framework allows both monitoring of the power of each hardware domain (using RAPL, NVML, /proc) and also synchronization with the measurements of the entire device (using Wattmeter, *etc.*). It provides a DVFS feature, allowing users to conduct experiments in a controlled environment. By allowing for multiple repetitions, it improves the reliability and meaningfulness of the results. We conducted an experiment on CPU nodes in Grid'5000 to initially validate the functional capabilities of our framework. In the future, we plan to test it on GPU nodes and apply it to well-configured FL setups to evaluate its performance more comprehensively. Additionally, we aim to ensure that the framework is fully reproducible, allowing others to replicate our experiments and results in their own environments.

Data Availability

Code, data and documentation of our work is available here [3].

Carbon footprint

According to Grid'5000 logs, this work used 1152 node.hours during the development of FedE-ator. Assuming a power consumption of 150 W for the core and an additional 150 W for the rest (memory, network, disk, cooling, *etc.*), the total electricity footprint of our Grid'5000 usage amounts to 345.6 kWh. Based on the carbon intensity value in France, this corresponds to an emission of 10.7136 kg CO_2.

Acknowledgment

This work was supported by the French National Research Agency (ANR) project Advancing Federated LearnIng while Reducing tHe Carbon Footprint – DELIGHT (ANR-22-CE23-0024). Experiment presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see `https://www.grid5000.fr`).

Bibliographie

1. Beutel (D. J.), Topal (T.), Mathur (A.), Qiu (X.), Fernandez-Marques (J.), Gao (Y.), Sani (L.), Li (K. H.), Parcollet (T.), de Gusmão (P. P. B.) et al. – Flower : A friendly federated learning research framework. *arXiv preprint arXiv :2007.14390*, 2020.
2. da Costa (G.). – MojitO/S, novembre 2021.
3. Hidden for the peer-review process.
4. Intel (I.). – and ia-32 architectures software developer's manual. *Volume 3A : System Programming Guide, Part*, vol. 1, n64, 64, p. 64.
5. Kairouz (P.), McMahan (H. B.), Avent (B.), Bellet (A.), Bennis (M.), Bhagoji (A. N.), Bonawitz (K.), Charles (Z.), Cormode (G.), Cummings (R.) et al. – Advances and open problems in federated learning. *Foundations and trends® in machine learning*, vol. 14, n1–2, 2021, pp. 1–210.
6. Reddi (S.), Charles (Z.), Zaheer (M.), Garrett (Z.), Rush (K.), Konečnỳ (J.), Kumar (S.) et McMahan (H. B.). – Adaptive federated optimization. *arXiv preprint arXiv :2003.00295*, 2020.
7. Schwartz (R.), Dodge (J.), Smith (N. A.) et Etzioni (O.). – Green ai. *Communications of the ACM*, vol. 63, n12, 2020, pp. 54–63.