PROCEEDINGS OF SCIENCE

Estimating energy consumption and carbon costs of GEN-SIM, DIGI and RECO jobs at LHC

Francesco Minarini[a]

[a] *University of Bologna - Department of Physics and Astronomy - DIFA , Viale Berti Pichat 6/2, Bologna, Italy*

E-mail: francesco.minarini3@unibo.it

As computing becomes substantial for achieving scientific and social progress, its environmental implications often remain underestimated. While the value of scientific computing is witnessed by its ubiquitous achievements, its growing demands have lead, in turn, to increased energy and carbon footprint costs. With the goal of describing such computational trace in subnuclear physics (SNP), this work estimates the energy consumption of benchmark SNP workloads with a containerized original monitoring software. The benchmark workloads used in this work are GEN-SIM, DIGI and RECO containerized jobs deployed by the HEPScore project. The monitoring software extracts the CPU and RAM usage of such jobs in real-time via process IDs and estimates, with this information, their energy (kWh) and carbon utilization (gCO2e). The results can be used as a starting point towards a "greener" approach to computing methods and integrate current benchmarking scores with energy efficiency-related metrics.

42nd International Conference on High Energy Physics (ICHEP2024)
18-24 July 2024
Prague, Czech Republic

PoS(ICHEP2024)1223

1. Introduction

Climate change poses a brand new challenge to the scientific community as a whole and, as a part of it, particle physicists should think about and optimize their energy footprint. In particle physics research, a great part of the environmental impact of the activities is actually related to the high-energy physics experiments's operations [2], with computing being a notable case. Given that this latter discipline is increasingly shaping the scientific and industrial world, several works are starting to pay attention to its implicit footprint, as its scale might quickly become too unsustainable [5], [3]. A data-driven perspective is needed in order to timely curb it.

2. Methods and Setup

To understand the footprint of Scientific Computing, it is necessary to first define the concept of energy footprint and develop a strategy to estimate it. In this work, the following definition is adopted:

Def 1. *(Energy Footprint) The energy footprint is defined as the energy, in kWh or kWs, consumed by a computing activity A running on a computing resource X.*

In order to estimate it, [6] proposes a model to evaluate such defined energy footprint:

$$E_{A\to X} = T \times PUE \times (n_c \times P_c \times u_c + n_m \times P_m), \tag{1}$$

where T is the elapsed computing time (in h), PUE is the Power Usage Effectiveness, n_c is the number of allocated CPU cores, P_c is the power draw of those cores (extrapolated from the CPU TDP as suggested in [6], [1]), u_c is the core usage percentage, n_m is the allocated RAM memory and P_m is the power draw of RAM (extrapolated from the RAM TDP).

2.1 Populating Eq.1

The duration T of a computing activity can be measured with built-in function such as the `time()` function in C++. The PUE is a metric of the computing infrastructure and can usually be obtained contacting system administrators. n_c is typically a user request and therefore is known by construction (or can be easily retrieved parsing job requests). P_c and P_m are "bare-metal" properties that are retrievable from vendor websites. u_c and n_m need to be evaluated in real-time: to do so it is possible to leverage the properties of linux systems.
The `/proc` folder contains informations about all processes running on a given machine. In `proc` each process has a PID-named sub-folder. In this sub-folder it is possible to find two files: `stat` and `status`. u_c can be calculated as: $\frac{\texttt{utime+stime}}{\texttt{uptime}-\texttt{starttime}}$ where `utime` is the time a process spent in user-mode (14th field of `stat`), `stime` is the time a process spent in kernel-mode (15th field of `stat`), `uptime` is the amount of time passed since boot and `starttime` is the timestamp of process start.
n_m can be obtained from `status`, parsing the `vmrss` field, that accounts the memory mapped to the process (and therefore allocated).
The remaining data, which is related to the infrastructure, is gathered in a separate configuration file written in `.toml` language for human-readability. This file is parsed by the monitoring tool as soon as the activity starts.

2.2 Measurement strategy

The tool [7] reads the `.toml` configuration file and acquires the data related to the hardware properties. Using the aforementioned strategy, the monitoring tool recovers u_c and n_m and samples these values every t seconds (5 in this work). When the process has finished running, the PID is retired and this creates the lever to break the loop. Sampled data is stored in two separate data vectors and used to extrapolate the average value of u_c and n_m. To avoid potential permissions issues, especially while handling the `/proc` folder content, the tool was containerized via Docker. The following flow (Figure 1) depicts the underlying logic.

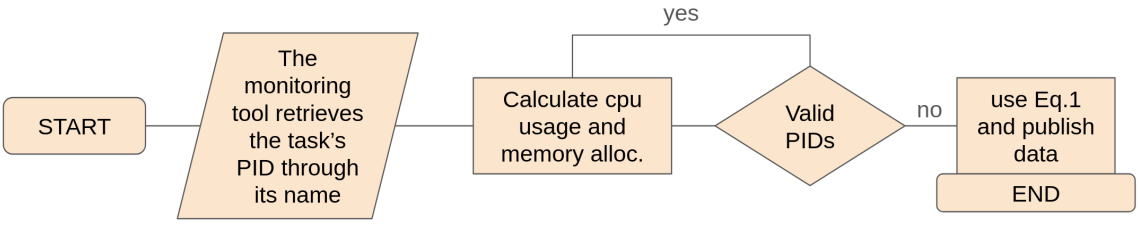

Figure 1: Pictorial representation of the logic flow of the monitoring tool.

2.3 Test application and setup

The test-bed for this study is represented by a Slurm node whose main specifications are showed in Table 1. The applications whose energy consumption was targeted are standard high-

Table 1: Slurm node used as a Test-bed for the application. The number of logical cores is 2x the number of physical ones due to Hyperthreading. (all trademarks are the property of their respective owners)

CPU	Physical Cores	Logical Cores	RAM(GB)
Intel Xeon E5-2640v2	16	32	128

energy physics workloads used by LHC experiments. In this work, we focused on one of the most CPU-intensive tasks of such kind: event generation and simulation (GENSIM in high-energy physics jargon). More precisely, we focused on the implementation of such workload made by the CMS experiment. A brief technical description of such application is hereafter reported:

GENSIM: generation and simulation of TTbar events at 14 TeV and 2021 CMS detector for the Run3 era. The workload executes a CMSSW GENSIM job, which embeds the event generation and the Geant4 simulation event by event. CMSSW is a multithreaded application; the default number of threads is 4 and the default number of copies is the number of cores divided by 4. [4]

3. Results

In the first data gathering experiment we focused on analyzing a pre-defined amount of events (10000) being executed over a growing number of computing cores (from 8 to 30). Small oscillations

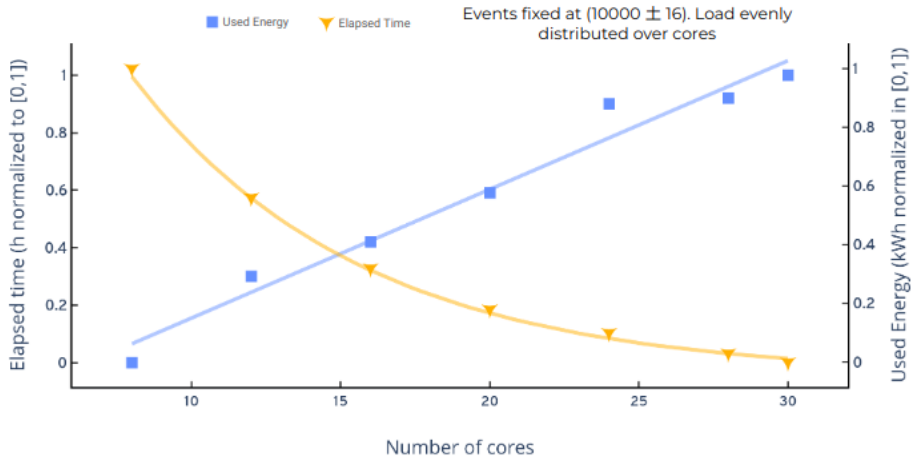

Figure 2: Ideal working point for CMS GENSIM over the Test node. The superposition of the two curves indicates that the best working point for this configuration of workload and computing node happens at ~ 15 cores.

in the number of events (±16) were allowed in order to equally split the computing load over cores. As showed in Figure 2, there seems to exist an optimal working point where performance gains (expressed in terms of elapsed time reduction) and energy consumption increments tend to balance.

We can use this information to attempt a more efficient job submission: instead of allocating all cores for a single copy of a GENSIM, we allocate 2 copies of this workflow on 14 cores each.[1] In this way it is possible to increase the overall throughput of events thus compensating for the increased running time. As it can be seen in Figure 3, this leads to a small gain in terms of energy efficiency per-event at the expense of a slightly slower overall execution.

4. Conclusions

The tool hereby presented represents a valid real-time measurement instrument for extracting a node-level energy consumption estimation of computing payloads. In this case, the focus was to highlight the energy consumption of standard high-energy physics workloads such as the event generation and simulation (GENSIM). The resulting data can be used to both acknowledge the footprint of physics computing (for instance, by including it in publications as supporting data) and to further investigate the margins of improvement of their performance. Figure 3, from an energy perspective, shows the chance of reconfiguring the job distribution and obtain a slightly improved energy efficiency. This indicates as well that many more improvements energy-wise might be highlighted and leveraged by the physics community in the future.

[1] the particular instance of Slurm allocated cores rounding up to the nearest physical core amount that satisfies the request (15 cores requested = 8 hyper-threaded cores fully allocated). The copies of the workflow would have filled the node without leaving space for the monitoring probe software. requesting 14 cores per copy (which allocates 7 hyper-threaded cores per copy) solved the problem.

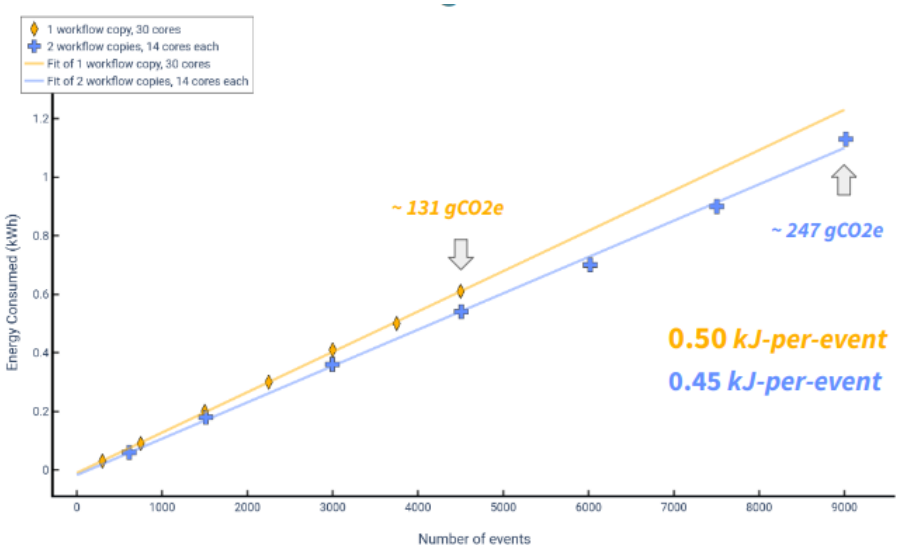

Figure 3: Energy consumption of 1 GENSIM on 30 cores (Orange) vs energy consumption of 2 GENSIM on 14 cores each (Blue). In this second case, data showes a smaller energy footprint. The carbon footprint was extracted by multiplying the energy footprint by the carbon intensity coefficient of Italy.

References

[1] Berthoud, F. et al. "Estimation de l'empreinte carbone d'une heure-coeur de calcul. Research Report", HAL Archives Ouvertes (2020).

[2] CERN. Cern environment report—rapport sur l'environnement 2021–2022.

[3] Freitag, C. et al. "The real climate and transformative impact of ict: a critique of estimates, trends, and regulations." Patterns 2 (2021).

[4] Giordano, D. et al. Hep-workloads (gitlab). https://gitlab.cern.ch/hep-benchmarks/hep-workloads. Last accessed July 2024.

[5] Jones, N. "How to stop data centres from gobbling up the world's electricity". Nature Vol. 561 (2018).

[6] Lannelongue, L. et al. "Green algorithms: Quantifying the carbon footprint of computation". Advanced Science 8 (2021).

[7] Minarini, F. Energy consumption monitoring tool. https://github.com/fminarini/green monitoring.

PoS(ICHEP2024)1223