PROCEEDINGS OF SCIENCE

KIG: a tool for Carbon footprint monitoring in Physics research

Francesco Minarini[a,b,*]

[a] *Department of Physics and Astronomy 'Augusto Righi', University of Bologna, V.le Carlo B. Pichat 6/2*
[b] *Istituto Nazionale di Fisica Nucleare, INFN*

E-mail: francesco.minarini3@unibo.it

Greenhouse gas (GHG) emissions have been recognized as accelerators of the global climate change phenomenon and several human activities take part in it. The contribution of the computing sector to the emissions is significant and deemed to grow in the near future. While on one side numerous relevant physics results have been obtained thanks to the increasing computational power available, on the other the heavy reliance on power-eager resources has started to lead scientific research to become energetically challenging and potentially cost-inefficient. In order to guarantee the overall sustainability of physics research, all the stakeholders, namely users and data centers, should be able to keep track - in addition to the currently adopted performance metrics- of the carbon footprint and energy-intensiveness associated to their operations. Through this reviewing activity, stakeholders can reach a deeper understanding of the burden related to their operations and take informed decisions to curb it without adding penalties on performance. For instance, users might plan energy optimizations of the workflow while data centers might adopt different management policies to abate the footprint of the facility. In this work, we introduce an open tool prototype, written in C++, that allows users and data centers to easily keep track, analyze and report the energy requirements and carbon footprint in gCO2e of their computing tasks. Such tool should help shedding some light on the often not-so-trivial trade-off between performance and environmental footprint. By gathering detailed data, such tool should also trigger meta-analyses on the behaviour of algorithms as well as computing infrastructures with the goal of better leveraging resources. Physics research related use-cases are discussed to present the tool.

International Symposium on Grids and Clouds (ISGC) 2023
BHSS. Academia Sinica
19-31 March 2023

*Speaker

PoS(ISGC&HEPiX2023)016

1. Introduction

The relationship between physics and computing has always been fruitful and intertwined. The increasing availability of high-performance technologies has allowed to implement powerful algorithms that pushed forward physics searches. Simulations and machine learning, for instance, have improved noticeably the design and optimization of detectors at the Large Hadron Collider (LHC) as well as and the signal/background discrimination on calorimeter data [1]. At the same time, scientific searches have systematically accelerated the evolution of computing paradigms by proposing computing models and requiring more powerful hardware to fuel searches. As an example of this, LHC analyses are made possible thanks to a dedicated computing infrastructure that was one of the very first successful examples of distributed computing. A massive distributed computing architecture (Worldwide LHC Computing Grid, WLCG) combining 1.4 million computer cores and 1.5 exabytes of storage from over 170 sites in 42 countries[1], connected with low-latency and high-performing networks, has been deployed giving particle physicists from all over the world the computational power they need to support their research.
So far, the evolution of technology has been able to keep up with the increasing computational demand thanks to the production of more powerful and energy-efficient resources. Three historical trends (but commonly referred to as "laws") indeed emerged:

Moore's law: Transistors in integrated circuits double every $\sim$ 2 years.

Dennard's scaling law: as transistors get smaller, the power density on chip stays constant. Power usage stays in proportion with the area of chip.

Koomey's law: Computations per Joule double approximately every 1.57 years.

Nowadays, these trends are considered to be weakening [2], mainly because manufacturers are reaching the physical limit of transistors that can be packed in a single chip without dealing with heavy bleeding currents or overheating issues. Meanwhile, the demand of computing resources is increasing both in the scientific sector and in the industrial environment, where several companies are starting to implement computationally intensive paradigms in their businesses. The clash between the hardware progress slow-down and the rising demand for high performance computing might lead to a worldwide spike in energy consumption and carbon footprint, if looked over [3, 4]. In a world where energy provisioning is already a problem due to historical events, it is imperative to secure future scientific results by quantitatively understanding the footprint of computing and set up alternative strategies to curb it in time. In the following, we will present the results obtained with KIG (Keep IT Green), a novel C++ tool prototype that can be used for measuring the footprint of computing activities. In **Section 2** we will discuss the methods implemented in the tool and the characteristics of the test-bed used to validate the output of measurements. In **Section 3** we will display the results obtained applying KIG to reference HEP workloads, namely GENSIM (Event GENeration and SIMulation), DIGI (Event DIGItization) and RECO (Event RECOnstruction).

[1] https://home.cern/science/computing/grid

2. Methods and Test-bed

In order to understand the footprint of scientific computing and plan strategies to curb it, we need first to formalize the concept of energy and carbon footprint and make explicit a methodology to measure it. In this section, the approach used to implement footprint measurements in KIG is described. Moreover, a brief technical description of the test-bed setup will be presented.

2.1 Methods

We consider these definitions for energy footprint and carbon footprint:

Definition 2.1 (**Energy footprint**). The energy footprint is defined as the energy, in kWh, consumed by a computing activity A (from start to finish) running on a computing resource X.

Definition 2.2 (**Carbon footprint**). The carbon footprint is defined as the cost, in gCO2e[5], related to the production of the energy consumed by a computing activity A running on a computing resource X in a region R.

Following [6], it is possible to calculate the energy footprint of some computing activity A running on a generic computing resource X with:

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

where T is the elapsed computing time (h), n_c is the number of cores used for computing, P_c is the power draw (in kW) of each CPU core, u_c is the core usage factor (bound in [0,1]), n_m is the allocated RAM memory and P_m is the power draw (in kW) of RAM. PUE (Power Usage Efficiency) is a cluster-defined metric that indicates the ratio between the energy absorbed by the cluster and the energy expensed in computations. $PUE = 1$ indicates ideal efficiency (i.e. all energy is used for computations).

Carbon footprint, can be calculated starting from (1):

$$F_{(A\to X)_R} = E_{(A\to X)} \times CI, \tag{2}$$

where CI is the regional carbon intensity coefficient. This coefficient indicates the gCO2e cost associated with the production of electricity in region R. Results presented in Section 3 will refer to the most updated (at the time when the study was performed) carbon intensity coefficient for Italy.[2]

Measurements & Data management Physical quantities indicated in Eqs.(1, 2) can be split in two groups: "static" data (i.e. data that does not change throughout the computation) and "dynamic" data (i.e. data that changes during the computation). The following variables belong to the former category:

- P_c and P_m were extrapolated from the Thermal Design Power (TDP) value indicated by the manufacturer. While the TDP formally indicates the maximum amount of heat generated by a component that the available cooling system can safely dissipate, [6] indicates that it can be used as a valid approximation of absorbed power.

[2]https://www.statista.com/statistics/1290244/carbon-intensity-power-sector-italy/

- n_c is the number of cores requested to perform the calculation.
- PUE metric indicates how efficient is the cluster in using energy to perform calculations. In this work, an approximated value was obtained through private communications with INFN-CNAF T1 User-Support.
- CI is a regional parameter that can be obtained through online searches or co-operating with energy providers.

These data are expected to be known before any computation, therefore the data was gathered in a separate configuration file that researchers can easily tailor around their experimental needs. For instance, if experiments run on the same node with different core request, the only parameter that must be eventually changed will be n_c. To ease the interaction with this configuration file, the *.toml* format was chosen for it is key-value based and allows to group data under human-friendly tags.
The remaining three variables are dynamic and need a separate treatment since they have to be measured without including any computational clutter coming from other processes running on the machine. This can be achieved leveraging the properties of the standard Linux filesystem structure, as shown in Figure 1:

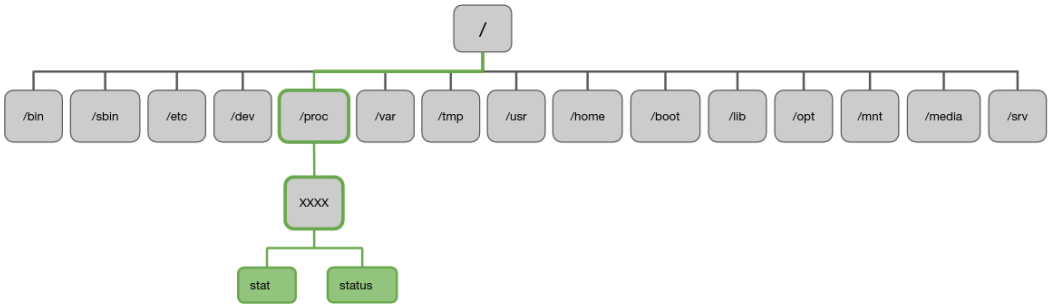

Figure 1: Schema of the Linux Filesystem. The absolute path used to retrieve data for measuring u_c, n_m and T is indicated in green.

The */proc* folder, whose access does not require *sudo* permissions, stores informations about all processes running on the machine. These process can be identified through their PID (Process ID), a unique integer identifier that the system associates them at start time and retires once they finish running. In */proc*, each process has a PID-named sub-folder (XXXX in Fig.1) where we can find *stat* and *status* files. We can use the data in them to calculate u_c and n_m respectively. u_c can be calculated through: $u_c = \frac{utime+stime}{uptime-starttime}$
where $utime$ is the time spent by the process in user-mode (14th field of *stat*), $stime$ is the time spent by the process in kernel-mode (15th field of *stat*), $uptime$ indicates how much time has passed since boot, obtainable with an inexpensive system call, and $starttime$ which is the uptime

value of process start (22nd field of *stat*). As for n_m, data is mainly stored in */proc/status* file, where we find the field *vmrss* which indicates the memory mapped, therefore allocated, to the process ID. Following [6] we adopt $n_m = \frac{P_m}{vmrss(GB)}$, where P_m is the power draw of installed RAM.
We can now discuss the sampling strategy for u_c and n_m.

Strategy The tool reads the *.toml* configuration file and acquires static data. Then, samples u_c, n_m every t seconds until the process has finished running, condition that retires the PID and allows to break the loop. Sampled data is stored in a data vector and used to extrapolate the average value of u_c,n_m that will be used in Eq.1.

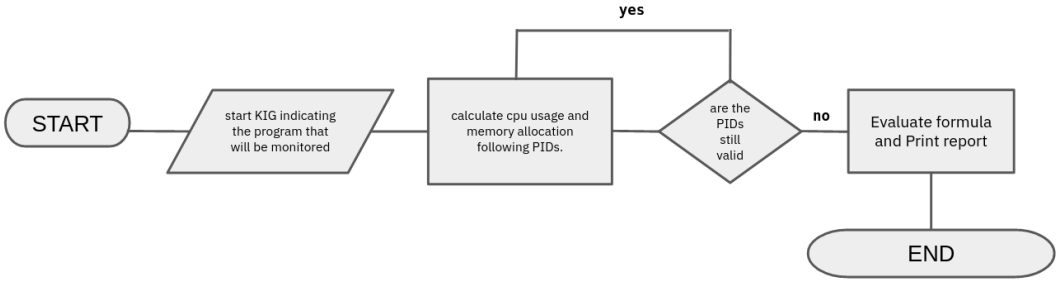

Figure 2: Simplified graphical representation of the algorithm implemented in KIG.

A C++ shared library and monitoring executable prototype have been written and made available on my Github space (https://github.com/fminarini/KIG). In the repository it will be possible to find a more "in-depth" description of the technical details (i.e. code-level) of KIG as well as links to documentation. To ease the adoption of the monitoring and make it compatible with common security policies of computing data centres, a Docker image configured with all the necessities has been uploaded on Docker Hub (https://l.infn.it/td) and can be freely downloaded. This image can be used also in Singularity/Apptainer. Technical details of this containerized solution can be found on the repository page of the project.

2.2 Test-bed

The service has been tested emulating the behaviour of a user launching some code on an "empty" and idle node. We tested it on two different nodes belonging to the High-Throughput Computing(HTC) Farm of INFN-CNAF Tier 1 Facility in Bologna. The first test node was a 172 HS06[3] machine (172HS06 in the following), while the second one was a 1293 HS06 machine (1293HS06 in the following). Meaningful specifications are shown below.

Table 1: Hardware specifications of the machines used as Test-bed for KIG.

	CPU Name	installed CPUs	Total Physical Cores	HyperThreading	RAM memory (GB)
172HS06	AMD Opteron 6320	2	16	NO	128
1293HS06	AMD EPYC 7313	2	32	YES	128

[3]HS06 is the HEP-wide benchmark for measuring CPU performance. It has been developed by the HEPiX Benchmarking Working Group. HS06 is based on the all-cpp benchmark subset (bset) of the SPEC CPU2006 benchmark suite.

We tested the monitoring service on a set of typical high-energy physics workloads involving Event generation and Simulation (GENSIM), Event digitization (DIGI) and Event Reconstruction (RECO). In particular, in order to test the prototype on computing activities that are compatible with typical practices of HEP research, we used containerized reference workloads also used in the context of the HEP benchmark suite presented in [7].

3. Results

In this section, the results obtained through the application of the monitoring software on reference HEP workflows will be discussed. We tested KIG on CMS GENSIM-DIGI-RECO workload. The choice of workflows belonging to this particular HEP experiment was merely opportunistic. The tool is designed to be completely independent from the experiment involved as well as the type of physics investigated. Results will be clustered in two subsections.
The first section will display the comparison between the two machines at test with respect to the same workload while increasing the number of total events processed. In this first session of analysis, workloads will run in "full-core" mode (they will access and use all the cores available on the machine). With this experiment we aimed at verifying that the tool could correctly display the expected performance differences between the two machines and also correctly exhibit the linearity of kWh absorption with respect to the number of events, since a larger number of events is expected to require a longer computing time, hence absorbing more energy.
In the second section we will display an analysis focused on a single node, specifically 172HS06. The goal of this analysis was capturing the behaviour elapsed computing time, power footprint and carbon footprint while partially loading the cores of the machine in order to determine whether a trade-off between these variables eventually exists.

3.1 Comparison between machines in full-core mode

The total amount of processed events in this mode can be calculated with $TPC \times EPT \times IC$, where:

- TPC is the number of threads per core (default value: 4).
- EPT is the number of events per thread (default value: 200).
- IC is the number of independent copies running on the machine (default value: ncores/TPC).

In this experiment, $TPC = 4$ and $IC = n_c/TPC$. EPT was increased manually for each data taking session.

3.1.1 GENSIM

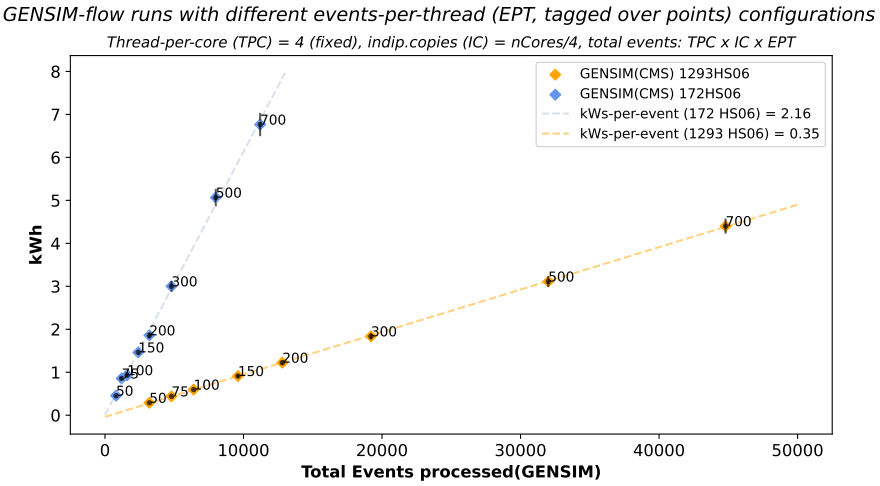

Figure 3: kWh vs Total GENSIM events plot for 172HS06 (blue) vs 1293HS06 (orange). The *x-axis* represents the total number of processed events. The *y-axis* represents the energy, in kWh, absorbed to process that number of events for that workload type.

Figure 3 shows qualitatively the performance difference between the two machines. To estimate the performance difference quantitatively, the kWs-per-event coefficient was extracted through a linear fit of data. The angular coefficient output of the linear fit was 2.16 kWs (kJ) for 172HS06 and 0.35 kWs (kJ) for 1293HS06. As a first estimation of error, the statistical fluctuation of measured kWh was adopted. To estimate the statistical fluctuation of measures, the standard deviation and average relative error were extracted over repeated measures of the cases in Table 3. This procedure resulted in a ±4% statistical error estimation. The intercept values are 0.007 kWh(25.2 kWs) for 172HS06 and -0.038 kWh(-136.8 kWs) for 1293HS06. These values display the estimated kWh absorption in the edge case of a workflow processing 0 events for which the estimation should be compatible with 0 kWh[4]. The values of energy consumption we extract from the intercepts are compatible, within the estimated statistical error of ±4%, with the expected measure of 0 kWh for such edge case.

Table 2: Summary table of Fit results in kWs. To obtain the equivalent kWh value, results must be divided by 3600.

	Angular Coeff. (kWs-per-event)	Intercept(kWs)
172HS06	2.16	25.2
1293HS06	0.35	-136.8

[4]Since KIG focuses on running computing processes only, idle state energy consumption of the server is neglected. Hence, a 0 event workflow should consume ≈ 0 kWh, as there would little work to do.

Table 3: Summary table of KIG measurements on 172HS06 and 1293HS06 for a GENSIM workload run in "full-core" mode.

GENSIM	172HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.6	0.45±0.02	800
75	0.87	0.85±0.03	1200
100	1.15	0.93±0.04	1600
150	1.07	1.46±0.06	2400
200	2.2	1.86±0.07	3200
300	3.36	2.99±0.12	4800
500	5.89	5.06±0.20	8000
700	7.73	6.76±0.27	11200

GENSIM	1293HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.27	0.29±0.01	3200
75	0.39	0.44±0.02	4800
100	0.52	0.59±0.02	6400
150	0.77	0.91±0.04	9600
200	1.01	1.22±0.05	12800
300	1.5	1.83±0.07	19200
500	2.5	2.24±0.09	32000
700	3.52	4.40±0.18	44800

Figure 3 and table 2, 3 show that on a GENSIM workload 172HS06 is less efficient than 1293HS06, specifically by a factor of ~ 6.2 (see Table 2). Since this workload is CPU-intensive and 172HS06 has fewer and less powerful cores, this behaviour is expected.

3.1.2 DIGI

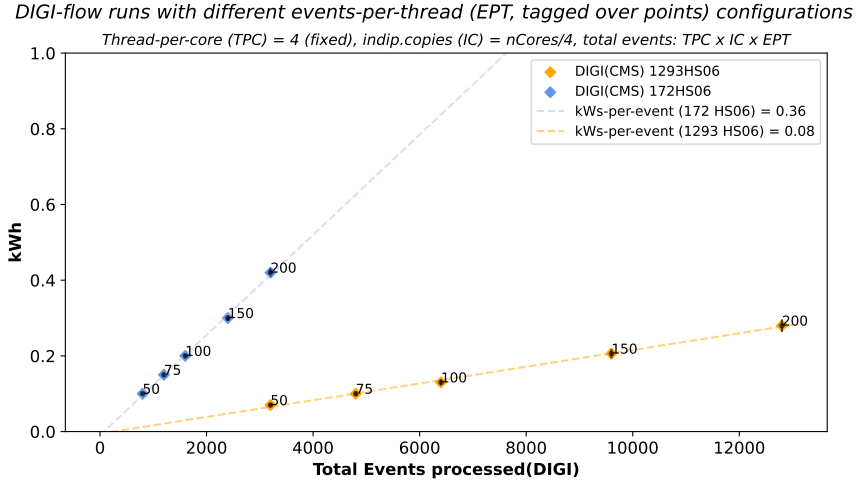

Figure 4: kWh vs Total DIGI Events plot for 172HS06 (blue) vs 1293HS06 (orange) against DIGI workload. The *x-axis* represents the total number of processed events. The *y-axis* represents the energy, in kWh, absorbed to process that number of events for DIGI.

Figure 4 shows qualitatively the performance difference between the two machines. To estimate the performance difference quantitatively, the kWs-per-event coefficient was extracted as explained in 3.1.1. The angular coefficient output of the linear fit was 0.36 kWs (kJ) for 172HS06 and 0.08 kWs (kJ) for 1293HS06. As a first estimation of error, the same procedure of Section 3.1.1 was adopted. The intercept values are -0.009 kWh (-32.4 kWs) for 172HS06 and -0.005 kWh (-18.0 kWs) for 1293HS06. These values display the estimated kWh absorption in the edge case of a workflow processing 0 events. The intercepts are compatible, within the estimated statistical error, with the expected measure of 0 kWh for such edge case.

Table 4: Summary table of Fit results in kWs.To obtain the equivalent kWh value, results must be divided by 3600.

	Angular Coeff. (kWs-per-event)	Intercept(kWs)
172HS06	0.36	-32.4
1293HS06	0.08	-18.0

Table 5: Summary table of KIG measurements on 172HS06 and 1293HS06 for a DIGI workload run in "full-core" mode.

DIGI	172HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.15	0.10±0.004	800
75	0.21	0.15±0.01	1200
100	0.27	0.20±0.01	1600
150	0.41	0.32±0.01	2400
200	0.52	0.42±0.02	3200

DIGI	1293HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.07	0.07±0.003	3200
75	0.09	0.10±0.004	4800
100	0.11	0.13±0.01	6400
150	0.18	0.20±0.01	9600
200	0.24	0.28±0.01	12800

Figure 4 and Table 4, 5 show the difference in performance of the two nodes against DIGI workloads. For such workloads, 172HS06 is less efficient than 1293HS06, specifically by a factor of ~ 4 (see Table 4). DIGI workloads, consisting mainly in the digitization of values, are much less CPU-intensive payloads than GENSIM, which can explain why the difference is less prominent.

3.1.3 RECO

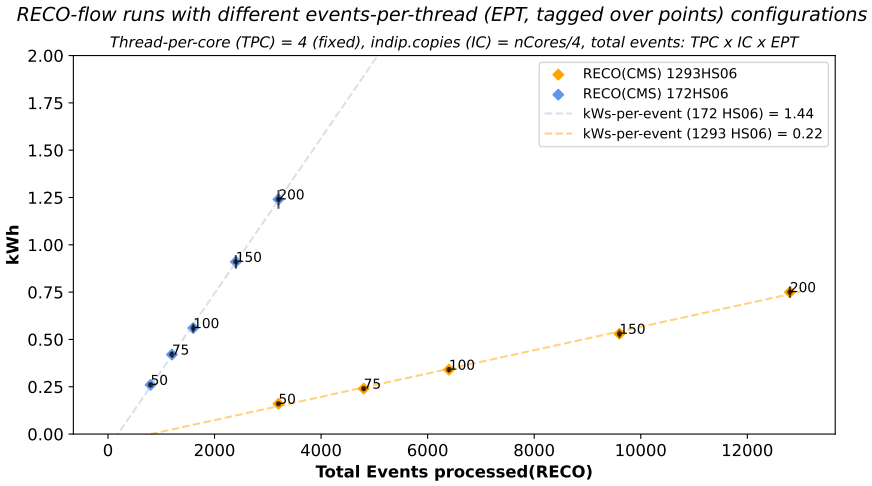

Figure 5: kWh vs Total RECO Events plot for 172HS06 (blue) vs 1293HS06 (orange) against RECO workload. The *x-axis* represents the total number of processed events. The *y-axis* represents the energy, in kWh, absorbed to process that number of events for RECO.

Figure 5 shows qualitatively the performance difference between the two machines. To estimate the performance difference quantitatively, the kWs-per-event coefficient was extracted through a linear fit of data. The angular coefficient output of the linear fit was 1.44 kJ for 172HS06 and 0.22 kJ for 1293HS06. The same error estimation procedure of Section 3.1.1 was adopted. The intercept values are 0.077 kWh (277.2 kWs) for 172HS06 and -0.049 kWh (-176.4 kWs) for 1293HS06. These values are slightly outside the ±4% boundary a measure of 0 kWh would have and may require further investigations over the error estimation procedure.

Table 6: Summary table of Fit results in kWs for RECO. To obtain the equivalent kWh value, results must be divided by 3600.

	Angular Coeff. (kWs-per-event)	Intercept(kWs)
172HS06	1.44	277.2
1293HS06	0.22	-176.4

Table 7: Summary table of KIG measurements on 172HS06 and 1293HS06 for a RECO workload running in "full-core" mode.

RECO	172HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.15	0.10±0.004	800
75	0.21	0.15±0.01	1200
100	0.27	0.20±0.01	1600
150	0.41	0.32±0.01	2400
200	0.52	0.42±0.02	3200

RECO	1293HS06		
EPT	Elapsed Time (h)	Energy(kWh)	Total Events
50	0.07	0.16±0.01	3200
75	0.09	0.24±0.01	4800
100	0.11	0.34±0.01	6400
150	0.18	0.53±0.02	9600
200	0.24	0.75±0.02	12800

Figure 5 and Table 6, 7 show that for RECO workloads 172HS06 is less efficient than 1293HS06 by a factor of 6.5 (see Table 6). RECO workloads, in terms of CPU-intensiveness, are much more similar to GENSIM than DIGI, which explains why the difference here is similar to the one showed in Table 2.

3.2 Trade-off analysis on 172HS06

The total amount of processed events in this mode can be calculated in the same fashion of Section 3.1. In this experiment, $TPC = 4$ and $IC = [4, 8, 12, 16]$. EPT was increased manually for each data taking session keeping the total number of events as constant as possible (the highest registered discrepancy was in the order of 1% and considered computationally irrelevant).

3.2.1 GENSIM

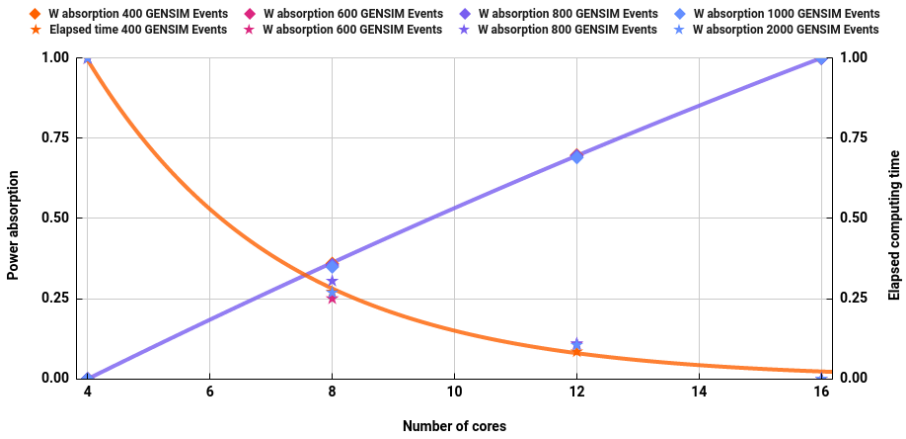

Figure 6: Trade-off kW vs Time for a GENSIM workload on 172HS06. The *x-axis* represents the number of cores used to perform the experiment. The *y-axis* represents the elapsed computing time and the *y'-axis* indicates the absorbed power. For visualization purposes, the elapsed computing time and absorbed power data were scaled in [0,1] for each case.

Figure 6 shows that for ≈ 8 cores the absorbed power line and the elapsed time line cross each other meaning that increasing the number of cores does decrease computing time but not so efficiently in terms of expensed power. Figure 7 and 8 show that increasing the number of cores dedicated to a GENSIM workflow does improve performance (from a user stand-point) but at the cost of a higher power absorption and footprint[5].

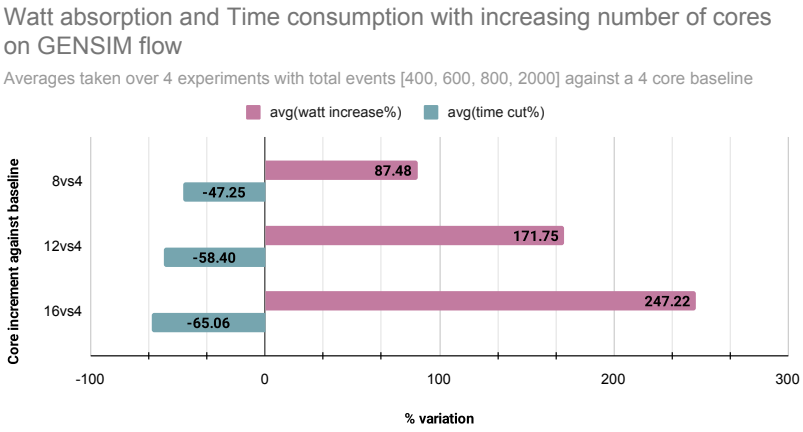

Figure 7: Analysis of power absorption increase and the elapsed time cut while increasing cores against a 4-core reference. The *x-axis* represents the percent variation of data. The *y-axis* represents the number of cores used in the experiment against 4-core reference baseline.

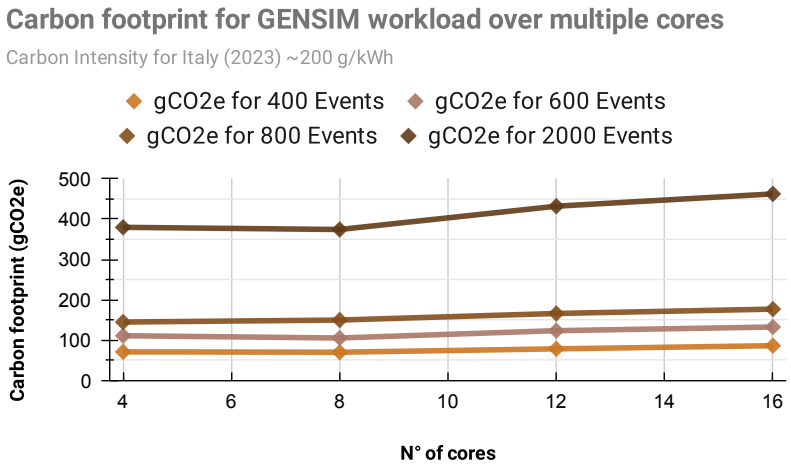

Figure 8: Carbon footprint of GENSIM workload. The *x-axis* represents the number of cores involved in the experiment. The *y-axis* indicates the carbon footprint (gCO2e) related to the experiment and calculated with (2).

As displayed in Figure 7, using more than 8 cores seems to lead to a non-optimized usage of energy.

[5]For all of the following footprint calculations, the Carbon Intensity coefficient of Italy(*https://www.statista.com/statistics/1290244/carbon-intensity-power-sector-italy/)* was adopted. At the time of the data taking, this coefficient was $\sim 200 g/kWh$

As a matter of fact, we see that using 12 cores leads to a $\sim$ 84% increase in terms of power absorption with only a $\sim$ 11% time reduction in return. A similar reasoning can be applied for the 16 cores case. Figure 8 brings up a similar conclusion, as it shows that the carbon footprint of calculations tends to exhibit a steeper increase after 8 cores.

3.2.2 DIGI

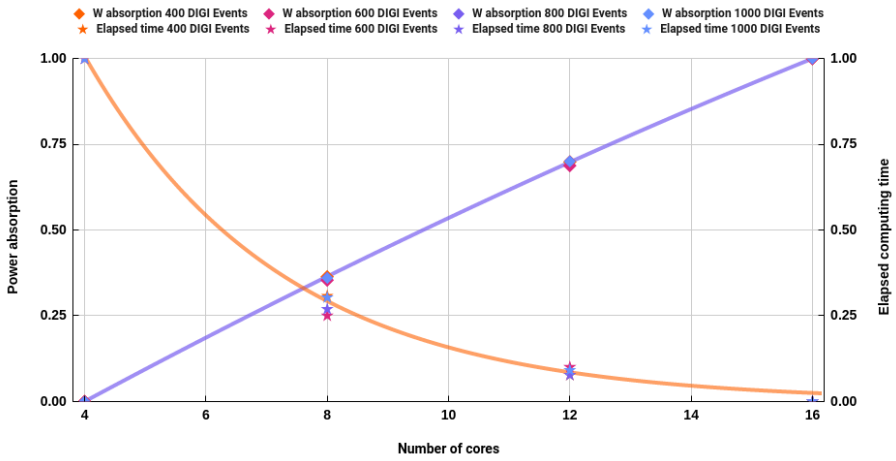

Figure 9: Trade-off kW vs Time plot for a DIGI workload on 172HS06. The *x-axis* represents the number of cores used to perform the experiment. The *y-axis* represents the elapsed computing time and the *y'-axis* indicates the absorbed power. For visualization purposes, the elapsed computing time and absorbed power data was scaled in [0,1] for each case.

Following the same approach of Section 3.2.1, we analyzed energy-wise the trade-off for DIGI workloads.

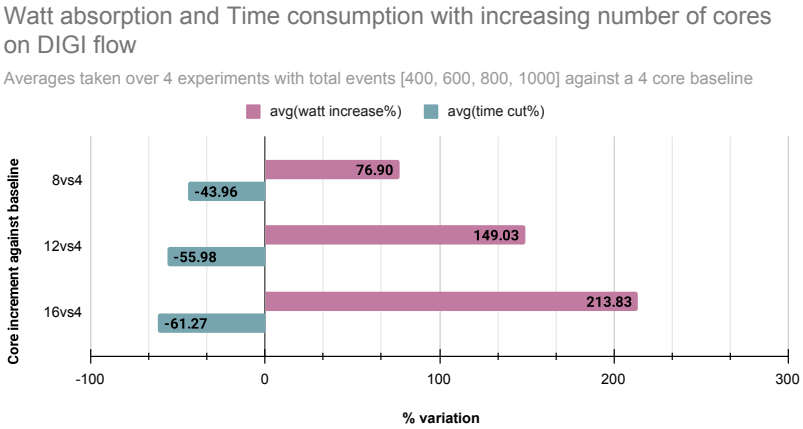

Figure 10: Analysis of power absorption increase and the elapsed time cut while augmenting the number of cores against a 4-core reference. The *x-axis* represents the percent variation of data. The *y-axis* represents the number of cores against 4-core reference.

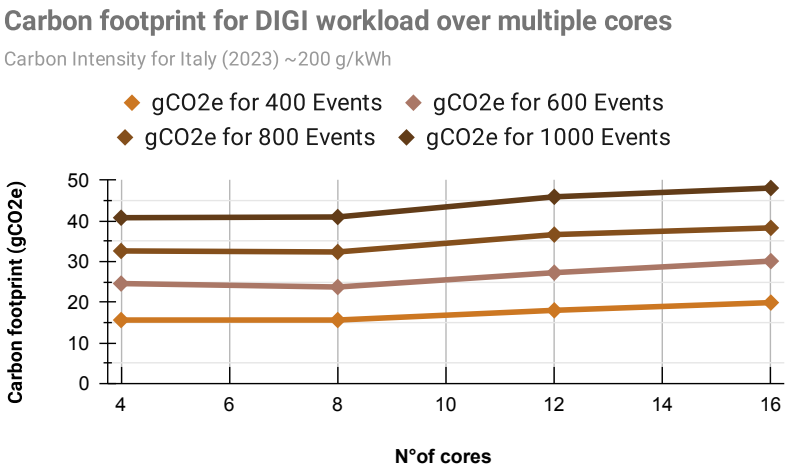

Figure 11: Carbon footprint of DIGI workload on 172HS06. The *x-axis* represents the number of cores involved in the experiment. The *y-axis* indicates the carbon footprint (gCO2e) related to the experiment, calculated with (2).

3.2.3 RECO

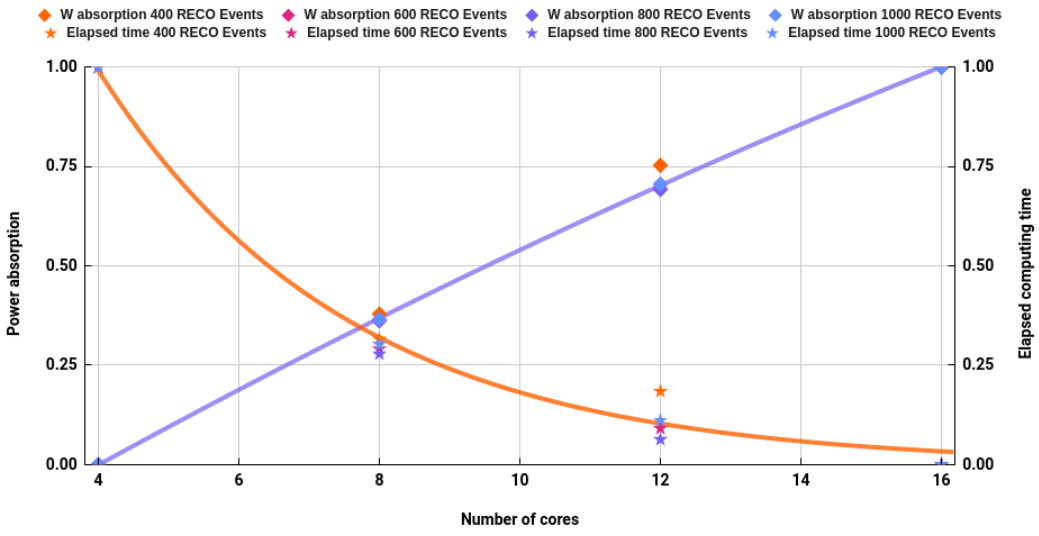

Figure 12: Trade-off kW vs Time plot for a RECO workload on 172HS06. The *x-axis* represents the number of cores used to perform the experiment. The *y-axis* represents the elapsed computing time and the *y'-axis* indicates the absorbed power. For visualization purposes, the elapsed computing time and absorbed power data was scaled in [0,1] for each case.

The power absorption increase and the elapsed time optimization with respect to 4-core reference are displayed hereafter following the indications of Section 3.2.1.

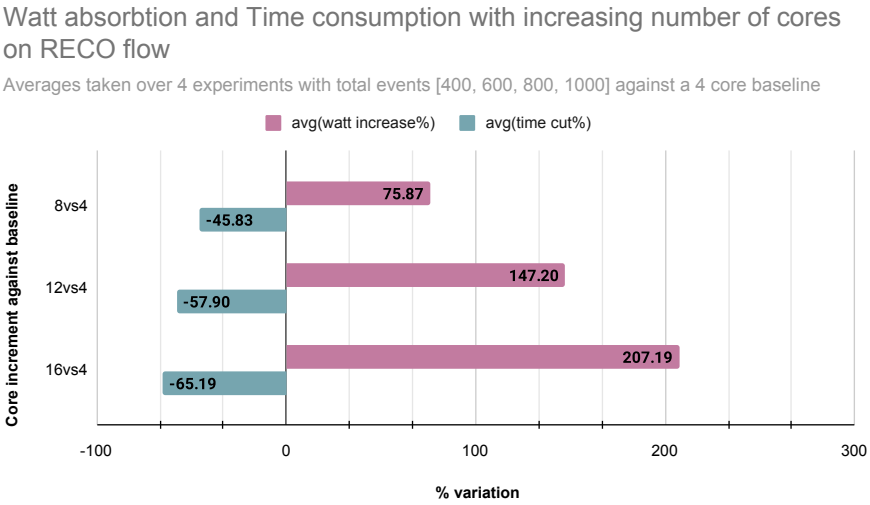

Figure 13: Analysis of power absorption increase and the elapsed time optimization of a RECO workload while augmenting the number of cores against a 4-core reference. The *x-axis* represents the percent variation of data.

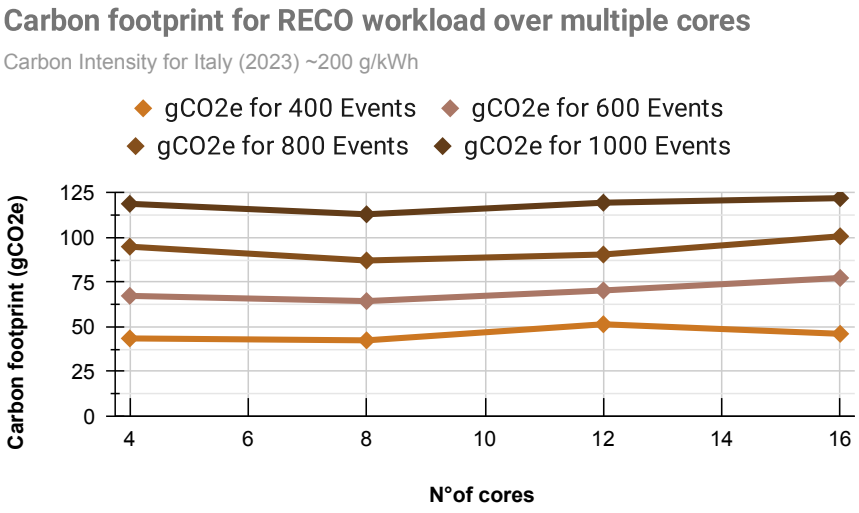

Figure 14: Carbon footprint of RECO workload on 172HS06. The *x-axis* represents the number of cores involved in the experiment. The *y-axis* indicates the carbon footprint (gCO2e) related to the experiment. calculated via (2).

4. Conclusions

In this work, we have developed a C++ tool prototype for monitoring the footprint of computing activities related to Physics research. The tool was validated in a comparison between two different machines. In this comparison, we expected the tool to be able to highlight the *a priori* known performance gap between the two resources. Figs. 3-5 displayed the expected difference and the "kWs-per-event" parameter numerically framed the performance gap. This datum, moreover, can also be used to characterize energy-wise any machine belonging to a computing cluster and therefore it is a metric of interest. We measured also the interplay between cores, computing time and absorbed power. Figs. 6-14 showed an apparently ideal trade-off point at 8 cores on 172HS06. While this information cannot be trivially used as a prescription for job submission balancing in the context of a data centre, it points out the need to perform further analyses in order to determine the reason why this point seems advantageous energy-wise. Some possible explanations might be software-bound (i.e. the trade-off happens due to Amdahl's law) or Hardware-bound (i.e. for more than 8 cores, the heat forces the CPU to throttle thus losing performance).

Next Steps In order of priority, the first step for further developments should involve passing from node-level measures to cluster-level ones, making the tool usable together with common submission systems. Thereafter, it might be interesting to evaluate the performance of non-standard computing architectures such as ARM and RISC platforms as well as popular computing accelerators such as GPUs and FPGAs. Finally, since the tool is physics research-agnostic, it could be interesting to include in the test-bed use-cases from non-HEP computing activities, in order to develop a broader perspective of how physics as a whole manages computing.

References

[1] D. Guest, K. Cranmer and D. Whiteson, *Deep learning and its application to LHC physics*, *Annual Review of Nuclear and Particle Science* **68** .

[2] M. Horowitz, *Computing's energy problem (and what we can do about it)*, in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014, DOI.

[3] E. Strubell, A. Ganesh and A. McCallum, *Energy and policy considerations for deep learning in NLP*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3645–3650, 2019, DOI.

[4] R. Schwartz, J. Dodge, N.A. Smith and O. Etzioni, *Green ai*, *CoRR* **abs/1907.10597** (2019) .

[5] L.A. Wright, S. Kemp and I. Williams, *'carbon footprinting': towards a universally accepted definition*, *Carbon Management* **2** (2011) .

[6] L. Lannelongue, J. Grealey and M. Inouye, *Green algorithms: Quantifying the carbon emissions of computation*, *CoRR* **abs/2007.07610** (2020) .

[7] D. Giordano, M. Alef, L. Atzori, J.-M. Barbet, O. Datskova, M. Girone et al., *HEPiX Benchmarking Solution for WLCG Computing Resources*, *Comput. Softw. Big Sci.* **5** .