

Article

C6EnPLS: A High-Performance Computing Job Dataset for the Analysis of Linear Solvers' Power Consumption

Marcello Artioli ¹, Andrea Borghesi ², Marta Chinnici ³, Anna Ciampolini ², Michele Colonna ², Davide De Chiara ⁴ and Daniela Loreti ^{2,*}

¹ ENEA-R.C. Bologna, 40121 Bologna, Italy; marcello.artioli@enea.it

² Department of Computer Science and Engineering, University of Bologna, 40136 Bologna, Italy; andrea.borghesi3@unibo.it (A.B.); anna.ciampolini@unibo.it (A.C.); michele.colonna3@studio.unibo.it (M.C.)

³ ENEA-R.C. Casaccia, 00196 Rome, Italy; marta.chinnici@enea.it

⁴ ENEA-R.C. Portici, 80055 Portici, Italy; davide.dechiara@enea.it

* Correspondence: daniela.loreti@unibo.it

Abstract: In recent decades, driven by global efforts towards sustainability, the priorities of HPC facilities have changed to include maximising energy efficiency besides computing performance. In this regard, a crucial open question is how to accurately predict the contribution of each parallel job to the system's energy consumption. Accurate estimations in this sense could offer an initial insight into the overall power requirements of the system, and provide meaningful information for, e.g., power-aware scheduling, load balancing, infrastructure design, etc. While ML-based attempts employing large training datasets of past executions may suffer from the high variability of HPC workloads, a more specific knowledge of the nature of the jobs can improve prediction accuracy. In this work, we restrict our attention to the rather pervasive task of linear system resolution. We propose a methodology to build a large dataset of runs (including the measurements coming from physical sensors deployed on a large HPC cluster), and we report a statistical analysis and preliminary evaluation of the efficacy of the obtained dataset when employed to train well-established ML methods aiming to predict the energy footprint of specific software.

Keywords: high-performance computing; power consumption dataset; parallel linear solvers



Academic Editors: Demetris Trihinas and Alexandros Karakasidis

Received: 25 March 2025

Revised: 16 April 2025

Accepted: 25 April 2025

Published: 30 April 2025

Citation: Artioli, M.; Borghesi, A.; Chinnici, M.; Ciampolini, A.; Colonna, M.; De Chiara, D.; Loreti, D. C6EnPLS: A High-Performance Computing Job Dataset for the Analysis of Linear Solvers' Power Consumption. *Future Internet* **2025**, *17*, 203. <https://doi.org/10.3390/fi17050203>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The computing capabilities of high-performance computing (HPC) systems have impressively increased in recent decades. This remarkable growth inevitably comes with a dramatic increment in power consumption [1]. Therefore, just like sustainability is becoming a key challenge in many human activities, maximising energy efficiency—besides performance—is becoming a priority for HPC facilities too.

Several works [2] focus on hardware efficiency by proposing technologies to reduce the environmental footprints of microprocessors, chips, and devices. Some recent attempts [3–6] focus instead on the software level by studying predictive models to assess the power required by parallel algorithms. The latter set of works stems from the observation that each execution of a job on an HPC infrastructure requires the allocation of specific resources, thus contributing to the system's total energy consumption. The ambitious goal is therefore to predict the energy consumption of an HPC system based on the energy consumed by individual jobs: a feature that could be particularly useful for many HPC tasks (e.g., energy-aware workload management, load balancing, infrastructure design, etc.). For example, if the HPC scheduler could rely on accurate prediction of the job's energy profiles

over time, it could take into account this information to allocate the jobs on the physical machines in a way that reduces the energy consumed by the whole infrastructure. More futuristic applications include energy consumption optimisations at compile time and programming tools able to suggest more energy saving ways to write the same algorithm (just as some developing tools are already able to understand the intent of the programmer as they write the code and suggest more computationally efficient solutions).

Software-level solutions can be divided into two categories: some works propose theoretical energy models [7–10] to mathematically connect the algorithm's steps with its energy footprint, whereas other techniques [11–17] try to predict the job's footprint based on historical records, usually by employing machine learning (ML) techniques. In this regard, a necessary condition to obtain accurate models is the availability of large sets of examples of past jobs' executions. However, the high variability in the characteristics of the workloads submitted to an HPC infrastructure (e.g., the kind of the algorithm, its CPU and RAM requirements, number of accesses to disk, inter-process communication scheme, etc.) can hinder the accuracy of the predictions, even if numerous historical examples are made available [12,17,18]. In general, fine-grained knowledge about the nature of the jobs can be crucial to improve the prediction accuracy of ML techniques.

In this work, we propose a first step towards this direction by restricting the attention to a particular kind of HPC workload, the parallel resolution of large-scale linear systems, and providing the scientific community with an extensive dataset of job runs and corresponding measured performance. The choice to focus on linear solvers stems from the observation that linear systems are widely used in the scientific field and often constitute the most suitable mathematical model to represent human activities, especially in technological and industrial contexts. Furthermore, as the complexity (and duration) of the resolution process is strictly connected to the matrix rank, the workload generated when solving large systems can have a significant impact on the power consumption of the whole application.

Our analysis relies on highly accurate measurements of energy performance, obtained through the use of sensors directly installed on each node of an HPC cluster, thus overcoming the limitations [19,20] of on-chip energy monitoring frameworks such as Intel's running average power limit (RAPL) [21].

The proposed dataset is particularly suitable for training ML techniques aiming to predict the energy consumption of HPC jobs. Nonetheless, the numerous and diverse sensor measures collected for each job, make it a good starting point for further analyses, possibly intertwining the hardware components' utilisations with their power consumption.

The contributions of this work can be summarised as follows.

- We present a novel execution framework to automate job launches with specific constraints, in terms of both resources and scheduling.
- We describe an extensive dataset of job runs with several dimensional configurations created through the aforementioned framework and made available to the scientific community. Each job run is reported together with the information coming from a variety of sensors.
- We provide a first glance at the proposed data by analysing the distribution of energy-related targets with respect to other meaningful dataset dimensions.
- We present a preliminary experimental evaluation of the predictive capabilities of standard ML models trained based on the proposed dataset. In this regard, we underline that proposing innovative ML models to analyse the data is out of the scope of this work. The evaluation is therefore intended as a first test of the efficacy of the obtained dataset when employed to train well-established ML methods.

The paper is structured as follows. We begin by providing a review of the state-of-the-art of techniques to assess and contain the power consumption of HPC jobs (Section 2). Then,

we describe our framework for automating job execution and data collection (Section 3). In Section 4, we provide a detailed breakdown of the collected dataset, delving into each specific piece of information gathered. Section 5 provides a statistical analysis of these data, while Section 6 evaluates the training of machine learning regression models given targets chosen from the entire dataset. The conclusion follows.

2. Related Work

Boosted by the need to reduce the ever-increasing environmental footprint of large datacenters, energy efficiency plays a role of growing importance in modern HPC systems [1]. Various research efforts have been devoted to this topic, tackling the problem at any architectural level, from hardware and infrastructure level to operating system, scheduling, and algorithm level.

The reduction in power consumption at the hardware level is often achieved through architectural changes and cooling methodologies [2,22]. Dynamic voltage and frequency scaling (DVFS) is one of the most used hardware-level techniques to reduce the power consumption of a processor [23,24]. Obviously, reducing the operating clock increases the duration of any running software, which may not only negatively affect the application performance [25–28], but also induce faults in the computation (making it more prone to radiation and thermal drifts) and, consequently, cause additional processing and power consumption [8,10,29]. Nonetheless, DVFS is a frequently employed energy-saving method, e.g., to improve job scheduling algorithms with power budgeting capabilities [30–32].

Higher-level measures to contain the power consumption of a system often rely on Intel’s RAPL [33,34], which offers a mechanism to access power consumption measures related to the system’s main hardware components. Although the reliability of these measures has been questioned in some works [19,20], RAPL is still widely employed to enforce power-aware job scheduling in HPC infrastructures [35,36] and an RAPL-alike interface is now also available for AMD’s architectures [37]. In our work, we avoid the debate about RAPL reliability by employing environmental and vendor-specific onboard sensors to measure the energy performance of the system.

Other approaches devoted to power consumption containment operate at the operating system- or job scheduling-level [4,38–40] by suggesting the “best” job execution order to minimise the energy requirement of the infrastructure—without any modification to hardware components nor nodes’ operational voltage/frequency. Concurrency throttling [12,41] can be classified as another system-level strategy to reduce energy consumption.

Some other attempts [3] operate at the software level and investigate the possibility of predicting the energy footprint of parallel algorithms from their characteristics (e.g., flops, memory occupation, communication scheme). Typically, these approaches can be classified into two classes.

The first class includes works [7–10,29] that propose mathematical models of power consumption based on the algorithm’s features. In particular, Choi et al. [8] develop a seminal model of the application’s power requirements mathematically combining the known features of the algorithm (i.e., operations, concurrency, and memory traffic) with those of the machine executing it (i.e., time and energy costs of each operation and each word of communication). Analogously to the approaches that could be tailored on top of the proposed dataset (which focuses just on linear solvers), Demmel et al. [42] restrict their attention to a particular kind of linear algebra task (i.e., matrix multiplication) and propose to relate energy with the algorithm’s number of flops, sent messages, and memory occupation. The work [9] proposes a simpler energy model relating the operational frequency of a multi-core machine with the algorithm’s features, and employs such a model to define an estimation methodology for the energy scalability of parallel algorithms. The

work by Aply et al. [29] builds on top of Shatz and Wang [20] reliability model to design a scheduling policy that includes makespan and energy in the equation. Their idea is to allow the re-execution of some tasks in case of faults, while we propose to investigate the checkpointing and ABFT methods such as IMe.

The second class of works investigating the power consumption of parallel algorithms encompass ML techniques to predict the energy utilisation of each job from historical records of past executions [11–17,43]. In [12–14], the authors focused on typical super-computer workloads and propose ML approaches, based on general user and application resource requests. As these features may not be enough to build a reliable predictor, Antici et al. [17] proposed to employ the natural language processing of the job's launching options to refine the predictions of the energy consumption of a trained ML model. Hu et al. [15] designed an effective cluster service that exploits the historical information about the past executions of GPU-enabled deep learning algorithms in order to decrease the overall energy utilisation of a datacenter. See [16] for a comprehensive study on ML-based energy prediction models for HPC.

All these works (except [42], that restrict the attention to matrix multiplication) do not focus on specific algorithms but try to tackle the problem of energy prediction in a more general way, by considering any kind of job running on the HPC infrastructure, even if this can hinder the accuracy of the predictions [12,17,18]. In contrast to these approaches, our idea is to focus on a specific kind of algorithm, for which crucial features (such as the input parameters) are well known and can be included in the training set. Although we expect the models built on top of the proposed dataset to be generally more accurate in their prediction, the obvious drawback will stand precisely in their generality. Nonetheless, the knowledge of the application's features before its execution (such as application's tags identifying similar jobs [28,44,45], job submission information [17,44], or the kind of algorithm as in our approach) is a very important point for the accurate prediction of power utilisation in HPC systems. Our proposed dataset goes precisely in this direction by suggesting that the attention to a specific—albeit rather pervasive—task be restricted so that the analysis of the algorithm's energy consumption can be conducted with a refined level of detail.

Other works related to our contribution are the workflow automation tools described in [46,47]. Both present interesting application-agnostic methodologies for benchmarking. While WA [47] is mainly designed for Android software and ARM architectures, JUBE [46] presents a more HPC-oriented approach that operates with Slurm and PBS schedulers.

3. Execution Framework

Jobs and measurements were carried out on ENEA's CRESCO6 cluster [48], a modern HPC facility consisting of 434 physical nodes, each of which has two sockets of 24 cores with 2.10 GHz Intel(R) Xeon(R) Platinum 8160 processors and 192 GB RAM (i.e., a total of 20,832 cores and over 80 TB RAM). An Intel Omni-Path 100 Gb/s network interconnects the machines. More importantly, from the point of view of this work, each CRESCO6's node is also equipped with a batch of sensors capable of making several types of measurements (e.g., power consumption, utilisation rate, temperature; each related to physical nodes, single CPUs, memory banks, cooling fans etc.) and various environmental sensors are placed in the server room to also provide information about the whole datacenter and its cooling system. See [49] for a detailed overview of the sensor infrastructure.

CRESCO6 is managed by the LSF scheduler (<https://www.ibm.com/docs/en/spectrum-lsf/10.1.0?topic=lsf-session-scheduler> accessed on 29 April 2025) and shared with many users. As shown in Figure 1, the system provides two login nodes on which job

runs can be built and submitted. One node is designated as a monitoring node to display and retrieve the data measured by the sensors.

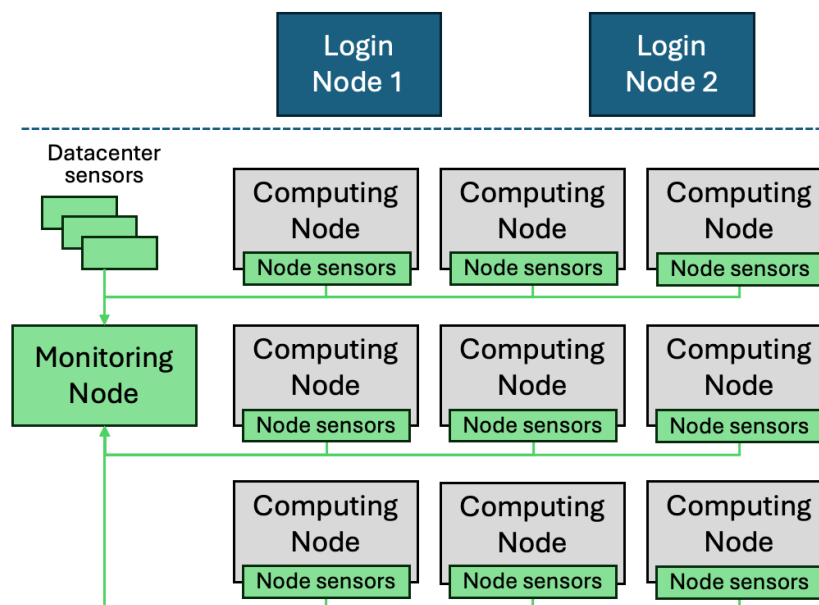


Figure 1. Architecture of CRESCO6 HPC systems.

To generate a dataset for subsequent analysis, we performed 7200 job runs. Each job is uniquely characterised by the input parameter configurations of the considered solvers: Gaussian-elimination (as implemented in PDGESV and PSGESV routines of ScaLAPACK library [50]) and the inhibition method (IME) (as parallelly implemented in the C-IMEFT procedure [51]). Both solvers are implemented as fault-tolerant MPI programs, but present key differences. In particular, since the ScaLAPACK routines are not fault-tolerant, this feature is achieved through checkpointing in our implementation. Differently, C-IMEFT employs the algorithm-based fault tolerance (ABFT) [52] and, therefore, it is intrinsically able to guarantee the computation from up to a configurable number of anywhere-located hard faults. See [51] for a detailed description and an evaluation of the two methods.

As regards the generation of the dataset, the manual management of the broad spectrum of possible configurations for the considered solvers would have been both impractical and inefficient. To address this challenge, we developed a dedicated framework with the primary objective of fully automating the process of generating and executing the combinatorial explosion of different job configurations. Figure 2 shows a diagram of how the submission and execution of a job take place. In the following, we provide a comprehensive exposition of the framework, underscoring its pivotal role in streamlining and optimising the process of execution and data collection.

As shown in Figure 2, the starting point of the designed framework is the *Launch Step*, which is responsible for the automatic generation of job submission commands. It encompasses the configurations for the various dimensions explored and the binding settings necessary to meet the resource requirements. Since the CRESCO6 cluster is used daily by many users, a simple launch of the devised jobs would have produced sensor measures highly influenced by the degree of infrastructure utilisation. To avoid this, the launch step is responsible for ensuring that the following requirements are met:

- (R1) No other user has to access/use the nodes employed by a job;
- (R2) The LSF job must use the minimum possible number of nodes;
- (R3) Jobs must be executed in sequence.

As each node consists of 48 cores, in order to meet the first requirement, the launch step always requests to LSF a number of cores equal to 48 multiplied by the desired number of nodes. The actual number of cores needed by the MPI algorithm will be defined in the MPI launch script, and if the algorithm uses fewer cores than specified, the unused processors will turn to the *idle* state but will not be released. In this way, we can test, for example, a job distributing its work to 48 ranks, in different settings: e.g., using all cores of one machine, using 24 cores of two different machines, or using 12 cores of four machines. Requirement R2 is needed because, by default, LSF aims to maximise the whole datacenter utilisation: if a physical node is partially used by other users, LFS may decide to assign some processes of a job to the remaining cores. Therefore, just requesting LSF a number of cores equal to 48 multiplied by the desired nodes (as stated to comply with R1) is not enough to actually guarantee that those 48 cores will belong to the same machine. The launch step is therefore responsible for specifying that each node must have 48 free cores in the LSF context of CRESCO6. Finally, the third requirement is needed to ensure the proper measurement of the job's energy consumption through the communication mechanism with the node acquiring data from the sensors. To achieve this, the launch step relies on the dependency requirements between jobs. In particular, the specified requirement is like "*ended(previous_job_name)*", indicating that the job at hand can only start after the previously submitted job has finished. The termination condition is valid independently of the termination status (error/success) of the predecessor job. This flexibility maximises the overall time efficiency of the benchmark workflow and helps to ensure continuity of operations by reducing possible delays due to individual failures.

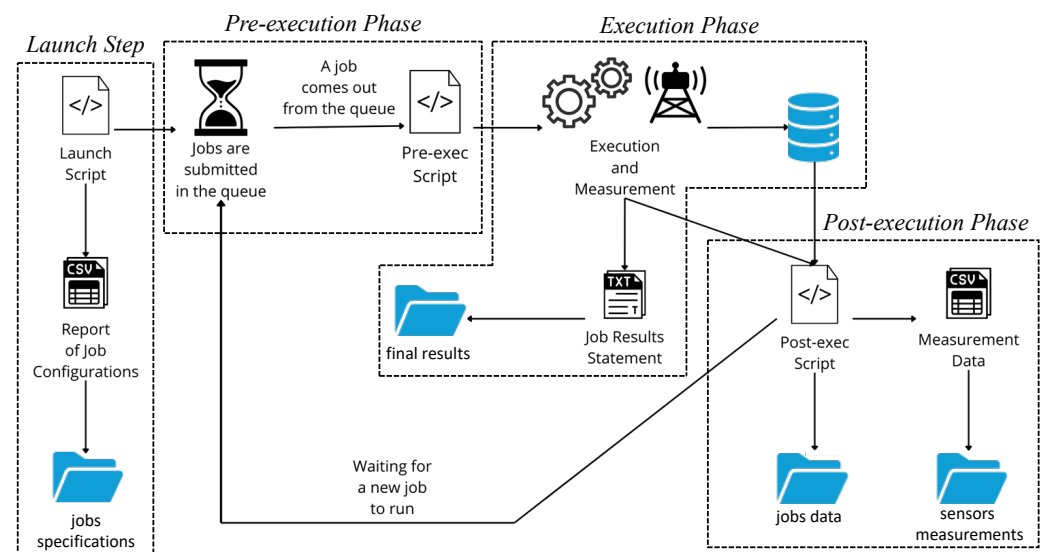


Figure 2. Framework scheme.

Since LSF dynamically decides at runtime the allocation of jobs to physical nodes, we implemented a mechanism to communicate this association to the script that collects sensor data. To communicate this information to the monitoring node, options were introduced to allow a *pre-execution phase* to run on the first host assigned to the job by LSF. In detail, once a job is submitted, it enters the LSF queue until sufficient resources are available to satisfy the constraints specified in the submit command. When it is picked from the queue, a pre-execution script starts. The success of this script is a prerequisite for the actual execution of the job. In our context, the usefulness of the pre-execution script is related to how LSF allocates resources (i.e., nodes) based on constraint satisfaction. Since the allocated nodes may vary from job to job, the pre-execution script communicates the resource information to the measurement software. In order to limit the overhead caused by the monitoring

task, our approach monitors only the nodes involved in the current job, rather than the entire system.

The third step in Figure 2 is the *Execution Phase*. Each machine in the CRESCO6 cluster is a Lenovo Think System SD530 equipped with vendor-specific sensors [49] that can measure various parameters, including energy consumption and node temperature. To obtain their data, during job execution, a command must be run on an isolated node in the CRESCO6 architecture (the monitoring node), which cannot be reached by user access nodes. Whenever this command is invoked, the instantaneous values sensed by the sensors on the CRESCO6's node passed as the input parameter are returned. The collection of the measures is performed through multiple parallel instances of the same measurement script. These instances act as daemons and operate cyclically, each collecting data from a single node with configurable frequency, and populating a database. For an efficient and effective operation of the monitoring system, a locking mechanism ensures that only the necessary parallel instances are used, while the others remain idle. The parallel execution of multiple daemons allows increasing the sampling rate of the sensors' measurements sufficiently to ensure the collection of an adequate number of measures for all executions, even for short jobs running on many cores.

At the end of all job runs, a *post-execution phase* takes place to stop the sampling activity of all active measurement daemons and save the collected relevant information. In particular, the monitoring data in the database are associated with the job identifier and saved in CSV format, while another file collects the job start and end timestamps in order to provide better specifications for filtering measurements. Other final summary information of the job run is saved by LSF in the job output file. A detailed account of all the information collected for each job is provided in the next section.

4. Dataset Structure

The CRESCO6 energy profiles of linear solvers (C6EnPLS) dataset, available on a GitHub repository [53], is divided into four blocks of information (corresponding to the four blue folders in Figure 2), providing a holistic overview of the executions: jobs specifications, jobs data, sensors measurements, and final results.

The *jobs specifications* block describes the input parameters of the routines, formed by the combinations of the input dimensions explored. The dimensions and values used are summarised in Table 1. In particular—besides the matrix size, number of computing processors, number of physical nodes, algorithm type, and single/double floating-point precision—we included the *fault tolerance level* and the *number of simulated faults* among the dimensions to be explored. Indeed, as previously described, both the considered solvers can be adapted to tolerate up to a configurable number of hard faults (i.e., errors that invalidate all the computation on certain processors). The fault tolerance level refers to the maximum number of faults located anywhere that the procedure can tolerate. In contrast, the number of faults refers to the actual number of errors that have been simulated and recovered during the job run. Furthermore, different *rank assignment* modes were considered when mapping MPI ranks to computing processors. In “fill” mode, each rank is assigned a core trying to fill the physical nodes as much as possible. In “span” mode, instead, the ranks are distributed equally on all the sockets of the involved physical nodes. In both cases, the cores not assigned to any rank are left idle but not released to LSF (in order to fulfil the requirement R1 described in Section 3).

The *jobs data* block contains all the information collected about each job run. That is the identification number assigned by LSF to each job, the starting/ending timestamps, and the assigned physical nodes.

The *sensors measurements* block contains measurements collected from the sensors placed on each node in the CRESCO6 cluster. These data include power and energy consumption, ambient temperature, and cooling system data for each node involved in the job run. Table 2 describes a list of the physical measures that are more relevant for this work. Other sensor data collected but not used in this preliminary analysis are described in [49].

Table 1. Description of the *job specifications* block fields and assigned values.

Dimension	Description	Values
<i>Job name</i>	Job identification name	-
<i>Matrix size</i>	Rank of the input matrix	5280, 10,560, 15,840, 21,120, 26,400, 31,680, 36,960, 42,240
<i>Calculation processes</i>	Number of processes dedicated exclusively to the calculation of the system's solution	64, 100, 144, 256, 400, 484, 576, 768
<i>Nodes</i>	Number of employed physical nodes	[1,...,16]
<i>algorithm</i>	Considered linear solvers	IMe, ScaLAPACK
<i>Precision</i>	Numerical representation of real numbers	Single, double
<i>Fault tolerance level</i>	Number of faulty processes that can be handled	0 (no fault tolerance), 1, 2, 4, 8
<i>Number of simulated faults</i>	Number of faults to be simulated (and recovered)	0, maximum fault tolerance level
<i>Rank assignment</i>	Way to assign ranks to computing processors	Span, fill

Table 2. Main *sensor measurements* block fields and their description. Other collected sensor data are described in [49].

Field	Description
<i>jobid</i>	ID of LSF job
<i>nodename</i>	Name of the physical node. Generally, a number
<i>timestamp_measure</i>	Timestamp of the measure expressed in Unix time
<i>sys_power</i>	Total instantaneous power measurement of the computing node in watts
<i>node_energy</i>	Energy meter consumed by the node up to the time of reading. Useful for making differences between two readings in kWh
<i>delta_e</i>	Difference between previous and current measurement of energy in kWh for that node

The *final results* block collects execution statistics including execution time, average, and total memory consumption, and any computational error discovered during the executions. Specifically, the main information in this block belongs to two different categories:

- *The LSF resource usage summary*, which includes: CPU time, maximum and average memory utilisation, maximum number of processes, maximum number of threads, runtime, turnaround time, etc.
- *The output of the tested algorithm*, which includes: norm-wise relative error of the solution, powercap energy counters, runtime of the algorithm's subparts (i.e., initialisation and execution), etc.

Through this elaborate dataset made available to the community [53], we hope to enable the development of novel techniques devoted to understanding the energy footprint of the considered solvers. The dataset also includes Jupyter notebooks to simplify data

inspection, generate summaries of relevant information about each job run, and visualise the data distributions.

5. Statistical Analysis of the Dataset

In order to provide a first glimpse of the collected data, Table 3 presents a comprehensive overview of the dataset through key statistical indices for each field: minimum, maximum, and mean value, standard deviation, and 25th–50th–75th percentiles. In addition to the dimensions already presented in Table 1, the list includes the number of spare processes (i.e., the number of additional ranks necessary to provide fault tolerance), the total number of processes, the number of processes assigned to each socket (directly connected to the aforementioned “span” or “fill” policy for rank assignment), ScaLAPACK blocking factor and checkpointing rate, and four key metrics related to each job run: total energy consumption, peak power, average power, and execution time. Focusing on the latter four metrics, Table 3 suggests that, in the proposed dataset, the total energy and runtime metrics have high variability, as indicated by their large standard deviations and the extensive range between the minimum and maximum values. Also, the median values are much lower than the mean, indicating that a significant number of jobs have energy and runtime values lower than the mean, but a few jobs increase the average. On the contrary, peak and average power consumption seem more uniformly distributed, with median values more similar to the mean ones.

Table 3. Description of the dataset data.

Fields\Index	Mean	Std	Min	25%	50%	75%	Max
<i>Matrix size</i>	22,176	11,760.75	5280	10,560	21,120	31,680	42,240
<i>Calculation processes</i>	314	181.084	64	144	256	484	576
<i>Spare processes</i>	29.73	45.73	0	2	8	40	192
<i>Total processes</i>	343.73	195.41	64	148	320	492	768
<i>Nodes</i>	7.67	4.02	2	4	7	11	16
<i>Fault tolerance level</i>	3.33	2.7	0	1	2	4	8
<i>Simulated faults</i>	1.67	2.58	0	0	0	2	8
<i>Processes per socket</i>	10.99	11.11	0	0	8	23	24
<i>ScaLAPACK checkpoint</i>	4928	6762.64	0	0	0	10,560	21,120
<i>ScaLAPACK blocking factor</i>	10.38	11.63	0	0	0	24	25
<i>Total energy (Wh)</i>	77.73	144.39	0.73	6.88	17.69	69.4	2626.28
<i>Peak power (W)</i>	2131.32	1195.63	280	1120	1900	3030	5700
<i>Average power (W)</i>	1960.72	1064.42	304	990.10	1835.87	2813.40	4879.89
<i>Runtime (s)</i>	142.07	282.77	6	15	36	129.75	9481

Figure 3 presents histograms for energy, runtime, peak, and average power consumption metrics, showing the general shape of the data distributions when considering the whole dataset samples. So, for example, the energy graph reports different ranges of energy consumed on the x axis, and the height of the bars indicates the number of samples (i.e., job runs present in the dataset) that fall into the corresponding energy range. Above the histograms, a density curve estimates the variable’s probability distribution. Given

the skewed nature of energy and runtime data deducible from the values in Table 3, these two metrics are presented in a logarithmic scale to better appreciate their trends. Besides an unusual, non-strict correlation between runtime and energy consumed, the graphs suggest that the average value of the system power follows a similar trend to that of the maximum system power. However, if we restrict our attention to different dataset slices, other trends emerge.

For example, Figure 4 focuses on a subset of job runs involving just 2, 8, or 16 physical nodes. As expected, the three data series appear to have very different domains, especially for average power consumption: e.g., while most jobs running on 2 nodes have average power consumption under 1000 W, running on 8 nodes consumes between 1000 W and 3000 W, and jobs on 16 nodes generally consume between 3000 W and 5000 W. We remark that the graph presents the average power consumption for different numbers of nodes on the x axis (cumulative power, summing all nodes), and on the y axis, the number of jobs showing that power. It was built by first computing the power profiles of each job over time (summing together the contribution of different nodes if the job runs in parallel on multiple machines) and then calculating the mean value of that profile. Depending on such a value, the job contributes to increasing by one the height of the corresponding bar in the plot. Hence, as expected, the plot suggests an increase in the average power consumption linear with the number of employed nodes. Delving more into the dataset, we observe that jobs running on 2 nodes have a mean power consumption of 504 W on average, on 8 nodes 2240 W (around $\times 4$ with respect to 2 nodes), and on 16 nodes 4271 W (around $\times 8$ with respect to 2 nodes).

On the other hand, Figure 4 shows that the energy and runtime profiles of the three series have domains not as distinct as that of the power metrics, highlighting that some jobs running on just two nodes needed more time to complete, and, as a consequence, consumed a considerable amount of energy, comparable to that of shorter jobs involving many more nodes. This behaviour, which may appear counterintuitive, is derived from two apparently unrelated factors: (i) the jobs are not always launched saturating the cores available on the involved nodes; (ii) both IMe and ScaLAPACK parallel solvers operate on large data structures generating contention between the MPI ranks executing on different cores of the same machine in the access to the bus, cache and memory. This is especially true if the job saturates the machine's available cores.

As a consequence, the execution of a certain job on the minimum number of physical machines may require so much more time than executing it on more machines, so that the increase in the energy derived by the employment of more nodes is actually counterbalanced by the fact that the job requires much less time to execute. The same considerations can be made looking at Figure 5, which correlates energy and runtime: jobs are ordered by runtimes on the x axis and the two lines show the energy consumed by the job (total energy) and its value normalised by the number of involved nodes (energy per node). As expected, the energy per node is linear in the runtime, while the total energy consumed by a job shows a much higher variability caused by the aforementioned resource contention.

Different trends are visible in Figure 6, where the data distributions are limited to three distinct matrix ranks (5280, 26,400, and 42,240). In general, jobs working on smaller matrices have lower runtimes and energy utilisation than solvers working on big matrices, while the distributions of average and peak power consumption seem to be less matrix-size-independent.

Figure 7 shows the distribution of the four metrics when different rank assignment methods are employed ("span" and "fill"). All in all, the trends do not highlight a significant difference in any metric between the two cases. Similar considerations can be made for Figure 8, which unveils the distributions when single or double precision is used in the

solving process. However, in this case, a slight difference can be spotted in the energy (and runtime) distribution domains: double-precision jobs can consume up to 1000 Wh, while single-precision ones do not go beyond 500 Wh.

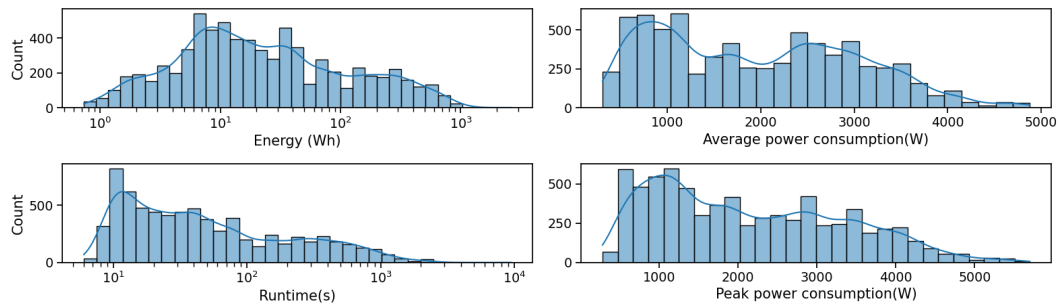


Figure 3. Distributions of energy-related metrics in C6EnPLS: total energy consumption, runtime, average, and peak power. The line highlights the kernel density estimation of the variables' probability distributions.

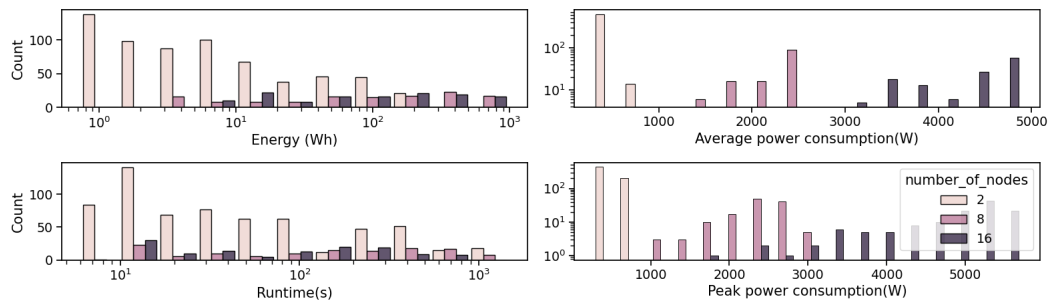


Figure 4. Comparison of energy-related metrics distributions for jobs running on different numbers of nodes.

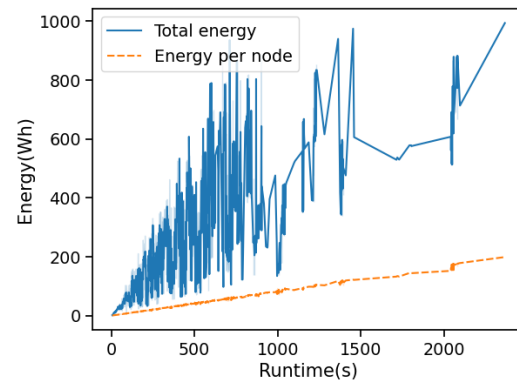


Figure 5. Relation between runtime and energy consumed by the jobs of the dataset.

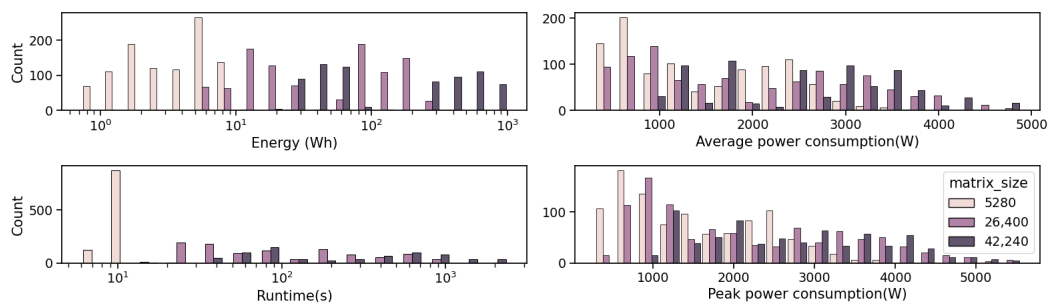


Figure 6. Comparison of energy-related metrics distributions for jobs working on matrices of different sizes.

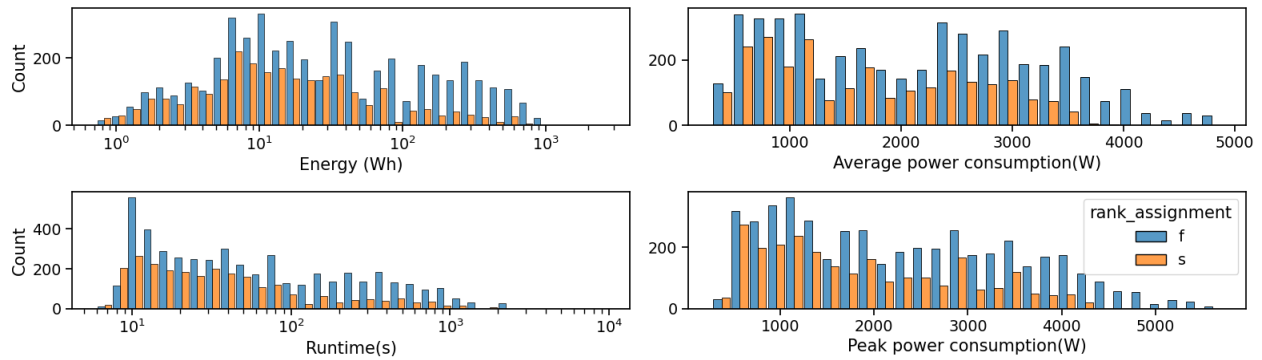


Figure 7. Comparison of energy-related metrics distributions for jobs with different rank assignment schemas (*f* and *s* refer to “fill” and “span” modes, respectively).

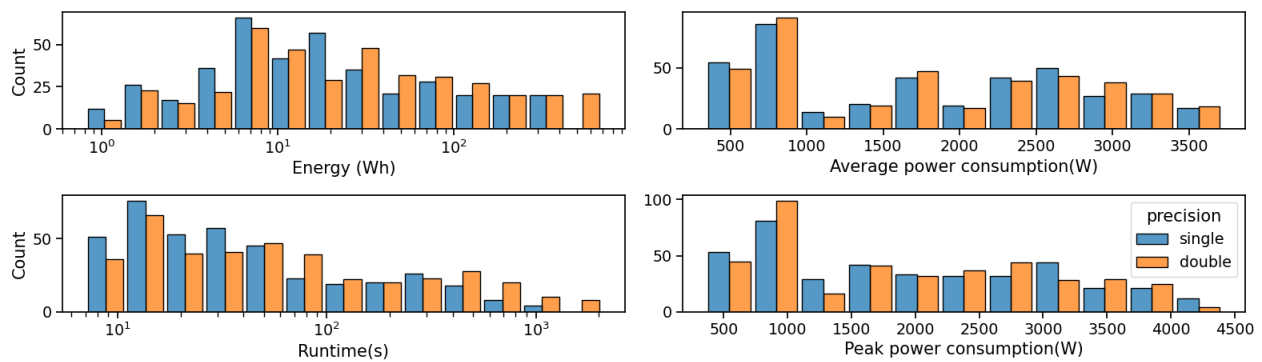


Figure 8. Comparison of energy-related metrics distributions for double and single precision solvers.

While a comparison by algorithm did not highlight significant differences, in Figure 9, the focus is on the level of fault tolerance that the solver can handle (i.e., the maximum number of processors that can incur a hard fault without invalidating the whole computation). For the sake of clarity, only two levels of fault tolerance are shown in the graph: 0 (i.e., the fault tolerance method is not implemented) and 8. Similar trends are visible for all the metrics, with only a slight difference in the shape of the histograms for average and peak power: the graph with a fault-tolerance level equal to 8 shows a longer tail to the right, corresponding to more cases with high average and peak power with respect to the no-fault-tolerance case. All in all, fault tolerance slightly influences the peak and average power without significantly impacting the overall energy consumption.

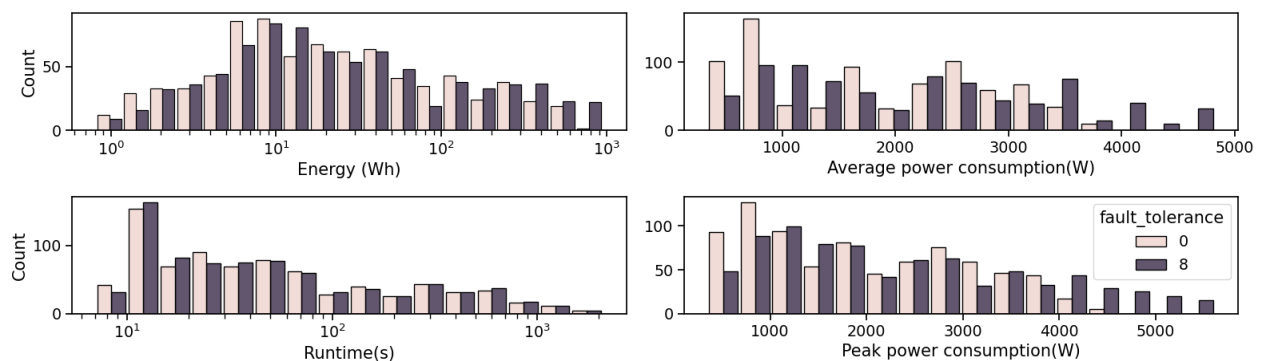


Figure 9. Comparison of energy-related metrics distributions for solvers with different fault tolerance levels.

6. Regressor Evaluation and Prediction Error Analysis

In this section, we want to verify the efficacy of the created dataset as training for the ML predictors of the four previously defined target variables (i.e., total energy consumption, maximum system power, mean system power and runtime). Specifically, we consider decision tree regressor, random forest regressor, and gradient boosting decision tree regressor. We remark that this work aims to provide the scientific community with a detailed dataset of HPC job runs, useful for the further analysis of the environmental footprint of linear solvers. As such, proposing novel ML models for energy consumption prediction is out of the scope of the paper.

Table 4 details the performance metrics for the chosen regression models in predicting the job's energy performance. For each model, we used GridSearchCV to optimise hyperparameters, with negative mean absolute error as the evaluation criteria. The table also includes metrics like RMSE, MAE, and MAPE, along with the optimal hyperparameter value for depth.

When considering all the evaluation metrics, the Random Forest Regressor emerges as the best overall performer for predicting the total energy consumption and the maximum power. Instead, the gradient boosting decision tree regressor shows the strongest performance for predicting the average power and runtime.

Table 4. Summary table of regressor prediction evaluation of the dataset.

TARGET	REGRESSOR	BEST DEPTH	RMSE	MAE	MAPE
Total Energy	<i>Decision tree</i>	19	0.013	0.004	0.076
Min value: 0.000470 kWh	<i>Random forest</i>	13	0.010	0.003	0.062
Max value: 0.938590 kWh	<i>GBDT</i>	5	0.010	0.003	0.119
Max power	<i>Decision tree</i>	10	12.676	7.966	0.024
Min value: 150.000000 W	<i>Random forest</i>	11	10.241	6.947	0.020
Max value: 440.000000 W	<i>GBDT</i>	6	10.339	7.089	0.021
Mean power	<i>Decision tree</i>	9	7.297	5.188	0.024
Min value: 115.000000 W	<i>Random forest</i>	11	6.530	4.632	0.022
Max value: 330.158420 W	<i>GBDT</i>	5	5.942	4.319	0.020
Runtime	<i>Decision tree</i>	10	22.238	7.296	0.065
Min value : 6.000000 s	<i>Random forest</i>	12	19.651	7.033	0.060
Max value: 2097.000000 s	<i>GBDT</i>	6	14.637	5.560	0.065

The histograms in Figure 10 depict the distribution of prediction errors for the four chosen targets. The *x axis* of each graph represents the prediction error value, while the *y axis* represents the error frequency, that is, the number of job runs tested that returned that error value. The models were trained on 70% of the data, and the remaining 30% was used for the testing. The errors for total energy consumption and runtime show a similar trend, with a significant peak centred around zero. The errors for the power targets (maximum and average power) exhibit a wider distribution, ranging from 0 to 40 W in absolute value. However, considering the average values of the power targets (with a maximum of approximately 2130 W and 1960 W on average), this error range remains very good.

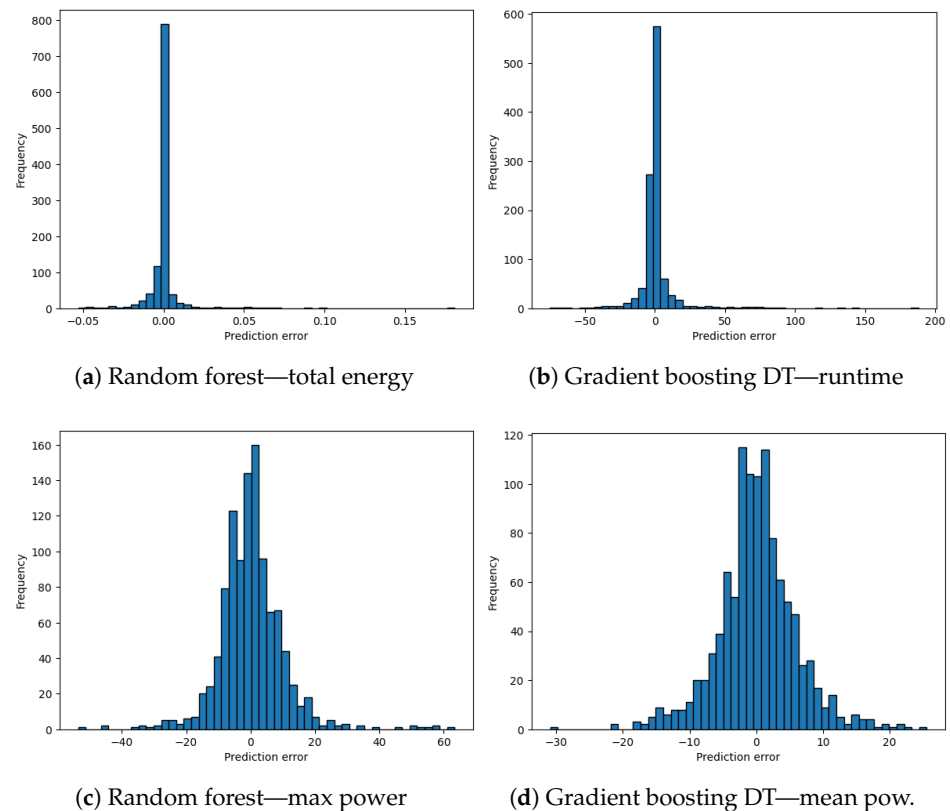


Figure 10. Best regressors' distribution of error predictions.

7. Conclusions and Future Work

HPC systems have witnessed explosive growth in computing power in recent decades, but this remarkable advancement comes at the cost of dramatically increased power consumption. In this paper, we reason about the use of ML techniques to predict the energy consumption of HPC workloads. In particular, we focus on the parallel resolution of large-scale linear systems and propose a large dataset of job runs equipped with information coming from physical sensors. We also describe the framework used for running the algorithms and retrieving the scattered data. Finally, we apply standard ML techniques on chosen energy-related targets and perform statistical analysis and error evaluation to assess the performance and effectiveness of such models when trained on the proposed dataset.

In future work, the challenges and goals outlined certainly include conducting analysis and evaluation for other targets already recorded in C6EnPLS, such as the utilisation and the temperature of the hardware components, the speed of the cooling fans on each machine, etc. After this, we will explore using alternative ML models, such as neural networks, which can potentially capture more complex relationships within the data by offering more accurate target predictions. Such an analysis should also help us understand which features play the most important role in determining the energy footprint of a linear solver. We also plan to enrich the dataset with further launch settings where other jobs are allowed to run on the same machine, thus making the collection more similar to real-life scenarios. However, we expect the energy consumption prediction task to be much more challenging in that setting.

Another interesting future work could involve extending the dataset and generalising the study to different solvers (e.g., indirect, iterative solvers, GPU-based implementations of IMe and Gaussian elimination, etc.), including a wider range of linear algebra problems (such as matrix multiplication and eigenvalue computation), or extending the dataset to different settings where parallel computation is supported by distributed processing en-

gines [54,55]. Furthermore, the methodology described in this work could be easily applied to other scenarios, where the HPC environment is dedicated to the execution of parallel tasks, known to be highly energy demanding—such as meteorological computations [56] or parallel process mining workloads [57–59] in order to identify the features that most likely contribute to the energy footprint in those settings. In general, it would be interesting to build a dataset with mixed types of jobs in order to generalise our study to situations in which the HPC infrastructure hosts various workloads (as it is in practice). We believe that ML techniques could benefit from real-time adaptability to account for the HPC intrinsic variability. Furthermore, the exploration of the effects of power capping mechanisms could also be an interesting direction for enriching the dataset. For example, DVFS is known to be an effective way to reduce the power requirements, but its side effect of increasing the application runtime can sometimes determine higher energy consumption overall. A dataset including several power-capped job runs could also be useful to shed light on this trade-off.

Finally, the integration of the proposed framework in the JUBE system [46], enabling communication with LSF scheduler (besides Slurm and PBS) and with the CRESCO6 sensors system, represents an interesting matter of future work.

Author Contributions: Conceptualisation, D.L. and M.A.; Data curation, D.L. and M.C. (Michele Colonna); Methodology, A.B., D.L., D.D.C., and M.A.; Software, D.L. and M.C. (Michele Colonna); Supervision, M.A., A.B., M.C. (Marta Chinnici), A.C., D.D.C., and D.L.; Validation, M.A., M.C. (Marta Chinnici), A.C., and D.D.C.; Visualisation, D.L.; Writing—Original draft preparation, D.L. and M.C. (Michele Colonna). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The C6EnPLS dataset produced by this study and the software to extract and analyse all the data are publicly available at <https://github.com/Reference-IMe/C6EnPLS> (accessed on 29 April 2025) DOI: 10.5281/zenodo.14135916. The code of both IMe solver and ScaLAPACK+checkpointing is available at <https://github.com/Reference-IMe/ime-tester.git> (accessed on 29 April 2025).

Acknowledgments: The computing resources and the related technical support used for this work have been provided by CRESCO/ENEAGRID High Performance Computing infrastructure and its staff. CRESCO/ENEAGRID is funded by ENEA and by Italian and European research programmes. Daniela Loreti has realised this work with a research contract co-financed by the European Union—PON Ricerca e Innovazione 2014–2020 ai sensi dell’art. 24, comma 3, lett. a), della Legge 30 dicembre 2010, n. 240 e s.m.i. e del D.M. 10 agosto 2021 n. 1062.



UNIONE EUROPEA
Fondo Sociale Europeo



Conflicts of Interest: Marcello Artioli, Marta Chinnici, and Davide De Chiara were employed by the company ENEA-R.C. Bologna, R.C. ENEA-Casaccia (Rome) and ENEA-R.C. Portici (Naples), respectively. The authors declare no conflicts of interest.

References

1. Malms, M.; Cargemel, L.; Suarez, E.; Mittenzwey, N.; Duranton, M.; Sezer, S.; Prunty, C.; Rossé-Laurent, P.; Pérez-Hernandez, M.; Marazakis, M.; et al. ETP4HPC's SRA 5—Strategic Research Agenda for High-Performance Computing in Europe—2022. *Zenodo* **2022**. [CrossRef]
2. Gupta, U.; Kim, Y.G.; Lee, S.; Tse, J.; Lee, H.H.S.; Wei, G.Y.; Brooks, D.; Wu, C.J. Chasing carbon: The elusive environmental footprint of computing. In Proceedings of the 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Republic of Korea, 27 February–3 March 2021; pp. 854–867.

3. Orgerie, A.; de Assunção, M.D.; Lefèvre, L. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.* **2013**, *46*, 1–31. [\[CrossRef\]](#)
4. Xie, G.; Xiao, X.; Peng, H.; Li, R.; Li, K. A Survey of Low-Energy Parallel Scheduling Algorithms. *IEEE Trans. Sustain. Comput.* **2022**, *7*, 27–46. [\[CrossRef\]](#)
5. Czarnul, P.; Proficz, J.; Krzywaniak, A. Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. *Sci. Program.* **2019**, *2019*, 8348791. [\[CrossRef\]](#)
6. Jin, C.; de Supinski, B.R.; Abramson, D.; Poxon, H.; DeRose, L.; Dinh, M.N.; Endrei, M.; Jessup, E.R. A survey on software methods to improve the energy efficiency of parallel computing. *Int. J. High Perform. Comput. Appl.* **2017**, *31*, 517–549. [\[CrossRef\]](#)
7. Tran, V.N.; Ha, P.H. ICE: A General and Validated Energy Complexity Model for Multithreaded Algorithms. In Proceedings of the 22nd IEEE International Conference on Parallel and Distributed Systems, ICPADS 2016, Wuhan, China, 13–16 December 2016; pp. 1041–1048. [\[CrossRef\]](#)
8. Choi, J.; Bedard, D.; Fowler, R.J.; Vuduc, R.W. A Roofline Model of Energy. In Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2013, Cambridge, MA, USA, 20–24 May 2013; pp. 661–672. [\[CrossRef\]](#)
9. Korthikanti, V.A.; Agha, G.; Greenstreet, M.R. On the Energy Complexity of Parallel Algorithms. In Proceedings of the International Conference on Parallel Processing, ICPP 2011, Taipei, Taiwan, 13–16 September 2011; pp. 562–570. [\[CrossRef\]](#)
10. Zhu, D.; Melhem, R.G.; Mossé, D. The effects of energy management on reliability in real-time embedded systems. In Proceedings of the 2004 International Conference on Computer-Aided Design, ICCAD 2004, San Jose, CA, USA, 7–11 November 2004; pp. 35–40. [\[CrossRef\]](#)
11. Tang, X. Large-scale computing systems workload prediction using parallel improved LSTM neural network. *IEEE Access* **2019**, *7*, 40525–40533. [\[CrossRef\]](#)
12. Borghesi, A.; Bartolini, A.; Lombardi, M.; Milano, M.; Benini, L. Predictive Modeling for Job Power Consumption in HPC Systems. In Proceedings of the High Performance Computing—31st International Conference, ISC High Performance 2016, Frankfurt, Germany, 19–23 June 2016; Volume 9697, pp. 181–199. [\[CrossRef\]](#)
13. Sirbu, A.; Babaoglu, O. Power consumption modeling and prediction in a hybrid CPU-GPU-MIC supercomputer. In Proceedings of the Euro-Par 2016: Parallel Processing: 22nd International Conference on Parallel and Distributed Computing, Grenoble, France, 24–26 August 2016; pp. 117–130.
14. Bugbee, B.; Phillips, C.; Egan, H.; Elmore, R.; Gruchalla, K.; Purkayastha, A. Prediction and characterization of application power use in a high-performance computing environment. *Stat. Anal. Data Min. ASA Data Sci. J.* **2017**, *10*, 155–165. [\[CrossRef\]](#)
15. Hu, Q.; Sun, P.; Yan, S.; Wen, Y.; Zhang, T. Characterization and prediction of deep learning workloads in large-scale gpu datacenters. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MO, USA, 14–19 November 2021; pp. 1–15.
16. O'Brien, K.; Pietri, I.; Reddy, R.; Lastovetsky, A.L.; Sakellariou, R. A Survey of Power and Energy Predictive Models in HPC Systems and Applications. *ACM Comput. Surv.* **2017**, *50*, 1–38. [\[CrossRef\]](#)
17. Antici, F.; Yamamoto, K.; Domke, J.; Kiziltan, Z. Augmenting ML-based Predictive Modelling with NLP to Forecast a Job's Power Consumption. In Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, 12–17 November 2023; pp. 1820–1830. [\[CrossRef\]](#)
18. Antici, F.; Ardebili, M.S.; Bartolini, A.; Kiziltan, Z. PM100: A Job Power Consumption Dataset of a Large-scale Production HPC System. In Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, 12–17 November 2023; pp. 1812–1819. [\[CrossRef\]](#)
19. Fahad, M.; Shahid, A.; Manumachu, R.R.; Lastovetsky, A. A Comparative Study of Methods for Measurement of Energy of Computing. *Energies* **2019**, *12*, 2204. [\[CrossRef\]](#)
20. Shahid, A.; Fahad, M.; Manumachu, R.R.; Lastovetsky, A.L. Improving the accuracy of energy predictive models for multicore CPUs by combining utilization and performance events model variables. *J. Parallel Distrib. Comput.* **2021**, *151*, 38–51. [\[CrossRef\]](#)
21. Intel Inc. Running Average Power Limit Energy Reporting. 2022. Available online: <https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html> (accessed on 12 March 2024).
22. Mei, X.; Chu, X.; Liu, H.; Leung, Y.; Li, Z. Energy efficient real-time task scheduling on CPU-GPU hybrid clusters. In Proceedings of the 2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, 1–4 May 2017; pp. 1–9. [\[CrossRef\]](#)
23. Chau, V.; Chu, X.; Liu, H.; Leung, Y. Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems. In Proceedings of the Eighth International Conference on Future Energy Systems, e-Energy 2017, Shatin, Hong Kong, China, 16–19 May 2017; pp. 1–11. [\[CrossRef\]](#)
24. Wang, Q.; Mei, X.; Liu, H.; Leung, Y.; Li, Z.; Chu, X. Energy-Aware Non-Preemptive Task Scheduling With Deadline Constraint in DVFS-Enabled Heterogeneous Clusters. *IEEE Trans. Parallel Distributed Syst.* **2022**, *33*, 4083–4099. [\[CrossRef\]](#)

25. Hsu, C.; Feng, W. A Power-Aware Run-Time System for High-Performance Computing. In Proceedings of the ACM/IEEE SC2005 Conference on High Performance Networking and Computing, Seattle, WA, USA, 12–18 November 2005; p. 1. [\[CrossRef\]](#)
26. Freeh, V.W.; Lowenthal, D.K.; Pan, F.; Kappiah, N.; Springer, R.; Rountree, B.; Femal, M.E. Analyzing the Energy-Time Trade-Off in High-Performance Computing Applications. *IEEE Trans. Parallel Distrib. Syst.* **2007**, *18*, 835–848. [\[CrossRef\]](#)
27. Fraternali, F.; Bartolini, A.; Cavazzoni, C.; Benini, L. Quantifying the Impact of Variability and Heterogeneity on the Energy Efficiency for a Next-Generation Ultra-Green Supercomputer. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 1575–1588. [\[CrossRef\]](#)
28. Auweter, A.; Bode, A.; Brehm, M.; Brochard, L.; Hammer, N.; Huber, H.; Panda, R.; Thomas, F.; Wilde, T. A Case Study of Energy Aware Scheduling on SuperMUC. In Proceedings of the Supercomputing—29th International Conference, ISC 2014, Leipzig, Germany, 22–26 June 2014; Volume 8488, pp. 394–409. [\[CrossRef\]](#)
29. Aupy, G.; Benoit, A.; Robert, Y. Energy-aware scheduling under reliability and makespan constraints. In Proceedings of the 19th International Conference on High Performance Computing, HiPC 2012, Pune, India, 18–22 December 2012; pp. 1–10. [\[CrossRef\]](#)
30. Kumbhare, N.; Marathe, A.; Akoglu, A.; Siegel, H.J.; Abdulla, G.; Hariri, S. A Value-Oriented Job Scheduling Approach for Power-Constrained and Oversubscribed HPC Systems. *IEEE Trans. Parallel Distributed Syst.* **2020**, *31*, 1419–1433. [\[CrossRef\]](#)
31. Etinski, M.; Corbalán, J.; Labarta, J.; Valero, M. Parallel job scheduling for power constrained HPC systems. *Parallel Comput.* **2012**, *38*, 615–630. [\[CrossRef\]](#)
32. Etinski, M.; Corbalán, J.; Labarta, J.; Valero, M. Optimizing job performance under a given power constraint in HPC centers. In Proceedings of the International Green Computing Conference 2010, Chicago, IL, USA, 15–18 August 2010; pp. 257–267. [\[CrossRef\]](#)
33. Raffin, G.; Trystram, D. Dissecting the Software-Based Measurement of CPU Energy Consumption: A Comparative Analysis. *IEEE Trans. Parallel Distrib. Syst.* **2025**, *36*, 96–107. [\[CrossRef\]](#)
34. David, H.; Gorbato, E.; Hanebutte, U.R.; Khanna, R.; Le, C. RAPL: Memory power estimation and capping. In Proceedings of the 2010 International Symposium on Low Power Electronics and Design, 2010, Austin, TX, USA, 18–20 August 2010; pp. 189–194. [\[CrossRef\]](#)
35. Bodas, D.; Song, J.J.; Rajappa, M.; Hoffman, A. Simple power-aware scheduler to limit power consumption by HPC system within a budget. In Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing, E2SC’14, New Orleans, LA, USA, 16–21 November 2014; pp. 21–30. [\[CrossRef\]](#)
36. Ellsworth, D.A.; Malony, A.D.; Rountree, B.; Schulz, M. Dynamic power sharing for higher job throughput. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2015, Austin, TX, USA, 15–20 November 2015; pp. 1–11. [\[CrossRef\]](#)
37. Schöne, R.; Ilsche, T.; Bielert, M.; Velten, M.; Schmidl, M.; Hackenberg, D. Energy Efficiency Aspects of the AMD Zen 2 Architecture. In Proceedings of the IEEE International Conference on Cluster Computing, CLUSTER 2021, Portland, OR, USA, 7–10 September 2021; pp. 562–571. [\[CrossRef\]](#)
38. Bhattacharya, A.A.; Culler, D.E.; Kansal, A.; Govindan, S.; Sankar, S. The need for speed and stability in data center power capping. *Sustain. Comput. Inform. Syst.* **2013**, *3*, 183–193. [\[CrossRef\]](#)
39. Khemka, B.; Friese, R.D.; Pasricha, S.; Maciejewski, A.A.; Siegel, H.J.; Koenig, G.A.; Powers, S.; Hilton, M.; Rambharos, R.; Poole, S. Utility maximizing dynamic resource management in an oversubscribed energy-constrained heterogeneous computing system. *Sustain. Comput. Inform. Syst.* **2015**, *5*, 14–30. [\[CrossRef\]](#)
40. Leal, K. Energy efficient scheduling strategies in Federated Grids. *Sustain. Comput. Inform. Syst.* **2016**, *9*, 33–41. [\[CrossRef\]](#)
41. Sensi, D.D.; Kilpatrick, P.; Torquati, M. State-Aware Concurrency Throttling. In Proceedings of the Parallel Computing is Everywhere, Proceedings of the International Conference on Parallel Computing, ParCo 2017, Bologna, Italy, 12–15 September 2017; Volume 32, pp. 201–210. [\[CrossRef\]](#)
42. Demmel, J.; Gearhart, A.; Lipshitz, B.; Schwartz, O. Perfect Strong Scaling Using No Additional Energy. In Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2013, Cambridge, MA, USA, 20–24 May 2013; pp. 649–660. [\[CrossRef\]](#)
43. Borghesi, A.; Di Santi, C.; Molan, M.; Seyedkazemi, M.; Mauri, A.; Guarrasi, M.; Galetti, D.; Cestari, M.; Barchi, F.; Benini, L.; et al. M100 ExaData: A data collection campaign on the CINECA’s Marconi100 Tier-0 supercomputer. *Sci. Data* **2023**, *10*, 288. [\[CrossRef\]](#)
44. Shoukourian, H.; Wilde, T.; Auweter, A.; Bode, A. Predicting the Energy and Power Consumption of Strong and Weak Scaling HPC Applications. *Supercomput. Front. Innov.* **2014**, *1*, 20–41. [\[CrossRef\]](#)
45. Chen, R.; Lin, W.; Huang, H.; Ye, X.; Peng, Z. GAS-MARL: Green-Aware job Scheduling algorithm for HPC clusters based on Multi-Action Deep Reinforcement Learning. *Future Gener. Comput. Syst.* **2025**, *167*, 107760. [\[CrossRef\]](#)
46. Lührs, S.; Rohe, D.; Schnurpfeil, A.; Thust, K.; Frings, W. *Flexible and Generic Workflow Management*; Advances in Parallel Computing; IOS Press: Amsterdam, The Netherlands, 2016; Volume 27, pp. 431–438. [\[CrossRef\]](#)
47. ARM. Workload Manager. Available online: <https://github.com/ARM-software/workload-automation> (accessed on 25 November 2024).

48. Iannone, F.; Ambrosino, F.; Bracco, G.; De Rosa, M.; Funel, A.; Guarnieri, G.; Migliori, S.; Palombi, F.; Ponti, G.; Santomauro, G.; et al. CRESCO ENEA HPC clusters: A working example of a multifabric GPFS Spectrum Scale layout. In Proceedings of the 2019 International Conference on High Performance Computing Simulation (HPCS), Dublin, Ireland, 15–19 July 2019; pp. 1051–1052.
49. Gebreyesus, Y.; Dalton, D.; Nixon, S.; De Chiara, D.; Chinnici, M. Machine Learning for Data Center Optimizations: Feature Selection Using Shapley Additive exPlanation (SHAP). *Future Internet* **2023**, *15*, 88. [\[CrossRef\]](#)
50. Blackford, L.S.; Choi, J.; Cleary, A.; D’Azevedo, E.; Demmel, J.; Dhillon, I.; Dongarra, J.; Hammarling, S.; Henry, G.; Petitet, A.; et al. *ScaLAPACK Users’ Guide*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 1997.
51. Loreti, D.; Artioli, M.; Ciampolini, A. Rollback-Free Recovery for a High Performance Dense Linear Solver With Reduced Memory Footprint. *IEEE Trans. Parallel Distrib. Syst.* **2024**, *35*, 1307–1319. [\[CrossRef\]](#)
52. Huang, K.; Abraham, J.A. Algorithm-Based Fault Tolerance for Matrix Operations. *IEEE Trans. Comput.* **1984**, *33*, 518–528. [\[CrossRef\]](#)
53. Colonna, M.; Loreti, D.; Artioli, M. C6EnPLS Dataset, 2024. Available online: <https://doi.org/10.5281/zenodo.14135916> (accessed on 29 April 2025).
54. Loreti, D.; Visani, G. Parallel approaches for a decision tree-based explainability algorithm. *Future Gener. Comput. Syst.* **2024**, *158*, 308–322. [\[CrossRef\]](#)
55. Mincoelli, G.; Marchi, M.; Giacobone, G.A.; Chiari, L.; Borelli, E.; Mellone, S.; Tacconi, C.; Cinotti, T.S.; Roffia, L.; Antoniazzi, F.; et al. UCD, Ergonomics and Inclusive Design: The HABITAT Project. *Adv. Intell. Syst. Comput.* **2019**, *824*, 1191–1202. [\[CrossRef\]](#)
56. Calori, G.; Briganti, G.; Ubaldi, F.; Pepe, N.; D’Elia, I.; Mircea, M.; Marras, G.F.; Piersanti, A. Implementation of an On-Line Reactive Source Apportionment (ORSA) Algorithm in the FARM Chemical-Transport Model and Application over Multiple Domains in Italy. *Atmosphere* **2024**, *15*, 191. [\[CrossRef\]](#)
57. Chesani, F.; Ciampolini, A.; Loreti, D.; Mello, P. Abduction for Generating Synthetic Traces. In Proceedings of the Business Process Management Workshops—BPM 2017 International Workshops, Barcelona, Spain, 10–11 September 2017; Volume 308, pp. 151–159. [\[CrossRef\]](#)
58. Loreti, D.; Chesani, F.; Ciampolini, A.; Mello, P. Generating synthetic positive and negative business process traces through abduction. *Knowl. Inf. Syst.* **2020**, *62*, 813–839. [\[CrossRef\]](#)
59. Alman, A.; Maggi, F.M.; Montali, M.; Patrizi, F.; Rivkin, A. Monitoring hybrid process specifications with conflict management: An automata-theoretic approach. *Artif. Intell. Med.* **2023**, *139*, 102512. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.