HORIZON-EUROHPC-JU-2021-COE-01

CEEC

Centre of Excellence in Exascale CFD

Best Practice Guide

Harvesting energy consumption on European HPC systems: Sharing Experience from the CEEC project

EuroHPC
Joint Undertaking

Copyright© 2023 – 2026 The CEEC Consortium Partners

Acknowledgment:

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Sweden, Germany, Spain, Greece, and Denmark under grant agreement No 101093393.

The opinions of the authors expressed in this document do not necessarily reflect the official opinion of the CEEC partners nor of the European Commission.

Gülçin Gedik[1], Kajol Kulkarni[2], Yanxiang Chen[1], Daniel Kempf[3], Samuel Kemmler[2], Dimitris Papageorgiou[4], Damaskinos Konioris[4], Sally Kiebdaj[5], Julita Corbalan[6], Harald Köstler[2], and Roman Iakymchuk[1]

[1]Umeå University, Sweden. {ggedik,ychen,riakymch}@cs.umu.se
[2]Friedrich-Alexandre-University Erlangen-Nuremberg, Germany. {kajol.kulkarni, samuel.kemmler,harald.koestler}@fau.de
[3]University of Stuttgart, Germany. daniel.kempf@iag.uni-stuttgart.de
[4]Aristotle University of Thessaloniki, Greece. {dpapageor,dkonioris}@meng.auth.gr
[5]HLRS, University of Stuttgart, Germany. sally.kiebdaj@hlrs.de
[6]Barcelona Supercomputing Center, Spain. julita.corbalan@bsc.es

August 12, 2024

Contents

1 Executive Summary

In this document, the EuroHPC JU Center of Excellence in Exascale CFD (CEEC) aims to provide users/ application developers with a brief overview of possibilities, limitations, and best practices for measuring energy consumption on European HPC systems[1]. CEEC is working to reduce the energy footprint of its consortium codes on such systems by applying novel algorithmic solutions. However, in initially exploring options for collecting energy measurements on both local and European HPC systems, we found no single approach for energy measurements and the process of taking these measurements comparatively more difficult than measuring time-to-solution with e.g. basic start-end time calls. This difficulty often stems from a requirement for privileged access to specific hardware counters. Mitigation strategies for this restriction exist and enable users to collect the energy metric, but they are not widely known. We describe these strategies in Section 3 followed by concrete examples from CEEC on how to harvest the energy measurements described in Section 5. We believe this will help to increase awareness and thus utilization of energy consumption measurements in the application development process.

Furthermore, we describe several other important issues: 1) granularity and overhead of measurements since $energy = power \times time$ and 2) what is included (there multiple factors) in the number delivered by a tool/ framework/ workload manager. We strive to be concise and precise aiming to provide a glimpse of energy measurement methods as well as many references for further exploration. Our takeaway messages are

1. The community/ data centers need to facilitate energy measurements on the European HPC systems and teach the community how to conduct such measurements.
2. The community/ data centers need to provide transparent and easy-to-use guides on each (at least large) European HPC system, outlining the ways to collect energy measurements[2].

In CEEC, we are taking the first steps towards spreading these messages, aiming to create a larger consortium including experts and data centers, who can contribute to and update this document. Explore and stay tuned!

[1] This is where we have access, but approaches and techniques can virtually be applied on any system.

[2] We would like to point out that CASTIEL2 – Coordination and Support for National Competence Centers and Centers of Excellence on a European Level Phase 2 – aims at providing more specific information per each EuroHPC JU HPC systems *soon*.

2 Introduction

The energy consumption constraint for using thousands of computers connected together (called supercomputing or large-scale computing) [14, 42, 33] encourages scientists to revise the architecture and design of hardware, frequently used linear algebra algorithms, and now applications. The main idea is to ensure the energy-efficiency (aka sustainability) of computational expenses while applying the 'lagom' principle, which is the Swedish ethos of balance or 'just enough', especially when it comes to working or storage precision in scientific computations. Similarly, ETP4HPC's Strategic Research Agenda [33] introduces the term energy-to-solution and connects it to the positive outcome of the mixed-precision strategies applied to solvers. However, the question of how to harvest energy-to-solution and make it a part of standard high performance computing (HPC) metrics alongside performance, scalability, and efficiency of resource utilization remains largely unanswered.

Therefore, this document aims to bridge this gap and provide the wider community a glimpse into energy consumption best practices and (user) experience. We intend to provide a brief but vital study to equip users in the European HPC landscapes, and hopefully others as well, with a sufficient understanding to start measuring energy. From the CEEC experience, we observe that the trend of energy-efficient computing has reached applications that are rather slow to adopt it up due to their long-standing development (often over decades), and also their complex and sophisticated code with thousands if not millions of lines. Thus, we believe that the time for including energy-to-solution in standard HPC metrics and facilitating its harvesting is now.

The remainder of this document is organized as follows: Section 3 provides an overview of possibilities for measuring energy consumption and its classification. This includes necessary information regarding the hardware measurement methods present in different architectures and both tools and frameworks built around these counters. Section 4 presents the metrics to enable and facilitate the comparison of energy efficiency on different systems. Section 5 shows our experience with energy measurement methods on the European systems with examples from CEEC. Finally, Section 6 reflects upon our experience and shares a few tips for a quickly starting with measuring energy consumption.

3 How To Measure Energy Consumption?

In high performance computing, we are accustomed to various optimisation techniques that help us to get the most out of hardware to boost algorithms' performance. Thus, HPC can be viewed as an optimisation field of fitting algorithms on hardware and making the best usage of that hardware with a single target in mind – performance. Unlike measuring performance though, measuring energy is not straightforward, and a several tools exist. Moreover, measuring energy on the EuroHPC JU machines can be a challenge since there is no single tool or method that supports all types of systems. Thus, optimization in terms of energy consumption may be contre intuitive and contrary to currently established optimization objectives: for some programs, performance correlates with energy consumption; for others, computing a result slower or using lower precision may lead to potentially large energy savings. In this document, we collect best practices and existing tools, as well as share our own experience with reliably measuring energy on different platforms. Figure 1 provides a bird eye view of the tools and frameworks that are going to be examined, and it presents the spectrum from physical methods to software techniques.

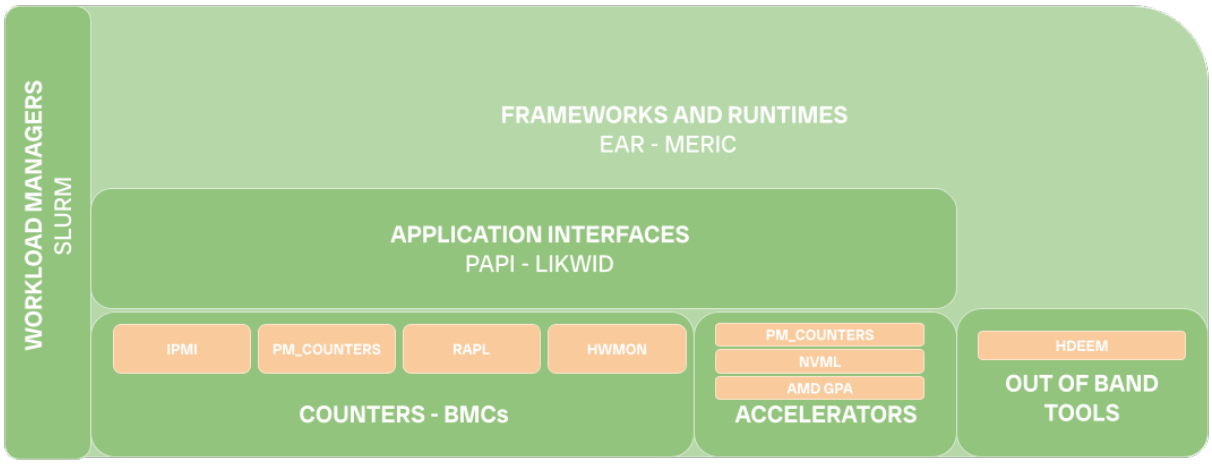

Figure 1: Hierarchical relation between energy measurement methods.

3.1 Hardware Counters

This section provides a brief introduction to the energy/ power measurement counters present in various architectures as summarized in Table 1. As mentioned above, this document aims to provide essential information on measuring energy efficiency of applications over their run-time. Therefore, we believe that the developers who seek to have a fine-grained profile of their application, such as profiling the consumption of DRAM, will find the provided overview and the references useful.

3.1.1 RAPL

Running Average Power Limit (RAPL) [29] is a set of tools that uses Model Specific Registers (MSRs), which are specific to each processor model, to create an interface collecting different power domains such as core, package, DRAM, and system, as depicted in Figure 2 . Therefore, they can be programmed through these registers in order to enforce power consumption constraints to stay within a thermal budget, achieve specific battery life goals, or control power-performance balances.

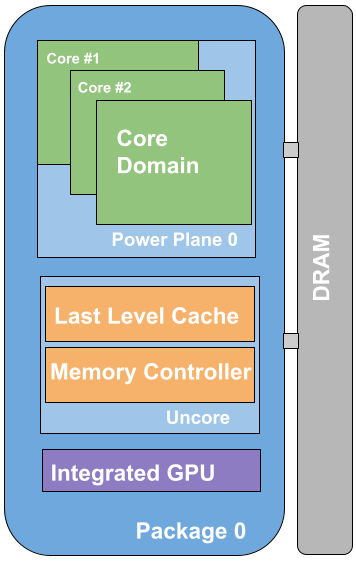

Figure 2: RAPL Power Domains.

Each RAPL domain supports MSR interfaces for Watt-based power limiting, offering status information for energy consumption in model specific energy units, generally micro Joules, and insights into performance effects due to power limits.

Intel RAPL operates on the socket level and provides hierarchical control over different power domains, and it can also function as a power measurement tool, as suggested by Hackenberg et al.[20], transitioning from modeling to actual measurements.

Schone, Ilsche et al. [43] in their study mention that with the Zen architecture, AMD replaced Application Power Management (APM) with RAPL. The way it's done seems similar to what Intel does, but AMD uses different MSRs. Unlike Intel, AMD only mentions registers for checking package and core domain power usage. However, AMD

provides per-core spatial resolution, while Intel offers per-package for its core domain (pp0). Additionaly, Intel changed from a model-based system to a measurement-based one with the Haswell architecture, but AMD still uses a model-based approach [36].

3.1.2 NVIDIA GPUs

When discussing HPC systems, it is essential to consider their inherent heterogeneity. As outlined in the SCALABLE Deliverable - D2.3 [12], one of the limitations we face is the accuracy of hybrid measurements. There is no common interface between GPUs and CPUs, except when the GPU is integrated (such as with RAPL). Consequently, we need to couple measurements from two different sources. For NVIDIA GPUs, energy consumption information can be obtained through the NVML API [39] using device queries. According to the API documentation, energy consumption data is provided since the last driver reload and includes the associated circuitry (e.g., memory). The accuracy and sampling intervals of these measurements depend on the specific architecture. For Grace SoCs, both CPU and GPU power measurements are available.

3.1.3 IPMI

The Intelligent Platform Management Interface (IPMI) [23] is a collection of interfaces to manage and monitor computer systems independent of the host system with the help of Baseboard Management Controllers (BMCs). It provides access to node power consumption monitoring. Various open-source software solutions exist for in-band and out-of-band IPMI sensors data collection [17] [27]. Hackenberg et al. [18] showed that power data at a low sampling rate is accurate although energy consumption might not be as precise. However, the issue of low sampling rates is particularly pertinent due to IPMI's recognized high operations overhead.

3.1.4 PM_Counters

On Cray Systems, like LUMI and Dardel at KTH, power monitoring is available both for pure CPU systems and hybrid CPU-GPU systems [34]. Counters that show instantaneous power draw and cumulative energy consumption in Joules on node level are exposed to the non-privileged users through `sysfs` with the help of `cray-pat` in directory `/sys/cray/pm_counters` on compute nodes. Since the counters update atomically every 100ms [21], it is important to match the energy consumption data with the corresponding measurement to ensure that a consistent set of values are obtained. Therefore, there is a second counter called 'freshness' incorporated in the measurements. This counter should be accessed both before and after obtaining energy measurements to guarantee accuracy. These measurements can be accessed also through extended prototypes of the Score-P performance measurement infrastructure and Vampir application performance monitoring visualiser.

3.1.5 Overview

Table 1 summarizes the variety of the counters presented in this section that are available on European HPC systems. We emphasize that considering energy-to-solution as a performance metric is still in development and therefore lacks a unified measurement approach.

Each measurement approach summarized in Table 1 comes with a unique advantage that is taken into account while designing the tools presented in Section 3.2. In this section, we presented RAPL for Intel and AMD processors, which allows for monitoring and limiting power across various domains like cores, socket and DRAM. It provides hierarchical control and actual measurements, additionally AMD's version offers per-core monitoring. For NVIDIA GPUs, the NVML API provides energy consumption data, although the accuracy depends on the specific GPU architecture. We also mentioned IPMI for monitoring power usage independently of the host system, though it has limitations due to a low sampling rates. Finally, PM counters on Cray systems allow monitoring of both CPU and hybrid CPU-GPU systems, with mechanisms in place to ensure accurate data collection.

Table 1: Summarizing Properties of Hardware Counters.

Type	Domains	Granularity	Measurement Type	Metric	Vendor Specific	Overhead[1]
RAPL	Socket(PP0) All cores DRAM System Integ. GPU (AMD) Per Core	1 ms	Counter	mili-Joules	✓	Low
NVML	Socket	-[2]	Counter	mili-Joules	✓	-
IPMI	Node	-[3]	BMC	Watt	✗	High
PM_COUNTERS	Node Network Card GPU	100 ms	Counter	Joules	✓	Medium
HDEEM	Blade Node DRAM	1 ms	FPGA BMC	Watt	✗	Low

[1] Short measurement intervals.
[2] Architecture dependant.
[3] BMC sensor dependant.

3.2 Tools

This section presents high-level measurement tools suitable for application wide energy and power analysis, leveraging the hardware counters previously described. Each of these tools coupled with the hardware counters covers different research objectives such as spatial or temporal granularity. Figure 3 represents how the mentioned hardware relates to the physical domain.

3.2.1 Perf

Perf or Perf Tools [10] is an integrated performance analysis tool within the Linux kernel that offers support for a range of counters, including hardware counters for energy. The `perf_event_open` system call is employed by Perf to access and manage these performance counters effectively. Some of the tools presented here are developed on top of Perf Tools, but they provide simpler solutions because development with Perf Tools requires some level of attention and more information regarding the measurements. For instance, developers working with `perf_event_open` should pay attention to the detailed implementation of the counters, such as the wrap around time, which requires advanced knowledge of the architecture of the counters. Additionally, Perf reads energy consumption directly from the hardware counters, therefore it requires users to have root privilege (pseudo-sudo access) or to set the perf event paranoid value to 0 or lower, which raises the security concerns.

3.2.2 HDEEM

High Definition Energy Efficiency Monitoring (HDEEM) [19] allows high accuracy energy/ power analyses of applications at high sampling rate with the help of inband measurements as well as with an out of band measurement with an FPGA (Field Programmable Gate Array) tool.

On the hardware level, HDEEM uses analog filters to overcome aliasing issues and noise. Additionally, HDEEM combines an FPGA and the existing BMC hardware instead of an autonomous measurement board. Its API also supports IPMI and SLURM for gathering data at a low temporal granularity, allowing administrators to use their existing tools for gathering information. Its scalable measurement readout interface is based on energy values, which allows us to read values with different temporal granularity without losing information from missed samples. For fine-grained measurements, it provides the samples via the PCIe bus, which is faster then going over 10/100 MBit Ethernet or USB.

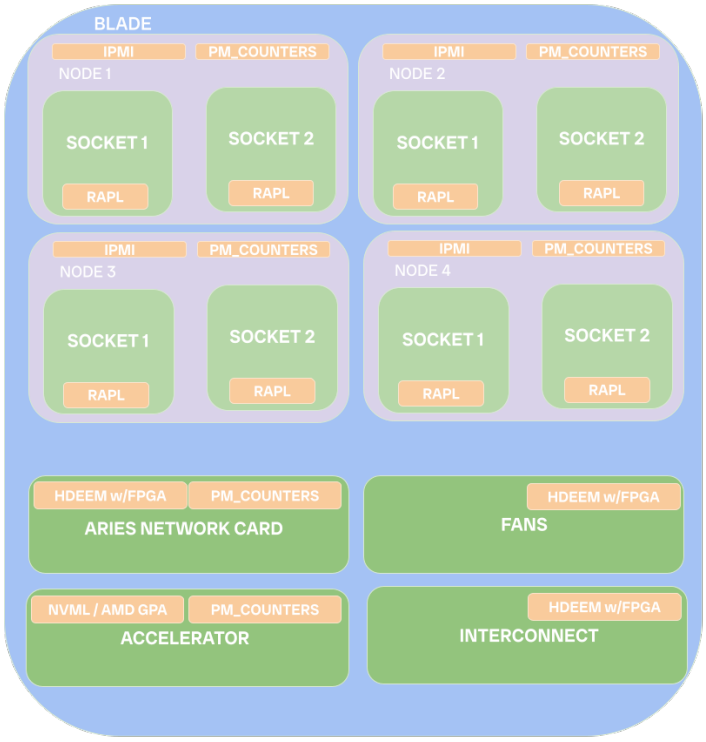

Figure 3: A generic blade to describe spatial granularity of components.

3.2.3 PAPI

Performance API (PAPI) library [35] is a cross platform library supporting external power meters as well as the internal ones. Its main advantage is the ability to measure a diverse set of hardware with a common interface [46]. Therefore, there exists multiple tools built on top of PAPI such as TAU [40], HPCToolkit [22], Vampir [15], etc. It is possible to read power and energy metrics via Intel RAPL for SandyBridge and its successors, as well as the NVML for NVIDIA GPUs.

3.2.4 LIKWID

LIKWID stands for 'Like I Knew What I'm Doing' [5] and is a Linux command line performance tool suite that offers users control over task affinity and the ability to plot live performance metric graphs along with other features. RAPL readings are accessible through `likwid-powermeter` for comprehensive measurements and `likwid-perfctr` for hardware performance event counting which can be used for the energy counters exposed as events on other architectures such as ARM, NVIDIA and more. Additionally, users can couple powermeter metrics with core temperature metrics for analysis.

3.3 Frameworks and Runtime Systems

This section aims to provide an overview of available ready-to-use runtime systems and APIs in order to monitor and optimize energy consumption with two different methods. Figure 1 shows how these systems relate to the physical measurement devices and tools previously described.

3.3.1 EAR

Energy Aware Runtime (EAR) [3] is a collection of tools offering system power consumption and job energy accounting as well as runtime energy optimization, cluster power management, and both cluster and node power capping without any application modification. Note that EAR requires to be installed with root privileges on the Linux kernel. From the user perspective, it aims to be machine agnostic. Additionally, it provides extensive reporting mechanisms. In order to provide energy optimizations, EAR takes advantage of its runtime library [25] , which selects an optimal CPU frequency based on the energy policy and application characteristics.

3.3.2 MERIC

The MERIC library [45] evaluates application's behavior in terms of resource consumption, and controls hardware and runtime parameters such as the Dynamic Voltage and Frequency Scaling (DVFS), Uncore Frequency Scaling (UFS), and number of OpenMP threads through external libraries. Applications can be instrumented using the MERIC manual instrumentation to analyze each part of the code separately. The energy measurements are provided by the HDEEM system, or by the RAPL counters. MERIC, coupled with another tool RADAR, generates a LaTeX report and a MERIC configuration file that outputs the best settings for the evaluated parameters.

3.3.3 LLview

LLview [37] consists of software components designed to monitor clusters managed by a resource manager and scheduler system. Its Job Reporting module offers detailed insights into every job currently running on the system by connecting to various sources within the system to gather and display data through a web portal. For example, the resource manager provides information about the jobs, while additional daemons collect extra data from compute nodes with minimal overhead, as metrics are updated every few minutes. Some of the metrics tracked include statistics about the GPU usage, memory usage rate, power, and temperature. The LLview portal not only links performance metrics to specific jobs but also allows users to access tables with aggregated performance information, timeline graphs detailing key metrics over the course of a job, and comprehensive reports in both interactive and PDF formats. Additionally, this efficient interface *updates performance metrics every minute* to provide real-time data, ensuring minimal system overhead. For instance, the values are obtained for each GPU from NVIDIA's Data Centre GPU Manager `dcgm` every minute.

3.4 Workload Manager: A SLURM Example

SLURM [1] stands out especially with its ease of use for application wide measurements and its availability. It provides a common interface for different socket level infrastructure options, such as `IPMI`, `RAPL` or `PM_Counters` and more; and, it is constantly developing.

Like EAR, the tool does not ask for any root privileges, since it has been already installed with them.

Users and system administrators, who want to measure energy consumption with the help of SLURM, should remember that the plug-in responsible for energy accounting and monitoring should be enabled in `slurm.conf`. This configuration file also details the socket-level measurement approach used. For instance, we verified that on the EuroHPC JU system LUMI, this plug-in uses the `PM_Counters` provided by HPE Cray Systems with the help of a BMC. Therefore, we know that the measurements are node level. Thus, the isolation of processes other than the targeted application should be addressed, taking into account the inherent restriction of the placement of the counters as shown in Figure 3. We have to make sure that no other jobs are intervening on node level measurements, with the least OS noise possible.

There is also the External Sensors Plugin that collects energy and temperature data generated out-of-band by an external system manager such as Nagios, or external sensors such as wattmeters.

4 Metrics

As we discussed, there is a need to establish an energy metric and to include it in the standard HPC set of metrics along with time-to-solution and its derivations, in order to compare different systems and configurations. We discuss a few variants below.

4.1 FLOPs per Watt & Bytes per Watt

Green500 [44] considers FLOPS per Watt as its primary energy efficiency metric to rank supercomputers since its introduction in 2007. The main goal is to track and improve the energy consumed per instruction as FLOPs per Watt. This metric can also be reduced to FLOP per Joule. In the same regard, Bytes per Watt indicates the energy efficiency of the memory systems.

4.2 Energy Delay Product

$$ED^wP = E * T^w$$

Energy Delay Product (EDP) [31], sometimes referred as ED^2P , is a metric that takes into account the runtime (T) and the energy (E) consumption of an application. Therefore it is often used to evaluate trade-offs (D) between power saving techniques and performance. A form of weighted EDP is used to prioritize or weight (w), performance over energy savings, therefore lower is better.

4.3 Energy and Carbon Footprint

The carbon footprint of an application [41] can be explained as the carbon intensity of the energy source per unit of energy. Therefore, one of the main ways to reduce the carbon footprint of an application is through lowering its energy consumption. This metric is also relatively new and needs to be estimated most of the time rather than being measured.

4.4 Energy-normalized Performance Index (EPID)

Within the FLEXI framework [30], which is being developed in the numerical research group at the University of Stuttgart, Performance Index (PID) is traditionally used to measure and compare the performance between systems. It is defined as

$$\text{PID} = \frac{\text{wall-time} \times \#\text{ranks}}{\#\text{RK-stages} \times \#\text{DOF}}, \tag{1}$$

and describes the average time it takes a computational rank to update a single degree of freedom (DOF) for one Runge–Kutta stage. An alternative interpretation of the PID is as an estimate of the normalized required time-to-solution, i.e. a low PID indicates good performance, and an increase in the PID indicates decreasing performance.

With the increasing use of accelerator-based HPC hardware architectures, a rank based comparison of the PID between traditional CPU-based HPC systems and accelerator-based HPC systems, where a rank is a traditional CPU rank or a single accelerator device like a GPU, loses its significance. Due to the sheer differences in computing power per rank, this PID can vary by orders of magnitude. For the comparison of different HPC architectures, we have decided to consider an energy-normalized PID (EPID) in order to take this inequality of the total energy consumption into account.

This EPID is defined as

$$\text{EPID} = \frac{\text{wall–time} \times \text{power}}{\#\text{RK-stages} \times \#\text{DOF}} = \underbrace{\frac{\text{power}}{\#\text{ranks}}}_{P_{\text{rank}}} \times \text{PID}, \tag{2}$$

and is designed to be a physical quantity with a unit of energy. It describes the energy required to compute the time update for a single DOF on the specific computing hardware and can be interpreted as the PID normalized by the specific power required per rank P_{rank}. The EPID is not yet applicable to all HPC systems, as the energy data for this is not accessible to the user everywhere. However, computing centers are actively striving to make this data available to users, so that the EPID represents a future-oriented performance parameter.

4.5 Energy Efficiency

The energy efficiency can be defined as the ratio of the energy consumed by parallel simulation to the energy consumed by a baseline simulation, which is run with the minimum number of racks possible. This allows users and developers for an assessment of energy efficiency in varying numbers of ranks. The energy efficiency can be computed as

$$\text{Energy Efficiency} = \frac{\text{Energy}_P}{\text{Energy}_{P_0}}, \tag{3}$$

where Energy_P is the energy required for a simulation with P ranks and Energy_{P_0} is the energy required for a simulation with the minimum number of ranks P_0. There is a clear indication that as parallel efficiency decreases, the simulation becomes more energy inefficient, with the degree of inefficiency being roughly proportional to the decrease of parallel efficiency.

5 CEEC examples of measuring energy-to-solution

This section showcases success stories from four European HPC systems using CEEC examples. For more details, we would like to refer to the CEEC 'Deliverable D2.3 – Intermediate performance and energy efficiency evaluation result' [11]. It is worth noting that one of CEEC's key performance indicators (KPIs) is `energy-to-solution`. Table 2 lists possible methods and tools CEEC code owners use for energy measurements on different European HPC machines. It is important to note that the codes are still working on harvesting energy measurements as development toward the proposed CEEC lighthouse cases[3] progresses. Furthermore, Appendix A outlines a range of methods and tools available for measuring energy on the European supercomputers. Each supercomputer is equipped with different tools and techniques to measure energy, catering to the specific requirements and capabilities of their respective supercomputing infrastructure.

Table 2: Energy measurement of the CEEC consortium codes on various European HPC systems using the most common methods and tools on each individual system.

Code	**Machine**	**Framework**	**Counter**
FLEXI [28]	HAWK Apollo 9000 Apollo 6500	HPE-HPCM [4]	CMC [5] PDU [6]
waLBerla [47] Nekbone [9]	LUMI-G LUMI-C	sacct	pm_counters
Neko [26]	HPE Cray EX Alvis Dardel	rocm nvidia-smi cray-pat	AMD GPU NVIDIA GPU AMD RAPL
NekRS/ Nek5000 [32]	JUWELS Booster	LLview	NVIDIA dcgm

[4] Customized HPE - High Perfromance Cluster Management.
[5] Chasis Management Controller.
[6] Metered Power Distribution Units.

5.1 FLEXI on HAWK

FLEXI and its GPU-based derivative GÆLEXI [28] are high-order accurate flow solvers based on the discontinuous Galerkin (DG) spectral element method. The local communication stencil of the DG method and the efficient parallelization allow for excellent scaling properties.

Within the CEEC project, FLEXI has collaborated with HLRS and HPE to evaluate the energy efficiency at HLRS on their CPU- and GPU-based systems. We has received assistance from the application support team at HLRS to collect the energy data of the simulations. This additional intermediate step will no longer be necessary in the future, as HLRS is actively working on making energy consumption available to the user via the project management tool, among others.

In this initial energy efficiency investigation, the CPU-based FLEXI solver on HAWK was compared with the GPU-based solver GÆLEXI on HAWK-AI with regard to the

[3] More information about the CEEC lighthouse cases at `https://ceec-coe.eu/lighthouse-cases/`

EPID as introduced in Section 4.4. For this purpose, a practical example of a rectilinear transonic compressor cascade equipped with NASA Rotor 37 profiles was used. This compressible case includes all relevant software functionalities, including the necessity of shock capturing. The simulations are initialized with a precomputed converged flow state and are advanced for a total of 8 characteristic time units $t^* = tu_\infty/c$, where t^* is defined with respect to the inflow velocity u_∞ and the chord length c. Table 3 shows that the GPU accelerated system exhibits a 56% reduction in energy consumption [28].

Table 3: Performance and energy measurements using the example of a rectilinear transonic compressor cascade equipped with NASA Rotor 37 profiles, more details in [28].

	P_{rank} [W]	PID [s]	EPID [J]	Walltime/t^* [s]	Energy/t^* [kWh]
GPU	448	4.58×10^{-9}	2.05×10^{-6}	9209	147
CPU	4.94	1.02×10^{-6}	5.06×10^{-6}	7538	339
Savings			59.5 %		56.8 %

5.2 waLBerla on LUMI-G

The waLBerla framework [47] is a widely applicable lattice Boltzmann simulation code from FAU, Erlangen-Nuremberg. Currently, it uses `pm_counters` through the SLURM energy plug-in to measure energy consumption on LUMI-G. The total energy consumed can be retrieved after the job has finished with `sacct` using the `--format` option and the desired field:

```
sacct --format="Account,JobID,JobName,ConsumedEnergy,NodeList" -j <jobId>
```

This command gives a report of each database mentioned in the above command at node level. It is important to note that energy is monitored at node level, indicating that only when a job is allocated exclusively to certain nodes can the energy usage measurements accurately represent the job's actual consumption. When using multiple nodes these measurements depict the energy utilized by all participating nodes but do not encompass the energy consumption of the interconnects or file system. For the energy measurement, we run the CEEC percolation/ porous media benchmark which is essentially a channel flow packed with two-way coupled particles, see Table 4. This benchmark covers all relevant software features required for the final lighthouse case – Localized erosion of

Table 4: Setup description and energy measurements of the percolation benchmark using one Graphics Compute Die of LUMI-G on an exclusively allocated node.

Cells	$256 \times 128 \times 128$
Time steps	1000
Total MLUP	4194
Total MDOF	113238
Repetitions	4
Wall time [s]	14.250
Total Energy [J]	12600
Energy efficiency [MLUPS/J]	0.333
Energy efficiency [MDOFS/J]	8.99
Standard deviation	0.041

an offshore wind-turbine foundation. We repeat the measurement five times, skip the first run, and report the mean along with the standard deviation since we experience the energy consumption is higher than in the other repetitions.

5.3 Nekbone on LUMI-C

In CEEC, Neko, Nek5000, and NekRS are three high-order, incompressible Navier-Stokes solvers based on the spectral element method. Results are provided on Nekbone [38], which is a mini-app capturing the basic structure and design of the extensive Nek5000 [32] software. Nekbone solves a standard Poisson equation using the Conjugate Gradient (CG) method with a simple multigrid preconditioner which is optional when compiling.

In order to measure energy-to-solution of Nekbone, we opted to use the energy accounting plugin provided by SLURM, owing to its simplicity. The energy consumption was retrieved after the job had finished with `sacct` using the `--format` option with the desired field `ConsumedEnergy` and specifying the job id. Table 5 reports the consumed energy in Joules of the entire Nekbone with double and mixed-precision versions. As a note, the mixed-precision version combines double and single precision, where single is only used in the CG loop. The LUMI-C partition is used with an exclusively allocated node, which is equipped with two AMD EPYC 7763 CPUs with 64 cores each running at 2.45 GHz. Each test is run five times, and the mean as well as the standard deviation (stddev) is computed. The reduction in energy-to-solution for the mixed-precision version correlates with the time-to-solution and notably shows better gain, confirming the efficiency of a mixed-precision algorithmic approach.

Table 5: Energy-to-solution in Joules of Nekbone for the case without preconditioner with different MPI ranks on the LUMI-C partition with a single exclusive node; more details in [9].

MPI ranks	32	stddev	64	stddev	128	stddev
Mixed	451.6	4.1%	653.6	2.4%	1089.4	1.0%
Double	990.6	3.9%	1424.8	4.5%	2061.2	3.2%
Savings	54.41%		54.12%		47.14%	

As a note, while conducting measurements on the node level, we need to keep in mind that these measurements do not take into account shared components like switches and cooling. Therefore, the collected energy numbers do not give us the full picture but rather provide a strong indication of the application's energy consumption.

5.4 Neko on Alvis, HPE Cray EX, and Dardel

Neko [24] is a portable framework for high-order spectral element based simulations on hexahedral meshes, mainly focusing on incompressible flow simulations. Karp et al. [26] conducted power measurements of Neko on the Flettner rotor case in a turbulent boundary layer, aiming to compare power efficiency of AMD MI250X and NVIDIA A100 GPUs. There are three main messages here: 1/ the standard deviation of the average power consumption per time step between these two GPUs is within 5 %; 2/ the computed energy consumption (power multiplied by the length of the time step) is nearly constant on GPUs. There are also numbers for the AMD EPYC 7742 CPU but those are higher than for GPUs as they include the entire node. 3/ The energy usage per time step remains

nearly constant with more GPUs, maintaining parallel energy efficiency above 80% for both GPUs and up to 90% for the MI250X, indicating minimal scaling penalties and competitive run times.

Additionally, Neko developers included a comparison between a pure CPU node against the one equipped with four AMD MI250X GPUs on the Dardel cluster at PDC-KTH, resulting in 3x higher power consumption of the latter one. To collect the power metrics, they used the power counters via `rocm/ nvidia-smi` for the GPUs and Cray Performance Analysis Tool (CrayPat) for the CPUs. Note that power consumption of the network and other peripherals are omitted.

Karp et al. [26] also highlights that obtaining accurate power numbers is a challenge as well as that there is a need for power/ energy measurements in order to make comparisons across heterogeneous systems.

5.5 NekRS/ Nek5000 on JUWELS Booster

Regarding energy efficiency for NekRS/ Nek5000 [16], the Jülich LLview profiling tool is used to track the energy consumption in JUWELS Booster. LLview provides a detailed report about various metrics, including the average GPU Power consumed by each GPU during the run. To compute the energy consumption of the simulation, the average power consumed by each GPU is multiplied by the number of GPUs utilized and the total walltime of the simulation. Figure 4a presents the energy consumption in kilowatt-hours (kWh) per 5000 timesteps for simulations with resolutions of 512^3, 1024^3, and 2048^3 using NVIDIA A100 GPUs across varying numbers of GPUs. Each increment in resolution increases the degree of freedom (DOFs) by a factor of eight compared to the previous resolution. The figure illustrates that the energy consumption scales approximately eight-fold with each increase in resolution which is the same as the factor of increase of DoFs.

Figure 4b depicts the parallel efficiency in relation to the energy ratio. This, along with the fact that NekRS can utilize 80-90% of the realizable peak memory bandwidth across multiple platforms while sustaining 1-2 TFLOPs (10-20% of the peak computational power of the most advanced GPUs), suggests that NekRS's parallel and energy efficiency is bandwidth-limited and constrained by communication overhead rather than by hardware (computational power) when a large number of GPUs is utilized, specifically when n/P drops below 2.5 million.

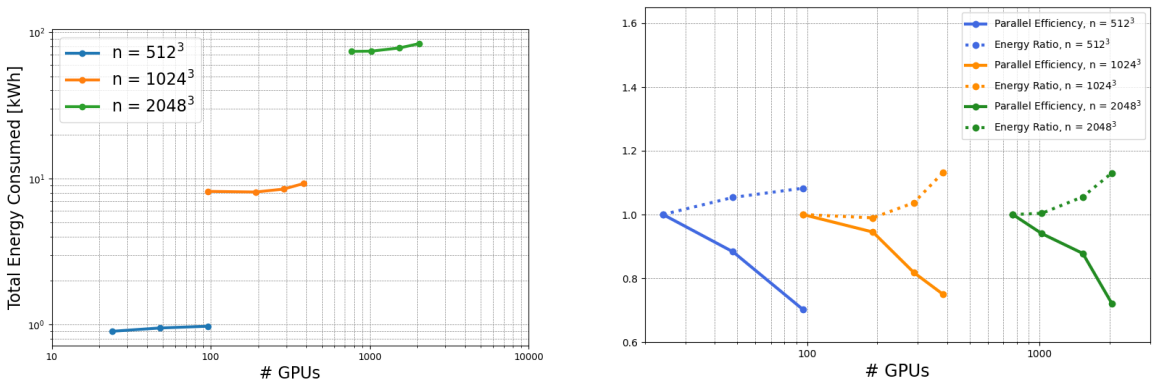

Figure 4: a) Energy consumption (kWh) for 5000 simulation timesteps for cases with resolutions 512^3, 1024^3 and 2048^3 on NVIDIA A100 (JUWELS Booster) GPUs across varying number of GPUs b) parallel efficiency and energy efficiency for the same cases.

6 Discussion

In this overview of experience and best practices, we provided a brief introduction to a wide range of energy measurement methods, their restrictions, common pitfalls, and mitigation strategies. We presented easily accessible tools and how these measurements are handled in different European HPC systems with examples from the CEEC project.

Despite its importance, the field of energy monitoring is still in its infancy and lacks standardization. Various tools offer different measurement possibilities. As seen from the examples above, the members of CEEC project have already started to take advantage of this diversity. We have started to examine available tools with respect to their ease of use as well as, coarse and fine granularity over temporal and spatial aspects of the application.

We observed that SLURM is the go-to way for newcomers who want to measure how much energy their application consumes over the job's runtime with minimal overhead. It doesn't require any root privileges and handles the underlying measurement infrastructure itself, leaving nothing to users but just the exclusive allocation of resources for the run. With this simple methodology, we were able to describe two cases in connection to the CEEC project. EAR is an additional alternative to the SLURM plugins that in fact can be coupled with SLURM, enabling energy measurements per job through a single line added to the bash script. In order to measure energy consumption of particular functions inside the code, both LIKWID and PAPI can be explored since they offer reduced developer time in addition to fine spatial and temporal granularity access with low overhead.

However, even with the best practices in place, generalizing results to different types of hardware and systems is a delicate task. To address this, we have introduced metrics used in CEEC projects as an example of how to standardize experimental results. During our research for Section 5, we found that the field lacks a convenient method for presenting metrics, making it harder for readers to grasp the implementation details. As a result, we recommend that researchers provide more detailed descriptions of their measurement techniques. These descriptions should include the scope and granularity of the measurements, such as whether they cover the node, core, or cooling equipment, and the intervals between measurements. Given the variety of tools and their interdependence, as shown in Figure 1, researchers should also specify the tools they used and their underlying methodologies. For example for SLURM finding this information is straightforward, as explained in Section 3.4, and providing such information is essential for ensuring the verifiability and reproducibility of research.

As future work, we plan to explore possible energy optimization strategies that hardware and software providers have to offer and how researchers may take advantage of them. We intend to engage both LIKWID and PAPI in our mixed-precision exploration. Furthermore, we consider to select 1-2 benchmarks from CFD codes and run them on all accessible HPC systems in order to determine which system is more suitable for energy-efficient CFD simulations.

Appendix A Tools and Methods to Measure Energy on European HPC Systems

As demonstrated, there are several system-dependent methods available for measuring energy. Table 6 shows most common and potential methods and tools that can be employed on several European HPC systems; the list of machines is not exhaustive and was formed as a result of our experiments or as a potential candidate for use. Let's begin with LUMI that supports SLURM energy account plug-in to measure energy on node level as explained in Section 3.4 and shown on two CEEC examples in Sections 5.2 and 5.3. Additionally, the system offers native access to CrayPat to measure energy consumption. CrayPat includes support for `cray_pm` and `cray_rapl` components with the help of PAPI, which is detailed in Section 3.2.2. The tool is available through `perftools` module [13].

Table 6: Energy measurements on European HPC systems.

HPC machine	**How to measure energy-to-solution**
LUMI-C/ LUMI-G	SLURM (pm_counters), CrayPat
Leonardo Booster/ Leonardo DGCP	Bull Energy Optimizer (IPMI and SNMP), Bull Dynamic Power Optimizer; NVIDIA `dcgm`
Marenostrum 4 and 5	EAR (RAPL and IPMI)
Meluxina	SLURM
Karolina	MERIC
Vega	Bull Energy Optimizer
Juwels Booster/ Jupiter (not yet installed)	LLview (NVIDIA `dcgm`)

Leonardo Booster and Leonardo DGCP utilize the Bull Energy Booster to monitor the power, energy, temperature, and performance of the entire cluster infrastructure. This monitoring is conducted out-of-band via IPMI and Simple Network Management Protocol (SNMP). These tools can work in conjunction with SLURM to adjust certain features, such as the way jobs are chosen according to the expected power consumption or how the CPU frequencies are dynamically capped according to the total consumption. Moreover, this dynamic tuning technique is complemented by a second tool called Bull Dynamic Power Optimizer, which analyzes power usage core by core in order to cap frequencies at the value that provides the best mix of energy savings and performance loss for running applications. In terms of GPU power consumption, NVIDIA `dcgm` allows users to reduce the GPU's clock speeds when they exceed a preset threshold [4, 7].

Energy consumption can be measured on Marenostrum 4 and Marenostrum 5 using EAR, see Section 3.3.1. Energy monitoring on Meluxina is performed using SLURM [6]. Energy consumption on the Karolina supercomputer can be measured using MERIC runtime system see Section 3.3.2 which was developed as a part of the READEX project. An example of energy consumption measurement for CFD solvers based on the Lattice Boltzmann method is explained in [2]. HPC Vega is a primary supercomputer system of the Slovenian national research infrastructure. It also supports Bull Energy Optimizer along with Bull SLURM and PMT to measure energy and temperature of the cluster [8]. As the installation of Jupiter, the first European exascale supercomputer, has not yet begun, information is subject to change, and the methods for measuring energy on this system remain uncertain. But, the Juwels Booster supercomputer hosted by Jülich Supercomputing Center supports LLview to monitor power and energy consumption as

explained in Section 3.3.3 and demonstrated on NekRS in Section 5.5.

References

[1] Morris Jette Alejandro Sanchez. *Workload Scheduling and Power Management, Presentation*. Details on https://slurm.schedmd.com/SLUG18/power_management.pdf. (Visited on 08/08/2024).

[2] Jayesh Badwaik. *ISC23 Project Poster: SCALABLE: Scalable Lattice-Boltzmann Leaps to Exascale*. 2023.

[3] Barcelona Supercomputing Center. *Energy Aware Runtime*. Details on https://github.com/sara-nl/ISC-2024-EAR-tutorial. (Visited on 08/08/2024).

[4] CINECA Supercomputing Center. *Leonardo Energy Monitoring*. Details on. URL: https://leonardo-supercomputer.cineca.eu/hpc-system/ (visited on 08/08/2024).

[5] Erlangen National High Performance Computing Center. *LIKWID*. Details on https://hpc.fau.de/research/tools/likwid/. (Visited on 08/08/2024).

[6] Meluxina Supercomputing Center. *Meluxina Energy Monitoring*. Details on. URL: https://docs.lxp.lu/first-steps/handling_jobs/#energy-monitoring (visited on 08/08/2024).

[7] Regale Supercomputing Center. *Bull Energy Optimizer*. Details on. URL: https://regale-project.eu/?page_id=421 (visited on 08/08/2024).

[8] Vega Supercomputing Center. *Vega Energy Monitoring*. Details on. URL: https://www.izum.si/en/vega-en/ (visited on 08/08/2024).

[9] Yanxiang Chen, Pablo de Oliveira Castro, Paolo Bientinesi, and Roman Iakymchuk. "Enabling mixed-precision with the help of tools: A Nekbone case study". In: *Appear at PPAM24*. May 2024. DOI: 10.48550/arXiv.2405.11065. arXiv: 2405.11065.

[10] Linux Kernel Community. *perf: Linux profiling with performance counters*. Details on https://perf.wiki.kernel.org/index.php/Main_Page. (Visited on 08/08/2024).

[11] CEEC Consortium. *Deliverable D2.3 – Intermediate performance and energy efficiency evaluation results*. Available via https://ceec-coe.eu/. June 2024.

[12] SCALABLE Consortium. *Deliverable D2.3 – Application performance, accuracy and energy efficiency*. Aug. 2022. URL: https://scalable-hpc.eu/publications/public-deliverables/ (visited on 08/08/2024).

[13] LUMI Documentation. *Perftools*. Details on https://docs.lumi-supercomputer.eu/development/profiling/perftools/. (Visited on 08/08/2024).

[14] J Dongarra, J Hittinger, J Bell, L Chacon, R Falgout, M Heroux, P Hovland, E Ng, C Webster, and S Wild. *Applied Mathematics Research for Exascale Computing*. U.S. Department of Energy. Feb. 2014. DOI: 10.2172/1149042.

[15] TU Dresden. *Vampir-Performance optimization*. Details on. URL: https://vampir.eu/ (visited on 08/08/2024).

[16] Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, and Tim Warburton. *NekRS, a GPU-Accelerated Spectral Element Navier-Stokes Solver*. 2021. arXiv: 2104.05829.

[17] GNU. *FreeIPMI*. Details on https://www.gnu.org/software/freeipmi/. (Visited on 08/08/2024).

[18] Daniel Hackenberg, Thomas Ilsche, Robert Schöne, Daniel Molka, Maik Schmidt, and Wolfgang E. Nagel. "Power measurement techniques on standard compute nodes: A quantitative comparison". In: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2013, pp. 194–204. DOI: 10.1109/ISPASS.2013.6557170.

[19] Daniel Hackenberg, Thomas Ilsche, Joseph Schuchart, Robert Schöne, Wolfgang E. Nagel, Marc Simon, and Yiannis Georgiou. "HDEEM: High Definition Energy Efficiency Monitoring". In: *2014 Energy Efficient Supercomputing Workshop*. 2014, pp. 1–10. DOI: 10.1109/E2SC.2014.13.

[20] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. "An Energy Efficiency Feature Survey of the Intel Haswell Processor". In: *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 2015, pp. 896–904. DOI: 10.1109/IPDPSW.2015.70.

[21] Alistair Hart, Harvey Richardson, Jens Doleschal, Thomas Ilsche, Mario Bielert, and Matthew Kappel. "User-level Power Monitoring and Application Performance on Cray XC30 Supercomputers". In: *Cray User Group* (2014).

[22] Intel. *Intel HPC Toolkit*. Details on https://www.intel.com/content/www/us/en/developer/tools/oneapi/hpc-toolkit-download.html. (Visited on 08/08/2024).

[23] Intel, HPE, NEC, and Dell. *IPMI Specification v2.0*. Details on https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/ipmi-intelligent-platform-mgt-interface-spec-2nd-gen-v2-0-spec-update.pdf. (Visited on 08/08/2024).

[24] Niclas Jansson, Martin Karp, Artur Podobas, Stefano Markidis, and Philipp Schlatter. "Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics". In: *Computers & Fluids* 275 (2024), p. 106243. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2024.106243.

[25] Luigi Brochard Julita Corbalan. *EAR: Energy management framework for supercomputers*. 2018. URL: https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/ear.pdf (visited on 08/08/2024).

[26] Martin Karp, Daniele Massaro, Niclas Jansson, Alistair Hart, Jacob Wahlgren, Philipp Schlatter, and Stefano Markidis. "Large-Scale direct numerical simulations of turbulence using GPUs and modern Fortran". In: *The International Journal of High Performance Computing Applications* 37.5 (2023), pp. 487–502. DOI: 10.1177/10943420231158616.

[27] Richard Kavanagh and Karim Djemame. "Rapid and accurate energy models through calibration with IPMI and RAPL". In: *Concurrency and Computation: Practice and Experience* 31.13 (2019), e5124. DOI: 10.1002/cpe.5124.

[28] Daniel Kempf, Marius Kurz, Marcel Blind, Patrick Kopper, Philipp Offenhäuser, Anna Schwarz, Spencer Starr, Jens Keim, and Andrea Beck. *GALÆXI: Solving complex compressible flows with high-order discontinuous Galerkin methods on accelerator-based systems*. 2024. arXiv: 2404.12703.

[29] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. "RAPL in Action: Experiences in Using RAPL for Power Measurements". In: *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3.2 (2018). ISSN: 2376-3639. DOI: 10.1145/3177754.

[30] Nico Krais, Andrea Beck, Thomas Bolemann, Hannes Frank, David Flad, Gregor Gassner, Florian Hindenlang, Malte Hoffmann, Thomas Kuhn, Matthias Sonntag, and Claus-Dieter Munz. "FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws". In: *Computers & Mathematics with Applications* 81 (2021), pp. 186–219. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2020.05.004.

[31] James H. Laros III, Kevin Pedretti, Suzanne M. Kelly, Wei Shu, Kurt Ferreira, John Vandyke, and Courtenay Vaughan. "Energy Delay Product". In: *Energy-Efficient High Performance Computing: Measurement and Tuning*. London: Springer London, 2013, pp. 51–55. DOI: 10.1007/978-1-4471-4492-2_8.

[32] James W. Lottes, Paul F. Fischer, and Stefan G. Kerkemeier. *NEK5000*. Details on http://nek5000.mcs.anl.gov. (Visited on 08/08/2024).

[33] Michael Malms et al. *ETP4HPC's SRA 5 - Strategic Research Agenda for High-Performance Computing in Europe - 2022*. Nov. 2022. DOI: 10.5281/zenodo.7347009.

[34] Steven J. Martin and Matthew Kappel. "Cray XC30 Power Monitoring and Management". In: *Cray User Group* (2014). URL: https://cug.org/proceedings/cug2014_proceedings/includes/files/pap130.pdf (visited on 08/08/2024).

[35] Heike McCraw, James Ralph, Anthony Danalis, and Jack Dongarra. "Power monitoring with PAPI for extreme scale architectures and dataflow-based programming models". In: *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. 2014, pp. 385–391. DOI: 10.1109/CLUSTER.2014.6968672.

[36] Senior Fellow at AMD Michael Clark. *The Zen Architecture*. Details on https://www.slideshare.net/slideshow/amd-ryzen-cpu-zen-cores-architecture/72835158. (Visited on 08/08/2024).

[37] Yannik Müller, Filipe Souza Mendes Guimarães, Carsten Karbach, and Wolfgang Frings. *LLview*. Details on https://apps.fz-juelich.de/jsc/llview/docu/. 2023. DOI: 10.5281/zenodo.10221407.

[38] Nek5000 developers. *Nekbone*. Details on https://github.com/Nek5000/Nekbone. (Visited on 08/08/2024).

[39] Nvidia. *NVML API Energy Device Query*. URL: https://docs.nvidia.com/deploy/archive/R535/nvml-api/group__nvmlDeviceQueries.html#group__nvmlDeviceQueries_1g732ab899b5bd18ac4bfb93c02de4900a (visited on 08/08/2024).

[40] University of Oregorn. *Tuning and Analysis utilities*. Details on https://www.cs.uoregon.edu/research/tau/home.php. (Visited on 08/08/2024).

[41] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. "The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink". In: *Computer* 55.7 (2022), pp. 18–28. DOI: 10.1109/MC.2022.3148714. arXiv: 2204.05149.

[42] Daniel A. Reed and Daniel Jack. "Exascale Computing and Big Data: The Next Frontier". In: *Communications of The ACM* (2015), pp. 56–68. DOI: 10.1145/2699414. URL: https://cacm.acm.org/research/exascale-computing-and-big-data/ (visited on 08/08/2024).

[43] Robert Schone, Thomas Ilsche, Mario Bielert, Markus Velten, Markus Schmidl, and Daniel Hackenberg. "Energy Efficiency Aspects of the AMD Zen 2 Architecture". In: *2021 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, Sept. 2021. DOI: 10.1109/cluster48925.2021.00087.

[44] TOP500.org. *Green500 List*. Details on https://www.top500.org/lists/green500/. (Visited on 08/08/2024).

[45] Ondrej Vysocky, Martin Beseda, Lubomír Říha, Jan Zapletal, Michael Lysaght, and Venkatesh Kannan. "MERIC and RADAR Generator: Tools for Energy Evaluation and Runtime Tuning of HPC Applications". In: *High Performance Computing in Science and Engineering*. Cham: Springer International Publishing, 2018, pp. 144–159. DOI: 10.1007/978-3-319-97136-0_11.

[46] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. "Measuring Energy and Power with PAPI". In: *2012 41st International Conference on Parallel Processing Workshops*. 2012, pp. 262–268. DOI: 10.1109/ICPPW.2012.39.

[47] Markus Wittmann, Viktor Haag, Thomas Zeiser, Harald Köstler, and Gerhard Wellein. "Lattice Boltzmann Benchmark Kernels as a Testbed for Performance Analysis". In: *CoRR* (2017). DOI: 10.1016/j.compfluid.2018.03.030.