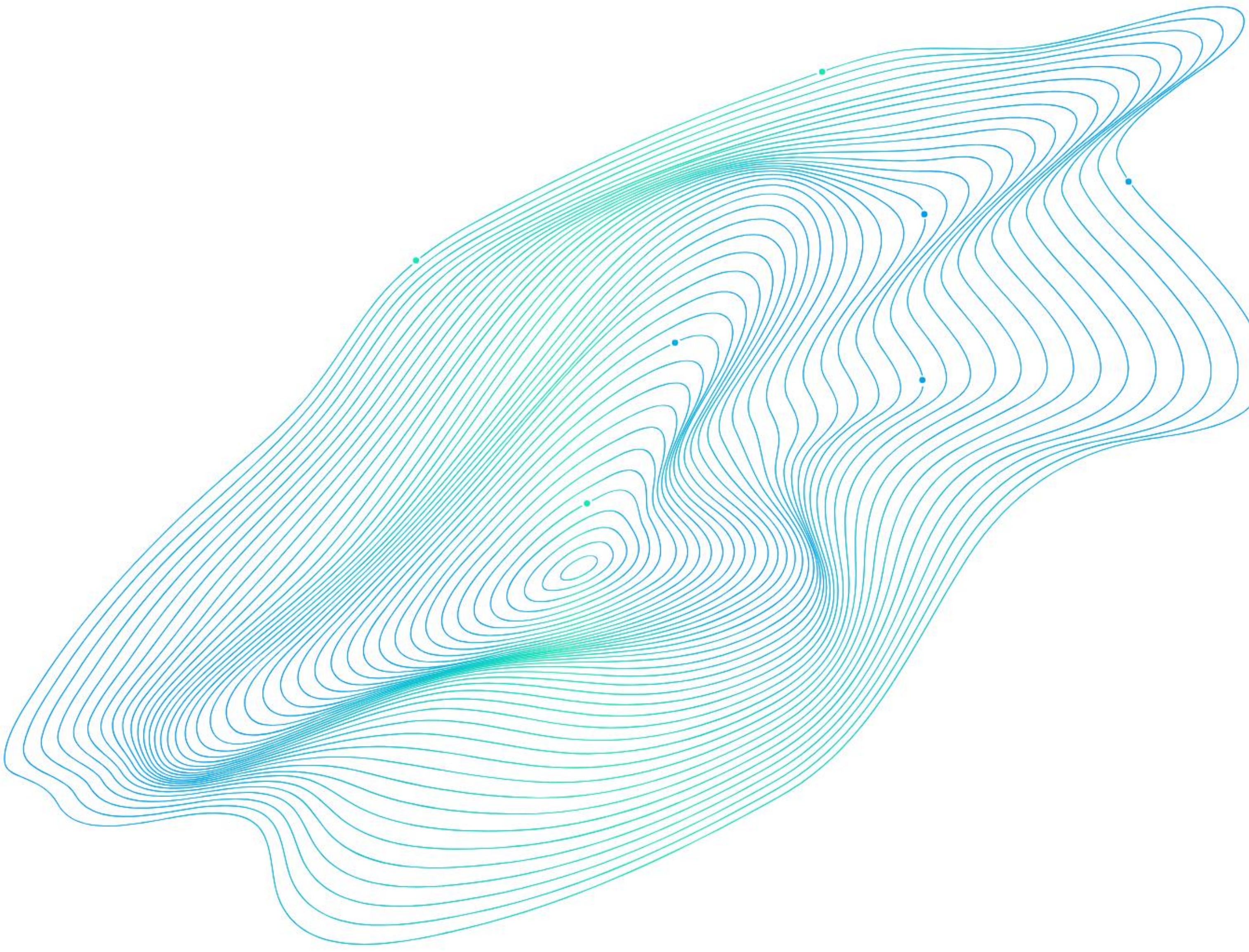
EPICURE

Unlocking European-level HPC Support

HIGH-LEVEL SPECIALISED APPLICATION SUPPORT SERVICE IN HIGH-PERFORMANCE COMPUTING (HPC)



Co-funded by the European Union

EuroHPC Joint Undertaking

This project has received funding from the European High Performance Computing Joint Undertaking under grant agreement No.101139786.

Best Practice Guide on Power Consumption Measurements in EuroHPC Systems

Project Title	High-level specialised application support service in High-Performance Computing (HPC)
Project Acronym	EPICURE
Project Number	101139786
Type of Action	DIGITAL JU Simple Grants
Topic	DIGITAL-EUROHPC-JU-2022-APPSUPPORT-01-01
Starting Date of Project	2024-02-01
Ending Date of Project	2028-01-31
Duration of the Project	48 months
Website	epicure-hpc.eu
Document version	2.0
Document publication date	2026-01-31

Executive Summary

This document provides practical guidance for users of EPICURE on accessing and interpreting power consumption data from various EuroHPC supercomputing systems. It outlines the different approaches each system uses to collect, aggregate, and expose energy and power measurements, and offers concrete examples and job script templates to help users monitor and analyse the energy footprint of their applications.

Changes in revision v2.0

The following changes have been added in version v2.0 relative to v1.0:

- **Benchmarks**
    - **Figures**
      Guidance with each figure.
    - **Results**
      Guidance with interpreting the data: significance of the results, explanation for the outliers in the data
- **Conclusions**
    - **Measurements**
      Comments on the results and the limitations of measuring the power consumption of different HPC sites.

Table of Contents

1. Introduction

This Best Practice Guide on Power Consumption Measurements in EuroHPC Systems provides an overview of how users can access and interpret power consumption data across all currently active EuroHPC supercomputers. It describes the tools and methods available to monitor and analyse energy usage during computation on these systems.

To support practical application, the guide also includes example job scripts and benchmark outputs collected from multiple EuroHPC machines. These resources are shared on EPICURE's shortbench GitLab repository (EPICURE's shortbench GitLab), enabling users to integrate power monitoring into their workflows more effectively.

2. Overview of the benchmarks

The benchmarks selected for this study are well-known within the HPC community and are typically available on all EuroHPC clusters. Each of these applications offers options to run on both CPUs and GPUs, allowing us to compare their performance and power consumption across different hardware configurations.

These applications are also widely used across HPC facilities, making the results particularly relevant for users deciding which machine best suits their workloads, or those seeking practical examples of job scripts and input configurations.

2.1. CPU

GROMACS

About the code

GROMACS is an open-source, high-performance molecular dynamics (MD) package widely used in the life science community It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids, but can be used also for non-biological system like in materials science.

About the benchmark

lignocellulose-rf is part of the PRACE Unified European Applications Benchmark Suite (UEABS). It simulates a complex lignocellulosic biomass system using reaction-field for electrostatics, making it relevant for large-scale simulations and scalability benchmarking.

CP2K

About the code

CP2K is an open-source quantum chemistry and solid-state physics software package. It is known for its efficiency and scalability on large parallel systems. CP2K provides a general framework for different modelling methods such as DFT which is the one used in our benchmark input.

About the benchmark

H2O-DFT-LS is one of CP2K's default benchmarks included in its installation package. It performs large-scale DFT calculations on water molecules and is commonly used to evaluate the scalability and parallel performance of DFT-based simulations on different computing architectures.

NAMD

About the code

NAMD is a computer software optimized for high-performance molecular dynamics simulations. It is noted for its parallel efficiency and is often used to simulate large systems (millions of atoms).

About the benchmarks

20stmv2fs.namd (memory-optimized) and 20stmv2fs-nonopt.namd (non–memory-optimized) are official benchmarks included with NAMD source code. Both are designed to test performance on large biomolecular systems like the Satellite Tobacco Mosaic Virus (STMV).

2.2. GPU

The same benchmarks were also executed on GPU-accelerated hardware, using the same input configurations as on the CPU. This approach enables a direct comparison of performance and scalability between CPU-only and GPU-accelerated runs.

By comparing CPU and GPU results on identical benchmarks, we can better evaluate how effectively each code takes advantage of GPU acceleration, as well as quantify improvements in both performance and power efficiency when running on GPU-enabled EuroHPC infrastructures. On systems like MareNostrum 5, Leonardo, and LUMI, where both CPU and GPU partitions are part of the same machine and share uniform power measurement tools, the comparison becomes especially valuable and reliable.

Co-funded by the European Union

3. EuroHPC systems

As part of the EPICURE project, we have access to all currently active EuroHPC supercomputing systems across participating sites. This unique collaboration enables us to run benchmarks and collect power consumption data directly on each of these systems, ensuring that the information and examples provided in this guide reflect real, up-to-date usage across the entire EuroHPC landscape.

In this section, we present an overview of each EuroHPC system included in our study. For each machine, we describe its architecture, available accelerators (CPU/GPU), and the tools or interfaces it provides for monitoring power and energy usage. This context will help users understand the capabilities and differences between systems, and how to apply the practical examples shared in this guide to their own jobs.

3.1. LUMI

LUMI is a pre-exascale EuroHPC supercomputer, supplied by HPE and in production since 2022. It is hosted by CSC in its Kajaani data centre in Finland.

Specifications

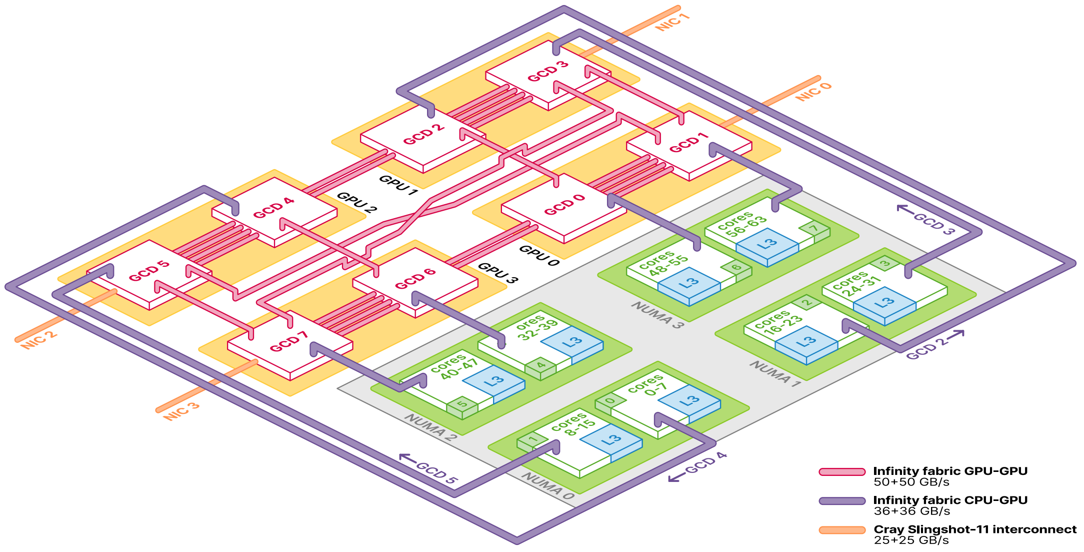


Figure 3-1: Overview of a LUMI-G compute node

GPU partition (LUMI-G)

- 2928 GPU nodes, detail in Figure 3-1
    - 4 AMD MI250 GPUs (128 GiB GPU memory)
    - 1 AMD Trento host-CPU (512 GiB host memory)

CPU partition (LUMI-C)

- 1888 CPU nodes, 2 x 64-core 2.45 GHz AMD Milan, 256 GiB RAM
- 128 CPU nodes, 2 x 64-core 2.45 GHz AMD Milan, 512 GiB RAM

Co-funded by the European Union

EuroHPC Joint Undertaking

- 32 CPU nodes, 2 x 64-core 2.45 GHz AMD Milan, 1 TiB RAM

Interactive data-analytics partition (LUMI-D)

- 8 big-memory nodes, 2 x 64-core 2.25 GHz AMD Rome, 4 TiB RAM
- 8 visualization nodes, 8 NVIDIA A40 GPUs (48 GiB GPU memory) and 2 x 64-core 2.25 GHz AMD Milan (2 TiB host memory)

Measurements

Energy is measured on node level and job consumed energy is reported through Slurm energy accounting.

Data from `pm_counters` on node level is available to the administrators.

Additional info

Slurm info

23.02.7; `acct_gather_energy` / `pm_counters`

Extra tools

Benchmarking environment

Manual executions

Performance analysis

CrayPat, rocprof. Other (Score-P, Scalasca) may be installed using EasyBuild recipes found in the LUMI Software Library but are not officially supported.

Links

- Main page
  https://www.lumi-supercomputer.eu
- Documentation
  https://docs.lumi-supercomputer.eu
- Energy Consumption
  https://docs.lumi-supercomputer.eu/runjobs/scheduled-jobs/jobenergy/
- Software Installation
  https://lumi-supercomputer.github.io/LUMI-EasyBuild-docs/

3.2. Leonardo

Leonardo is a next-generation pre-exascale Tier-0 supercomputer, part of the EuroHPC Joint Undertaking, in production since August 2023. It is hosted by CINECA at the Bologna Technopole in Italy and it is developed and supplied by EVIDEN ATOS.

Specifications

Leonardo is structured into two main compute partitions, both connected via DragonFly+ (NVIDIA Mellanox Infiniband HDR) 200 Gbps and managed using Slurm workload manager.

Booster Partition

- 3456 heterogenous nodes with 32 cores/node and 4 GPUs/ node
- Based on single socket Intel Ice Lake CPU (Intel Xeon Platinum 8358, 2.60 GHz, TDP 250 W)
- Equipped with NVIDIA Ampere GPUs, 64 GB HBM2e NVLink 3.0 (200 GB/s)
- 2 x dual port HDR100 per node

Data Centric General Purpose (DCGP) Partition

- 1536 nodes with 112 cores/node
- Based on dual socket 56 cores Intel Sapphire Rapids CPU (2 x Intel Xeon Platinum 8480p, 2.00 GHz, TDP 350 W)
- Single port HDR100 per node

Measurements

- Energy can be measured at the node and job level by installing **COUNTDOWN**, for the Booster partition only.
- GPU energy on Booster can be measured by users via nvidia-smi and NVML.
- CPU energy can be retrieved by reading RAPL sampling data on Booster and DCGP.
- The **CINEMON** tool, developed by CINECA staff and based on **RAPL** and **NVML** power measurements, can be installed on Leonardo cluster to measure the overall CPU, RAM, GPU, NODE and JOB energy consumed. Time series are currently available, environment variables can be used to adapt the sampling period of RAPL and NVML. More information regarding its deployment and measurement configurations can be found on the project `README.md`.

Additional info

Slurm info

22.05.10

Extra tools

COUNTDOWN, Intel RAPL and NVIDIA NVML, NVIDIA-SMI, CINEMON

Benchmarking environment

JUBE

Performance analysis

SCORE-P, NSYS, NCU

Links

- Main Page
  https://leonardo-supercomputer.cineca.eu
- Energy usage
  https://leonardo-supercomputer.cineca.eu/hpc-system/#jump-efficiency
- Documentation
  https://docs.hpc.cineca.it/index.html

Co-funded by the European Union

EuroHPC Joint Undertaking

3.3. MareNostrum 5

MareNostrum 5 is a pre-exascale EuroHPC supercomputer supplied by Bull SAS that combines Lenovo ThinkSystem SD650 V3 and Eviden BullSequana XH3000 architectures, providing two partitions with different technical characteristics.

Specifications

MareNostrum 5 GPP (General Purpose Partition)

The MareNostrum 5 GPP, a general-purpose system, houses 6,408 nodes based on Intel Sapphire Rapids (4th Generation Intel Xeon Scalable Processors), along with an additional 72 nodes featuring Intel Sapphire Rapids HBM (High Bandwidth Memory). This configuration results in a total count of 726,880 processor cores and 1.75PB of main memory. The different configuration of nodes within this partition is present below:

- 6192 nodes, 2 x Intel Xeon Platinum 8480+ 56 cores, 2 GHz, 256 GiB
- 216 high memory nodes, 2 x Intel Xeon Platinum 8480+ 56 cores, 2 GHz, 1024 GiB
- 72 HBM nodes, 2 x Intel Xeon Platinum 8480+ 56 cores, 2 GHz, 128 GiB
- 10 Data nodes, 2 x Intel Xeon Platinum 8480+ 56 cores, 2 GHz, 2048 GiB

MareNostrum 5 ACC (Accelerated Partition)

The MareNostrum 5 ACC accelerated system comprises 1,120 nodes based on Intel Xeon Sapphire Rapids processors and NVIDIA Hopper GPUs, offering a total (CPUs + GPUs) of 680,960 compute units. The nodes are configured with the following components:

- 1120 nodes, 2x Intel Xeon Platinum 8460Y+ 40cores, 2.3 GHz, 512 GB, 4x NVIDIA Hopper H100 64GB HBM2

Measurements

- Energy usage is reported through Slurm energy accounting and the Energy Aware Runtime (EAR) tool.
- On the **GPP partition**, energy consumption is monitored using both EAR and Slurm energy accounting.
- On the **ACC partition**, energy consumption is monitored using EAR only.

Additional info

Slurm info

23.02.7

Extra *tools*

EAR

Co-funded by the European Union

EuroHPC Joint Undertaking

Benchmarking environment

JUBE

Performance analysis

TALP, Extrae and Paraver

Links

- Main Page
  https://www.bsc.es/supportkc/docs/MareNostrum5/intro/
- Job Submission
  https://www.bsc.es/supportkc/docs/MareNostrum5/slurm

3.4. MeluXina

The system is in production since November 2021. The supercomputer is based on Atos Sequana XH2000, with 813 compute nodes, which are interconnected with InfiniBand (Dragonfly+ topology).

Specifications

CPU partition

- 573 CPU nodes, 2x AMD Rome 7H12 (64c, 2.6 GHz, 280W), 512GiB RAM

GPU partition:

- 200 GPU nodes 4x Nvidia A100 40 GiB HBM2, 2x AMD Rome 7452 (32c, 2.3 GHz, 155W), 512 GiB RAM

Large memory partition

- 20 fat nodes, 2x AMD Rome 7H12 (64c, 2.6 GHz, 280W), 4 TiB RAM

FPGA partition

- 20 FPGA nodes, 2x BittWare 520N-MX 16 GiB HBM2 (Intel Stratix 10MX chip), 2x AMD Rome 7452 (32c, 2.3 GHz, 155W), 512 GiB RAM

Measurements

- Energy is measured on node level and job consumed energy is reported through Slurm energy accounting.
- Data from IPMI sensors on node level is available to admins.
- For FPGA cards, we use the bittware `minitor` executable which is only available to admins.

Additional info

Slurm info

23.11.9; `acct_gather_energy` / `ipmi`

Extra tools

Benchmarking environment

Reframe

Performance analysis

Score-P, perf, Intel VTune, NVIDIA Nsight Systems

Links

- Main page
  https://docs.lxp.lu
- Energy usage
  https://docs.lxp.lu/first-steps/handling_jobs/#energy-monitoring

3.5. Karolina

Karolina is HPE Apollo (Apollo 200 and Apollo 6500) system with fully non-blocking fat-tree InfiniBand interconnect. The system is in operation from Q2 of 2021. The Karolina cluster consists of several partitions which together gives over 15.7 PFLOP/s theoretical peak performance.

Specifications

CPU partition

- 720 nodes, 2 x AMD Zen 2 EPYC 7H12 (280W TDP), 256 GiB DDR4

GPU partition

- 72 nodes, 8 x NVIDIA A100 (40 GiB HBM2) (400 W TDP), 2 x AMD Zen 3 EPYC 7763 (280 W TDP), 1024 GiB DDR4

Measurements

In the Karolina system, **MERIC energy efficient HPC software suite** is deployed. Using its Job budgeting service every user may read energy consumption of jobs executed under the project the user participates in. Administrators can access all jobs. It is also possible to extract energy consumption of a project, a cluster, a user, or specific period. In login nodes, a command line utility `get_energy` is available for users.

The MERIC Job budgeting service on Karolina provides job energy consumption at several levels:

- **CPU energy consumption** – In band (performance counters).
- **GPU energy consumption** – In band (performance counters), if GPUs available.
- **Node energy consumption** – Combination of CPU and GPU energy consumption (high frequency power sampling, typically 1kHz) and Out-of-Band power monitoring of the node (low frequency, typically 0.017 up to 1 Hz).
- **Overall energy consumption** – Node energy consumption multiplied by system Power Usage Effectiveness (PUE) at the moment of the job execution.

- **CO2e** – Overall energy consumption multiplied by carbon intensity (gCO2eq/kWh) at the moment of the job execution. The source of the carbon intensity can be site-specific, or universal solution reading the data from https://app.electricitymaps.com/, which provides the carbon intensity per region (in Europe typically per country).

Besides the command line utility which prints the CO2e and energy consumption at all the levels, the Job budgeting service also provides web interface which in addition presents power consumption timeline (power consumption of each CPU, each GPU, each node) during the job. The timeline granularity is 0.017 Hz (one sample per minute) presenting average power consumption during the past minute.

For users, MERIC runtime system (used as a user-tool instead of runtime system) is available as a software module to measure energy consumption per application execution using command line utility, or energy consumption and energy-efficiency metrics per application region if linked with the library and application's regions of interest instrumented. See section 4.3 for more details.

Administrators have additional power monitoring dashboards presenting power and temperature level per node and chassis in a rack, per rack, cluster and other infrastructure levels according to site-specific availability.

Energy efficiency optimization

From February 2023 the Karolina cluster is operated in the static energy efficient mode, which reduces CPU core frequency limit of CPU partition from 3.3 GHz to 2.1 GHz, and GPU SMs frequency limit of GPU partition from 1.41 GHz to 1.29 GHz.

Additionally, a group of users (extended on request) may access CPU and GPU power management knobs to optimize energy efficiency of the executed workload. MERIC runtime system is available as a software module to expose these knobs and provide static and automatic dynamic tuning to improve executed application energy efficiency. See section 4.3 for more details.

Additional info

Slurm info

23.11.10

Extra tools

MERIC

Benchmarking environment

Gitlab runners with Jacamar CI driver available in IT4Innovations' GitLab (available to all system users) which allows to execute continuous integration and continuous benchmarking jobs in compute nodes.

Performance analysis

POP CoE tools (Score-P, Scalasca, Extrae, MAQAO, DLB, MERIC, MUST, CARM), NVIDIA Nsight Systems, Linaro's software tools, Intel Advisor, Intel VTune, AMD μProf

Links

- Main page
  https://docs.it4i.cz/karolina/introduction/
- Energy usage
  https://docs.it4i.cz/general/energy/?h=energy
- Meric Suite
  https://code.it4i.cz/energy-efficiency/meric-suite
- POP tools
  https://pop-coe.eu/

3.6. Discoverer

The system is in production since September 2021. The supercomputer is based on Atos Sequana XH2000, with 1128 compute nodes, which are interconnected with InfiniBand (Dragonfly+ topology).

Specifications

CPU partition

- 1110 CPU nodes, 2x AMD Rome 7H12 (64c, 2.6 GHz, 280 W), 256 GiB RAM

  Large memory partition:
  - 18 fat nodes, 2x AMD Rome 7H12 (64c, 2.6 GHz, 280 W), 1 TiB RAM

Discoverer+ GPU partition

- 32 (4 × 8) NVIDIA H200 GPU accelerators, 448 (112 × 4) hardware CPU cores, 7.84 (1.96 × 4) TiB RAM

Measurements

Energy is measured on node level and job consumed energy is reported through custom web-based interface.

Data from IPMI sensors on node level is available to administrators.

Additional info

Slurm info

20.02.6-Bull.1.1

Extra tools

Benchmarking environment

Manual executions

Performance analysis

Intel Vtune, NVIDIA Nsight, perf, AMD μProf, Score-P, TAU, HPCToolkit.

Links

- Main page
  https://docs.discoverer.bg/index.html

3.7. Vega

The system is in production since April 2021. The supercomputer is based on Atos Sequana XH2000, with 1020 compute nodes, which are interconnected with InfiniBand (Dragonfly+ topology).

Specifications

CPU partition

- 768 CPU nodes, 2 x AMD Rome 7H12 (64c, 2.6 GHz, 280 W), 256 GiB RAM
- 192 CPU nodes, 2 x AMD Rome 7H12 (64c, 2.6 GHz, 280 W), 1 TiB RAM

GPU partition

- 60 GPU nodes 4 x Nvidia A100, 2 x AMD Rome 7H12, 512 GiB RAM

Measurements

Energy is measured on node level and job consumed energy is reported through Slurm energy accounting (IPMI).

Data from IPMI sensors on node level is available to admins. Kernel module for RAPL is loaded but not readable for users.

Additional info

Slurm info

24.11.4; `acct_gather_energy/ipmi`

Extra tools

NVML

Benchmarking environment

Manual Execution.

Performance analysis

LIKWID, TotalView, Score-P, perf, Intel VTune, PAPI, nways, ...

Links

- Main page
  https://www.izum.si/en/hpc-en/
- Energy usage
  https://doc.vega.izum.si/energy-usage/

Co-funded by the European Union

EuroHPC Joint Undertaking

3.8. Deucalion

Deucalion is a peta-scale EuroHPC supercomputer, supplied by Fujitsu (currently Fsas Technologies) and in production since June 2024. It is hosted by FCT at Universidade do Minho in Guimarães, Portugal.

Deucalion has 3 partitions: one partition based on the Fujitsu ARM A64FX processors, one based on AMD Epyc 7742 processors (2 sockets per node) and an accelerated partition based on AMD Epyc 7742 accelerated with Nvidia A100 GPUs (4 per node, including both A100 with 40 and 80 GiB of VRAM).

The ARM partition is interconnected with Infiniband HDR Fat-Tree with 1:1.6 blocking factor and the AMD and GPUs partitions are interconnected with Infiniband HDR Fat-Tree with 1:1 non-blocking.

Specifications

CPU (A64FX) partition

- 1632 ARM FX700 nodes, Fujitsu's A64FX (48c, 2.0 GHz), 32 GiB RAM

CPU (x86) partition

- 500 nodes, 2x AMD Epyc 7742 (64c, 2.25 GHz), 256 GiB RAM

GPU partition

- 17 nodes, 4 x Nvidia A100 GPUs (40 GiB GPU memory), 2 x AMD Epyc 7742 (64c, 2.25 GHz), 512 GiB RAM
- 16 nodes, 4 x Nvidia A100 GPUs (80 GiB GPU memory), 2 x AMD Epyc 7742 (64c, 2.25 GHz), 512 GiB RAM

Measurements

Deucalion uses the MERIC energy-efficient HPC software suite, the same as Karolina (See section 3.5).

Additional info

Slurm info

23.11.8

Extra tools

MERIC

Benchmarking environment

Manual Execution

Performance analysis

POP CoE tools (Score-P, Scalasca, MAQAO, DLB, MERIC), Intel Vtune

Links

- Main page
  https://docs.deucalion.macc.fccn.pt

3.9. JUPITER

JUPITER, the “Joint Undertaking Pioneer for Innovative and Transformative Exascale Research", will be the first exascale supercomputer in Europe. The system is provided by a ParTec-Eviden supercomputer consortium and was procured by EuroHPC JU in cooperation with the Jülich Supercomputing Centre (JSC). It is installed in the Forschungszentrum Jülich campus in Germany.

Specifications

JUPITER Booster consists of ~6000 standard compute nodes

- 4 × NVIDIA GH200 Grace-Hopper Superchip (see Figure 3-2)
  - CPU: NVIDIA Grace (Arm Neoverse-V2), 72 cores at 3.1 GHz base frequency; 120 GiB LPDDR5X memory at 512 GiB/s (8532 MHz)
  - GPU: NVIDIA Hopper H100, 132 multiprocessors, 96 GiB HBM3 memory at 4 TiB/s
  - NVIDIA NVLink-C2C CPU-to-GPU link at 900 GiB/s
  - TDP: 680 W (for full GH200 superchip)
- NVLink 4 GPU-to-GPU link, 300 GiB/s between pairs of GPUs (150 GiB/s per direction)
- Network: 4 × InfiniBand NDR200 (Connect-X7)

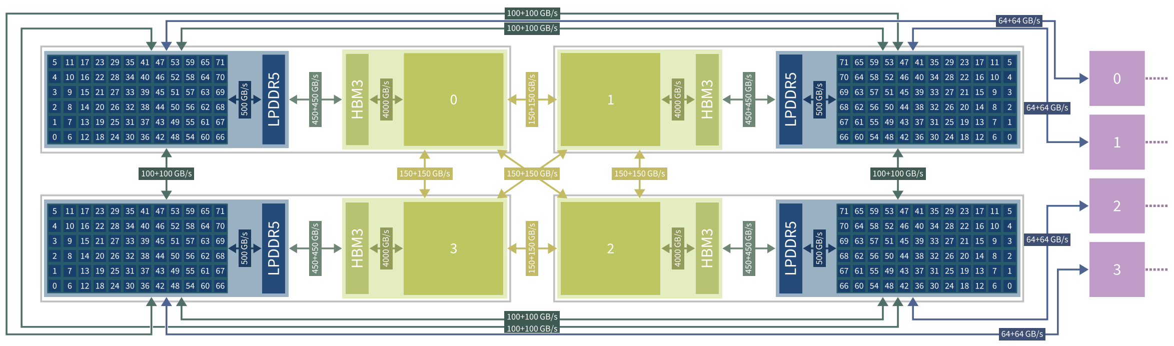


Figure 3-2: Node diagram of the 4× NVIDIA GH200 node design of JUPITER Booster.

Measurements

LLview (see section 4.6) can report power metrics (in Watts) at several levels, i.e. node power, CPU/GPU power, superchip power.

Additional info

Slurm info

Extra tools

LLview

Benchmarking environment

JUBE

Performance analysis

Score-P, Scalasca, CUBE, Vampir

Links

- Main page
  https://jupiter.fz-juelich.de/
- LLview
  https://llview.fz-juelich.de/

4. Tools

4.1. Slurm

SLURM is an open-source, fault-tolerant, and highly scalable workload manager designed for both large and small Linux clusters.

For power management, SLURM offers plugins that collect energy consumption data on a per-job basis. These plugins can use various hardware interfaces, such as IPMI, RAPL counters, or external scripts, depending on what is available on the system. The collected data is stored alongside each job and can be retrieved later using the `sacct` command, as described below.

More details about SLURM's power measurement options can be found in the Slurm documentation on AcctGatherEnergyType.

In the benchmarks presented in this document, when power data was obtained through SLURM, we used the commands given in Figure 4-1 to report the job type and the corresponding energy consumption.

```
$ sacct -j jobid.0 \
> -o nnodes,ntasks,ncpus,consumedenergy,consumedenergyraw, elapsed,elapsedraw
```

Figure 4-1:The command to report the job type and the energy consumption.

4.2. EAR

EAR software is a management framework optimizing the energy and efficiency of a cluster of interconnected nodes. To improve the energy of the cluster, EAR provides energy control, accounting, monitoring and optimization of both the applications running on the cluster and of the overall global cluster.

At EAR's core is a monitoring tool which gathers data on the nodes and on the applications running on the cluster. Therefore, on top of optimizing the energy consumed by the applications running on the cluster and the overall global cluster, EAR reports system and application information.

EAR components are the EAR library (EARL), EAR DB manager (EARDBD), EAR Daemon (EARD), EAR Slurm plugin (EARplug) and EAR Global Manager (EARGM). EAR offers a highly configurable and extensible infrastructure for energy management. Last version of EAR includes a plugin mechanism to dynamically load power policies, power and time models, energy readings and application traces generation. To offer a simple install and test approach, EAR includes default powerful plugins for all these features. Slurm is the batch scheduler full compatible with EAR thanks to EAR's Slurm SPANK plug-in. With EAR's Slurm plug-in, running an application with EAR is as easy as submitting a job with either `srun`, `sbatch` or `mpirun`. The EAR Library (EARL) is automatically loaded with some applications when EAR is enabled by default.

EAR Features

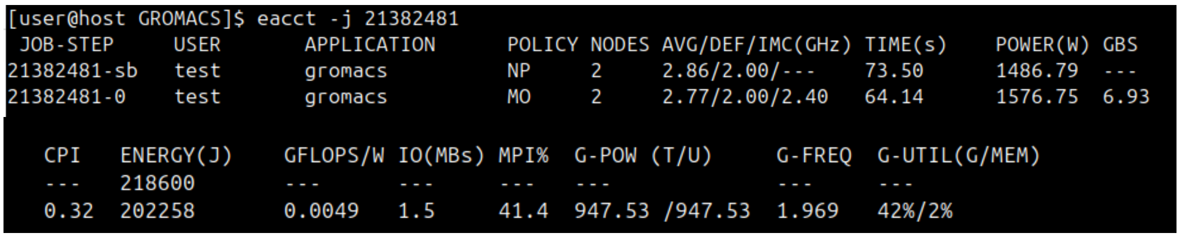
The following list highlights the main functionalities and features provided by EAR. While the accompanying examples are demonstrated on the MareNostrum 5 supercomputer, these capabilities are designed to be available in any standard installation of EAR.

EAR job Accounting

The `eacct` command shows accounting information stored in the EAR DB for jobs (and steps) IDs. The command uses EAR's configuration file to determine if the user running it is privileged or not, as *non-privileged users can only access their information*. It provides the following options.

Usage examples

The basic usage of `eacct` retrieves the last 20 applications (by default) of the user executing it. If a user is *privileged*, they may see all users' applications. The default behaviour shows data from each job-step, aggregating the values from each node in said job-step. If using Slurm as a job manager, a `sb` (`sbatch`) job-step is created with the data from the entire execution. A specific job may be specified with -j option.

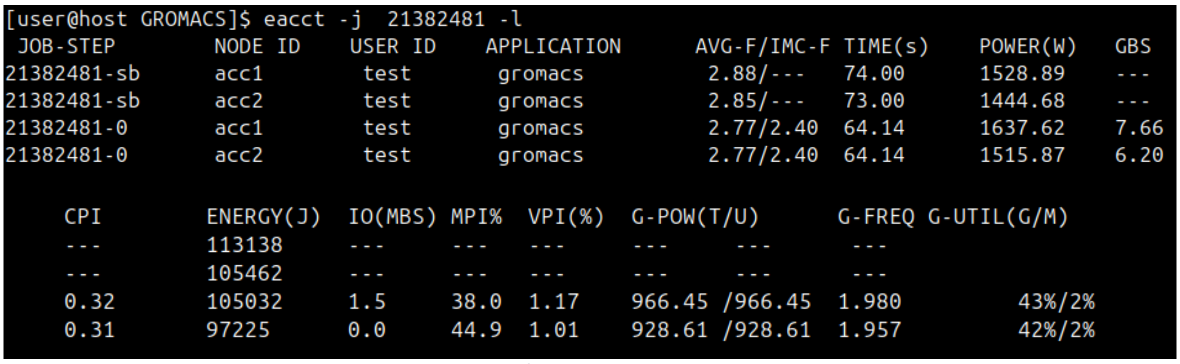


```
[user@host GROMACS]$ eacct -j 21382481
 JOB-STEP      USER       APPLICATION      POLICY NODES AVG/DEF/IMC(GHz) TIME(s)    POWER(W) GBS
21382481-sb   test       gromacs          NP     2      2.86/2.00/---    73.50      1486.79  ---
21382481-0    test       gromacs          MO     2      2.77/2.00/2.40   64.14      1576.75  6.93

   CPI   ENERGY(J)    GFLOPS/W IO(MBs) MPI%  G-POW (T/U)      G-FREQ  G-UTIL(G/MEM)
   ---   218600       ---      ---     ---   ---              ---     ---
   0.32  202258       0.0049   1.5     41.4  947.53 /947.53   1.969   42%/2%
```

Figure 4-2: Output obtained using the `eacct` command for a specific job.

For node-specific information, the `-l` (i.e., long) option provides detailed accounting of each individual node. If EARL was loaded during an application execution, runtime data (i.e., EAR loops) may be retrieved by using -r flag. An example of both their usage is shown below.



```
[user@host GROMACS]$ eacct -j  21382481 -l
 JOB-STEP          NODE ID    USER ID    APPLICATION      AVG-F/IMC-F TIME(s)    POWER(W)   GBS
21382481-sb       acc1        test       gromacs           2.88/---    74.00      1528.89    ---
21382481-sb       acc2        test       gromacs           2.85/---    73.00      1444.68    ---
21382481-0        acc1        test       gromacs           2.77/2.40   64.14      1637.62    7.66
21382481-0        acc2        test       gromacs           2.77/2.40   64.14      1515.87    6.20

    CPI         ENERGY(J)  IO(MBS) MPI%  VPI(%)  G-POW(T/U)      G-FREQ G-UTIL(G/M)
    ---         113138     ---     ---   ---     ---     ---      ---
    ---         105462     ---     ---   ---     ---     ---      ---
    0.32        105032     1.5     38.0  1.17    966.45 /966.45  1.980         43%/2%
    0.31        97225      0.0     44.9  1.01    928.61 /928.61  1.957         42%/2%
```

Figure 4-3: `eacct` showing detailed accounting of each node.

To easily transfer the output from `eacct`, you can use the `-c` option to save the requested data in CSV format. This can be done as follows:

```
[user@host EAR]$ eacct -j 21382481-c test.csv
```

Figure 4-4: Saving the output of `eacct` to test.csv.

If successful, you'll see a message like:

```
Successfully written applications to csv. Only applications with EARL will
have its information properly written.
```

Figure 4-5: The success message from the EAR application.

Example: Using EAR with Slurm+srun on MareNostrum5

When submitting jobs with `sbatch`, EAR options can be specified using the ear module, available in both partitions. For example:

```
#SBATCH --ear=on              # Enable Energy-Aware Runtime (EAR) monitoring
#SBATCH --ear-verbose=1       # Enable verbose EAR output

module load ear               # load the ear module
mkdir -p ear_metrics          # create directory to store EAR results

srun --ear-user-db=ear_metrics/app_metrics gmx_mpi mdrun \
   -s lignocellulose-rf.tpr -pin on -noconfout -nsteps 20000 -nstlist 200
```

Figure 4-6: Example job script using EAR on MareNostrum 5

EAR policies

EAR offers three energy policies plugins: `min_energy`, `min_time` and `monitoring`. The last one is not a power policy, is used just for application monitoring where CPU frequency is not modified (neither memory nor GPU frequency). The energy policy is selected by setting the `--ear-policy=policy` option when submitting a Slurm job. A policy parameter, which is a particular value or threshold depending on the policy, can be set using the flag `--ear-policy-th=value`.

`min_energy`

The goal of this policy is to minimise the energy consumed with a limit to the performance degradation. This limit is set in the Slurm `--ear-policy-th` option or the configuration file.

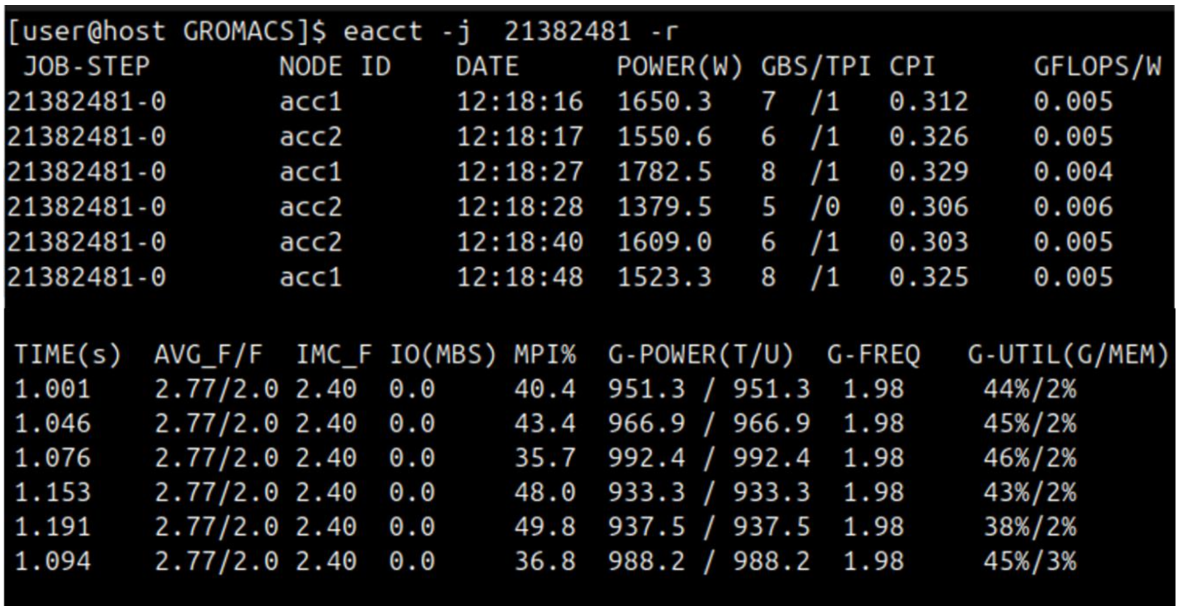
```
srun --ear-policy=min_energy \
    --ear-user-db=more_test_min_energy/app_metrics gmx_mpi mdrun \
    -s lignocellulose-rf.tpr -pin on -noconfout -nsteps 20000 -nstlist 200
```

Figure 4-7: Selecting `min_energy` in a Slurm job.

`min_time`

The goal of this policy is to improve the execution time while guaranteeing a minimum ratio between performance benefit and frequency increment that justifies the increased energy consumption from this frequency increment.

For instance, if `--ear-policy-th=0.70`, EAR will prevent scaling to upper frequencies if the ratio between performance gain and frequency gain do not improve at least 70% $(\text{PerfGain} \geq \text{FreqGain} \cdot \text{Threshold})$.



```
[user@host GROMACS]$ eacct -j  21382481 -r
 JOB-STEP          NODE ID    DATE       POWER(W) GBS/TPI CPI      GFLOPS/W
21382481-0         acc1       12:18:16  1650.3    7  /1   0.312    0.005
21382481-0         acc2       12:18:17  1550.6    6  /1   0.326    0.005
21382481-0         acc1       12:18:27  1782.5    8  /1   0.329    0.004
21382481-0         acc2       12:18:28  1379.5    5  /0   0.306    0.006
21382481-0         acc2       12:18:40  1609.0    6  /1   0.303    0.005
21382481-0         acc1       12:18:48  1523.3    8  /1   0.325    0.005

 TIME(s)  AVG_F/F  IMC_F IO(MBS) MPI%  G-POWER(T/U)  G-FREQ   G-UTIL(G/MEM)
 1.001    2.77/2.0 2.40  0.0     40.4  951.3 / 951.3  1.98     44%/2%
 1.046    2.77/2.0 2.40  0.0     43.4  966.9 / 966.9  1.98     45%/2%
 1.076    2.77/2.0 2.40  0.0     35.7  992.4 / 992.4  1.98     46%/2%
 1.153    2.77/2.0 2.40  0.0     48.0  933.3 / 933.3  1.98     43%/2%
 1.191    2.77/2.0 2.40  0.0     49.8  937.5 / 937.5  1.98     38%/2%
 1.094    2.77/2.0 2.40  0.0     36.8  988.2 / 988.2  1.98     45%/3%
```

Figure 4-8: Example job script using EAR on MareNostrum 5

```
srun --ear-policy=min_time --ear-policy-th=0.70 \
    --ear-user-db=more_test_min_time/app_metrics gmx_mpi mdrun \
    -s lignocellulose-rf.tpr -pin on -noconfout -nsteps 20000 -nstlist 200
```

Figure 4-9: Selecting min_time policy in a Slurm job.

CPU Frequency selection in EAR

Within EAR, you can manually select a CPU frequency in combination with a specific optimization policy.

- Use the `--ear-policy=policy_name` flag to select the desired policy.
- Use the `--ear-cpufreq=value` flag to specify the desired CPU frequency.
  The value must be provided in **kHz** (e.g., 2000000 for 2.0 GHz).

We evaluated the performance and energy consumption of GROMACS on two nodes using different EAR policy and threshold values.

- Without EAR, the performance was 55.856 ns/day.

Min-Time Policy:

- With the default threshold value (`--ear-policy-th=0.65`), performance was 53.363 ns/day.
- Using `--ear-policy-th=0.70`,
  performance slightly decreased to 53.215 ns/day.

Min-Energy Policy:

- With the default threshold (`--ear-policy-th=0.05`), performance increased to 56.635 ns/day.
- Using a higher threshold (`--ear-policy-th=0.10`), performance was 55.084 ns/day.

Monitoring Policy (CPU Frequency Scaling):

- At 2.0 GHz, performance was 55.722 ns/day.

- At 1.9 GHz, performance was 56.216 ns/day.
- At 1.8 GHz, performance dropped to 56.138 ns/day.

The energy consumption of GROMACS configuration is shown below:

Configuration	**Performance (ns/day)**	**Energy(J)**
GROMACS (No EAR)	55.856	238297
GROMACS (threshold=0.65)	53.363	232404 (`min_time`)
GROMACS (threshold=0.70)	53.215	219767 (`min_time`)
GROMACS (threshold=0.05)	56.635	223957 (`min_energy`)
GROMACS (threshold=0.10)	55.084	218022 (`min_energy`)
GROMACS (freq=2 GHz)	56.322	207252 (`monitoring`)
GROMACS (freq=1.9 GHz)	56.216	238364 (`monitoring`)
GROMACS (freq=1.8 GHz)	56.138	239232 (`monitoring`)

Links

- **EAR Policies**
  a complete guide on EAR policies (official documentation)
  https://gitlab.bsc.es/ear_team/ear/-/wikis/Architecture#policies
- **Example job scripts**
  Slurm job scripts using EAR are included in the "shortbench" repository of EPICURE's GitLab (see section 5.1)
  `Platforms/MareNostrum5/<partition>/<benchmark>/<benchmark>_job.sh`
- **Results**
  the results from the examples from above can be found in:
  `Platforms/MareNostrum5/EAR_metrics/<partition>/<benchmark>/`
- **EAR user guide**
  a complete user guide on using EAR (official documentation)
  https://gitlab.bsc.es/ear_team/ear/-/wikis/User-guide.

4.3. MERIC runtime system

MERIC runtime system from the MERIC energy efficient HPC software suite is one of the flagship codes of Performance Optimisation and Productivity (POP) EuroHPC Centre of Excellence (CoE). As a CoE flagship code, the MERIC is being deployed as public software module to all EuroHPC systems to provide energy consumption measurement, and in some systems also power management (currently Karolina, and Deucalion). Thanks to the MERIC, the user has unified interface to read energy consumption despite the underlying hardware is different, using a hardware-specific power monitoring system, or the energy consumption is exposed by one of many possible ways.

The user may use a command line utility to measure energy consumption of an application complete execution or link the application with MERIC library and instrument application's regions of interest.

```
mericStatic -e RAPL,NVML -- start &
/path/to/benchmark [app params]

mericStatic -- stop
mericStatic – eval
```

Figure 4-10: Example single-node usage of the `mericStatic` command line utility to measure energy consumption using RAPL and NVML performance counters.

```
mpirun -np $nnodes --map-by ppr:1:node mericStatic -e RAPL,NVML -- start &
# srun --overlap --ntasks-per-node 1 --nodes $nnodes mericStatic \
#    -e RAPL,NVML -- start &

srun /path/to/benchmark [app params]

mpirun -np $nnodes --map-by ppr:1:node mericStatic – stop
# srun  --ntasks-per-node 1 --nodes $nnodes mericStatic -- stop

mericStatic -- eval
```

Figure 4-11: Example multi-node usage of the `mericStatic` command line utility using `srun` or `mpirun` to start and stop the measurement in all allocated nodes (`$nnodes`).

```
Runtime [s] = 279.835
PCKG_ACTIVE_CORES_AVG_0 [J] = 858.128
PCKG_ACTIVE_CORES_AVG_1 [J] = 837.706
PCKG_0 [J] = 58676.202
PCKG_1 [J] = 59319.097

RAPL Energy consumption [J] = 117995.299
RAPL Energy consumption [Wh] = 32.776

Runtime [s] = 279.835
GPU_0 [J] = 21789.773
GPU_1 [J] = 22634.641
GPU_2 [J] = 21576.006
GPU_3 [J] = 21356.166
GPU_4 [J] = 20476.886
GPU_5 [J] = 20994.002
GPU_6 [J] = 20145.965
GPU_7 [J] = 21866.124

NVML Energy consumption [J] = 170839.563
NVML Energy consumption [Wh] = 47.455
```

Figure 4-12: Example output of the `mericStatic -- stop` command from a single node of Karolina equipped with two AMD EPYC CPUs and eight Nvidia GPUs (benchmark executed in two compute nodes).

```
job_id :2685249
job_id :2685249
Max Runtime [s] = 279.881
NVML Energy consumption [J] = 338331.189
RAPL Energy consumption [J] = 235217.793

Total Energy consumption [J] = 573548.982
Total Energy consumption [Wh] = 159.319
```

Figure 4-13: Example output of the `mericStatic -- eval` command summarising measurement from all used compute nodes (the same measurement as in Figure 4-12).

MERIC requires that user specify what power monitoring to use. Thus, user must know which ones are available in the system. For that purpose, the user may use the systemInfo MERIC utility, which prints details about the underlying hardware and its power monitoring and power management possibilities.

```
# SYSTEM INFORMATION
        CPU name:          AMD EPYC 7763 64-Core Processor
        Sockets per Node: 2
        Cores per Socket: 64
        Threads per Core: 1
        GPU name:          NVIDIA A100-SXM4-40GB
        GPUs per node:     8

# CPU FREQUENCIES
        Turbo CPU core frequencies:   3525000 kHz
        Nominal CPU core frequency:   2450000 kHz

# GPU FREQUENCIES
        Memory: 1215000 kHz      SM: 1410000 - 210000 kHz (81 steps)
        Default memory frequency: 1215000 kHz
        Default streaming multiprocessor frequency: 1095000 kHz

# CPU POWER LIMITS
        CPU max power limit:      280 W
        CPU power limit:              280 W

# GPU POWER LIMITS
        GPU max power limit:      400 W
        GPU min power limit:      100 W
        GPU default power limit: 400 W

# AVAILABLE POWER MONITORING SYSTEMS
        RAPL
        NVML
```

Figure 4-14: Output of the `systemInfo` utility when executed in a Karolina accelerated node.

Slurm job scripts using **`mericStatic`** utility can be found in the shortbench repository of EPICURE's GitLab (see section 5.1) in **`Platforms/Karolina/<partition>/<benchmark>/<benchmark>_job_meric.sh`**.

See MERIC runtime system user guide for more information on how to use MERIC to instrument an application, and how use MERIC to optimize an application energy efficiency.

https://code.it4i.cz/energy-efficiency/meric-suite/meric

4.4. COUNTDOWN

COUNTDOWN is a power management tool able to track MPI and application phases to automatically reduce power consumption of the computing elements during MPI communication and synchronization. The tool intercepts all MPI calls and execute the communication via an equivalent PMPI call, but after and before a prologue and an epilogue routine. These routines are defined in the "profile" and "event" COUNTDOWN modules, supporting monitoring and power management, respectively. Environment variables can be set to control the kind of HW performance counter, the configuration of the monitoring/management or the verbosity of logging. COUNTDOWN can be preloaded at runtime without source code modifications, or if needed it also provides a static-linking library to be used at compile time.

COUNTDOWN implements three complementary profiling strategies to monitor application behaviour at varying levels of granularity:

1. **MPI Profiler**
   This component collects detailed information about the MPI activity of each process. It records MPI communicators, groups, and the core ID where the process is running. These metrics help characterize communication behaviour and detect potential inefficiencies.
2. **Fine-Grain Micro-Architectural Profiler**
   Running in parallel with the MPI profiler, this component gathers micro-architectural metrics at every MPI call using the RDPMC instruction in user space to access Intel's Performance Monitoring Units (PMUs). It records values such as average core frequency, Time Stamp Counter (TSC), and instructions retired. Up to 8 configurable counters are available, allowing users to monitor application-specific low-level performance events. This fine-grain insight is valuable for identifying computational inefficiencies within MPI regions.
3. **Coarse-Grain Profiler**
   This profiler samples a broader range of hardware performance counters, including TSC, instructions retired, frequency, temperature, and C-state residencies at both core and uncore levels. It also monitors energy consumption and power usage using Intel's Running Average Power Limit (RAPL) interface. Because access to these low-level hardware counters typically requires elevated privileges, COUNTDOWN uses the `msr_safe` driver. This driver enables secure access for standard users to a restricted subset of privileged architecture registers without compromising system security. Due to the overhead of frequent sampling, this profiler operates on a time-based interval: data is collected only if a predefined time Ts has passed since the last sample. The fine-grain profiler checks this interval and, if exceeded, triggers a new coarse-grain sample to maintain synchronization.

Detailed data from the MPI profiler is stored in binary format to manage file size during long executions. In parallel, a human-readable text summary is also generated to provide an accessible overview of profiling results.

CNTD source code can be downloaded from:
https://github.com/EEESlab/countdown.git

The installation, using CMake, must be configured with the same MPI and compiler used for the application. A basic installation on Leonardo Booster can be achieved, for example, with the following commands

```
module purge
module load openmpi/4.1.6--gcc--12.2.0
module load cuda/12.1
cmake -DCNTD_ENABLE_CUDA=ON ../
make
```

Figure 4-15: Basic installation of CNTD.

where `-DCNTD_ENABLE_CUDA=ON` is needed to profile GPU metrics. For additional configuration options, the user can rely on installation instructions in https://github.com/EEESlab/countdown/blob/develop/README.md.

For the sampling approach of CNTD, the user needs to preload the library `libcntd.so`; the measurement can be customized with parameters setting, for example, the sampling interval and the name of the output directory.

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=8
#SBATCH --time=00:20:00
#SBATCH --exclusive
#SBATCH --gres=gpu:4
#SBATCH --partition=boost_usr_prod

module purge
module load profile/lifesc
module load gromacs/2022.3--openmpi--4.1.6--gcc--12.2.0-cuda-12.1

export OMP_NUM_THREADS=8
export OMP_PLACES=cores
export OMP_PROC_BIND=close

export GMX_ENABLE_DIRECT_GPU_COMM=1

export CNTD_SAMPLING_TIME=0.1
export CNTD_OUTPUT_DIR=CNTD
export LD_PRELOAD=<path-to-installation>/src/libcntd.so

srun -n 4 --cpu-bind=cores --cpus-per-task=$SLURM_CPUS_PER_TASK \
    gmx_mpi mdrun -s lignocellulose-rf.tpr -pin on -noconfout -nsteps 20000 \
    -nstlist 200 -dlb yes
```

Figure 4-16: Example job script using CNTD.

The default measurement configuration will provide tables with the following metrics:

- execution time,
- parallel information (number of MPI tasks, GPUs, nodes),
- energy of the job (PKG, DRAM, GPU),
- average power (PKG, DRAM, GPU),
- performance information (Time in MPI communications, maximal memory usage, CPU frequency, ...),
- GPU reporting (utilization, memory, temperature),
- more detailed information about MPI.

4.5. RAPL and NVIDIA NVML

Intel and AMD performance counters are exposed using tools and plugins like `perf`, `intel_rapl`, `amd_energy`, `msr_safe`, EAR, MERIC, GEOPM, COUNTDOWN, and many more. Each EuroHPC site uses a different interface to expose these counters. MERIC provides unified way to read RAPL from them all. It has been deployed already on several EuroHPC systems.

Similarly, information provided by NVML is used by MERIC and COUNTDOWN.

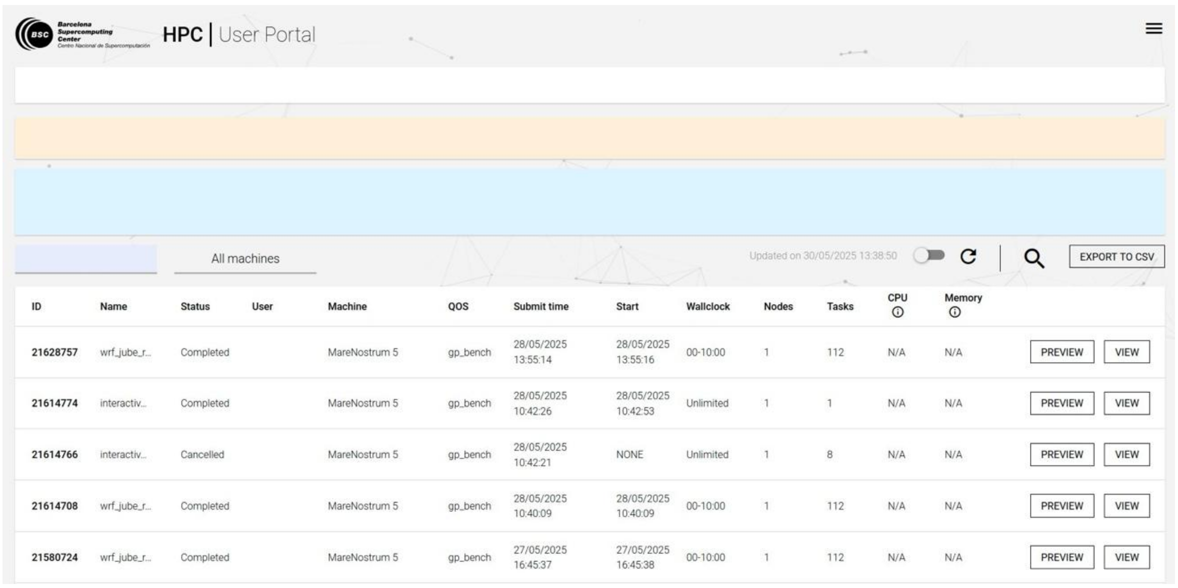
4.6. Dashboards

BSC HPC | User Portal

The "HPC User Portal" (https://hpcportal.bsc.es/) is a job and resource monitoring platform designed with the needs of HPC users in mind. It allows users to check the status and general resource usage metrics of their submitted jobs. In addition, the portal provides machine statistics, such as the number of available and allocated CPUs, for BSC's primary HPC systems. The platform is still under active development and will progressively offer more features over time. Currently, for the Accelerated (ACC) partition, power consumption data is not yet available, but it is being worked on. Some of the features provided by HPC portal are listed below:

Job monitoring

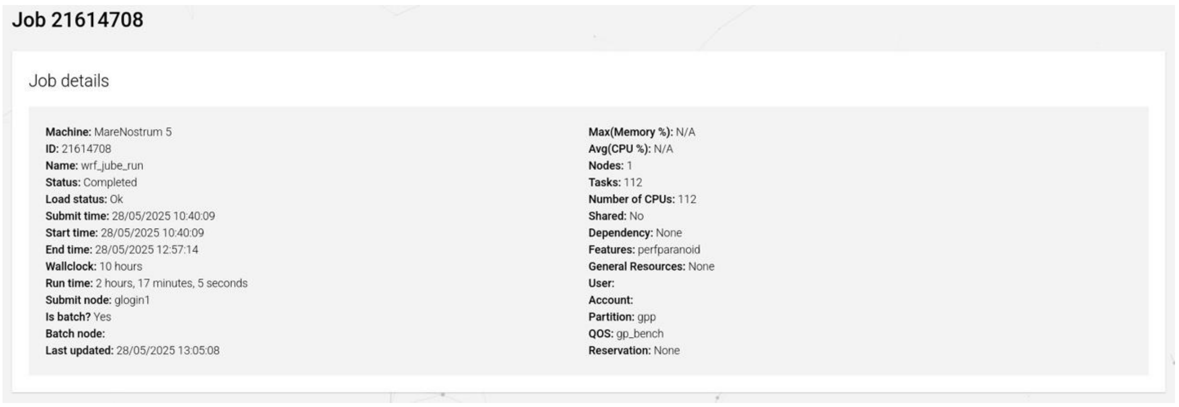
The main page of the HPC User Portal is the job monitoring screen. It will list all your jobs launched in all the machines by every account you have. This list contains a brief listing of the general characteristics of each job (like its name, user, status, node/task configuration...). If the job listed is in the "running" status, it will also show you the current CPU and memory usage.


Barcelona Supercomputing Center
Centro Nacional de Supercomputación
HPC | User Portal
All machines
Updated on 30/05/2025 13:38:50
EXPORT TO CSV

ID	Name	Status	User	Machine	QOS	Submit time	Start	Wallclock	Nodes	Tasks	CPU	Memory	
21628757	wrf_jube_r...	Completed		MareNostrum 5	gp_bench	28/05/2025 13:55:14	28/05/2025 13:55:16	00-10:00	1	112	N/A	N/A	PREVIEW VIEW
21614774	interactiv...	Completed		MareNostrum 5	gp_bench	28/05/2025 10:42:26	28/05/2025 10:42:53	Unlimited	1	1	N/A	N/A	PREVIEW VIEW
21614766	interactiv...	Cancelled		MareNostrum 5	gp_bench	28/05/2025 10:42:21	NONE	Unlimited	1	8	N/A	N/A	PREVIEW VIEW
21614708	wrf_jube_r...	Completed		MareNostrum 5	gp_bench	28/05/2025 10:40:09	28/05/2025 10:40:09	00-10:00	1	112	N/A	N/A	PREVIEW VIEW
21580724	wrf_jube_r...	Completed		MareNostrum 5	gp_bench	27/05/2025 16:45:37	27/05/2025 16:45:38	00-10:00	1	112	N/A	N/A	PREVIEW VIEW

Figure 4-17: Main page of the HPC User Portal.

Once a specific job is selected, we get the job details:



Job 21614708

Job details

Machine: MareNostrum 5
ID: 21614708
Name: wrf_jube_run
Status: Completed
Load status: Ok
Submit time: 28/05/2025 10:40:09
Start time: 28/05/2025 10:40:09
End time: 28/05/2025 12:57:14
Wallclock: 10 hours
Run time: 2 hours, 17 minutes, 5 seconds
Submit node: glogin1
Is batch? Yes
Batch node:
Last updated: 28/05/2025 13:05:08

Max(Memory %): N/A
Avg(CPU %): N/A
Nodes: 1
Tasks: 112
Number of CPUs: 112
Shared: No
Dependency: None
Features: perfparanoid
General Resources: None
User:
Account:
Partition: gpp
QOS: gp_bench
Reservation: None

Figure 4-18: Job details.

CPU usage, memory usage and power consumption are also provided.

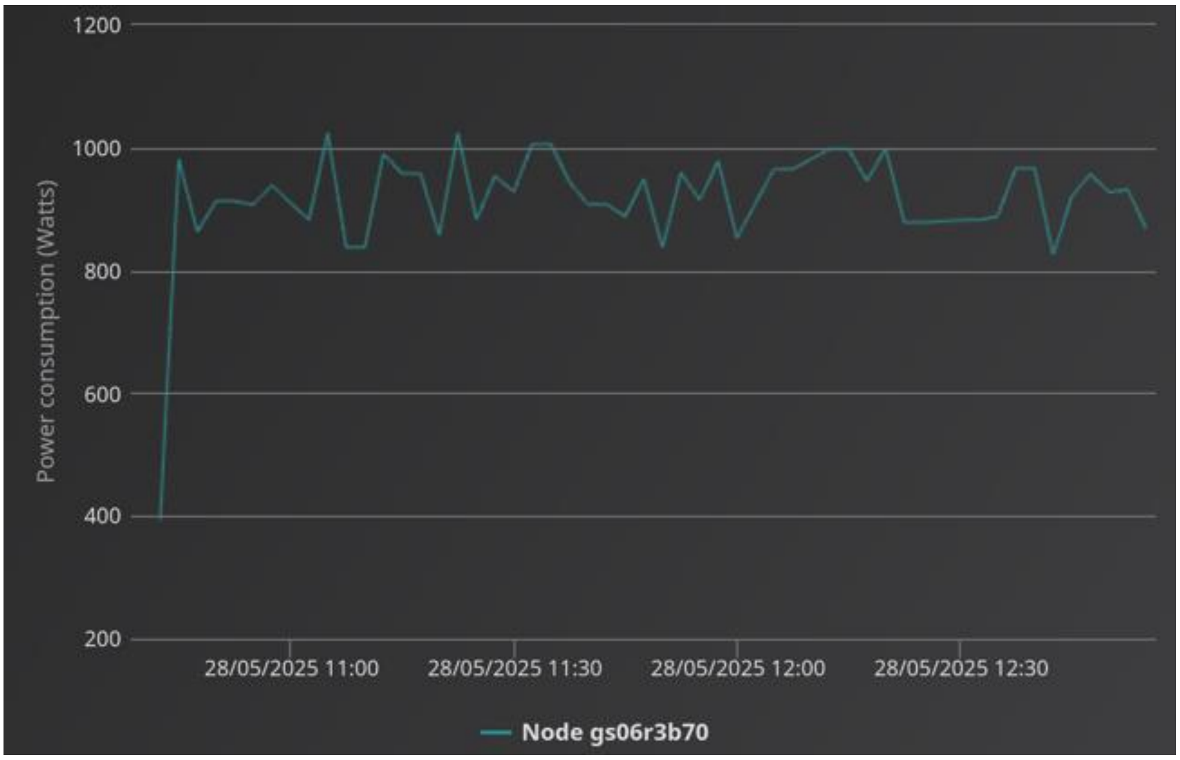


Figure 4-19: Power consumption.

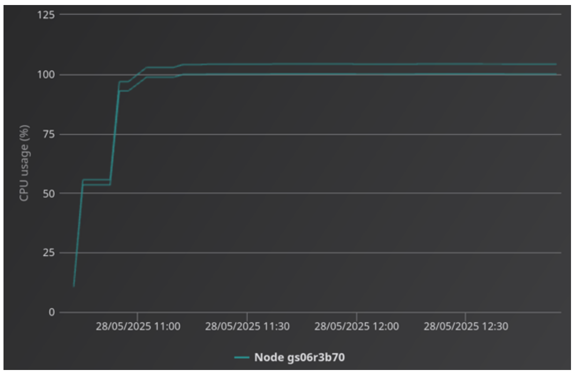


Figure 4-20: CPU usage.

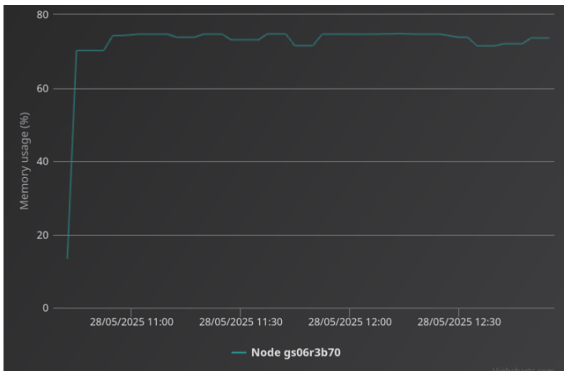


Figure 4-21: Memory usage.

LLview

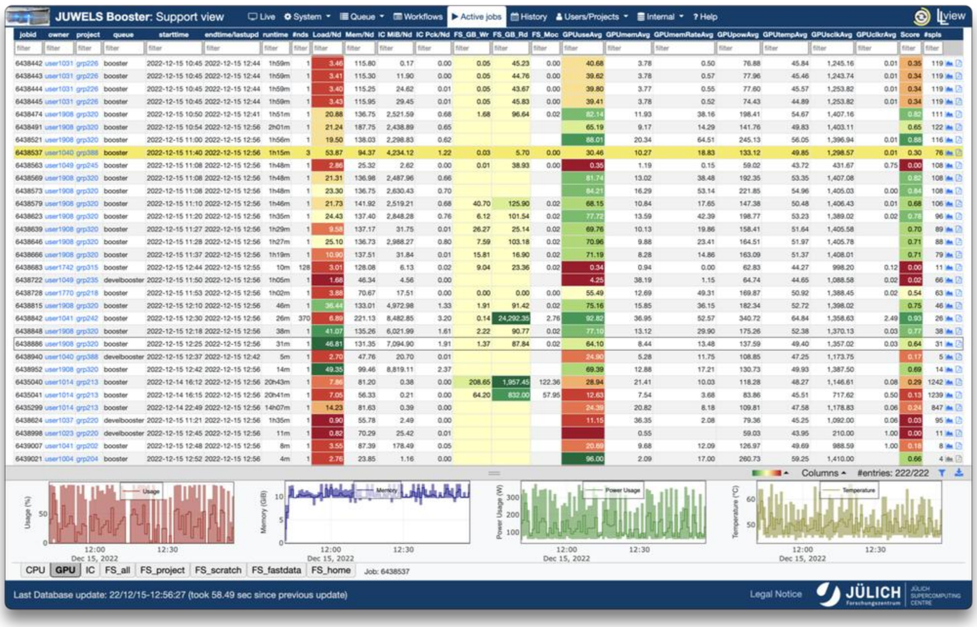
LLview is a set of software components for monitoring clusters controlled by a resource manager and scheduler system. It collects existing data from the system and presents it to the user via a web portal. (https://llview.fz-juelich.de/)

At JSC, LLview collects data from the Slurm workload manager, various daemons running on compute nodes, and sensors that either log information to files or interface with the Prometheus monitoring system. LLview then aggregates and reorganises the monitoring data, stores the information required for reporting in separate SQLite databases and presents it to the user via a web-based front-end portal.

The LLview Job Reporting web portal provides:

- job list tables, containing their aggregated performance information (for jobs that are running or have already finished within three weeks),
- timeline graphs per job for the key performance metrics,
- access to detailed job reports, including an interactive report or a static PDF version,
- role-based access to different levels of information,
- live view of the system.

In Figure 4-22, a snapshot of the Jobs Dashboard shows a table with one job per row and each column containing one of its metrics. Some columns are colour coded to indicate potential problems. Detailed job reports can be accessed in the rightmost columns. Selecting a row displays aggregated graphs at the bottom of the page.



jobid	owner	project	queue	starttime	endtime/lastupd	runtime	#nds	Load/Nd	Mem/Nd	IC MiB/Nd	IC Pck/Nd	FS_GB_Wr	FS_GB_Rd	FS_Moc	GPUuseAvg	GPUmemAvg	GPUmemRateAvg	GPUpowAvg	GPUtempAvg	GPUsclkAvg	GPUclkrAvg	Score	#spls
6438442	user1031	grp226	booster	2022-12-15 10:45	2022-12-15 12:44	1h59m	1	3.46	115.80	0.17	0.00	0.05	45.23	0.00	40.68	3.78	0.50	76.88	45.84	1,245.16	0.01	0.35	119
6438443	user1031	grp226	booster	2022-12-15 10:45	2022-12-15 12:44	1h59m	1	3.41	115.30	11.90	0.00	0.05	44.76	0.00	39.62	3.78	0.57	77.96	45.46	1,243.74	0.01	0.34	119
6438444	user1031	grp226	booster	2022-12-15 10:45	2022-12-15 12:44	1h59m	1	3.40	115.25	24.62	0.01	0.05	43.67	0.00	39.80	3.77	0.55	77.60	45.57	1,253.82	0.01	0.34	119
6438445	user1031	grp226	booster	2022-12-15 10:45	2022-12-15 12:44	1h59m	1	3.43	115.95	29.45	0.01	0.05	45.83	0.00	39.41	3.78	0.52	74.43	44.89	1,253.82	0.01	0.34	119
6438474	user1908	grp320	booster	2022-12-15 10:50	2022-12-15 12:41	1h51m	1	20.88	136.75	2,521.59	0.68	1.68	96.64	0.02	82.14	11.93	38.16	198.41	54.67	1,407.16		0.82	111
6438491	user1908	grp320	booster	2022-12-15 10:54	2022-12-15 12:56	2h01m	1	21.24	187.75	2,438.89	0.65				65.19	9.17	14.29	141.76	49.83	1,403.11		0.65	122
6438521	user1908	grp320	booster	2022-12-15 11:00	2022-12-15 12:56	1h56m	1	19.50	138.03	2,298.83	0.62				88.01	20.34	64.51	245.13	56.05	1,396.94	0.01	0.88	116
6438537	user1040	grp388	booster	2022-12-15 11:40	2022-12-15 12:56	1h15m	3	53.87	94.37	4,234.12	1.22	0.03	5.70	0.00	30.46	10.27	18.83	133.12	49.85	1,298.57	0.01	0.30	76
6438563	user1049	grp245	booster	2022-12-15 11:08	2022-12-15 12:56	1h48m	1	2.86	25.32	2.62	0.00	0.01	38.93	0.00	0.35	1.19	0.15	59.02	43.72	431.67	0.75	0.00	108
6438569	user1908	grp320	booster	2022-12-15 11:08	2022-12-15 12:56	1h48m	1	21.31	136.98	2,487.96	0.66				81.74	13.02	38.48	192.35	53.35	1,407.08		0.82	108
6438573	user1908	grp320	booster	2022-12-15 11:08	2022-12-15 12:56	1h48m	1	23.30	136.75	2,630.43	0.70				84.21	16.29	53.14	221.85	54.96	1,405.03	0.00	0.84	108
6438579	user1908	grp320	booster	2022-12-15 11:10	2022-12-15 12:56	1h46m	1	21.73	141.92	2,519.21	0.68	40.70	125.90	0.02	68.15	10.84	17.65	147.38	50.48	1,406.43	0.01	0.68	106
6438623	user1908	grp320	booster	2022-12-15 11:20	2022-12-15 12:56	1h35m	1	24.43	137.40	2,848.28	0.76	6.12	101.54	0.02	77.72	13.59	42.39	198.77	53.23	1,389.02	0.02	0.78	96
6438639	user1908	grp320	booster	2022-12-15 11:27	2022-12-15 12:56	1h29m	1	9.58	137.17	31.75	0.01	26.27	25.14	0.02	69.76	10.13	19.86	158.41	51.64	1,405.58		0.70	89
6438646	user1908	grp320	booster	2022-12-15 11:28	2022-12-15 12:56	1h27m	1	25.10	136.73	2,988.27	0.80	7.59	103.18	0.02	70.96	9.88	23.41	164.51	51.97	1,405.78		0.71	88
6438666	user1908	grp320	booster	2022-12-15 11:37	2022-12-15 12:56	1h19m	1	10.90	137.51	31.84	0.01	15.81	16.90	0.02	71.19	8.28	14.86	163.09	51.37	1,408.01		0.71	79
6438683	user1742	grp315	booster	2022-12-15 12:44	2022-12-15 12:55	10m	128	3.01	128.08	6.13	0.02	9.04	23.36	0.02	0.34	0.94	0.00	62.83	44.27	998.20	0.12	0.00	11
6438722	user1049	grp235	develbooster	2022-12-15 11:50	2022-12-15 12:56	1h05m	1	1.68	46.34	4.56	0.00				4.25	38.19	1.15	64.74	44.65	1,088.58	0.02	0.02	66
6438728	user1770	grp218	booster	2022-12-15 11:53	2022-12-15 12:56	1h02m	1	3.88	70.67	17.51	0.00	0.00	0.00	0.00	55.49	12.69	49.31	169.87	50.92	1,388.45	0.02	0.54	63
6438815	user1908	grp320	booster	2022-12-15 12:10	2022-12-15 12:56	46m	1	36.44	133.01	4,972.98	1.33	1.91	91.42	0.02	75.16	15.85	36.15	182.34	52.72	1,398.02		0.75	46
6438842	user1041	grp242	booster	2022-12-15 12:30	2022-12-15 12:56	26m	370	6.89	221.13	8,482.85	3.20	0.14	24,292.35	2.76	92.82	36.95	52.57	340.72	64.84	1,358.63	2.49	0.93	26
6438848	user1908	grp320	booster	2022-12-15 12:18	2022-12-15 12:56	38m	1	41.07	135.26	6,021.99	1.61	2.22	90.77	0.02	77.10	13.12	29.90	175.26	52.38	1,370.13	0.03	0.77	38
6438886	user1908	grp320	booster	2022-12-15 12:25	2022-12-15 12:56	31m	1	46.81	131.35	7,094.90	1.91	1.37	87.84	0.02	64.10	8.44	13.48	137.59	49.40	1,357.02	0.03	0.64	31
6438940	user1040	grp388	develbooster	2022-12-15 12:37	2022-12-15 12:42	5m	1	2.70	47.76	20.70	0.01				24.90	5.28	11.75	108.85	47.25	1,173.75		0.17	5
6438952	user1908	grp320	booster	2022-12-15 12:42	2022-12-15 12:56	14m	1	49.35	99.46	8,819.11	2.37				69.39	12.88	17.21	130.73	49.93	1,387.50		0.69	14
6435040	user1014	grp213	booster	2022-12-14 16:12	2022-12-15 12:56	20h43m	1	7.86	81.20	0.38	0.00	208.65	1,957.45	122.36	28.94	21.41	10.03	118.28	48.27	1,146.61	0.08	0.29	1242
6435041	user1014	grp213	booster	2022-12-14 16:15	2022-12-15 12:56	20h41m	1	7.05	56.33	0.21	0.00	64.20	832.00	57.95	12.63	7.54	3.68	83.86	45.51	717.62	0.50	0.13	1239
6435299	user1014	grp213	booster	2022-12-14 22:49	2022-12-15 12:56	14h07m	1	14.23	81.63	0.39	0.00				24.39	20.82	8.18	109.81	47.58	1,178.83	0.06	0.24	847
6438624	user1037	grp220	develbooster	2022-12-15 11:21	2022-12-15 12:56	1h35m	1	0.90	55.78	2.49	0.00				11.15	36.35	2.08	79.36	45.25	1,092.00	0.06	0.03	95
6438998	user1023	grp220	develbooster	2022-12-15 12:45	2022-12-15 12:56	11m	1	0.82	70.29	25.42	0.01					0.55		59.03	43.95	210.00	1.00	0.00	11
6439007	user1041	grp202	booster	2022-12-15 12:48	2022-12-15 12:56	8m	1	3.55	87.39	178.49	0.05				20.69	9.68	12.09	126.97	49.69	988.59	1.00	0.18	8
6439021	user1004	grp204	booster	2022-12-15 12:52	2022-12-15 12:56	4m	1	2.76	23.85	1.16	0.00				96.00	2.09	17.00	260.73	59.25	1,410.00		0.66	4

Last Database update: 22/12/15-12:56:27 (took 58.49 sec since previous update)

Figure 4-22: LLview Job reporting web portal.

The dashboard provides a wide range of metrics, grouped into the following categories:

- **Job metadata**
  Job ID, username, project id, start time, estimated end time,
- **CPU**
  average CPU usage, number of active physical/logical cores, memory usage,

- **GPU**
  GPU utilization, memory usage, temperature and performance states, indicators for potential throttling or reduced performance,
- **I/O and network activity**
  read/write throughput, file open/close operation rates, data and packet input/output rates.

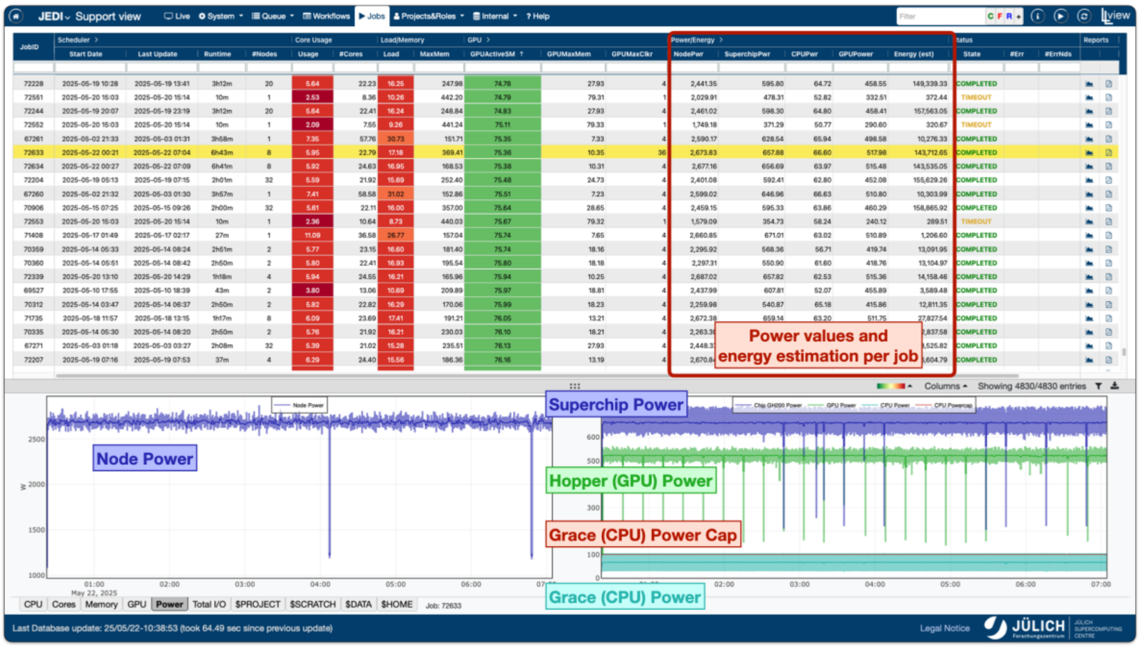
This comprehensive set of metrics enables users to monitor system performance, detect inefficiencies, and troubleshoot job behaviour effectively.

A significant recent enhancement to the LLview is the support for multiple levels of power telemetry data. This addition allows granular and comprehensive tracking of power consumption during job execution. Currently supported power metrics include:

- **Node Power**
  the total power draw for the entire node at the moment of sampling,
- **CPU Power**
  the instantaneous power consumed by the CPU package, including its memory controllers and system I/O,
- **GPU Power**
  the current power draw of each GPU device, including its onboard memory
- **Superchip Power**
  the power usage for each "superchip" (i.e. combined Grace and Hopper modules).

Energy consumption is calculated by aggregating power consumption data collected at one-minute intervals throughout the duration of a job. The resulting energy values are presented in a variety of units, including watt-hours (Wh), megajoules (MJ) and kilowatt-hours (kWh), providing flexibility for different analysis needs.

Figure 4-23 illustrates the display of power and energy values on the LLview web portal. The metrics highlighted within the red frame represent power and energy values, with each row corresponding to an individual job. For the currently selected job (indicated by the yellow highlighted row), detailed energy metric timelines are displayed at the bottom. These timelines provide a temporal view of power consumption, simplifying analysis of energy usage patterns throughout the job's execution.



JobID	Scheduler				Core Usage		Load/Memory		GPU			Power/Energy					Status		
	Start Date	Last Update	Runtime	#Nodes	Usage	#Cores	Load	MaxMem	GPUActiveSM ↑	GPUMaxMem	GPUMaxClkr	NodePwr	SuperchipPwr	CPUPwr	GPUPower	Energy (est)	State	#Err	#ErrNds
72228	2025-05-19 10:28	2025-05-19 13:41	3h12m	20	5.64	22.23	16.25	247.98	74.78	27.93	4	2,441.35	595.80	64.72	458.55	149,339.33	COMPLETED		
72551	2025-05-20 15:03	2025-05-20 15:14	10m	1	2.53	8.36	10.26	442.20	74.79	79.31	1	2,029.91	478.31	52.82	332.51	372.44	TIMEOUT		
72244	2025-05-19 20:07	2025-05-19 23:19	3h12m	20	5.64	22.41	16.24	248.84	74.83	27.93	4	2,461.02	598.30	64.80	458.41	157,563.05	COMPLETED		
72552	2025-05-20 15:03	2025-05-20 15:14	10m	1	2.09	7.55	9.26	441.24	75.11	79.33	1	1,749.18	371.29	50.77	290.60	320.67	TIMEOUT		
67261	2025-05-02 21:33	2025-05-03 01:31	3h58m	1	7.35	57.76	30.73	151.71	75.35	7.33	4	2,590.17	628.54	65.94	498.58	10,276.33	COMPLETED		
72633	2025-05-22 00:21	2025-05-22 07:04	6h43m	8	5.95	22.79	17.18	369.41	75.36	10.35	36	2,673.83	657.88	66.60	517.98	143,712.65	COMPLETED		
72634	2025-05-22 00:27	2025-05-22 07:09	6h41m	8	5.92	24.63	16.95	168.53	75.38	10.31	4	2,677.16	656.69	63.97	515.48	143,535.05	COMPLETED		
72204	2025-05-19 05:13	2025-05-19 07:15	2h01m	32	5.59	21.92	15.69	252.40	75.48	24.73	4	2,401.08	592.41	62.80	452.08	155,629.26	COMPLETED		
67260	2025-05-02 21:32	2025-05-03 01:30	3h57m	1	7.41	58.58	31.02	152.86	75.51	7.23	4	2,599.02	646.96	66.63	510.80	10,303.99	COMPLETED		
70906	2025-05-15 07:25	2025-05-15 09:26	2h00m	32	5.61	22.11	16.00	357.00	75.64	28.65	4	2,459.15	595.33	63.86	460.29	158,865.92	COMPLETED		
72553	2025-05-20 15:03	2025-05-20 15:14	10m	1	2.36	10.64	8.73	440.03	75.67	79.32	1	1,579.09	354.73	58.24	240.12	289.51	TIMEOUT		
71408	2025-05-17 01:49	2025-05-17 02:17	27m	1	11.09	36.58	26.77	157.04	75.74	7.65	4	2,660.85	671.01	63.02	510.89	1,206.60	COMPLETED		
70359	2025-05-14 05:33	2025-05-14 08:24	2h51m	2	5.77	23.15	16.60	181.40	75.74	18.16	4	2,295.92	568.36	56.71	419.74	13,091.95	COMPLETED		
70360	2025-05-14 05:51	2025-05-14 08:42	2h50m	2	5.80	22.41	16.93	195.54	75.80	18.18	4	2,297.31	550.90	61.60	418.76	13,104.97	COMPLETED		
72339	2025-05-20 13:10	2025-05-20 14:29	1h18m	4	5.94	24.55	16.21	165.96	75.94	10.25	4	2,687.02	657.82	62.53	515.36	14,158.46	COMPLETED		
69527	2025-05-10 17:55	2025-05-10 18:39	43m	2	3.80	13.06	10.69	209.89	75.97	18.81	4	2,437.99	607.81	52.07	455.89	3,589.48	COMPLETED		
70312	2025-05-14 03:47	2025-05-14 06:37	2h50m	2	5.82	22.82	16.29	170.06	75.99	18.23	4	2,259.98	540.87	65.18	415.86	12,811.35	COMPLETED		
71735	2025-05-18 11:57	2025-05-18 13:15	1h17m	8	6.09	23.69	17.41	191.21	76.05	13.21	4	2,672.38	659.14	63.20	511.75	27,827.54	COMPLETED		
70335	2025-05-14 05:30	2025-05-14 08:20	2h50m	2	5.76	21.92	16.21	230.03	76.10	18.21	4	2,263.3	[illegible]	[illegible]	[illegible]	2,837.58	COMPLETED		
67271	2025-05-03 01:18	2025-05-03 03:27	2h08m	32	5.39	21.02	15.28	235.51	76.13	27.93	4	2,448.3	[illegible]	[illegible]	[illegible]	,525.82	COMPLETED		
72207	2025-05-19 07:16	2025-05-19 07:53	37m	4	6.29	24.40	15.56	186.36	76.16	13.19	4	2,670.8	[illegible]	[illegible]	[illegible]	,604.79	COMPLETED		

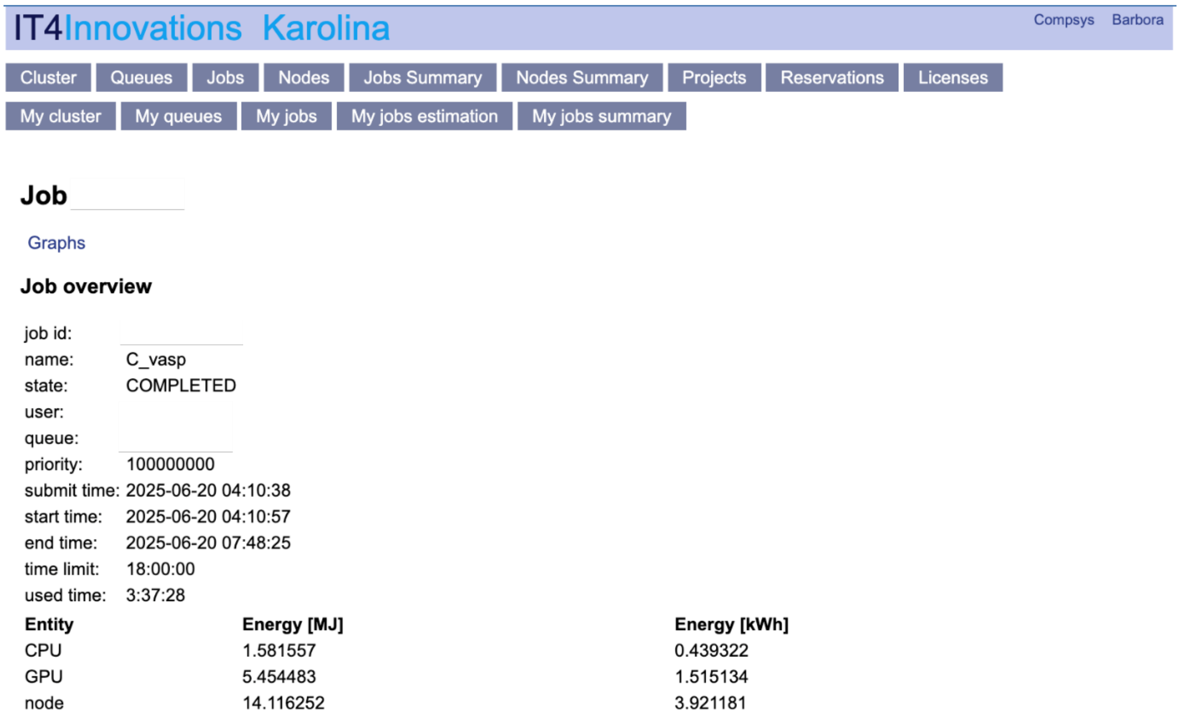
Last Database update: 25/05/22-10:38:53 (took 64.49 sec since previous update)

Figure 4-23: Power and energy values displayed on LLview web portal.

IT4Innovations User Portal and SCS Information System

For information about the current clusters' usage, IT4I users can go to User Portal https://extranet.it4i.cz/rsweb. They can switch between the clusters by clicking on their names in the upper right corner. Users can filter their search by clicking on the respective keywords.

In addition to general information about the jobs, like runtime, queue, etc., the portal now also contains information about CPU, GPU, and entire node energy consumption for **any job**. Users can also check the power consumption timeline of selected components of the compute nodes. Examples of these reports are shown in the figures below.



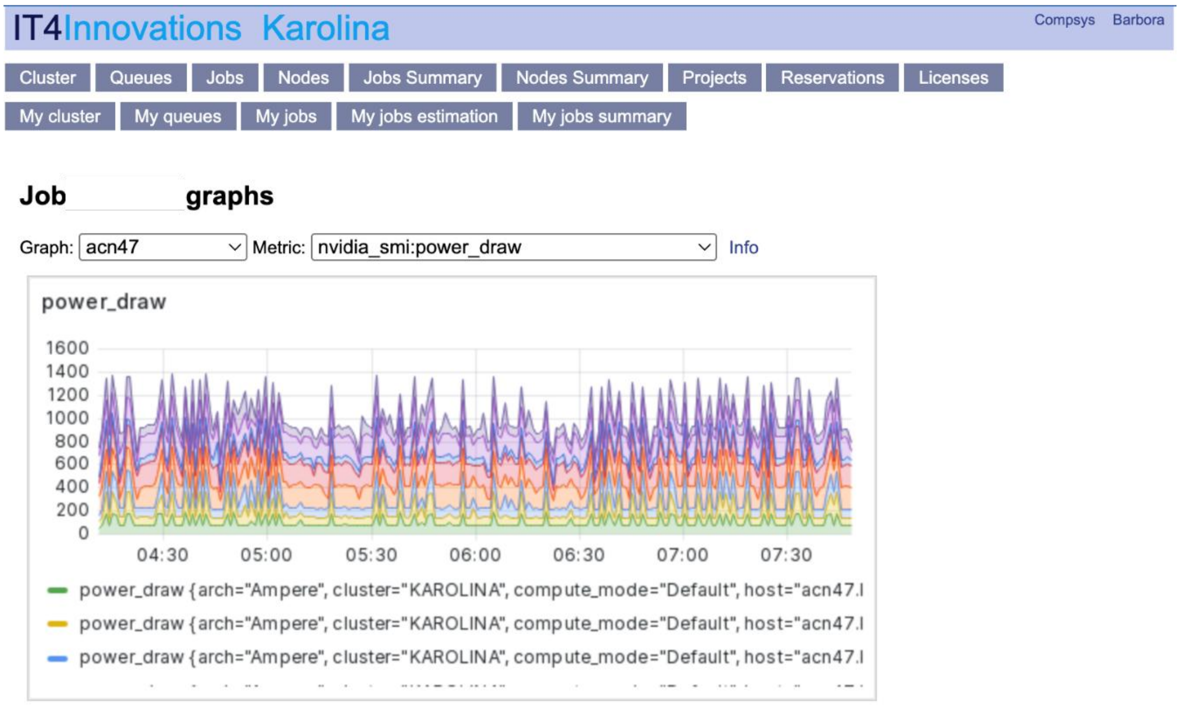
Job

Job overview

job id:	
name:	C_vasp
state:	COMPLETED
user:	
queue:	
priority:	100000000
submit time:	2025-06-20 04:10:38
start time:	2025-06-20 04:10:57
end time:	2025-06-20 07:48:25
time limit:	18:00:00
used time:	3:37:28

Entity	Energy [MJ]	Energy [kWh]
CPU	1.581557	0.439322
GPU	5.454483	1.515134
node	14.116252	3.921181

Figure 4-24: Example of job information provided to users, including energy consumption of CPUs and GPUs.



Job graphs

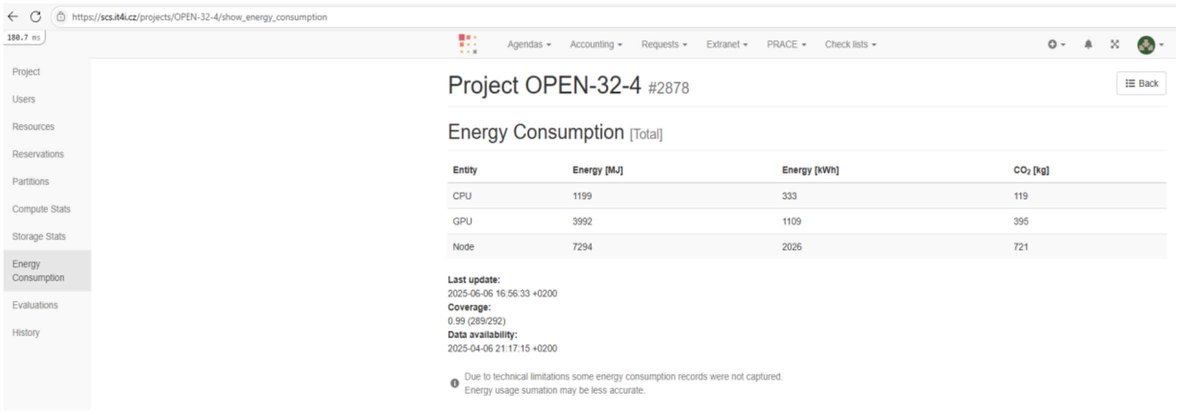
Graph: acn47 Metric: nvidia_smi:power_draw

power_draw {arch="Ampere", cluster="KAROLINA", compute_mode="Default", host="acn47.I
power_draw {arch="Ampere", cluster="KAROLINA", compute_mode="Default", host="acn47.I
power_draw {arch="Ampere", cluster="KAROLINA", compute_mode="Default", host="acn47.I

Figure 4-25: Users can also visualize the power consumption of their job in time. This example shows the power consumption of individual GPUs on a selected compute node.

The IT4Innovations Information System (SCS IS) (https://scs.it4i.cz) is a comprehensive platform for managing the lifecycle of HPC projects. It allows users and primary investigators to manage project applications, memberships, and resources from the initial request through to completion.

During the project's active phase, the system provides detailed monitoring capabilities. This includes tracking the usage of allocated computing resources, which are measured in node hours, against the approved allocation. As shown in the provided image, the system also offers a specific “Energy Consumption” report. This report details the energy used in MegaJoules (MJ) and Kilowatt-hours (kWh), and the associated carbon footprint in CO_2 (kg), with data broken down by CPU, GPU, and node usage.



Project OPEN-32-4 #2878

Energy Consumption [Total]

Entity	Energy [MJ]	Energy [kWh]	CO_2 [kg]
CPU	1199	333	119
GPU	3992	1109	395
Node	7294	2026	721

Last update:
2025-06-06 16:56:33 +0200
Coverage:
0.99 (289/292)
Data availability:
2025-04-06 21:17:15 +0200

Due to technical limitations some energy consumption records were not captured.
Energy usage sumation may be less accurate.

Figure 4-26: Example of energy consumption of CPUs, GPUs and nodes per project.

5. Overview

Since not all machines use the same metrics (see the "Measurements" section for the different machines), the plots shown are just an indication. Moreover, energy usage by e.g. network components or storage is not measured. If in the future one runtime system would be available on all machines, it would be easier to compare these values. We also learned that the sampling frequency for the metrics might influence the results significantly.

All runs always use full nodes. The most optimal configuration on one node (combination of MPI ranks, threads and GPUs if applicable) is taken as a baseline. This configuration is used in subsequent runs on two, four, eight, … nodes. Consider that the graphs show always the number of nodes, irrespective of the number of CPU cores or GPUs that might be different in the machines.

It becomes clear from the graphs, if not known already, that it does not make sense to keep increasing the number of nodes hoping that computations will finish more rapidly at a much lower energy cost. And it is important to determine the most optimal combination of MPI ranks, threads and GPUs before submitting a whole bunch of computations.

For GROMACS, CP2K and NAMD, we show following plots:

- "Performance-Energy" plot per machine,
- "Energy usage" for all machines.

In addition, we also show for GROMACS and NAMD

- "Normalized energy usage per ns/day".

All the data shown in the graphs and tables in the following sections are available in the different Platform folders of the *shortbench* repository of EPICURE's GitLab. The Excel file combining all data can be found there as well.

5.1. Job script examples

The table below contains references to job script examples used for application executions with power energy measurements. For some machines only the regular Slurm scripts are available. For others, the example scripts (also) contain references to external systems (MareNostrum5, EAR; Leonardo, CINEMON and COUNTDOWN; Karolina, MERIC).

Machine	Application	Partition (library)	Job script example
MareNostrum5	GROMACS	CPU-X86 (EAR)	MN5-GROMACS-CPU-X86 example
		GPU (EAR)	MN5-GROMACS-GPU example
	CP2K	CPU-X86 (EAR)	MN5-CP2K-CPU-X86 example
		GPU (EAR)	MN5-CP2K-GPU example
	NAMD	CPU-X86 (EAR)	MN5-NAMD-CPU-X86 example
		GPU (EAR)	MN5-NAMD-GPU example
DECAULION	GROMACS	CPU-X86	DECAULION-GROMACS-CPU-X86 example
		CPU-ARM	DECAULION-GROMACS-CPU-ARM example
		GPU	DECAULION-GROMACS-GPU example
	CP2K	CPU-X86	DECAULION-CP2K-CPU-X86 example
		CPU-ARM	DECAULION-CP2K-CPU-ARM example
		GPU	DECAULION-CP2K-GPU example
	NAMD	CPU-X86	DECAULION-NAMD-CPU-X86 example
		CPU-ARM	N/A
		GPU	DECAULION-NAMD-GPU example
MELUXINA	GROMACS	CPU-X86	MELUXINA-GROMACS-CPU-X86 example
		GPU	MELUXINA-GROMACS-GPU example
	CP2K	GPU-X86	MELUXINA-CP2K-CPU-X86 example
		GPU	MELUXINA-CP2K-GPU example
	NAMD	CPU-X86	MELUXINA-NAMD-CPU-X86 example
		GPU	MELUXINA-NAMD-GPU example
VEGA	GROMACS	CPU	VEGA-GROMACS-CPU-X86 example
		GPU	VEGA-GROMACS-GPU example
	CP2K	CPU	VEGA-CP2K-CPU-X86 example
		GPU	VEGA-CP2K-GPU example
	NAMD	CPU	VEGA-NAMD-CPU-X86 example
		GPU	VEGA-NAMD-GPU example
DISCOVERER	GROMACS	CPU	DISCOVER-GROMACS-CPU-X86 example
	CP2K	CPU	DISCOVER-CP2K-CPU-X86 example
	NAMD	CPU	DISCOVER-NAMD-CPU-X86 example

Machine	Application	Partition (library)	Job script example
CINECA	GROMACS	CPU (CINEMON)	CINECA-GROMACS-CPU-X86 example
		GPU (CINEMON)	CINECA-GROMACS-GPU example
		GPU (COUNTDOWN)	CINECA-GROMACS-GPU example
	CP2K	CPU (CINEMON)	CINECA-CP2K-CPU-X86 example
		GPU (CINEMON)	CINECA-CP2K-GPU example
		GPU (COUNTDOWN)	CINECA-CP2K-GPU example
	NAMD	CPU (COUNTDOWN)	CINECA-NAMD-CPU-X86 example
		GPU (CINEMON)	CINECA-NAMD-GPU example
KAROLINA	GROMACS	CPU	KAROLINA-GROMACS-CPU-X86 example
		CPU (MERIC)	KAROLINA-GROMACS-CPU-X86 example
		GPU	KAROLINA-GROMACS-GPU example
		GPU (MERIC)	KAROLINA-GROMACS-GPU example
	CP2K	CPU	KAROLINA-CP2K-CPU-X86 example
		CPU (MERIC)	KAROLINA-CP2K-CPU-X86 example
		GPU	KAROLINA-CP2K-GPU example
		GPU (MERIC)	KAROLINA-CP2K-GPU example
	NAMD	CPU	KAROLINA-NAMD-CPU-X86 example
		CPU (MERIC)	KAROLINA-NAMD-CPU-X86 example
		GPU	KAROLINA-NAMD-GPU example
		GPU (MERIC)	KAROLINA-NAMD-GPU example
LUMI	GROMACS	CPU	LUMI-GROMACS-CPU-X86 example
		GPU	LUMI-GROMACS-GPU example
	CP2K	CPU	LUMI-CP2K-CPU-X86 example
		GPU	LUMI-CP2K-GPU example
	NAMD	CPU	LUMI-NAMD-CPU-X86 example
		GPU	LUMI-NAMD-GPU example
JEDI	GROMACS	GPU	JEDI-GROMACS-GPU example

5.2. Performance-energy graph

The performance for GROMACS and NAMD is reported by the programs in units of [ns/day]. This reported performance gives an indication of the number of nanoseconds in simulation time can be executed with an equivalent runtime of 1 day of HPC resources.

For CP2K, the performance is not reported directly by the program. Instead, the efficiency (η_i) is used, calculated with the following formula:

$$\eta_i = \frac{n_0 \cdot t_0}{n_i \cdot t_i},$$

with n_0, n_i the amount nodes in the baseline and i-th calculation, and t_0, $t_i$the wall time for the calculation to complete.

Depending on the benchmark, the normalized performance (GROMACS, NAMD) or efficiency (CP2K) and the consumed energy are shown simultaneously in the performance-energy graph. Both quantities are plotted such that they start in the same point, at the baseline of one full node ($n_0 = 1$). From this baseline quantity, a dotted line is drawn. The graph has two y-axes, on the left and the right of the graph, where the normalized performance or efficiency, and total energy consumption are given respectively.

The normalized performance is obtained by dividing the performance by the number of nodes used for that calculation. The resulting quantity, the normalized performance, is an indication for the computation time if the equivalent calculation is performed on the baseline system of 1 node. For GROMACS and NAMD, the performance is given in units of [ns/day], which indicates how many nanoseconds of simulation time can be computed in one day of HPC calculation. This value is expressed in units of [ns/day/node] and [μs/day/node] for GROMACS and NAMD respectively. Example: if the performance is 12.90 ns/day on two nodes, it is shown as 12.90 / 2 = 6.45 ns/day/node.

The energy on the graph is the total energy consumption reported for the number of nodes, expressed in [kJ]. This quantity does not need to be rescaled as the same calculation is performed on the different systems number of nodes. In general, more nodes require less computation time but more simultaneous power consumption, such that the overall energy consumption is in general larger for multiple nodes.

Furthermore, the 'Net Efficiency Loss' is given in grey on the figures, which corresponds to a product of the net performance by the consumed energy. The resulting quantity gives an indication to the increased power, taking into account the changes in performance when the code is run on a different number of nodes. A line that goes up from the baseline indicates a higher than expected power consumption, when the line drops below the baseline, the calculation will use less power taking into account the performance loss of running on multiple nodes.

In general, it is expected that the efficiency or performance of the calculation will go down on multiple nodes, and the energy consumption and net efficiency loss will increase. The values will deviate from the baseline. In the ideal cases, the three lines would stay as close to the dotted line as possible.

CPU

GROMACS

In Figure 5-1 and Figure 5-2, the result for the GROMACS tests on CPU are shown.

The results for LUMI, Karolina, Deucalion and Discoverer are similar, with relatively good scaling around ~75% on 16 nodes and a relative low power increase to ~1.3 times the baseline. The net efficiency loss always lies relatively close to the baseline, which means that the energy increase originates from the lower performance on multiple nodes.

For Vega, the baseline power consumption is higher, with an efficiency that stays close to the baseline. Comparing with the machines mentioned above, it gives the suspicion that the baseline values have a lower performance, whereas the code runs slightly better on multiple nodes.

On Leonardo, MareNostrum 5 and MeluXina, the power increase is more significant. On Leonardo, this is accompanied with a lower performance, where the increase in power thus probably originates from the less efficient scaling of the code across the nodes. This is visible in the net efficiency loss that stays close to the baseline

MareNostrum 5 shows a better performance, where the increase in power originates from the higher number of nodes and not the efficiency. This is visible in the net efficiency loss that goes up from the baseline.

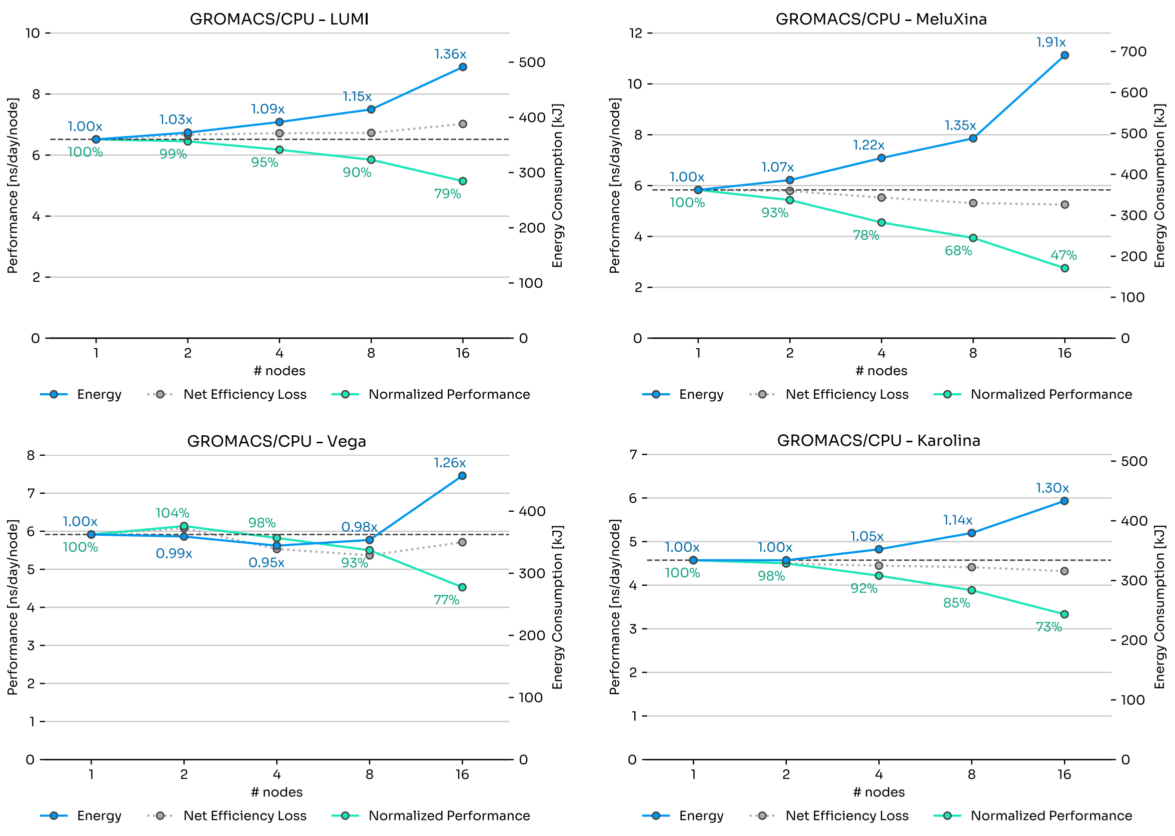


Figure 5-1: Graphs for GROMACS CPU for LUMI, MeluXina Vega and Karolina.

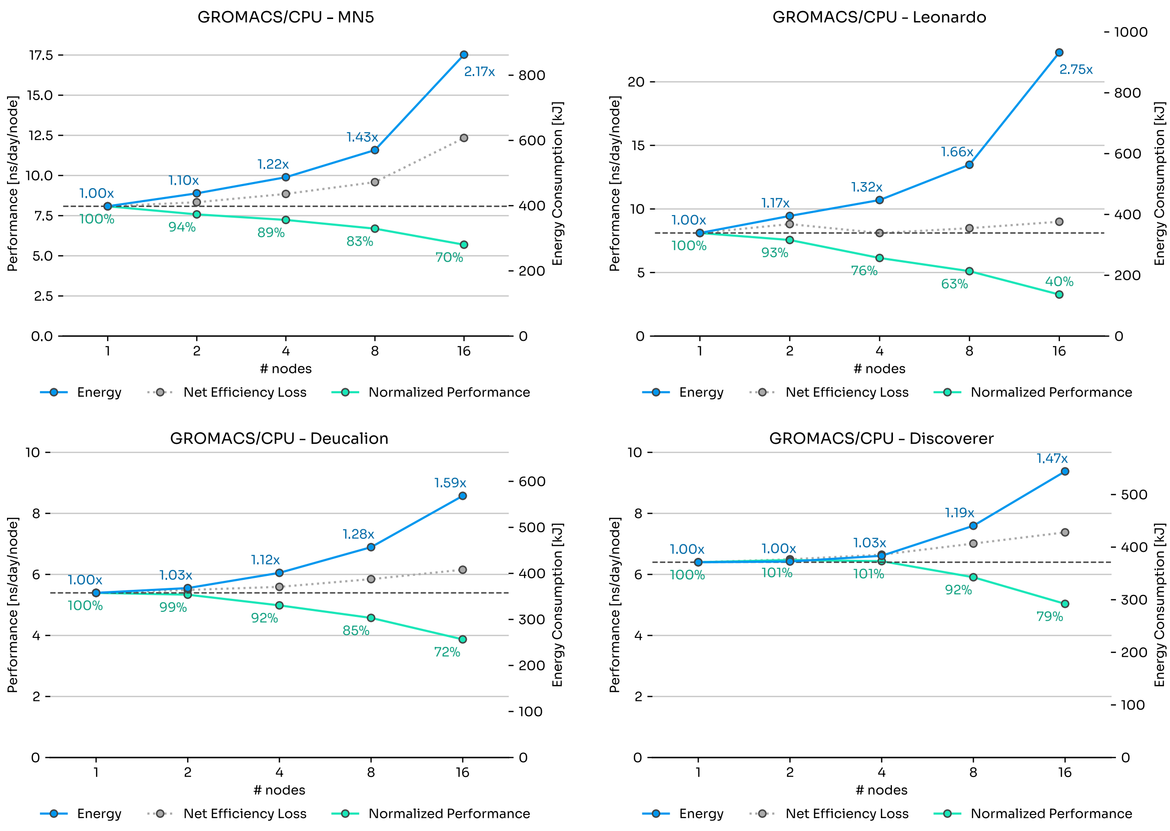


Figure 5-2: Graphs for GROMACS CPU for MareNostrum 5, Leonardo, Deucalion and Discoverer.

CP2K

The power consumption in [kJ] and efficiency in [%] for CP2K on CPU for the various machines is given in Figure 5-3 and Figure 5-4.

For Leonardo, there is a large increase in power consumption for 16 nodes, but this likely originates from the relative bad scaling of CP2K across several nodes; the net efficiency loss stays almost constant across the calculation.

The other machines give a gradual increase in energy consumption towards ~2 times the baseline power, with the efficiency dropping towards ~30%.

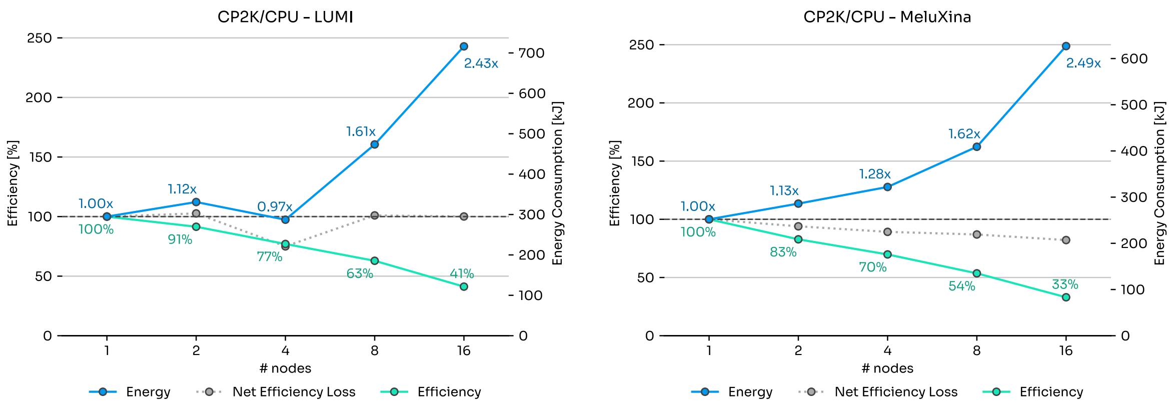


Figure 5-3: Graphs for CP2K CPU for LUMI and MeluXina.

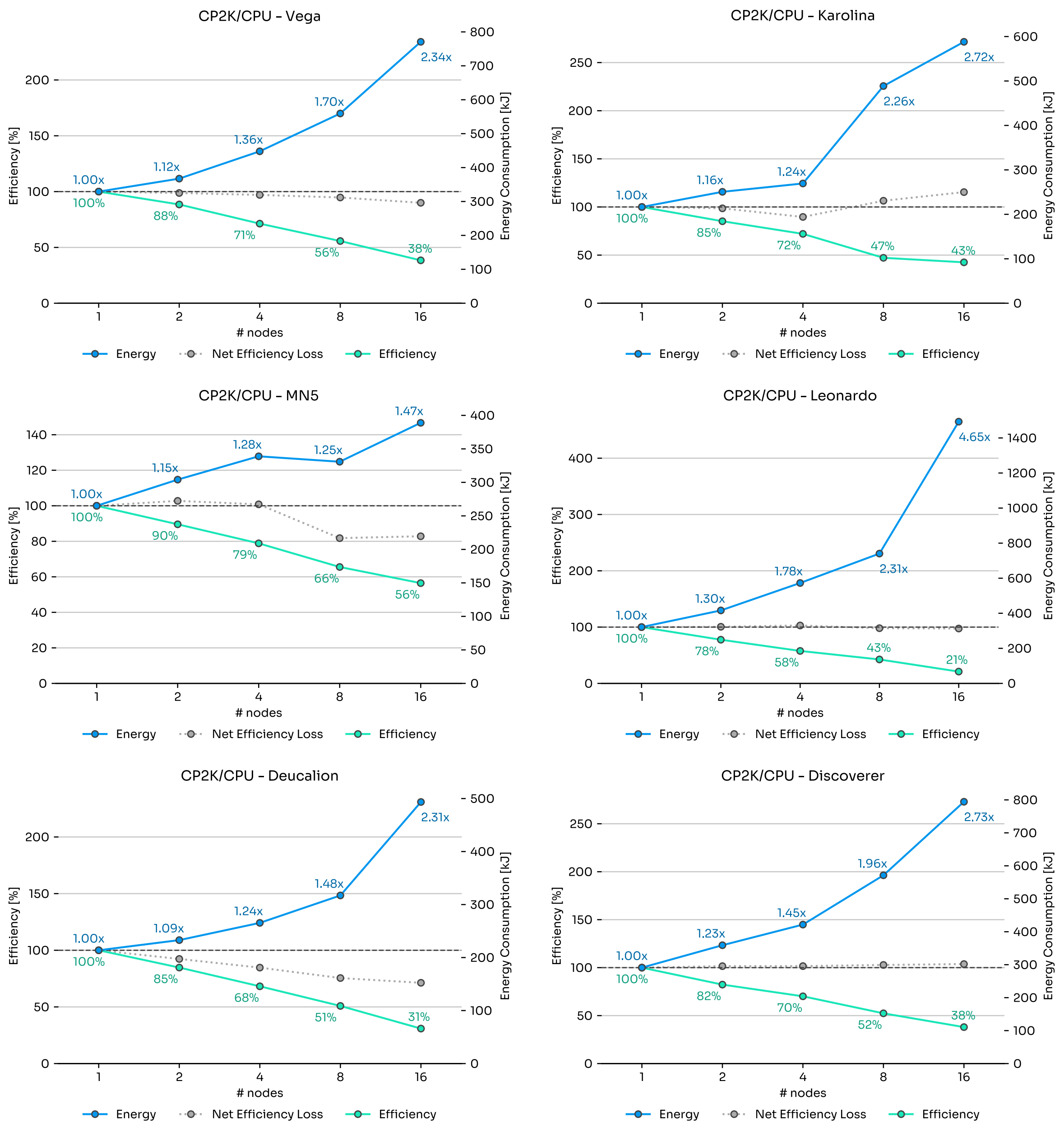


Figure 5-4: Graphs for CP2K CPU for Vega, Karolina, MareNostrum 5, Leonardo, Deucalion and Discoverer.

NAMD

In Figure 5-5, the results for NAMD on CPU are shown. On LUMI, Leonardo, MareNostrum 5 and Deucalion relatively good scaling is achieved with a net efficiency loss that stays close to the baseline. On MeluXina, the efficiency drops significantly but the power increase is relatively small, leading to a net efficiency loss that goes down from the baseline.

NAMD has internal parallelisation routines. One of these sets the FFT-grid, which is thus dependent on the number of tasks. This can explain the jumps in efficiency, while the net efficiency loss stays close to the baseline. For example, for Discoverer, the parallelization for 8 nodes is not a direct multiple of the number of tiles, which appears to be more efficient compared to a perfect tiling on the other configurations. A similar behaviour appears on MareNostrum 5 with 16 nodes.

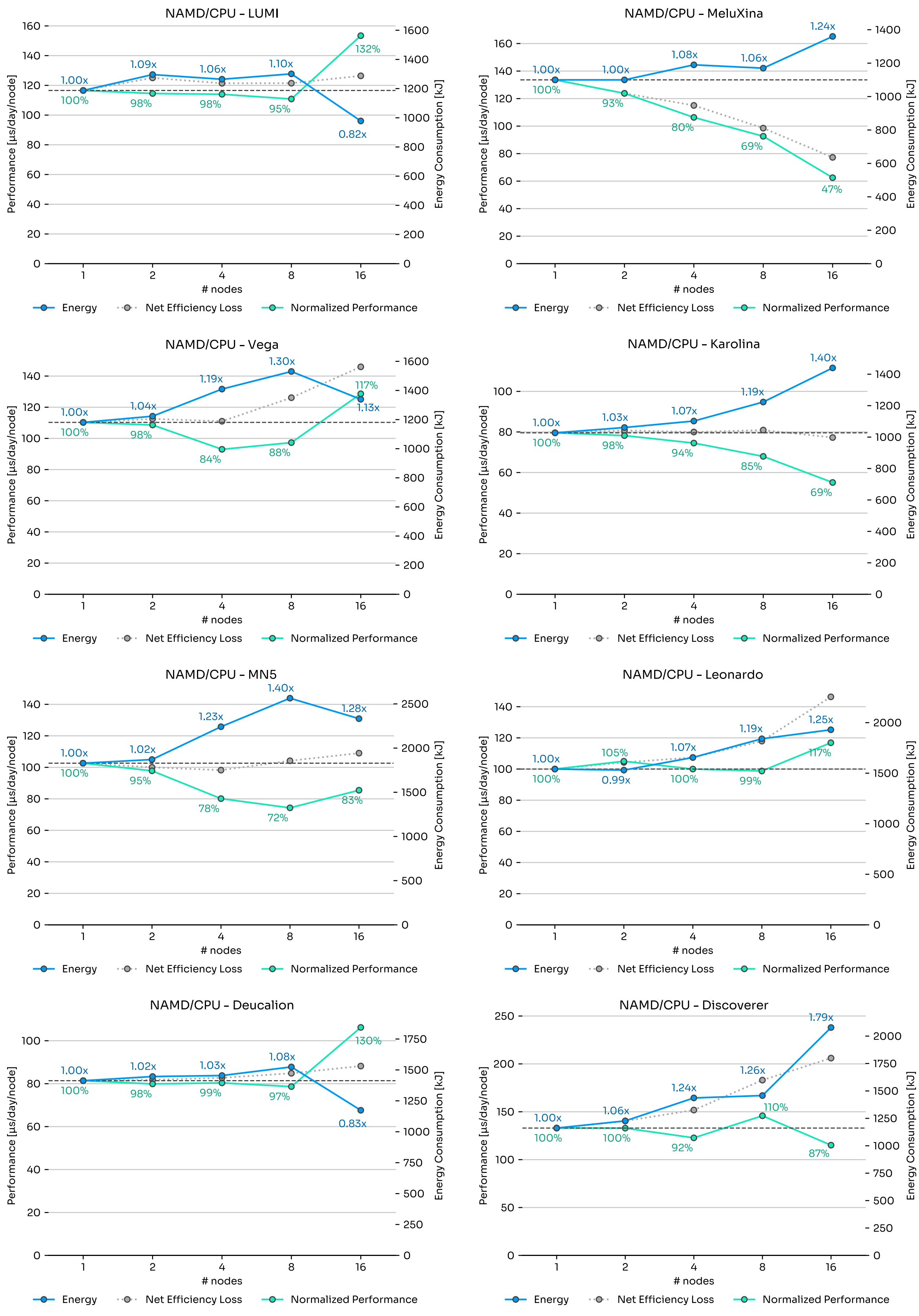


Figure 5-5: Graphs for NAMD CPU.

GPU

GROMACS

The results for the performance in [ns/day/node] and the power consumption in [kJ] is shown in Figure 5-7.

The results for LUMI, MeluXina and Karolina are similar, with an efficiency that drops to ~30% and a power increase of ~3 times the baseline.

For Vega, MareNostrum 5 and JEDI, the power increase is more substantial as well as the efficiency that drops significantly. For Vega, there is a large increase of power consumption around 8 nodes. As the net efficiency loss is under the baseline. This indicates that there is a problem with the scaling beyond 8 nodes.

Leonardo reports the best scaling for GROMACS on GPU, but the reported baseline performance is lower than on the other machines.

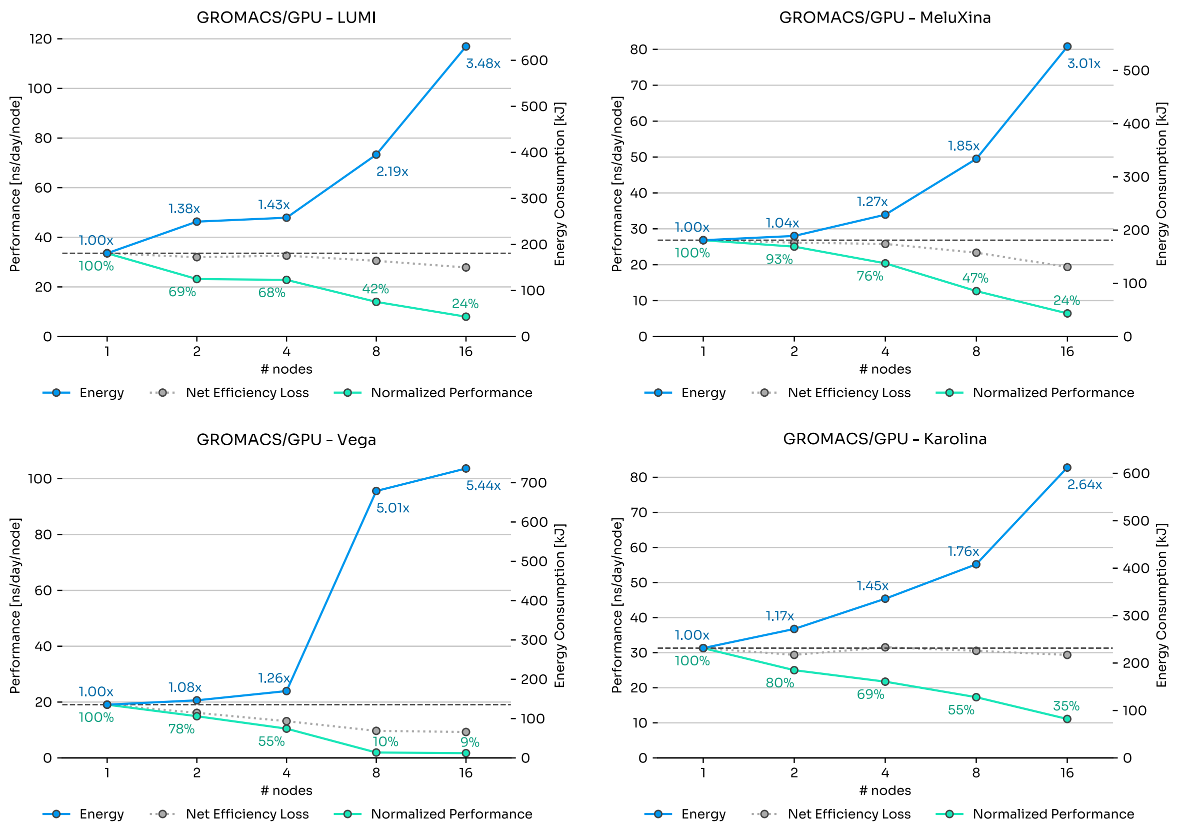


Figure 5-6: Graphs for GROMACS GPU for LUMI, MeluXina, Vega and Karolina.

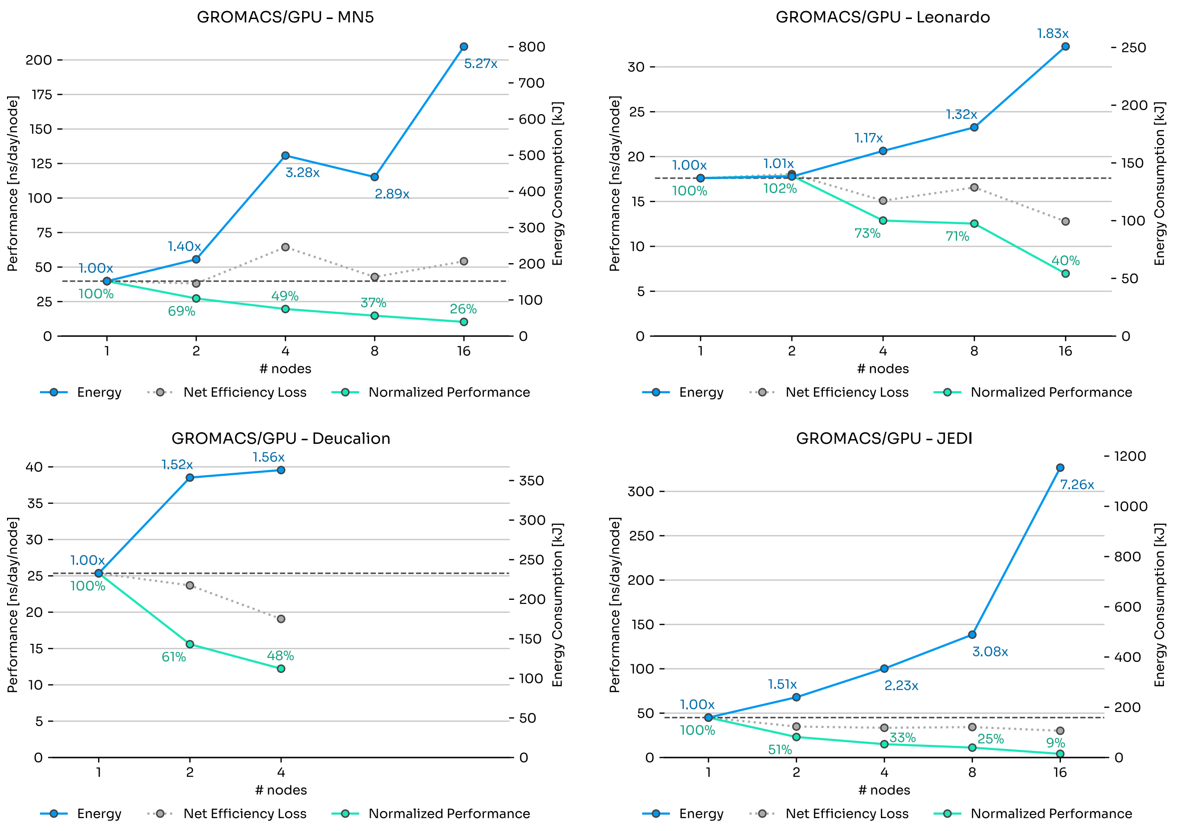


Figure 5-7: Graphs for GROMACS GPU for MareNostrum 5, Leonardo, Deucalion and JEDI.

CP2K

In Figure 5-8 the results for the scaling of CP2K on GPU are shown.

MeluXina, Vega, Karolina, MareNostrum 5 and Deucalion report similar scaling on multiple nodes, with a large power increase towards ~5.5 times the baseline and a low efficiency of ~15%. For Vega, the net efficiency loss drops significantly below the baseline, which indicates performance scaling problems. The other listed machines have a relative stable net efficiency loss, indicating that the increased power consumption originates from a less ideal scaling.

For LUMI and Leonardo, the scaling and power consumption is slightly better than the other machines.

Most machines indicate a larger jump around 8 nodes in both a lower performance and a higher power consumption. This is likely because the program has difficulties scaling to such many GPUs.

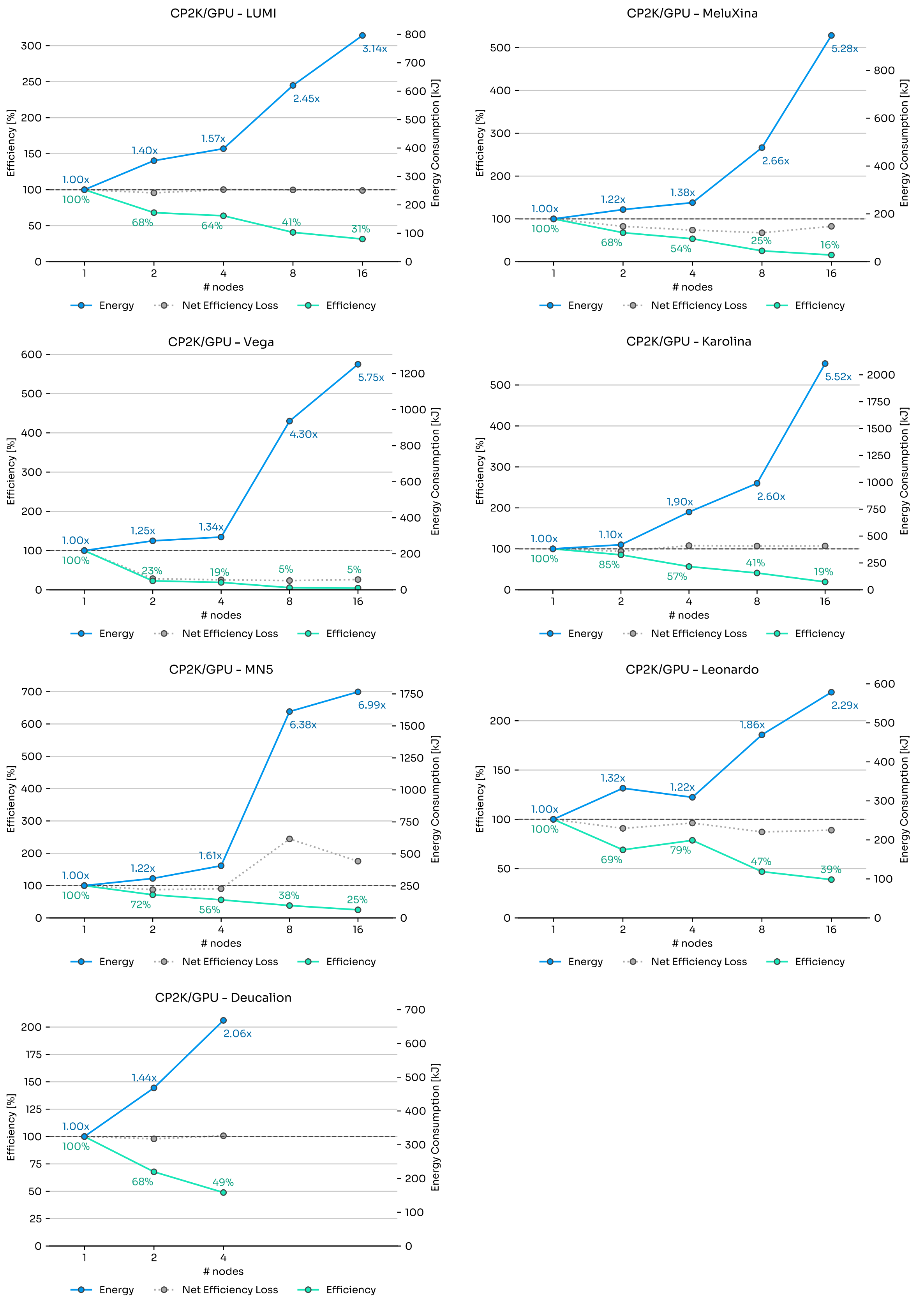


Figure 5-8: Graphs for CP2K GPU.

NAMD

The results for the scaling of NAMD on GPU are shown in Figure 5-9 and Figure 5-10.

For LUMI, MeluXina and Leonardo, the baseline value was taken for two nodes. On Leonardo, the single node configuration could not perform the benchmark. The single node run is less efficient for the two other systems, probably due to the internal automatic parallelisation on NAMD that chooses a less optimal configuration for these systems.

On Vega, there is a clear scaling issue on the GPUs, with efficiency taking a sharp drop on multiple nodes. The net efficiency loss is below the baseline, indicating that the increase in computational resources comes from the reduced efficiency.

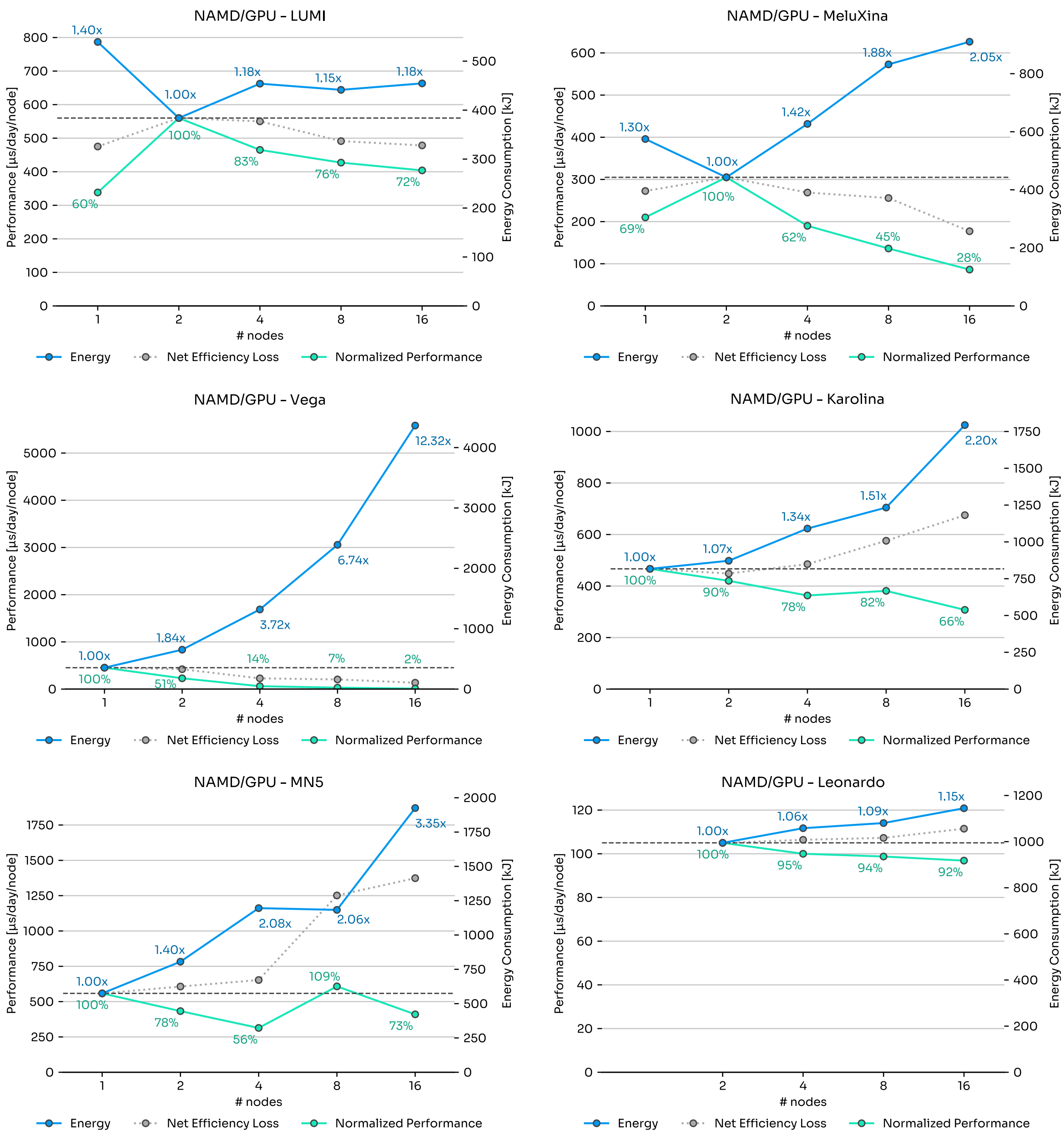


Figure 5-9: Graphs for NAMD GPU for LUMI, MeluXina, Vega, Karolina, MareNostrum 5 and Leonardo.

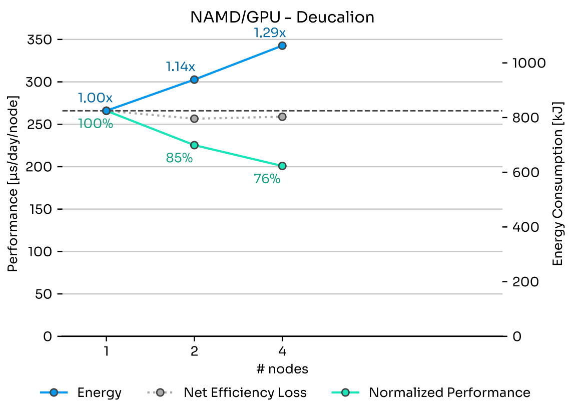


Figure 5-10: Graph for NAMD GPU for Deucalion.

ARM

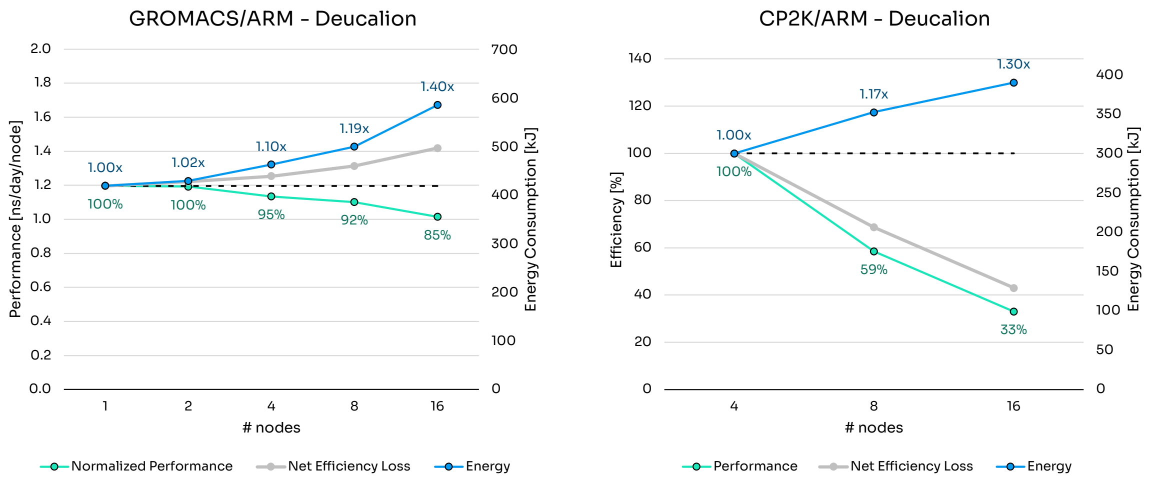


Figure 5-11: Graph for GROMACS ARM (left) and CP2K ARM (right) for Deucalion.

GROMACS

The scaling of GROMACS on ARM for Deucalion has a similar scaling as on GPU, seen in Figure 5-11 (left), but the performance of the baseline is considerably lower for a similar amount of consumed energy for the benchmark.

CP2K

The CP2K on ARM for Deucalion in Figure 5-11 (right) shows a bad scaling of CP2K on ARM. The net efficiency loss is below the baseline, which indicates that the increase in power originates from the less efficient run on multiple nodes.

5.3. Energy usage

This plot shows the energy usage as reported by the different systems, expressed in [kJ]. As the full benchmark is always performed on each run, the values are directly comparable between the system. Variations in the power consumption originate from the different architectures and the methods used to measure the power consumption.

This figure does not give an indication of the total runtime or efficiency. This will be covered in the next section.

CPU

GROMACS

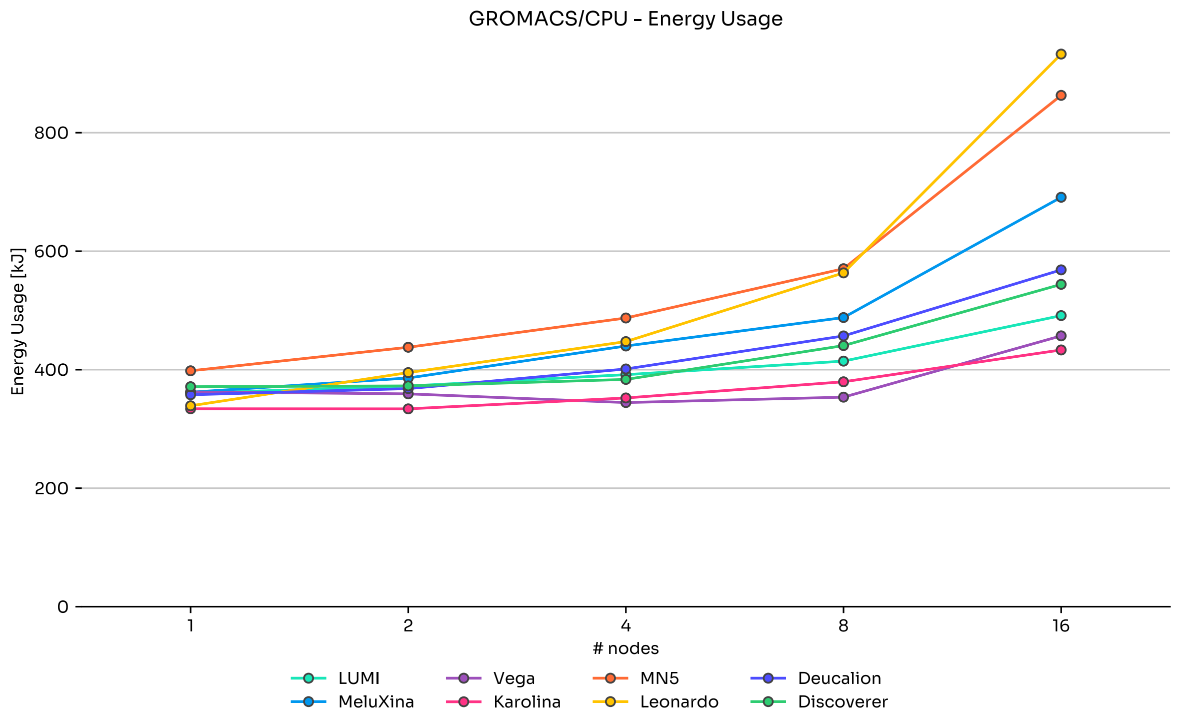


Figure 5-12: Energy usage graph for GROMACS CPU.

In Figure 5-12, the energy consumption for the different systems for GROMACS on CPU is compared.

In general, the different systems follow a similar trend. Vega and Karolina seem to have an overall low energy consumption, whereas Leonardo and MareNostrum 5 have a larger energy increase with the number of nodes.

CP2K

The results for the energy consumption for CP2K on CPU are shown in Figure 5-13.

Again, the systems have similar characteristics, except for Vega which seems to use a lot more energy on the benchmark. MareNostrum 5 now performs much better than the GROMACS-CPU case.

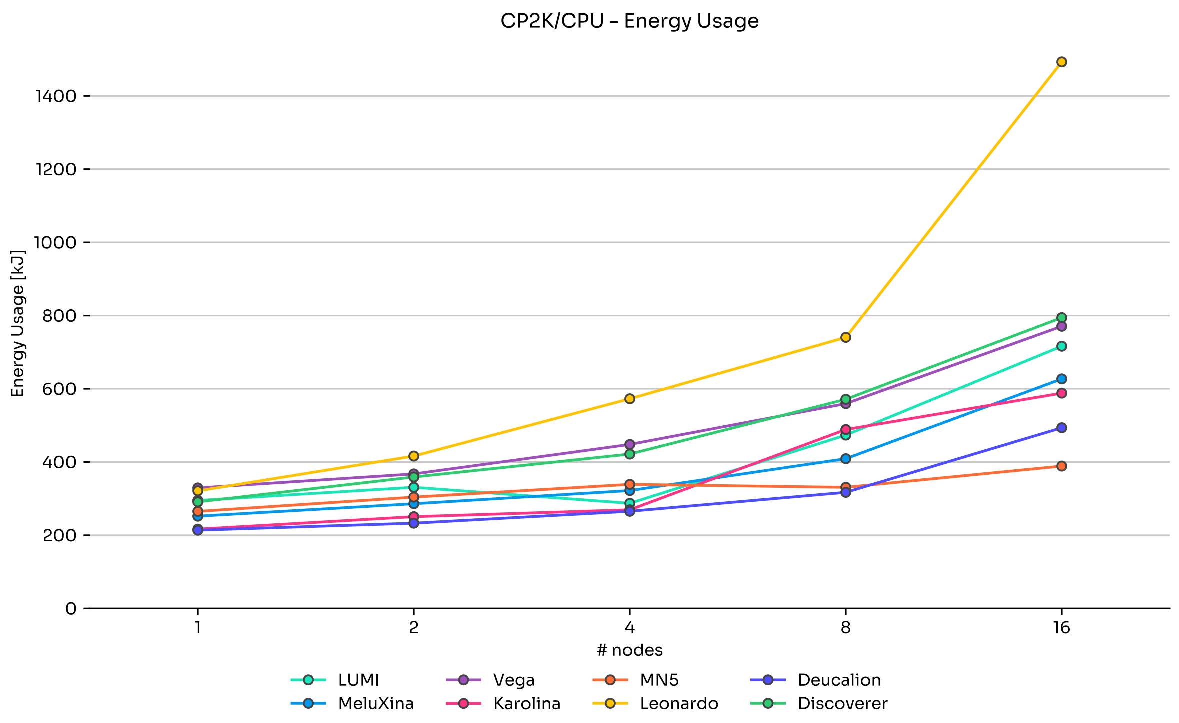


Figure 5-13: Energy usage graph for CP2K CPU.

NAMD

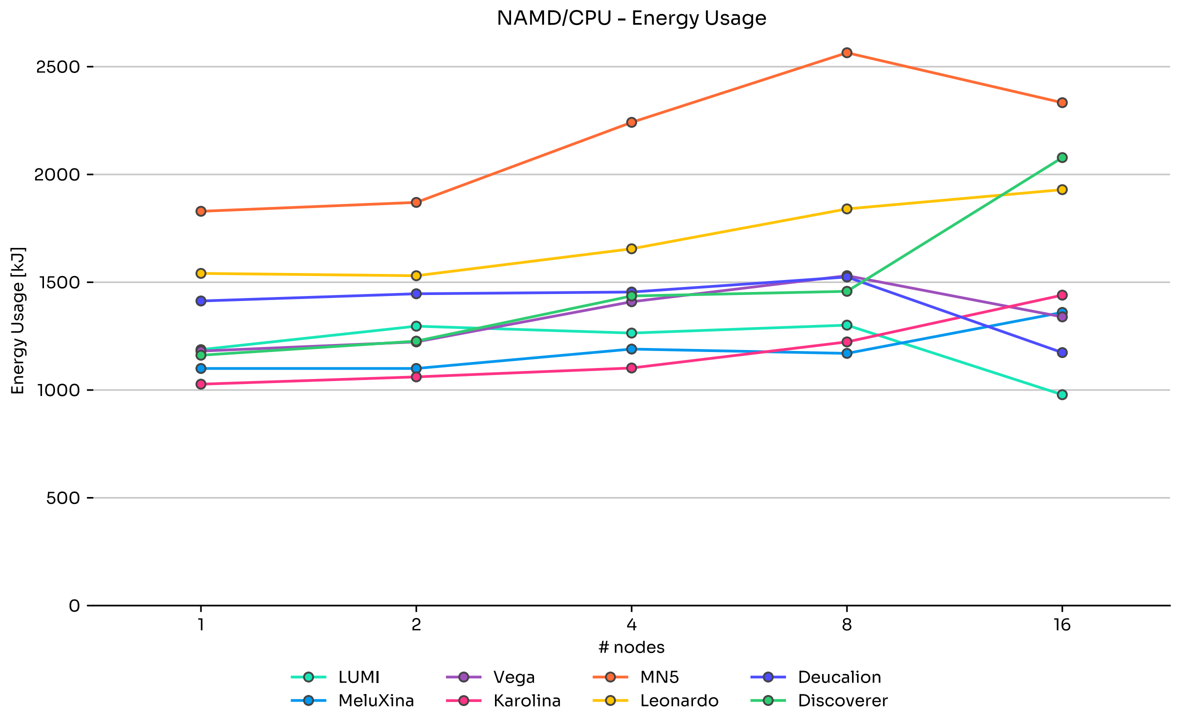


Figure 5-14: Energy usage graph for NAMD CPU.

The results for energy consumption for NAMD-CPU are shown in Figure 5-14. The systems are deviating more than in the other two CPU benchmarks, but the energy increase is less significant. Now, MareNostrum 5 has the highest energy consumption. The most efficient systems seem to be Karolina, MeluXina and LUMI.

GPU

GROMACS

The results for the power consumption of GROMACS on GPU are shown in Figure 5-15. Compared with the CPU results in previous section, the power consumption seems to be much higher with the increasing number of nodes. Leonardo seems to be the exception, with a relatively low power consumption and increase with the number of nodes. Deucalion, Vega, MareNostrum 5 and JEDI all have a relative high power consumption depending on the number of nodes they used.

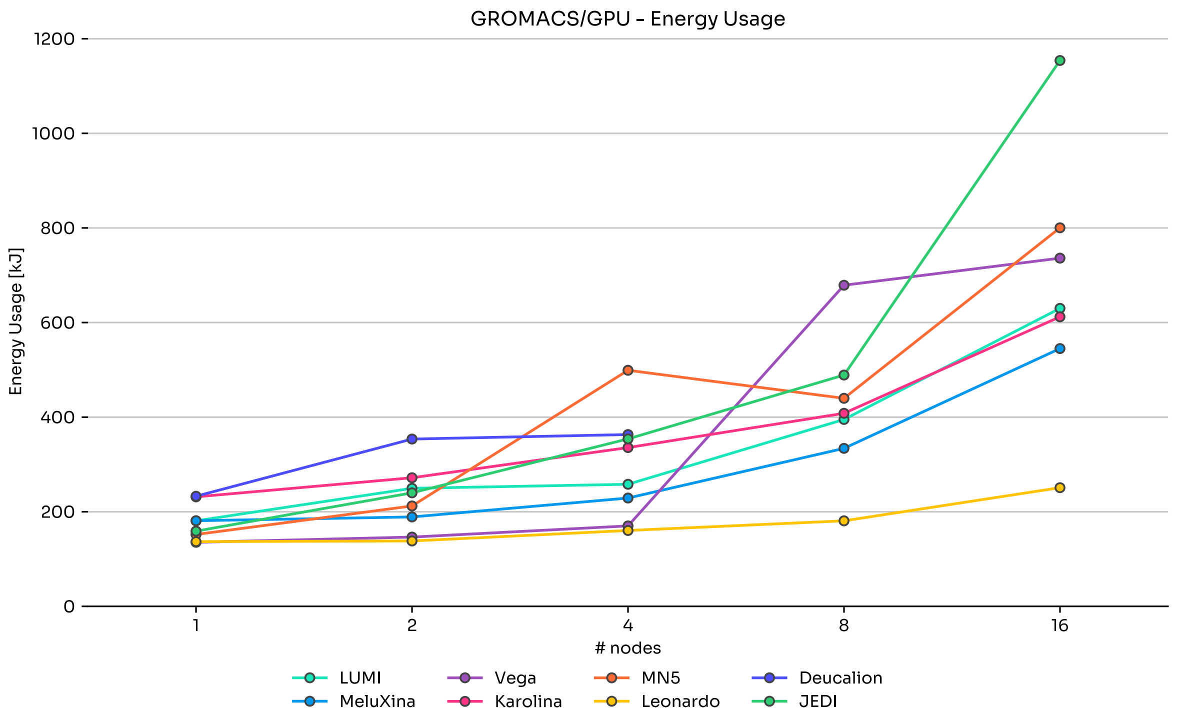


Figure 5-15: Energy usage graph for GROMACS GPU.

CP2K

The results for the power consumption for CP2K on GPU are shown in Figure 5-16. The trends for the power consumption are similar as the GROMACS GPU results, with a wider spread in results.

Again, Leonardo and MeluXina seem to perform efficiently, and Deucalion, Karolina and MareNostrum 5 are consuming the most energy.

For most systems, there seems to be a more pronounced increase in the power consumption at 8 nodes. This is likely due to the less efficient calculation as explained with the *net efficiency loss* from previous section.

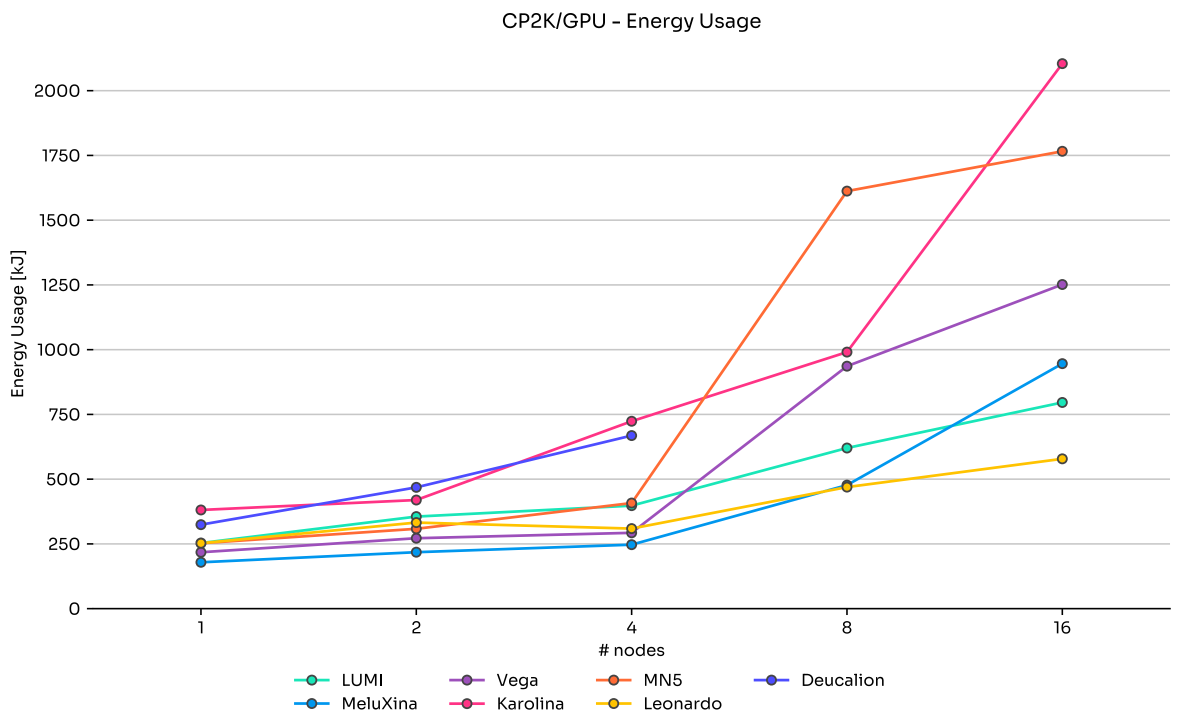


Figure 5-16: Energy usage graph for CP2K GPU.

NAMD

The results for the power consumption of NAMD on GPU are shown in Figure 5-17.

Vega has a large increase in power consumption with the number of nodes. Now, LUMI is the most efficient system, with Leonardo as one of the less efficient systems, followed by Karolina and MareNostrum 5.

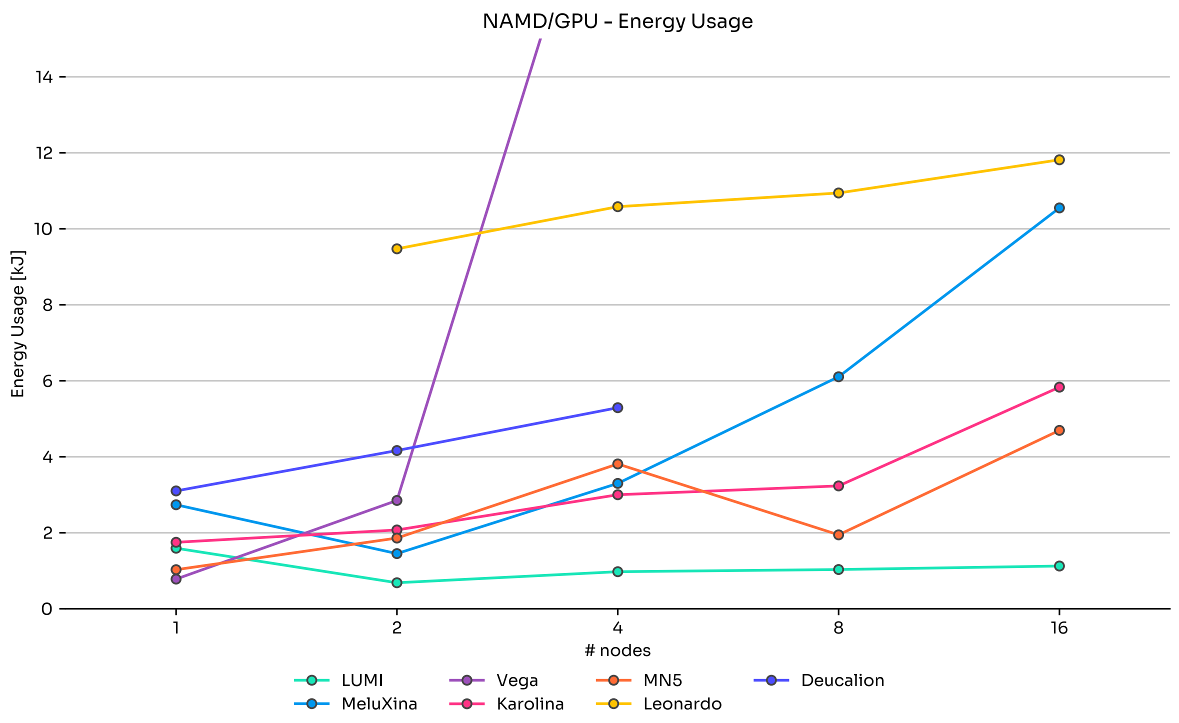


Figure 5-17: Energy usage graph for NAMD GPU.

5.4. Normalized energy usage per ns/day and per 1/h

This plot shows the energy cost to perform a similar computation on a one node equivalent for one computational cycle. It uses both the concept of normalized performance of the performance-energy plot, and the total energy usage. The total energy usage is divided by the normalized performance to obtain the quantity given on this plot, expressed in [kJ/(ns/day/node)], [kJ/(μs/day/node)] or [kJ/(1/h/node)]. It shows an increase of the energy cost by increasing number of nodes, and a general "measure" of the efficiency of the machine. As this value is rescaled with the performance, it also includes the relative speedup between the different machine, but also the relative additional energy consumption for this speedup. This value should be low, as this means a relative low energy usage and a relatively high (normalized) performance. The main difference between this graph and the total energy difference, is that this graph also includes the runtime or performance of the calculation, where the energy usage just reports the total energy consumed over the whole calculation. For CP2K, the energy usage is divided by the inverse of the runtime in hours.

CPU

GROMACS

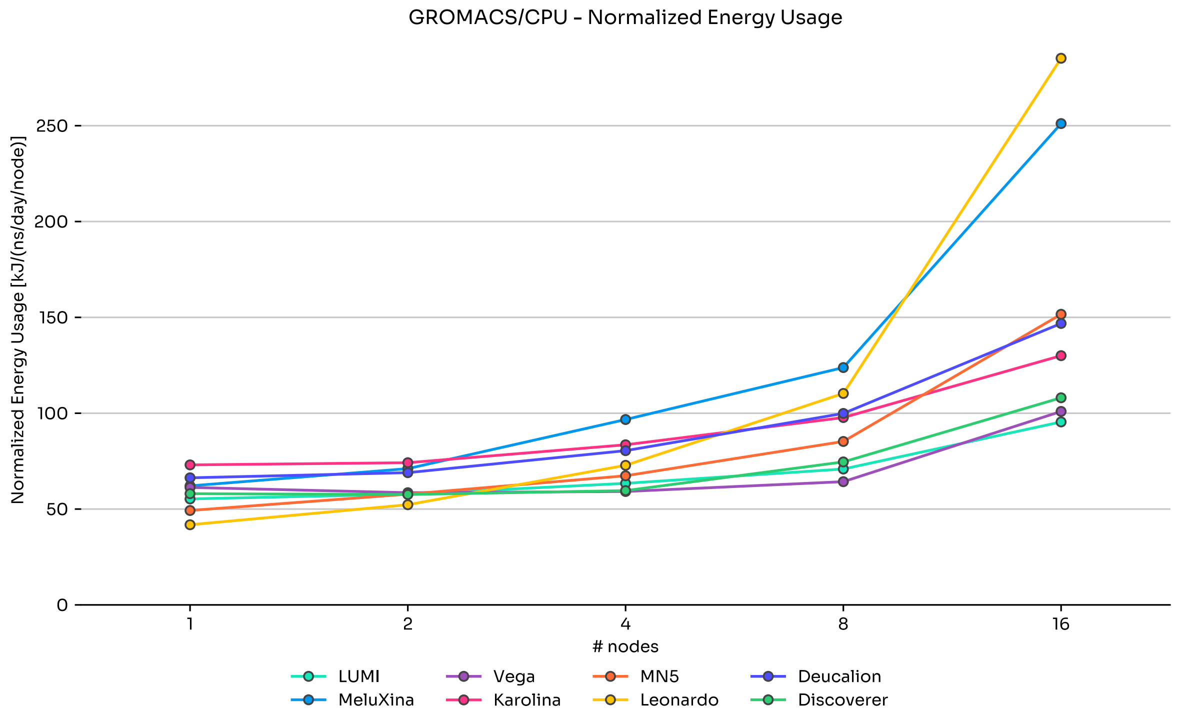


Figure 5-18: Normalized energy usage graph for GROMACS CPU.

The results for the normalized energy usage for GROMACS on CPU are shown in Figure 5-18.

The systems have similar characteristics, with only Leonardo and MeluXina having two outliers from 8 nodes onwards. From the net efficiency loss in Figure 5-1 and

Figure 5-2, which remains close to the baseline, it seems that there are performance issues leading to a higher energy consumption.

Below 8 nodes the normalized energy usage is almost flat for most systems. This means that a higher performance (lower runtime) corresponds to a similar amount of energy increase.

CP2K

The normalized energy usage for CP2K on CPU is shown in Figure 5-19.

For CP2K, we use the runtime in hours as a reference for the performance.

Leonardo again has a large increase from 16 nodes on.

Compared to GROMACS on CPU, the results seem to increase more with the number of nodes used on the system. This means that a faster runtime corresponds to a much higher energy usage than expected. It is much less efficient to run on multiple nodes with a lower runtime than to run slower on a lower number of nodes.

Except for Leonardo and MareNostrum 5, all the machines seem to give similar characteristics. MareNostrum 5 is the most efficient machine according to these results.

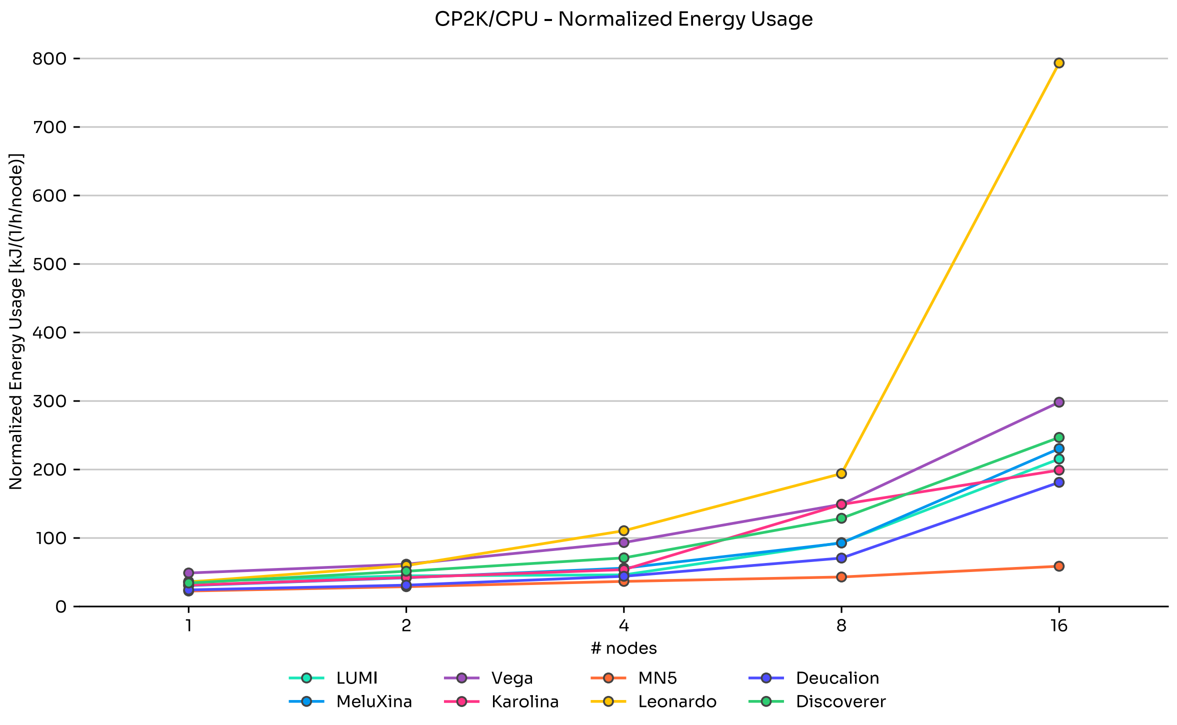


Figure 5-19: Normalized energy usage graph for CP2K CPU.

NAMD

The results for NAMD on CPU for the normalized energy usage are shown in Figure 5-20. There is a larger spread in the results compared to the other two benchmarks on CPU. However, the results are flatter than the other benchmarks, which means that the program will be faster with a similar ratio in the increase in power consumption.

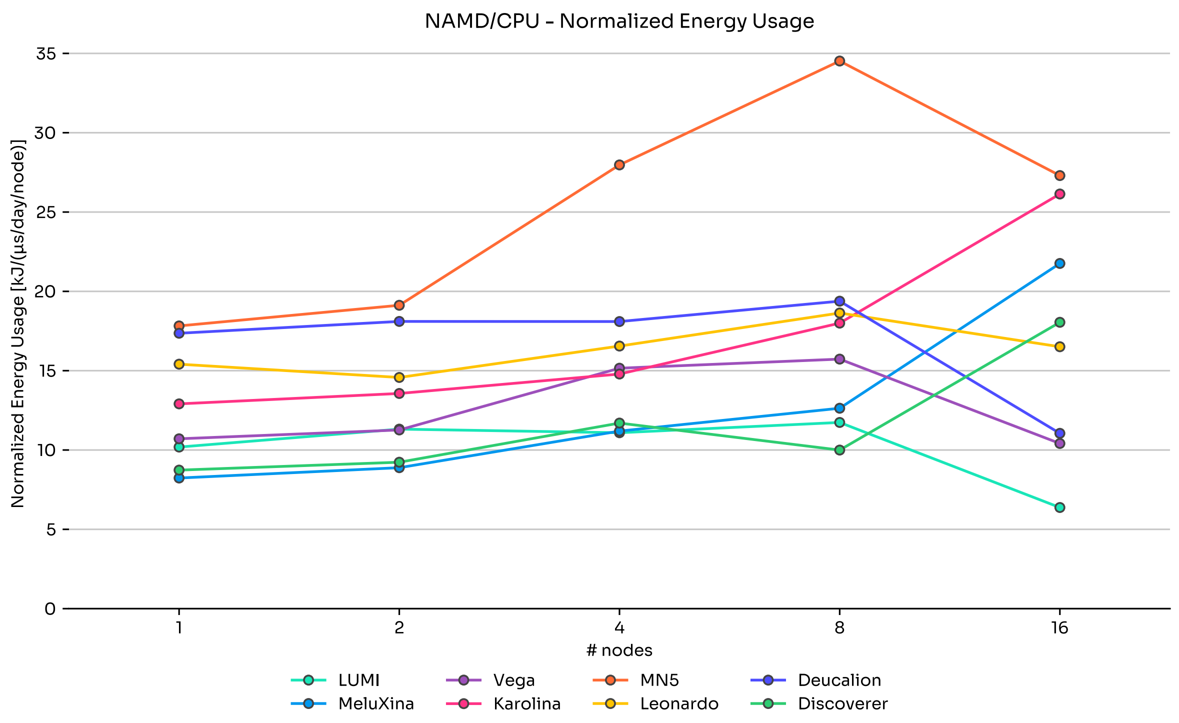


Figure 5-20: Normalized energy usage graph for NAMD CPU.

GPU

GROMACS

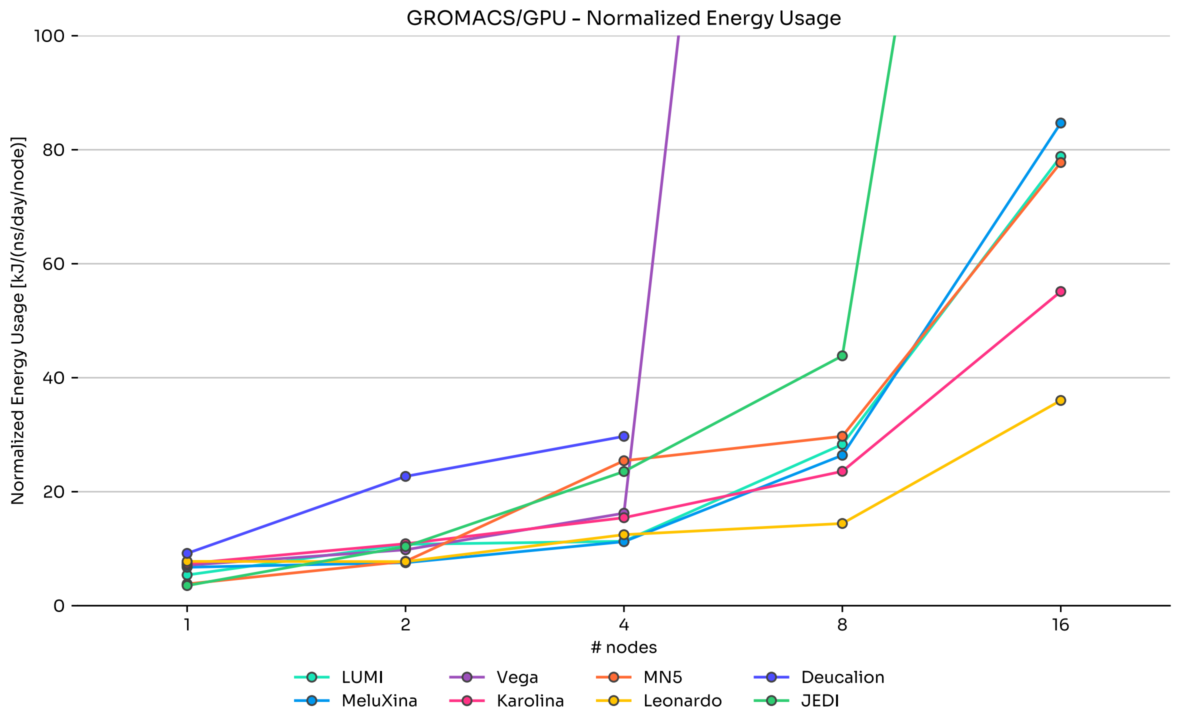


Figure 5-21: Normalized energy usage graph for GROMACS GPU.

For GROMACS on GPU, the results for the normalized energy usage are shown in Figure 5-21. Compared to the GROMACS CPU results, the values are much higher for 16 nodes. This means that the faster runtime on 16 nodes uses much more energy than the efficiency gain provides.

The results for Vega and JEDI are not efficient from 8 nodes onwards. Any reduction in runtime by a larger number of nodes results in a much larger increase in the energy consumption on these systems.

Leonardo has a relatively flat curve, where the increase in efficiency has an equal relative increase in power consumption.

CP2K

The results for the normalized energy usage for CP2K on GPU are shown in Figure 5-22.

Again, Vega has a pronounced energy increase from 8 nodes onwards, followed by MareNostrum on 5 nodes.

Leonardo is the most efficient, with a relatively low line and energy usage.

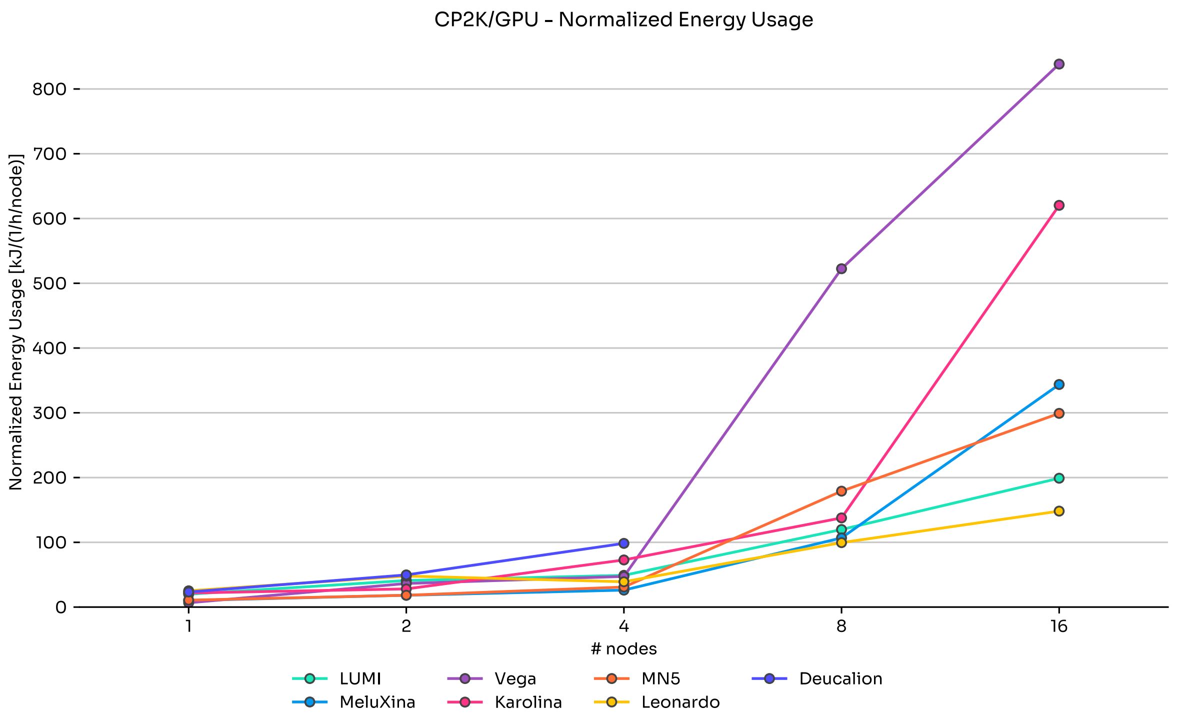


Figure 5-22: Normalized energy usage graph for CP2K GPU.

NAMD

The results for NAMD on GPU are shown in Figure 5-23.

The power consumption for Vega goes up drastically from 2 nodes. It is not efficient to have the relative reduction in runtime for the increase in power consumption for these configurations.

Interestingly, Leonardo now is one of the less efficient systems, requiring more energy to get an increase in performance, but this normalized energy usage still is relatively flat.

LUMI is the most efficient system, with also a relative flat normalized energy usage.

Co-funded by the European Union

EuroHPC Joint Undertaking

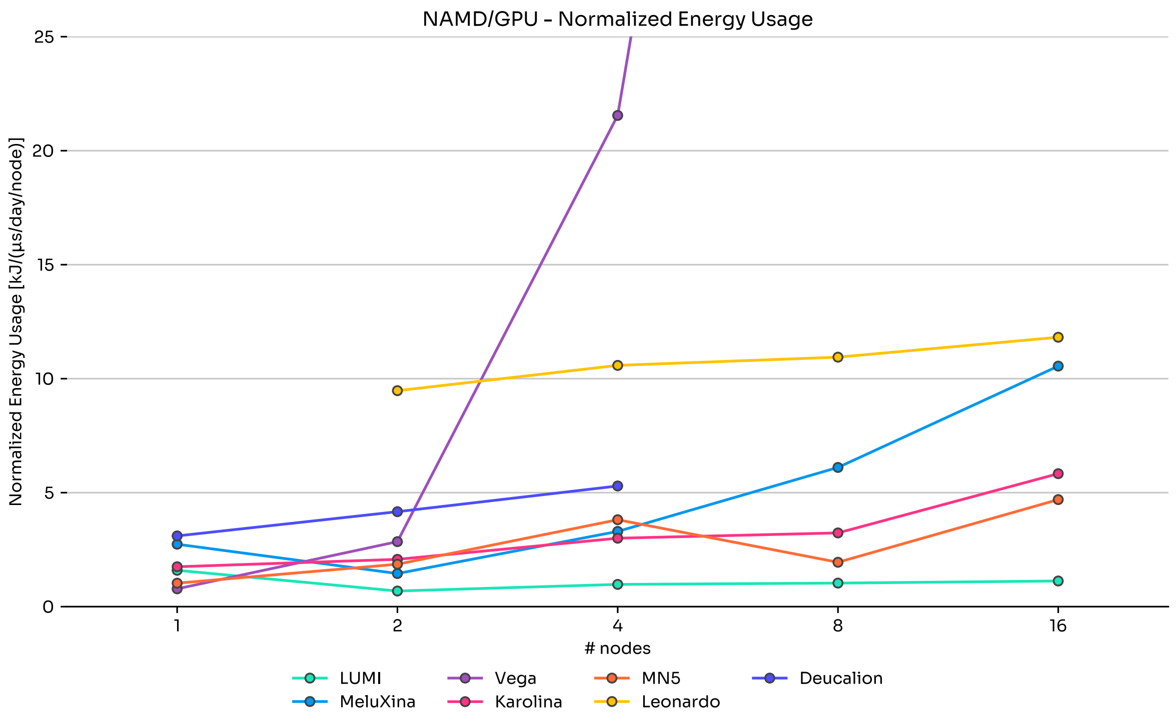


Figure 5-23: Normalized energy usage graph for NAMD GPU.

5.5. Performance and energy heat maps

The heat maps below show the performance and the energy, as was shown in graphs in Section 5.3. Better results have a greener colour, worse results are more red. The best and worst results are in a bold font.

Performance

For GROMACS and NAMD, the reported performance in [ns/day] is used, for CP2K the wall time in [s] is used.

CPU

GROMACS

The performance for GROMACS on CPU is reported in Figure 5-24, higher values (green) indicate a better performance. The best performance for GROMACS CPU was obtained on MareNostrum 5 with a total of 91.08 ns/day using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	6.52	5.83	5.92	**4.57**	8.08	8.11	5.40	6.40
2	12.90	10.86	12.27	9.00	15.16	15.12	10.67	12.94
4	24.70	18.20	23.29	16.87	28.94	24.58	19.94	25.74
8	46.77	31.53	44.01	31.07	53.53	40.86	36.60	47.26
16	82.35	44.03	72.47	53.32	**91.08**	52.33	61.95	80.56

Figure 5-24: Performance reported by GROMACS/CPU in [ns/day].

CP2K

The wall time for CP2K benchmark on CPU is reported in Figure 5-25, lower values (green) indicate a better performance. MareNostrum 5 completed the CP2K CPU benchmark in 34 seconds using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	446	437	535	**518**	307	401	408	425
2	244	264	303	304	172	258	241	258
4	145	157	188	180	97	174	150	152
8	89	102	120	137	59	118	100	101
16	68	83	87	76	**34**	120	83	70

Figure 5-25: Total wall time for CP2K/CPU in [s].

NAMD

The performance for NAMD on CPU is reported in Figure 5-26, higher values (green) indicate a better performance. The best performance for NAMD CPU was obtained on LUMI with a total of 2.45 ns/day using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	0.12	0.13	0.11	**0.08**	0.10	0.10	**0.08**	0.13
2	0.23	0.26	0.22	0.16	0.20	0.21	0.16	0.27
4	0.46	0.45	0.37	0.30	0.32	0.40	0.32	0.49
8	0.89	0.81	0.78	0.54	0.59	0.79	0.63	1.17
16	**2.45**	1.21	2.06	0.88	1.37	1.87	1.70	1.84

Figure 5-26: Performance reported by NAMD/CPU in [ns/day].

GPU

GROMACS

The performance for GROMACS on GPU is reported in Figure 5-27, higher values (green) indicate a better performance. The best performance for GROMACS GPU was obtained on Karolina with a total of 177.66 ns/day using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	JEDI
1	33.56	26.83	19.07	31.30	39.81	**17.60**	25.35	45.02
2	46.35	50.08	29.86	50.02	54.55	35.73	31.18	46.23
4	91.34	81.56	41.92	86.97	78.49	51.50	48.90	60.15
8	111.71	101.24	15.46	138.51	118.35	100.28		89.23
16	127.88	102.97	27.33	**177.66**	164.69	111.47		66.42

Figure 5-27: Performance reported by GROMACS/GPU in [ns/day].

CP2K

The wall time for CP2K on GPU is reported in Figure 5-28, lower values (green) indicate a better performance. MareNostrum 5 completed the CP2K GPU benchmark in 38 seconds using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	283	205	110	206	153	**359**	259
2	208	151	241	120	107	260	191
4	111	95	146	91	68	114	133
8	87	101	251	63	50	95	
16	56	82	151	66	**38**	58	

Figure 5-28: Total wall time for CP2K/GPU in [s].

NAMD

The performance for NAMD on GPU is reported in Figure 5-29, higher values (green) indicate a better performance. The best performance for NAMD GPU was obtained on MareNostrum 5 with a total of 6.56 ns/day using 16 nodes, closely followed by LUMI with a total of 6.46 ns/day using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	0.34	**0.21**	0.45	0.47	0.56		0.27
2	1.12	0.61	0.46	0.84	0.87	0.21	0.45
4	1.86	0.76	0.24	1.45	1.25	0.40	0.80
8	3.42	1.09	0.24	3.05	4.86	0.79	
16	6.46	1.38	0.18	4.92	**6.56**	1.55	

Figure 5-29: Performance reported by NAMD/GPU in [ns/day].

Energy usage

For the benchmarks, the reported consumed energy is given in units of [kJ].

CPU

GROMACS

The consumed energy for GROMACS on CPU is reported in Figure 5-30, lower values (green) indicate a better energy usage. The lowest energy was consumed by MeluXina using 4 nodes with 326.97 kJ, closely followed by Karolina and Leonardo using 1 node with respectively 334.05 kJ and 339.05 kJ.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	360.16	362.00	362.47	**334.05**	398.12	339.05	357.69	371.30
2	372.47	386.00	359.22	333.90	437.86	395.10	368.07	372.64
4	391.61	440.00	344.51	352.31	487.24	447.42	401.19	383.56
8	414.44	488.00	353.64	379.58	570.45	563.39	456.98	440.65
16	491.29	691.00	457.11	433.31	863.00	**932.68**	568.48	544.05

Figure 5-30: Energy usage for GROMACS/CPU in [kJ].

CP2K

The consumed energy for CP2K on CPU is reported in Figure 5-31, lower values (green) indicate a better energy usage. MareNostrum 5 completed the CP2K CPU benchmark in 34 seconds using 16 nodes. The lowest energy was consumed by Deucalion using 1 node with 213.75 kJ, closely followed by Karolina with 216.57 kJ using 1 node.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	294.92	252.00	329.14	216.57	265.07	321.14	**213.75**	291.05
2	330.93	286.00	367.40	250.65	304.10	416.23	232.90	359.03
4	287.19	322.00	447.95	269.45	338.92	572.71	265.42	421.65
8	473.51	409.00	559.47	488.66	330.81	740.46	317.17	571.28
16	716.28	627.00	770.77	588.12	388.89	**1492.97**	493.50	794.34

Figure 5-31: Energy usage for CP2K/CPU in [kJ].

NAMD

The consumed energy for NAMD on CPU is reported in Figure 5-32, lower values (green) indicate a better energy usage. The lowest energy was consumed by LUMI as well using the same 16 nodes with 978.26 kJ.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	Discoverer
1	1187.32	1100.00	1180.70	1027.15	1829.13	1541.03	1413.09	1161.45
2	1296.20	1100.00	1223.24	1060.93	1870.39	1530.21	1446.58	1226.76
4	1264.26	1190.00	1409.53	1102.18	2241.73	1655.20	1454.59	1436.22
8	1301.04	1170.00	1530.47	1223.17	**2564.66**	1840.27	1524.36	1457.86
16	**978.26**	1360.00	1339.35	1440.17	2333.07	1929.51	1173.66	2078.29

Figure 5-32: Energy usage for NAMD/CPU in [kJ].

Co-funded by the European Union

GPU

GROMACS

The consumed energy for GROMACS on GPU is reported in Figure 5-33, lower values (green) indicate a better energy usage. The lowest energy was consumed by Vega using 1 node with 135.42 kJ.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion	JEDI
1	180.83	181.00	135.42	231.48	151.94	136.81	232.79	159.00
2	249.57	189.00	146.46	271.86	212.24	138.23	353.70	240.00
4	258.09	229.00	170.06	335.65	499.10	160.39	363.21	354.00
8	395.12	334.00	678.87	408.16	439.82	180.81		489.00
16	630.01	545.00	736.15	612.11	800.31	250.89		1154.00

Figure 5-33: Energy usage for GROMACS/GPU in [kJ].

CP2K

The consumed energy for CP2K on GPU is reported in Figure 5-34, lower values (green) indicate a better energy usage. The lowest energy was consumed by MeluXina using 1 node with 179 kJ.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	253.35	179.00	217.76	381.05	252.54	252.71	324.28
2	355.49	218.00	271.94	419.43	308.31	332.41	468.38
4	397.70	247.00	292.82	723.56	407.75	309.14	668.31
8	620.25	477.00	936.23	990.64	1612.20	469.34	
16	796.12	946.00	1251.58	2104.44	1765.96	578.52	

Figure 5-34: Energy usage for CP2K/GPU in [kJ].

NAMD

The consumed energy for NAMD on GPU is reported in Figure 5-35, lower values (green) indicate a better energy usage. The lowest energy was consumed by VEGA using 1 node with 354.38 kJ

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	539.28	575.00	354.38	816.90	574.61		824.83
2	383.83	443.00	652.61	871.85	805.60	994.75	938.86
4	454.08	627.00	1319.18	1090.58	1195.85	1058.25	1063.01
8	441.45	832.00	2389.53	1233.31	1183.12	1080.64	
16	454.64	910.00	4365.54	1793.60	1925.27	1144.74	

Figure 5-35: Energy usage for NAMD/GPU in [kJ].

Energy advantage GPU/CPU

For the benchmarks, the reported consumed energy for the CPU is compared relative to the GPU, the division of the consumed energy of the CPU by the GPU is given.

CPU

GROMACS

The consumed energy for GROMACS on CPU relative to the GPU is reported in Figure 5-36, higher values (green) indicate a better energy usage of the GPU. The lowest relative energy was consumed by Leonardo using 16 nodes with 3.72 times increased efficiency on the GPU relative to the CPU.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	1.99	2.00	2.68	1.44	2.62	2.48	1.54
2	1.49	2.04	2.45	1.23	2.06	2.86	1.04
4	1.52	1.92	2.03	1.05	0.98	2.79	1.10
8	1.05	1.46	**0.52**	0.93	1.30	3.12	
16	0.78	1.27	0.62	0.71	1.08	**3.72**	

Figure 5-36: Relative energy usage for GROMACS CPU/GPU.

CP2K

The consumed energy for CP2K on CPU relative to the GPU is reported in Figure 5-37, higher values (green) indicate a better energy usage of the GPU. Leonardo completed the CP2K benchmark with a 2.58 better efficiency on the GPU using 16 nodes.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	1.16	1.41	1.51	0.57	1.05	1.27	0.66
2	0.93	1.31	1.35	0.60	0.99	1.25	0.50
4	0.72	1.30	1.53	0.37	0.83	1.85	0.40
8	0.76	0.86	0.60	**0.49**	**0.21**	1.58	
16	0.90	0.66	0.62	0.28	0.22	**2.58**	

Figure 5-37: Relative energy usage for CP2K CPU/GPU.

NAMD

The consumed energy for NAMD on CPU relative to the GPU is reported in Figure 5-38, higher values (green) indicate a better energy usage of the GPU. The best relative energy usage was performed by MareNostrum 5 on 16 nodes with a 4.21 times better energy usage of the GPU relative to the CPU.

	LUMI	MeluXina	Vega	Karolina	MN5	Leonardo	Deucalion
1	2.20	1.91	3.33	1.26	3.58		1.71
2	3.38	2.48	1.87	1.22	1.04	1.54	1.54
4	2.78	1.90	1.07	1.01	1.09	1.56	1.37
8	2.95	1.41	0.64	0.99	4.21	1.70	
16	2.15	1.49	0.31	0.80	2.23	1.69	

Figure 5-38: Relative energy usage for NAMD CPU/GPU.

6. Conclusion

Chapter 2 describes the benchmarks being used to give an overview of energy measurements on the different machines. Chapter 3 presents the available EuroHPC machines, including specifications, measurement tools and other available libraries. Chapter 4 discusses tools which provide extra information outside of the default data gathered by Slurm, useful for collecting and/or influencing energy usage, together with an overview of dashboards available on some sites. Chapter 5 contains the results of running the benchmarks using GROMACS, CP2K and NAMD, on CPU and GPU, providing both performance and energy usage data.

It might be tempting to pick to the most “green-ish” machine from the heat maps for your next computations. However, we suggest to not blindly follow the tables and take the following remarks into account.

- Different versions of the same program might have been used, or the same version with different compilation options.
- The placement of the jobs by the scheduler might be different.
- The pinning might be different.
- Energy measurements might be different: output directly from sensors, or via specific libraries; sampling rate; …
- Even if the hardware is very similar (MeluXina, Vega and Discoverer), results might be different.
- The number of CPU cores or GPUs might be different.
- Make sure to use full nodes (`--exclusive`) when comparing machines.