MAchinE Learning for Scalable meTeoROlogy and cliMate

Final Hardware Performance Benchmarking

FZJ

www.maelstrom-eurohpc.eu

D 3.7 Final Report on Hardware Performance Benchmarking for ML Solutions with the Full Implementation of the Workflow Tools

Author(s):	FZJ
	Stepan Nassyr (FZJ), Andreas Herten (FZJ), Mattia Paladino (E4)
Dissemination Level:	Public
Date:	March 7, 2024
Version:	1.0
Contractual Delivery Date:	01/2024
Work Package/ Task:	WP3/ T3.3
Document Owner:	FZJ
Contributors:	FZJ, E4; All
Status:	Final

MAELSTROM

Machine Learning for Scalable Meteorology and Climate

Research and Innovation Action (RIA)
H2020-JTI-EuroHPC-2019-1: Towards Extreme Scale Technologies and Applications

Project Coordinator:	Dr. Peter Dueben (ECMWF)
Project Start Date:	01/04/2021
Project Duration:	36 months

Published by the MAELSTROM Consortium

Contact:
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom
Peter.Dueben@ecmwf.int

The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Unions Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway.

Contents

1 Executive Summary **10**

2 Introduction **11**
- 2.1 About MAELSTROM . . . 11
- 2.2 Scope of this deliverable . . . 11
  - 2.2.1 Objectives of this deliverable . . . 11
  - 2.2.2 Work performed in this deliverable . . . 12
  - 2.2.3 Computing configuration and Storage . . . 12
  - 2.2.4 Deviations and counter measures . . . 13

3 Metrics **14**

4 Benchmarks **16**
- 4.1 ALL Comparative Analysis . . . 18
  - 4.1.1 Device Parallelism . . . 18
  - 4.1.2 GPU Scaling . . . 19
  - 4.1.3 Energy-to-Solution . . . 20
  - 4.1.4 Time-to-Solution . . . 21
  - 4.1.5 Node energy consumption . . . 22
- 4.2 AP1 . . . 24
  - 4.2.1 Notes . . . 24
  - 4.2.2 Runtime split . . . 26
  - 4.2.3 Non-IO benchmarks . . . 29
- 4.3 AP2 . . . 34
  - 4.3.1 Notes . . . 34
  - 4.3.2 Runtime split . . . 37
- 4.4 AP3 . . . 42
  - 4.4.1 Notes . . . 42
  - 4.4.2 JURECA-DC A100 . . . 44
  - 4.4.3 JURECA-DC H100 . . . 47
  - 4.4.4 JURECA-DC MI250 . . . 50
  - 4.4.5 E4 A2 . . . 55
- 4.5 Non-IO Experiments . . . 58
- 4.6 Node energy measurements . . . 61
- 4.7 AP4 . . . 64
  - 4.7.1 Notes . . . 64

4.7.2 Runtime split . . . 65
4.8 AP5 . . . 69
4.8.1 Notes . . . 69
4.8.2 Non-IO benchmarks . . . 72
4.8.3 Runtime split . . . 76
4.9 AP6 . . . 80
4.9.1 Notes . . . 80
4.9.2 JURECA-DC A100 . . . 81
4.9.3 JURECA-DC H100 . . . 81
4.9.4 JURECA-DC MI250 . . . 82
4.9.5 E4 A2 . . . 82
4.9.6 E4 GH200 . . . 82
4.9.7 Runtime split . . . 83
4.9.8 Node Energy Consumption . . . 83
4.9.9 Conclusion . . . 86
4.10 AtmoRep . . . 89

5 Conclusion **90**

6 Appendix **92**
6.1 AP1 . . . 93
6.2 AP2 . . . 99
6.3 AP3 . . . 102
6.4 AP4 . . . 118
6.5 AP5 . . . 120
6.6 AP6 . . . 125

List of Figures

1 ALL Device Parallelism . . . 18
2 ALL Scaling . . . 19
3 ALL Energy . . . 19
4 ALL Energy Usage . . . 20
5 ALL Duration . . . 22
6 ALL Duration (no A2) . . . 22
7 ALL Node and GPU Energy . . . 23
8 AP1 JURECA-DC MI250 . . . 26
9 AP1 JURECA-DC A100 . . . 26
10 AP1 JUWELS Booster . . . 27

11 AP1 JURECA-DC H100 27
12 AP1 E4 GH200 28
13 AP1 E4 A2 28
14 AP1 Energy 29
15 AP1 Energy 30
16 AP1 Time 31
17 AP1 Time 31
18 AP1 Node and GPU Energy comparison 32
19 AP2 JURECA-DC MI250 37
20 AP2 JURECA-DC A100 38
21 AP2 JUWELS Booster 38
22 AP2 JURECA-DC H100 39
23 AP2 E4 GH200 39
24 AP2 E4 A2 40
25 AP2 Node and GPU Energy comparison 41
26 AP3 JURECA-DC A100 Runtime 44
27 AP3 JURECA-DC A100 Epoch Time 45
28 AP3 JUWELS Booster Energy 46
29 AP3 JUWELS Booster Inference Runtime 46
30 AP3 JURECA-DC H100 Runtime 47
31 AP3 JURECA-DC H100 Epoch Time 48
32 AP3 JURECA-DC H100 Energy 48
34 AP3 JURECA-DC MI250 Runtime 50
35 AP3 JURECA-DC MI250 Runtime 50
36 AP3 JURECA-DC MI250 Epoch Time 51
37 AP3 JURECA-DC MI250 Epoch Time 52
38 AP3 JURECA-DC MI250 Energy 52
39 AP3 JURECA-DC MI250 Energy 53
40 AP3 JURECA-DC MI250 Inference Runtime 53
41 AP3 JURECA-DC MI250 Inference Runtime 54
42 AP3 E4 A2 Runtime 55
43 AP3 E4 A2 Epoch Time 56
44 AP3 E4 A2 Energy 56
45 AP3 E4 A2 Inference Runtime 57
46 AP3 Non-IO Throughput 58
47 AP5 Energy 59
48 AP5 Energy 59

49 AP5 Time 60
50 AP5 Time 60
51 AP3 Node and GPU Energy comparison during the training phase . . . 62
52 AP3 Node and GPU Energy comparison during the inference phase . . 63
53 AP3 Node and GPU Energy comparison for Non-IO experiments 63
54 AP4 JURECA-DC MI250 65
55 AP4 JURECA-DC A100 65
56 AP4 JURECA-DC H100 66
57 AP4 E4 GH200 66
58 AP4 E4 A2 67
59 AP4 Node and GPU Energy comparison 68
60 AP5 Energy 72
61 AP5 Energy 73
62 AP5 Time 73
63 AP5 Time 74
64 AP5 Time 74
65 AP5 Node and GPU Energy for Non-IO Benchmarks 76
66 AP5 JURECA-DC MI250 77
67 AP5 JURECA-DC A100 77
68 AP5 JURECA-DC H100 78
69 AP5 E4 GH200 78
70 AP5 E4 A2 79
71 AP6 JURECA-DC MI250 83
72 AP6 JURECA-DC A100 84
73 AP6 JURECA-DC H100 84
74 AP6 E4 GH200 85
75 AP6 E4 A2 85
76 AP6 Node/GPU Energy 87
77 AP6 Action 88
78 AtmoRep JUWELS Booster Scaling 90

List of Tables

1 AP1 Action metric for A2 and GH200 systems. 33
2 AP2 Action metric for A2 and GH200 systems. 40
3 Summary of hardware tested for Application 3. 42
4 AP4 Action metric for A2 and GH200 systems. 68
5 AP5 Action metric for A2 and GH200 systems. 75

6 AP1 JUWELS Booster NVIDIA A100 training benchmark 93
7 AP1 JURECA NVIDIA A100 training benchmark 94
8 AP1 JURECA NVIDIA H100 training benchmark. 95
9 AP1 JURECA AMD MI250 training benchmark 96
10 AP1 E4 NVIDIA A2 training benchmark 97
11 AP1 E4 NVIDIA GraceHopper H200 training benchmark. 98
12 AP2 Jureca A100 benchmarks 99
13 AP2 Jureca H100 benchmarks 99
14 AP2 Jureca Mi250 benchmarks 100
15 AP2 E4 Grace Hopper benchmarks 100
16 AP2 E4 A2 101
17 AP3 JURECA-DC A100 training benchmark A100 103
18 AP3 JURECA-DC H100 training benchmark H100 104
19 AP3 JURECA-DC MI250 training benchmark Mi250 - 8 GPUS 105
20 AP3 JURECA-DC MI250 training benchmark Mi250 - 1 GPUS 106
21 AP3 E4 A2 training benchmark E4 - A2 107
22 AP3 JURECA-DC A100 inference benchmark A100 108
23 AP3 JURECA-DC H100 inference benchmark H100 109
24 AP3 JURECA-DC MI250 inference benchmark Mi250 - 1 GPU 110
25 AP3 JURECA-DC MI250 inference benchmark Mi250 - 8 GPUs 111
26 AP3 E4 A2 inference benchmark E4 -A2 112
27 AP3 Non-io Experiments - A100 113
28 AP3 Non-io Experiments - Grace Hopper 114
29 AP3 Non-io Experiments - MI250 - 8 GPUS 115
30 AP3 Non-io Experiments - H100 116
31 AP3 Non-io Experiments - GRAPGHCORE IPU 117
32 AP4 JUWELS Booster training benchmark 118
33 AP4 JUWELS Cluster training benchmark 119
34 AP4 E4 A2 training benchmark 119
35 AP1 JURECA NVIDIA A100 training benchmark 120
36 AP5 JURECA NVIDIA H100 training benchmark 121
37 AP5 JURECA AMD MI250x training benchmark 121
38 AP5 E4 NVIDIA GraceHopper H200 training benchmark 122
39 AP5 E4 NVIDIA A2 training benchmark 123
40 AP5 E4 A2 inference runtime. 124
41 AP5 E4 AMD System inference benchmark 124
42 AP6 JUWELS Booster training benchmark 126

43 AP6 JUWELS Cluster training benchmark 126
44 AP6 E4 A2 training benchmark 127

1 Executive Summary

A final benchmarking phase of the MAELSTROM applications was conducted using the knowledge gained from the previous benchmarks described in D3.4 and D3.6. The number of hardware configurations has been expanded again, including 5 GPUs (NVIDIA A2, A100, H100, and GH200; AMD MI250) and the Graphcore GC200 IPU.
In this benchmarking phase, an emphasis was put on evaluating comparative results and energy consumption. Additionally, benchmarks omitting data loading (*non-io*) were performed to decouple the measurement of compute performance from the filesystem employed by the host system, which was a major contributor of uncertainty in the previous deliverables.
Because the majority of applications has started employing multi-device parallelism at the time of this benchmarking phase, more in-depth analysis of the GPU/IPU scaling has been performed for both Energy-to-Solution and Time-to-Solution.
On a subset of the platforms, full-node energy consumption was measured for the benchmarks and the results were contrasted with the GPU-only energy measurements, leading to valuable results which will be used for the bespoke system design, the next WP3 deliverable.

2 Introduction

2.1 About MAELSTROM

MAELSTROM aims to create Europe's next-generation computer architecture by co-designing custom compute system designs for optimal application performance and energy efficiency, along with a software framework to improve usability and training efficiency for large-scale machine learning applications in weather and climate science.

To achieve this, MAELSTROM will benchmark these applications across various computing systems based on energy consumption, time-to-solution, numerical precision, and solution accuracy. Customised compute systems will be designed that are optimised for application needs in order to enhance Europes high-performance computing portfolio and to pull recent hardware developments towards the unique requirements of weather and climate applications. The MAELSTROM software framework will enable scientists to apply and compare machine learning tools and libraries across a wide range of computer systems with ease. This will be supported by a user interface that links application developers with compute system designers. Also, during the development phase, automated benchmarking and error detection of machine learning solutions will be conducted. These tools will be published as open source.

The MAELSTROM machine learning applications will cover all the key components involved in the workflow of weather and climate predictions. This includes processing of observations, assimilation of observations to generate initial and reference conditions, model simulations, as well as post-processing of model data and development of forecast products. For each application, benchmark datasets with up to 10 terabytes of data will be available online for training and machine learning tool-development on the fastest supercomputers in the world. The machine learning solutions developed by MAELSTROM will serve as a blueprint for future machine learning applications on supercomputers.

2.2 Scope of this deliverable

2.2.1 Objectives of this deliverable

Deliverable 3.7 is a report on the work done for Task 3.3, as final benchmarking of ML solutions depicted in D1.3 on a wider range of hardware and monitoring tools

to investigate new configuration compared to D3.4 and D3.6. Alongside the new monitoring tools used in D3.6, new GPU-specific power-measurement scripts have been utilized.
Deliverable 3.7 is the third and final MAELSTROM deliverable that provides information on the benchmarks of the applications on HPC hardware. It measures a large variety of metrics and plots related to application executions on heterogeneous HPC systems to allow a detailed performance evaluation.

2.2.2 Work performed in this deliverable

The performance evaluation metrics were agreed upon with the WP1 application developers and are the same as those used in the previous benchmarking phases in D3.4 and D3.6 with the addition of some new metrics related to the individual GPU energy consumption of each node. A spreadsheet was provided to the application developers to enter their benchmark results on the available HPC machines.
In the second benchmarking phase, application developers were granted access to resources at JSC and E4. Information on system access, benchmark runs, and metric measurement was compiled and documented on the project's Confluence page, where it was accessible to all project members.
The application developers ran the benchmarks, and the results were recorded in the spreadsheet.
The results were analyzed to examine performance, scalability behavior, energy efficiency, and potential issues. In this phase, we used multiple evaluation platforms with different configurations to ensure a thorough analysis of the applications' performance.

2.2.3 Computing configuration and Storage

The computational systems used in this benchmarking phase are

- JSC
  - JURECA-DC GPU (NVIDIA A100)[1]
  - JURECA-DC Evaluation Platform 1 (AMD MI250)[2]
  - JURECA-DC Evaluation Platform 2 (NVIDIA H100)[3]

[1] `https://apps.fz-juelich.de/jsc/hps/jureca/configuration.html#hardware-configuration-of-the-system-name-dc-module-phase-2-as-of-may-2021`
[2] `https://apps.fz-juelich.de/jsc/hps/jureca/evaluation-platform-overview.html#mi200-nodes`
[3] `https://apps.fz-juelich.de/jsc/hps/jureca/evaluation-platform-overview.html#h100-node`

  - JURECA-DC Evaluation Platform 3 (Graphcore IPU)[4]
- E4
  - NVIDIA Grace-Hopper Superchip (GH200)
  - NVIDIA A2

Note: JURECA-DC is sometimes abbreviated to *JRDC*.

In the following, a brief description of the Hardware systems provided by E4 is provided.

- 2× NVIDIA Grace-Hopper nodes, each one with:
  - 1× NVIDIA Grace 72-Core
  - 1× LPDDR5X 480GB RAM
  - Mellanox CX7 NDR Dual Port interconnection
  - 1× NVIDIA GH200 GPU
- 2× NVIDIA A2 nodes, each one with:
  - 2× NVIDIA Xeon(R) Gold 6426Y CPU
  - 32× DDR5-4800 32GB
  - Mellanox CX6 HDR Single Port interconnection
  - 2× NVIDIA A2 Tensor Core GPU

2.2.4 Deviations and counter measures

The deliverable was delayed by 4 weeks to wait for the availability of Grace Hopper GPUs, as this hardware is promising significant speed-ups when compared to the A100 generation of NVIDIA GPUs and as the integrated CPU/GPU memory is of particular interest for Earth system applications.

[4] `https://apps.fz-juelich.de/jsc/hps/jureca/evaluation-platform-overview.html#graphcore-ipu-pod4`

3 Metrics

The metrics selected for performance evaluation fall into four categories: time-related metrics, model-related metrics, energy-related metrics and general score metric. The following metrics have been measured and documented for all of the applications:

- Time-related
  - Total runtime
  - Total training time
  - Loading Data Time
  - Min. training time per epoch
  - Max. training time per epoch
  - Avg. training time per epoch
  - First epoch training time
  - Avg. training time per iteration
  - Saving model time
- Model-related
  - Final training loss
  - Final validation loss
- Energy-related
  - Max. GPU power
  - GPU energy consumption
  - Total node energy consumption
- General Score
  - Action

From a general benchmarking perspective, metrics such as total runtime, training time, and data loading and storing times are relevant. Timing metrics provided by the ML frameworks, such as epoch training time, are also included. Additionally, the final training and validation loss metrics are important from the ML perspective. In order to measure energy efficiency, we have recorded the power and energy consumption of the GPU, as well as the energy consumption of the nodes used in the benchmarks. Moreover, we defined a general score metric named "Action", the name is due to the metric unit energy × time, that shows the server quality related to the application, the value that minimize the action represents the best configuration that optimize the application execution performance.

In E4 premises, the node power consumption was recorded for all benchmarking runs in this phase thanks to the presence of an intelligent Power Distribution Unit (PDU), which allows the overall power measurement of the single server node and enables the automatic recording of power consumption data at regular intervals during the benchmarking runs.
Power consumption of the individual GPUs was measured using the GetPower script developed at the JSC.
On JSC systems, a power-measurement script GetPower was used to measure individual GPU power and energy consumption for each benchmark run. The GetPower script was available for the NVIDIA A100, H100 and AMD MI250 GPUs
The recorded power consumption data was then used to calculate the energy efficiency metrics for each benchmarking run.

4 Benchmarks

Applications were mainly benchmarked on five different hardware devices, as outline in section 2.2.3: NVIDIA A100 GPU, AMD Instinct MI250 GPU, NVIDIA H100 GPU, Graphcore IPU (all on JURECA-DC); NVIDIA GH200 GPU, NVIDIA A2 GPU (both at E4). The focus was both on training benchmarks and inference performance, with energy consumption a core metric of investigation. For some applications, multiple configurations were investigated. In cases where inconsistencies were found in the metrics of the first 3 runs, the developers were asked to perform more measurements.
In addition to the metrics mentioned in section 3, job-specific information was recorded for each job, which enables querying of job-specific information at a later stage. The following details were recorded:

- Number of CPUs used
- Number of GPUs used
- Number of Nodes used
- Number of MPI tasks
- Job ID
- Node IDs

For each of the application, an overview of the application is given, including the following characteristics of the application:

- Memory training dataset
- Memory validation dataset
- Training samples
- Input shape sample
- batch size
- Trainable parameters
- Non-trainable parameters
- Loss function

- Experimental notes

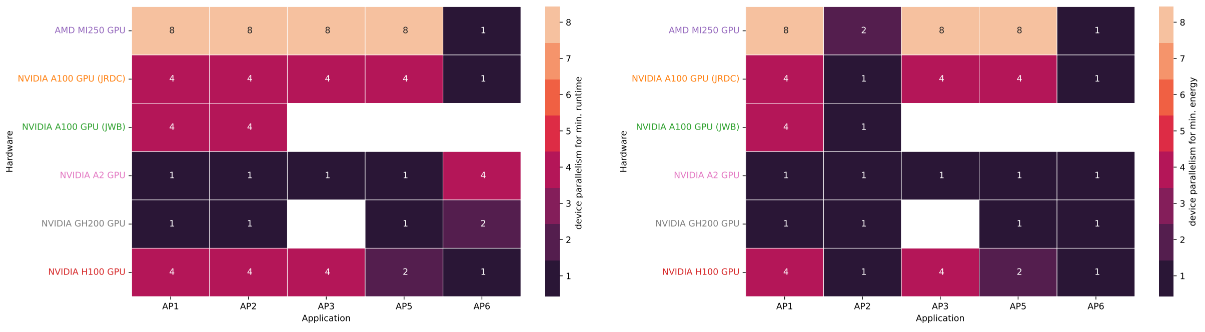
In the following sections, we evaluate the benchmark data, while the total raw data is available in the appendix. First, results from the individual applications are compared to each other, then each application is visited separately.

4.1 Comparative Analysis

In this section, the six MAELSTROM applications are compared to each other, to determine differences and uniquenesses, which can only be seen in relation to each other. If not specified differently, full benchmarks were performed (i.e. no *non-io* variants were used).

4.1.1 Device Parallelism

For this deliverable, the six applications conducted benchmarks on the various systems, evaluation not only different hardware flavours but also the amount of used devices. Each experiment resulted in a certain time of execution (*Time-to-Solution*) and energy used for it (*Energy-to-Solution*). While generally more devices lead to lower Time-to-Solution, Energy-to-Solution can vary and might increase with more devices utilized.



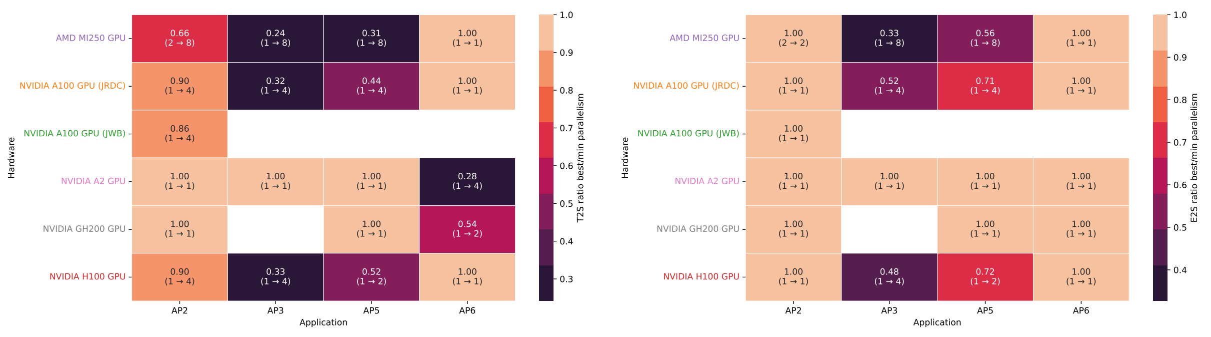
(a) Time-to-Solution (b) Energy-to-Solution

Figure 1: ALL Device Parallelism: Number of devices for minimal Time-to-Solution / Energy-to-Solution all-ttos-device-parallelism

Figure 1 shows the results of the experiment, giving the number of devices needed for best Time-to-Solution (Figure 1a) and Energy-to-Solution (Figure 1b). With the caveats mentioned below, it can be seen that indeed all applications which have tested multiple device scale positively, so that the largest amount of GPUs always leads to best Time-to-Solution. Looking at Energy-to-Solution, AP2 and AP6 are special, as their optimal energy-using configuration only uses a fraction of the available devices in the nodes.
The following caveats apply: AP3 did not conduct full benchmarks with multiple GPUs for experiments on H100 and A100, so the non-io benchmarks were used instead. AP6 was the only application to use multiple nodes. The multi-GPU execution of AP6 had execution errors on JSC machines, which are still under investigation;

both AP2 and AP6 are also the outliers in Figure 1b, which hence should be taken with caution.
All following comparative plots were made with the device parallelism that lead to the lowest Time-to-Solution for the specific application and GPU type.

4.1.2 GPU Scaling



(a) Time-to-Solution

(b) Energy-to-Solution

Figure 2: ALL Scaling: Multi-GPU scaling Time-to-Solution / Energy-to-Solution `all-ttos-scaling`

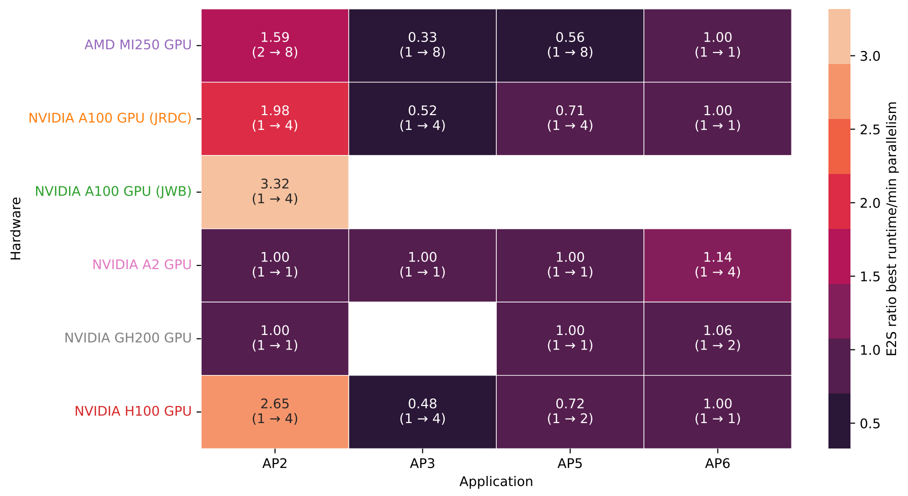


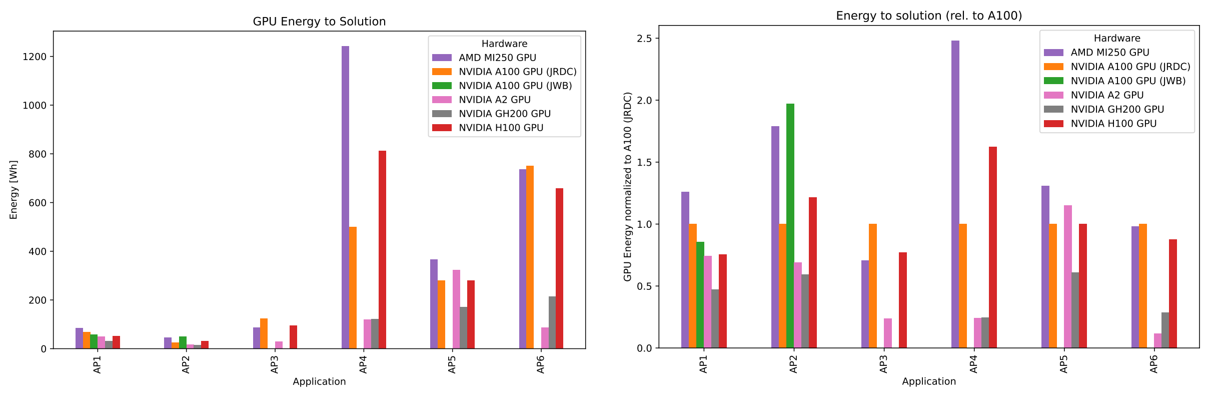
Figure 3: ALL Energy: Energy consumption scaling for parallelism with best runtime; lower is better `all-etos-bttos-scaling`

To visualize the scaling behaviour going from one device to multiple devices, Figure 2 compares relative runtimes/energy usages for the respective applications. In each cell is shown the fraction by which the metric (either Time-to-Solution (Figure 2a) or Energy-to-Solution (Figure 2b)) is improved, when using the number of devices leading to the best value of the metric, and the number of devices which are used for the comparison. For example, AP2 on NVIDIA H100 takes 206 s when executed on 4 GPUs, but 230 s when using 1 GPU; the according cell shows 0.9 (the runtime ratio) and 1 → 4 (the smallest number of tested devices and the best number of tested devices, respectively).

Apart from the previous caveats from section 4.1.1 regarding AP6, AP1 is not shown in this graph, as only multi-device experiments were conducted, and AP4 is not shown, as only single-device experiments were made.

In general, it can be seen that Time-to-Solution scales better than Energy-to-Solution. All applications can benefit from more devices, but to significantly different extent. AP3 and AP5 both scale well, and have lower Time-to-Solution and Energy-to-Solution with more GPUs. AP1 scales very inefficiently, as a four-fold increase in devices only leads to a 10% improvement in runtime.

4.1.3 Energy-to-Solution



(a) Absolute values

(b) Relative values (to each A100)

Figure 4: ALL Energy Usage: Energy-to-Solution for each Application and GPU on JURECA-DC; Benchmark performed with device parallelism that lead to minimal Time-to-Solution; Lower values are better all-eos

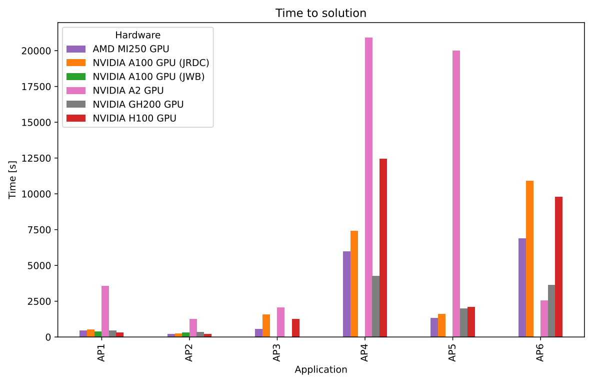
Comparison of Energy-to-Solutions of Figure 4 shows that the GH200 GPU is the most efficient for workloads of AP1, AP2, and AP5. For AP3 and AP6, the A2 GPU is the most efficient, with the H100 GPU coming in the second place for AP3 and GH200 for AP6. Finally, for AP4, A2 is best again with GH200 being in close second. The data for GH200 was not available for AP3.

The MI250 GPU consistently shows higher Energy-to-Solution for most workloads apart from AP6 where it is comparable to A100.

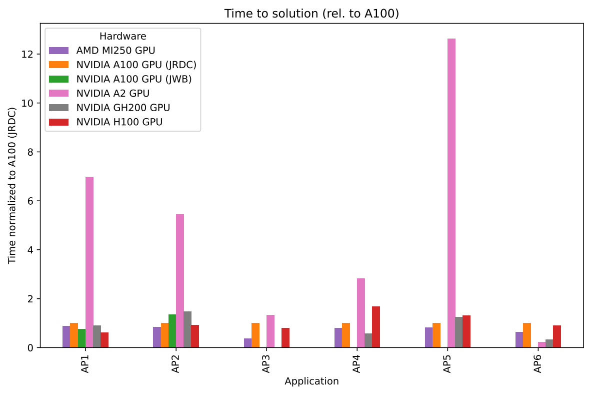
The A2 GPU can achieve quite high energy efficiency in comparison to other devices. However, the A2 is a low-power device, and therefore, its usage is a tradeoff between Energy-to-Solution and Time-to-Solution. In that context, it would be interesting to look at other configurations of systems using the device.

4.1.4 Time-to-Solution

The low-powered A2 GPU is in general slower than the other GPUs, however the slowdown factor varies significantly from application to application - for AP5 it is $> 12\times$ slower than the A100, while for AP3 the factor is just $1.3\times$. AP6 is the only application that used multiple A2 GPUs and is also the only application where the $4\times$ A2 (2 nodes with each 2 GPUs) outperforms all other configurations, delivering $0.23\times$ of the runtime of a single A100. The caveat of AP6 having issues with multi-GPU runs on JSC systems applies here.
The MI250 GPU shows better Time-to-Solution than the A100 for all applications. It must be noted that AP3 didn't perform full-training benchmarks on the A100 and H100 nodes, as well as that the MI250 also shows higher Energy-to-Solution.
The H100 configuration performs best for AP1 and performs better than the A100 for AP2, AP3 and AP6. During the benchmarking, the H100 node was upgraded, which resulted in a performance degradation. The regression is currently under investigation. AP4, AP5 and AP6 benchmarks were performed after the upgrade and are therefore affected, but the extent is unclear.
The GH200 GPU (a single-GPU configuration) is the best for AP4 and worse or on-par with full-node A100/MI250 for AP1, AP2 and AP5. For AP6, the GH200 is the best single-GPU performer and when using two GH200 nodes is the second-best overall (behind $4\times$ A2). It must be noted that nodes with $4\times$ GH200 can be expected in the future and can change the overall picture drastically.

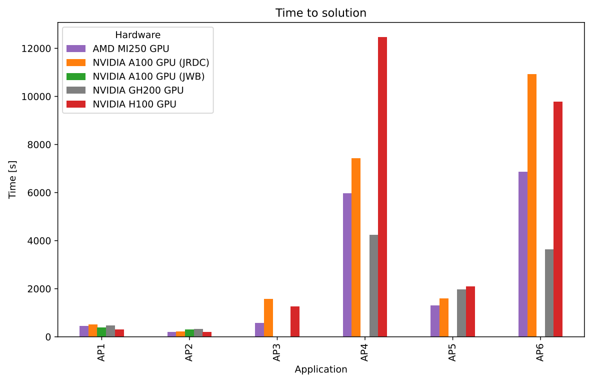


(a) Absolute values

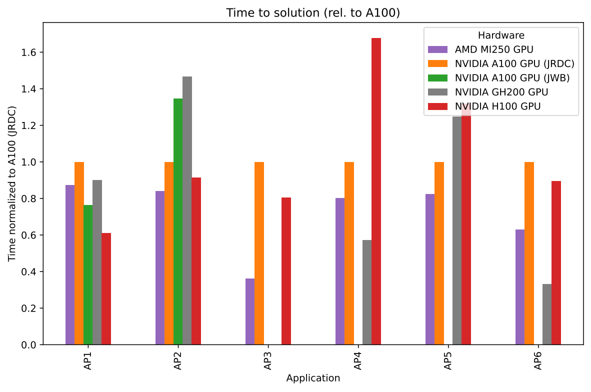


(b) Relative values (to each A100)

Figure 5: ALL Duration: Time-to-Solution for each Application and GPU; Benchmark performed with device parallelism that lead to minimal Time-to-Solution `all-ttos`



(a) Absolute values



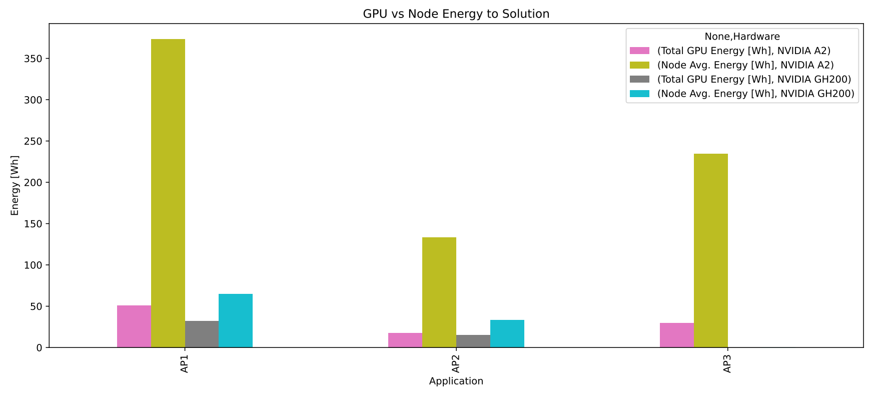
(b) Relative values (to each A100)

Figure 6: ALL Duration (no A2): Time-to-Solution for each Application and GPU except NVIDIA A2; Benchmark performed with device parallelism that lead to minimal Time-to-Solution `all-ttos-noa2`

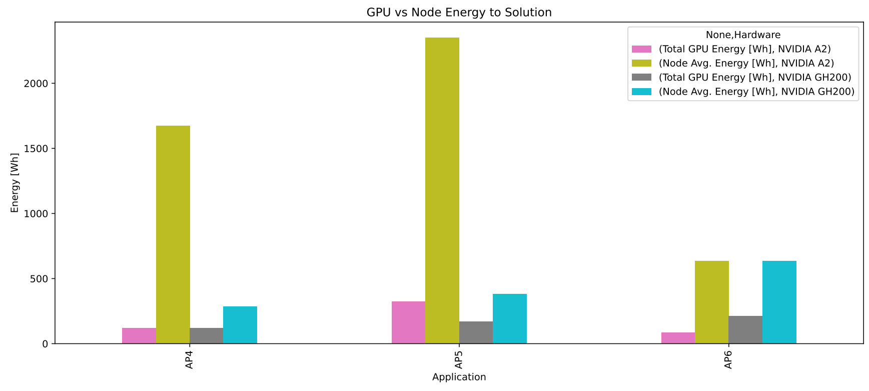
4.1.5 Node energy consumption

The comparison between GPU and node energy consumption (Fig. 7) is presented for the NVIDIA A2 and GH200 systems. In these systems, distribution has been implemented only for the AP6 application.
Overall, due to differences in computational power and execution times, it is observed that, with an equal number of nodes and GPUs, the GH200 system is energetically more efficient than the A2 system for the AP1, AP2, AP4, and AP5 applications.
However, within the context of AP6, the best configuration was plotted considering

distribution as well. The values depicted in the graph illustrate the energy consumption for the A2 and GH200 systems with different hardware configurations. Specifically, the NVIDIA A2 configuration utilizes two nodes with two GPUs each, while the GH200 configuration involves the parallel usage of two nodes, each with one GH200 GPU. From an energy standpoint, it's notable that the two energy values are comparable. These plots serve to provide an overview of the GPU and node energy consumption data across various applications, with detailed analysis of the results to follow in subsequent sections.



(a) AP1, AP2 and AP3



(b) AP4, AP5 and AP6

Figure 7: ALL Node and GPU Energy: Comparison between full-node and GPU energy consumption for A2 and GH200 GPUs; lower is better. all-nodeE-vs-GPU

4.2 AP 1

4.2.1 Notes

In this deliverable, we performed two benchmark studies with Application 1. The application has been modified slightly since D3.6. Here are the main changes:

- Single leadtimes were extracted from files, instead of loading all leadtimes for a given file at once. This improved the shuffling of the data for training.
- Batch normalization was added in each level of the U-Net.
- CPU hyper-threading was exploited where available. That is, all available CPU threads were used.

We performed two benchmark tests. The raw data discussed in this section can be found as tables in appendix 6.1. Firstly we tested the performance of the full application pipeline, end-to-end including I/O, pre-processing, and training.

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
329.83 GB	13.74 GB	89208	[256,256,17]	32

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
1314019	6016	Quantile score (10,50,90%)	3 epochs

Secondly we tested the performance of just the accelerator. For this test, we created synthetic data on the fly to be able to better isolate the performance of the accelerator. In this test we included the Graphcore IPU. As we found the performance benefit of each hardware technology is heavily dependent on the batch size, we performed benchmarks for a wide range of batch sizes to gain insight on performance trade-offs, which may generalize the results to other applications of similar network structure, but with different batch sizes.
The benchmark used the following configuration:

Data Loading dataset	Memory validation dataset	Training samples	Input shape sample	batch size
34.00 GB	N/A	8,192	[256,256,17]	32

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
1314019	6016	Quantile score (10,50,90%)	10 epochs. 1 warmup epoch was discarded

The main findings where:

- Graphcore IPUs performs better than all other hardware for small batch sizes
- Graphcore IPUs used the least amount of energy regardless of batch size
- For larger batch sizes, GPUs perform better.
- Newer generation NVIDIA GPUs are faster and use less energy than the older generation

4.2.2 Runtime split

Loading times on all GPUs are around the 1-2% mark of the total runtime except for one single outlier experiment on JUWELS Booster where it was 5.8%

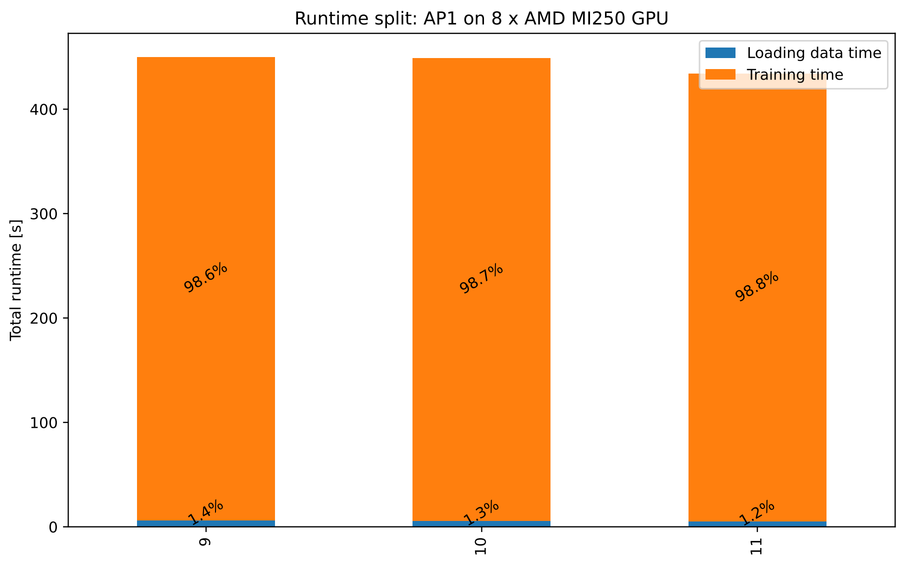


Figure 8: AP1 JURECA-DC MI250: Percentages of runtime spent for training and loading data. ap1-mi250-runtime-split-graph

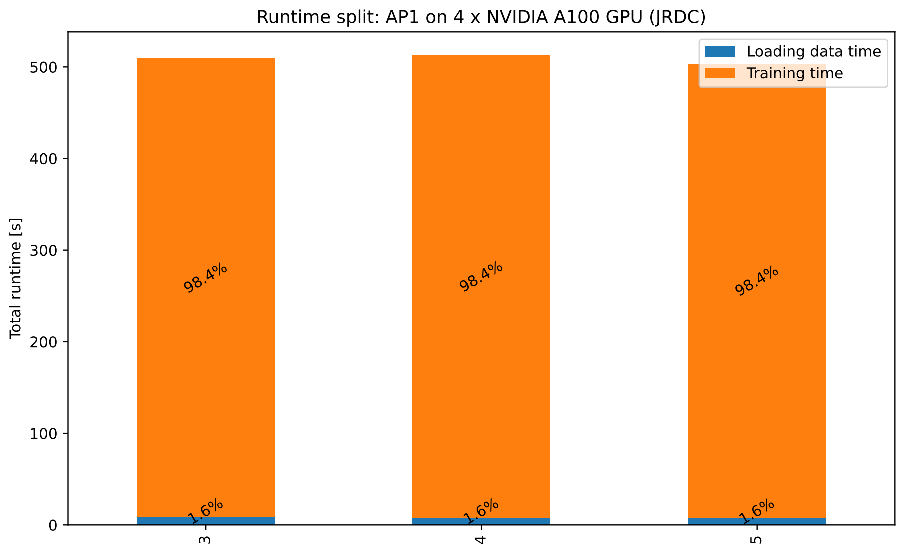


Figure 9: AP1 JURECA-DC A100: Percentages of runtime spent for training and loading data. ap1-a100jrdc-runtime-split-graph

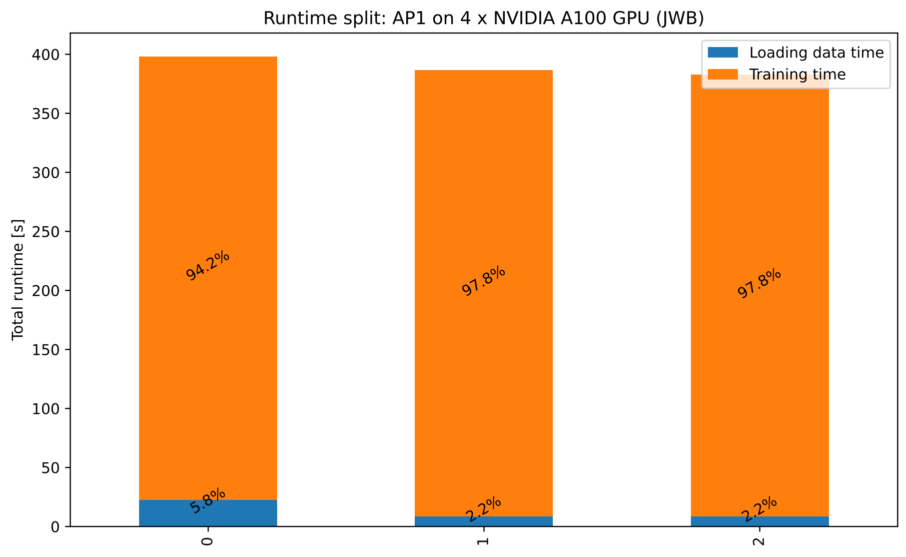


Figure 10: AP1 JUWELS Booster: Percentages of runtime spent for training and loading data. ap1-a100jwb-runtime-split-graph

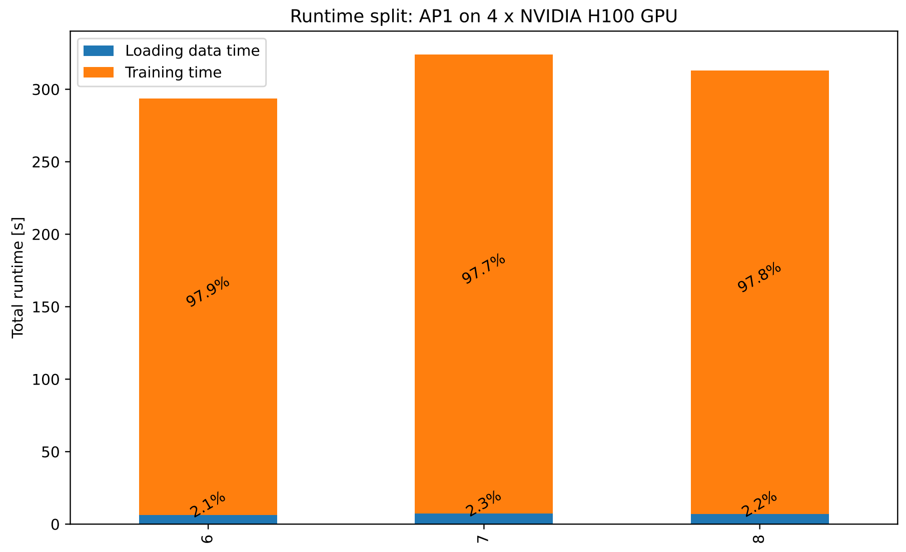


Figure 11: AP1 JURECA-DC H100: Percentages of runtime spent for training and loading data. ap1-h100-runtime-split-graph

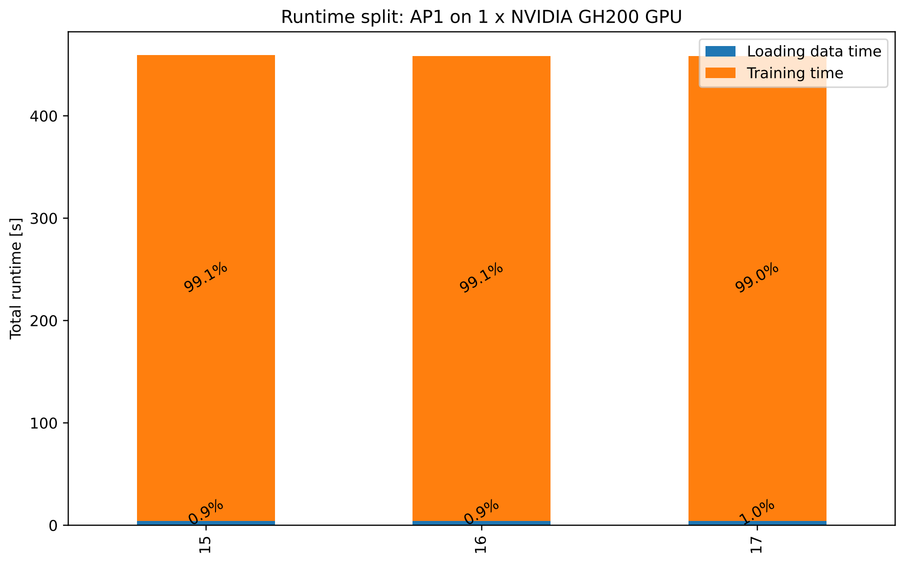


Figure 12: AP1 E4 GH200: Percentages of runtime spent for training and loading data. ap1-gh200-runtime-split-graph

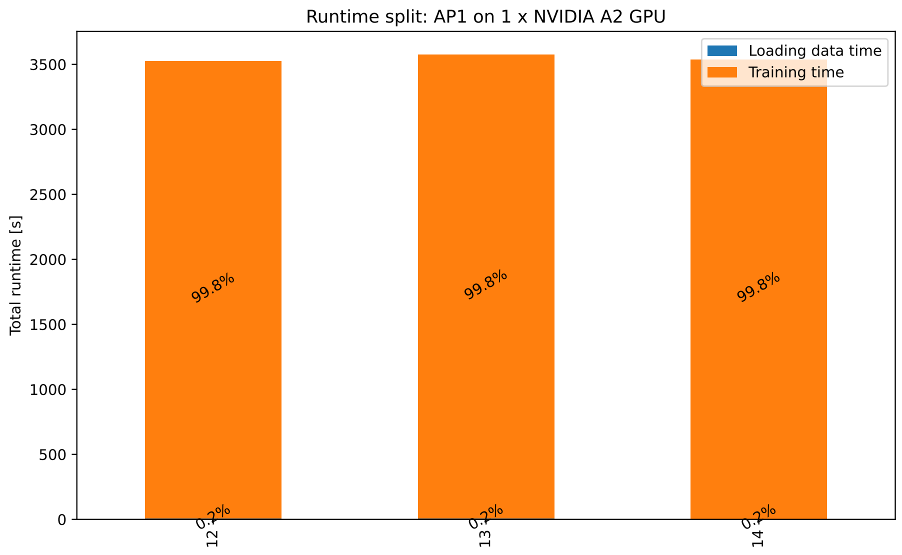


Figure 13: AP1 E4 A2: Percentages of runtime spent for training and loading data. ap1-a2-runtime-split-graph

4.2.3 Non-IO benchmarks

For these training benchmarks, file system I/O was minimized by utilizing synthetic data. They represent how well the utilized hardware is suited for this particular application, as well as how well the application can utilize the hardware. The benchmarks also show how well the tested systems scale with increased batchsize for this application.

4.2.3.1 Energy-to-Solution

In general, we can see that the energy to solution mostly is inversely related to batch size, apart from H100, where we see a slight local minimum for batch size of around 100MB.
In comparison between devices, the IPU has better energy to solution than any other device for batch sizes of upto 8.5MB, beating nearest competitor, the GH200, by around a factor of 4.
For larger batch sizes, GH200 remains the the most efficient device, performing better than previous generations of Nvidia devices.
The AMD MI250 has an energy to solution on par with Nvidia V100 devices, while the A100 in JURECA are marginally but consistently more efficient than the A100 in Juwels Booster by around 5% for large batch sizes.

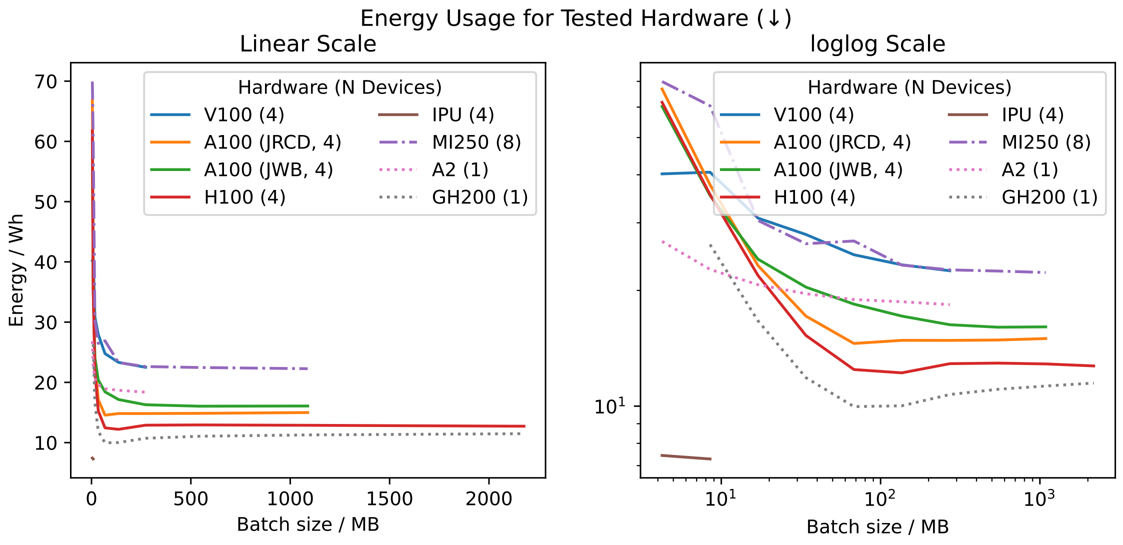
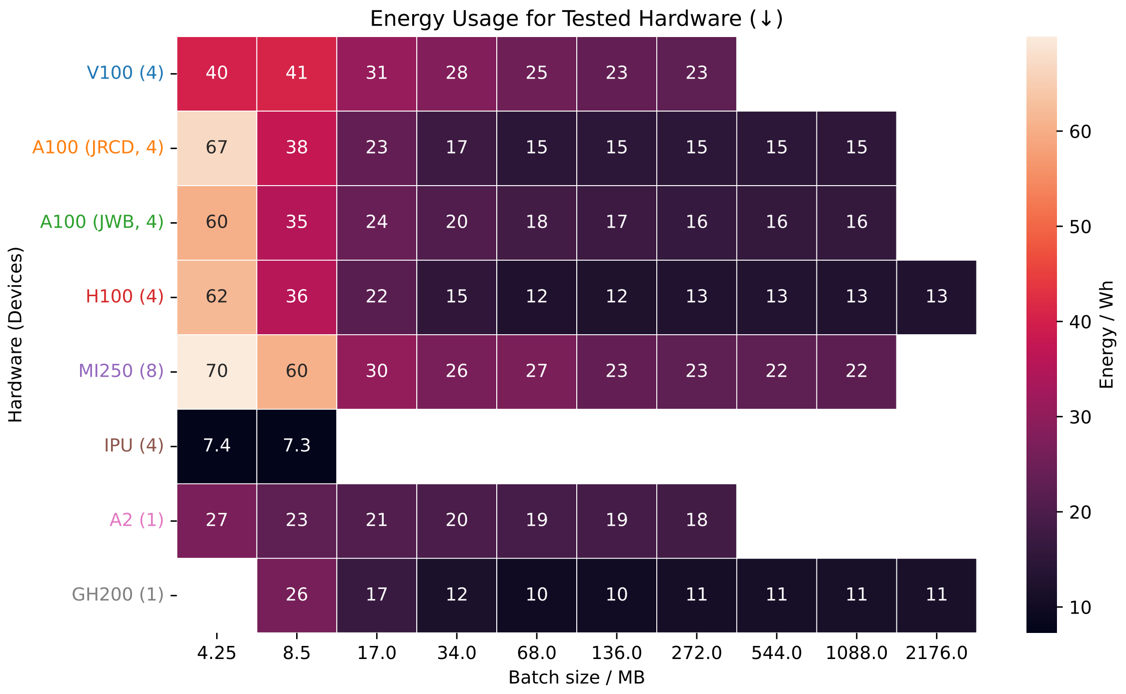


Figure 14: AP1 Energy: Energy-to-Solution on different hardware as a function of batch size. ap1-energy-vs-hardware-graph



Hardware (Devices)	4.25	8.5	17.0	34.0	68.0	136.0	272.0	544.0	1088.0	2176.0
V100 (4)	40	41	31	28	25	23	23			
A100 (JRCD, 4)	67	38	23	17	15	15	15	15	15	
A100 (JWB, 4)	60	35	24	20	18	17	16	16	16	
H100 (4)	62	36	22	15	12	12	13	13	13	13
MI250 (8)	70	60	30	26	27	23	23	22	22	
IPU (4)	7.4	7.3								
A2 (1)	27	23	21	20	19	19	18			
GH200 (1)		26	17	12	10	10	11	11	11	11

Figure 15: AP1 Energy: Energy-to-Solution on different hardware as a function of batch size. `ap1-energy-vs-hardware-heatmap`

4.2.3.2 Time-to-Solution

In time to solution, for small batch sizes, the IPU again performs better than all other devices. However, when the batch size is increased, H100 and A100 have better time to solution than the IPU. IPU however still performs better than V100.
The configuration of 4 × H100 performs better than 8 × MI250, which in turns performs better than 4 × A100. The single GH200 performs worse than 4 × H100, however, the difference is only around a factor of 2 and not a factor of 4 as would be expected in a perfect-scaling workload in a like-for-like comparison.

4.2.3.3 Node Energy Consumption

Comparing both the GPU and Node (Fig. 18) energy consumption between the two hardware types, NVIDIA A2 and NVIDIA GH200, reveals significant performance and energy consumption differences. For E4-A2 hardware, which boasts the longest total runtime, the average GPU energy consumption amounts to 50.09 Wh, while the E4-GH200 hardware, with a shorter total runtime, exhibits lower GPU energy consumption at 32.16 Wh.
Further analysis of the average node energy consumption highlights substantial

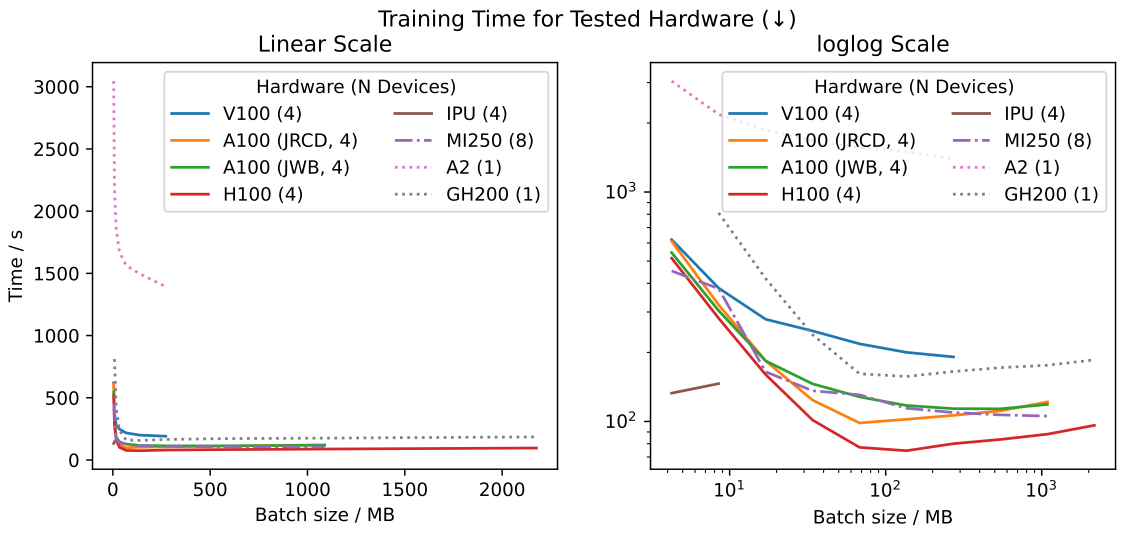


Figure 16: AP1 Time: Time-to-Solution on different hardware as a function of batch size. *ap1-time-vs-hardware-graph*

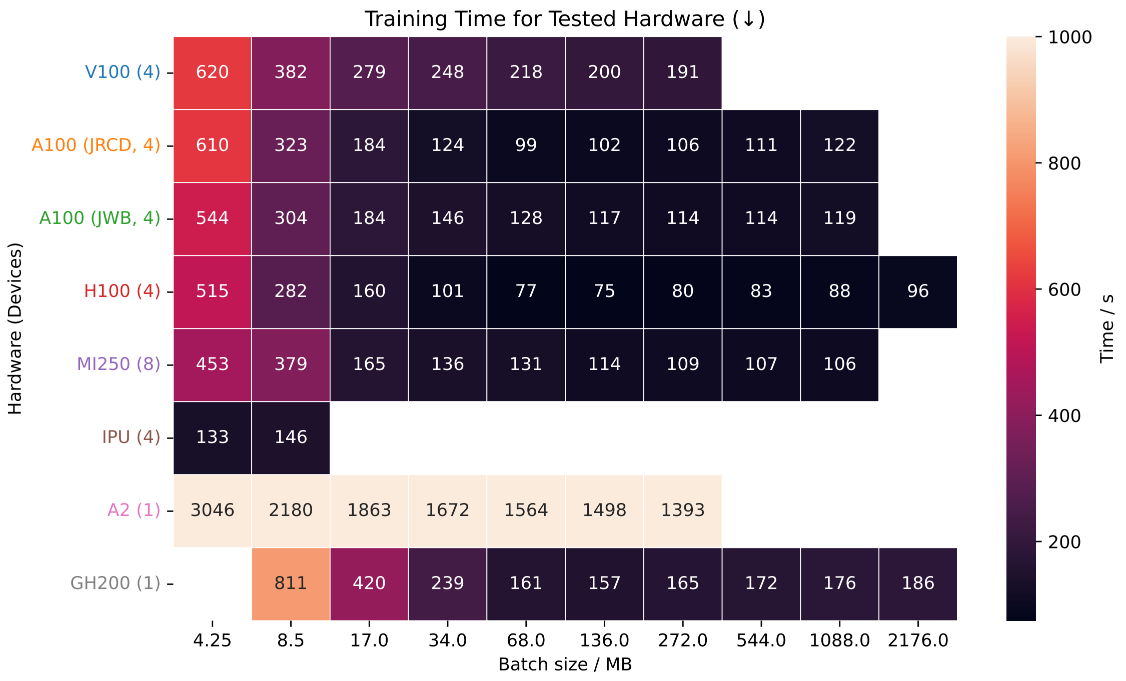
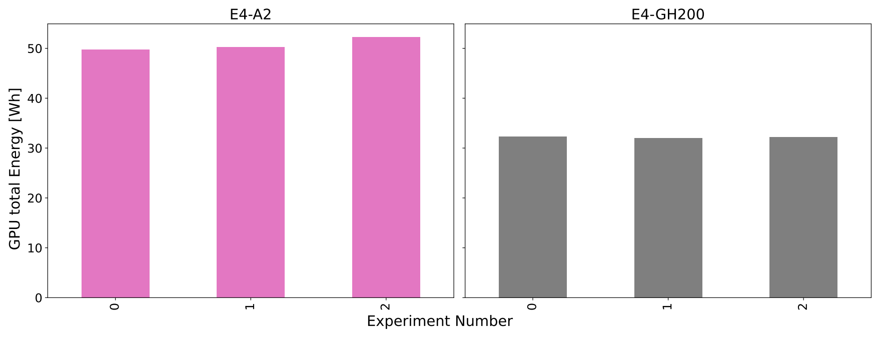


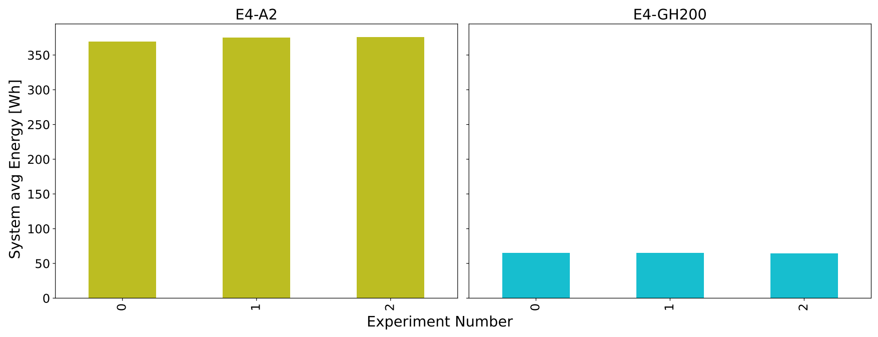
Figure 17: AP1 Time: Time-to-Solution on different hardware as a function of batch size. *ap1-time-vs-hardware-heatmap*

variations between the two hardware types. E4-A2 hardware demonstrates an average system energy consumption of 373.09 Wh, whereas E4-GH200 hardware operates at a significantly lower average system energy consumption of 64.72 Wh.

Additionally, GH200 hardware exhibits a notably higher percentage of energy used by the GPU compared to A2 hardware, averaging 49.70% compared to 13.59%. Moreover, analyzing the Action metric (Table 1) offers valuable insights into the overall system performance and energy efficiency. The distribution of Action reveals a higher score for A2 hardware compared to the GH200 system, with the latter emerging as the most performant option.



(a) GPU energy consumption



(b) Node energy consumption

Figure 18: AP1 Node and GPU Energy comparison: Comparison between full-node and GPU energy consumption for A2 and GH200 systems. all-nodeE-vs-GPU

Experiment	Hardware	Action [MJs]
0	E4-A2	4684.42
1	E4-A2	4828.46
2	E4-A2	4786.74
3	E4-GH200	107.31
4	E4-GH200	106.95
5	E4-GH200	106.24

Table 1: AP1 Action metric for A2 and GH200 systems.

4.3 AP 2

4.3.1 Notes

The application is showing the following details:

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
12 MB	3.8 MB	148k	(128100, 768)	32

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
141 896 450	0	cross entropy loss	Finetuning pre-trained model with small benchmark dataset

Data formats	Frameworks (to be) used
NetCDF, CSV (GH, A2 only)	PyTorch 2.2.0a0+81ea7a4 (GH, A2, A100), 2.1.0a0+32f93b1 (H100), 2.1.2 (MI250

The task of Application 2 is to obtain weather-related information from social media posts and use them as an additional data source to improve weather predictions. As a test case, we aim to classify Tweets as "raining" or "not raining". Our current solution is based on a deep transformer based neural network ("deberta-v3-small") that is pre-trained on a large corpus. We focus on fine tuning the model to adopt it to our specific domain (see Deliverable D1.4 for more details). A single epoch suffices to finetune our model. The model can be trained on multiple GPUs in parallel. Here, we vary the number of used GPUs to analyze the efficiency of parallelization and its efficiency on different systems. To allow for more iterations, we only use a tenth of our full training set.
First, we outline the definitions for our measured timescales as used throughout the analysis. Our "Total training time" includes model setup, actual training time and quick evaluation of model performance. Our dataset is small enough to be loaded into RAM memory. We exclude time required to load the dataset into RAM from "Total training time". We train for a single epoch. The time required for the actual training is reported as "Training time for epoch". During this training step, the data has to be provided to the model, the total time required for this process defines our "Total IO time".

Instead of re-running our model for dedicated evaluation runs, we instead opt to compute predictions for a holdout validation set. This allows us to measure performance of model inference termed "Evaluation Time". While this ignores the effect of data and model loading, we verified that these timescales are negligible for our application when compared to inference time.
Regarding the power consumption, different measurements were taken depending on the system. In the case of both JSC and E4, we provide power consumption of the GPU only. In the case of E4, we additionally report power consumption of the whole node.
Overall, training time is dominated by actual training tasks (gradient computation, backpropagation, etc.). Additional processes like model setup, IO, etc. make up less than < 10% of total training time. Therefore, most significant speed ups are expected from more efficient training algorithms and/or parallelization of model training.
Note, that we used a significantly older PyTorch version for the previous deliverable. We therefore expect significant overall performance differences for the model compared to previous Deliverable 3.6.
As demonstrated in Deliverable 3.6 training time scales well below linearly with the number of GPUs used. When using 2 (4) GPUs on JUWELS Booster, we found speed ups of 1.6x (2.5x). Here, we retested performance on multi GPU systems as we expect significant improvements in this field from software and hardware developers. However, the increase in performance when using more GPUs is still rather small for our model across all tested systems. The highest increase in performance ("Training time") is seen for the Mi250 system where we get speed ups of 1.4x (1.6x) for 2 (4) GPUs, respectively. For A100/H100, speed ups are only of order 1.1x for 4 GPUs. Using two A100 GPUs even seems to slightly decrease performance compared to a single A100 GPU.
To compare performance differences by GPU model, we now use A100 as the baseline. Comparing GPU models, we see clear differences in performance when comparing single to multi-gpu setups for our application. While 2 (4) Mi250 are 1.2x faster than 2 (4) A100, a single Mi2500 is slower than a single A100 by 0.9x. A single H100 (four H100) provides 1.1x speed-ups compared to a single (four A100). Interestingly, two H100 provide higher speed-ups of 1.2x compared to two A100 GPUs. The single Grace-Hopper GPU reaches 0.8x speeds of an A100 GPU. For the single A2 GPU tested the performance is only 0.2x of a single A100 GPU. This significant gap in performance may be related to a subpar setup, as the measured power draw of the GPU itself is merely 50 W. Further investigation is required to

clarify the issue.
For inference time, we note that measured inference time do not necessarily correlate with training time. We note that the required overhead for parallelization appears to degrade performance (at least for our small dataset) when using more than a single GPU for both A100 and H100. Only, Mi250 GPUs show clear performance gains when run in parallel and using 4 GPUs is substantially faster than a single Mi250 (1.3x faster). However, a single H100 is still 1.1 faster than 4 Mi250 (at significantly smaller power draw). Interestingly, while training performance of the Grace-Hopper was significantly slower compared to other systems, a single Grace-Hopper is on par with a single H100, the fastest tested system for inference of our application.
Turning to GPU energy efficiency, we will focus on single GPU measurements as multi-gpu systems currently only provide modest relative speed ups when compared to a single GPU. Therefore, even though nodes at JSC usually provide four GPUs minimum, we only quote power consumptions based on the GPUs that were actually used for computation. This is in contrast to Deliverable 3.6, where we gave total powers for the whole node.
Based on total energy consumed by the GPU, we find that a reference single A100 GPU uses 1.1x more energy than a single H100. All other tested systems consume more energy than the A100, making the single H100 GPU the most efficient system. While systems with four GPUs are faster, the single H100 consumes only 0.4x (4xH100, 4xA100) or 0.3x (4xMi250) of the energy. Systems with two GPUs are slightly more efficient where a single H100 consumes 0.6x (2xH100), 0.5x (2xA100) and 0.3x (2xMi250) the energy. The Mi250 It appears that the special design of the Mi250 may be more fitting for applications that are efficiently parallelized.
Memory consumption for CPUs and GPU(s) are comparable between all tested systems at $< 10\,\mathrm{GB}$ and $< 5\,\mathrm{GB}$, respectively. This makes training feasible on consumer-grade systems with a high end graphics card model. For evaluation, even just CPUs are sufficient for our current dataset.

4.3.2 Runtime split

Loading times on JSC systems (MI250, A100, H100) are around the 6-8% of the total runtime, on E4 systems they are lower - between 1.7 and 4%. It has to be noted that the runtime for the A2 GPU is significantly longer than on other GPUs, so the loading time, which is between 1.7% to 1.9% is larger as an absolute value. We can also observe a measurable portion of time spend on neither training nor data loading. This portion is between 1-3% on most systems, with JUWELS Booster and the MI250 nodes being an exception with ≈ 5 and ≈ 4.5% respectively. One outlier experiment on the MI250 node shows a larger 12% portion.

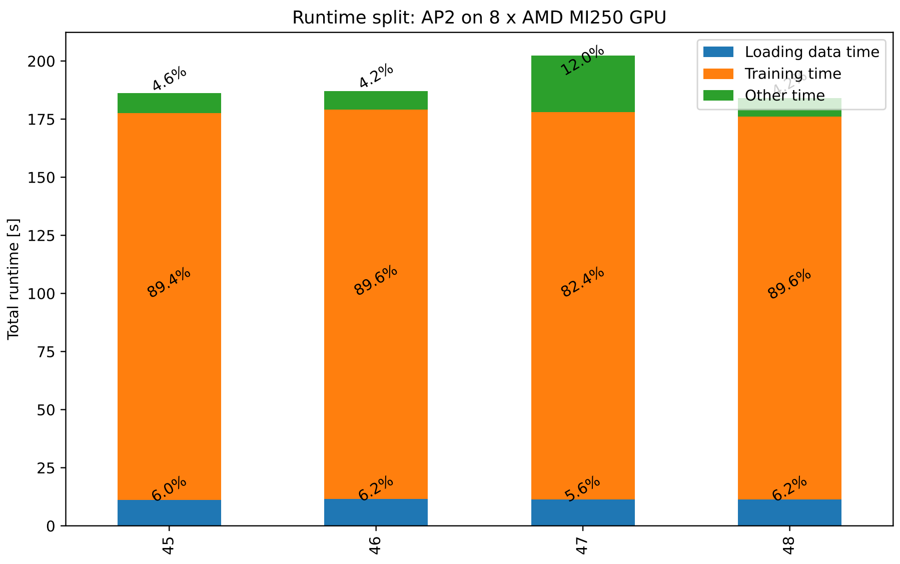


Figure 19: AP2 JURECA-DC MI250: Percentages of runtime spent for training and loading data. ap2-mi250-runtime-split-graph

4.3.2.1 Node Energy Consumption

Considering energy consumption, as illustrated in Figure 25, the GH200 system demonstrates a total GPU energy consumption of 15.51 Wh, with a system average energy of approximately 33.39 Wh. Notably, the GPU significantly contributes to the overall energy consumption, representing an average of 46.92% of the total energy usage.
In contrast, the A2 system showcases slightly higher total GPU energy consumption, averaging around 17.70 Wh while the system average node energy is notably higher, at approximately 133.57 Wh. Additionally, the GPU's contribution to the total energy consumption is substantially lower for E4-A2, accounting for approxi-

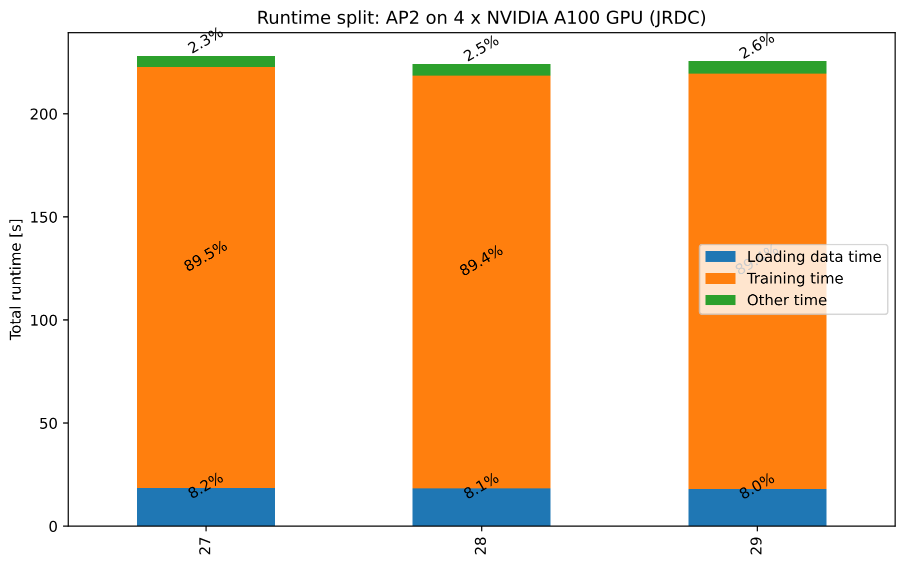


Figure 20: AP2 JURECA-DC A100: Percentages of runtime spent for training and loading data. ap2-a100jrdc-runtime-split-graph

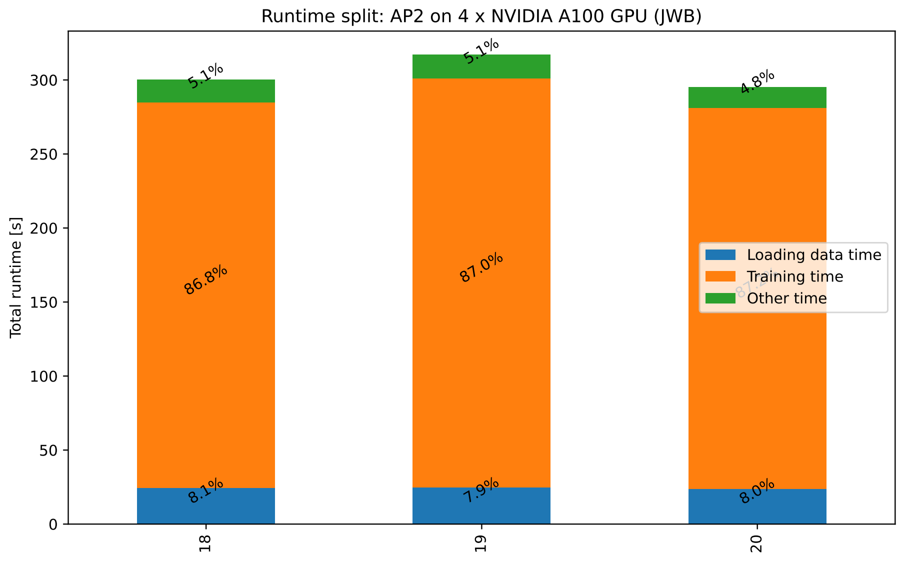


Figure 21: AP2 JUWELS Booster: Percentages of runtime spent for training and loading data. ap2-a100jwb-runtime-split-graph

mately 13.29% of the node's energy. This suggests that with a low-power GPU like A2, the rest of the system consumes more power than the GPU itself.

Furthermore, the energy comparison findings are reinforced by the Action metric (Table 2). The average Action for the GH200 system is approximately 40.95, whereas for A2, it notably surpasses that of E4-GH200, reaching around 592.64.

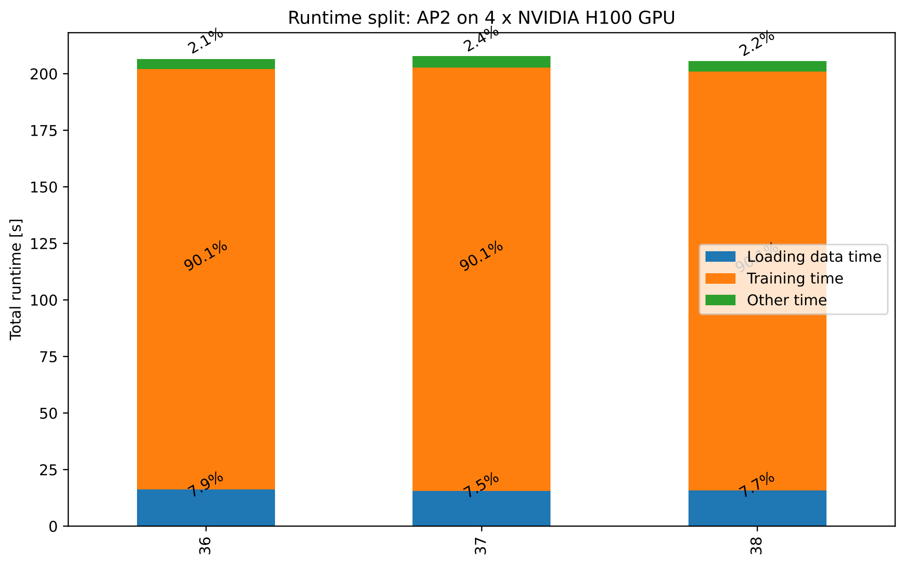


Figure 22: AP2 JURECA-DC H100: Percentages of runtime spent for training and loading data. ap2-h100-runtime-split-graph

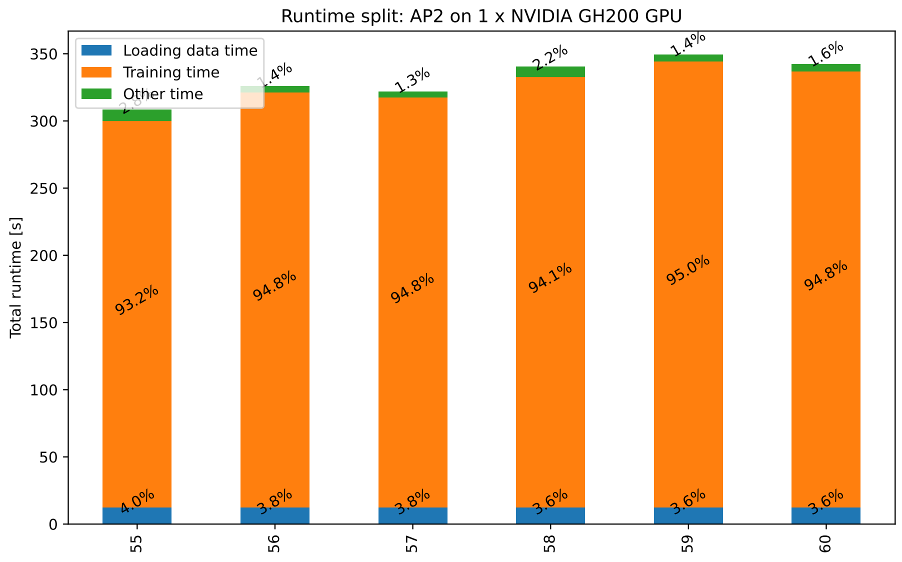


Figure 23: AP2 E4 GH200: Percentages of runtime spent for training and loading data. ap2-gh200-runtime-split-graph

This disparity indicates better and more consistent performance and energy consumption behavior across experiments for E4-GH200, while E4-A2 exhibits higher consumption and lower performance.

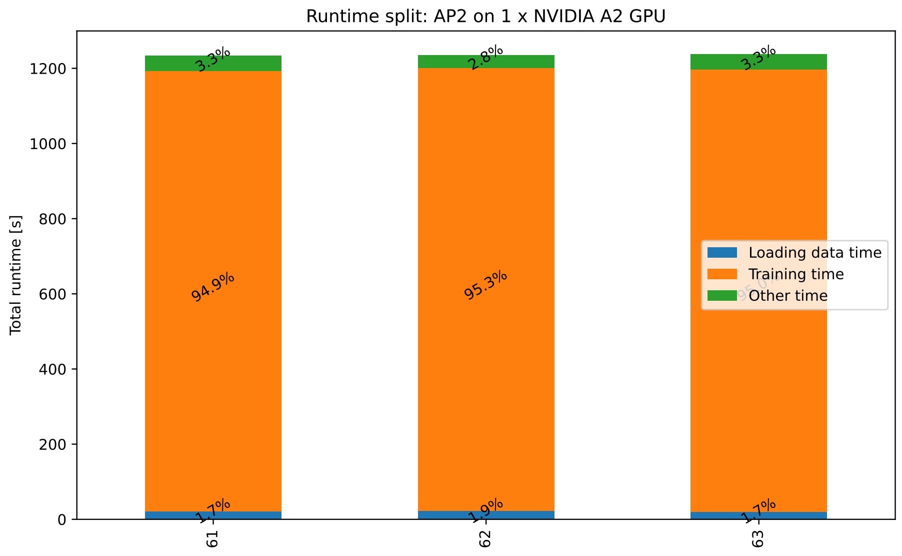
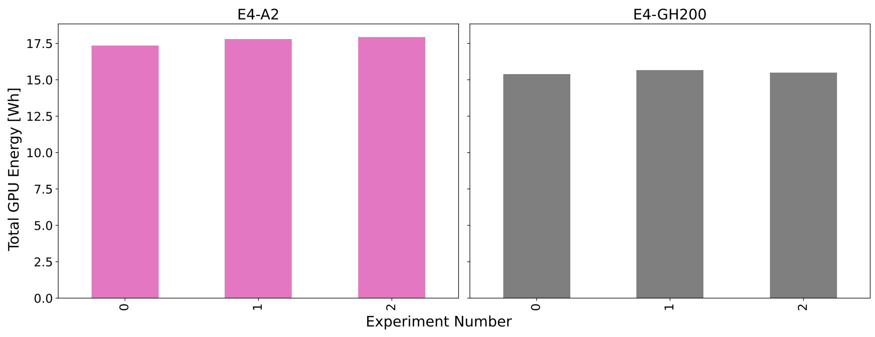


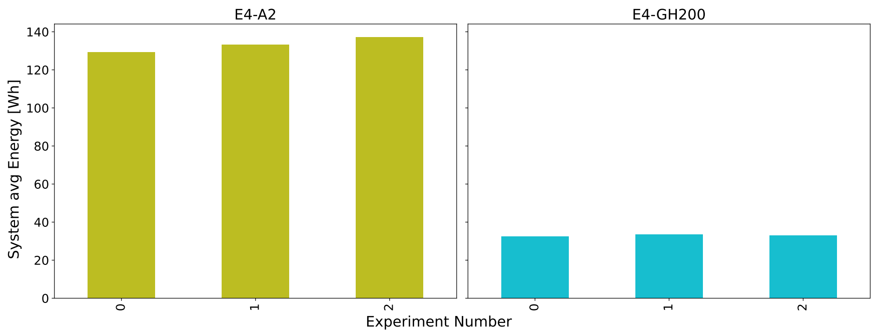
Figure 24: AP2 E4 A2: Percentages of runtime spent for training and loading data.
ap2-a2-runtime-split-graph

Experiment	Hardware	Action [MJs]
0	E4-A2	574.32
1	E4-A2	592.18
2	E4-A2	611.44
3	E4-GH200	39.96
4	E4-GH200	42.26
5	E4-GH200	40.62

Table 2: AP2 Action metric for A2 and GH200 systems.



(a) GPU energy consumption



(b) Node energy consumption

Figure 25: AP2 Node and GPU Energy comparison: Comparison between GPU and full-node energy consumption for A2 and GH200 systems. ap2-nodeE-vs-GPU

4.4 AP 3

4.4.1 Notes

Compared to the previous benchmarking deliverable, no new modifications have been introduced for AP3 relating to datasets or model architecture. Details of the application are displayed in the table below.

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
60 GB	4.2 GB	2 984 960	(17), (137, 27), (138, 2), (138, 1)	512

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
261 515	0	MSE (multiple output vectors)	Model not trained to convergence for cost reasons (only 5 epochs), 50 epochs required

Data formats	Frameworks (to be) used
NetCDF	TensorFlow 2.X

AP3 experiments have been performed on both JURECA-DC and the E4 systems. The experiments have evaluated the training and inference phases as done during the previous deliverable. In this deliverable, we have introduced a **Non-IO test**, where no data loading from disk is done, rather a fake dataset is defined with the aim to focus on the model performance between the different hardware configurations, including NVIDIA GPUs, AMD GPUs, and Graphcore IPUs. The table below provides a summary of the specific hardware tested, with the associated phase (training/inference or Non-IO) and system.

Experiment Set	**HARDWARE TESTED**
NON-IO Experiments	A100, H100, Mi-250, Graphcore IPUs, GH200 and A2
Training Phase	A100, H100, Mi-250, A2
Inference Phase	A100, H100, Mi-250, A2

Table 3: Summary of hardware tested for Application 3.

In the experiemnts performed using JURECA-DC, the training and inference phases were tested using 3 configurations. The configurations are:

- *None*: Default version without special flags
- `--nocache`: Avoid using TensorFlow dataset cache
- `--dl_test`: Iterate through the training dataset without training the model, to test data-loading capabilities.

Raw data for graphs and discussions of this section is listed in appendix 6.3.

4.4.2 JURECA-DC - NVIDIA A100

On JURECA-DC, 9 experiments have been carried out using as hardware NVIDIA A100 GPUs. Dividded into 3 triplets:

- Triplet 1: `--nocache` flag and SCRATCH filesystem.
- Triplet 2: no flag and SCRATCH filesystem.
- Triplet 3: `--dl_test --nocache` flag and SCRATCH filesystem.

The number of GPUs and MPI tasks used in these experiments was set to 1. Results are also reported for the inference phase, where a total of three experiments were performed.

4.4.2.1 Training

To assess the results obtained during the training phase, we will analyse the runtime as well as the comparison between the training time for the first epoch and average training time per epoch. All times reported are meassured in seconds.
In line with the results obtained during the first benchmarking efforts in deliverable D3.6, the runs with the nocache flags are the ones with the largers training times as this flag avoids tensorflow to use the dataset caching features. Between the nocache flag and the default bersion we see a reduction of the total training time of approximately 25.5%. The rest time sits between for 0.25% for the runs using default and dl-test nocache runs and 0.15% for the nocache runs.
The total runtime is shown in Figure 26 for the runs with different flags.

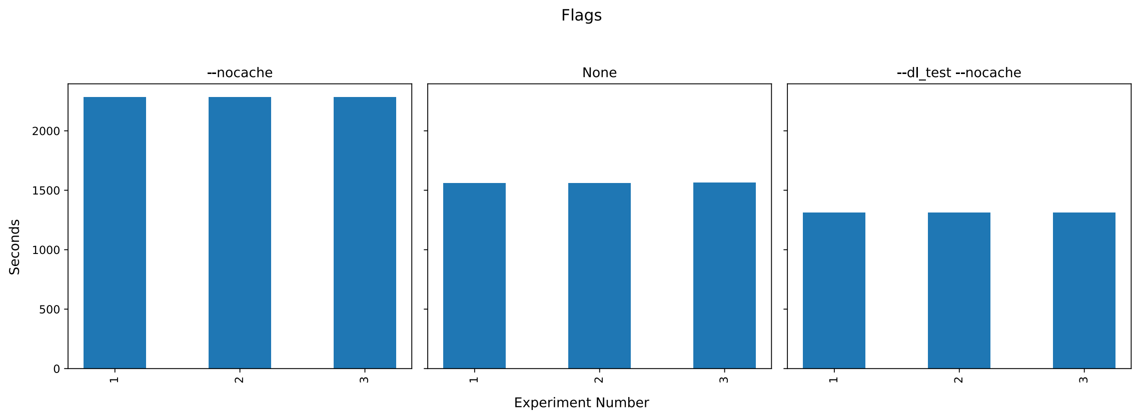


Figure 26: AP3 JURECA-DC A100 Runtime: Runtime for multiple experiments during the training phase. *ap3-jrca100-runtime-share*

Figure 27 shows the comparison between the first epoch training time and the average training time per epoch, and we can see similar values between them or a ratio close to 1 for the triplets 1 and 3. In the runs using the default configuration that takes advantage of the TensorFlow caching we see a bigger difference, which could be caused by time required to the cache the data during the first epoch.

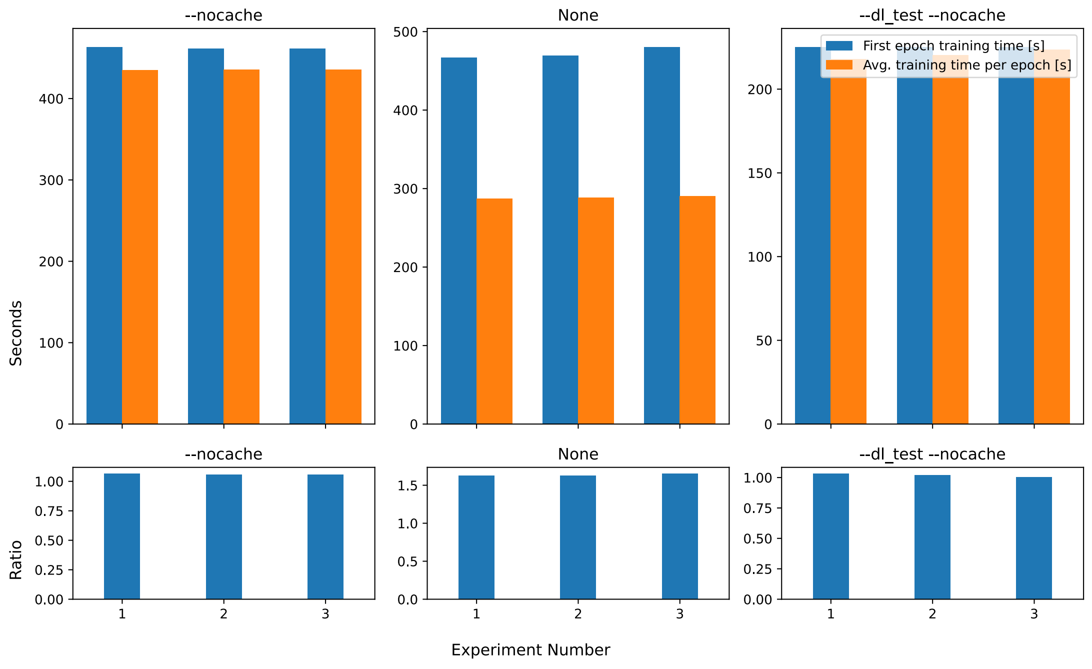


Figure 27: AP3 JURECA-DC A100 Epoch Time: Comparison of time for first epoch and average time for all epochs (top); ratio of both quantities (bottom). ap3-jrca100-epoch-time

GPU Energy Consumption has beeen measured using NVIDIA pynvml package. For the A100 GPUS, as displayed in Figure 28 we obtained an average consumption of 124.44 Wh for the default configuration. When we use the nocachin configuration this amount increases by 26%. In the data loading runs the average consumption goes down to 88.76 Wh.

4.4.2.2 Inference

The results of the inference phase can be seen in Figure 29. In this case, we run 3 runs without any flags. When using the A100 GPUs, we obtained an average inference time of 46 seconds with a data loading overhead time of around 2.7

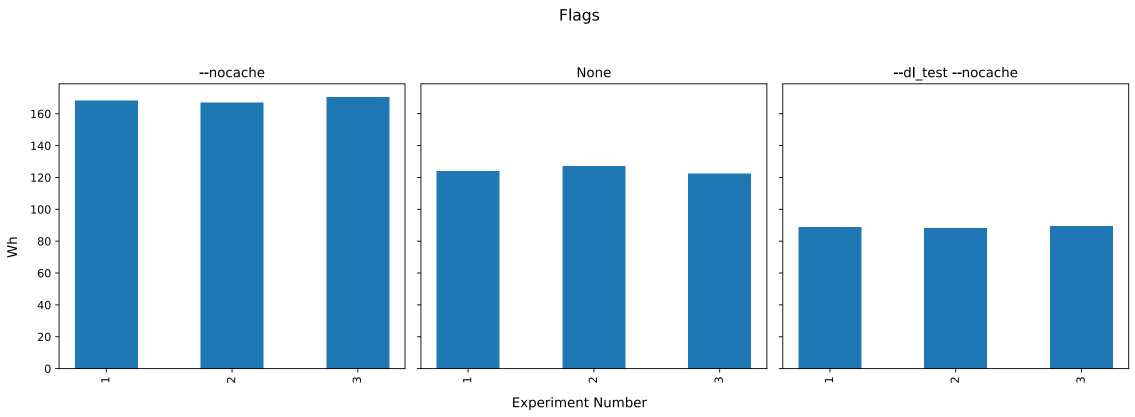


Figure 28: AP3 JUWELS Booster Energy: Total GPU energy consumption during the training phase. *ap3-jrca100-energy*

seconds. The rest time which referes to the difference between the end of the inference process and the end of the total runtime, has an average value of 4.5 seconds.

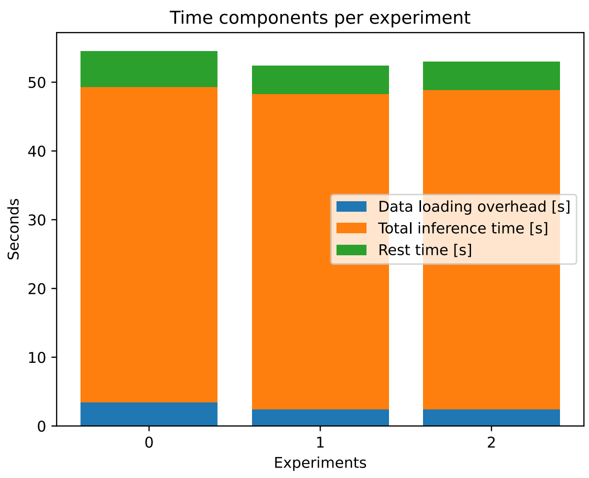


Figure 29: AP3 JUWELS Booster Inference Runtime: Runtime and relative share for multiple experiments during the inference phase *ap3-jrca100-inf-runtime-share*

4.4.3 JURECA-DC - NVIDIA H100

Using JURECA-DC, 9 experiments have been carried out using as hardware NVIDIA H100 GPUs. Divided into 3 triplets: Triplet 1: nocache flag and SCRATCH filesystem. Triplet 2: no flag and SCRATCH filesystem. Triplet 3: dltest nocache flag and SCRATCH filesystem. The number of GPUs and MPI tasks used in these experiments was set to 1. Results are also reported for the inference phase, where a total of three experiments were performed.

4.4.3.1 Training

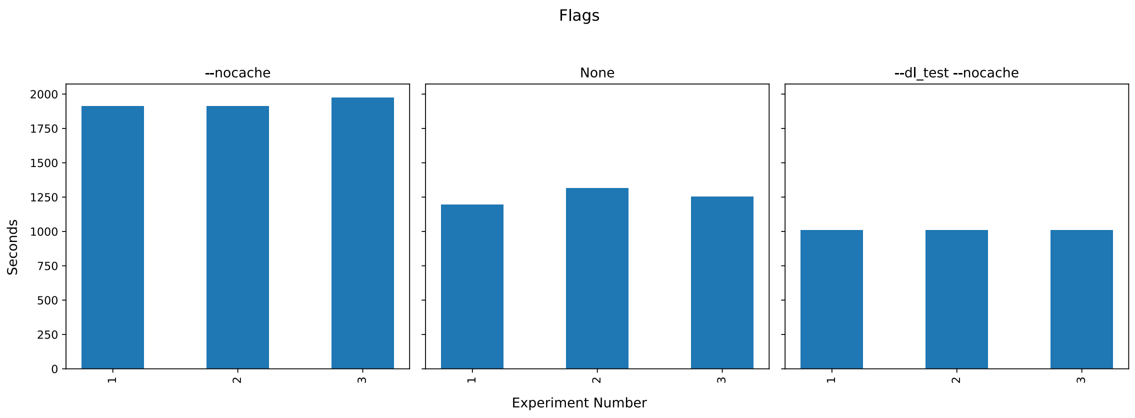


Figure 30: AP3 JURECA-DC H100 Runtime: Runtime for multiple experiments during the training phase. ap3-jrch100-runtime-distri

4.4.3.2 Inference

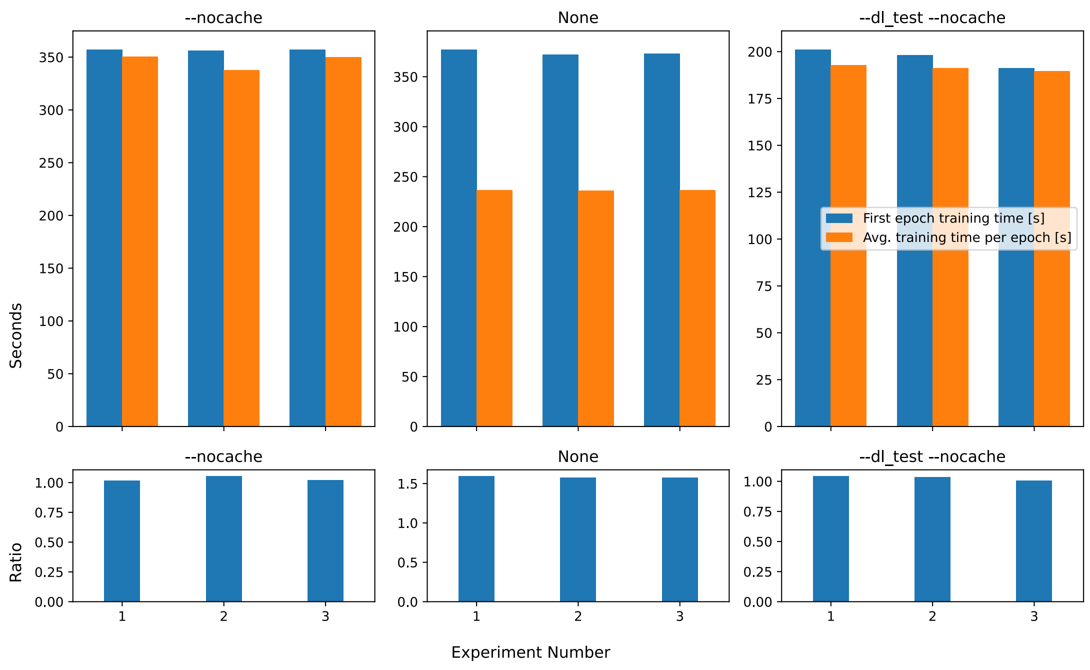


Figure 31: AP3 JURECA-DC H100 Epoch Time: Comparison of time for first epoch and average time for an epoch (top); ratio of both quantities (bottom). ap3-jrch100-epoch-time

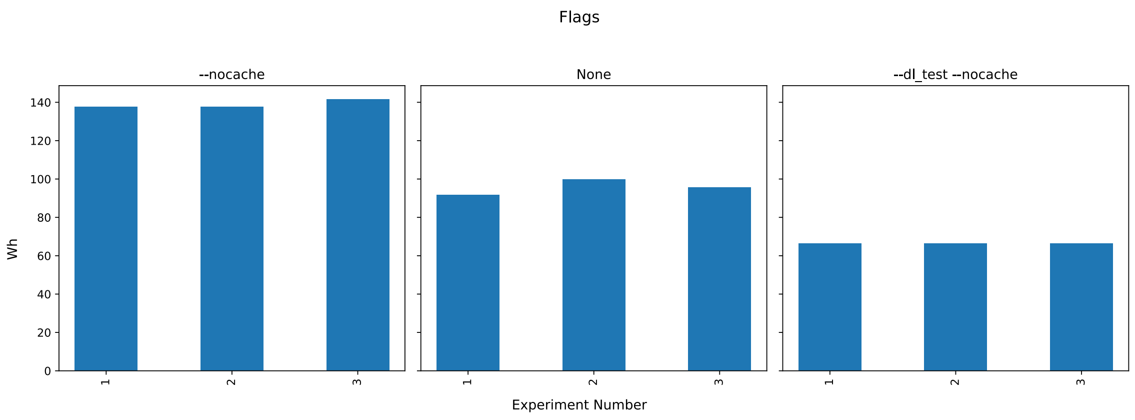


Figure 32: AP3 JURECA-DC H100 Energy: Total GPU energy consumption during the training phase. ap3-jrch100-energy

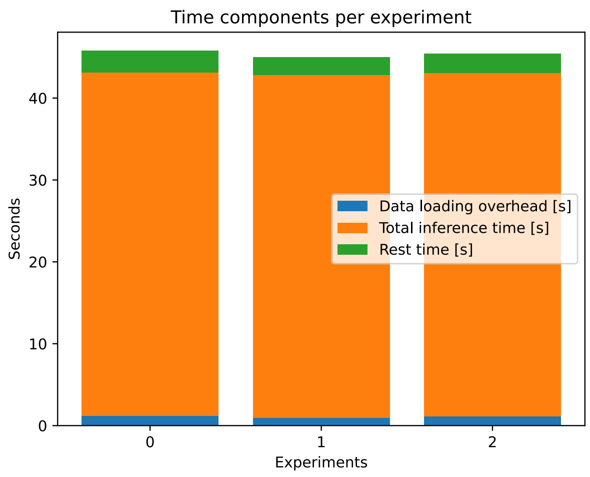


Figure 33: : Runtime and relative share for multiple experiments during the inference phase `ap3-jrch100-inf-runtime-share`

4.4.4 JURECA-DC - AMD MI200

JURECA-DC also provides access to AMD GPUs, so for this deliverable we carried out 18 experiments have been carried out using as AMD Mi200 GPUs. Dividded into 3 triplets: Triplet 1: nocache flag and SCRATCH filesystem. Triplet 2: no flag and SCRATCH filesystem. Triplet 3: dltest nocache flag and SCRATCH filesystem.
Results are also reported for the inference phase, where a total of six experiments were performed. On JURECA-DC, a single node provides access to 8 MI250 GPU Chip Dies (GCD). To assess the scalability of the MI200 nodes, we decided to run a first group of experiments (9 tests for the training phase and 3 tests for inference) where we forced TensorFlow to use only one GCD, and a second set of experiments where we use all the GCDs.

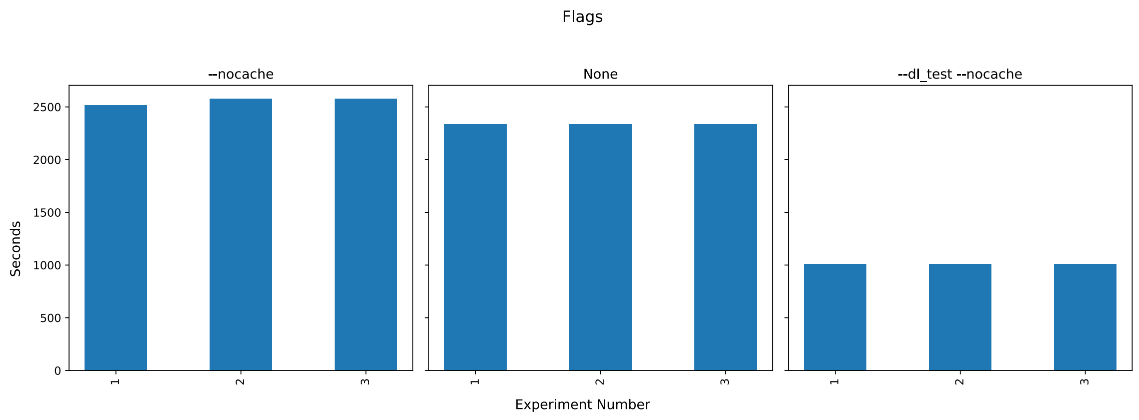


Figure 34: AP3 JURECA-DC MI250 Runtime: Runtime for multiple experiments during the training phase. ap3-jrcmi250-runtime-share

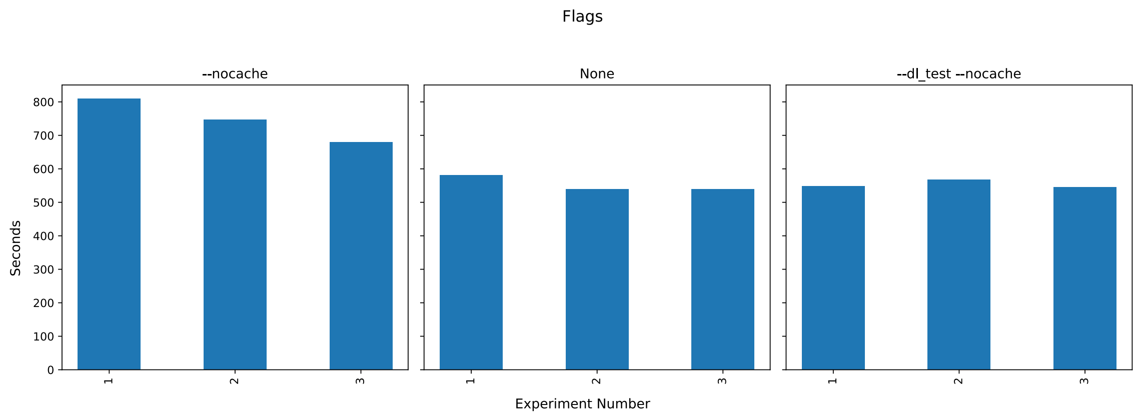


Figure 35: AP3 JURECA-DC MI250 Runtime: Runtime for multiple experiments during the training phase (8 MI250 GPU Chip Dies). ap3-8jrcmi250-runtime-share

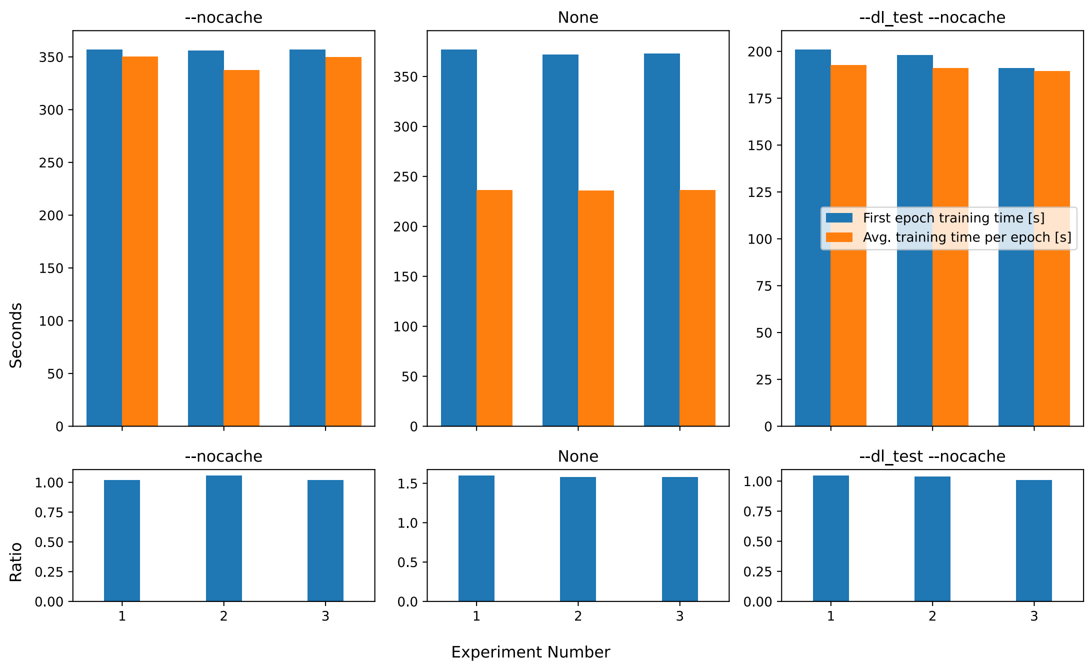


Figure 36: AP3 JURECA-DC MI250 Epoch Time: Comparison of time for first epoch and average time for an epoch (top); ratio of both quantities (bottom). ap3-jrcmi250-epoch-time

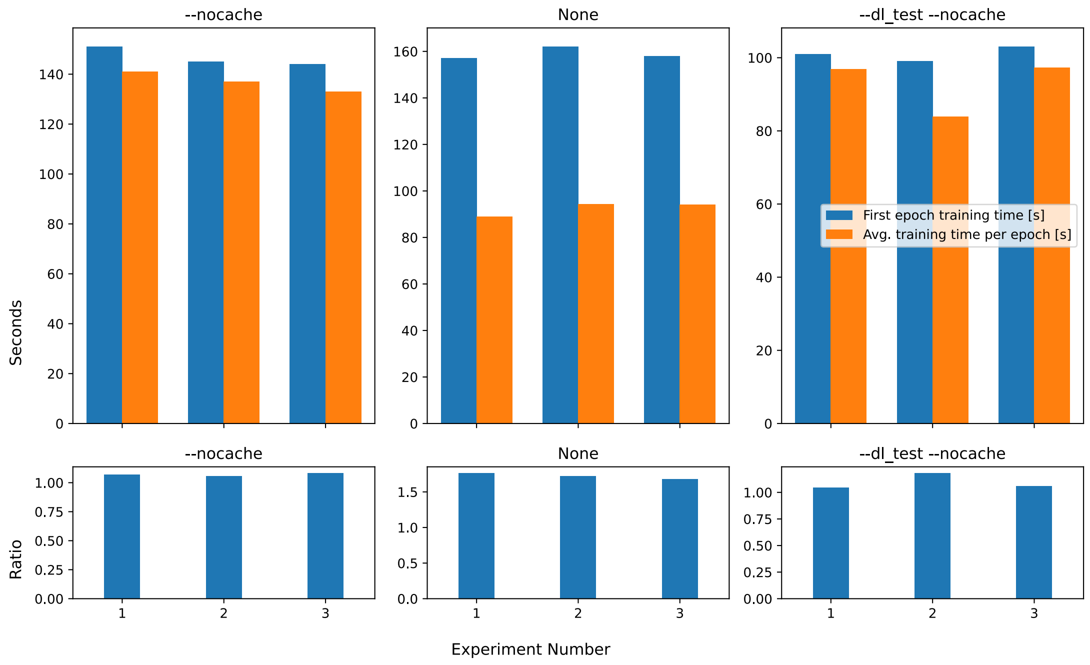


Figure 37: AP3 JURECA-DC MI250 Epoch Time: Comparison of time for first epoch and average time for an epoch (top); ratio of both quantities (bottom) (Mi200 8GPUS). ap3-8jrcmi250-epoch-time

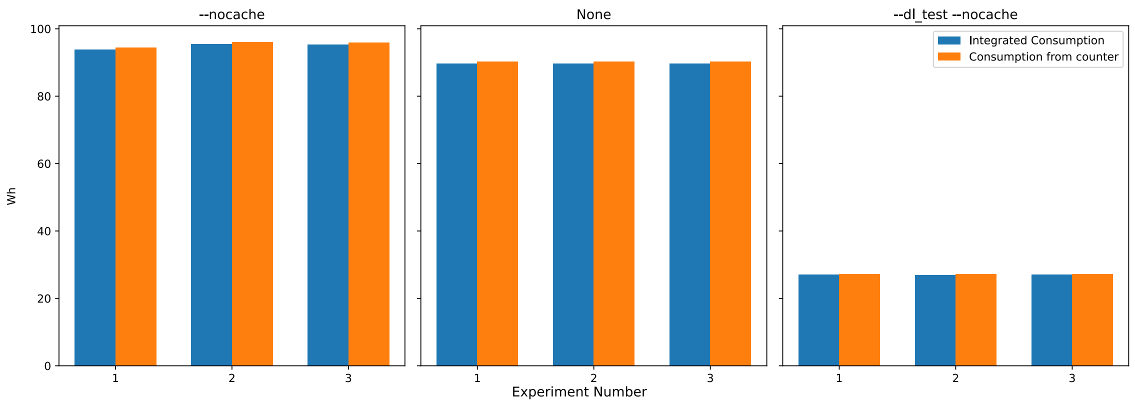


Figure 38: AP3 JURECA-DC MI250 Energy: Total GPU energy consumption during the training phase. ap3-jrcmi250-energy

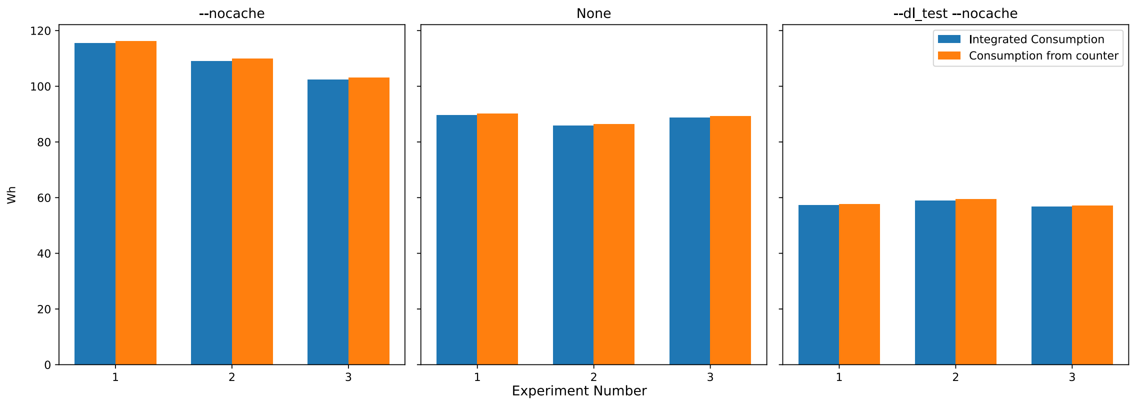


Figure 39: AP3 JURECA-DC MI250 Energy: Total GPU energy consumption during the training phase (Experiments using 8 GPUs). ap3-8jrcmi250-energy

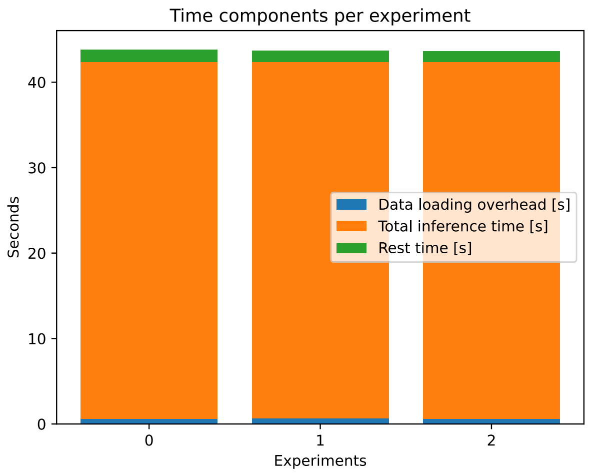


Figure 40: AP3 JURECA-DC MI250 Inference Runtime: Runtime and relative share for multiple experiments during the inference phase ap3-jrcmi250-inf-runtime-share

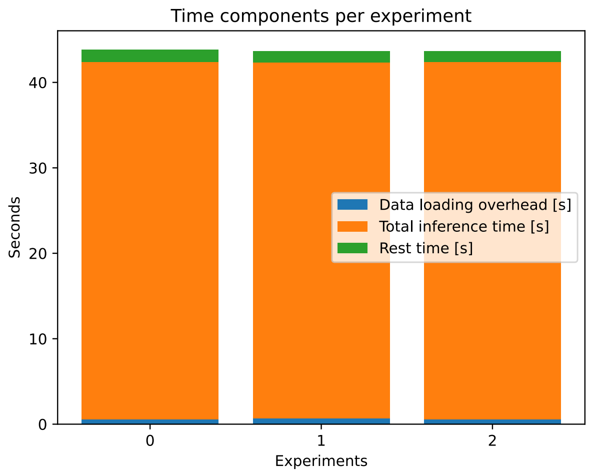


Figure 41: AP3 JURECA-DC MI250 Inference Runtime: Runtime and relative share for multiple experiments during the inference phase (Experiments using 8 GPUs) *ap3-8jrcmi250-inf-runtime-share*

4.4.5 E4 Intel System - NVIDIA A2

On the A2 GPU, 12 experiments have been conducted, divided into four triplets with different flags and configurations. The number of GPUs and MPI tasks used in these experiments was set to 1. The configuration for each triplet is as follows:

- Triplet 1: `--nocache` flag and the default (NFS) filesystem.
- Triplet 2: no flag and the default (NFS) filesystem.
- Triplet 3: `--dl_test --nocache` flag and the default (NFS) filesystem.

The runtime of the inference phase has also been reported, with three experiments performed using this system.

4.4.5.1 Training

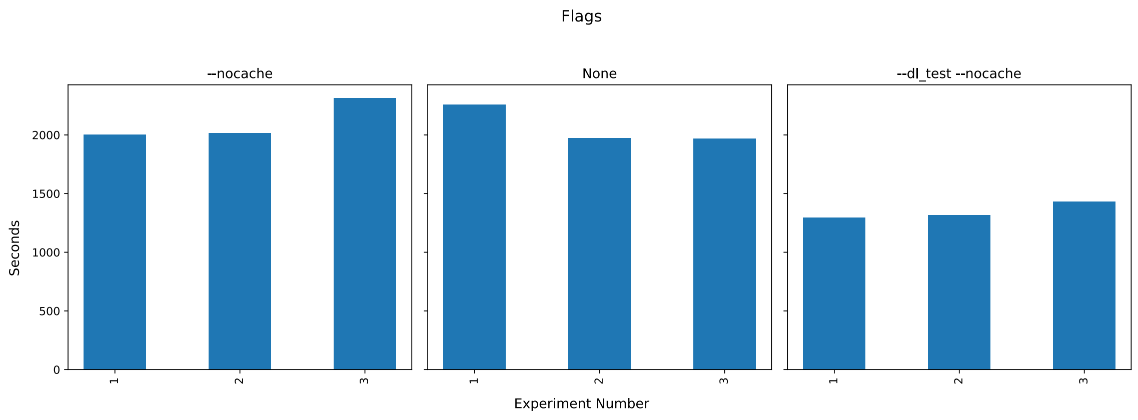


Figure 42: AP3 E4 A2 Runtime: Runtime for multiple experiments during the training phase. *ap3-a2-runtime-share*

4.4.5.2 Inference

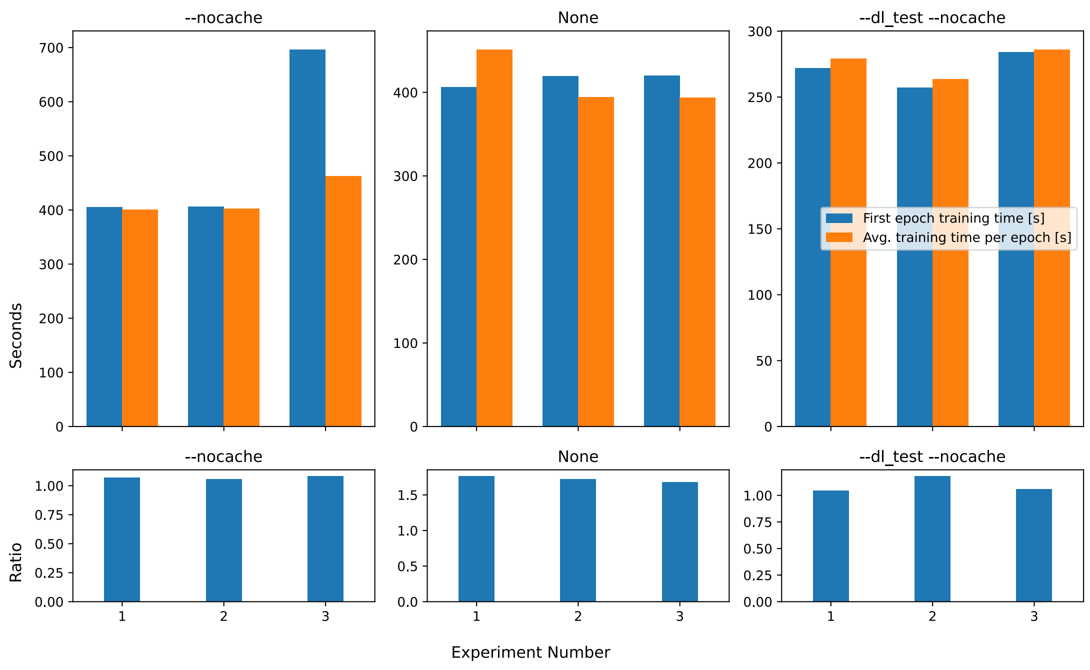


Figure 43: AP3 E4 A2 Epoch Time: Comparison of time for first epoch and average time for an epoch (top); ratio of both quantities (bottom). ap3-a2-epoch-time

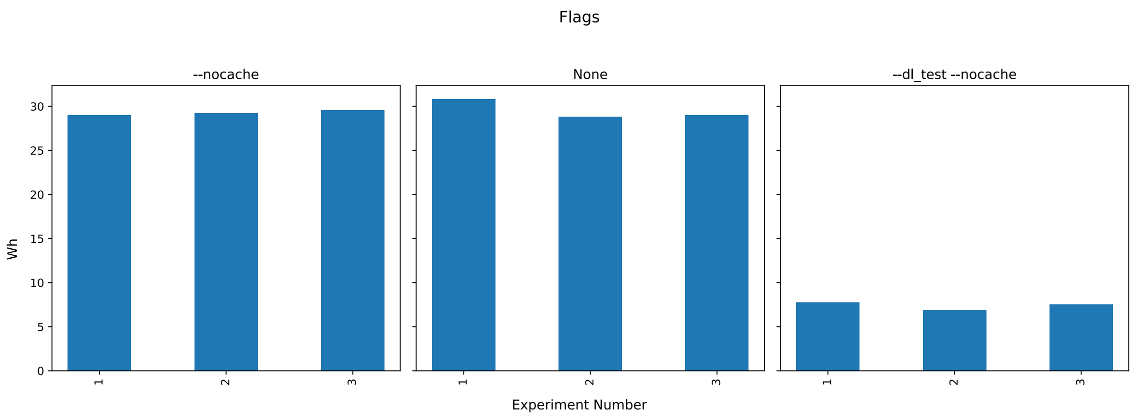


Figure 44: AP3 E4 A2 Energy: Total GPU energy consumption during the training phase. ap3-a2-energy

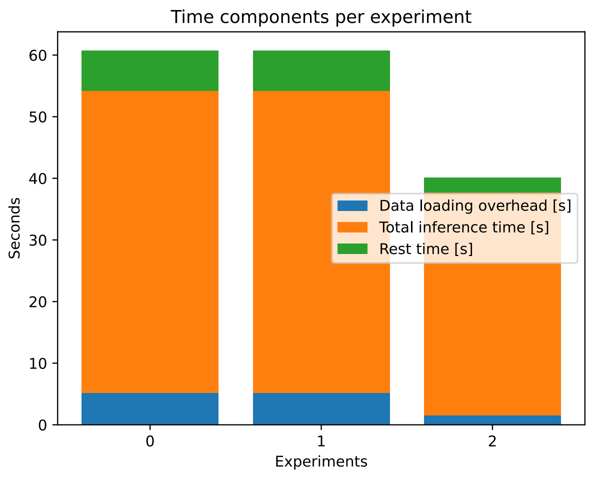


Figure 45: AP3 E4 A2 Inference Runtime: Runtime and relative share for multiple experiments during the inference phase ap3-a2-inf-runtime-share

4.5 Non-IO Experiments

4.5.0.1 Memory bandwidth

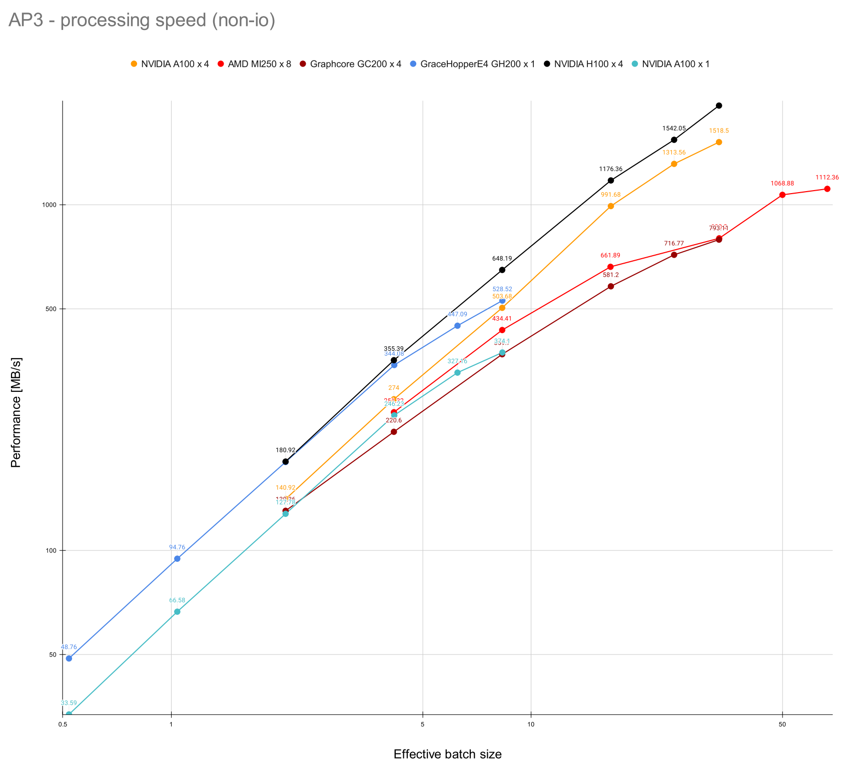


Figure 46: AP3 Non-IO Throughput: Peformance in MB/s for the non-io experiments across different hardware. ap3-nonio-performance

4.5.0.2 Energy-to-solution

The most energy-efficient configuration is 1xGH200, followed by 4xH100. For GPUs we see the energy-to-solution decrease with increasing batch size. The IPU shows different behaviour with the graph forming a valley - the lowest value achieved by a different batch size depending on the level of parallelism - increasing the batch size further leads to a decrease in performance. All GPUs at their best configuration beat the best IPU result.

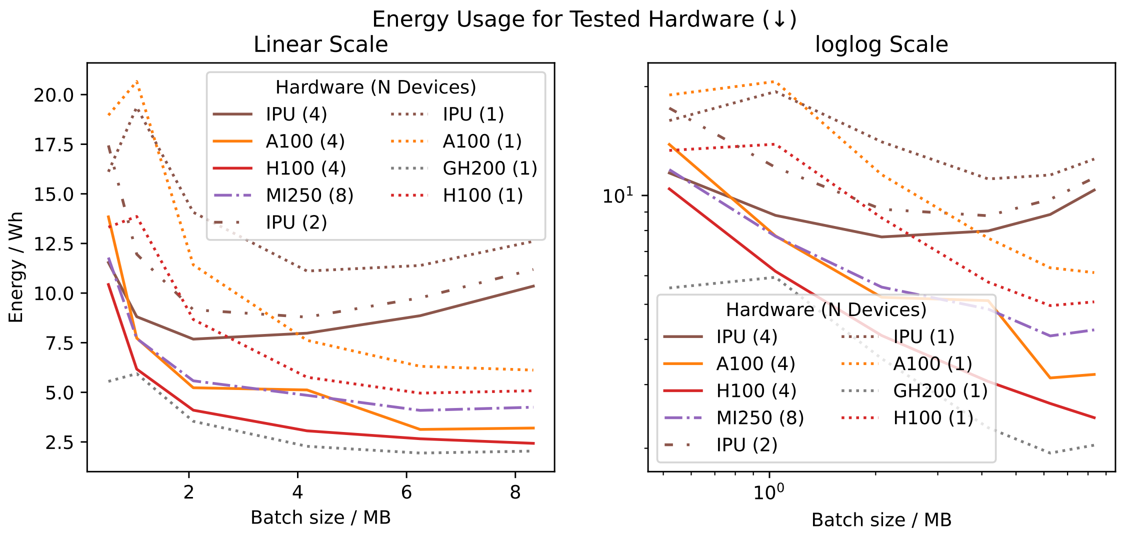


Figure 47: AP5 Energy: Energy-to-Solution on different hardware as a function of batch size. ap3-energy-vs-hardware-graph

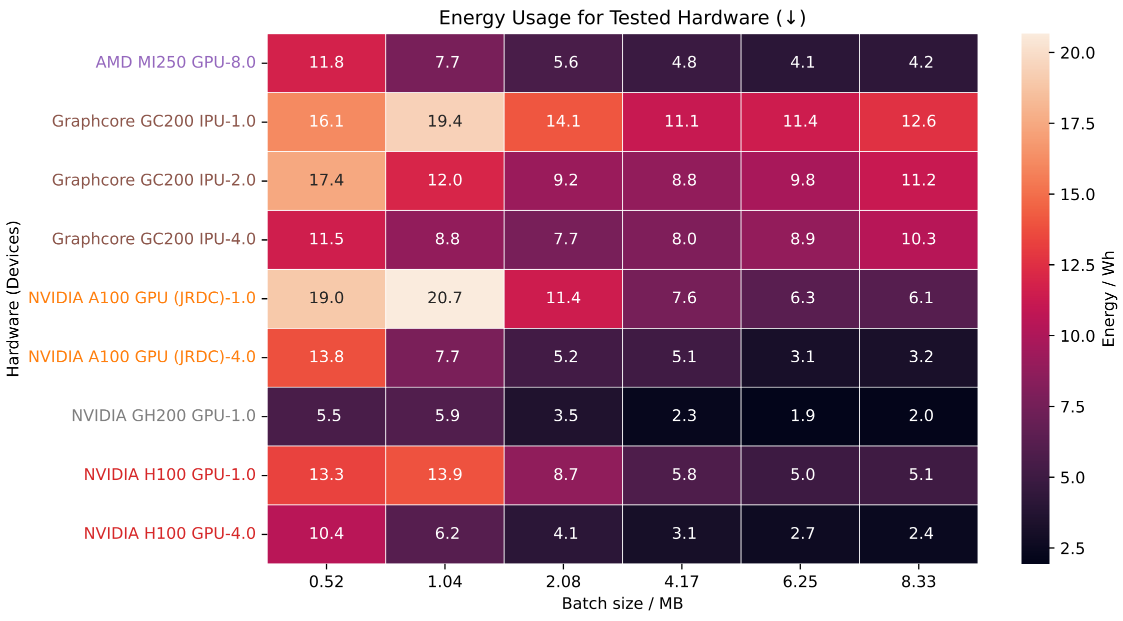


Figure 48: AP5 Energy: Energy-to-Solution on different hardware as a function of batch size. ap3-energy-vs-hardware-heatmap

4.5.0.3 Time-to-solution

The 4xH100 configuration delivers the fastest runtime, followed closely by 4xA100 and 8xMI250 (4 cards). The GPUs and IPU show similar behaviour to the Energy-to-solution results, with the GPUs runtime decreasing with higher batchsize and IPU runtime forming a valley. The IPU is again outperformed by all GPUs.

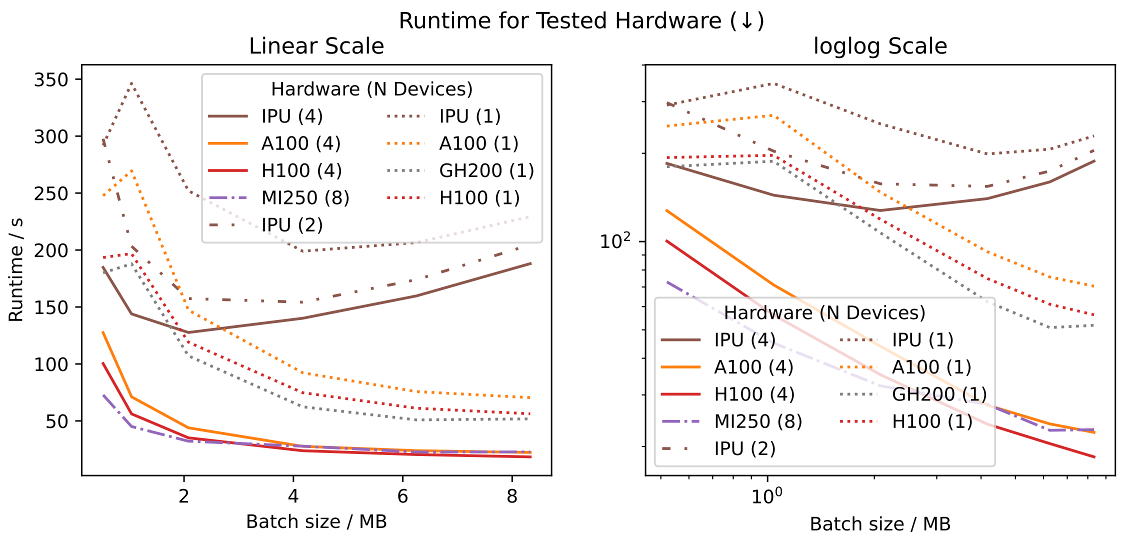


Figure 49: AP5 Time: Time-to-Solution on different hardware as a function of batch size. ap3-runtime-vs-hardware-graph

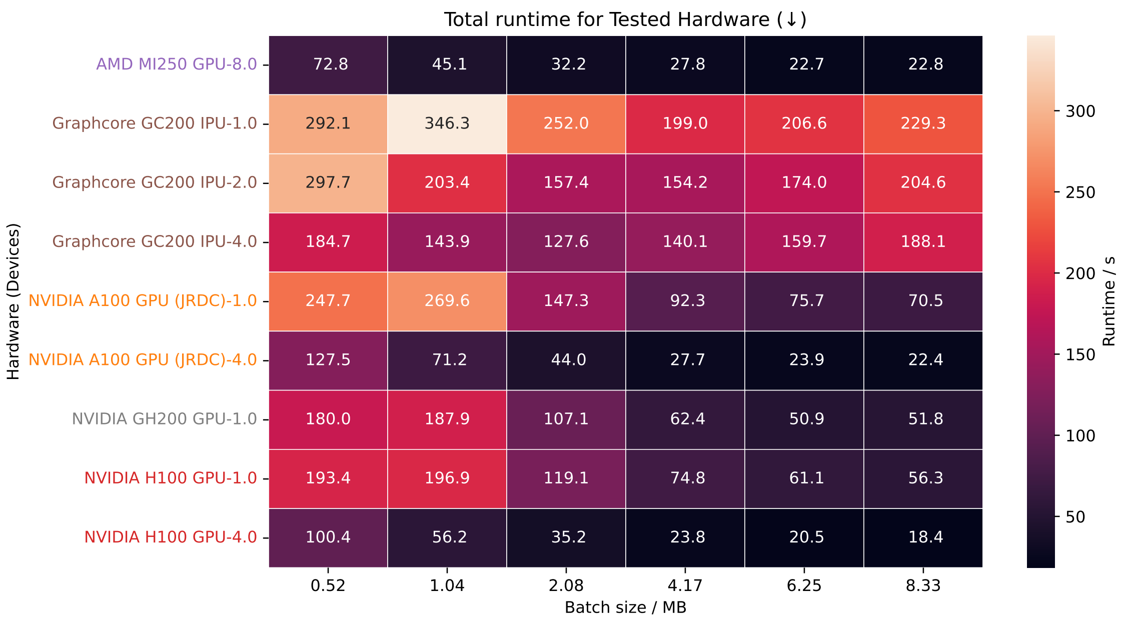


Figure 50: AP5 Time: Time-to-Solution on different hardware as a function of batch size. ap3-runtime-vs-hardware-heatmap

4.6 Node energy measurements

For AP3, node energy consumption data were gathered for the NVIDIA A2 system during both the Training and Inference phases, whereas consumption for the GH200 system was measured during the Non-IO experiments.

4.6.0.1 Training

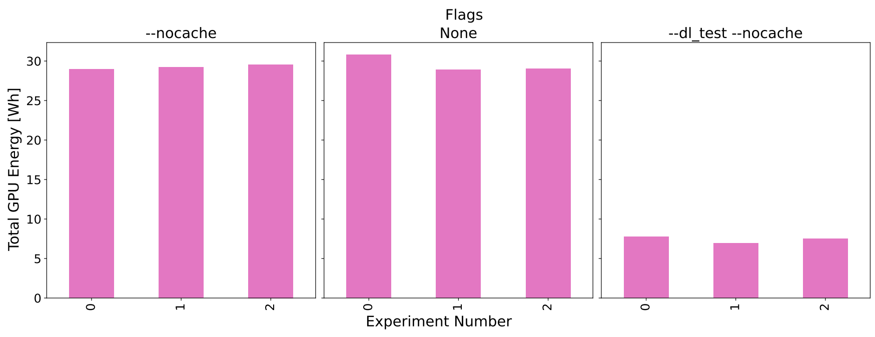
Upon analyzing Fig. 51, we observe that the NVIDIA A2 GPU energy consumption varies across experiments, ranging from 6.91 Wh (observed with the –dl_test –nochache configuration) to 30.80 Wh (associated with the None flag). Notably, experiments conducted with the –nocache flag exhibit energy consumption levels similar to those with the None configuration.
Similarly, the examination of average node energy consumption reveals a parallel trend. Values range from 127.32 Wh to 252.44 Wh, with experiments employing the –dl_test –nochache setup demonstrating the lowest node energy consumption, mirroring the observed behavior in GPU energy consumption.

4.6.0.2 Inference

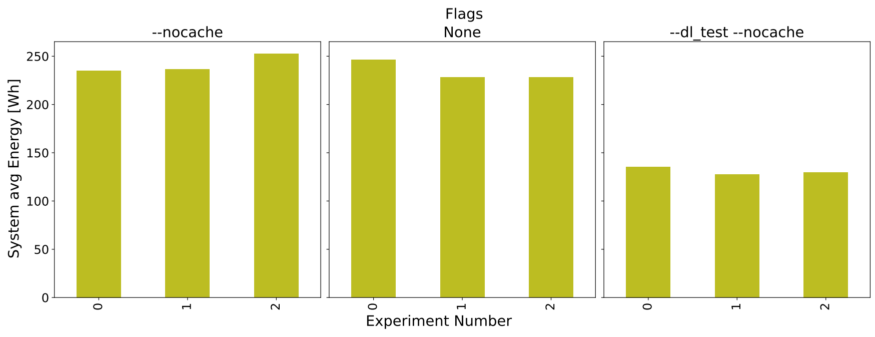
In examining the inference phase, as depicted in Figure 52, experiments 0 and 1 showcase notably similar and comparable values for both total GPU energy and average node energy consumption. Conversely, Experiment 2 exhibits the lowest total energy consumption for both the GPU and the node, with values of 0.37 Wh and 3.96 Wh, respectively. Experiments 0 and 1 display slightly higher consumption, with the GPU consuming 0.46 Wh and the node recording a consumption of 5.33 Wh.

4.6.0.3 Non-IO Experiments

Fig. 53 presents insights into energy consumption patterns across varying batch sizes during computational processes. As already pointed out in the previous section regarding the Energy-to-Solution, one prominent trend is the inverse relationship between batch size and energy efficiency, where smaller batches tend to incur higher energy consumption per unit of work processed.
As the batch size decreases, energy consumption increases, as evidenced by both the GPU and node energy consumption. This trend suggests that smaller batches demand more energy for processing, likely due to increased computational overhead associated with handling smaller data sets.



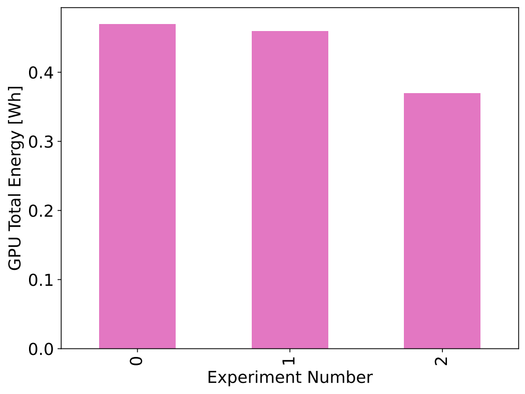
(a) GPU energy consumption



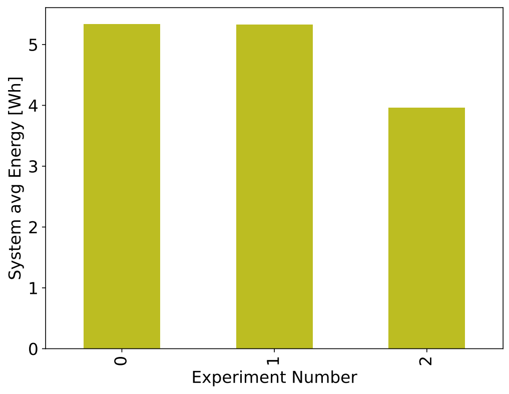
(b) Node energy consumption

Figure 51: AP3 Node and GPU Energy comparison during the training phase: Comparison between GPU and full-node energy consumption for A2 system considering multiple experiments. ap3-tr-nodeE-vs-GPU

For instance, the smallest batch size of 0.52 MB exhibits the highest energy consumption, with an average node energy of 15.14 Wh. Conversely, the largest batch size of 8.33 MB shows the lowest energy consumption, with an average node energy of 4.87 Wh. This highlights the trade-off between batch size and energy efficiency, where smaller batches offer finer granularity but require more energy for processing.

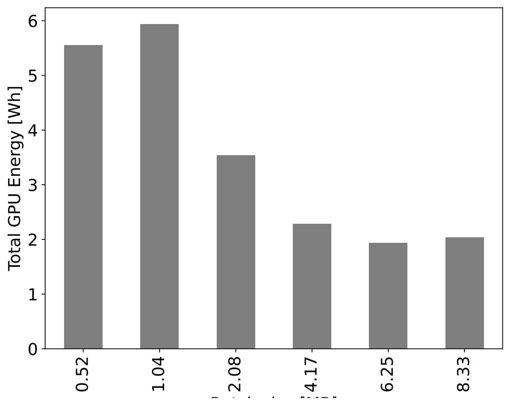


(a) GPU energy consumption

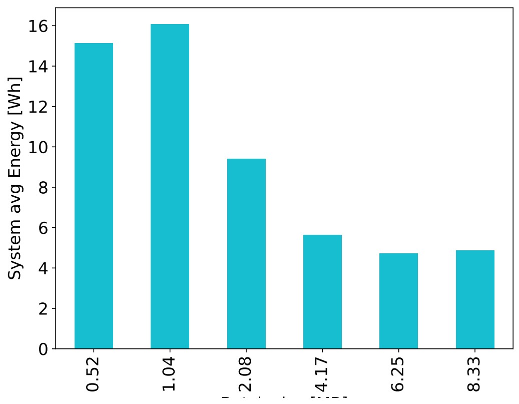


(b) Node energy consumption

Figure 52: AP3 Node and GPU Energy comparison during the inference phase: Comparison between GPU and full-node energy consumption for A2 system considering multiple experiments. ap3-inf-nodeE-vs-GPU



(a) GPU energy consumption



(b) Node energy consumption

Figure 53: AP3 Node and GPU Energy comparison for Non-IO experiments: Comparison between GPU and full-node energy consumption for GH200 system by changing the Batch Size [MB]. ap3-noio-nodeE-vs-GPU

4.7 AP 4

4.7.1 Notes

The application is showing the following details:

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
64 GB	2.5 GB	1889	[14, 361, 720]	1

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
633698	0	CRPS	None

Data formats	Frameworks (to be) used
NetCDF (.nc)	PyTorch 1.11

Application 4 aims to improve the precision and effectiveness of weather forecasts by utilizing deep neural networks to process the ensemble outputs of numerical weather prediction systems. This is achieved through the use of the ENS-10[5] dataset, which contains ten ensemble members spanning 20 years from 1998 to 2017. The UNet model is used to predict geopotential at 500 hPa, represented by Z500.
To train the model, we used the entire ENS-10 dataset at 500hPa and used the ERA-5 dataset as the ground truth. The model was trained for three epochs with a batch size of one, and the Adam optimizer was used in all our experiments. In addition, we utilized the NetCDF data format and implemented a PyTorch dataloader to efficiently process the data for the model.
The underlying data of illustrations in this section can be found in appendix 6.4.

[5] `https://arxiv.org/abs/2206.14786`

4.7.2 Runtime split

Loading times on all GPUs are between 1.5% to 2.5% of the total runtime. Time spent on neither IO nor training is between 0.6% and 1.6% of the total

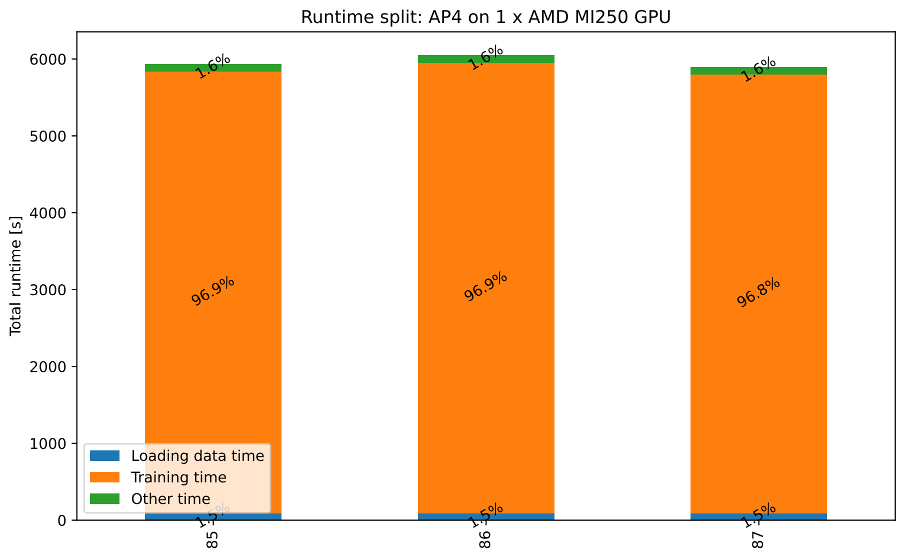


Figure 54: AP4 JURECA-DC MI250: Percentages of runtime spent for training and loading data. ap4-mi250-runtime-split-graph

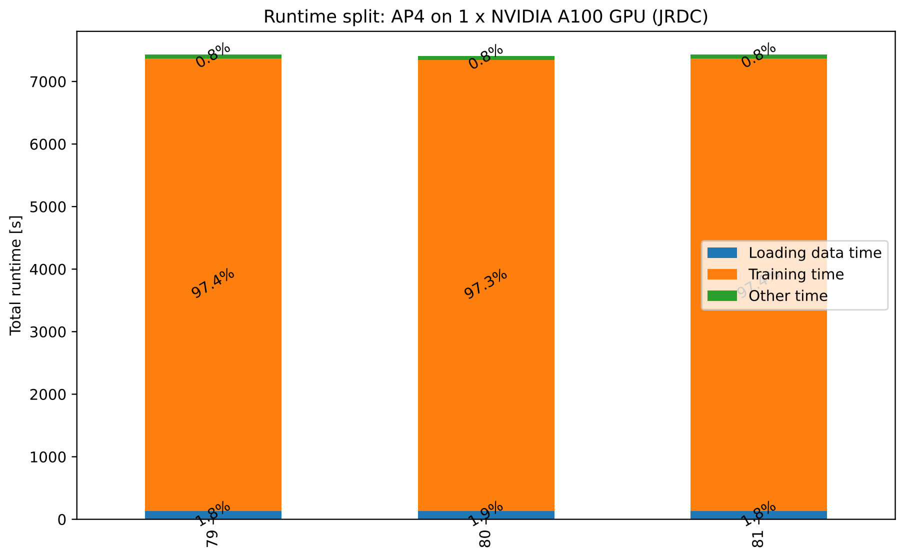


Figure 55: AP4 JURECA-DC A100: Percentages of runtime spent for training and loading data. ap4-a100jrdc-runtime-split-graph

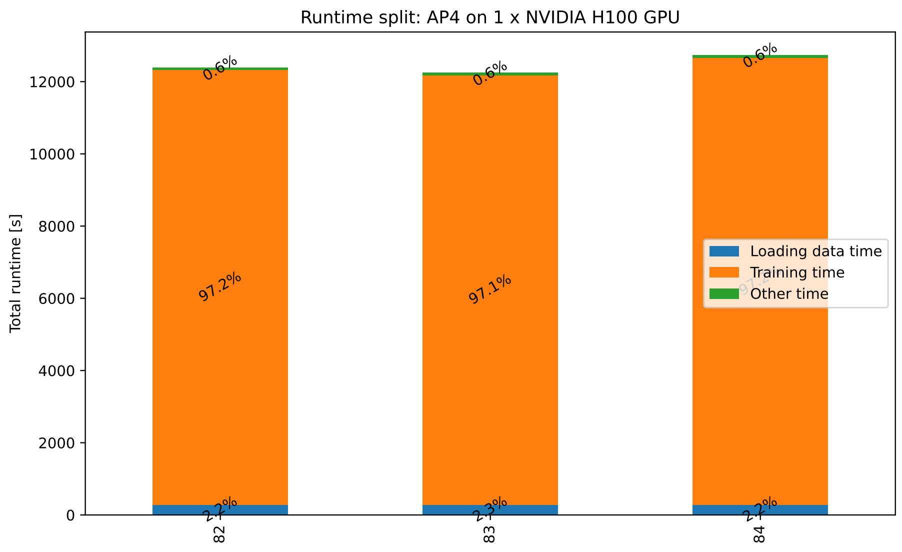


Figure 56: AP4 JURECA-DC H100: Percentages of runtime spent for training and loading data. *ap4-h100-runtime-split-graph*

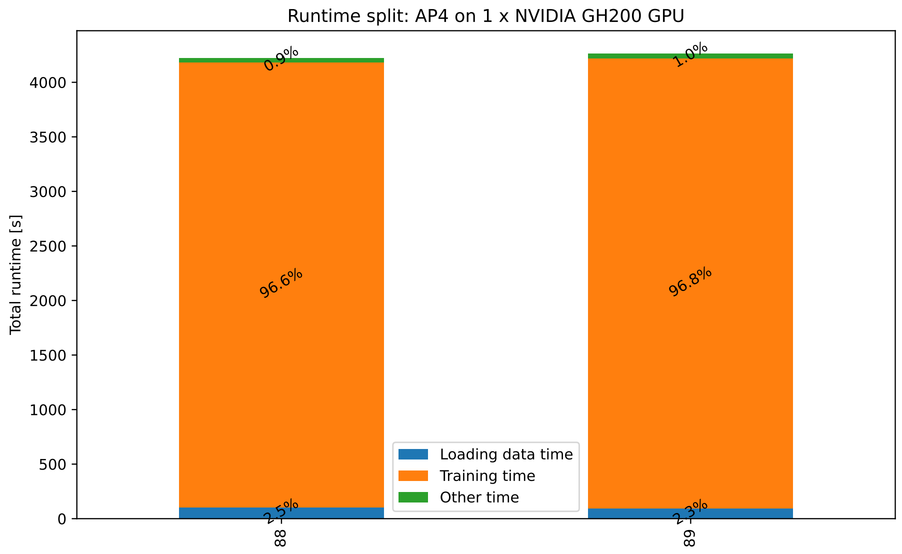


Figure 57: AP4 E4 GH200: Percentages of runtime spent for training and loading data. *ap4-gh200-runtime-split-graph*

4.7.2.1 Node Energy Consumption

In the case of the GH200 node, experiments show total execution times between 4220.0 and 4260.0 seconds. For this duration, the average power consumption (Fig. 59a) of the integrated GPU is around 122.345 Wh. In contrast, experiments conducted on the NVIDIA A2 node reveal significantly longer total execution times,

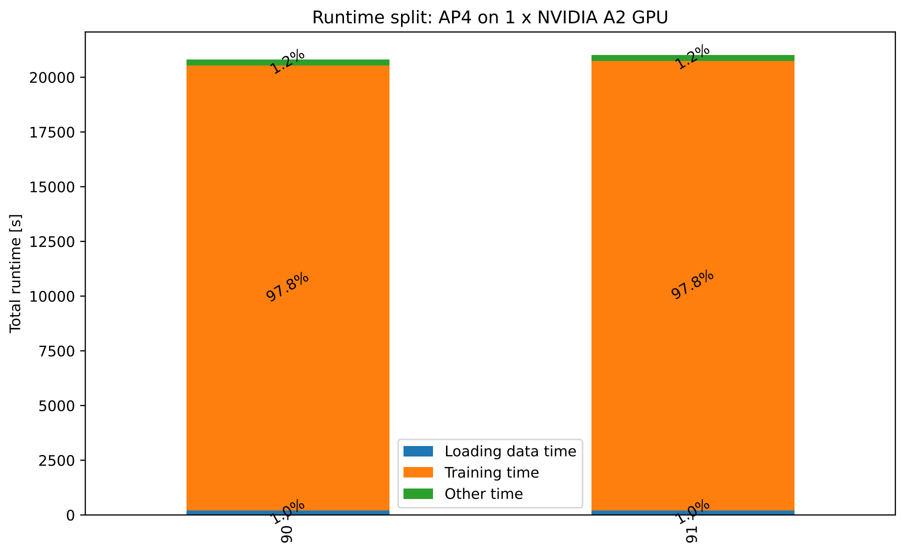
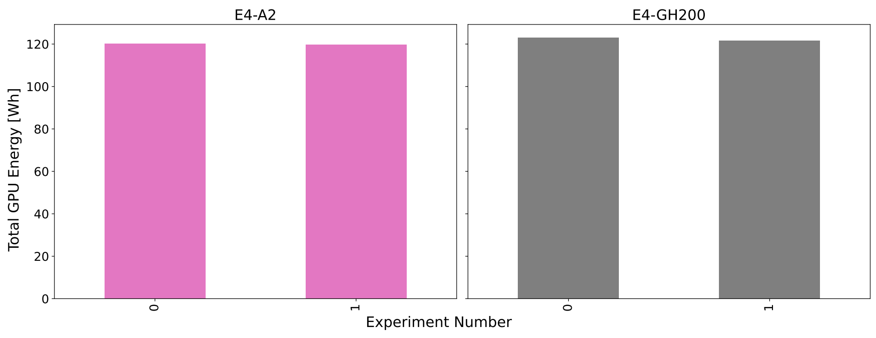


Figure 58: AP4 E4 A2: Percentages of runtime spent for training and loading data. ap4-a2-runtime-split-graph

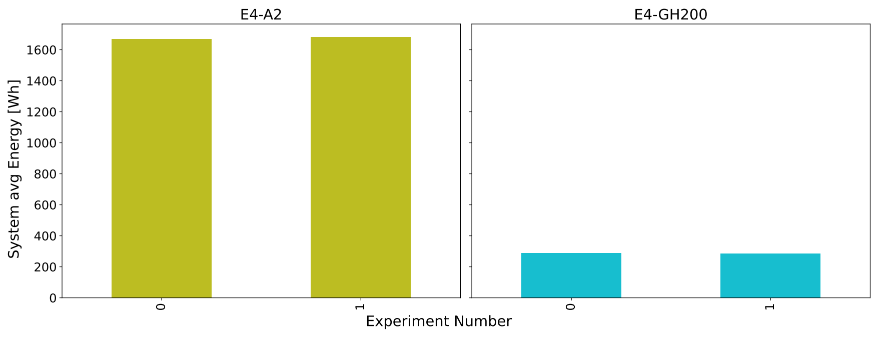
ranging between 20807.0 and 21018.0 seconds. Despite this discrepancy, the average power consumption of the integrated GPU remains comparable to that of the GH200 node, at approximately 120.05 Wh.

The average system power consumption (Fig. 59b) of the GH200 node remains relatively stable at around 286.958 Wh. These results suggest a consistent performance profile for this system, with a significant dependency on GPU resources for computing tasks. As for the NVIDIA A2 node, the most noticeable difference from the previous system concerns the node's average power consumption, which averages around 1676.182 Wh. This substantial difference underlines the considerable energy consumption associated with the operation of the NVIDIA A2 node.

Examining the Action metric shown in Table 4, the GH200 node shows a lower score, averaging about 4380.11 MJs. This suggests a more efficient use of computational resources during the training process than A2, which reports a significantly higher action score of approximately 126193.56 MJs. Once again, GH200 outperforms NVIDIA A2, demonstrating shorter execution times and more efficient use of energy.



(a) GPU energy consumption



(b) Node energy consumption

Figure 59: AP4 Node and GPU Energy comparison: Comparison between GPU and full-node energy consumption for A2 and GH200 systems. ap4-nodeE-vs-GPU

Experiment	**Hardware**	**Action [MJs]**
0	E4-A2	125117.13
1	E4-A2	127269.99
2	E4-GH200	4363.06
3	E4-GH200	4397.16

Table 4: AP4 Action metric for A2 and GH200 systems.

4.8 AP 5

4.8.1 Notes

Data formats	Frameworks (to be) used
NetCDF	Tensorflow v2.6.0 with Keras API

4.8.1.1 Changes with respect to D3.6:

Since the last deliverable, the WGAN architecture for downscaling has been tuned. In particular, the activation function in all convolutional layers of the U-Net generator and the critic model has been changed from ReLu to Swish. Furthermore, the bilinear upsampling in the decoder-part of the U-Net has been replaced by a subpixel-layer (see `https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Shi_Real-Time_Single_Image_CVPR_2016_paper.html`). These adaptions have been found to provide better results in the downscaling product (lower RMSE of the downscaled 2m temperature field). The increased number of trainable parameters (from about 5M to 9M) should thereby largely explain the increased time per epoch (from about 575s to 865s on a single A100 GPU), even though the training on the Jureca's A100 was found to be 40-45s/epoch slower compared to Juwels Booster (see e-mail from 09th Feb).
Finally, the data pipeline has been revised to allow data distributed training.

4.8.1.2 Experimental set-up:

Two series of experiments have been run, that are a series of idealized experiments without I/O (non I/O-experiments) and a series of experiments with I/O (i.e. real training by reading data from the netCDF-files instead of creating random synthetic data on-the-fly). The former was done to investigate the upper bound of the performance on the different systems. The idealized experiments were also complemented by testing different batch sizes to figure out the performance's sensitivity on it. The real-case experiments can therefore be compared to the idealized one to a) distill I/O-bottlenecks and b) to consider changes to the batch size in the future to boost the computational performance.

4.8.1.3 Results:

The non I/O-experiments show that increasing the batch size from 8 to 256 can boost the computational performance by 25-40% (25% on the A100, 31% on the

MI250 and 42.5% on the GH200). On the A2-nodes, the varying the batch size has the smallest effect, and an out-of-memory error has been noted for a mini-batch size of 256. However, it is noted that the largest performance gains are attained when increasing from small mini-batch sizes (e.g. from 8 to 16 or 16 to 32). For large mini-batch sizes, the performance gets saturated, especially on the MI250x, where training with a mini-batch size of 256 is indeed slower than with a mini-batch size of 32 (4295s vs. 4421s). Since the default batch size for real-case applications is 32, no big performance gains are expected for varying the batch size. The real-case experiments show that the performance on a single GPU is nearly optimal. For instance, one epoch takes about 865s with a single A100-node for the real-case test, which is in-line with the non I/O-experiments. Only the first epoch seems to require a bit more time (930s vs. < 900s). For data-distributed training however, I/O-bottlenecks are apparent. While the non I/O-experiments scale fairly well (e.g. 4277s for a single MI250x GPU vs. 581s on eight MI250x GPUs -> speed-up factor of about 7.4), the scaling factors are smaller for the real-case experiments (A100: 2.6 when using 4 instead of 1 GPU(-s), MI250x: 3.42 when using 8 instead of 1 GPU(-s)). Thus, further work will be dedicated to increase the efficiency of the data pipeline.

4.8.1.4 Other notes:

No experiments have been conducted on the IPU, since the Keras extensions for the IPU do not support overwriting the `train_step`-method (see Subsection 19.4.3 in `https://docs.graphcore.ai/projects/tensorflow-user-guide/en/latest/keras/keras.html#model-subclass`). This is, however, required for a composite model such as the WGAN, where the critic and the generator are updated asynchronously (in general, the critic is updated 5x before the generator gets updated once).
The computing nodes on the E4-cluster do not support correct distributed training with Horovod yet. Either, Horovod fails to detect all allocated GPUs or it assigns rank 0 to all workers. Since both cases make experimenting meaningless, no real-case experiments beyond training on a single GPU have been conducted. The issue has been reported to the support team, but a solution for this is still pending.
Furthermore, an update of CUDA-driver on the H100 at JSC has resulted into a major performance degradation. Before the update, one epoch on a single H100-GPU was processed in about 580s. After the update, the epoch time is about three times larger (1500s). Changing the optimizer to `keras.optimizer.legacy.Adam`, the degradation can be reduced, but still remains significant with about 700s/epoch. As you already know, the issue is still under investigation.

Finally, a memory leak has been discovered with newer xarray versions (>=2023.0.1). The initial containers for the H100- and MI250x-nodes have been using a newer xarray-version, resulting in memory accumulation during training and ultimately in out-of-memory errors with >=4GPUs. Re-building the containers with xarray 0.20.1 (as provided from the module stack), solved the issue on MI250x, but produced follow-up issues on the H100 with the pynvml-package. Due to this, the results of the real case experiments are retained for the container with xarray 2023.0.1 incl. energy measurements, whereas no energy measurements have been possible for the non I/O-experiments using the container with xarray 0.20.1. For both set-ups, no experiments utilizing 4 H100 GPUs have been possible.
The real-case experiments (full data pipeline) have the following properties:

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
69.43 GB	6.31 GB	96 296	[96, 120, 15]	32/192

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
9 332 299	5280	Earth Mover distance, gradient penalty and L1	Training for 4 epochs

The non-I/O experiments have the following properties:

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
62.21 GB	N/A	96 296	[96, 120, 15]	Variable

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
9 332 299	5280	Earth Mover distance, gradient penalty and L1	Training for 3 epochs

4.8.2 Non-IO benchmarks

4.8.2.1 Energy-to-Solution

The GH200 is by far the most efficient with regard to energy-to-solution, a single GH200 beating all other configurations. All GPUs show positive scaling with increased batch size and number of GPUs.

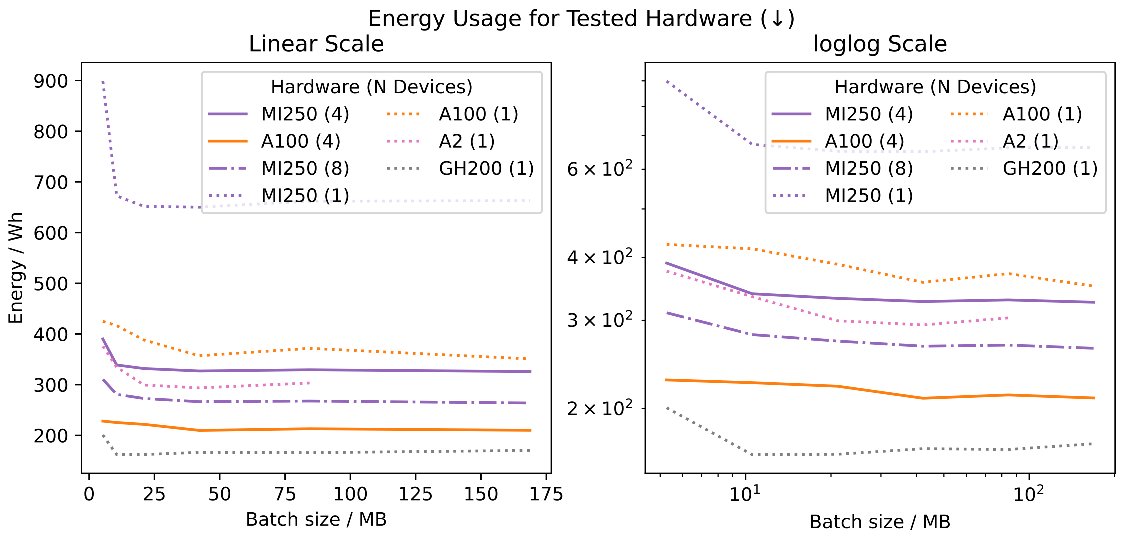


Figure 60: AP5 Energy: Energy-to-Solution on different hardware as a function of batch size. ap5-energy-vs-hardware-graph

4.8.2.2 Time-to-Solution

With the caveat of the previously degraded performance of the H100 which is under the investigation, as well as the 4xH100 runs failing, the 8xMI250 configuration beats the rest at the largest batch size with a runtime of 580.5 s. As with energy, decreased runtimes are seen across the Hardware spectrum and device parallelism with increasing batch size. The A2 is significantly slower than other GPUs which is also significant for energy when taking the host system consumption into account. A separate plot shows the runtime without the A2 in 64

4.8.2.3 Node Energy Consumption

Let us take a closer the energy consumption patterns for both GPUs and nodes in the GH200 and A2 systems (see Fig. 65).
On the GH200 system, training times for experiments vary within a range of approximately 1806.25 to 3023.54 seconds. Despite fluctuations in training duration,

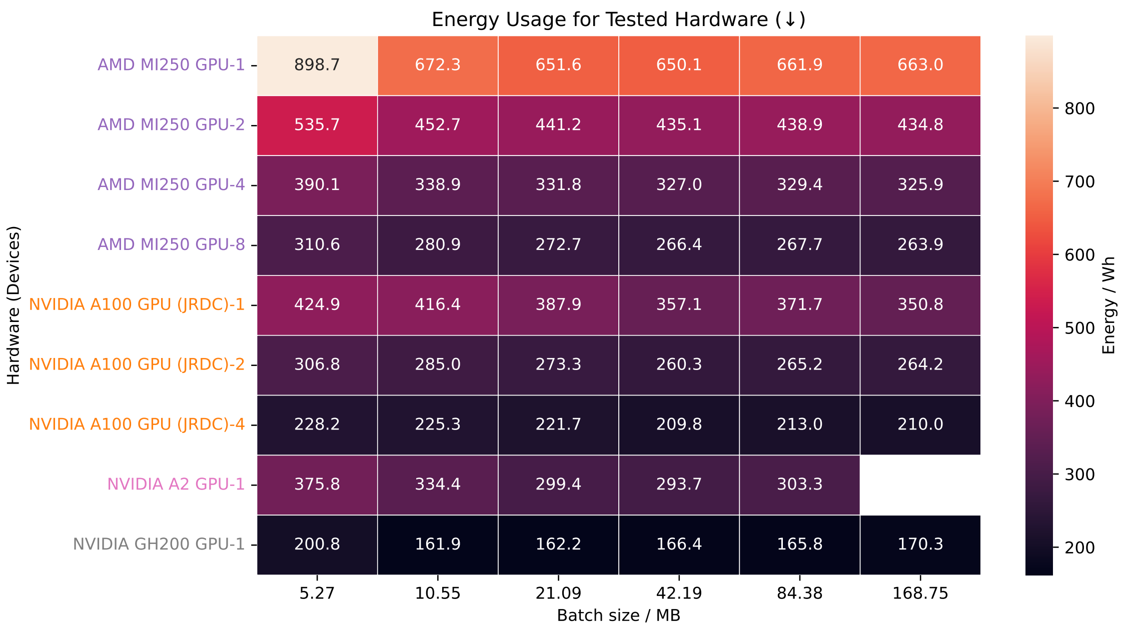


Figure 61: AP5 Energy: Energy-to-Solution on different hardware as a function of batch size. ap5-energy-vs-hardware-heatmap

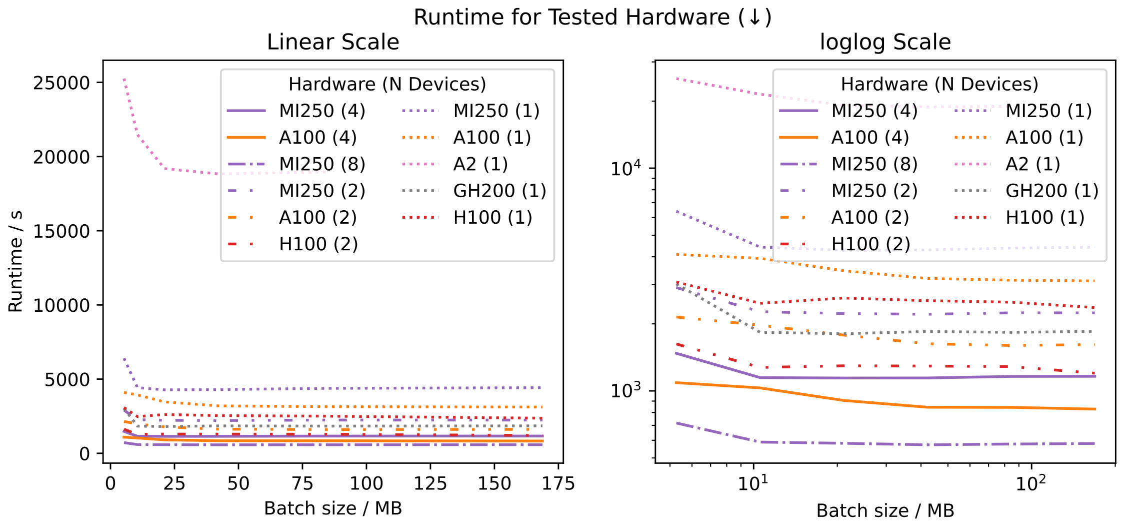


Figure 62: AP5 Time: Time-to-Solution on different hardware as a function of batch size. ap5-runtime-vs-hardware-graph

GPU energy consumption remains relatively stable, ranging from about 161.94 to 200.80 Wh. This suggests that while the training duration may vary, the overall energy demand for computational tasks on the GH200 GPU remains consistent across experiments. Examining the average node energy consumption reveals a similar

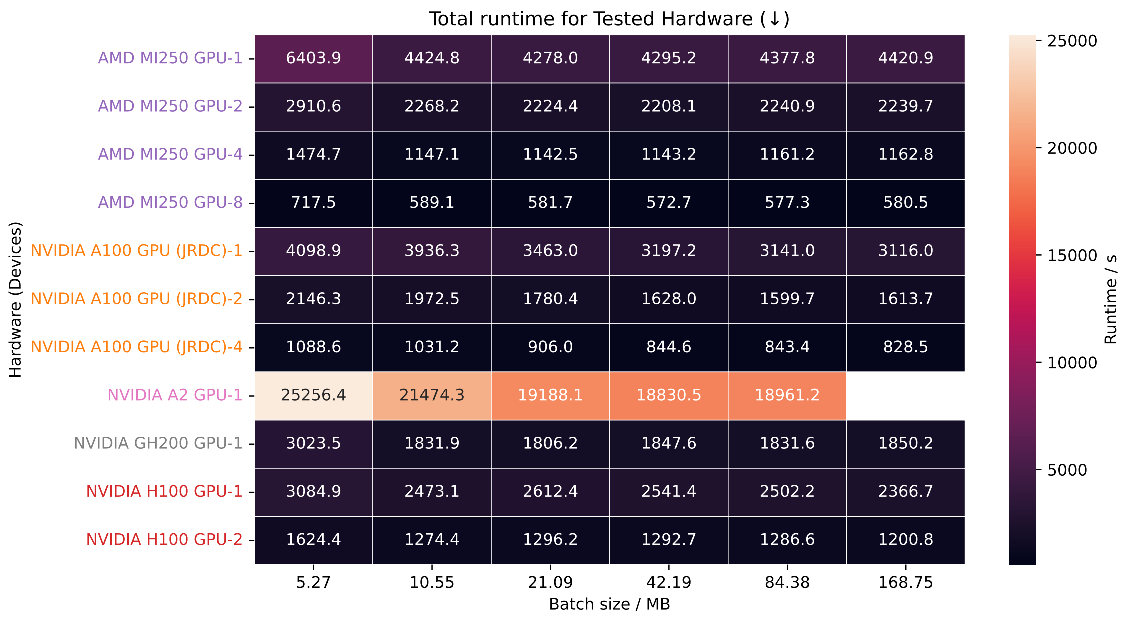


Figure 63: AP5 Time: Time-to-Solution on different hardware as a function of batch size. ap5-runtime-vs-hardware-heatmap

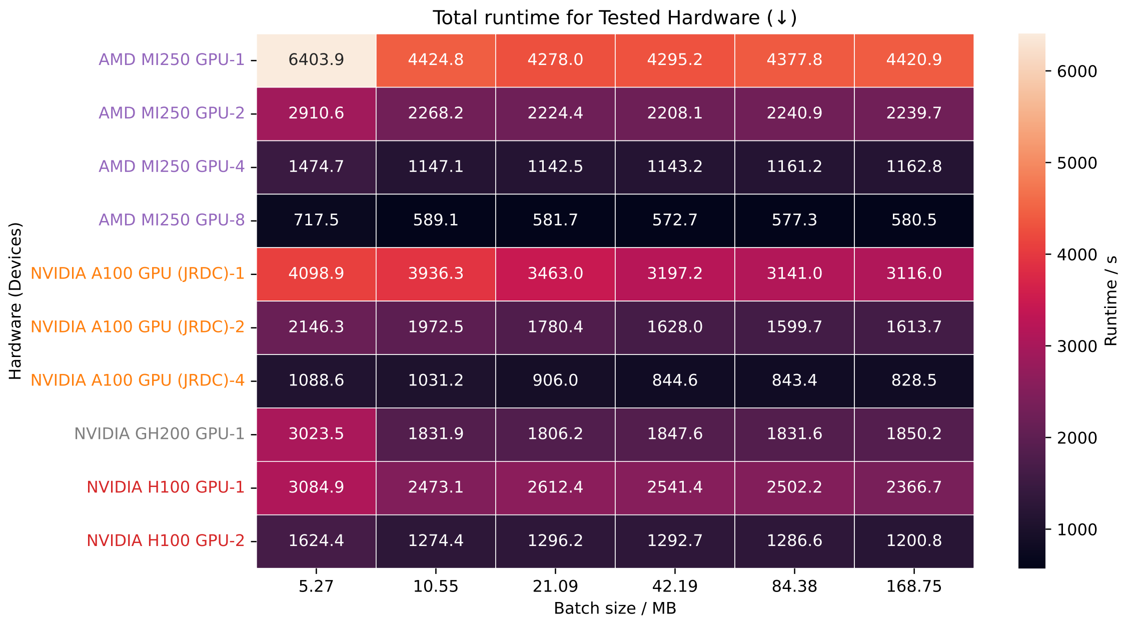
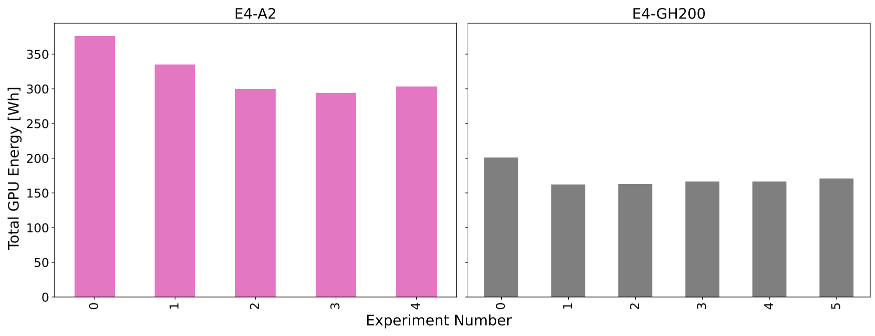


Figure 64: AP5 Time: Time-to-Solution on different hardware as a function of batch size (A2 excluded for better contrast). ap5-runtime-vs-hardware-noa2-heatmap

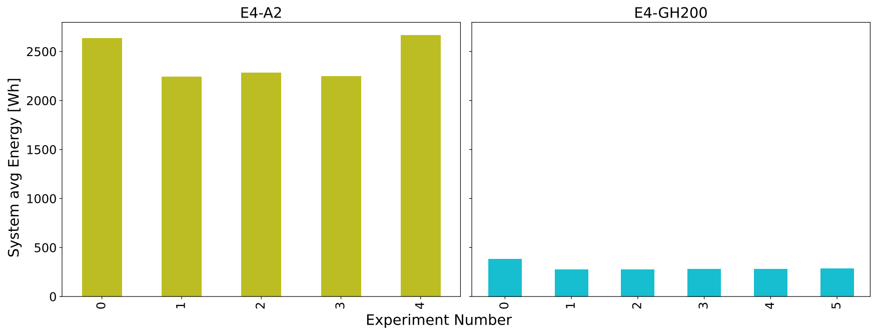
trend, with values ranging from approximately 274.89 to 384.86 Wh. The percentage of energy utilized by the GPU ranges from approximately 52.17% to 59.78%, indicating a substantial reliance on GPU acceleration for computational tasks and its significant contribution to the overall energy consumption.
The Action metric (Table 5) shows variability across experiments, ranging from approximately 1790.02 to 4189.14. This suggests differing levels of computational efficiency due to fluctuations, with Experiment 7 achieving the best score.
In contrast, experiments conducted on the E4-A2 hardware exhibit considerably longer training times, ranging from approximately 18830.54 to 25256.40 seconds. GPU energy consumption varies from approximately 293.67 to 375.77 Wh, indicating higher values compared to E4-GH200.
The average node energy consumption for E4-A2 is notably higher, ranging from approximately 2240.37 to 2664.37 Wh, indicating a higher overall energy usage for computational tasks on the NVIDIA A2 node. Moreover, the percentage of energy utilized by the GPU ranges from approximately 11.38% to 14.93%, indicating a lower reliance on GPU acceleration for computational tasks compared to E4-GH200.
The Action metric (in MJs) for E4-A2 ranges from approximately 152278.35 to 239577.13 MJs, suggesting varying levels of computational efficiency across experiments, with generally substantially lower efficiency compared to E4-GH200.

Experiment	**Hardware**	**Action [MJs]**
0	E4-A2	239577.13
1	E4-A2	173197.70
2	E4-A2	157549.59
3	E4-A2	152278.35
4	E4-A2	181870.57
5	E4-GH200	4189.14
6	E4-GH200	1812.86
7	E4-GH200	1790.02
8	E4-GH200	1869.91
9	E4-GH200	1847.89
10	E4-GH200	1897.22

Table 5: AP5 Action metric for A2 and GH200 systems.



(a) GPU energy consumption



(b) Node energy consumption

Figure 65: AP5 Node and GPU Energy for Non-IO Benchmarks: Comparison between full-node and GPU energy consumption for A2 and GH200 GPUs. ap5-nodeE-vs-GPU

4.8.3 Runtime split

Loading times on JSC systems are larger than on E4 systems due to a difference in filesystem. on MI250 we see proportions of 3.3% to 6.9%, on A100 between 3.3 and 3.7% and on H100 between 2.4% and 2.6%. Loading times couldn't be recorded for all experiments on E4 machines. Where they were recorded they were 1.0% 1.7% of the total for GH200 and around 0.2% on A2.

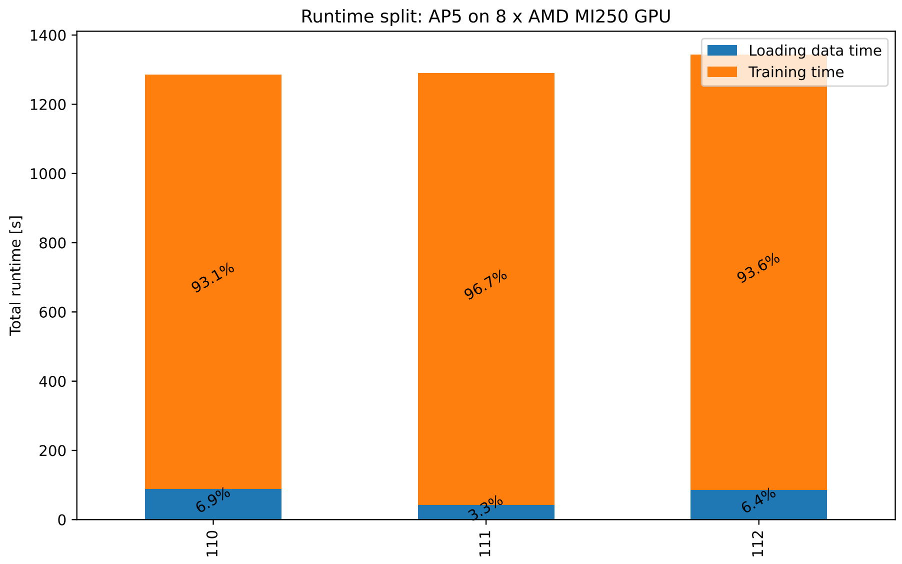


Figure 66: AP5 JURECA-DC MI250: Percentages of runtime spent for training and loading data. ap5-mi250-runtime-split-graph

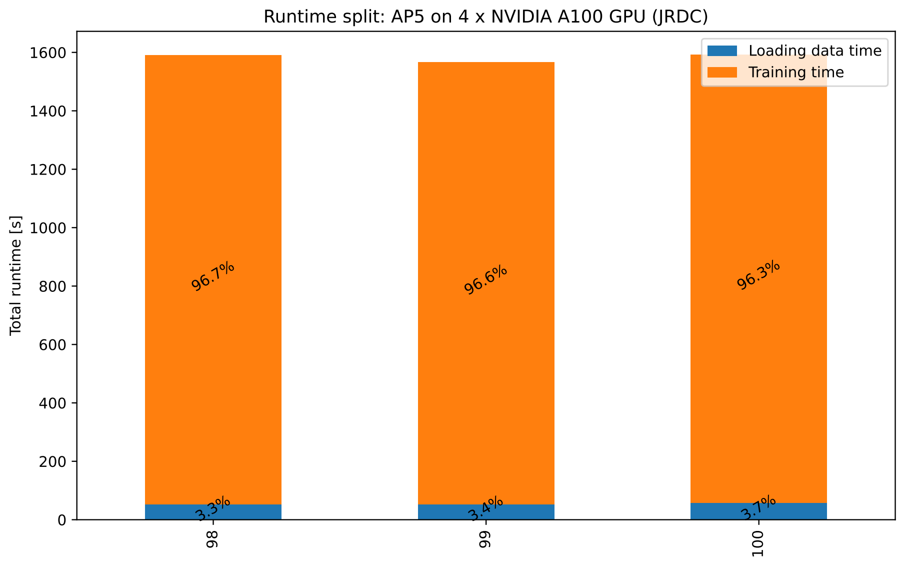


Figure 67: AP5 JURECA-DC A100: Percentages of runtime spent for training and loading data. ap5-a100jrdc-runtime-split-graph

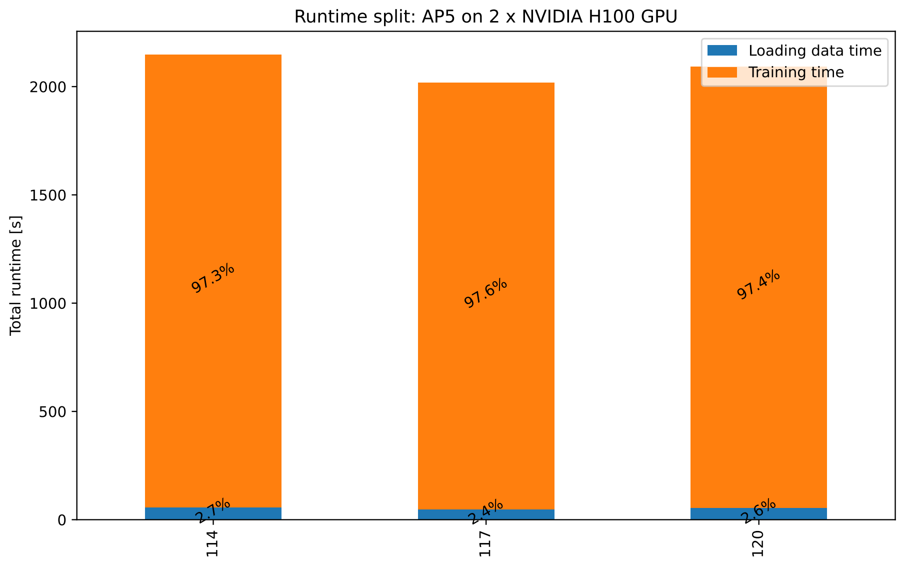


Figure 68: AP5 JURECA-DC H100: Percentages of runtime spent for training and loading data. ap5-h100-runtime-split-graph

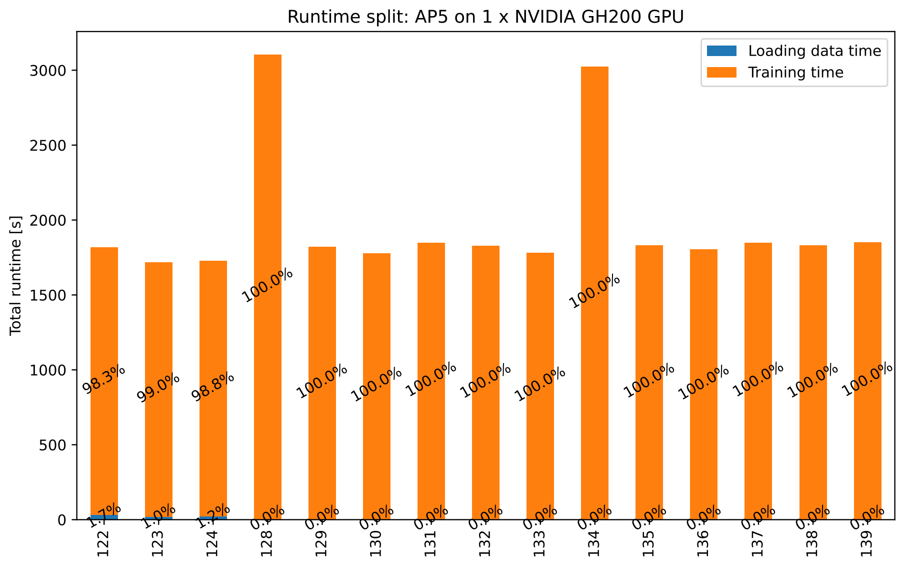


Figure 69: AP5 E4 GH200: Percentages of runtime spent for training and loading data. ap5-gh200-runtime-split-graph

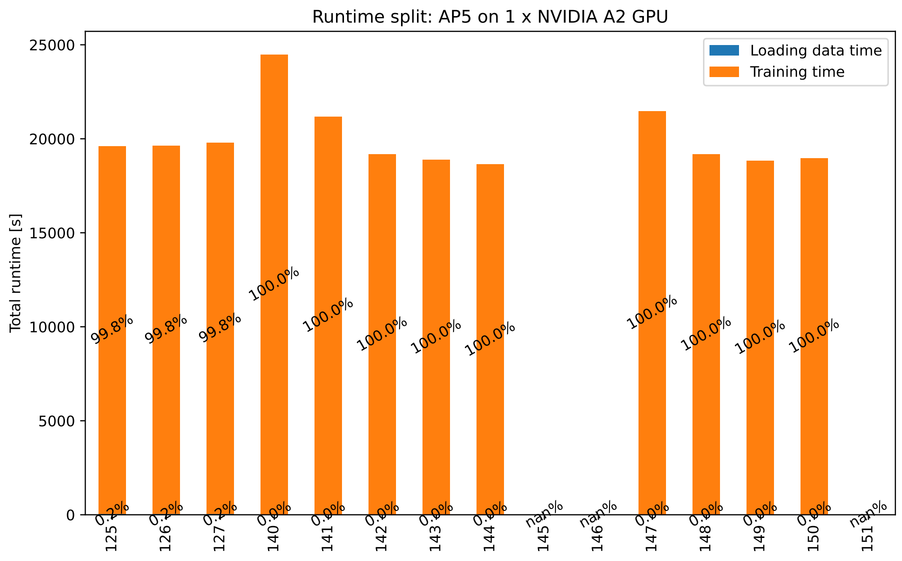


Figure 70: AP5 E4 A2: Percentages of runtime spent for training and loading data.
ap5-a2-runtime-split-graph

4.9 AP 6

4.9.1 Notes

The application is showing the following details:

Training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
32.43 GB 1.4 GB	-	35 064	(550, 350, 25)	750

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
-	-	cross-entropy	20 epochs,1.7 GB dataset on JSC 50 epochs, 1.4 GB dataset on E4

Data formats	Frameworks (to be) used
NetCDF	PyTorch

AP6 aims to achieve a non-linear dimensionality reduction using a siamese NN structure called DeepCluster v2 (DCv2) that allows self-supervised clustering of visual features [1, 2]. The algorithm allows for distributed computation on multiple nodes and GPUs. To attempt achieve that, we make use of PyTorch's `torchrun` command, which spawns multiple workers (processes) that can operate on an individual GPU and connect to other worksers, even across nodes.
DCv2 makes use of two branches to achieve self-supervised clustering:

1. The first branch runs the input image through a ResNet-50 [3] followed by a simple multi-layer perceptron (MLP). The output feature vector of the MLP is then used to apply a spherical K-means on the feature vector: the algorithm is initiated with a random set of centroids, and each sample gets a label assigned based on the lowest cosine similarity between its feature vector and the cluster centers.

2. The second branch is almost identical to the first branch: it runs the image through a ResNet-50 and a MLP. Here, the resulting feature vector is then used – together with the output of the upper branch – to calculate the cross-entropy loss function. The network then tries to minimize this loss function by

sequentially adjusting the parameters in each layer through back propagation. The resulting set of parameters is then used for the next iteration.

Each branch uses an individual random crop that is 75% of the original input data size in terms of area. Here, we use daily ERA5 data provided in the form of netCDF files.
AP6 makes use of Apptainer containers to run the application. For the Grace Hopper experiments a separate Docker image had to be developed to allow running the application since the node employed an ARM-based CPU.
In sections 4.9.2 to 4.9.6, we are going to analyze the single-GPU training performance of the AP6 benchmark.
Data for the benchmarks which are shown here can be found in appendix 6.6.
The application code is accessible on GitHub[6].
For this application, an issue has been discovered that prevents the application from effectively utilizing the GPUs on the JSC/JURECA-DC systems. Due to the issue not being present on E4 systems, the likely cause is a misconfiguration in the job launch environment. As of the time of this report, the issue is still under investigation.

4.9.2 JURECA DC - NVIDIA A100

- Running on 4 GPUs couldn't be achieved (workers time out), which is really surprising since 1, 2, and 3 GPUs works well.
- Running on multiple nodes couldn't be achieved, whereas it worked in the previous benchmarks. We moved from Facebook Research's VISSL library to our own implementation using torchrun for distributed computing.
- Runtime increases with the number of GPUs almost linearly.

4.9.3 JURECA DC - NVIDIA H100

- Running on 4 GPUs couldn't be achieved (workers time out)
- For single GPU (1 process), the H100 outperforms the A100
  - ~11% better (faster) compared to the A100 in terms of total training, epoch, batch, and data load time
  - ~55% less peak power consumption compared to A100
- Runtime increases with the number of GPUs, but slightly better than for the A100

[6] `https://github.com/4castRenewables/maelstrom-a6`

4.9.4 JURECA DC - AMD MI250

- Running more than 2 workers (GPUs/GCDs) couldn't be achieved
- For single GPU (1 process), the MI250 outperforms the H100 (A100):
  - ~28% (36%) less total training, epoch, batch, and data load time
  - ~10% (~56%) less power consumption
- Runtime increases with the number of GPUs, but less drastically than for the NVIDIA GPUs

4.9.5 E4 Intel System - NVIDIA A2

- Running on multiple GPUs and nodes *decreases* the runtime, roughly linearly with $N_{\text{nodes}} \times N_{\text{GPUs}}$.
- For single GPU (1 process), the A2 performance-wise is between the H100 and the MI250
  - ~8% less training time than H100, but ~27% more than MI250
  - ~54% less peak power consumption compared to H100, even ~49% less than the MI250
-

4.9.6 E4 ARM System - NVIDIA Grace Hopper GH200

- Running on multiple GPUs and nodes *decreases* the runtime, roughly linearly with $N_{\text{nodes}} \times N_{\text{GPUs}}$.
- For single GPU (1 process), the GH200 outpfermors all the other GPUs:
  - ~8% less total training, epoch, batch, and data load time than MI250
  - has ~94% *more* peak power consumption than MI250

4.9.7 Runtime split

Loading data time is around 6%-7% of the total on JSC systems (A100, H100, MI250; Figure 71, Figure 72, Figure 73) and around 12.2% on E4 systems (GH200, A2; Figure 74, Figure 75). Of note is the fact that the best Time-to-solution on E4 systems was achieved with multiple nodes used (2 nodes for both GH200 and A2).

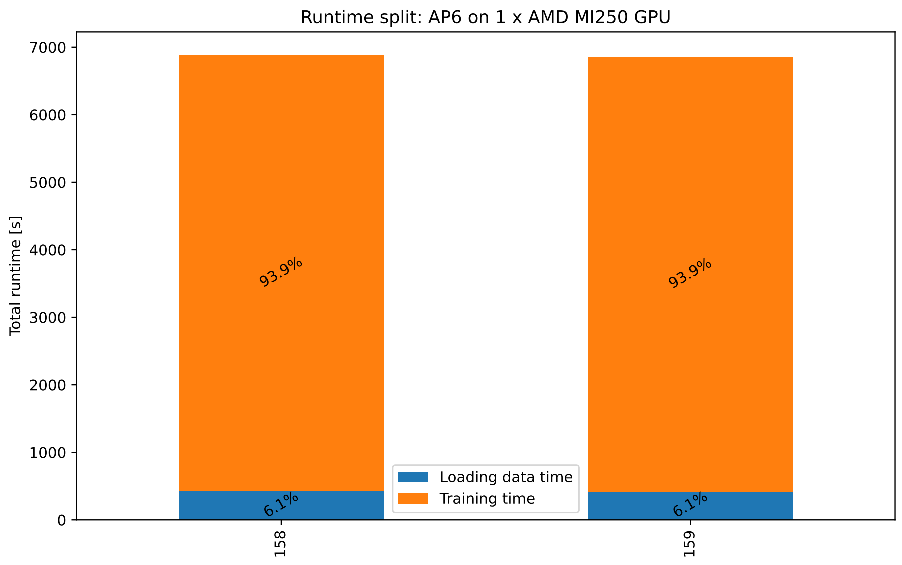


Figure 71: AP6 JURECA-DC MI250: Percentages of runtime spent for training and loading data. *ap6-mi250-runtime-split-graph*

4.9.8 Node Energy Consumption

In Figures 76 and 77, we present the comparison between GPU and node energy measurements as the number of GPUs and nodes involved changes.
A2 Hardware Configurations:

1. **1 Node, 1 GPU (Experiment 0):** This configuration exhibits the longest total runtime, approximately 8959.40 seconds. The Total GPU energy amounts to 75.55 Wh, while the node average energy is 928.66 Wh. The Action MJs score of 29952.87 indicates efficient utilization of computational resources.

2. **1 Node, 2 GPUs (Experiment 1):** With a shorter runtime of 4711.49 seconds, this configuration shows a slightly higher total GPU energy of 84.43 Wh and a lower node energy (588.65 Wh) compared to the previous configuration. The Action MJs score of 9984.34, almost one-third of the previous case, suggests a good improvement in performance and energy efficiency.

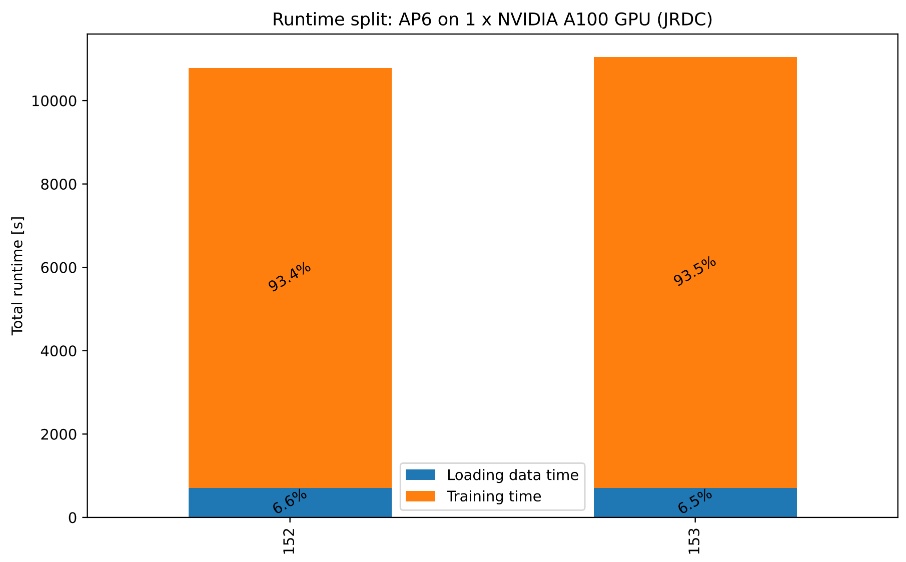


Figure 72: AP6 JURECA-DC A100: Percentages of runtime spent for training and loading data. *ap6-a100jrdc-runtime-split-graph*

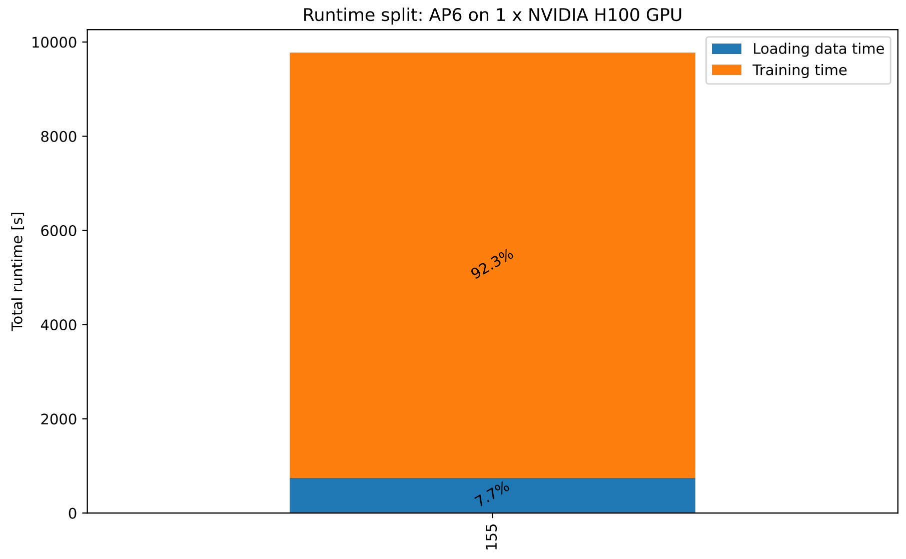


Figure 73: AP6 JURECA-DC H100: Percentages of runtime spent for training and loading data. *ap6-h100-runtime-split-graph*

3. **2 Nodes, 2 GPUs (Experiment 2):** Similar to the previous configuration in terms of runtime, this setup exhibits comparable Total GPU energy (83.20 Wh) but higher System average energy (1055.62 Wh) due to the usage of two nodes instead of one. The Action score of 19079.84 indicates efficient performance with higher energy consumption.

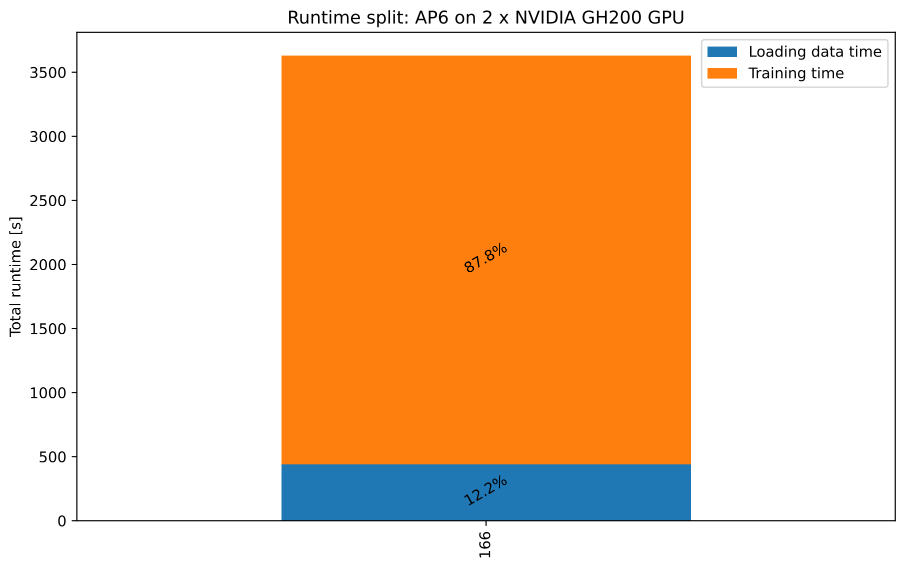


Figure 74: AP6 E4 GH200: Percentages of runtime spent for training and loading data. *ap6-gh200-runtime-split-graph*

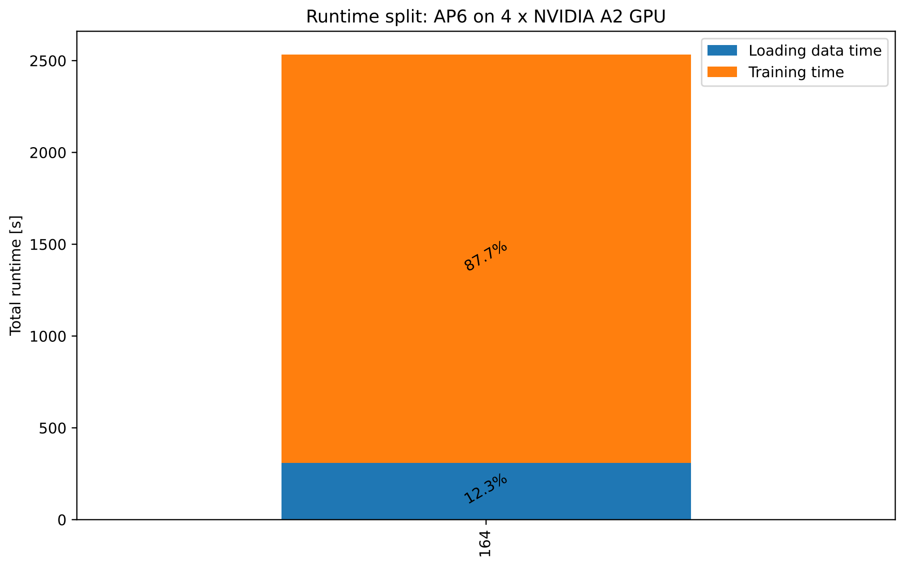


Figure 75: AP6 E4 A2: Percentages of runtime spent for training and loading data. *ap6-a2-runtime-split-graph*

4. **2 Nodes, 4 GPUs (Experiment 3):** This configuration achieves the shortest runtime of 2533.19 seconds.It shows a still comparable total GPU energy (86.42 Wh) and a System average energy that amounts to 636.29 Wh. The Action MJs score of 5802.69 tells us that, considering both the energy and the runtime, this configuration gets the best performance compared to other

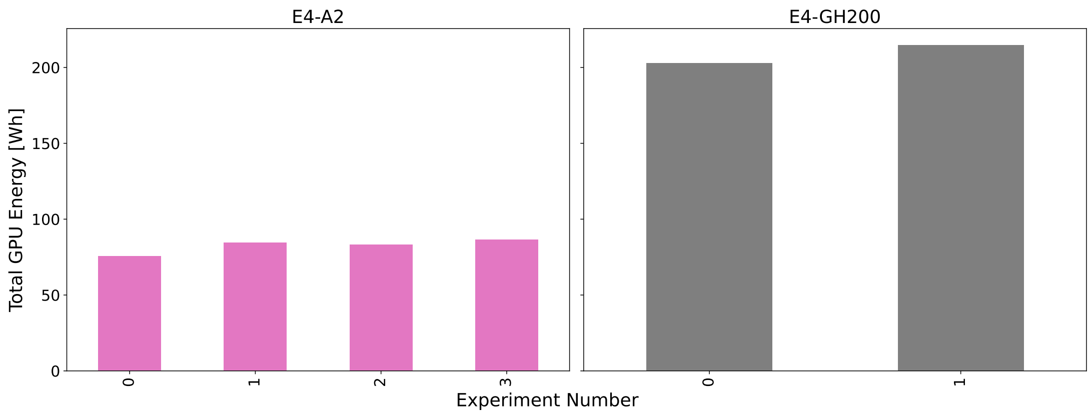
configurations.

GH200 Hardware Configurations:

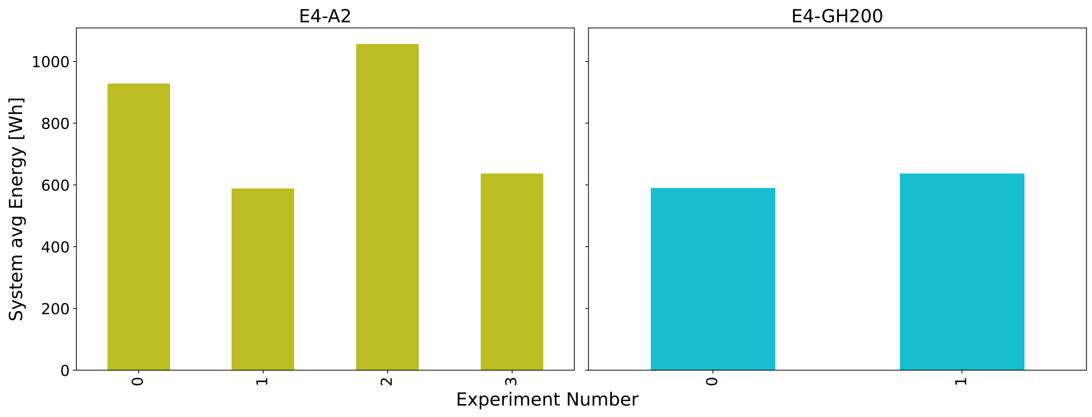
1. **1 Node, 1 GPU (Experiment 0):** It shows a total runtime of 6682.52 seconds. This configuration exhibits a total GPU energy of 202.85 Wh and System average energy of 590.41 Wh. The Action MJs score of 14203.63 indicates moderate performance and energy efficiency.
2. **2 Nodes, 2 GPUs (Experiment 1):** This configuration shows a shorter runtime compared to the previous experiment (3629.93 s) and slightly higher GPU and node average energy (respectively 214.86 Wh and 636.73 Wh). The Action MJs score of 8320.57 places this configuration as the best one after Experiment 3 on NVIDIA A2.

4.9.9 Conclusion

On JSC machines, there must be some sort of communication bottleneck between the processes/workers/GPUs when using `torchrun`. This became obvious in the previous benchmarks already as well. The cause of this bottleneck is currently under investigation. On the E4 machines, the total training, epoch, batch, and data load time decreases nearly linearly with the number of GPUs and nodes, as one would expect.
Overall, the GH200 seems to yield the best per-GPU performance, although the power measurements indicate that this performance comes with higher energy consumption. The MI250 closely follows the H100 in terms of performance, but has much less energy consumption. Overall, the A2 seems to give the best performance-energy tradeoff.
As the comparative benchmarks in section 4.1 have shown, the application scales almost linearly with the number of GPUs. The best runtime was achieved using 2 A2 nodes with 2 GPUs each. Furthermore, for the single-gpu case, the A2 shows around 75% of the performance of the best performer GH200 at around 25% of the max. power draw and 37.2% of the energy (both A2 GPUs). Using both GPUs in the A2 node outperforms the GH200 node by 41% using only 42% of the energy.
Taking into account the full node energy consumption of both systems, specifically considering the optimal configurations (2 nodes with 2 GPUs each for A2 nodes and 2 nodes with 1 GPU each for GH200 nodes), the system average energy consumption was nearly identical, approximately ∼636 Wh for both configurations.



(a) GPU energy consumption



(b) Node energy consumption

Figure 76: AP6 Node/GPU Energy: Comparison between GPU and full-node energy consumption for A2 and GH200 GPUs. ap6-nodeE-vs-GPU

However, despite the similar energy consumption, the GH200 configuration exhibited a longer total runtime. Analyzing the Action scores (Fig. 77) for these two configurations suggests that the A2 configuration with two nodes and four GPUs demonstrates better performance and energy efficiency compared to the GH200 configuration.

Due to the nature of the multi-node/multi-GPU results, further scaling experiment should be performed once the issue on JSC systems has been fixed. These results will be considered for the bespoke system design which will be documented in D3.8.

Comparison to the previous benchmarks is not possible since the application setup changes significantly (the VISSL library was used), and also data size and shape

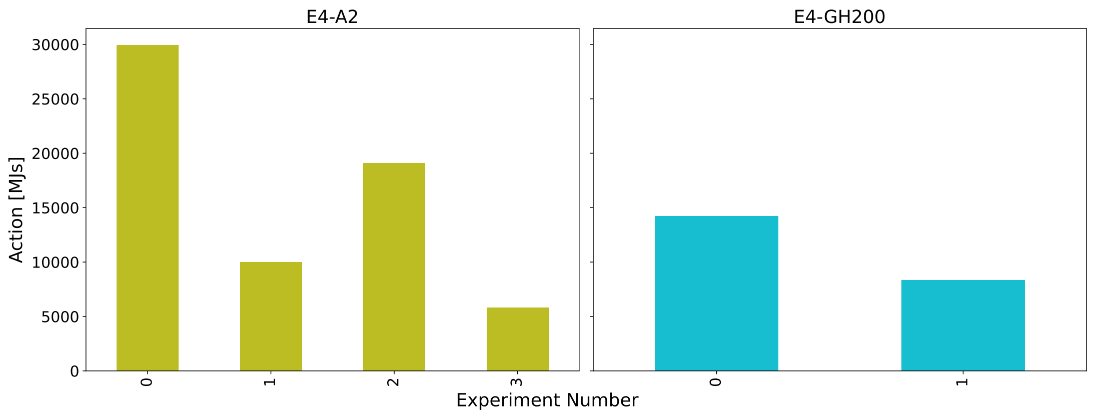


Figure 77: AP6 Action: Action Metric for A2 and GH200 nodes. *ap6-full-node-action*

changed.

4.10 AtmoRep

Most of the MAELSTROM applications remained rather small regarding the use of many compute nodes for training over the course of the project. However, as the field of machine learning in weather and climate predictions has developed at a neck-breaking pace during the last four years, MAELSTROM has also contributed to new machine learning applications that were not foreseeable during the proposal writing. This includes pure machine learning models that are today competitive with conventional models such as NeuroGCM [4] and AIFS[7]. However, it also includes the AtmoRep model that is making first steps towards the use of representation learning and a Foundation Model for weather and climate [5].

AtmoRep should be of particular interest for the high-performance computing community. As a Foundation Model, the need for training data is much larger when compared to task-specific machine learning models, as several datasets for input and output are combined and as the training datasets are covering several application domains. Furthermore, the model is larger as it should be generalising into more application areas. On the other hand, the model can – once it is trained – be applied to various application domains. In contrast to models such as AIFS, AtmoRep could not only be used for global weather predictions, but also for limited area modelling, local downscaling, and post-processing.

Since the training of a Foundation Model will likely be the most costly HPC application from the weather and climate modelling community in the next years, MAELSTROM has supported the development and has also used the tool in the context of the local downscaling application.

Figure 78 shows the weak scaling behaviour when training AtmoRep on JUWLES Booster. Here, a configuration of AtmoRep is exemplified which incorporates three variables, the temperature and the horizontal wind field components to learn an abstract representation of atmospheric dynamics. While the model fits on a single GPU, this configuration would only allow for a small mini-batch size of 4 samples. Since larger mini-batch sizes are essential for the optimization efficiency of the transformer-based model, data parallelism is used to incorporate more samples per mini-batch. Increasing the number of nodes from 1 to 32 (corresponding to the utilization of 4 and 128 A100 GPUs) improves the accuracy (smaller training loss), but only has a modest effect on the wallclock time. Thus, the overhead due to the required communication between the worker GPUs (allreduce on gradients and synchronising parameter updates) is small and allows effective processing of more

[7] https://www.ecmwf.int/en/about/media-centre/aifs-blog/2023/ECMWF-unveils-alpha-version-of-new-ML-model

training data.

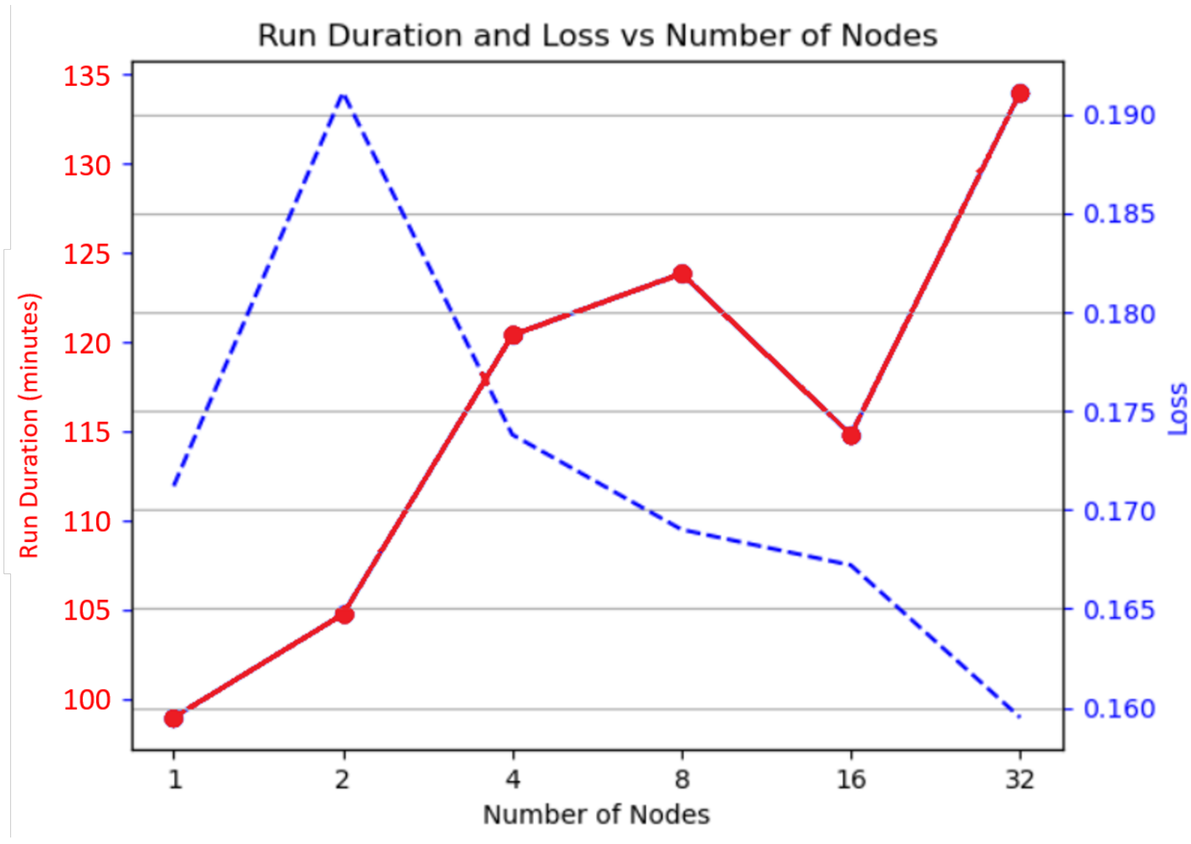


Figure 78: AtmoRep JUWELS Booster Scaling: Total GPU energy consumption for the training phase

5 Conclusion

The work done in this deliverable is based on a strong cooperation between the WP3 partners providing the computing systems and the application developers from WP1, often running applications in a coordinated manner between developers and hardware engineers to verify all the physical parameters of the computing systems.

For each application, data were collected and presented in the form of graphs in the various dedicated sections.

In comparison to the results in D3.4 and D3.6 more applications have implemented multi-GPU parallelisation. It was possible to measure the reduction of time-to-solution and energy-to-solution with increasing number of utilized GPUs. Furthermore the range of tested accelerators has been expanded to include newer generations of AMD and NVIDIA hardware. Another addition to the tested platforms is the Graphcore IPU GC200, which showed excellent performance with Application 1. GPU/IPU energy consumption was measured on all devices and full-node energy consumption on E4 machines.

In the process of the benchmarking collaborations, multiple issues were uncovered and fixed. Some caveats that remained have been documented and are under investigation. WP3 provided software support to WP1 to enable the use of the JSC and E4 machines, correct misconfigurations, help analyse the data and acquire necessary statistics.

The results of this deliverable will be used for the bespoke system design which will be documented in D3.8.

The insights gained through this deliverable – especially regarding energy-efficiency – will be refined and exploited in form of a paper, which is currently under preparation.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018.

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2021.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Dmitrii Kochkov, Janni Yuval, Ian Langmore, Peter Norgaard, Jamie Smith, Griffin Mooers, James Lottes, Stephan Rasp, Peter Düben, Milan Klöwer, et al. Neural general circulation models. *arXiv preprint arXiv:2311.07222*, 2023.

[5] Christian Lessig, Ilaria Luise, Bing Gong, Michael Langguth, Scarlet Stadler, and Martin Schultz. Atmorep: A stochastic model of atmosphere dynamics using large scale representation learning. *arXiv preprint arXiv:2308.13280*, 2023.

6 Appendix

JUBE id	56	62	64
JUBE WP	0	0	0
JobID	9342181	9346645	9346701
NodeID	jwb0428	jwb0067	jwb0099
Hardware	JSC-A100	JSC-A100	JSC-A100
MPI tasks	4	4	4
CPUs/task	24	24	24
Total runtime	398.08	386.40	382.44
Training time	375.19	377.84	373.95
avg. epoch time [s]	124.87	125.78	124.48
performance [GB/s]	2.64	2.62	2.65
first epoch time [s]	156.06	152.02	149.72
min epoch time [s]	107.39	108.34	107.57
max epoch time [s]	156.06	152.02	149.72
avg. batch time [s]	0.05	0.05	0.05
loss	0.20	0.20	0.197
val loss	0.26	0.25	0.235
max cpu mem	55.30	48.31	48.13
max gpu mem	4.75	7.23	7.23
Total energy [Wh]	61.11	56.98	56.62
Max power [W]	310.39	309.80	312.06
Max aggregate power [W]	1077.59	1065.81	1124.62
Avg aggregate power [W]	546.06	527.24	527.78

Table 6: AP1 JUWELS Booster NVIDIA A100 training benchmark

6.1 AP 1

JUBE id	60	54	68
JUBE WP	0	0	0
JobID	12616016	12614888	12619712
NodeID	jrc0332	jrc0377	jrc0213
Hardware	JSC-A100	JSC-A100	JSC-A100
MPI tasks	4	4	4
CPUs/task	32	32	32
Total runtime	509.81	512.61	503.34
Training time	501.49	504.4	495.11
avg. epoch time [s]	167.03	167.96	164.95
performance [GB/s]	1.97	1.96	2
first epoch time [s]	192.73	190.61	186.39
min epoch time [s]	147.62	148.51	148.72
max epoch time [s]	192.73	190.61	186.39
avg. batch time [s]	0.06	0.06	0.06
loss	0.201	0.204	0.202
val loss	0.253	0.253	0.264
max cpu mem	53.02	53.61	53.22
max gpu mem	7.23	7.23	7.23
Total energy [Wh]	68.62	68.15	67.88
Max power [W]	323.31	304.85	310.72
Max aggregate power [W]	1101.56	1127.38	1133.39
Avg aggregate power [W]	480.96	476.4	483.05

Table 7: AP1 JURECA NVIDIA A100 training benchmark

JUBE id	28	63	65
JUBE WP	0	0	0
JobID	12520406	12616043	12619705
NodeID	jrc0880	jrc0880	jrc0880
Hardware	JSC-H100	H100_GPU	H100_GPU
MPI tasks	4	4	4
CPUs/task	36	36	36
Total runtime	293.47	323.93	312.83
Training time	287.25	316.41	305.91
avg. epoch time [s]	95.62	105.31	101.91
performance [GB/s]	3.44	3.13	3.23
first epoch time [s]	119.88	137.58	121.5
min epoch time [s]	81.73	87.9	91.45
max epoch time [s]	119.88	137.58	121.5
avg. batch time [s]	0.04	0.04	0.04
loss	0.20	0.203	0.2
val loss	0.25	0.266	0.247
max cpu mem	67.48	65.73	60.56
max gpu mem	4.95	4.64	4.64
Total energy [Wh]	49.18	53.76	51.62
Max power [W]	245.63	242.75	241.13
Max aggregate power [W]	882.12	881.65	869.14
Avg aggregate power [W]	596.02	589.92	588.99

Table 8: AP1 JURECA NVIDIA H100 training benchmark.

JUBE id	50	66	67
JUBE WP	0	0	0
JobID	12563980	12619706	12619707
NodeID	jrc0851	jrc0851	jrc0851
Hardware	JSC-MI250X	JSC-MI250X	JSC-MI250X
MPI tasks	8	8	8
CPUs/task	12	12	12
Total runtime	450.10	448.76	433.91
Training time	443.77	443.14	428.52
avg. epoch time [s]	147.82	147.59	142.74
performance [GB/s]	2.23	2.23	2.31
first epoch time [s]	175.77	170.63	166.65
min epoch time [s]	124.17	124.27	123.71
max epoch time [s]	175.77	170.63	166.65
avg. batch time [s]	0.05	0.05	0.05
loss	0.21	0.21	0.21
val loss	0.35	0.25	0.26
max cpu mem	47.93	49.45	47.93
max gpu mem	4.30	4.29	4.29
Total energy [Wh]	86.66	86.16	84.86
Max power [W]	241.00	243.00	243.00
Max aggregate power [W]	951.00	952.00	950.00
Avg aggregate power [W]	693.09	691.21	704.05

Table 9: AP1 JURECA AMD MI250 training benchmark

JUBE id	45	47	48
JUBE WP	0	0	0
JobID	8570	8572	8575
NodeID	icnode05	icnode05	icnode04
Hardware	E4-A2	E4-A2	E4-A2
MPI tasks	1	1	1
CPUs/task	32	32	32
Total runtime	3525.22	3574.60	3537.01
Training time	3518.85	3566.64	3531.05
avg. epoch time [s]	1172.91	1188.72	1176.83
performance [GB/s]	0.28	0.28	0.28
first epoch time [s]	1188.50	1206.52	1196.14
min epoch time [s]	1161.25	1168.48	1164.90
max epoch time [s]	1188.50	1206.52	1196.14
avg. batch time [s]	0.42	0.43	0.42
loss	0.14	0.14	0.13
val loss	0.18	0.17	0.17
max cpu mem	72.47	70.62	72.06
max gpu mem	4.75	4.87	4.87
Total energy [Wh]	49.76	50.20	52.30
Max power [W]	57.25	57.17	58.81
Max aggregate power [W]	57.25	57.17	58.81
Avg aggregate power [W]	50.79	50.47	53.13
System avg power [W]	376.95	377.88	382.62
System avg VA [W]	415.10	415.83	420.02

Table 10: AP1 E4 NVIDIA A2 training benchmark

JUBE id	39	34	38
JUBE WP	0	0	0
JobID	8412	8407	8411
NodeID	108b0901e6ce	91cf0750e504	5c1ae1417f89
Hardware	E4-GH200	E4-GH200	E4-GH200
MPI tasks	1	1	1
CPUs/task	32	32	32
Total runtime	459.13	458.20	458.26
Training time	454.83	453.94	453.88
avg. epoch time [s]	151.56	151.27	151.25
performance [GB/s]	2.18	2.18	2.18
first epoch time [s]	155.62	155.41	155.78
min epoch time [s]	149.09	149.03	148.97
max epoch time [s]	155.62	155.41	155.78
avg. batch time [s]	0.05	0.05	0.05
loss	0.14	0.14	0.14
val loss	0.18	0.19	0.18
max cpu mem	57.72	58.67	57.97
max gpu mem	4.86	4.86	4.86
Total energy [Wh]	32.30	32.02	32.17
Max power [W]	322.41	315.61	322.54
Max aggregate power [W]	322.41	315.61	322.54
Avg aggregate power [W]	252.88	251.26	252.34
System avg power [W]	509.05	509.39	505.88
System avg VA [W]	543.12	544.35	539.68

Table 11: AP1 E4 NVIDIA GraceHopper H200 training benchmark.

Experiment number	1	2	3	4	5	6	7	8	9
Job ID	12618724	12618736	12618737	12618738	12618739	12618740	12618729	12618741	12618746
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	4	4	4	2	2	2	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	48	48	48	48	48	48	48	48	48
Total runtime	227.9508	224.1946	225.6415	259.3447	258.3036	254.8442	251.8506	252.5185	252.2961
Total training time	204.0506	200.3901	201.6572	233.51	232.7687	229.5489	231.1401	231.9173	231.3979
Training time for epoch	178.382	173.4357	173.8687	206.6352	206.0338	202.9273	208.4492	209.0497	208.7236
Total IO time	18.6026	18.2687	18.0437	18.5293	17.8638	18.2255	18.5294	18.5378	18.5689
Avg. training time per batch	0.1286	0.1257	0.1264	0.071	0.071	0.0702	0.0329	0.0328	0.0328
Max. training time per batch	11.0836	10.7414	10.6878	5.3022	5.2828	3.6121	2.0141	1.8561	2.1086
Final training loss	0.6241	0.6241	0.6241	0.6038	0.6038	0.6038	0.6054	0.6054	0.6054
Final validation loss	0.576	0.576	0.576	0.5744	0.5744	0.5744	0.5748	0.5748	0.5748
Max CPU memory per MPI task [GB]	5.1833	5.238	5.2412	4.6814	4.6821	4.6844	4.3706	4.3752	4.3738
MAX GPU memory per MPI task[GB]	4.4138	4.4138	4.4138	4.4158	4.4143	4.4158	3.4038	3.4038	3.4038
Node ID	jrc0384	jrc0223	jrc0224	jrc0255	jrc0256	jrc0223	jrc0448	jrc0384	jrc0224
GPU energy consumption [Wh]	25.75	26.05	25.02	22.65	23.22	22.55	12.75	13.39	12.62
Max. GPU power [W]	158.52	174.37	157.44	232.8	236.76	239.23	211.96	218.68	209.06
Avg. aggr. GPU power [W]	349.67	356.74	342.79	272.95	280.69	277.8	156.75	164.21	154.42
Max. aggr. GPU power [W]	482.97	516.79	503.19	427.49	415.92	444.01	211.96	218.68	209.06
Evaluation time	22.7373	22.3363	22.4349	24.3741	24.1272	24.1239	19.8265	19.7702	19.7796

Table 12: AP2 Jureca A100 benchmarks

Experiment number	1	2	3	4	5	6	7	8	9
Job ID	12579747	12579817	12579818	12583012	12583016	12583017	12583130	12583131	12583132
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	4	4	4	2	2	2	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	48	48	48	48	48	48	48	48	48
Total runtime	206.4499	207.7464	205.5181	216.3987	215.8093	211.5154	230.5017	230.6238	230.8618
Total training time	185.9111	187.0928	185.085	195.2764	191.7324	190.6831	212.6544	212.7636	213.2606
Training time for epoch	163.3003	164.1674	162.5144	171.9455	168.6813	167.6624	192.565	192.7854	192.9922
Total IO time	16.3036	15.6125	15.9008	12.251	11.9919	12.0208	18.4792	18.5371	18.2943
Avg. training time per batch	0.1139	0.1146	0.1142	0.0579	0.057	0.0568	0.0292	0.029	0.0291
Max. training time per batch	21.8324	21.6073	21.823	11.4498	9.4602	9.2181	3.4512	3.1545	2.908
Final training loss	0.6016	0.6016	0.6016	0.5669	0.5669	0.5669	0.5763	0.5763	0.5763
Final validation loss	0.5774	0.5774	0.5774	0.5756	0.5756	0.5756	0.5763	0.5763	0.5763
Max CPU memory per MPI task [GB]	5.6463	5.6776	5.6717	4.8882	4.868	4.9039	4.6551	4.6603	4.678
MAX GPU memory per MPI task[GB]	4.4625	4.4625	4.4625	4.4607	4.4607	4.4607	3.46	3.46	3.46
Node ID	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880
GPU energy consumption [Wh]	30.88	30.88	31.72	19.77	19.68	20.21	11.5	11.93	11.88
Max. GPU power [W]	164	160.73	168.75	209.79	206.2	215.42	204.54	201.25	204.69
Avg. aggr. GPU power [W]	473.23	470.82	486.63	287.6	287.56	301.07	159.96	165.86	164.95
Max. aggr. GPU power [W]	596.34	591.74	607.88	374.99	378.43	381.86	204.54	201.25	204.69
Evaluation time	19.4151	19.502	19.3908	20.0309	19.7846	19.8373	16.8245	16.7745	16.5816

Table 13: AP2 Jureca H100 benchmarks

6.2 AP 2

Experiment number	1	2	3	4	5	6	7	8	9	10
Job ID	12579609	12579708	12579819	12579820	12583134	12583339	12583340	12583341	12583342	12583349
#Nodes	1	1	1	1	1	1	1	1	1	1
#GPUs	8	8	8	8	4	4	4	2	2	2
#MPI tasks	1	1	1	1	1	1	1	1	1	1
#CPUs per task										
Total runtime	186.2039	187.0761	202.205	183.9739	216.4282	219.4606	214.1795	287.2455	289.4993	288.4124
Total training time	166.479	167.6768	166.6323	164.7586	192.7832	197.2303	192.2601	262.0113	264.1076	263.3007
Training time for epoch	146.2653	147.0875	146.2421	144.5845	169.7902	173.8169	169.4139	235.8763	237.625	237.1474
Total IO time	11.2154	11.5401	11.3537	11.4348	11.6496	11.7507	11.678	11.3317	11.2366	11.3611
Avg. training time per batch	0.2122	0.2152	0.214	0.2109	0.1177	0.118	0.1161	0.0778	0.0786	0.0777
Max. training time per batch	13.2631	13.2907	13.4857	13.3556	3.652	3.2395	3.2407	1.8919	1.7841	1.846
Final training loss	0.5267	0.5267	0.5267	0.5267	0.6471	0.6471	0.6471	0.5962	0.5962	0.5962
Final validation loss	0.5787	0.5787	0.5787	0.5787	0.5755	0.5755	0.5755	0.5749	0.5749	0.5749
Max CPU memory per MPI task [GB]	8.3346	8.3285	8.3659	8.3562	6.1951	6.1935	6.2062	5.1827	5.0808	5.1925
MAX GPU memory per MPI task[GB]	4.6122	4.6122	4.6122	4.6122	4.6016	4.6016	4.6016	4.6002	4.6002	4.6002
Node ID	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851
GPU energy consumption [Wh]	45.44	45.47	47.2	45.27	35.41	35.56	35.26	28.89	28.92	28.86
Max. GPU power [W]	249	248	249	250	324	323	325	369	368	368
Avg. aggr. GPU power [W]	787.59	785.5	759.05	794.55	533.54	531.59	539.24	336.91	334.97	335.85
Max. aggr. GPU power [W]	955	946	951	956	623	621	625	369	368	368
Evaluation time	18.1947	18.2681	18.2484	18.1464	21.0891	21.1703	20.8598	24.1969	24.3783	24.104

Table 14: AP2 Jureca Mi250 benchmarks

Experiment number	1	2	3	4	5	6
Job ID	7468	7469	7470	8398	8403	8404
#Nodes	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1
#CPUs per task						
Total runtime	308.6113	325.6811	321.9197	340.4599	349.4078	342.2153
Total training time	287.7142	308.6535	305.2625	320.5216	331.9882	324.4152
Training time for epoch	255.1178	264.3815	261.1508	270.6704	286.9699	277.0629
Total IO time	12.3666	12.5104	12.3285	12.3634	12.5654	12.3089
Avg. training time per batch	0.0279	0.0281	0.0275	0.0276	0.0279	0.0292
Max. training time per batch	0.7032	0.6415	0.6459	0.6798	0.6414	0.6438
Final training loss	0.2408	0.2408	0.2408	0.2408	0.2408	0.2408
Final validation loss	0.5765	0.5765	0.5765	0.5765	0.5765	0.5765
Max CPU memory per MPI task [GB]	2.6812	2.6593	2.6421	2.6444	2.6869	2.6685
MAX GPU memory per MPI task[GB]	3.4573	3.4573	3.4573	3.4573	3.4573	3.4573
Node ID						
GPU energy consumption [Wh]	14.48	14.93	14.84	15.37	15.67	15.49
Max. GPU power [W]	236.53	196.61	199.09	217.33	206.78	220.85
Avg. aggr. GPU power [W]	158.01	155.55	156.49	152.73	152.71	154.36
Max. aggr. GPU power [W]	236.53	196.61	199.09	217.33	206.78	220.85
Evaluation time	16.6406	16.6458	16.2798	16.6056	17.0616	17.4694

Table 15: AP2 E4 Grace Hopper benchmarks

Experiment number	1	2	3
Job ID	8399	8400	8406
#Nodes	1	1	1
#GPUs	1	1	1
#MPI tasks	1	1	1
#CPUs per task			
Total runtime	1233.4279	1235.294	1237.6037
Total training time	1171.1144	1177.4848	1175.9654
Training time for epoch	1097.1279	1097.637	1095.2149
Total IO time	21.275	23.2722	20.7384
Avg. training time per batch	0.1597	0.1596	0.1599
Max. training time per batch	4.1171	1.7956	4.9655
Final training loss	0.2377	0.2377	0.2377
Final validation loss	0.5755	0.5755	0.5755
Max CPU memory per MPI task [GB]	2.8672	2.8672	2.8672
MAX GPU memory per MPI task[GB]	3.4083	3.4083	3.4083
Node ID	icnode05	icnode04	icnode04
GPU energy consumption [Wh]	17.36	17.81	17.94
Max. GPU power [W]	62.08	61.81	61.25
Avg. aggr. GPU power [W]	49.71	51.12	51.32
Max. aggr. GPU power [W]	62.08	61.81	61.25
Evaluation time	56.9683	57.3468	57.4546

Table 16: AP2 E4 A2

6.3 AP 3

Data location	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH
Experiment flags	–nocache	–nocache	–nocache				–dl_test –nocache	–dl_test –nocache	–dl_test –nocache
Experiment number	0	1	2	0	1	2	0	1	2
Job ID	12505616	12505617	12505618	12505619	12505620	12505621	12505622	12505623	12505624
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	48	48	48	48	48	48	48	48	48
Total runtime [s]	2281.12	2281.15	2281.22	1561.08	1561.21	1564.7	1312.88	1312.86	1312.8
Total training time [s]	2278.15	2278.12	2278.11	1558.18	1558.17	1558.33	1309.8	1309.77	1309.8
Avg. training time per epoch [s]	434.6	435.2	435.2	287.2	288.4	290.4	218	220.4	223.6
Performance [GB/s]	0.15	0.15	0.15	0.22	0.22	0.22	0.27	0.27	0.27
First epoch training time [s]	463	461	461	467	469	480	225	225	225
Min. training time per epoch	427	428	428	242	243	243	210	219	221
Max. training time per epoch	463	461	461	467	469	480	225	225	225
Avg. training time per batch	0.08	0.08	0.08	0.05	0.05	0.05	0.05	0.05	0.05
Final training loss	0.0453	0.0389	0.0419	0.0437	0.04	0.045			
Final validation loss	0.073	0.0549	0.0742	0.0581	0.0545	0.0658			
Max CPU memory per MPI task [GB]	4.67	4.62	4.62	65.63	65.57	65.66	2.1	1.87	1.83
MAX GPU memory per MPI task[GB]	0.63	0.63	0.63	0.63	0.63	0.63	0	0	0
Integrated Total Energy [Wh]	168.18	166.9	170.26	123.94	127.13	122.25	88.92	88.07	89.29
Max Power [W]	147.27	152.01	149.94	148.75	157.77	149.15	68.55	68.5	68.94

Table 17: AP3 JURECA-DC A100 training benchmark A100

Data location	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH
Experiment flags	–nocache	–nocache	–nocache				–dl_test –nocache	–dl_test –nocache	–dl_test –nocache
Experiment number	0	1	2	0	1	2	0	1	2
Job ID	12548185	12548186	12548187	12548188	12548189	12548190	12548191	12548192	12548193
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	72	72	72	72	72	72	72	72	72
Total runtime [s]	1917.85	1917.19	1917.15	11991.1	1318.76	1258.83	1013.22	1013.35	1013.65
Total training time [s]	1914.73	1914.77	1975.69	1194.71	1314.74	1254.74	1009.85	1009.82	1009.84
Avg. training time per epoch [s]	350.4	337.6	350	236.4	235.8	236.2	192.6	191.2	189.4
Performance [GB/s]	0.18	0.18	0.18	0.29	0.27	0.28	0.35	0.35	0.35
First epoch training time [s]	357	356	357	377	372	373	201	198	191
Min. training time per epoch	335	332	325	201	201	202	189	189	183
Max. training time per epoch	368	356	397	377	372	373	201	198	192
Avg. training time per batch	0.07	0.07	0.07	0.04	0.05	0.04	0.03	0.03	0.03
Final training loss	0.0418	0.0207	0.0453	0.0353	0.0363	0.0423			
Final validation loss	0.0651	0.0312	0.066	0.0525	0.0525	0.0633			
Max CPU memory per MPI task [GB]	4.42	4.16	4.43	63.33	63.43	63.42	3	3.56	3.43
MAX GPU memory per MPI task[GB]	0.61	0.61	0.61	0.61	0.61	0.63	0	0	0
Integrated Total Energy [Wh]	137.73	137.69	141.59	91.7	99.89	95.79	66.47	66.42	66.49
Max Power [W]	138.8	139.82	135.88	142.71	141.84	142.83	86.97	86.23	86.4

Table 18: AP3 JURECA-DC H100 training benchmark H100

Data location	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH
Experiment flags	–nocache	–nocache	–nocache				–dl_test –nocache	–dl_test –nocache	–dl_test –nocache
Experiment number	0	1	2	0	1	2	0	1	2
Job ID	12543545	12543546	12543547	12543548	12543549	12543550	12543551	12543552	12543553
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	8	8	8	8	8	8	8	8	8
#MPI tasks	8	8	8	8	8	8	8	8	8
#CPUs per task	6	6	6	6	6	6	6	6	6
Total runtime [s]	810.9	746.87	680.23	581.6	539.92	571.07	549.34	568.05	545.42
Total training time [s]	805.39	742.12	675.52	575.86	532.34	564.38	544.34	562.04	540.86
Avg. training time per epoch [s]	141	137	133	89	94.2	94	96.8	93.8	97.2
Performance [GB/s]	0.43	0.47	0.52	0.61	0.66	0.62	0.64	0.62	0.65
First epoch training time [s]	151	145	144	157	162	158	101	98	101
Min. training time per epoch	123	124	122	60	69	63	75	58	76
Max. training time per epoch	168	165	144	157	162	158	115	103	116
Avg. training time per batch	0.03	0.03	0.03	0.02	0.02	0.02	0.02	0.02	0.02
Final training loss	0.054	0.0545	0.0453	0.0577	0.0546	0.0559			
Final validation loss	0.0791	0.0793	0.0738	0.0792	0.0789	0.0836			
Max CPU memory per MPI task [GB]	3.04	3.12	3.05	9.48	9.46	9.46	2.5	1.91	2.51
MAX GPU memory per MPI task[GB]	0.69	0.67	0.67	0.67	0.67	0.69	0	0	0
Integrated Total Energy [Wh]	115.54	109.12	102.4	89.64	85.85	88.69	57.27	59.02	56.77
Total Energy from Counter [Wh]	137.73	137.69	141.59	91.7	99.89	95.79	66.47	66.42	66.49
Max Power [W]	180	180	180	205	201	205	101	100	101

Table 19: AP3 JURECA-DC MI250 training benchmark Mi250 - 8 GPUS

Data location	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH
Experiment flags	–nocache	–nocache	–nocache				–dl_test –nocache	–dl_test –nocache	–dl_test –nocache
Experiment number	0	1	2	0	1	2	0	1	2
Job ID	12551471	12551472	12551473	12551474	12551475	12551476	12551477	12551478	12551479
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	48	48	48	48	48	48	48	48	48
Total runtime [s]	2516.11	2575.67	2575.39	2335.45	2335.6	2335.58	1011.8	1011.62	1011.57
Total training time [s]	2513.46	2573.5	2573.44	2333.45	2333.43	2333.43	1009.68	1009.69	1009.68
Avg. training time per epoch [s]	491.8	498.3	485.6	426	424	424	149.4	147.2	146.4
Performance [GB/s]	0.14	0.14	0.14	0.15	0.15	0.15	0.35	0.35	0.35
First epoch training time [s]	503	506	508	543	537	537	152	148	147
Min. training time per epoch	488	493	491	396	395	395	148	148	146
Max. training time per epoch	503	506	508	543	537	547	152	148	147
Avg. training time per batch	0.09	0.09	0.09	0.08	0.08	0.08	0.03	0.03	0.3
Final training loss	0.0449	0.0445	0.039	0.046	0.042	0.037			
Final validation loss	0.0631	0.0631	0.055	0.067	0.057	0.062			
Max CPU memory per MPI task [GB]	3.04	3.12	3.05	9.48	9.46	9.46	2.5	1.91	2.51
MAX GPU memory per MPI task[GB]	0.75	0.75	0.77	0.76	0.76	0.75	0	0	0
Integrated Total Energy [Wh]	93.99	95.48	95.35	89.73	89.69	88.7	27.10	27	27.01
Total Energy from Counter [Wh]	95.4	96.10	95.97	90.29	90.23	90.27	27.25	27.25	27.25
Max Power [W]	161	158	155	183	183	184	100	100	100

Table 20: AP3 JURECA-DC MI250 training benchmark Mi250 - 1 GPUS

Data location	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH	SCRATCH
Experiment flags	–nocache	–nocache	–nocache				–dl_test –nocache	–dl_test –nocache	–dl_test –nocache
Experiment number	0	1	2	0	1	2	0	1	2
Job ID	8314	8315	8316	8317	8318	8319	8320	8321	8322
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1
#CPUs per task	32	32	32	32	32	32	32	32	32
Total runtime [s]	2006.58	2017.11	2316.71	2060.84	1973.65	1971.69	1389.47	1319.64	1432.43
Total training time [s]	2004.11	2014.64	2311.61	2255.73	1970.66	1968.7	1395.67	1317.35	16429.56
Avg. training time per epoch [s]	400.6	402.6	462.2	451	394	393.8	279	263.6	285.8
Performance [GB/s]	0.17	0.17	0.17	0.16	0.18	0.18	0.25	0.27	0.24
First epoch training time [s]	405	406	696	696	419	420	272	257	284
Min. training time per epoch	392	397	403	388	387	388	268	255	275
Max. training time per epoch	405	406	696	696	419	420	298	287	303
Avg. training time per batch	0.07	0.07	0.07	0.06	0.06	0.07	0.06	0.06	0.06
Final training loss	0.0398	0.0348	0.0369	0.0451	0.0457	0.0434			
Final validation loss	0.0535	0.0581	0.051	0.053	0.0657	0.0576			
Max CPU memory per MPI task [GB]	5.04	4.99	5.51	66.51	67.55	67.55	2.25	2.24	2.35
MAX GPU memory per MPI task[GB]	0.63	0.63	0.64	0.63	0.64	0.61	0	0	0
Integrated Total Energy [Wh]	28.97	29.24	29.56	30.8	28.89	29.01	7.76	6.91	7.51
Max Power [W]	57.36	57.47	55.47	58.53	57.56	58.53	21.13	19.51	19.36
Max Power - full node [W]	498	498	443	443	443	492	426	358	351
Avg Power - full node [W]	421.71	421.71	392.27	392.27	416	416	348.59	347.73	325.97
Max Apparent Power - full node [VA]	534	534	479	479	527	527	464	394	387

Table 21: AP3 E4 A2 training benchmark E4 - A2

Experiments	0	1	2
Job ID	12531253	12531875	12531877
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	48	48	48
Total runtime [s]	44.58	44.62	44.5
Data loading overhead [s]	3.48	2.39	2.39
Total inference time [s]	40.61	41.84	42.42
Performance [GB/s]	1.72	1.67	1.65
Max CPU memory per MPI task [GB]	3.95	3.94	3.95
MAX GPU memory per MPI task[GB]	0.3	0.3	0.3
MAX Power [W]	123.66	123.88	133.58
Integrated Total Energy[Wh]	3.19	3.3	3.28

Table 22: AP3 JURECA-DC A100 inference benchmark A100

Experiments	0	1	2
Job ID	12548181	12548182	12548183
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	72	72	72
Total runtime [s]	44.58	44.07	44.29
Data loading overhead [s]	1.18	0.93	1.15
Total inference time [s]	41.93	41.9	41.95
Performance [GB/s]	1.67	1.67	1.66
Max CPU memory per MPI task [GB]	1.92	1.92	1.92
MAX GPU memory per MPI task[GB]	0.3	0.3	0.3
MAX Power [W]	114.12	114.16	112.44
Integrated Total Energy[Wh]	3.01	2.97	2.98

Table 23: AP3 JURECA-DC H100 inference benchmark H100

Experiments	0	1	2
Job ID	12543480	12543483	12543486
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	6	6	6
Total runtime [s]	43.25	42.95	43.06
Data loading overhead [s]	0.7	0.6	0.57
Total inference time [s]	41.79	41.66	41.79
Performance [GB/s]	1.67	1.68	1.66
Max CPU memory per MPI task [GB]	3.95	3.98	3.95
MAX GPU memory per MPI task[GB]	0.31	0.3 2	0.32
MAX Power [W]	154	155	154
Integrated Total Energy[Wh]	1.42	1.41	1.42

Table 24: AP3 JURECA-DC MI250 inference benchmark Mi250 - 1 GPU

Experiments	0	1	2
Job ID	12551023	12551024	12551025
#Nodes	1	1	1
#GPUs	8	8	8
#MPI Tasks	8	8	8
#CPUs per task	6	6	6
Total runtime [s]	43.32	45.06	45.13
Data loading overhead [s]	2.5	2.45	2.5
Total inference time [s]	41.83	41.82	41.81
Performance [GB/s]	1.67	1.67	1.67
Max CPU memory per MPI task [GB]	4.24	4.25	4.26
MAX GPU memory per MPI task[GB]	0.33	0.33 2	0.33
MAX Power [W]	272	278	273
Integrated Total Energy[Wh]	7.76	7.62	7.84

Table 25: AP3 JURECA-DC MI250 inference benchmark Mi250 - 8 GPUs

Experiments	0	1	2
Job ID	8386	8387	8362
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	32	32	32
Total runtime [s]	55.57	55.57	38.52
Data loading overhead [s]	5.15	5.17	1.57
Total inference time [s]	49	49.01	36.14
Performance [GB/s]	1.43	1.43	1.93
Max CPU memory per MPI task [GB]	3.79	3.76	3.78
MAX GPU memory per MPI task[GB]	0.31	0.31	0.31
Max Power [W]	47.24	45.26	44.5
Integrated Total Energy[Wh]	0.46	0.47	0.37
Max Power - Full Node [W]	410	419	434
Avg. Power - Full Node [W]	345.89	345.18	370.16
Max Apparent Power - Full Node [VA]	447	447	470

Table 26: AP3 E4 A2 inference benchmark E4 -A2

#Job ID	12501516	12501517	12501518	12501519	12501520	12501521	12501522	12501523	12501524	12501525	12501526	12501527
#Replicas	1	1	1	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	4	4	4	4	4	4
#Steps per execution	491	655	983	1966	3932	4000	122	163	245	491	983	1966
Dataset size [MB]	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03
Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	8.22	6.24	4.17	2.08	1.04	0.52
Effective Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	33.32	25	16.68	8.32	4.15	2.08
#Samples per batch	512	384	256	128	64	32	512	384	256	128	64	32
Training Time [s]	70.46	75.73	92.26	147.26	269.57	247.68	22.35	23.89	27.68	43.95	71.15	127.46
(NOn-first epoch)Performance [GB/s]	374.41	327.16	246.23	127.78	66.58	35.39	1528.5	1313.56	991.68	503.68	274	140.92
Best Epoch [s]	10.73	12.5	16.63	32.02	61.24	58.75	2.67	3.1	4.11	8.09	14.91	28.98
Max CPU memory	38.74	38.73	38.82	38.83	38.75	21.41	12.34	12.35	12.3	12.4	12.37	12.34
Max GPU memory	18.22	18.09	17.98	38.83	17.77	9.02	4.96	4.83	4.72	4.58	4.53	4.49
MAX Power [W]	125.53	185.77	164.43	121.42	96.78	128.52	155.96	126.77	132.52	112.49	107.08	109.28
Integrated Total Energy[Wh]	6.12	6.31	7.62	11.42	20.77	18.97	3.2	3.12	5.12	5.23	7.73	13.84

Table 27: AP3 Non-io Experiments - A100

#Job ID	8380	8381	8382	8383	8384	8385
#Replicas	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#Steps per execution	491	655	983	1966	3932	4000
Dataset size [MB]	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03
Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52
Effective Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52
#Samples per batch	512	384	256	128	64	32
Training Time [s]	51.81	50.89	62.38	107.11	179.98	187.92
(NOn-first epoch)Performance [GB/s]	528.52	447.09	344.08	180.39	48.76	94.76
Best Epoch [s]	7.55	.9.15	11.9	22.68	42.68	43.2
Max CPU memory	36.93	36.99	37.03	36.95	19.84	37.02
Max GPU memory	18.22	18.09	18	17.84	9	17.76
MAX Power [W]	160.93	151.38	140.79	122.71	112.28	116.23
Integrated Total Energy[Wh]	2.09	1.94	2.28	3.54	5.45	5.04
Mat Power - full node [W]	368	367	341	334	310	323
Avg. Power - full node [W]	338.41	333.79	325	315.95	302.91	308.12
Max Apparent Power - full node [VA]	382	380	354	349	326	339

Table 28: AP3 Non-io Experiments - Grace Hopper

#Job ID	12543346	12543347	12543348	12543349	12543350	12543351
#Replicas	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#Steps per execution	491	655	983	1966	3932	4000
Dataset size [MB]	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03
Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52
Effective Batch size [MB]	33.32	25	16.68	8.32	4.15	2.08
#Samples per batch	512	384	256	128	64	32
Training Time [s]	43.09	50.11	63.18	105.57	181.86	187.67
(NOn-first epoch)Performance [GB/s]	712.06	712.88	585.36	506.87	362.91	224.09
Best Epoch [s]	3.64	3.78	5.04	6.14	9.4	16.24
Max CPU memory	6.03	6	6.07	6.04	6.04	6.03
Max GPU memory	2.89	2.72	2.58	2.4	2.31	2.26
MAX Power [W]	193	188	173	171	158	149
Counter Total Energy[Wh]	4.46	4.29	5.02	5.74	7.86	11.95
Integrated Total Energy[Wh]	4.25	4.09	4.85	5.58	7.71	11.79

Table 29: AP3 Non-io Experiments - MI250 - 8 GPUS

#Job ID	12541427	12541428	12541429	12541430	12541431	12541432	12541433	12541434	12541435	12541436	12541437	12541438
#Replicas	1	1	1	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	4	4	4	4	4	4
#Steps per execution	491	655	983	1966	3932	4000	122	163	245	491	983	1966
Dataset size [MB]	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03
Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	8.22	6.24	4.17	2.08	1.04	0.52
Effective Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	33.32	25	16.68	8.32	4.15	2.08
#Samples per batch	512	384	256	128	64	32	512	384	256	128	64	32
Training Time [s]	56.27	61.14	74.66	119.12	196.86	193.55	18.44	20.47	23.84	35.15	56.19	100.36
(NOn-first epoch)Performance [GB/s]	493.29	403.82	305.05	168.44	93.75	46.53	1945.71	1550.63	1186.39	653.91	356.52	181.31
Best Epoch [s]	8.28	10.07	13.42	24.48	43.58	44.77	2.09	2.64	3.44	6.26	11.49	22.59
Max CPU memory	38.74	40.33	40.33	42.55	42.59	27.54	10.87	11.24	11.34	11.79	11.71	11.86
Max GPU memory	18.22	18.09	18	18.83	17.76	9	4.97	4.84	4.72	4.58	4.51	4.47
MAX Power [W]	133.57	124.15	116	102.45	97.86	94.06	133.17	122.78	113.66	100.41	95.67	93.05
Integrated Total Energy[Wh]	5.07	4.96	5.76	8.67	13.86	13.32	2.44	2.66	3.06	4.1	6.17	10.43

Table 30: AP3 Non-io Experiments - H100

#Job ID	12501559	12501562	12501565	12501568	12501571	12501574	12501561	12501564	12501567	12501570	12501573	12501576
#Replicas	1	1	1	1	1	1	4	4	4	4	4	4
#GPUs	1	1	1	1	1	1	1	1	1	1	1	1
#Steps per execution	491	655	983	1966	3932	4000	122	163	245	491	983	1966
Dataset size [MB]	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03	4091.03
Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	8.22	6.24	4.17	2.08	1.04	0.52
Effective Batch size [MB]	8.33	6.24	4.17	2.08	1.04	0.52	33.32	25	16.68	8.32	4.15	2.08
#Samples per batch	512	384	256	128	64	32	512	384	256	128	64	32
Training Time [s]	229.29	206.57	198.99	252.04	346.29	292.05	188.07	159.75	140.12	127.63	143.92	187.74
(NOn-first epoch)Performance [GB/s]	207.6	187.26	150.89	95.79	56.26	33.18	793.11	716.77	581.2	369.7	220.6	130.36
Best Epoch [s]	19.6	21.75	27.03	42.66	72.69	62.68	5.02	5.57	6.92	10.87	18.46	31.3
Max CPU memory	49.36	49.55	47.6	44.96	42.96	24.88	50.65	51.15	47.94	45.67	44.35	42.07
MAX Power [W]	112.1	82.6	98.6	74.6	75.4	68.1	106.4	84.1	79.1	92.9	78.1	71.2
Integrated Total Energy[Wh]	12.61	11.39	11.11	14.07	19.37	16.11	10.25	8.86	7.98	7.68	8.81	11.54

Table 31: AP3 Non-io Experiments - GRAPGHCORE IPU

Experiment number	Booster-SCRATCH	Booster-SCRATCH	Booster-SCRATCH	Booster-CSCRATCH	Booster-CSCRATCH	Booster-CSCRATCH
Job ID	6748650	6748654	6748658	6752412	6752591	6752592
#Nodes	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1
#CPUs	1	1	1	1	1	1
Loading data time	3,141.000	3,112.000	3,033.000	17433	16785	17,000.000
Total training time [s]	75.000	73.000	74.000	90.000	75.000	763.000
Total runtime [s]	3,305.000	3,260.000	3,158.000	17680	17010	17,914.000
Total training time	3,216.000	3,185.000	3,107.000	17523	16860	17,763.000
Avg. training time per epoch [s]	1,072.000	1,061.000	1,035.000	5841	5620	5,921.000
First epoch training time [s]	1,159.000	1,060.200	1,020.000	6238	5582	6,245.000
Min. training time per epoch	1,023.000	1,033.000	1,020.000	5618	5653	5,713.000
Max. training time per epoch	1,159.000	1,092.000	1,044.000	6238	5624	6,245.000
Avg. training time per iteration	5.60E-01	5.50E-01	5.40E-01	3.08	2.97	3.10E+00
Final training loss	4.90E-05	1.10E-05	1.30E-06	1.50E-05	3.50E-06	2.40E-06
Final validation loss	2.38E-06	5.00E-06	3.40E-07	3.20E-07	2.10E-06	9.60E-06
Saving model time	0.250	0.130	0.050	0.04	0.04	0.040
Max. GPU power	94.13	98.21	100.38	124.90	113.10	101.70
Avg. GPU Power	60.32	65.10	71.12	58.53	61.20	56.10
GPU Energy consumption [Wh]	55.37711111	58.95166667	62.38804444	287.4473333	289.17	279.1598333

Table 32: AP4 JUWELS Booster training benchmark

6.4 AP 4

Experiment number	SCRATCH	SCRATCH	SCRATCH	CSCRATCH	CSCRATCH	CSCRATCH
Job ID	6754748	6754749	6754750	6762119	6762120	6762121
#Nodes	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1
#CPUs	1	1	1	1	1	1
Loading data time	7,594.000	7,731.000	7,580.000	20,005.000	19,999.000	19,680.000
Total training time [s]	84.000	98.000	84.000	81.000	82.000	80.000
Total runtime	7,818.000	7,967.000	7,814.000	20,314.000	20,308.000	19,971.000
Total training time	7,678.000	7,829.000	7,664.000	20,086.000	20,081.000	19,760.000
Avg. training time per epoch	2,559.300	2,609.000	2,554.000	6,695.333	6,693.667	6,586.667
First epoch training time	2,559.000	2,689.000	2,559.000	6,712.000	6,685.000	6,585.000
Min. training time per epoch	2,430.000	2,409.000	2,455.000	6,668.000	6,685.000	6,585.000
Max. training time per epoch	2,689.000	2,731.000	2,649.000	6,712.000	6,698.000	6,588.000
Avg. training time per iteration	1.30E+00	1.33E+00	1.30E+00	3.54E+00	3.54E+00	3.48E+00
Final training loss	3.50E-06	3.40E-06	3.46E-06	1.60E-06	4.90E-06	9.30E-06
Final validation loss	2.00E-04	1.40E-06	1.18E-06	6.70E-06	4.30E-04	3.80E-06
Saving model time	0.830	0.220	0.750	0.030	0.070	0.100

Table 33: AP4 JUWELS Cluster training benchmark

Experiment number	1	2	3
Job ID	2384	2385	2386
#Nodes	1	1	1
#GPUs	1	1	1
#MPI tasks	1	1	1
#CPUs	16	16	16
Loading data time	8,686.000	6,794.000	6,582.000
Total training time [s]	68.000	68.000	59.000
Total runtime	8914	7,018.000	6,754.000
Total training time	8,754.000	6,862.000	6,641.000
Avg. training time per epoch	2,918.000	2,287.333	2,213.667
First epoch training time	3,215.000	2,240.000	2,207.000
Min. training time per epoch	2,769.000	2,240.000	2,207.000
Max. training time per epoch	3,215.000	2,348.000	2,237.000
Avg. training time per iteration	1.50E+00	1.20E+00	1.16E+00
Final training loss	3.80E-05	4.20E-06	2.40E-06
Final validation loss	4.30E-06	1.50E-06	8.90E-06
Saving model time	0.110	0.120	0.030

Table 34: AP4 E4 A2 training benchmark

JUBE WP	3	4	5	6	7	8	9	10	11
JobID	12579734	12579735	12579736	12579738	12579739	12579740	12579741	12579742	12579743
NodeID	jrc0384	jrc0352	jrc0256	jrc0255	jrc0384	jrc0352	jrc0256	jrc0255	jrc0256
MPI tasks	1	1	1	2	2	2	4	4	4
Total runtime [s]	3550.2	3588.8	3553.3	2691.3	2703.3	2690.7	1590.5	1566.0	1592.5
Training time	3478.7	3540.9	3506.5	2637.0	2646.9	2642.2	1538.1	1513.3	1533.6
avg. epoch time [s]	862.25	873.75	862.0	617.25	616.5	616.5	336.0	336.75	334.5
performance [MB/s]	492.772	492.983	488.606	652.318	649.867	651.028	1118.393	1136.667	1121.612
first epoch time [s]	923	939	927	659	661	653	384	387	383
min. epoch time [s]	838	843	836	599	596	602	315	314	313
max. epoch time [s]	923	939	927	659	661	653	384	387	383
avg. batch time [s]	0.287	0.290	0.286	0.410	0.410	0.410	0.447	0.448	0.445
loss	0.122	0.124	0.160	0.251	0.138	0.166	0.206	0.175	0.232
val loss	0.107	0.113	0.190	0.119	0.128	0.126	0.117	0.134	0.151
max. cpu mem	108.39	113.42	107.83	43.23	45.28	41.82	35.58	37.24	37.3
max. gpu mem	5.04	4.98	5.05	5.31	5.14	5.46	5.44	5.28	5.13
Integrated Total Energy [Wh]	396.67	396.3	401.09	345.72	344.54	346.58	283.68	275.65	283.3
max. power [W]	364.47	358.54	358.17	358.68	359.73	359.64	361.91	358.26	361.26
max. aggregate power [W]	537.82	536.0	537.02	813.7	809.48	818.66	1340.67	1327.99	1332.71
avg. aggregate power [W]	402.5	397.69	406.46	462.5	459.0	463.63	640.52	631.65	638.82

Table 35: AP1 JURECA NVIDIA A100 training benchmark

6.5 AP 5

JUBE WP	3	6	9	4	7	10
JobID	12598886	12598889	12598892	12598887	12598890	12598893
NodeID	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880	jrc0880
MPI tasks	1	1	1	2	2	2
Total runtime [s]	3789.2	4141.1	4011.7	2147.9	2018.9	2090.9
Training time	3705.2	4070.5	3946.1	2090.2	1971	2036.3
avg. epoch time [s]	926.25	1017.75	986.5	522.25	492.75	509
performance [MB/s]	469.709	425.733	442.793	828.676	878.801	850.621
first epoch time [s]	973	998	1005	547	541	563
min. epoch time [s]	877	998	889	489	466	470
max. epoch time [s]	973	1071	1080	555	541	563
avg. batch time [s]	0.308	0.338	0.328	0.347	0.328	0.338
loss	0.130	0.118	0.109	0.169	0.252	0.135
val loss	0.228	0.281	0.145	0.192	0.184	0.188
max. cpu mem	127.85	134.63	133.08	51.23	50.14	51.46
max. gpu mem	4.4	4.36	4.39	4.44	4.44	4.7
Total Energy [Wh]	377.84	402.01	393.17	284.78	275.88	281.42
max. power [W]	338.27	334.01	325.08	326.67	328.48	333.39
max. aggregate power [W]	488.42	483.92	475.59	695.26	694.17	
avg aggregate power [W]	358.71	349.19	352.51	476.82	491.64	483.91

Table 36: AP5 JURECA NVIDIA H100 training benchmark

JUBE WP	4	8	12	5	9	13	6	10	14	7	11	15
JobID	12600092	12600096	12600100	12600093	12600097	12600101	12600094	12602470	12602483	12601899	12601916	12600103
NodeID	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851	jrc0851
MPI tasks	1	1	1	2	2	2	4	4	4	8	8	8
total runtime [s]	4261.6	4258.9	4276.3	2439.7	2448.6	2474.4	1499.3	1568.1	1602.9	1285	1290.2	1343.8
total training time [s]	4225.8	4231.1	4246.7	2405.1	2412.7	2443.5	1464.5	1525.8	1569.2	1196.2	1247.2	1257.5
avg. epoch time [s]	1056.75	1057.75	1061.75	601.25	603	610.75	366	381.5	392.25	298.75	311.75	314.75
performance [MB/s]	412.459	412.753	410.413	715.205	712.947	703.976	1174.576	1124.067	1092.988	1472.797	1433.649	1421.939
first epoch time [s]	1106	1101	1108	641	644	653	416	417	479	357	390	410
min. epoch time [s]	1034	1039	1041	584	583	595	343	366	360	277	282	271
max. epoch time [s]	1106	1101	1108	641	644	653	416	417	479	357	390	410
avg. batch time [s]	0.351	0.352	0.353	0.400	0.401	0.406	0.487	0.507	0.522	0.795	0.829	0.837
loss	0.127	0.197	0.127	0.132	0.1356	0.152	0.184	0.208	0.140	0.185	0.219	0.220
val loss	0.164	0.115	0.241	0.176	0.166	0.167	0.136	0.145	0.165	0.191	0.178	0.196
max. cpu mem	121.81	121.34	121.71	44.36	44.78	44.41	38.08	36.97	37.17	25.05	25.05	25.05
max. gpu mem	3.42	3.4	3.39	3.45	3.49	3.43	3.45	3.48	3.43	3.39	3.55	3.49
Total Energy [Wh]	660.66	660.53	662.65	474.38	475.53	478.25	383.43	391.37	395.42	365.36	365.89	371.6
max. power [W]	314	313	311	516	515	512	512	510	509	479	479	487
max. aggregate power [W]	585	586	583	787	786	783	1183	1189	1189	1892	1888	1853
avg aggregate power [W]	558.08	558.31	557.85	699.99	699.17	695.82	920.62	898.47	888.16	1022.39	1019.82	994.21

Table 37: AP5 JURECA AMD MI250x training benchmark

JUBE WP	1	2	3
JobID	8631	8627	8620
NodeID	ngnode02	ngnode02	ngnode02
MPI tasks	1	1	1
total runtime [s]	1759.0	1847.2	1843.6
total training time [s]	1740.2	1825.7	1813.2
avg. epoch time [s]	435.25	456.25	453.25
performance [MB/s]	974.573	936.664	948.789
first epoch time [s]	451	459	463
min. epoch time [s]	412	434	443
max. epoch time [s]	451	474	463
avg. batch time [s]	0.148	0.141	0.142
loss	N/A	N/A	N/A
last recon. loss val	N/A	N/A	N/A
max. cpu mem	110.18	114.45	118.79
max. gpu mem	3.69	3.7	3.68
Total Energy [Wh]	156.99	168.63	169.28
max. power [W]	443.79	634.07	500.86
max. aggregate power [W]	443.79	634.07	500.86
avg aggregate power [W]	346.79	335.58	335.05
System avg power [W]	604.02	586.09	590.05
System avg VA [W]	657.57	640.49	645.47
System total VA [Wh]	321.30	328.64	330.55

Table 38: AP5 E4 NVIDIA GraceHopper H200 training benchmark

JUBE WP	1	2	3
JobID	8649	8657	8658
NodeID	icnode04	icnode04	icnode04
MPI tasks	1	1	1
total runtime [s]	19766.6	19694.2	19556.1
total training time [s]	19727.3	19655.5	19517.3
avg. epoch time [s]	4931.8	4914.0	4879.3
performance [MB/s]	85.49	85.12	85.74
first epoch time [s]	5043	5023	4988
min. epoch time [s]	4892	4873	4838
max. epoch time [s]	5043	5023	4988
loss	N/A	N/A	N/A
val loss	N/A	N/A	N/A
max. cpu mem	104.77	103.51	103.04
max. gpu mem	5.28	5.08	5.05
Total Energy [Wh]	321.10	319.37	317.35
max. power [W]	443.79	634.07	500.86
max. aggregate power [W]	61.99	62.08	62.16
avg aggregate power [W]	58.48	58.38	58.42
System avg power [W]	391.06	392.70	394.09
System avg VA [W]	426.79	431.08	429.77
System total VA [Wh]	2343.40	2358.29	2334.63

Table 39: AP5 E4 NVIDIA A2 training benchmark

Experiment number	1	2	3
Data location	/data	/data	/data
Job ID	2166	2167	2168
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	40	40	40
Total runtime [s]	152.02	81.42	77.36
Model loading [s]	2.99	4.52	2.77
Data loading [s]	110.75	60.61	59.15
Total inference time [s]	43.15	16.19	15.39
Performance [GB/s]	0.10	0.26	0.27
Max CPU memory per MPI task [GB]	31.71	31.7	31.91
MAX GPU memory per MPI task[GB]	5.61	5.61	5.61
Node ID	icnode01	icnode02	icnode01

Table 40: AP5 E4 A2 inference runtime.

Experiment number	1	2	3
Data location	/data	/data	/data
Job ID	2169	2170	2171
#Nodes	1	1	1
#GPUs	1	1	1
#MPI Tasks	1	1	1
#CPUs per task	40	40	40
Total runtime [s]	68.81	70.33	68.41
Model loading [s]	2.44	3.96	2.01
Data loading [s]	51.45	51.71	51.58
Total inference time [s]	14.83	14.60	14.82
Performance [GB/s]	0.28	0.29	0.28
Max CPU memory per MPI task [GB]	31.97	31.44	31.58
MAX GPU memory per MPI task[GB]	5.27	5.27	5.27
Node ID	acnode02	acnode01	acnode02

Table 41: AP5 E4 AMD System inference benchmark

6.6 AP 6

Experiment number	1	2	3	4	5
Job ID	7427176	7427177	7421256	7421257	7421258
#Nodes	1	1	1	2	4
#GPUs	1	2	4	4	4
#CPUs	48	48	48	48	48
Total runtime [s]	9,496.58	12,658.78	15,031.87	9,749.03	5,278.38
Total training time [s]	9,496.58	12,658.78	15,031.87	9,749.03	5,278.38
Avg. training time per epoch [ms]	1.84	4.59	9.72	10.24	5.34
First epoch training time [ms]	2.16	3.70	8.78	8.29	8.28
Final training loss	1.94E+00	1.85E+00	1.95E+00	1.95E+00	1.94E+00
Node ID	jwb1246	jwb1247	jwb1077	jwb[0985,0991]	jwb[0578,0588,0608,1034]
Max. GPU power	332.95	321.57	330.04	210.03	220.54
Avg. GPU power	68.56	72.72	86.58	81.23	81.82
GPU energy consumption [Wh]	180.86	255.71	361.52	219.98	119.97

Table 42: AP6 JUWELS Booster training benchmark

Experiment number	1	2	3	4	5
Job ID	7427182	7427184	7421259	7421260	7421261
#Nodes	1	1	1	2	4
#GPUs	1	2	4	4	4
#CPUs	48	48	48	48	48
Total runtime [s]	16,266.83	17,029.43	18,581.30	11,343.47	5,763.21
Total training time [s]	16,266.83	17,029.43	18,581.30	11,343.47	5,763.21
Avg. training time per epoch [ms]	3.17	6.35	12.59	13.37	6.71
First epoch training time [ms]	3.01	5.17	9.65	6.76	8.59
Final training loss	1.95E+00	1.92E+00	1.95E+00	1.95E+00	1.93E+00
Node ID	jwc09n117	jwc09n096	jwc09n183	jwc09n[069,084]	jwc09n[087,090,093,099]
Max. GPU power	291.17	286.18	247.71	210.03	223.31
Avg. GPU power	55.17	62.67	76.60	71.70	69.16
GPU energy consumption [Wh]	249.29	296.45	395.37	225.92	110.72

Table 43: AP6 JUWELS Cluster training benchmark

Experiment number	1	2
Dataset ID	2	2
Job ID	2058	2140
#Nodes	1	2
#GPUs	1	2
Total runtime [s]	6,240.740	3,936.077
Total training time	6,240.740	3,936.077
Avg. training time per epoch	2,458.308	2,844.843
Final training loss	1.90E+00	1.91E+00
Node ID	icnode01	icnode[01-02]
Avg. Power Consumption [W]	600.61	599.09
Avg. apparent Power [VA]	620.36	617.85
GPU energy consumption [Wh]	1,041.18	1,310.04
Action [MJs]	23,391.89	18,563.09

Table 44: AP6 E4 A2 training benchmark

Document History

Version	Author(s)	Date	Changes
	Name (Organisation)	dd/mm/yyyy	
1.0	FZJ	04/03/2024	Final version

Internal Review History

Internal Reviewers	Date	Comments
Peter Dueben (ECMWF)	29/02/2024	Accepted with minor revisions

Estimated Effort Contribution per Partner

Partner	Effort
E4	2PM
FZJ	9PM
MetNor	2PM
ECMWF	2PM
ETHZ	1PM
4-cast	1PM
Total	17PM