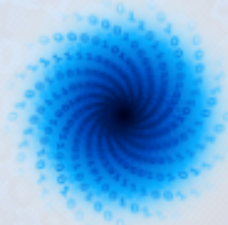




MACHinE Learning for Scalable meTeoROlogy and climate



MAELSTROM

Initial hardware performance benchmarking

Forschungszentrum Jülich GmbH, JSC

www.maelstrom-eurohpc.eu



D3.4 Report on hardware performance benchmarking for simplistic ML solutions for benchmark data sets in D1.2 on existing hardware solutions

Author(s):

Forschungszentrum Jülich GmbH, JSC

Stepan Nassyr (FZJ), Andreas Herten (FZJ)

Dissemination Level:

Public

Date:

April 11, 2022

Version:

1.0

Contractual Delivery Date:

31/03/2022

Work Package/ Task:

WP3/ T3.3

Document Owner:

FZJ

Contributors:

MetNor,ECMWF,ETHZ,4-cast

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:

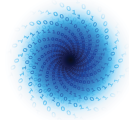
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom

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 Union's Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway.

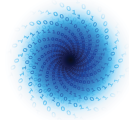


EuroHPC
Joint Undertaking



Contents

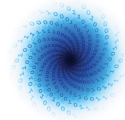
1	Executive Summary	8
2	Introduction	9
2.1	About MAELSTROM	9
2.2	Scope of this deliverable	9
2.2.1	Objectives of this deliverable	9
2.2.2	Work performed in this deliverable	10
2.2.3	Deviations and counter measures	10
3	Metrics	11
4	Benchmarks	13
4.1	AP1	15
4.1.1	Notes	15
4.1.2	JUWELS Booster	16
4.1.3	JUWELS Cluster	18
4.1.4	E4 Intel+NVIDIA	20
4.1.5	Results	22
4.2	AP2	23
4.3	AP3	24
4.3.1	Notes	24
4.3.2	JUWELS Booster	25
4.3.3	JUWELS Cluster	29
4.3.4	E4 Intel+NVIDIA	32
4.3.5	JUWELS Booster Inference	33
4.3.6	Results	35
4.4	AP4	36
4.4.1	Notes	36
4.4.2	JUWELS Booster	37
4.4.3	JUWELS Cluster	39
4.4.4	E4 Intel+NVIDIA	40
4.4.5	Results	41
4.5	AP5	42
4.5.1	Notes	42
4.5.2	JUWELS Booster (Small Dataset)	43
4.5.3	JUWELS Booster (Large Dataset)	45



4.5.4	JUWELS Cluster	46
4.5.5	E4 Intel+NVIDIA	48
4.5.6	JUWELS Booster Inference	50
4.5.7	Results	51
4.6	AP6	52
4.6.1	Notes	52
4.6.2	JUWELS Booster	53
4.6.3	JUWELS Cluster	53
4.6.4	E4 Machines	54
4.6.5	Results	54
5	Conclusion	55
6	Appendix	56
6.1	AP1	57
6.2	AP3	59
6.3	AP4	64
6.4	AP5	65
6.5	AP6	69

List of Figures

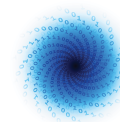
1	AP1 JUWELS Booster Runtime	16
2	AP1 JUWELS Booster Epoch Time	17
3	AP1 JUWELS Booster Energy	17
4	AP1 JUWELS Cluster Runtime	18
5	AP1 JUWELS Cluster Epoch Time	18
6	AP1 JUWELS Cluster Energy	19
7	AP1 E4 Intel+NVIDIA Runtime	20
8	AP1 E4 Intel+NVIDIA Epoch Time	21
9	AP3 JUWELS Booster Runtime	25
10	AP3 JUWELS Booster Epoch Time	26
11	AP3 JUWELS Booster Energy	27
12	AP3 JUWELS Cluster Runtime	29
13	AP3 JUWELS Cluster Epoch Time	30
14	AP3 JUWELS Cluster Energy	30
15	AP3 E4 Intel+NVIDIA Runtime	32
16	AP3 E4 Intel+NVIDIA Epoch Time	33
17	AP3 JUWELS Booster Inference Runtime	33



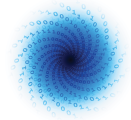
18	AP4	JUWELS Booster	Runtime	37
19	AP4	JUWELS Booster	Epoch Time	38
20	AP4	JUWELS Booster	Energy	39
21	AP4	E4 Intel+NVIDIA	Runtime	40
22	AP4	E4 Intel+NVIDIA	Epoch Time	41
23	AP5	JUWELS Booster	Runtime (Small Dataset)	43
24	AP5	JUWELS Booster	Epoch Time (Small Dataset)	44
25	AP5	JUWELS Booster	Energy (Small Dataset)	44
26	AP5	JUWELS Booster	Runtime (Large Dataset)	45
27	AP5	JUWELS Booster	Epoch Time (Large Dataset)	46
28	AP5	JUWELS Booster	Energy (Large Dataset)	46
29	AP5	JUWELS Cluster	Runtime	47
30	AP5	JUWELS Cluster	Epoch Time	47
31	AP5	JUWELS Cluster	Energy	48
32	AP5	E4 Intel+NVIDIA	Runtime	49
33	AP5	E4 Intel+NVIDIA	Epoch Time	49
34	AP5	JUWELS Booster	Inference Runtime	50
35	AP6	JUWELS Booster	Runtime	53
36	AP6	JUWELS Cluster	Runtime	53
37	AP6	E4	Runtimes	54

List of Tables

1	AP1	JUWELS Booster	training benchmark	57
2	AP1	JUWELS Cluster	training benchmark	57
3	AP1	E4 Intel+NVIDIA	training benchmark	58
4	AP3		Experiment flag reference	60
5	AP3	JUWELS Booster	training benchmark	60
6	AP3	JUWELS Cluster	training benchmark	61
7	AP3	E4 Intel+NVIDIA	training benchmark	62
8	AP3	JUWELS Booster	inference benchmark	63
9	AP4	JUWELS Booster	training benchmark	64
10	AP4	E4 Intel+NVIDIA	training benchmark	64
11	AP5	JUWELS Booster	training benchmark (small dataset)	66
12	AP5	JUWELS Booster	training benchmark (large dataset)	66
13	AP5	JUWELS Cluster	training benchmark (large dataset)	67
14	AP5	E4 Intel+NVIDIA	training benchmark (large dataset)	67
15	AP5	JUWELS Booster	inference benchmark	68

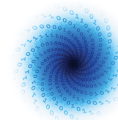


16	AP6 JUWELS Booster benchmark	69
17	AP6 JUWELS Cluster benchmark	69
18	AP6 E4 benchmarks	69



1 Executive Summary

This early benchmarking serves to establish a performance baseline for the MAELSTROM Applications in their early development stage. Multiple relevant performance metrics were agreed upon in cooperation between WP1 and WP3. Benchmarks are designed to investigate how the runtime of the applications is distributed, making it possible to identify what first improvements should focus on, and which parts of the hardware systems are utilized to what degree. Another aspect that was surveyed is how well the applications in their early stage are able to make use of currently available hardware with regard to available compute throughput, parallelism, as well as energy efficiency.



2 Introduction

2.1 About MAELSTROM

To develop Europe's computer architecture of the future, MAELSTROM will co-design bespoke compute system designs for optimal application performance and energy efficiency, a software framework to optimise usability and training efficiency for machine learning at scale, and large-scale machine learning applications for the domain of weather and climate science.

The MAELSTROM compute system designs will benchmark the applications across a range of computing systems regarding energy consumption, time-to-solution, numerical precision and solution accuracy. Customised compute systems will be designed that are optimised for application needs to strengthen Europe's high-performance computing portfolio and to pull recent hardware developments, driven by general machine learning applications, toward needs of weather and climate applications. The MAELSTROM software framework will enable scientists to apply and compare machine learning tools and libraries efficiently across a wide range of computer systems. A user interface will link application developers with compute system designers, and automated benchmarking and error detection of machine learning solutions will be performed during the development phase. Tools will be published as open source.

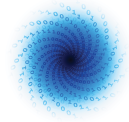
The MAELSTROM machine learning applications will cover all important components of the workflow of weather and climate predictions including the processing of observations, the assimilation of observations to generate initial and reference conditions, model simulations, as well as post-processing of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

2.2 Scope of this deliverable

2.2.1 Objectives of this deliverable

Deliverable 3.4 is a report on the work done for Task 3.3, the initial benchmarking of simplistic ML solutions on pre-existing hardware.

Deliverable 3.4 is one of two MAELSTROM deliverables that provide the basis for benchmarking the applications on HPC hardware. Deliverable 3.3 reports on the



available benchmarking and test infrastructure, including the available HPC systems on which the benchmarks are performed. Deliverable 3.4 presents the results of initial benchmarks of the existing ML solutions on the available systems.

2.2.2 Work performed in this deliverable

Metrics relevant for performance were discussed and agreed upon together with the WP1 applications developers. This resulted in a spreadsheet that was provided to the application developers to insert results of their benchmarks on available HPC machines.

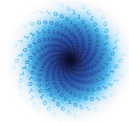
Access to available resources at JSC and E4 has been provided to application developers. Information on how to access the system, execute benchmarks, and measure the metrics has been compiled and documented on the project's Confluence installation, where it was made available for all members of the project.

Benchmarks were performed by the application developers, the results were documented in the spreadsheet.

An analysis of the initial performance results has been performed to investigate initial performance, scaling behaviour, energy efficiency, and potential issues.

2.2.3 Deviations and counter measures

There are no significant deviations from the planned contributions of the deliverable.



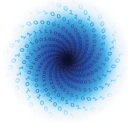
3 Metrics

The following metrics have been decided on for all applications:

- Time-related
 - Total runtime
 - Total training 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

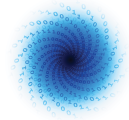
The metrics chosen are categorized into time-, model- and energy-related metrics. From the general benchmarking perspective, metrics such as total runtime and runtime distribution are relevant, such as total runtime and training time, times for loading and storing data. Further, timing metrics are provided by the ML frameworks themselves, such as the epoch training time. Additionally, in a discussion with application developers, some other metrics have been brought forward that are important from the ML side - such as final training and validation loss. Finally, in order to measure energy efficiency, GPU power and energy consumption was recorded.

Most metrics allow for an early analysis of the data. However - the training and validation loss mostly serve as a reference for future benchmarks after more performance optimizations have been implemented.



The infrastructure for measuring GPU and full-node energy consumption is currently under development, however some tools are already available. On JSC systems, GPU power consumption is provided by the LLview¹ job report - internally, `nvidia-smi` is used to measure GPU power periodically during the job execution. On E4 systems, a solution using NVIDIA tools is currently in the works. Additionally, on E4 systems, the node power consumption is recorded periodically - a tool for automatic extraction of this data is in development at the point of writing, therefore this metric has not been recorded for this deliverable yet.

¹https://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/LLview/_node.html



4 Benchmarks

Applications were mainly benchmarked on 3 different systems: JUWELS Booster, JUWELS Cluster and E4's Intel+NVIDIA cluster. AP6 also ran some tests on other E4 hardware, including a second Intel CPU-based and an AMD CPU-based cluster. The focus was on training benchmarks, however for applications 3 and 5 also inference performance was noted. For some applications, multiple configurations were investigated. For each configuration, at least 3 runs were performed. In cases where inconsistencies were found in the metrics of the first 3 runs, the developers were asked to perform more measurements.

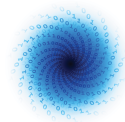
Apart from the metrics listed in Section 3, also the job information was recorded for each job. This also allows to query job-specific information at a later stage. The following information is recorded:

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

At this stage, no application has performed a benchmark using more than 1 node and the except for application 3, which experiments using 2 GPUs, all applications still use only one GPU. As the applications scale to multiple GPUs and nodes in the future, the single-node, single-GPU performance will serve as a baseline.

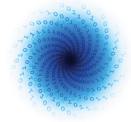
For each of the applications an overview of the application is given, including the following characteristics of the application:

- Memory training dataset
- Memory validation dataset
- Training samples
- Input shape sample
- batch size
- Trainable parameters



- Non-trainable parameters
- Loss function
- Experimental notes

As well as any special characteristics of the application. Benchmark data is evaluated and first interpretations are given. The total raw data is given in the appendix.



4.1 AP 1

4.1.1 Notes

Memory training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
20.9 GB	5.2 GB	128	[256,256,784]	8

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
194264	0	Quantile score (10,50,90%)	2 days of training data, 64 patches per day

Data formats	Frameworks (to be) used
NetCDF	TensorFlow2,Flux

The raw data discussed in this section can be found as tables in appendix 6.1.



4.1.2 JUWELS Booster

4.1.2.1 Runtime

One of the goals of this early benchmarking is to identify where potential performance bottlenecks lie for the applications. For this reason, the developers were asked to separately measure the runtime for loading the data and training, as well as the total runtime. With this information, the relative share of I/O and training loads as well as any unaccounted time can be identified.

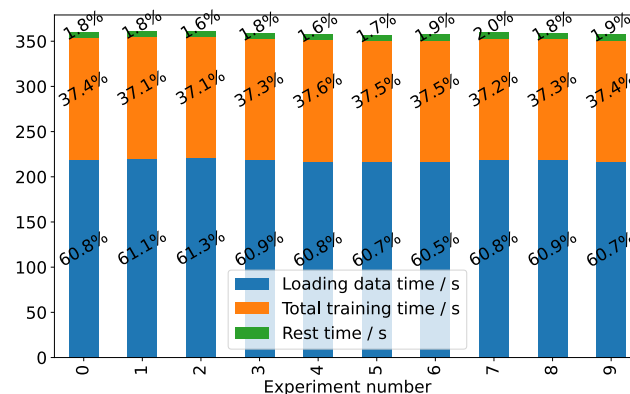


Figure 1: **AP1 JUWELS Booster** Runtime: Runtime and relative share for multiple experiments *ap1-jwb-runtime-share*

As seen in Figure 1, the majority of the runtime, around 61 %, is spent on loading the training data into memory. Loading time is not reduced in subsequent experiments, possibly due to filesystem caches not. A small percentage, 1.8 % to 2.0 %, is spent outside training and I/O.

Training one epoch takes 26.81 ± 0.05 s on average. The first epoch takes on average 1.65 ± 0.05 times longer than the average epoch training time, revealing some initialization overhead or a caching effect – see Figure 2. There is also another trend visible - the first 3 experiments have a worse ratio than the rest, the ratio decreasing monotonously from 1.75 in the first experiment to 1.69 in the 3rd.

4.1.2.2 GPU power consumption

The GPU energy consumption is consistent across the experiments (Figure 3), with an average of 5.98 ± 0.38 W h used per run. The maximum GPU power of 60 W-82 W indicates that the GPUs are not properly utilized, or the measurement is failing to truthfully capture the max. GPU power. Investigating the job reports, the GPU

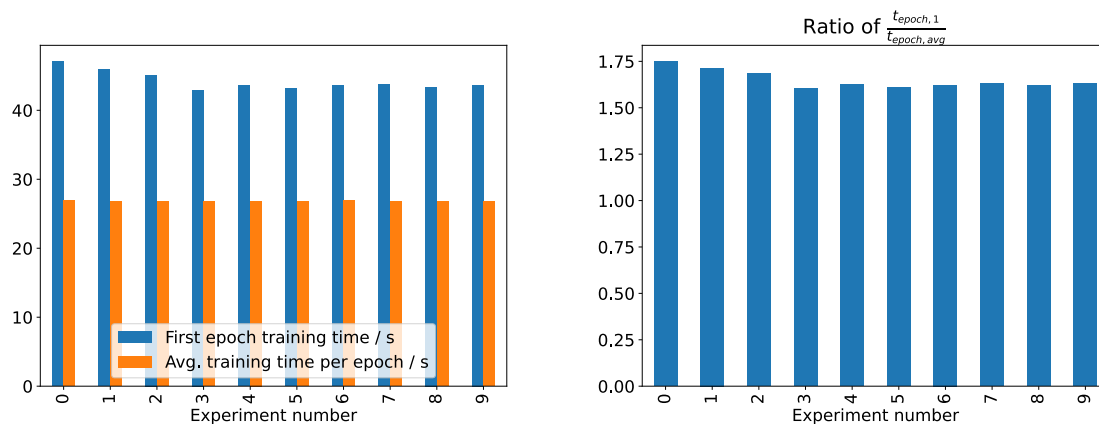
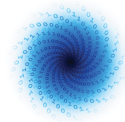


Figure 2: **AP1 JUWELS Booster** Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right)
ap1-jwb-epoch-time

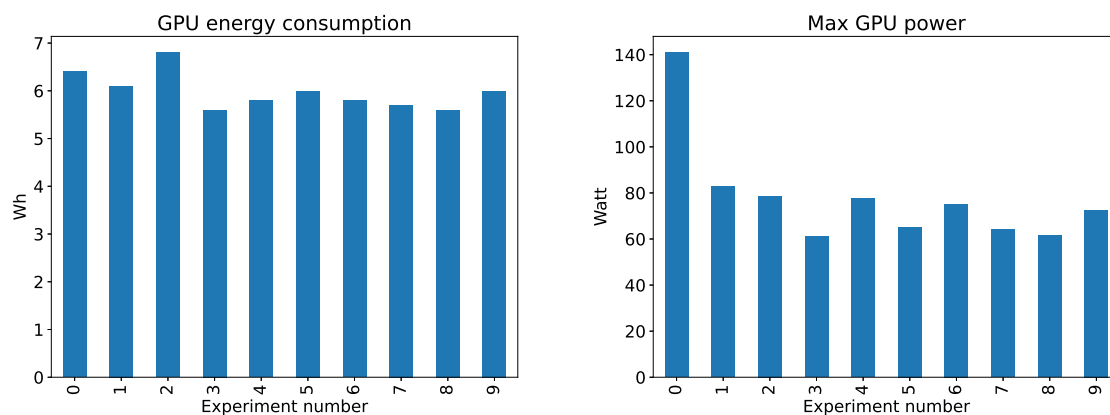
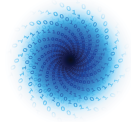


Figure 3: **AP1 JUWELS Booster** Energy: Total GPU energy consumption (left); peak GPU power draw (right) *ap1-jwb-energy*

utilization stays below 10 % and is in multiple cases reported as 0 %. Together with the short total runtime of roughly 6 min, it is fair to conclude that the benchmark is too short to properly measure GPU utilization and power usage. Investigating this further will require increasing the runtime of the test case.



4.1.3 JUWELS Cluster

4.1.3.1 Runtime

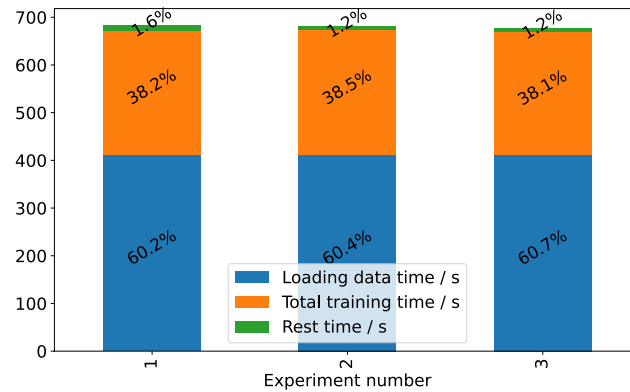


Figure 4: **AP1 JUWELS Cluster** Runtime: Runtime and relative share for multiple experiments *ap1-jwc-runtime-share*

For the runtime on JUWELS Cluster in Figure 4, we see a rough doubling of both the training and loading data time compared to JUWELS Booster. This can be an indicator of the data loading time scaling with the GPU memory bandwidth instead of available file I/O. It can however also mean that data loading and training are strongly coupled, and the measurement can not properly separate the two. It is necessary to investigate and, if necessary, improve the time measurements.

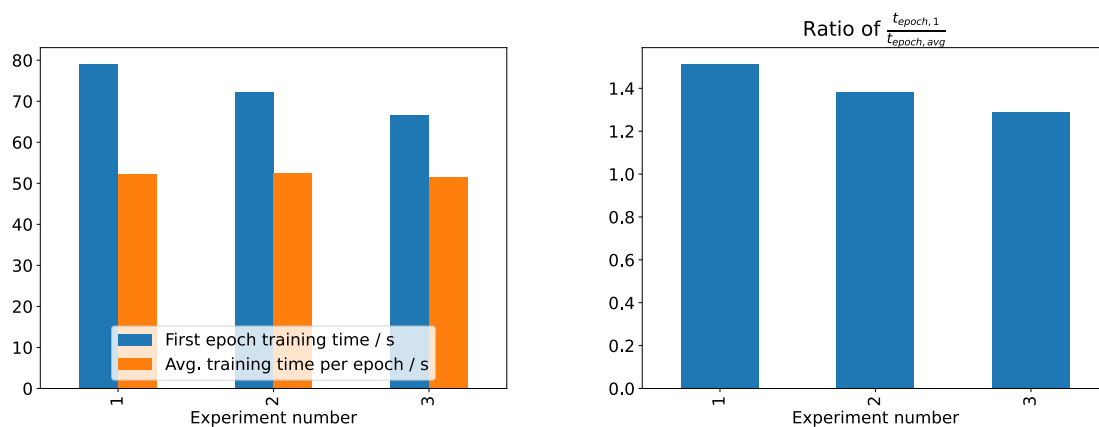
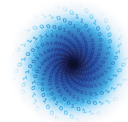


Figure 5: **AP1 JUWELS Cluster** Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right) *ap1-jwc-epoch-time*



Training one epoch takes 52.1 ± 0.4 s on average. The first epoch takes on average 1.40 ± 0.11 times longer than the average epoch training time (see Figure 5). A rough doubling of the training time is consistent with the increase in the total training time. The lower ratio of first vs average epoch training time shows that the overhead is at least to some degree decoupled from the training performance of the underlying hardware.

4.1.3.2 GPU power consumption

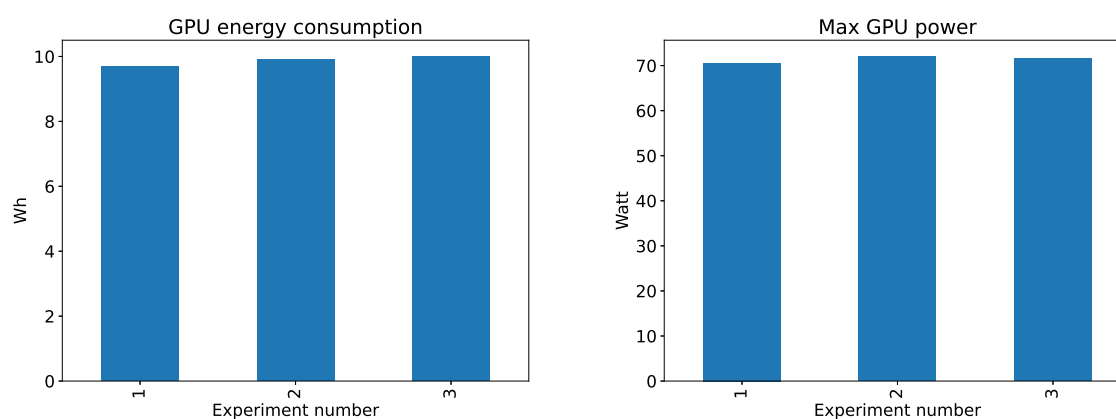
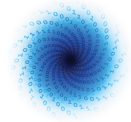


Figure 6: **AP1 JUWELS Cluster** Energy: Total GPU energy consumption (left); peak GPU power draw (right) *ap1-jwc-energy*

Similarly to JUWELS Booster, the GPU power usage is consistent across experiments, as can be seen in Figure 6. The average energy consumption however is higher with on average 9.87 ± 0.15 Wh used per run, 1.65 times more than on the Booster. Just as on the Booster, the reported max GPU power is low with roughly 70 W and the job reports similarly show low GPU utilization. Scaling the benchmark up or improving how GPU power is measured is required.



4.1.4 E4 Intel+NVIDIA

4.1.4.1 Runtime

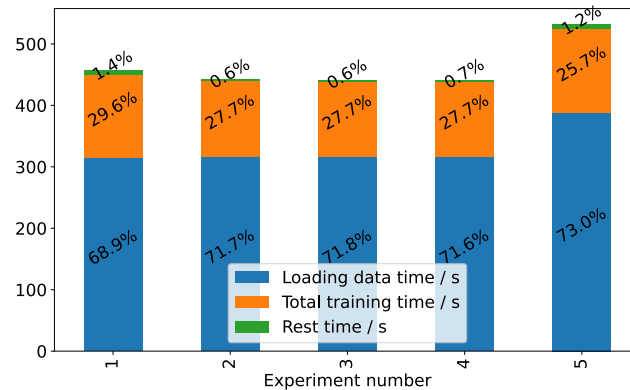


Figure 7: **AP1 E4 Intel+NVIDIA** Runtime: Runtime and relative share for multiple experiments *ap1-e4i-runtime-share*

Training on the Intel+NVIDIA cluster of E4 takes, as expected for similar GPUs², roughly the same amount of time as on JUWELS Booster – see Figure 7. In 3 out of 5 experiments, the training time is 10 % lower. Since the difference is significant, there is value in investigating whether and under which conditions the increased performance can be reproduced. The difference can also be attributed to system differences, i.e. a different cooling solution.

The data loading time is roughly 50 % higher than on JUWELS Booster. A possible explanation is the filesystem backend - JUWELS Booster uses GPFS (IBM Spectrum Scale), while on E4 the NFS filesystem is used.

As shown in Figure 8, the 10 % performance uplift is present in the epoch training time for the same experiments as in the total training time. It makes sense to group the experiments with similar performance.

Experiments 1 and 5 have on average an epoch training time of 27.19 ± 0.17 s and a $\frac{t_{epoch,1}}{t_{epoch,avg}}$ ratio of 2.199 ± 0.015 .

Experiments 2 through 4 have on average an epoch training time of 24.46 ± 0.06 s and a $\frac{t_{epoch,1}}{t_{epoch,avg}}$ ratio of 1.891 ± 0.006 .

The decreased ratio for the better performing experiments shows that the boost shortens the overhead stronger than the training time.

²The GPU installed in the cluster at E4 is the PCIe version of the A100 GPU, while in JUWELS Booster SXM4 modules are installed. Among other differences, the SXM4 version offers a higher thermal design power.

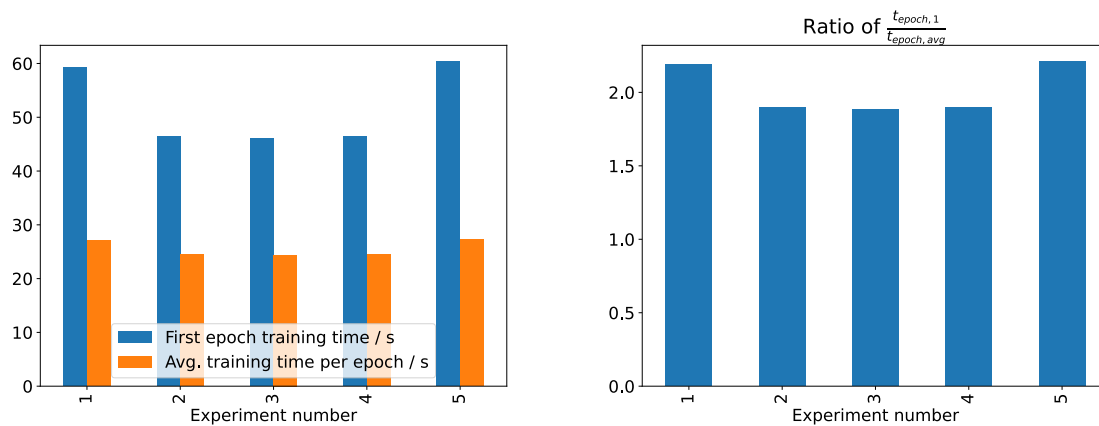
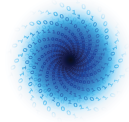
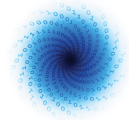


Figure 8: **AP1 E4 Intel+NVIDIA** Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right)
apl-e4i-epoch-time

A difference in the performance for the first epoch is observable between JUWELS Booster and the E4 Intel+NVIDIA cluster, the E4 cluster taking longer. Possibly, some delayed I/O is involved in the first step of the training.

4.1.4.2 GPU power consumption

On E4 only one experiment was performed with GPU power measurement. The energy consumption measured was 6 W h and max. GPU power reported was 232.4 W. The energy consumption is similar to JUWELS Booster. The higher max. GPU power could be an indicator of manual measurement with nvidia-smi being more accurate for smaller jobs, but it should be noted that only one data point was recorded as the tool for automatic GPU power measurement on E4 is still in development.

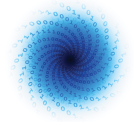


4.1.5 Results

The initial numbers for AP1 show that the majority of the total runtime on all systems is spent on loading the data - between 60 % and 73 % depending on the underlying system. There are indicators for the performance being bound by both filesystem performance as well as GPU memory bandwidth. A deeper investigation into the I/O and memory performance is necessary.

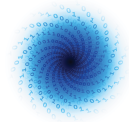
Training time can be observed to decrease by $\approx 50\%$ going from the less powerful V100 GPUs in JUWELS Cluster to the A100 GPUs in JUWELS Booster and E4's Intel+NVIDIA cluster. We have also occasionally observed a 10 % higher performance on the E4 machine compared to JUWELS Booster, which presents an interesting point of future investigation between the two systems concerning the circumstances in which the performance boost can be reproduced.

Because of the benchmark's short length, GPU power and utilization measurements appear unreliable. It is necessary to either scale up the benchmark or improve the measurement methods. It is also necessary to scale up the benchmark to investigate how the proportion of loading data time scales with longer benchmark duration and whether the application remains I/O-bound.



4.2 AP 2

Work on AP2 was faced with challenges regarding data acquisition and a staff bottleneck. The dedicated personnel will only start working on it from April 2022. Therefore, benchmarking data for AP2 could not be provided until the deadline of the deliverable.



4.3 AP 3

4.3.1 Notes

Memory 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

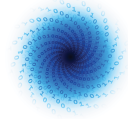
Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
261 515	0	MSE (multiple output vectors)	Model not trained to convergence for cost reasons (only 5 epochs), 50 epochs required

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

For AP3 experiments were performed with different flags supplied to the application. The configurations are:

- *None*: Default version without special flags
- `--synthetic_data`: Use synthetic data (repeated sample)
- `--nocache`: Avoid using TensorFlow dataset cache
- `--gpus 2`: Use 2 GPUs instead of 1
- `--batch 1024`: Use a batch size of 1024
- `--batch 1024 --gpus 2`: Use a batch size of 1024 and 2 GPUs

We know from a discussion with the application developers, that for this application, data is loaded/streamed in during training, which has as a consequence that the default approach of measuring the elapsed time for code sections loading data and performing training does not capture the full I/O load in the loading runtime. This can be observed in experiments where the I/O load varies with different flags. Raw data for graphs and discussions of this section is listed in appendix 6.2.



4.3.2 JUWELS Booster

4.3.2.1 Runtime

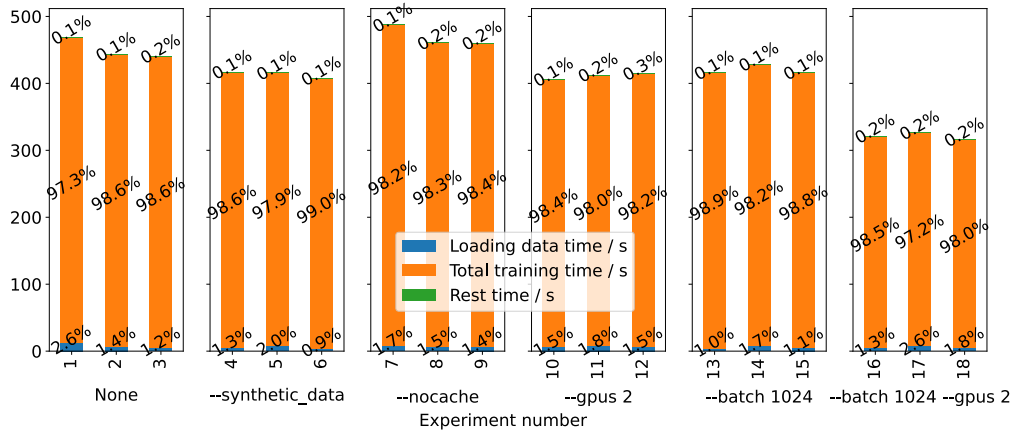


Figure 9: **AP3 JUWELS Booster Runtime: Runtime and relative share for multiple experiments** *ap3-jwb-runtime-share*

As seen in the left-most plot (*None*) of Figure 9, the main part of the runtime (98 % to 99 %) of AP3 is taken up by training, 442.62 ± 11.78 s on average. However, from discussions with the application developers, it is known that the application streams in data during training. This is also evident from comparing experiments with and without the `--synthetic_data` flag. When synthetic data is used, the amount of data loaded is reduced, as this data essentially just repeats samples. We can see a reduction of training time when using this flag by on average 8 %, corroborating our knowledge of data being streamed in during training.

AP3 also tested extending GPU usage from 1 GPU to 2 GPUs. A speed-up factor of just 1.095 can be seen for the default benchmark when using 2 GPUs for a parallel efficiency³ of $\epsilon_{\text{par}} = 0.548$. Increasing the batch size to 1024 improves the performance, resulting in a speed-up $1.406 \times$ and parallel efficiency of $\epsilon_{\text{par}} = 0.703$. Solely increasing the batch size while using only 1 GPU results in a speed-up of $1.068 \times$. It appears that the default batch size (512) is too small to occupy 2 GPUs efficiently; a benefit can only be seen with a larger size.

Foregoing the TensorFlow data cache results in factor of 0.956 lower performance compared to the baseline default performance.

Studying runtimes of first epochs and all epochs in Figure 10, an average training

³ $\epsilon_{\text{par}} = \frac{1}{N} \frac{t_1}{t_N}$ with N being the number of involved computing entities and t_1 and t_N the time one computing entity and N computing entities taking to compute, respectively.

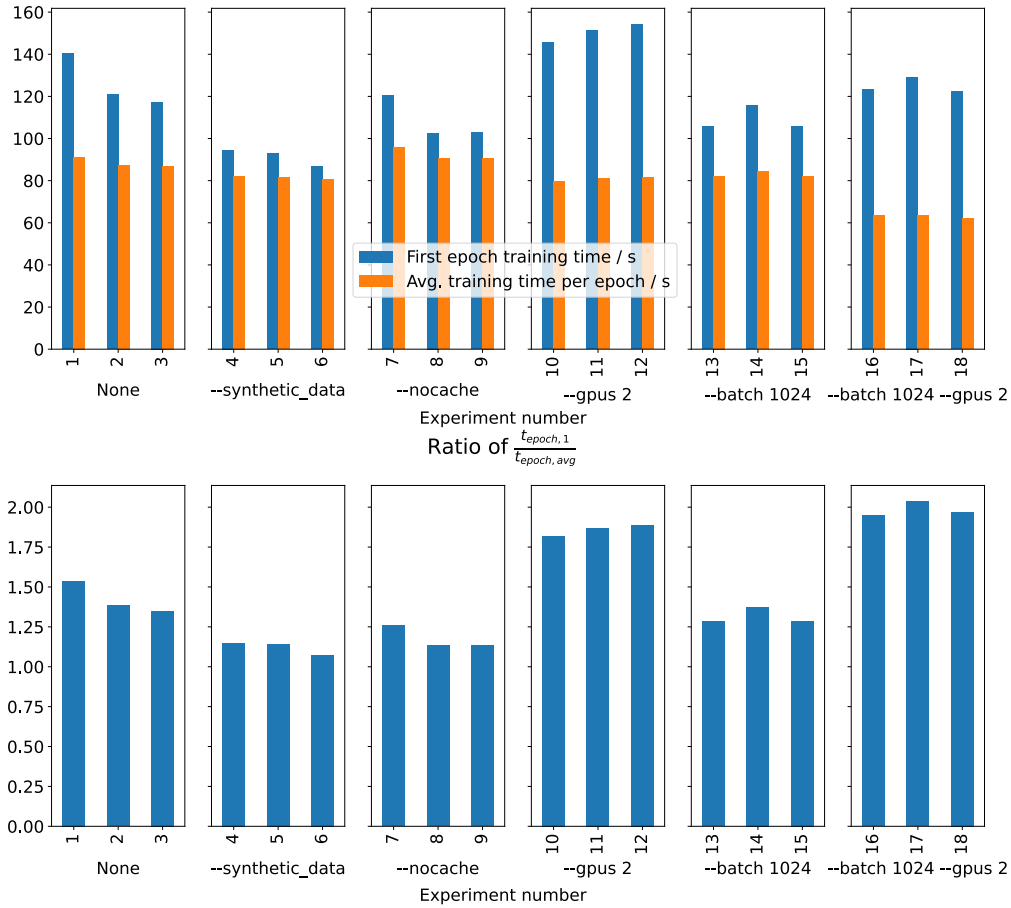
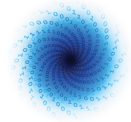


Figure 10: **AP3 JUWELS Booster** Epoch Time: Comparison of time for first epoch and average time for an epoch (top); ratio of both quantities (bottom)
ap3-jwb-epoch-time

time per epoch of 88.52 ± 2.36 s can be seen for the default case. In the various experiments, the speed-ups for average times are in line with the total training time. A ratio of 1.420 ± 0.001 of first vs. average epoch training time for the base case can be seen. Using 2 GPUs increases the ratio to 1.86 ± 0.03 – a larger overhead is expected for the initialization when using multiple devices.

When using synthetic data or not using cache, the ratio is reduced to 1.12 ± 0.04 and 1.18 ± 0.07 , respectively.

4.3.2.2 GPU power consumption

Total energy consumption and maximum GPU power draw were successfully measured for the application and are shown in Figure 11.

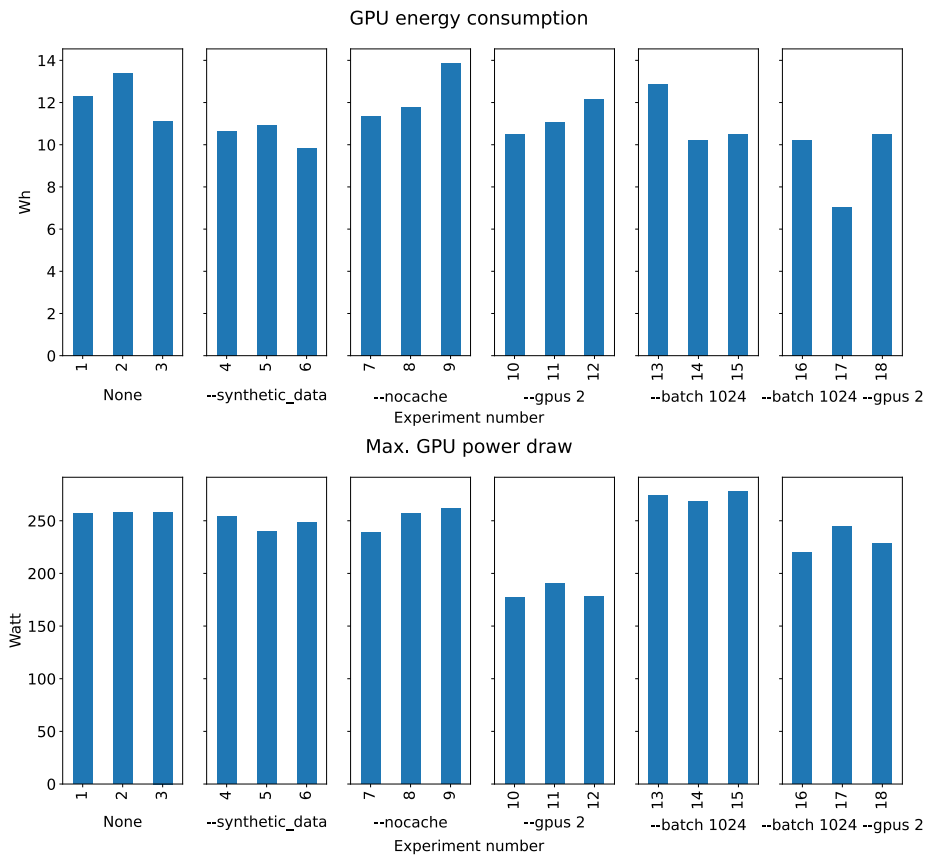
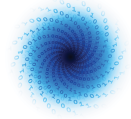
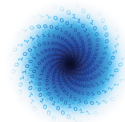


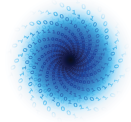
Figure 11: **AP3 JUWELS Booster** Energy: Total GPU energy consumption (top); peak GPU power draw (bottom) *ap3-jwb-energy*

The base configuration uses 12.32 ± 1.13 Wh to complete the benchmark. Using synthetic data reduces the energy consumption by 15 % to 10.47 ± 0.55 Wh. Foregoing the TensorFlow dataset cache results in a consumption of 12.32 ± 1.34 Wh within error margins of the base case. Using 2 GPUs reduces the energy consumption down per GPU by 9 % down to 11.25 ± 0.84 Wh. Roughly the same reduction can be observed when using a batch size of 1024, going down to 11.18 ± 1.47 Wh. Finally, when both using 2 GPUs and a larger batch size, the energy consumption is reduced by 25 % to 9.25 ± 1.92 Wh per GPU. More energy is consumed in total when using 2 GPUs.

The maximum GPU power draw averages at 257.79 ± 0.88 W for the base case. The power draw for both the synthetic data and no cache cases stays within the margin of error of the base case. When using 2 GPUs we see a reduction in max. power draw down to 182.0 ± 7.6 W, corroborating the previously observed suboptimal GPU utilization. When increasing the batch size, the energy consumption goes up



to 273.2 ± 4.8 W, which indicates more efficient use of the device. Finally, when both the batch size is increased to 1024 and 2 GPUs are used, the power draw is 230.9 ± 12.8 W per GPU.



4.3.3 JUWELS Cluster

4.3.3.1 Runtime

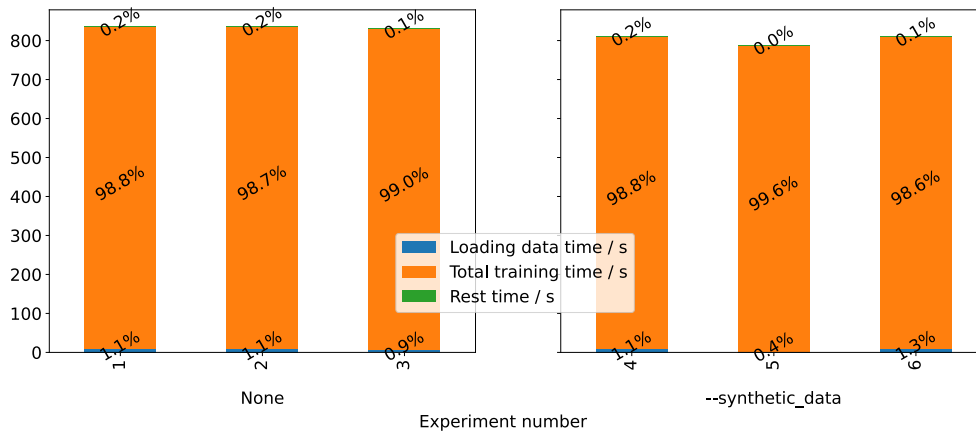


Figure 12: **AP3 JUWELS Cluster** Runtime: Runtime and relative share for multiple experiments *ap3-jwc-runtime-share*

On JUWELS Cluster, next to a default run, one additional configuration was evaluated; the configuration using synthetic data. The results for the runtime can be seen in Figure 12.

The average total training time for the default configuration is 824.63 ± 2.45 s, a factor of 1.86 slower than on JUWELS Booster. A slight reduction in training time of 3.7 % can be observed when using synthetic data, a smaller increase than on the Booster - this is expected, as the training takes much longer, therefore reducing the I/O load has a smaller effect.

The average training time per epoch increases compared to JUWELS Booster, as seen in Figure 13. This is in line with the increase in total training time. The ratio between the first and average epoch training time is 1.212 ± 0.004 for the base case and 1.08 ± 0.04 with synthetic data, an expected decrease from the ratio on the Booster.

4.3.3.2 GPU power consumption

On JUWELS Cluster the benchmark uses 52.54 ± 5.90 Wh of energy for the default configuration (see Figure 14). With synthetic data, the consumption is 51.76 ± 3.17 Wh which is within the margin of error of the default configuration. Around $4.26 \times$ more energy is consumed by the GPU to complete the benchmark on JUWELS Cluster compared to JUWELS Booster.

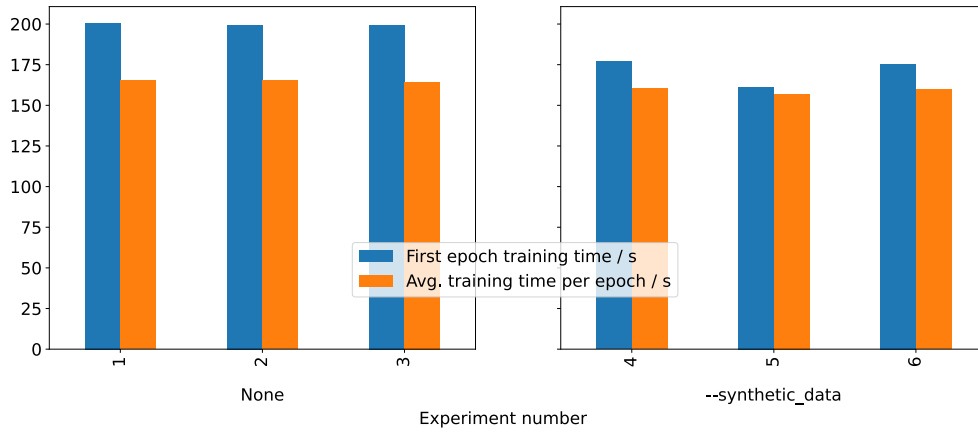
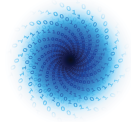


Figure 13: **AP3 JUWELS Cluster** Epoch Time: Comparison of time for first epoch and average time for an epoch for two configurations *ap3-jwc-epoch-time*

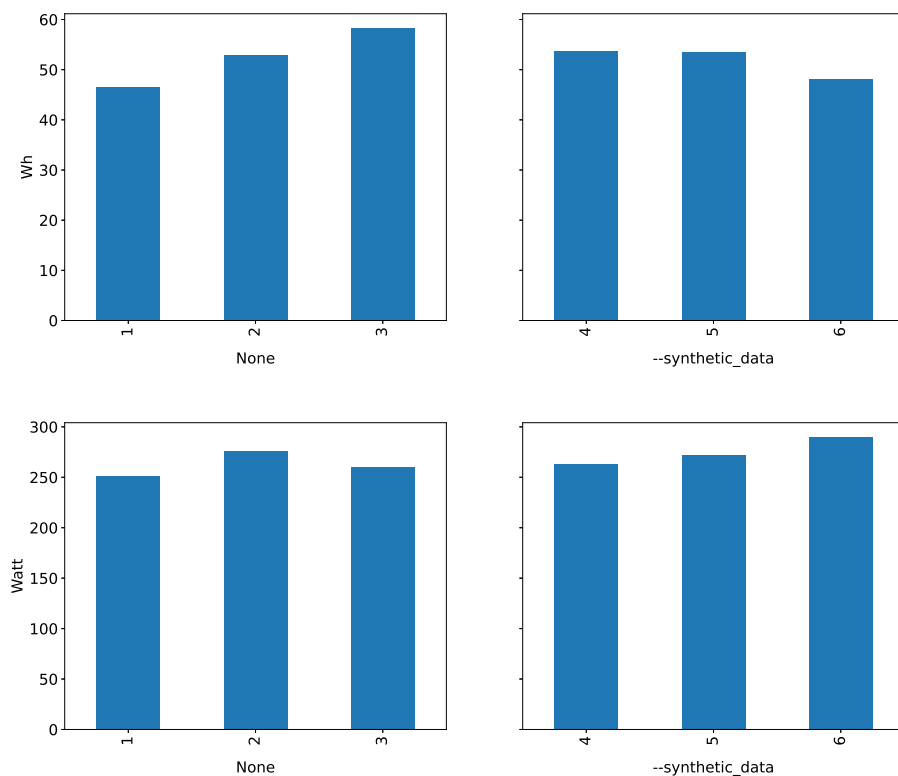
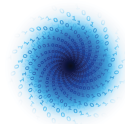
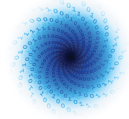


Figure 14: **AP3 JUWELS Cluster** Energy: Total GPU energy consumption (top); peak GPU power draw (bottom) *ap3-jwc-energy*

Similarly, the maximum GPU power draw varies only slightly between default and



synthetic data configurations, averaging 262.38 ± 13.01 W and 274.95 ± 13.62 W, respectively



4.3.4 E4 Intel+NVIDIA

4.3.4.1 Runtime

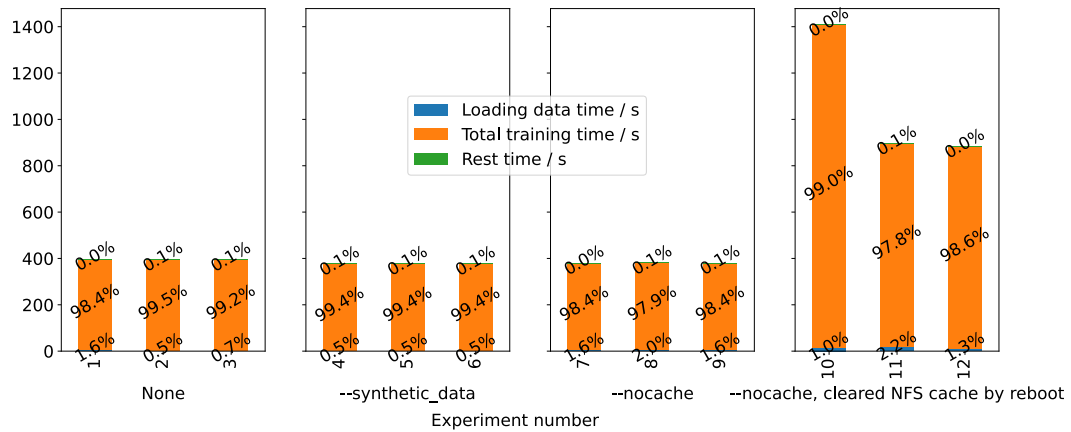


Figure 15: **AP3** **E4 Intel+NVIDIA** Runtime: Runtime and relative share for multiple experiments *ap3-e4i-runtime-share*

As showcased in Figure 15, better training performance than on JUWELS Booster can be seen on the E4 machine. For the default configuration, 391.54 ± 0.72 s are measured. This is similar to what was previously observed with AP1 in Figure 7 and Figure 8; in the case at hand, all experiments exhibiting the performance improvement.

One interesting feature can be seen when clearing the NFS cache by rebooting the node; in this case, the training time is dramatically increased. This is strong evidence for I/O operations being performed during training, i.e. data being streamed in while training occurs in parallel. For this run configuration, the slow-down is $2.23 \times$ to $3.56 \times$.

The average epoch training time (Figure 16) is in line with the total training time. A large increase in the ratio between the first and average epoch training time is also evident - for the default case we see a ratio of 1.212 ± 0.003 and when the NFS cache is cleared before running the benchmark the ratio is 2.97 ± 0.58 .

4.3.4.2 GPU power consumption

GPU power consumption on the E4 system is awaiting the development of the measurement tool.

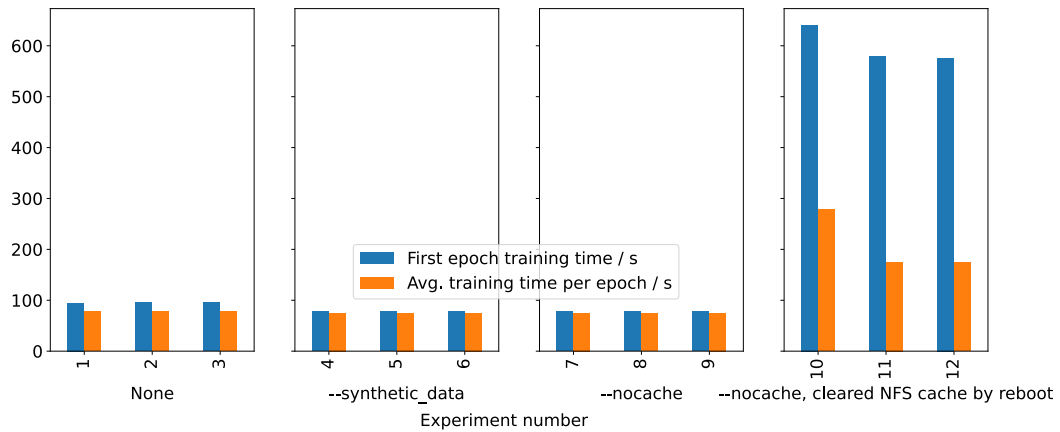
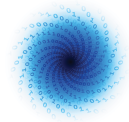


Figure 16: **AP3 E4 Intel+NVIDIA** Epoch Time: Comparison of time for first epoch and average time for an epoch for multiple configurations
ap3-e4i-epoch-time

4.3.5 JUWELS Booster Inference

4.3.5.1 Runtime

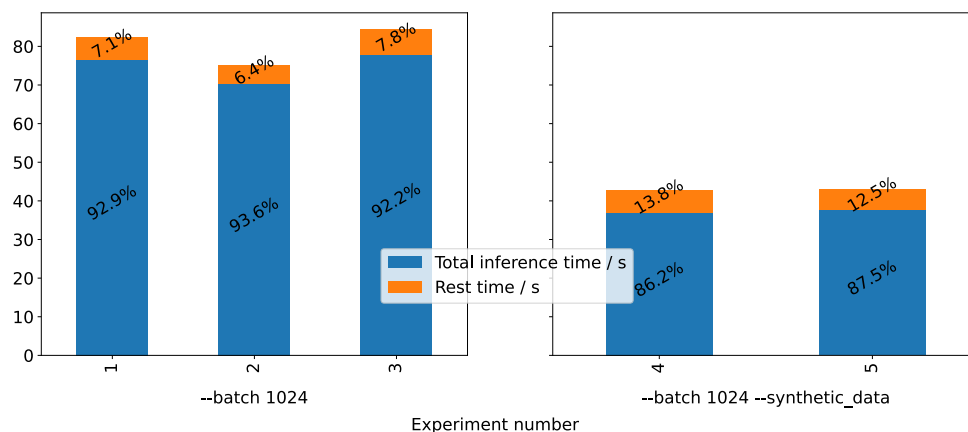
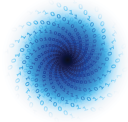


Figure 17: **AP3 JUWELS Booster** Inference Runtime: Runtime and relative share for multiple experiments and two configurations
ap3-jwb-inf-runtime-share

AP3 also measured time needed for inference on JUWELS Booster, as seen in Figure 17.

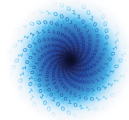
Inference runtime is significantly reduced when using synthetic data, a relative reduction between 46.3 % and 52.6 % can be seen. A portion of the time is spent on



overhead - between 6.4 % and 7.8 % of the total runtime for the default configuration and a higher 12.5 % to 13.8 % for synthetic data. This curious observation is worthwhile of investigation in the future.

4.3.5.2 GPU power consumption

The inference benchmark is too short for the GPU power consumption to be accurately captured by the job reports.



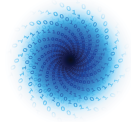
4.3.6 Results

The initial measurements for AP3 show that the majority of the total runtime on all systems is spent on actual training - between 98 % and 99 %. However, the application streams in data while training and there is strong evidence of the performance being I/O-bound.

Using multiple configurations leads to further insight into the performance behaviour of the application. In particular, configurations varying the I/O load, like using synthetic data, or changing the I/O performance of the underlying hardware, like clearing the NFS cache on the E4 machine, highlighted the implicit usage of I/O resources during training.

Training time can be observed to decrease by $\approx 46.4\%$ going from the less powerful V100 GPUs in JUWELS Cluster to the A100 GPUS in JUWELS Booster and E4s Intel+NVIDIA cluster. At the same time, the GPU energy consumption is $4.26 \times$ less when using JUWELS Booster. Similarly to AP1, an uplift of 10 % can be observed on the E4 machine compared to JUWELS Booster. For this application, however, the uplift is present with all experiments. One possible reason for this improvement can be data streaming and filesystem cache, calling for more in-depth investigation in the future.

The benchmark duration allows capture of GPU energy consumption via the JSC job reports. We observe a 4.26 better energy efficiency for JUWELS Booster compared to JUWELS Cluster.



4.4 AP 4

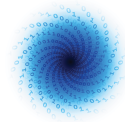
4.4.1 Notes

Memory training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
1080 GB	12 GB	945	[42, 352, 704]	2

Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
3.8 M	0	MSE+SSIM	None

Data formats	Frameworks (to be) used
GRIB	TensorFlow 1.14, PyTorch 1.8(transitioning)

The underlying data of illustrations in this section can be found in appendix 6.3.



4.4.2 JUWELS Booster

4.4.2.1 Runtime

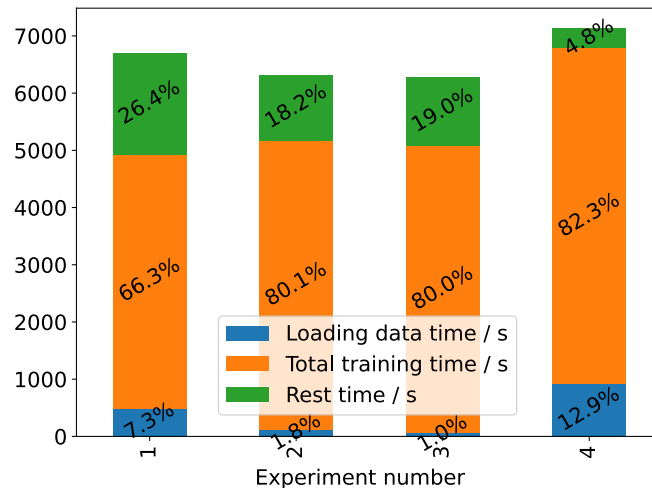


Figure 18: AP4 JUWELS Booster Runtime: Runtime and relative share for multiple experiments *ap4-jwb-runtime-share*

Some variation between different benchmark runs can be observed in Figure 18 with regard to the distribution of time spent between loading data, training time, and rest time, as well as the absolute time spent on training. Loading data time varies between 1.0 % and 12.9 %, training time between 66.3 % and 82.3 %, and unaccounted time between 4.8 % and 26.4 %. The large proportion of rest time means that other areas of the application code apart from loading data and training are worth investigating in a more in-depth performance analysis. In addition, the variation of absolute runtime as well as runtime distribution should be further analysed.

The time to train the first epoch and the average of all epochs is shown in Figure 19 for different repeating experiments. The middle two experiments have a ratio of first vs. average epoch training time of roughly 1, meaning that no significant portion of time is spent on overhead or initialization during the first epoch. This is not the case for experiments number 1 and 4. Due to the larger runtime of the benchmark, it is possible that experiments 2 and 3 profit from some form of caching, while the first and the last don't. However, we also see that during the first run, the average training time per epoch is significantly smaller than during the other runs, while the last experiment takes considerably longer. In subsequent analyses,

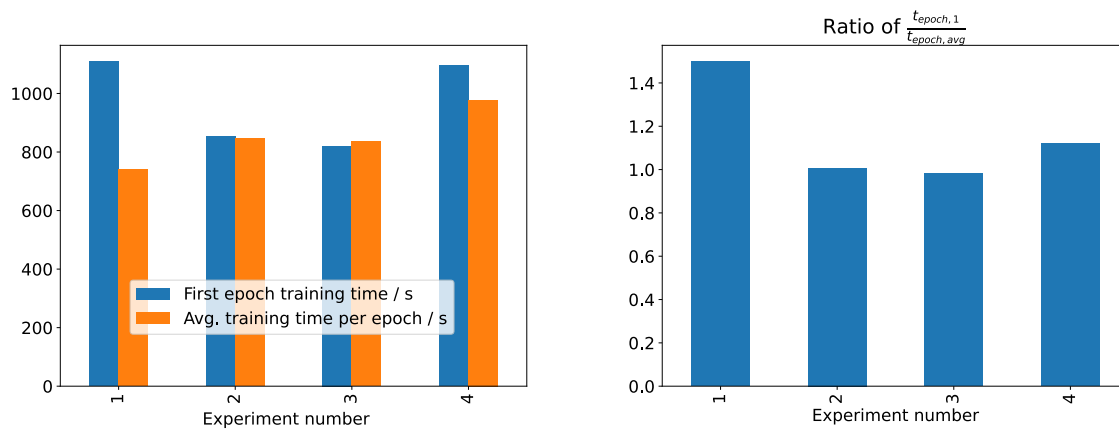
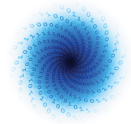
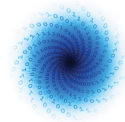


Figure 19: **AP4 JUWELS Booster** Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right)
ap4-jwb-epoch-time

attention must be paid to potential underlying, systematic or statistical hardware effects.



4.4.2.2 GPU power consumption

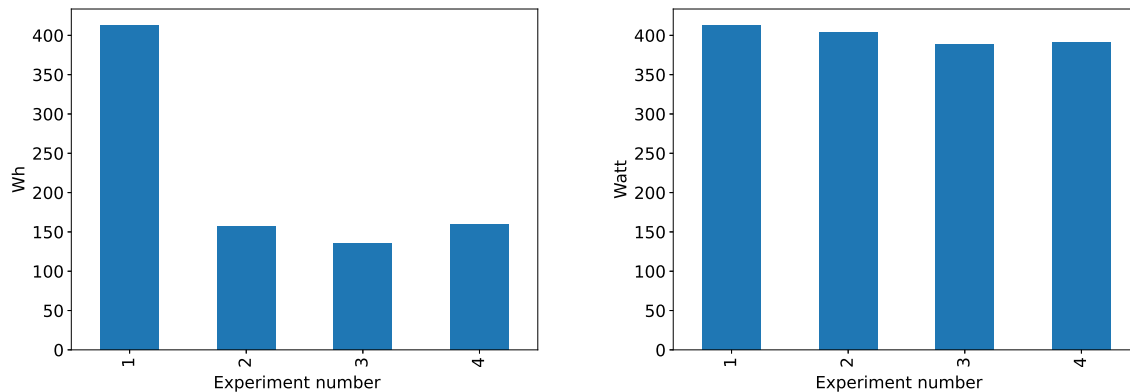
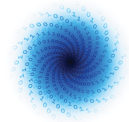


Figure 20: AP4 JUWELS Booster Energy: Total GPU energy consumption (left); peak GPU power draw (right) *ap4-jwb-energy*

All experiment runs seem to be causing a similar max. GPU power draw of on average 399.0 ± 10.2 W, as shown in the right plot of Figure 20. However, the left plot of Figure 20 shows that the very first run uses much more energy - 412.9 Wh, while the subsequent three runs use 2.74 times less, averaging 150.8 ± 13.2 W. Further investigation into why the first run uses much more energy is required.

4.4.3 JUWELS Cluster

No benchmarks were performed on JUWELS Cluster for AP4



4.4.4 E4 Intel+NVIDIA

On the E4 machine, the benchmark ran with a batch size of 1 – compared to a batch size of 2 on JUWELS.

4.4.4.1 Runtime

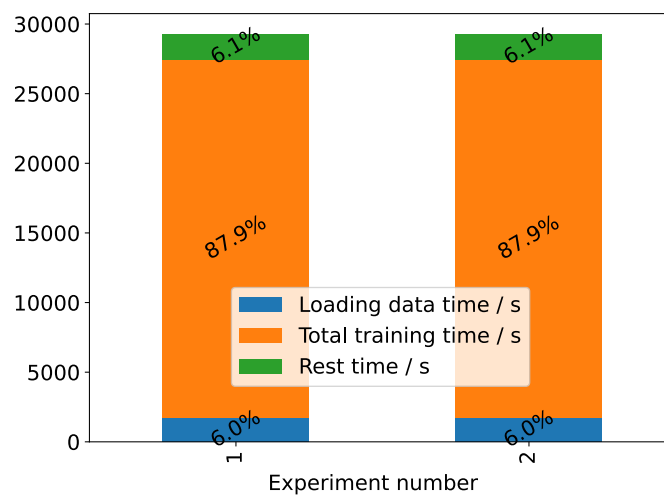


Figure 21: AP4 E4 Intel+NVIDIA Runtime: Runtime and relative share for multiple experiments *ap4-e4i-runtime-share*

Between the two runs on the E4 system, the time distribution seems consistent (Figure 21), training taking $25\,729 \pm 12$ s on average. Due to different parameters, a direct comparison to JUWELS Booster is not possible in this case.

Looking at Figure 22, no significant difference between the first and the average epoch training time can be observed.

4.4.4.2 GPU power consumption

GPU Power consumption on the E4 system is awaiting the development of the measurement tool.

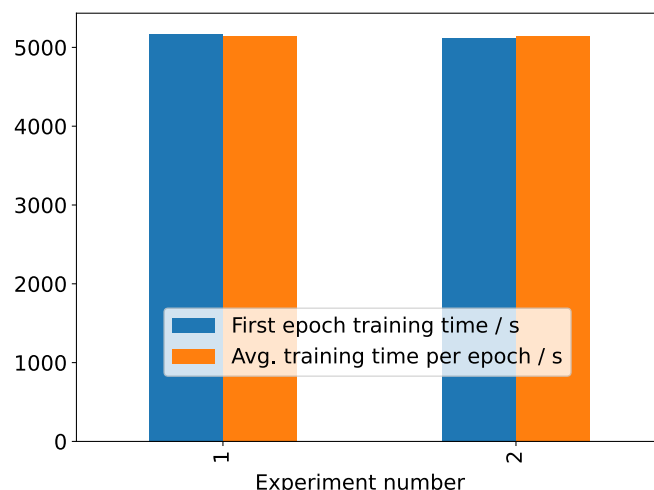
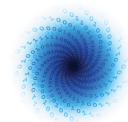
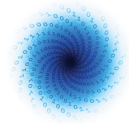


Figure 22: AP4 E4 Intel+NVIDIA Epoch Time: Comparison of time for first epoch and average time for an epoch *ap4-e4i-epoch-time*

4.4.5 Results

The AP4 benchmarks runs the longest, taking 6275 s to 7128 s on JUWELS Booster and 29267 s to 29284 s on the E4 machine with a different configuration. We observe strong variation of training, loading, and rest time between the 4 experiments on the JUWELS Booster with regard to both the time distribution and absolute numbers. A similar variation can be seen in the energy consumption, where the first benchmark run is significantly higher on JUWELS Booster.

The runtime and energy consumption results warrant further benchmarking to gather more statistics and a more in-depth performance analysis, including an investigation of code and hardware to identify where the strong variation in metrics is coming from.



4.5 AP 5

4.5.1 Notes

Data formats	Frameworks (to be) used
NetCDF	Tensorflow v2.3.1 or v2.5 with Keras API

The raw data associated to graphs of this section is listed in appendix 6.4.

4.5.1.1 Small Dataset

Memory training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
274.51 MB	34.88 MB	732	[96, 128,3]	32

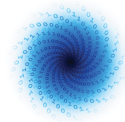
Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
3 525 650	3360	MAE of normalized temperature	None

The initial benchmarks performed on JUWELS Booster used a very small dataset. This benchmark was found out to be inadequate to capture performance metrics correctly and a larger dataset was produced.

4.5.1.2 Large Dataset

Memory training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
7.506 GB	0.954 GB	20 496	[96, 128,3]	32

The large dataset was used subsequently for JUWELS Booster experiments and benchmarking on all other systems.



Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
3 525 650	3360	MAE of normalized temperature	None

4.5.2 JUWELS Booster (Small Dataset)

4.5.2.1 Runtime

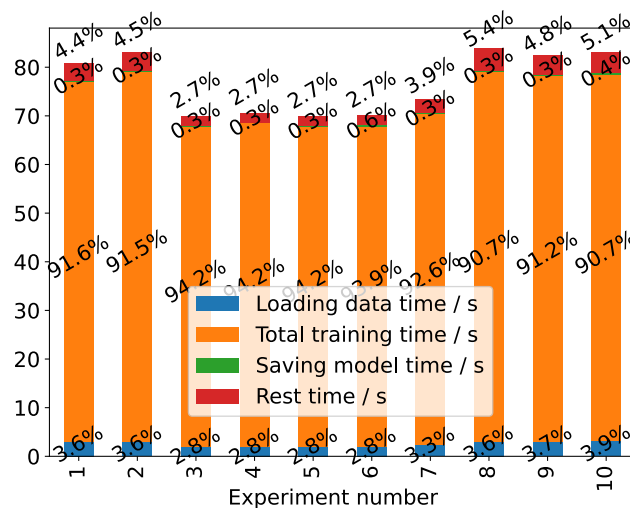


Figure 23: **AP5 JUWELS Booster** Runtime (Small Dataset): Runtime and relative share for multiple experiments *ap5-jwb-runtime-share-small*

The total training time for the small dataset is on average 70.8 ± 4.8 s, however a difference is evident between experiments 3-7, which take significantly less time with 66.40 ± 0.94 s total training time on average, and the rest of the experiments, averaging 75.30 ± 0.86 s. In all experiments, the dominant part of the runtime is spent on training, 90.7 % to 94.2 % while the time spent loading data constitutes 2.8 % to 3.9 %. AP5 also recorded the time spent saving the model, which is the smallest portion of the runtime with 0.3 % to 0.6 %. Finally, the unaccounted time is 2.7 % to 5.4 % of the total.

The average training time per epoch is very short with 1.01 ± 0.07 s – see Figure 24. A high ratio between first and average epoch training time shows that the runtime is dominated by overhead.

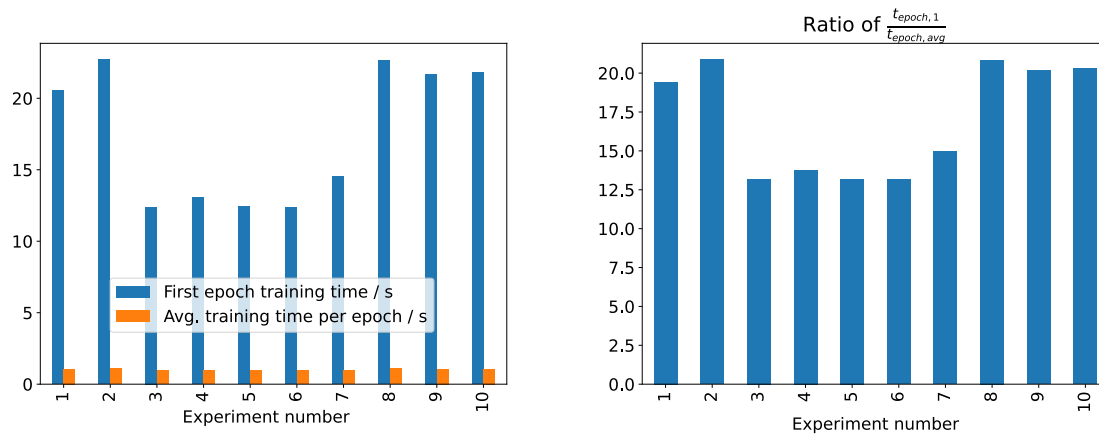
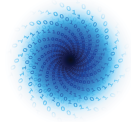


Figure 24: **AP5 JUWELS Booster** Epoch Time (Small Dataset): Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right) *ap5-jwb-epoch-time-small*

4.5.2.2 GPU power consumption

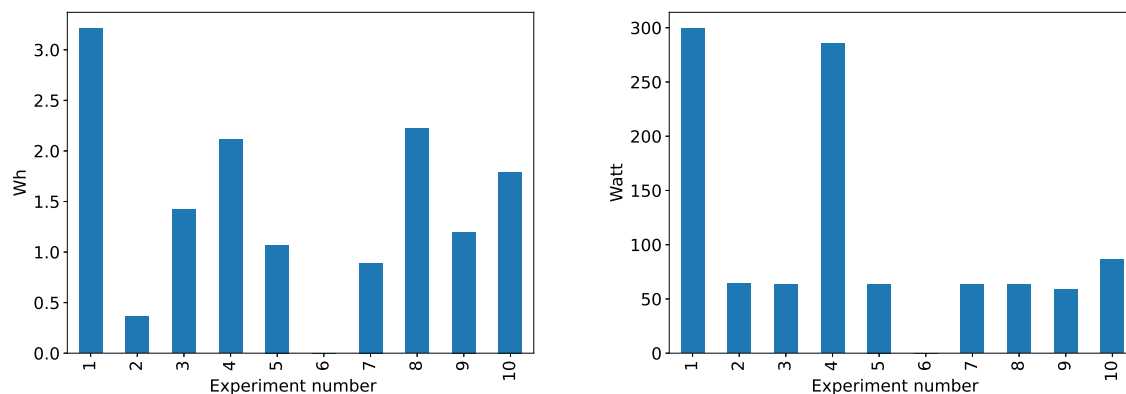
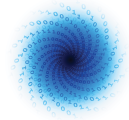


Figure 25: **AP5 JUWELS Booster** Energy (Small Dataset): Total GPU energy consumption (left); peak GPU power draw (right) *ap5-jwb-energy-small*

The data in Figure 25 shows that the benchmark with the small dataset fails to capture the GPU power consumption and power draw correctly.



4.5.3 JUWELS Booster (Large Dataset)

4.5.3.1 Runtime

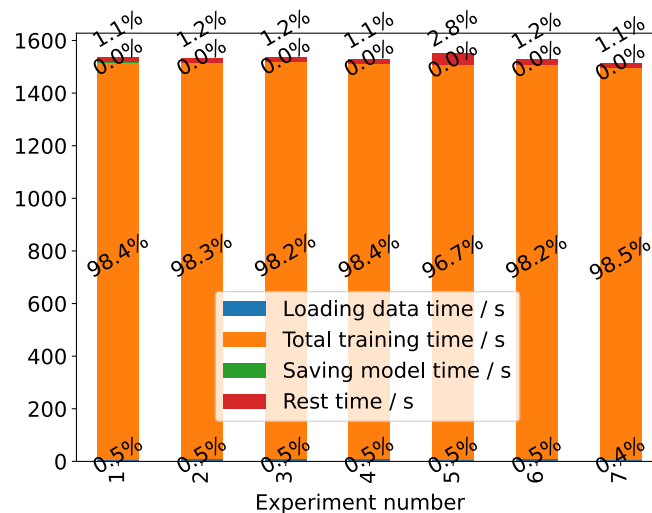


Figure 26: **AP5 JUWELS Booster** Runtime (Large Dataset): Runtime and relative share for multiple experiments *ap5-jwb-runtime-share*

With the larger dataset, we see the proportion of training time go up to 98 %, taking 1504 ± 7 s on average (Figure 26). 0.5 % is spent on loading data. Saving the model takes an insignificant amount of time and on average 1.2 % of the total runtime is unaccounted for, with one slight outlier of 2.8 %. With the larger dataset, the consistency of the result is much higher; furthermore, the training time is now even more dominant while the I/O overhead and unaccounted runtime shrink.

The average epoch training time is 21.5 ± 0.1 s, the ratio between the first and average epoch being 1.81 ± 0.14 . The lower ratio is an indicator of the initialization/overhead time not dominating the benchmark. Both observations can be seen in Figure 27.

4.5.3.2 GPU power consumption

Maximum GPU power draw and GPU energy consumption were captured properly for the larger dataset (see Figure 28, with the maximum GPU power draw averaging 322.1 ± 6.4 W and GPU energy consumption 49.0 ± 2.1 W h. Scaling the benchmark up resulted in accurate data, which can be used as a point of reference for other systems.

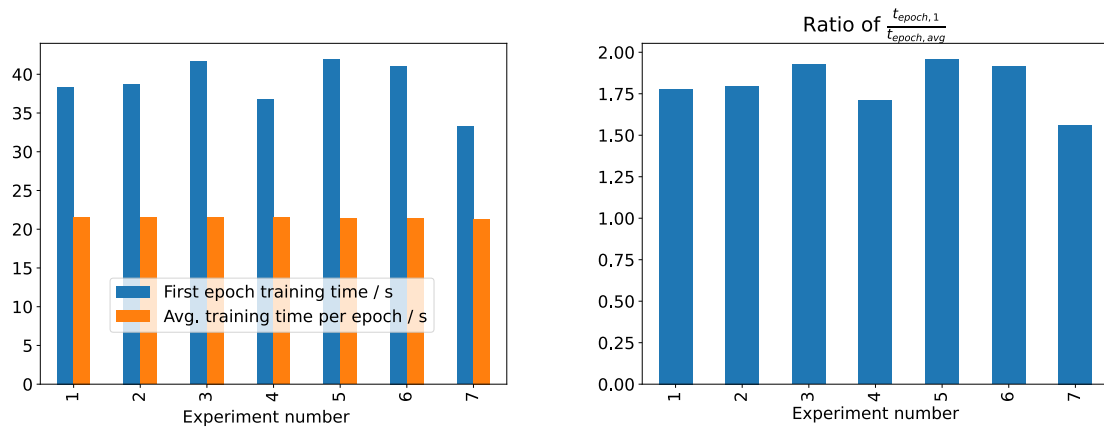
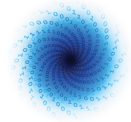


Figure 27: **AP5 JUWELS Booster** Epoch Time (Large Dataset): Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right) *ap5-jwb-epoch-time*

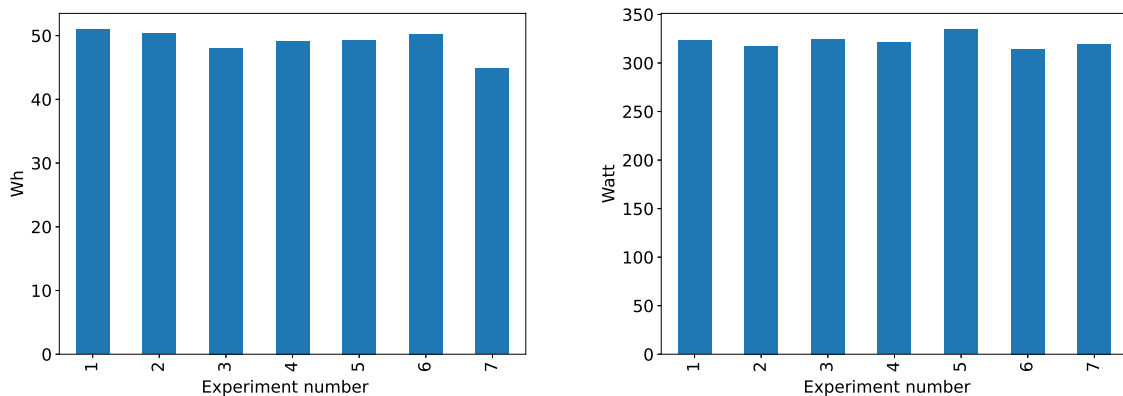


Figure 28: **AP5 JUWELS Booster** Energy (Large Dataset): Total GPU energy consumption (left); peak GPU power draw (right) *ap5-jwb-energy*

4.5.4 JUWELS Cluster

4.5.4.1 Runtime

As seen in Figure 29, the average total training time on JUWELS Cluster is 2784.9 ± 6.8 s, $1.85 \times$ slower than on JUWELS Booster. With 98.4 %, training takes the majority of the runtime.

On JUWELS Cluster, the average epoch training time is 39.8 ± 0.1 s, the ratio between the first and average epoch being 1.45 ± 0.14 (Figure 30). The lower value of the ratio compared to JUWELS Booster is expected, as the longer training time reduces the overhead/initialization proportion.

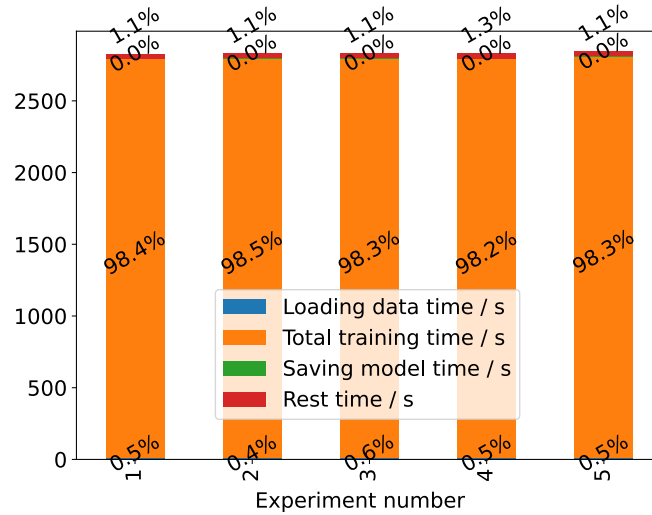
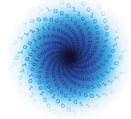


Figure 29: **AP5 JUWELS Cluster Runtime: Runtime and relative share for multiple experiments**
ap5-jwc-runtime-share

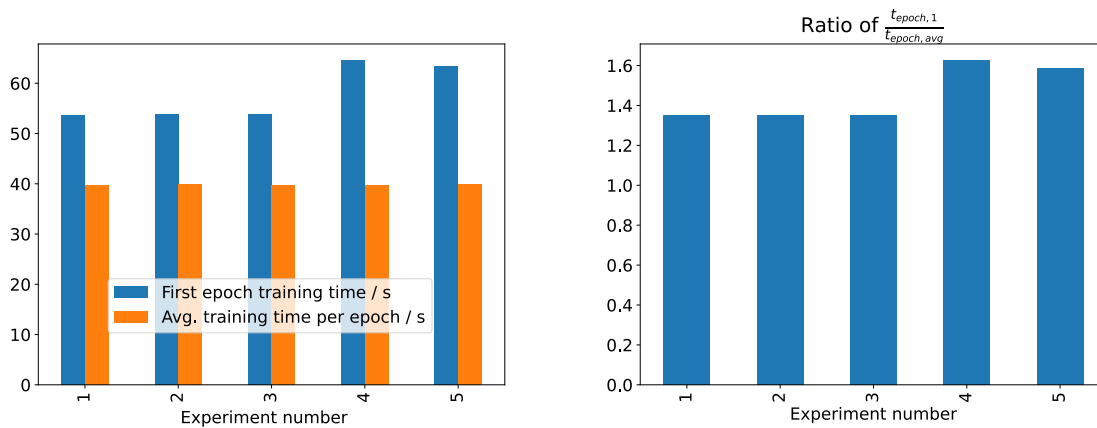


Figure 30: **AP5 JUWELS Cluster Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right)**
ap5-jwc-epoch-time

4.5.4.2 GPU power consumption

No issues capturing GPU power metrics were encountered, as seen in Figure 31. The average max. GPU power draw was measured at 295.0 ± 7.6 W and GPU energy consumption at 205.2 ± 6.2 Wh, $4.19 \times$ more than on JUWELS Booster. It is evident that the newer A100 GPUs in the Booster not only outperform the older V100, but also consume a much smaller amount of energy.

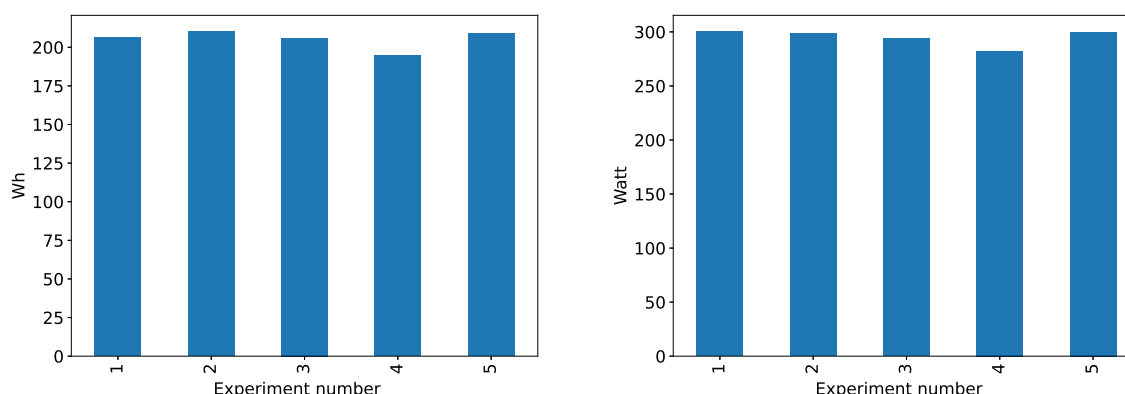
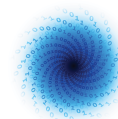


Figure 31: **AP5 JUWELS Cluster** Energy: Total GPU energy consumption (left); peak GPU power draw (right) *ap5-jwc-energy*

4.5.5 E4 Intel+NVIDIA

For the benchmarks on the E4 system, some additional contextual information is available for the experiments. Experiments 1 and 4 were the first experiments on their respective days and experiment 6 was performed with a 3 h delay from the preceding runs. The other experiments were always subsequent runs to previous invocations and could profit from the NFS data cache. Experiments 1, 4 and 6 are in the following section referred to as *non-cached* and 2, 3, 5, 7 and 8 as *cached*.

4.5.5.1 Runtime

Non-cached experiments spent 14.5 times longer on loading data – see Figure 32. The non-cached loading time is on average 82.4 ± 0.3 s, while the cached loading data time is 5.7 ± 0.1 s. There is also a slight change in the total training time, the non-cached experiments taking 1579.6 ± 16.0 s on average and the cached runs taking 1549.8 ± 9.8 s. Compared to JUWELS Booster there is a 5 % difference in the training time favouring JUWELS Booster.

An average epoch training time of 22.6 ± 0.2 s can be observed for the non-cached experiments, and a very slightly shorter time of 22.1 ± 0.1 s for the cached experiments (Figure 33). The ratio of first vs. average epoch training time for the non-cached experiments is 1.8 ± 0.1 and for the cached experiments 1.281 ± 0.002 s. The higher ratio for non-cached experiments is in line with the expectation of higher overhead and initialization time.

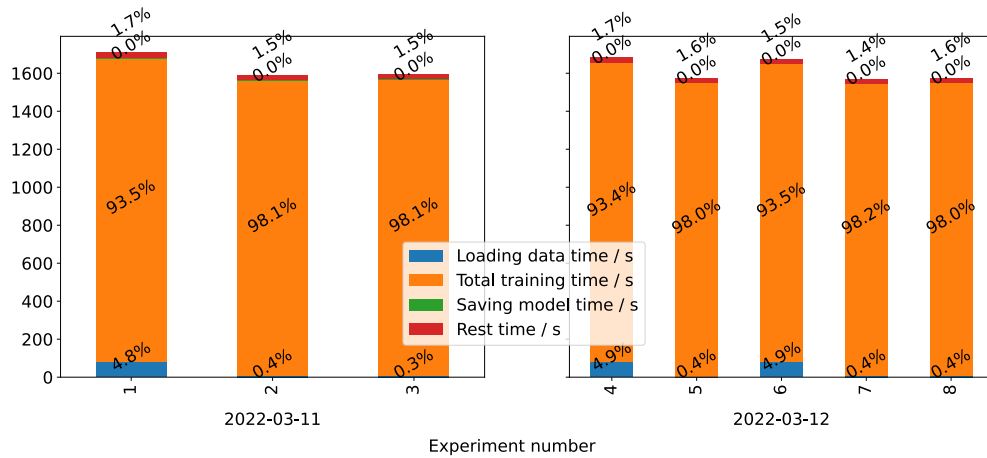
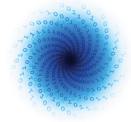


Figure 32: **AP5 E4 Intel+NVIDIA** Runtime: Runtime and relative share for multiple experiments *ap5-e4i-runtime-share*

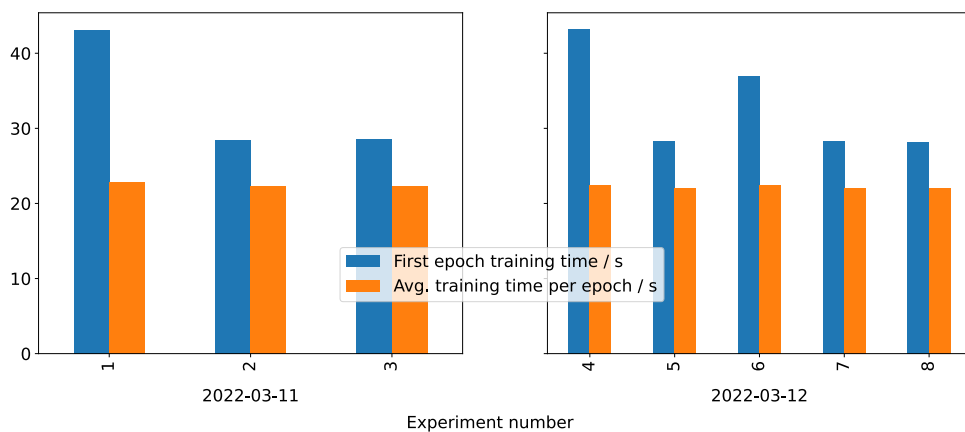
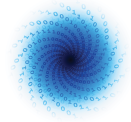


Figure 33: **AP5 E4 Intel+NVIDIA** Epoch Time: Comparison of time for first epoch and average time for an epoch (left); ratio of both quantities (right) *ap5-e4i-epoch-time*

4.5.5.2 GPU power consumption

GPU Power consumption on the E4 system is awaiting the development of the measurement tool.



4.5.6 JUWELS Booster Inference

4.5.6.1 Runtime

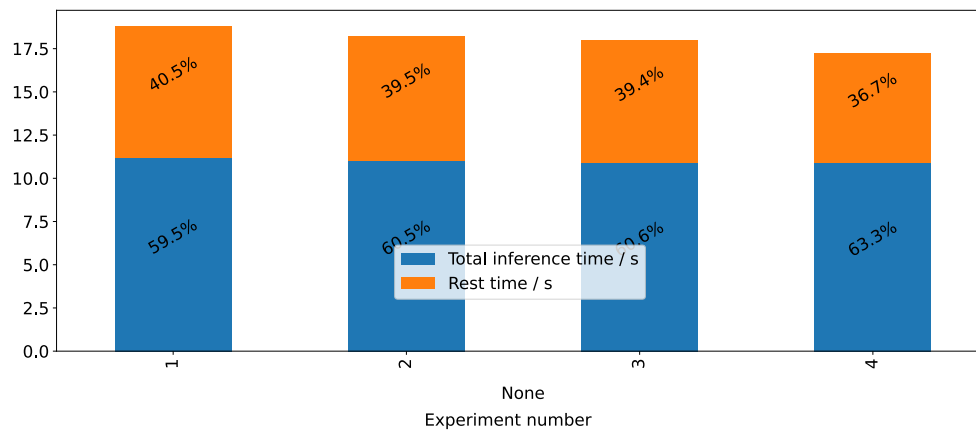
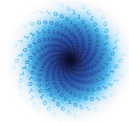


Figure 34: AP5 JUWELS Booster Inference Runtime: Runtime and relative share for multiple experiments *ap5-jwb-inf-runtime-share*

Inference takes 11.0 ± 0.1 s on average, see Figure 34. A significant portion of the total runtime, however, is spent on overhead, accounting for 36.7 % to 40.5 % of the runtime.

4.5.6.2 GPU power consumption

GPU power metrics could not be captured for the inference benchmark as the benchmark was too short.



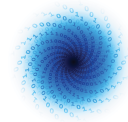
4.5.7 Results

AP5 has successfully scaled up their benchmark in time for the deliverable by using a larger dataset. This shows how a larger runtime improves the accuracy of the benchmark.

With over 98 % of the total runtime is spent on training, the application is mostly compute-bound, not heavily dependent on I/O. Experiments on E4 – where it was possible to distinguish between runs that used data with and without cache – confirm this, with the I/O portion remaining below 5 % in the worst-case scenario.

A speed-up factor of $1.85 \times$ and a $4.19 \times$ lower energy consumption was observed between the V100 GPUs of JUWELS Cluster and the A100s in JUWELS Booster.

AP5 has also benchmarked inference – however the inference runtime is only on the order of 10 seconds and, due to the short benchmark time, a 40 % unaccounted runtime/overhead can be observed, also GPU power metrics could not be captured for inference.



4.6 AP 6

4.6.1 Notes

Memory training dataset	Memory validation dataset	Training samples	Input shape sample	batch size
1.13 GB	-	-	(1461, 351, 551)	-

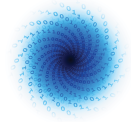
Trainable parameters	Non-trainable parameters	Loss function	Experimental notes
-	-	-	None

Data formats	Frameworks (to be) used
NetCDF, Grib,csv	TensorFlow 2 or PyTorch

Currently, A6 does not implement neural networks (NNs), but uses rather conservative ML solutions (principal component analysis and clustering). Metrics aiming to investigate the performance of NNs are irrelevant for A6 at the moment. Furthermore, A6 has not yet implemented algorithms to use GPUs for acceleration, it is planned for the future. Hence, GPU-related metrics can not be given for the application.

As an alternative, a simple benchmark capturing the total runtime and data loading time was performed.

Data for the benchmarks which are shown here can be found in appendix 6.5.



4.6.2 JUWELS Booster

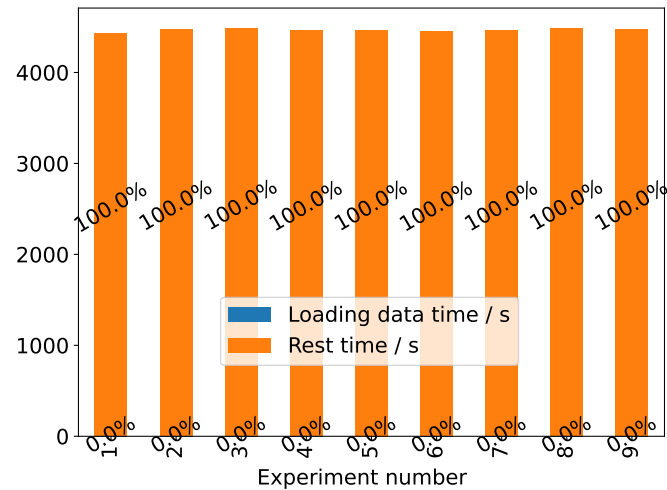


Figure 35: AP6 JUWELS Booster Runtime: Runtime and relative share for multiple experiments *ap5-jwb-runtime-share*

4.6.3 JUWELS Cluster

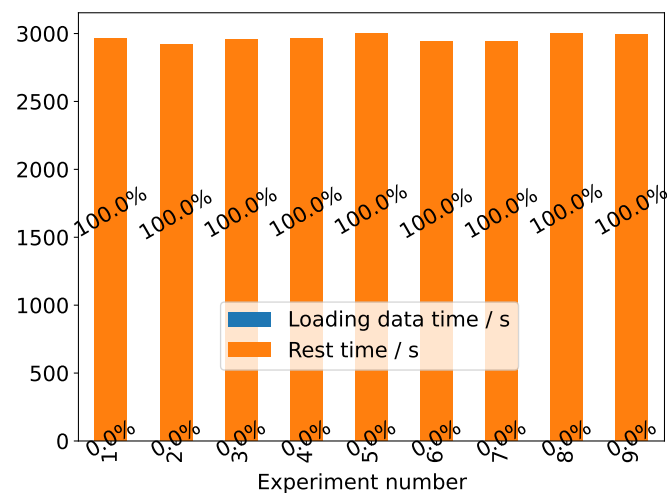
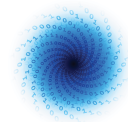


Figure 36: AP6 JUWELS Cluster Runtime: Runtime and relative share for multiple experiments *ap5-jwc-runtime-share*



4.6.4 E4 Machines

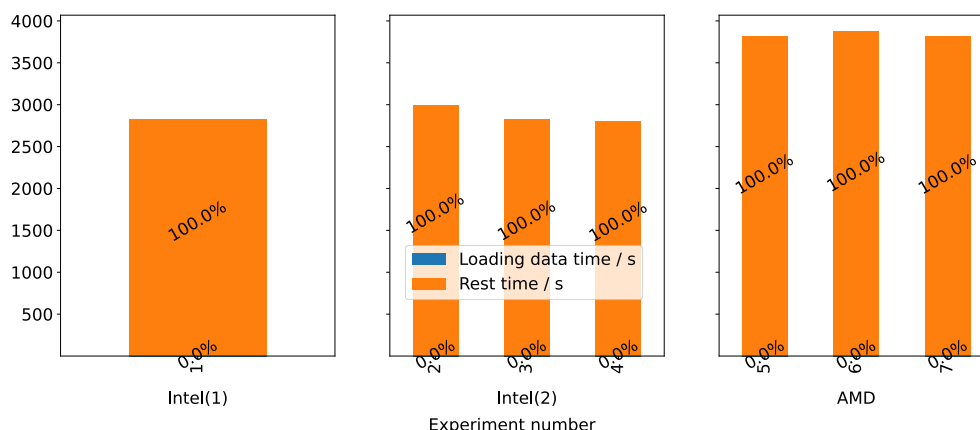


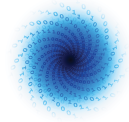
Figure 37: AP6 E4 Runtimes: Runtime and relative share for multiple experiments on different systems hosted at E4 *ap5-e4-runtime-share*

4.6.5 Results

For AP6 the runtime was measured on various systems, on JUWELS Booster (Figure 35), on JUWELS Cluster (Figure 36), and on different systems hosted at E4 (Figure 37). The total runtime ranges from 2822.3 s on one of E4s Intel-based machines to 4464.1 ± 16.3 s on JUWELS Booster. The difference likely comes from different microarchitectures and available threads.

As the application has not gone through optimizations and does currently not utilize GPUs, the measurement largely serves as a baseline point of reference for future benchmarking.

Of note is also the negligible time spent on loading the data – the total runtime is determined by the processing time.



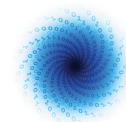
5 Conclusion

Initial benchmarking of the MAELSTROM applications provides a baseline performance and an initial insight into what the first steps in terms of optimization should be.

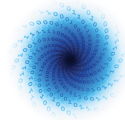
As expected, we see a training performance uplift when going from the NVIDIA V100 to the A100 GPU, resulting in a speed-up of around $2 \times$. This goes together with a much lower GPU energy consumption, the A100 GPUs requiring up to $4.25 \times$ less power than the older V100s.

We see evidence of I/O playing a significant role – both with explicitly measured loading time taking up the majority of the runtime in AP1, as well as streaming I/O being utilized and making use of filesystem caching in AP3.

No application makes extensive use of multiple GPUs yet, which is going to change in the future. It is important to not only scale application out to multiple GPUs, but also improve upon the points of investigation and improvement outlined in this document.



6 Appendix



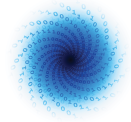
6.1 AP 1

Experiment number	0	1	2	3	4	5	6	7	8	9	AVG
Job ID	5028634	5028635	5028636	5028668	5028669	5028670	5028671	5028787	5028789	5028790	
#Nodes	1	1	1	1	1	1	1	1	1	1	
#GPUs	1	1	1	1	1	1	1	1	1	1	
#MPI tasks	1	1	1	1	1	1	1	1	1	1	
#CPUs	1	1	1	1	1	1	1	1	1	1	
Loading data time[s]	218.948	220.510	221.347	218.577	217.182	216.846	216.710	218.926	218.543	217.303	218.489
Total runtime[s]	360.000	361.000	361.000	359.000	357.000	357.000	358.000	360.000	359.000	358.000	359.000
Total training time[s]	134.558	133.922	133.822	133.901	134.154	134.013	134.418	134.027	133.881	134.014	134.071
Avg. training time per epoch[s]	26.912	26.784	26.764	26.780	26.831	26.803	26.884	26.805	26.776	26.803	26.814
First epoch training time[s]	47.065	45.906	45.126	42.961	43.617	43.134	43.621	43.740	43.367	43.694	44.223
Min. training time per epoch[s]	20.436	20.460	20.436	20.455	20.442	20.460	20.446	20.442	20.457	20.465	20.450
Max. training time per epoch[s]	47.065	45.906	45.126	42.961	43.617	43.134	43.621	43.740	43.367	43.694	44.223
Avg. training time per iteration[s]	3.36	3.35	3.35	3.35	3.35	3.35	3.36	3.35	3.35	3.35	3.35
Final training loss	5.46	4.85	5.39	5.06	5.56	4.22	4.90	4.52	5.01	4.97	5.00
Final validation loss	4.32	3.75	4.64	3.19	4.27	3.01	3.99	2.83	4.46	3.26	3.77
Saving model time	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	-
Node ID	jwb0667	jwb0039	jwb0040	jwb0040	jwb0667	jwb0856	jwb0864	jwb0101	jwb0040	jwb0245	
Max. GPU power	140.90 W	82.7 W	78.7 W	61.0 W	77.5 W	65.2 W	75.2 W	64.1 W	61.6 W	72.6 W	77.95 W
GPU energy consumption	6.4 Wh	6.1 Wh	6.8 Wh	5.6 Wh	5.8 Wh	6.0 Wh	5.8 Wh	5.7 Wh	5.6 Wh	6.0 Wh	5.98 Wh!

Table 1: AP1 JUWELS Booster training benchmark

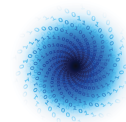
Experiment number	1	2	3
Job ID	5177051	5177052	5177053
#Nodes	1	1	1
#GPUs	1	1	1
#MPI tasks	1	1	1
#CPUs	1	1	1
Loading data time	411.744	411.034	411.370
Total runtime	684.000	681.000	678.000
Total training time	261.237	261.966	258.259
Avg. training time per epoch	52.247	52.393	51.652
First epoch training time	79.104	72.368	66.603
Min. training time per epoch	40.866	40.848	40.869
Max. training time per epoch	79.104	72.368	68.731
Avg. training time per iteration	6.53E+00	6.55E+00	6.46E+00
Final training loss	4.94E+00	5.65E+00	4.90E+00
Final validation loss	3.05E+00	4.22E+00	3.98E+00
Saving model time	Not calculated	Not calculated	Not calculated
Node ID (job report)	jwc09n054	jwc09n084	jwc09n054
Max. GPU power (job report)	70.57 W	72.03 W	71.55 W
GPU energy consumption (job report)	9.7 Wh	9.9 Wh	10.0 Wh

Table 2: AP1 JUWELS Cluster training benchmark



Experiment number	1	2	3	4	5
Job ID	309	312	313	314	359
#Nodes	1	1	1	1	1
#GPUs	1	1	1	1	1
#MPI tasks	1	1	1	1	1
#CPUs	1	1	1	1	1
Loading data time	315.067	316.742	316.430	316.554	387.847
Total runtime	457.000	442.000	441.000	442.000	531.000
Total training time	135.341	122.566	121.977	122.373	136.571
Avg. training time per epoch	27.068	24.513	24.395	24.474	27.314
First epoch training time	59.230	46.459	45.997	46.384	60.355
Min. training time per epoch	18.977	18.999	18.897	18.887	18.984
Max. training time per epoch	59.230	46.460	45.997	46.384	60.355
Avg. training time per iteration	3.38E+00	3.06E+00	3.05E+00	3.06E+00	3.41E+00
Final training loss	4.83E+00	4.93E+00	4.85E+00	4.89E+00	4.57E+00
Final validation loss	3.50E+00	3.75E+00	4.85E+00	3.30E+00	3.51E+00
Saving model time	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated
Node ID	icnode02	icnode02	icnode02	icnode02	icnode02
Max. GPU power	Not monitored	Not monitored	Not monitored	Not monitored	232.39W
GPU energy consumption	Not monitored	Not monitored	Not monitored	Not monitored	6 Wh

Table 3: **AP1** **E4 Intel+NVIDIA** training benchmark



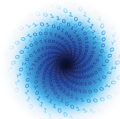
6.2 AP 3

Number	1	2	3	4	5
Experiment flag	--synthetic_data	--nocache	--gpus 2	--batch 1024	--batch 1024 --gpus 2

Table 4: AP3 Experiment flag reference

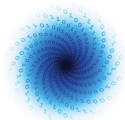
Experiment number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Job ID	4974643	4973509	5088401	4974675	5095095	5095622	4974703	5096117	5096472	4999533	5098020	5102945	4974716	4973493	5103162	4974749	4999514	5103365
#Nodes	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	2	2	2	2	1	1	1	2	2	2
#MPI tasks	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
#CPUs	8	8	8	8	8	8	8	8	8	16	16	16	8	8	8	16	16	16
Loading data time	12.198	6.020	5.491	5.440	8.173	3.501	8.227	6.785	6.279	6.067	7.604	6.141	4.222	7.214	4.718	4.179	8.468	5.549
Total runtime	468.884	443.668	440.730	417.052	415.771	407.598	487.373	460.919	460.780	406.168	412.896	415.473	415.985	428.657	416.113	321.380	326.565	316.294
Total training time	456.130	437.256	434.474	411.252	407.092	403.587	478.826	453.104	453.403	399.570	404.475	408.116	411.449	421.057	410.989	316.660	317.531	310.099
Avg. training time per epoch	1.142	1.094	1.087	1.029	81.418	80.717	1.198	90.621	90.681	1.000	80.895	81.623	1.030	1.054	82.198	0.792	0.795	62.020
First epoch training time	140.241	120.953	117.338	94.600	92.992	86.713	120.519	102.647	102.817	145.474	151.230	154.097	105.636	115.788	105.631	123.363	129.194	122.264
Min. training time per epoch	78.924	78.965	78.985	78.720	78.321	79.017	87.778	87.346	87.443	63.184	62.948	62.994	76.325	76.064	76.168	48.058	46.801	46.699
Max. training time per epoch	140.241	120.953	117.338	94.600	92.992	86.713	120.519	102.647	102.817	145.474	151.230	154.097	105.636	115.788	105.631	123.363	129.194	122.264
Avg. training time per iteration	1.07E+00	1.03E+00	1.02E+00	9.64E-01	1.40E-02	1.38E-02	1.12E+00	1.55E-02	1.56E-02	9.37E-01	1.39E-02	1.40E-02	2.43E+00	2.48E+00	2.82E-02	1.87E+00	1.87E+00	2.13E-02
Final training loss	5.92E+02	5.97E+02	5.83E+02	2.49E+02	2.39E+02	2.80E+02	6.24E+02	6.36E+02	1.88E+03	5.92E+02	7.12E+02	5.95E+02	6.66E+02	1.36E+03	6.38E+02	6.63E+02	6.65E+02	1.97E+03
Final validation loss	5.31E+02	6.19E+02	5.83E+02	8.54E+02	1.24E+03	2.39E+03	5.36E+02	6.28E+02	2.31E+03	5.94E+02	7.05E+02	7.15E+02	6.51E+02	6.68E+02	564.3922	7.26E+02	7.41E+02	2.39E+03
Saving model time	Not calculated	Not calculated	0.425	Not calculated	0.094	0.076	Not calculated	0.700	0.755	Not calculated	0.158	0.602	Not calculated	Not calculated	0.081609	Not calculated	Not calculated	0.108318
Node ID	jwb0001	jwb0001	jwb0001	jwb0001	jwb0172	jwb0406	jwb0065	jwb0393	jwb0012	jwb0129	jwb0085	jwb0117	jwb0065	jwb0053	jwb0965	jwb0085	jwb0129	jwb0117
Max. GPU power	256.78 W	258.27 W	258.33 W	253.87 W	240.13 W	248.20 W	239.22 W	256.96 W	261.56 W	177.36 W	190.75 W	178.01 W	274.13 W	267.92 W	277.43 W	219.47 W	244.72 W	228.39 W
GPU energy consumption	12.29 Wh	13.38 Wh	11.12 Wh	10.65 Wh	10.91 Wh	9.86 Wh	11.34 Wh	11.76 Wh	13.85 Wh	10.50 Wh	11.09 Wh	12.16 Wh	12.87 Wh	10.20 Wh	10.48 Wh	10.22 Wh	7.04 Wh	10.49 Wh
Experiment flags				1	1	1	2	2	2	3	3	3	4	4	4	5	5	5

Table 5: AP3 JUWELS Booster training benchmark



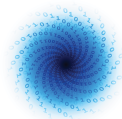
Experiment number	1	2	3	4	5	6
Job ID	5217303	5221634	5221743	5240098	5240195	5244184
#Nodes	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1
#CPUs	8	8	8	8	8	8
Loading data time	9.00374	9.553724	7.530015	8.761098	3.018038	10.541473
Total runtime	836.413651	836.97661	830.49475	811.495556	788.535332	810.530319
Total training time	825.993124	826.098056	821.806106	801.37837	785.142682	798.869927
Avg. training time per epoch	165.198625	165.219611	164.361221	160.275674	157.028536	159.773985
First epoch training time	200.695661	199.369829	199.480087	176.896373	161.182102	175.378102
Min. training time per epoch	156.018997	156.506337	155.199191	155.388412	155.819325	154.7021
Max. training time per epoch	200.695661	199.369829	199.480087	176.896373	161.182102	175.378102
Avg. training time per iteration	0.028336	0.02834	0.028192	0.027492	0.026935	0.027405
Final training loss	608.533813	632.423584	634.193298	283.179626	250.776031	178.831802
Final validation loss	594.885681	619.718384	659.64032	2450.437256	2419.190918	1568.444092
Saving model time	1.064228	0.924225	0.821395	1.003861	0.077052	0.690811
Node ID	jwc09n096	jwc09n069	jwc09n078	jwc09n075	jwc09n075	jwc09n093
Max. GPU power	250.76 W	276.44 W	259.93 W	262.69 W	272.52 W	289.63 W
GPU energy consumption	46.46 Wh	52.91 Wh	58.24 Wh	53.65 Wh	53.53 Wh	48.10 Wh
Experiment flags				1	1	1

Table 6: AP3 JUWELS Cluster training benchmark



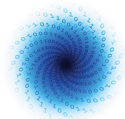
Experiment number	1	2	3	4	5	6	7	8	9	10	11	12
Job ID	106	350	351	354	355	356	107	108	110	111	208	209
#Nodes	1	1	1	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1	1	1	1
#CPUs	8	8	8	8	8	8	8	8	8	8	8	8
Loading data time	6.183	1.814218	2.907598	1.825382	1.889511	1.880253	5.958	7.571	5.906	14.148	19.492	11.548
Total runtime	397.517	394.481124	394.303094	376.322659	377.431983	376.628997	379.658	380.638	380.367	1,407.753	896.041	882.983
Total training time	391.143	392.375967	391.099987	374.205191	375.251604	374.454181	373.511	372.829	374.270	1,393.038	875.967	871.055
Avg. training time per epoch	0.281	78.475193	78.219997	74.841038	75.050321	74.890836	0.268	0.268	0.269	1.000	0.629	0.625
First epoch training time	94.636	95.083092	95.067289	77.606224	77.693101	77.681781	77.357	77.412	78.000	641.000	579.208	575.733
Min. training time per epoch	73.755	73.946844	73.604111	73.807829	74.075026	73.96691	74.436	73.520	73.703	74.035	73.621	73.466
Max. training time per epoch	94.636	95.083092	95.067289	77.606224	77.693101	77.681781	77.357	77.412	78.000	641.000	579.208	575.733
Avg. training time per iteration	2.69E-01	0.013461	0.013417	0.012837	0.012873	0.012846	2.57E-01	2.57E-01	2.58E-01	9.59E-01	6.03E-01	6.00E-01
Final training loss	6.10E+02	596.397949	638.586365	310.563568	375.341797	206.835526	6.51E+02	6.35E+02	6.56E+02	5.86E+02	6.29E+02	6.65E+02
Final validation loss	5.24E+02	572.302734	659.216858	2976.624756	1100.272095	4658.901855	7.23E+02	6.35E+02	6.14E+02	6.35E+02	6.23E+02	9.62E+02
Saving model time	Not calculated	0.100865	0.105034	0.101347	0.099817	0.100544	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated	Not calculated
Node ID	icnode02	icnode01	icnode01	icnode01	icnode01	icnode01	icnode02	icnode01	icnode01	icnode01	icnode01	icnode01
Max. GPU power	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
GPU energy consumption	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Experiment flags				1	1	1	2	2	2	2 + cleared NFS cache by reboot	2 + cleared NFS cache by reboot	2 + cleared NFS cache by reboot

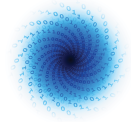
Table 7: AP3 E4 Intel+NVIDIA training benchmark



Experiments	1	2	3	4	5
Job ID	5001975	5001986	5031242	5003338	5088400
Total runtime (in seconds)	82.47	75.07	84.47	42.85	43.13
Total inference time	76.65	70.28	77.85	36.92	37.76
Maximum training time per iteration	NA	NA	NA	NA	NA
Minimum training time per iteration	NA	NA	NA	NA	NA
Estimated Average training time per iteration	1349.85	1237.66	1370.88	650.26	664.90
Final inferenced loss function defined by the applicatoin lead	58.1475	58.1474	58.1475	54.0472	54.0472
Inference time per sample	1.32E+00	1.21E+00	1.34E+00	6.35E-01	6.49E-01
#Nodes	1	1	1	1	1
#GPUs	1	1	1	1	1
#MPI Tasks	1	1	1	1	1
#Cores	8	8	8	8	8
Data size (volume and number of samples)	60Gb, 2984960 samples	60Gb, 2984960 samples	60Gb, 2984960 samples	60Gb, 2984960 samples	60Gb, 2984960 samples
If multi-node/multi-GPU: communication time	NA	NA	NA	NA	NA
If multi-node/multi-GPU: communication volume	NA	NA	NA	NA	NA
If possible: energy consumption	1.16 Wh	0.83 Wh	NA	NA	NA
Experiment flags	4	4	4	4+1	4+1
Experiment notes				Last two jobs failed to create job reports, too short?	

Table 8: AP3 JUWELS Booster inference benchmark





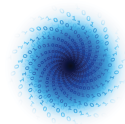
6.3 AP 4

Experiment number	1	2	3	4
Job ID	5168001	5168098	5168101	5234855
#Nodes	1	1	1	1
#GPUs	1	1	1	1
#MPI tasks	1	1	1	1
#CPUs	1	1	1	1
Loading data time	490.000	111.400	64.000	917.000
Total runtime	6,701.000	6,322.400	6,275.000	7128
Total training time	4,445.000	5,062.000	5,021.000	5866
Avg. training time per epoch	740.000	847.000	836.800	977.6
First epoch training time	1,109.000	853.000	821.000	1095
Min. training time per epoch	831.000	840.000	821.000	881
Max. training time per epoch	1,109.000	853.000	872.000	1095
Avg. training time per iteration	1.14E+00	1.08E+00	1.07E+00	1.24
Final training loss	1.20E-01	1.20E-01	1.20E-01	0.121
Final validation loss	2.00E-01	2.00E-01	2.00E-01	0.2
Saving model time	-	-	-	-
Node ID	jwb0001	jwb0053	jwb0065	jwb0129
Max. GPU power	412.85 W	403.39 W	388.53 W	391.33 W
GPU energy consumption	412.85 W	157.61 Wh	135.66 Wh	159.2712

Table 9: AP4 JUWELS Booster training benchmark

Experiment number	1	2
Job ID	399	400
#Nodes	1	1
#GPUs	1	1
#MPI tasks	1	1
#CPUs	8	8
Loading data time	1,751.000	1,749.000
Total runtime	29284	29,267.000
Total training time	25,737.000	25,720.000
Avg. training time per epoch	5,147.400	5,144.000
First epoch training time	5,175.000	5,124.000
Min. training time per epoch	5,122.000	5,124.000
Max. training time per epoch	5,175.000	5,168.000
Avg. training time per iteration	2.06E+02	2.07E+02
Final training loss	1.20E-01	1.20E-01
Final validation loss	2.00E-01	1.99E-01
Saving model time	-	-
Node ID	icnode01	icnode02
Max. GPU power	-	-
GPU energy consumption	-	-
Experiment flags	batch-size = 1, Epochs=5	batch-size = 1, Epochs=5

Table 10: AP4 E4 Intel+NVIDIA training benchmark



6.4 AP 5

Experiment number	1	2	3	4	5	6	7	8	9	10
Job ID	4971144	4973701	4973716	4973721	4973739	4973760	4973783	4973816	4973840	4973857
#Nodes	1	1	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1	1	1
#CPUs	1	1	1	1	1	1	1	1	1	1
Loading data time	2.941	3.015	1.970	1.975	1.959	1.946	2.436	2.986	3.024	3.214
Total runtime	80.766	83.162	69.904	70.531	69.832	70.019	73.404	83.853	82.470	83.000
Total training time	73.994	76.126	65.830	66.472	65.798	65.749	67.946	76.040	75.230	75.247
Avg. training time per epoch	1.057	1.088	0.940	0.950	0.940	0.939	0.971	1.086	1.075	1.075
First epoch training time	20.532	22.708	12.391	13.075	12.409	12.356	14.512	22.629	21.701	21.801
Min. training time per epoch	0.774	0.773	0.774	0.773	0.773	0.773	0.773	0.773	0.775	0.774
Max. training time per epoch	0.777	0.798	0.776	0.774	0.775	0.776	0.808	0.776	0.778	0.802
Avg. training time per iteration	1.44E-03	1.49E-03	1.28E-03	1.30E-03	1.28E-03	1.28E-03	1.33E-03	1.48E-03	1.47E-03	1.47E-03
Final training loss	1.61E-01	1.21E-01	1.51E-01	1.54E-01	1.38E-01	1.44E-01	1.44E-01	1.42E-01	1.51E-01	1.73E-01
Final validation loss	8.89E-02	8.46E-02	8.94E-02	8.28E-02	8.36E-02	8.61E-02	8.79E-02	8.48E-02	8.22E-02	8.52E-02
Saving model time	0.270	0.243	0.196	0.189	0.181	0.443	0.186	0.270	0.229	0.327
Node ID	jwb0065	jwb0053	jwb0053	jwb0053	jwb0053	-	jwb0053	jwb0053	jwb0001	jwb0001
Max. GPU power	299.24 W	64.50 W	63.21 W	285.37 W	63.75 W	-	63.21 W	63.21 W	58.70 W	86.16 W
GPU energy consumption	3.21 Wh	0.36 Wh	1.42 Wh	2.11 Wh	1.07 Wh	-	0.89 Wh	2.22 Wh	1.20 Wh	1.79 Wh

Table 11: AP5 JUWELS Booster training benchmark (small dataset)

Experiment number	1	2	3	4	5	6	7
Job ID	5182831	5183303	5183892	5184088	5184458	5186284	5188612
#Nodes	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1
#CPUs	1	1	1	1	4	4	4
Loading data time	7.670	7.906	7.819	8.017	7.750	7.778	5.677
Total runtime	1,534.515	1,534.272	1,538.915	1,529.947	1,550.244	1,528.083	1,513.865
Total training time	1,509.511	1,508.399	1,511.840	1,504.842	1,499.402	1,501.004	1,491.081
Avg. training time per epoch	21.564	21.549	21.598	21.498	21.420	21.443	21.301
First epoch training time	38.337	38.731	41.637	36.709	41.897	41.080	33.263
Min. training time per epoch	21.308	21.282	21.291	21.267	21.056	21.065	21.008
Max. training time per epoch	21.329	21.322	21.316	21.288	21.503	21.553	21.519
Avg. training time per iteration	1.05E-03	1.05E-03	1.05E-03	1.05E-03	1.05E-03	1.05E-03	1.04E-03
Final training loss	5.32E-02	5.34E-02	5.33E-02	5.36E-02	5.35E-02	5.37E-02	5.25E-02
Final validation loss	5.83E-02	5.86E-02	5.82E-02	5.83E-02	5.83E-02	5.85E-02	5.84E-02
Saving model time	0.262	0.251	0.664	0.275	0.204	0.225	0.237
Node ID	jwb0053	jwb0053	jwb0053	jwb0053	jwb0732	jwb0165	jwb0085
Max. GPU power	322.90 W	317.70 W	324.76 W	321.24 W	334.37 W	314.36 W	319.59 W
GPU energy consumption	50.94 Wh	50.34 Wh	48.06 Wh	49.17 Wh	49.32 Wh	50.24 Wh	44.85 Wh

Table 12: AP5 JUWELS Booster training benchmark (large dataset)

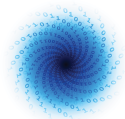


Experiment number	1	2	3	4	5
Job ID	5189851	5190745	5191505	5192560	5203052
#Nodes	1	1	1	1	1
#GPUs	1	1	1	1	1
#MPI tasks	1	1	1	1	1
#CPUs	4	4	1	1	1
Loading data time	12.849	10.211	16.080	14.217	13.997
Total runtime	2,824.686	2,828.528	2,829.731	2,831.227	2,843.450
Total training time	2,780.171	2,786.616	2,782.327	2,779.307	2,795.933
Avg. training time per epoch	39.717	39.809	39.748	39.704	39.942
First epoch training time	53.566	53.829	53.777	64.595	63.387
Min. training time per epoch	39.498	39.555	39.514	39.304	39.551
Max. training time per epoch	39.532	39.624	39.561	39.370	39.627
Avg. training time per iteration	1.94E-03	1.94E-03	1.94E-03	1.94E-03	1.95E-03
Final training loss	5.35E-02	5.34E-02	5.28E-02	5.34E-02	5.34E-02
Final validation loss	5.87E-02	5.82E-02	5.84E-02	5.82E-02	5.85E-02
Saving model time	0.281	0.222	0.224	1.033	0.844
Node ID	jwc09n000	jwc09n000	jwc09n000	jwc09n177	jwc09n000
Max. GPU power	300.32 W	298.51 W	293.87 W	282.22 W	299.95 W
GPU energy consumption	206.47 Wh	210.17 Wh	206.08 Wh	194.65 Wh	208.87 Wh

Table 13: **AP5** JUWELS Cluster training benchmark (large dataset)

Experiment number	1	2	3	4	5	6	7	8
Job ID	345	346	347	361	364	368	371	372
#Nodes	1	1	1	1	1	1	1	1
#GPUs	1	1	1	1	1	1	1	1
#MPI tasks	1	1	1	1	1	1	1	1
#CPUs	8	8	8	1	1	1	1	1
Loading data time	82.330	5.753	5.570	82.741	5.786	82.151	5.793	5.588
Total runtime	1,708.986	1,588.292	1,593.658	1,685.549	1,574.241	1,675.573	1,570.149	1,574.974
Total training time	1,597.712	1,557.484	1,563.032	1,573.712	1,543.239	1,567.276	1,541.177	1,544.238
Avg. training time per epoch	22.824	22.250	22.329	22.482	22.046	22.390	22.017	22.061
First epoch training time	43.093	28.495	28.564	43.229	28.268	36.943	28.246	28.229
Min. training time per epoch	22.121	21.774	21.869	21.719	21.517	21.772	21.500	21.527
Max. training time per epoch	22.607	22.216	22.288	22.236	22.012	22.223	21.960	22.016
Avg. training time per iteration	1.11E-03	1.09E-03	1.09E-03	1.10E-03	1.08E-03	1.09E-03	1.07E-03	1.08E-03
Final training loss	5.44E-02	5.32E-02	5.36E-02	5.31E-02	5.34E-02	5.38E-02	5.30E-02	5.59E-02
Final validation loss	5.88E-02	5.91E-02	5.84E-02	5.85E-02	5.83E-02	6.02E-02	5.83E-02	5.87E-02
Saving model time	0.565	0.493	0.528	0.606	0.490	0.491	0.545	0.496
Node ID	icnode02	icnode02	icnode02	icnode02	icnode02	icnode02	icnode02	icnode02
Max. GPU power	n/A	n/A	n/A	n/A	n/A	n/A	713.993 W	n/A
GPU energy consumption	n/A	n/A	n/A	n/A	n/A	n/A	309.29 Wh	n/A
Experiment note	first exp. on 03-11	cached data?	cached data?	first exp. on 03-12	cached data?	exp. after 3h break	tracked power	cached data?

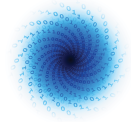
Table 14: **AP5** E4 Intel+NVIDIA training benchmark (large dataset)





Experiments	1	2	3	4
Job ID	5193124	5193173	5193174	5193197
Total runtime (in seconds)	18.78	18.19	17.95	17.21
Total inference time	11.17	11.01	10.88	10.89
Maximum training time per iteration	8.74	6.7	6.71	4.43
Minimum training time per iteration	0.01	0.01	0.01	0.01
Estimated Average training time per iteration	0.165	0.1	0.106	0.07
Final inferenced loss function defined by the applicatoin lead	1.0957	1.0957	1.0957	1.0957
Inference time per sample	4.43E-03	4.37E-03	4.32E-03	4.32E-03
#Nodes	1	1	1	1
#GPUs	1	1	1	1
#MPI Tasks	1	1	1	1
#Cores	1	1	1	1
If multi-node/multi-GPU: communication time	-	-	-	-
If multi-node/multi-GPU: communication volume	-	-	-	-
Node ID	jwb0097	jwb0085	jwb0097	jwb0097

Table 15: AP5 JUWELS Booster inference benchmark



6.5 AP 6

Experiment number	1	2	3	4	5	6	7	8	9
Job ID	5137253	5137254	5137255	5137256	5137257	5137258	5137259	5137260	5137261
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	0	0	0	0	0	0	0	0	0
#MPI tasks									
#CPUs	96	96	96	96	96	96	96	96	96
Loading data time	0.077	0.074	0.079	0.075	0.066	0.077	0.071	0.058	0.079
Total runtime	4,430.173	4,471.083	4,484.009	4,460.041	4,466.300	4,452.089	4,460.984	4,481.748	4,471.480
Node ID	jwb0035	jwb0683	jwb0120	jwb0022	jwb0574	jwb0213	jwb0726	jwb0345	jwb0873

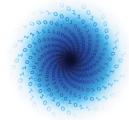
Table 16: AP6 JUWELS Booster benchmark

Experiment number	1	2	3	4	5	6	7	8	9
Job ID	5126457	5126458	5126459	5126460	5126461	5126462	5126463	5126465	5126477
#Nodes	1	1	1	1	1	1	1	1	1
#GPUs	0	0	0	0	0	0	0	0	0
#MPI tasks									
#CPUs	96	96	96	96	96	96	96	96	96
Loading data time	0.15	0.13	0.10	0.12	0.21	0.09	0.09	0.12	0.08
Total runtime	2964.63	2919.65	2953.26	2962.05	2997.01	2938.25	2942.33	3002.82	2989.50
Node ID	jwc02n005	wc02n006	jwc02n007	jwc02n008	jwc02n009	jwc02n010	jwc00n227	jwc00n231	jwc02n011

Table 17: AP6 JUWELS Cluster benchmark

Experiment number	1	2	3	4	5	6	7
Job ID	291	305	306	307	89	93	92
#Nodes	1	1	1	1	1	1	1
#GPUs	0	0	0	0	0	0	0
#MPI tasks							
#CPUs	64	8	16	32	8	16	32
Loading data time	0.020	0.023	0.022	0.025	0.025	0.022	0.022
Total runtime	2,822.354	2,989.960	2,820.132	2,794.057	3,817.864	3,874.626	3,815.817
Node ID	icnode01	iwnode02	iwnode02	iwnode01	awnnode04	awnnode03	awnnode04

Table 18: AP6 E4 benchmarks



Document History

Version	Author(s)	Date	Changes
	Name (Organisation)	dd/mm/yyyy	
0.6	JSC	31/03/2022	Start internal review
0.8	JSC	03/04/2022	Add missing data
0.9	JSC	06/04/2022	Incorporate comments, tables in appendix
1.0	JSC	11/04/2022	Final version

Internal Review History

Internal Reviewers	Date	Comments
Name (Organisation)	dd/mm/yyyy	
Karthick PANNER SELVAM (UL-SnT)	05/04/2022	Accepted with minor revisions
Thomas Niepen (MetNor)	08/04/2022	Accepted with minor revisions

Estimated Effort Contribution per Partner

Partner	Effort
FZJ	2.6 PM
MetNor	0.3 PM
ECMWF	0.3 PM
ETHZ	0.3 PM
4-cast	0.3 PM
Total	3.8 PM