MAchinE Learning for Scalable meTeoROlogy and climate

MAELSTROM

Hardware-Related Requirements

Kaveh Haghighi-Mood, Andreas Herten,
Albert Kahira, Stepan Nassyr

www.maelstrom-eurohpc.eu

MAELSTROM

D3.1 Initial list of hardware-related requirements for ML solutions in W&C

Author(s):	Kaveh Haghighi-Mood (FZJ), Andreas Herten (FZJ), Albert Kahira (FZJ), Stepan Nassyr (FZJ),

Dissemination Level:	Public
Date:	30/09/2021
Version:	1.0
Contractual Delivery Date:	30/09/2021
Work Package/ Task:	WP3/ T3.1
Document Owner:	FZJ
Contributors:	MetNor, 4cast
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

Contents

Tables

1 Executive Summary

The various MAELSTROM applications target different use-cases, use different software frameworks, and utilize different hardware features. This document provides an overview of the requirements of the applications regarding the hardware used for computation.

In a first step, requirements are laid out as provided by the respective applications, and the information is turned into a coherent overview afterwards.

The document only provides a snapshot overview of the current state of requirements at the beginning of the MAELSTROM project. With future developments of the applications, the growing MAELSTROM ecosystem, and progressing hardware solutions, the hardware requirements will also change explicitly and implicitly.

2 Introduction

2.1 About MAELSTROM

To develop Europe's future computer architecture, 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 computing 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 computer system designers.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 a 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 3.1 is a report on the results of the work done for Task 3.1, a systematic analysis of the hardware requirements of the target applications with a focus on accelerators.

Deliverable 3.1 is one of four MAELSTROM deliverables that survey the state-of-the-art in terms of methods, tools, and developments at the beginning of the project and aim to build additional links between the three work packages involved in the MAELSTROM co-design cycle. Deliverable 1.2 is a survey of machine learning methods and tools that are currently used for weather and climate applications. Deliverable 2.1 is a survey of existing machine learning workflow tools and a summary of the MAELSTROM protocol and machine learning requirements. Deliverable 3.1 and 3.2 provide a systematic analysis of the hardware requirements for the MAELSTROM applications and a roadmap analysis of hardware that will be relevant for machine learning in MAELSTROM.

2.2.2 Work performed in this deliverable

A questionnaire[1] is provided to the developers of each of the six MAELSTROM applications in order to evaluate the respective initial hardware requirements. The acquired data from the questionnaires is refined with assessments from personal interviews with application developers, to foster interactive discussions. Finally, developers iteratively gave feedback on the compiled requirements, summarized in this deliverable.

[1] See Initial Hardware Questionnaire in Appendix.

3 Hardware Usage

This section briefly describes the hardware requirement based on the initial evaluations of the characteristics of the applications.

3.1 Application 1

Blend citizen observations and numerical weather forecasts

The main objective of Application 1 (A1) is the training of a neural network with a grid of training data in the order of TB to improve short-range forecasts of temperature and precipitation.

TensorFlow 2 (Python, C++) and Flux (Julia) are used for the core of A1 on externally pre-processed data. FP32-based calculations are dominant. A minimum of 64 GB of main memory on the host side and 16 GB on the accelerator is required. A minimum of 10 TB of storage is necessary.

Currently, a local cluster is used with 28 nodes equipped with two Intel Xeon 6140 CPUs, each with 18 physical cores. Four of the nodes have a Nvidia P100 GPU. Nodes are interconnected via InfiniBand with 100 GBit/s available bandwidth. However, there is currently no inter-node communication, as training is done on a single node with data that fits the CPU memory. A 13 PB Lustre file system is used for storage.

A reduced workload exists as a test case to allow for research with fast turn-around time (with about 5 GB of input data). The proper workload consists of 10 TB of input data. The data is contained in multiple NetCDF files. In the future, TFRecords or other formats, which allow for efficient input pipelines and distributed data processing will be used.

Extending the application to use modern GPU accelerators (like Nvidia A100) can bring significant performance improvements, and this hardware is targeted to be used for MAELSTROM. Improvements in strategies for data read-in are also investigated to accompany the increase in computational performance.

3.2 Application 2

Incorporate social media data into the prediction framework

Within Application 2 data from social media are loaded and searched for weather-related content. This content is analysed and processed such that it can be used as additional input for weather models. The combination of social media data and weather predictions could potentially be used to develop customised weather predictions for individual users.

Scikit-learn (Python), TensorFlow 2, and PyTorch (Python, C++) are used for the application with external pre-processing. Calculations are mainly double-precision floating points (FP64) using at least 16 GB of RAM. A minimum of 2 TB storage is needed for the (full) data set.

At the moment, a small cluster of four nodes with desktop-grade Intel i7 CPUs and 64 GB of RAM on each is used, as well as flexible cloud resources.

A test data set for research with about 2 GB of input data is under development, assuming approval of the project at Twitter; the full data set (2 TB) is expected later on. At the moment, it is still unclear which load the full dataset generates. Part of the data set is also used by other applications (see D1.1) which A2 will combine with further data. It is likely that the required computing resources will also be increased over the requirements of the other applications.

For the future, utilization of larger data sets as well as leveraging GPUs to accelerate training are planned. It is too early for an accurate estimation of required HPC resources. Communication between nodes is not expected to be a significant bottleneck, if needed at all. Distribution into independent tasks is planned.

3.3 Application 3

Build neural network emulators to speed-up weather forecast models and data assimilation

A3 will investigate the emulation of physical parameterization by employing neural networks. Building efficient neural networks that provide solutions with sufficient quality is the primary goal of the application.

TensorFlow 2 is the framework chosen for this application. Some pre-processing is performed prior to main calculations using Fortran, especially for preparing inference data. Calculations mainly use single-precision floats (FP32). At least 16 GB of RAM is needed as well as storage space for the data set in order of TB.

A local cluster is currently used for development, consisting of nodes equipped with single Intel Xeon Gold 6254 (*Cascade Lake*) CPUs, 32 GB of RAM, and a single Nvidia V100 GPU. 5 TB of storage is available on the cluster.

The size of datasets for learning is in the order of TB, for testing a factor of 10 lower (order of 100 GB), and for inference in the order of GB. Current training is using single GPU calculations, hence no internode communication is necessary. Internode network fabric would play a crucial role in the case of multi-GPU training, which is on the horizon as the next development step.

Pre-processing for inference is done in a distributed fashion amongst many nodes. The communication between nodes in this stage can potentially become a bottleneck for larger weather models. On the other hand, pre-processing can be rewritten to utilize most of the new hardware, and reassessments are recommended with the project's progress.

For the future, lower-precision computations in FP16 are going to be evaluated, as well as using newer hardware (AMD EPYC CPUs, Nvidia A100 GPUs).

3.4 Application 4

Improve ensemble predictions in forecast post-processing

A4 will investigate whether probability distribution estimates of weather forecasts can be improved by post-processing with deep learning. For weather predictions, ensemble simulations are used to make estimates about probability distributions. The more ensemble members are available, the better the prediction, but the more expensive the forecast. A4 is using an ensemble with five ensemble members as input and corrects the ensemble mean error and the ensemble standard deviation with the aim to obtain results that are equally good as predictions with a larger number of ensemble members.

PyTorch is used as the basis for A4, with pre-processing using homegrown Python/C++ libraries. FP32 calculations are dominant, and FP16 floats are also used. A minimum amount of 256 GB of RAM is needed on the host, 32 GB RAM on the GPU. Four GPUs are needed for training (exchanging data with each other frequently); for hyperparameter studies, multiple loosely-connected nodes are employed as an ensemble.

Currently, clusters with datacenter-grade nodes are used containing Intel Xeon Gold 6140 CPUs with 512 GB host-side RAM, 4 Nvidia V100 GPUs with 32 GB RAM (connected solely via PCIe Gen 3). The nodes are interconnected with InfiniBand, but the application does not leverage multiple nodes. Future improvements will consider extending the application to multiple nodes. In addition to fast, node-local storage for staging, permanent storage with 4 TB for the data set, model weights, and checkpoints are used.

While the main data set is about 3 TB of size, a reduced data set with 9 GB exists for research by the community. The input data is pre-processed extensively offline before the training, using raw input GRIB data to create multiple binary-format files of 5D-multidimensional data continuous in time. Every sample in the data set is read once per epoch and randomly distributed amongst the GPUs (as minibatches of 1-16 samples) for training.

In the future, larger data sets (up to 15 TB) will be employed, as well as faster GPUs. In addition, training will be done on more than one node, requiring fast interconnect between the nodes ideally matching the currently limiting PCIe Gen 3 exchange bandwidth (16 GB/s per direction and GPU).

3.5 Application 5

Improve local weather predictions in forecast post-processing

A5 aims to predict local temperature by developing a neural network that is taking weather predictions at a coarse resolution as input to predict the weather situations at a higher spatial resolution. This process is called “downscaling” in the weather and climate prediction community.

Besides TensorFlow 2 as the cornerstone framework, other libraries like Horovod (Python, C++) and NetCDF (C) are used for A5. Calculations are mainly single-precision floats (FP32). A minimum of 96 GB of RAM on the host and 32 GB RAM on the accelerator is necessary. Storage to the extent of 10 TB is required. Pre-processing is done using the Climate Data Operators software and homegrown Python scripts on CPU.

Currently, the JUWELS (Cluster and Booster) supercomputer is used for development. JUWELS Cluster nodes are equipped with 2 Intel Xeon Gold 6148 CPUs with 192 GB RAM and 4 Nvidia V100 (16 GB) GPUs and 2 InfiniBand EDR cards; JUWELS Booster nodes contain 2 AMD EPYC Rome 7402 CPUs with 512 GB RAM and 4 Nvidia A100 (40 GB) GPUs and 4 InfiniBand HDR200 adapters. They are connected to the GPFS parallel file-system JUST, with multiple Petabyte of storage available, and HPST, offering high-throughput storage.

In addition to the full data set (about 10 TB size), a test case data set of 5 GB size exists for development purposes. Raw data is preprocessed to yield netCDF files. Number and size of files are chosen to avoid single large and small files (that is, between 100 MB and 10 GB). Best formats for data streaming are currently under investigation.

Currently, single nodes are used for training and no necessity for the interconnect between nodes exists. Once the application matures, multi-node training is required. In that case, the existing HPC-grade InfiniBand fabric on JUWELS will likely be sufficient and is not expected to be a critical bottleneck in the foreseeable future.

The use of larger data sets and faster GPUs with larger memories and/or more nodes are potentially beneficial and will be investigated in the future. Employment of FP16 and mixed-precision training will be explored.

3.6 Application 6

Provide bespoke weather forecasts to support energy production in Europe

Application 6 is focused on developing new and better AI solutions to forecast the energy production of the near and mid-term future by fusing simulation and measurement. Weather prediction data is used as input of deep learning tools to predict the weather measurements close to the sites that produce energy. This allows to derive customised weather predictions for the energy production site based on future weather predictions.

PyTorch is the framework of choice for this A6, with additional tooling in Python and Scala. Calculations are mainly double-precision floating points (FP64), at least 64 GB RAM is needed. The size of the data set is in the order of TBs.

Currently, a small cluster of four nodes with desktop-grade Intel i7 CPUs and 64 GB of RAM on each as well as different cloud resources (AWS, GCP) are used for development purposes. A larger number of nodes is expected to be used for larger datasets to match memory requirements of the application and availability in the compute nodes (512 GB for JUWELS Booster, for example)

The setup of A6 is currently under development and prototyped by a reduced dataset. At the current stage, only feature engineering requires communication between nodes. Ethernet is used at the moment and might need to be replaced by an HPC-grade network fabric when the full dataset is going to be used.

In the near future, A6 will investigate usage of larger data sets. For this purpose, GPUs and distributed algorithms will be used. Ideally with independent optimisation, e.g. for fast hyperparameter tuning based on subsets (train/test split) of data, requiring little inter-node communication.

4 Hardware Requirements

4.1 Methodology of Characterisation

Based on the overview provided in Section 3, a list of hardware requirements can be created on a per-application basis.

The requirements can be summarized according to the following key aspects:

- Host CPU
- Host Memory
- Accelerator
- Accelerator Memory
- Storage
- Network Fabric

Also, Frameworks/Libraries and Arithmetic Precision are of importance, as they usually have strong ties to the underlying hardware.

While quantitative requirements can be given for memory and storage driven by data, we provide technology levels for CPU, GPU, and Network capabilities to match the diverse and multi-facetted respective hardware landscape in a vendor-agnostic way.

MAELSTROM, from its inception, has a clear path forward to improve its 6 applications. Each of the applications will soon be extended and improved, not only in terms of algorithmic precision, scope, and quality, but also in terms of hardware-related aspects. To respect that, we also take note of imminent upcoming hardware requirements of the near future (also see D3.2). It is expected that the hardware requirements will change in the future as the size of training datasets will grow, and the ability of applications to train in parallel will be extended.

4.2 Requirements Overview

A detailed summary of key hardware requirements by the six MAELSTROM applications can be found in Table 1.

Four of the six applications utilize accelerators to speed up their workload. In all cases, Nvidia GPUs are chosen. The devices are all HPC-grade, ranging from current-generation devices (A100) to previous-generation devices (V100) to older devices (P100), with the majority utilizing Nvidia V100 GPUs. For the GPU-using applications, the hosting CPUs seem to be of rather subordinate importance, as the available performance of the GPU vastly surpasses the performance of the CPU. Still, CPU performance is not unimportant as pre-processing is mostly done with home-grown applications typically on host side and not the large, GPU-accelerated ML/DL frameworks used for training. In all four cases, HPC-grade CPUs (Intel, AMD) with many cores and SIMD support are used.

Two applications don’t yet use accelerators, but are solely relying on CPU performance for their workloads. In both cases, Desktop-grade CPUs (Intel) are used.

Typical to ML/DL applications, especially from the W&C community, demands on storage are intense as it is needed to host input data and intermediate data for training and inference. Two applications need storage space in the order of 10 TB; two in the order of 1 TB; one in the order of 100 GB. In three cases, a parallel file system (for example GPFS or Lustre) is used for storage, providing the necessary bandwidth to feed input data to the participating computing entities. During data preparation, data is split, partly shuffled, and sent to the respective devices.

Host memory is important for all applications, as even for the GPU-accelerated applications, data is staged and pre-processed through the host. The requirements to host memory are various and application-dependent. Two applications (A4 and A1) use multiple hundred MB of memory (512 GB and 192 GB, respectively); the other applications require between 96 GB and 32 GB. There is a correlation between the size of the input data set and the required memory. The memory of the accelerators (between 12 GB and 32 GB) is utilized fully by the GPU-enabled applications, with the application developers expressing explicit interest in using GPUs with larger device-side memory.

The speed/level of the interconnect between nodes currently is no limiting factor for the applications, as they currently employ little internode data exchange during training and rather use multiple nodes for ensemble studies. Going to larger workloads in the future will change that fact, but already now most respective working environments provide HPC-grade interconnect available to the applications (for example InfiniBand HDR200, the latest generation).

Available performance on a compute device can be dependent on the type of the performed computations. With GPU accelerators, the employed arithmetic precision is a key factor, with performance roughly doubling when halving precision. Most GPU-accelerated applications are using FP32 ("single precision") computations, while the CPU-only applications stick to FP64 ("double precision") as CPUs traditionally don't penalize higher arithmetic precisions.

ML/DL is heavily reliant on frameworks to enable training and inference. Via the frameworks, even advanced and intricate hardware features can be used easily. All applications are using either PyTorch or TensorFlow 2 which are both optimized for CPU and GPU features, partly by the respective vendors themselves.

The hardware-related outlook for the future matches the increase in computational load of the respective applications. While the CPU-only-using applications target first enablement and optimization on GPUs, applications already accelerated to GPUs aim for focusing on newer GPU devices with more performance and larger memory. At the same time, the workload of some applications will require extending the scope of one node to multiple nodes, which requires the according high-speed network fabric. Some advanced features of the computing devices are targeted explicitly in the future, like using a mix of reduced and full precision computing (FP32/FP16). Partly, demands to the storage will increase as well with the extension of the size of the input data sets.

Table 1 is printed on the next page.

Table 1: Requirements overview. Future hardware/features are written in italics

Application	Host CPU	Host Memory	Accelerator	Accelerator Memory	Storage	Network Fabric	Arithmetic Precision	Frameworks / Libraries
A1	HPC-Grade	192 GB	GPU, HPC-Grade (Nvidia P100) *Nvidia A100*	12 GB	HPC-Grade Parallel File System (10 TB)	*High-speed*	FP32	TensorFlow 2, Flux
A2	Desktop-Grade	64 GB	N/A *GPU*	N/A	N/A	N/A	FP64	TensorFlow 2, PyTorch, Scikit-Learn
A3	HPC-Grade	32 GB	GPU, HPC-Grade (Nvidia V100) *Nvidia A100*	16 GB	O(TB)	N/A *High-Speed*	FP32 *FP16*	TensorFlow 2
A4	HPC-Grade	512 GB	GPU, HPC-Grade (Nvidia V100) *Nvidia A100*	32 GB	HPC-Grade Parallel File System (4 TB)	*High-Speed*	FP32	PyTorch
A5	HPC-Grade	96 GB	GPU, HPC-Grade (Nvidia V100, Nvidia A100)	32 GB *42 GB*	HPC-Grade Parallel File System O(10 TB)	*High-Speed*	FP32 *FP16 mixed*	TensorFlow 2
A6	Desktop-Grade	64 GB	N/A *GPU*	N/A	500 GB	N/A	FP64	PyTorch

5 Conclusion

In this document, after an initial assessment of the unique features and specifications of six MAELSTROM applications, the current hardware used by application owners is documented.

Four out of six applications currently use GPUs. The other two have plans to use accelerators. Multi-node calculations are used, for example, for hyperparameter search for few applications, but none need high-speed network fabric for training at the moment. The situation would change as soon as model parallelization is employed. Three applications leverage HPC-Grade Parallel File Systems, for the other three, storage is not currently a prime concern.

The document can be a valuable source for decision-making about hardware infrastructure in the later project stages.

6 Appendix

6.1 Initial Hardware Questionnaire

As mentioned in the methodology section, a questionnaire was sent to application owners to gather information on hardware specifications as the basis for the analysis.

Hardware

- State any well-known explicit requirements (leave out unknowns):
    - Per node
        - Compute performance (i.e X TFLOP/S with FP16/FP16+FP32 mixed/etc...)
        - Amount of RAM
        - Memory bandwidth
        - Amount of VRAM (if computing on GPU)
        - Amount of non-volatile memory (storage)
        - Speed of non-volatile storage
    - Internode
        - Interconnect bandwidth
        - Interconnect latency
        - Number of nodes
    - General performance limiters
- What hardware is being used for development? Please sketch details!
    - CPU
    - GPU
    - Other accelerators
    - Storage
    - Memory
    - Number of nodes
    - Network
- What hardware is being targeted?
    - CPU
    - GPU
    - Other accelerators
    - Storage
    - Memory
    - Number of nodes
    - Network
- What test cases do you have?
    - How big are your test cases?
    - How long does the test case take on development hardware?
    - How big are real workloads?
- What kind of and how much data is being stored on a node's permanent storage?
- What kind of and how much data is being stored on a node's volatile memory?
- What kind of and how much data is being exchanged between processors?
- What kind of and how much data is being exchanged between nodes?
- What parallelization and distribution strategies do you employ?

    - CPU cores (how many usually?)
    - MPI ranks (how many usually?)
    - GPU devices (how many usually?)
- Have you observed crashes when increasing/decreasing the problem size? If yes:
    - What was increased/decreased?
    - What kind of errors did you get?
- Have you observed crashes when increasing/decreasing the available resources?
    - What resource was increased/decreased? (number of cores/nodes/memory/network speed/...)
    - What kind of errors did you get?
- Have you observed unexpected drops or jumps in the performance when steadily increasing/decreasing the problem size?
    - What was increased/decreased?
    - How much did the performance decrease? (increase?)
- Have you observed unexpected drops or jumps in the performance when steadily increasing the available resources?
    - What resource was increased/decreased? (number of cores/nodes/memory/network speed/...)
    - How much did the performance decrease? (increase?)
- Did you run your software on multiple hardware platforms that differ significantly (i.e. AMD vs. NVIDIA GPUs vs. CPUs)?
    - What differences did you observe?
        - Memory usage
        - Runtime
        - HW Utilization

Hardware Ecosystem

- What frameworks and languages are being used for development?
    - Do you use vendor-specific frameworks or libraries? Which?
- Do you use vendor-specific features? (i.e. NVPTX assembly?)
    - How much of the code base is explicitly using this feature?
- Do you compile with specific flags, potentially implicitly targeting vendor-specific features? (-Ofast, -mavx2, …)

6.2 Document History

Version	Author(s)	Date	Changes
0.1	Herten, Nassyr	15/08/2021	Create Skeleton
0.8	Haghighi-Mood, Herten, Kahira, Nassyr	22/09/2021	Initial document
1.0	Haghighi-Mood, Herten, Kahira, Nassyr	30/09/2021	Refinements after review

6.3 Internal Review History

Internal Reviewers	Date	Comments
Peter Dueben (ECMWF)	29/09/2021	Accepted with minor revisions
John Bjørnar Bremnes (MetNor)	29/09/2021	Accepted with minor revisions

6.4 Estimated Effort Contribution per Partner

Partner	Effort
FZJ	2.5 PM
ECMWF	0.3 PM
MetNor	0.3 PM
Total	**3.1 PM**