A software ecosystem for multi-level provenance management in large-scale scientific workflows for AI applications

G. Padovani[1], V. Anantharaj[2], L. Sacco[1], T. Kurihana[2], M. Bunino[3], K. Tsolaki[3], M. Girone[3], F. Antonio[4], C. Sopranzetti[1], M. Fronza[1], S. Fiore[1]
[1]University of Trento, Trento, Italy
[2]Computational Sciences and Engineering Directorate, Oak Ridge National Laboratory, USA
[3]CERN, Geneva, Switzerland
[4]Euro-Mediterranean Center on Climate Change Foundation, Lecce, Italy

***Abstract*—Scientific workflows and provenance are two faces of the same medal. While the former addresses the coordinated execution of multiple tasks over a set of computational resources, the latter relates to the historical record of data from its original sources. This paper highlights the importance of tracking multi-level provenance metadata in complex, AI-based scientific workflows as a way to (i) foster and (ii) expand documentation of experiments, (iii) enable reproducibility, (iv) address interpretability of the results, (v) facilitate performance bottlenecks diagnosis, and (vi) advance provenance exploration and analysis opportunities.**

***Index Terms*—Provenance, workflow, ML tasks, multi-level.**

I. Introduction

Scientific workflows and provenance are two faces of the same medal. While the former addresses the coordinated execution of multiple tasks over a set of computational resources, the latter relates to the historical record of data from its original sources. Provenance represents valuable accompanying documentation for scientific data and experiments (and so workflows) providing historical context, documenting data transformations, and strenghtnening trust and authenticity of data. Additionally, provenance is a key enabling factor for reproducibility, thus playing a relevant role in Open Science. On the same line, provenance documents should also adhere to FAIR guiding principles to facilitate findability, accessibility, interoperability and reusability. However, due to the complexity of scientific workflows and applications, provenance needs to be managed across multiple tasks, heterogeneous services, different teams and departments as well as institutions, thus leading to the need for a web of FAIR provenance documents & services as well as tracking provenance information at multiple levels of granularity.

This paper focuses on yProv, a research project on multi-level provenance management which provides scientists with a rich software ecosystem consisting of a web service to manage provenance documents (yProv service), two libraries to track provenance in scientific experiments (yProv4ML and yProv4WFs) and a data science tool for provenance inspection, navigation, visualization, and analysis (yProv Explorer). In particular, this paper showcases how the yProv software ecosystem addresses multi-level provenance in complex scientific workflows including an in-depth provenance tracking concerning AI models training. This paper is organized as follows: Section II provides the motivation for a multi-level provenance software ecosystem and it discusses the current state of art in the area; Section III presents the key components of the yProv project, while Section IV discusses the application of the yProv provenance support to three scientific experiments in Climate Science, High Energy Physics and Earth Observation. Finally, Section V draws the final conclusions, highlighting future work.

II. State of the art

Tracking provenance of large-scale scientific workflows helps in understanding, verifying, and reproducing the results of computational processes, ensuring data integrity and trustworthiness [1]. Often, Workflow Management Systems (WfMS) use specific domain languages or do not adhere to common standards to create workflows [2]. The World Wide Web Consortium (W3C) has provided the PROV family of standards[1] to support the gathering and exchange of provenance information. Nevertheless, even by adopting the same standards, tracing provenance at different levels of granularity or using provenance services built for specific WfMS can make interoperability difficult. Especially when workflows need to transition from one execution environment to another and different components may be required to maintain control over the workflow execution [3]. To address this issue, a solution could be developing third-party tools to track data provenance across different environments and integrate metadata from these systems into a cohesive representation.

A first attempt of covering this subject has been done by the Common Workflow Language (CWL) team, providing a hierarchical provenance framework to standardize provenance and enable comprehensive, fully re-executable workflows, called CWLProv [4]. The latter extends the PROV-DM[2], providing

[1]https://www.w3.org/TR/prov-overview/
[2]https://www.w3.org/TR/prov-dm/

a representation for each of its elements, activity, entity and agent. It uses open source standards such as Research Objects [5], BagIt [6] and JSON-LD [7] to support interoperability and portability of the runtime environments. It supports multi-level provenance, giving a certain flexibility in the depth of provenance information captured, while on the other hand, it also has some weak points. Although widely adopted by many workflow design and execution platforms such as Rabix [8], Bcbio [9], Cromwell [10], etc., CWLProv can only be used by those WfMS which adopt the CWL language [11]. Thus, becoming a disadvantage in the case of using a different language to create a workflow. Furthermore, the use of CWLProv includes multiple metadata files and different PROV serializations to comply with different formats, making the generation of provenance documents and their utilization a bit complex. Another recent effort is Workflow Run RO_Crate [12], a framework for packaging research data alongside their metadata, with a focus on its application and extension in the context of computational workflows. It adopts both the Schema.org [13] vocabulary and the W3C PROV standard to capture provenance at different level of granularity. It has a flexible approach handling both prospective and retrospective provenance and it has been validated over six different WfMS, making it a first concrete proposal of a platform-independent service for tracking provenance. Conversely, some fragilities are noted: (i) the provenance document given in output in JSON-LD format appears complex, probably due to the articulated structure of the adoped data model; (ii) the input/output connections between the various steps of the workflow are not easily identifiable and the navigation of the provenance graph results therefore difficult. PROV-ML [14] is another attempt at creating a provenance data model for ML workflows, which revolves around the creation of a taxonomy consisting of three predetermined phases: data curation, data preparation and learning. Where this effort comes short is in this latter feature, limiting the end-user to a set of specific phases, and not allowing for full flexibility in the ML pipeline. With respect to prior work, yProv tries to address provenance challenges in a more holistic manner, from management, tracking, analysis and exploration through a seamless, consistent and interoperable rich software ecosystem.

III. The yProv project

A. The yProv service

The yProv[3] service is a software component aiming at addressing multi-level provenance management in end-to-end scientific workflows [16]. More specifically, it is a lightweight and interoperable service for provenance management compliant with the W3C PROV family of standards, exploiting a graph data model and exposing a well-defined RESTful API. From a software architecture perspective, the yProv service consists of three main components: (i) the yProv Web Service front-end, implemented in Python; (ii) a graph database engine back-end, relying on Neo4J; (iii) the yProv Command Line Interface (CLI)[4], providing a set of commands that are basically Python wrappers to the RESTful API calls.

The yProv API has been designed by following the REST principles to enable all the needed CRUD (Create, Read, Update, Delete) logic for managing provenance information. The yProv API can be used to perform many types of queries against the yProv service, thanks to the flexible query language support provided by the graph-database technology running in the back-end. These include extracting the full provenance document, retrieving sub-graphs or filtering on the provenance level in complex workflows. An in-depth view about the yProv service can be found in [16].

B. The yProv4WFs library

Fully addressing provenance in scientific experiments at the proper level of detail is a critical challenge in Open Science contexts. Tracking provenance at different level of granularity becomes a fundamental opportunity for scientists who can easily and quickly manage lineage information throughout the entire experiments life cycle (at level 1), drilling-down into a specific task (at level 2) when a more in-depth and detailed provenance analysis and exploration are needed. The multi-level provenance approach enables the navigation of specific provenance information across multiple axes, both horizontal and vertical, meeting complementary and multifaceted needs. Indeed, some users might be interested in the complete series of macro-tasks within the workflow, while others could be more fascinated by the data lineage details of a specific data-intensive (or a data-driven) step. Thus, the introduction of two different provenance perspectives is needed:

- a *coarse-grain* perspective refers to the overall set of tasks in a workflow and presents no distinction between workflows and their sub-workflows (yProv4WFs scope).
- a *fine-grain* perspective refers to in-depth details of a specific task (yProv4ML scope, w.r.t. AI training processes).

The possibility to widely explore the provenance space is then established, providing a way to switch from an horizontal dimension which provides an overall view of the end-to-end workflow, to a vertical one where a specific task can be deeply investigated (i.e., AI training tasks).
yProv4WFs[5] is the Python library in the yProv software ecosystem that addresses the coarse-grain provenance level. It collects provenance information at runtime, i.e., during the workflow execution, keeping track of both the overall workflow metrics and the ones related to each specific task. Flexibility is ensured by allowing the end-user to specify however many tasks are deemed necessary, not restricting the workflow to a sequence of predefined phases.

The gathered provenance information is then written into a JSON file, allowing easy sharing and evaluation of the experiments results. Such document is compliant with the W3C PROV standard and it is designed according to the data model reported in Fig. 1.

[3]https://github.com/HPCI-Lab/yProv

[4]https://github.com/HPCI-Lab/yProv-CLI

[5]https://github.com/HPCI-Lab/yProv4WFs

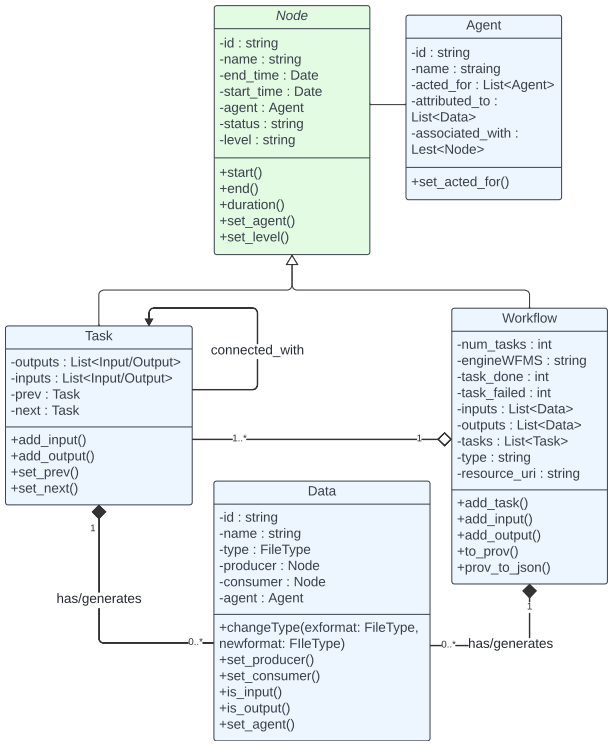


Fig. 1: Data model of the yProv4WFs library.

The UML diagram models four different concepts. The element `Node` represents the nodes of the provenance graph. Directly connected to the class `Node`, `Task` and `Workflow` inherit from the parent class both attributes, methods and in addition present some additional information. A `Task` represents the execution of a single step, it is equipped with a list of inputs, a list of outputs and a link to previous and next task. A `Workflow` has one or more tasks, it provides various action to performs on inputs/outputs lists and has the role of orchestrating the succession of the various steps. `Data` represents the resource used or produced by a node, so it acts both as *input* or *output*. Finally, `Agent` referes to who launched the experiment. The yProv4WFS library has been designed to integrate different WfMSs; at the moment it supports Streamflow [17] and Cylc [18].

C. The yProv4ML library

The rapid development of machine learning (ML) has created a need for robust and reliable methods to track and manage the provenance of ML workflows. In particular, it is no longer sufficient to merely track the execution of the training process as a whole, since numerous internal steps are conducted and metrics are generated accordingly. The collection of provenance data would support the reproducibility of all tasks that impact the model's performance as well as better interpretability of the AI-process. Despite its importance in supporting reproducibility, accountability and transparency, current ML tracking solutions exhibit several significant gaps. The first crucial challenge is posed by scalability, as ML experiments can grow in complexity and scale very rapidly, and existing tracking systems may struggle to handle the increased volume and granularity of data. Managing ML experiments in a lightweight manner is critical to avoid performance bottlenecks and enable large-scale processes. The second real challenge comes from interoperability; while many ML frameworks have been widely adopted, the vast majority generate provenance information in proprietary formats, leading to interoperability challenges. This lack of standardization makes it difficult to aggregate, compare and share provenance data across platforms and tools, limiting the usefulness of provenance information in collaborative and cross-disciplinary research.

The development of yProv4ML[6] is intended to address these critical gaps by providing a robust, user-friendly, and scalable solution. The library has been designed with the objective of efficiently handling large-scale ML experiments, by saving information in a clustered format. Its scalable architecture ensures that provenance information can be managed effectively, regardless of the complexity or size of the machine learning project. Since the library saves provenance data in the JSON format, the interoperability between different ML tools and platforms is enhanced, as user are allowed to choose their preferred data analysis method. This facilitates the sharing and comparison of provenance information across diverse research groups and projects, which in turn fosters collaboration and advances the state of the art in ML research. Furthermore, yProv4ML was developed in accordance with the specifications set forth by MLFlow [19], which allows for seamless integration of provenance tracking into existing workflows. This library separates the collected information

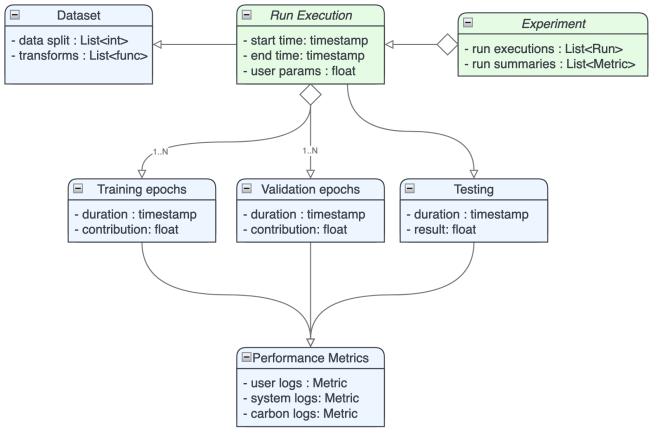


Fig. 2: High-level data model of the Prov4ML library.

into three modalities: *artefacts*, *parameters* and *metrics*. The first refers to any kind of output from the program in the form of files. These types are registered by paths in the final JSON file so that they can be retrieved. Parameters, on the other hand, are one-off values that can be logged with a label for easy retrieval. Finally, metrics are time series information that are

[6]https://github.com/HPCI-Lab/yProvML

stored along with the timestamp in which the data was logged and a user-selected step indicator. The time step, while being a user defined parameter, in a ML experiment is often assumed to be either the current epoch or the index of the current train step. As shown in Fig. 2, yProv4ML distinguishes between experiment execution and run execution, where the latter is a subset of the former, and multiple runs of the same experiment can be executed with small changes in, for example, model configuration, hyperparameters, or computing devices. The Experiment entity of the ML data-model represents a single `Task` in the generic yprov4WFs data model in Fig. 1.

yProv4ML offers a set of function calls that enable users to track metrics and parameters that are useful for the subsequent analysis of the training process and embed them within interoperable provenance documents. Examples of metrics include CPU and GPU load, power consumption, memory usage as well as others more AI-oriented related to the training, validation and testing phases. Carbon footprint metrics are also available, but for page limit constraints they will be discussed in a future work.

D. yProv Explorer

The yProv explorer is a data science application that allows scientists to access, analyze, explore and navigate provenance documents. Due to page limit constraints and considering the scope of the paper, this software component will be described in detail in a future work.

IV. Science cases

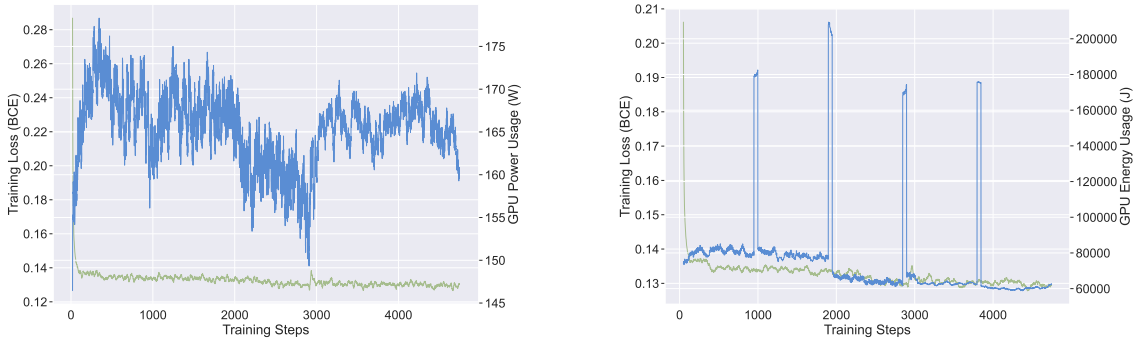
This sections presents three data-driven scientific cases, in Climate Science, High Energy Physics and Earth Observation. For each of them, the Science case workflow is described (level-1) with an in-depth analysis (level-2) of provenance information related to the specific AI model training tasks.

Although the provenance graphs have been produced for all the scientific cases, for page limit constraints they will be presented and discussed in a future work.

A. AI-based pipeline for Climate Extremes

Tropical Cyclones (TCs) are between the most destructive extreme events and their strength is boosted by global warming. The benefits of detecting their potential presence are manifold: (i) general human safety, where accurate predictions allows for timely evacuations and can minimize casualties and (ii) the deterioration of the environment, which can for example be mitigated by the containment of oil spills and other hazards during the cyclones' season. This TC detection use-case is developed within the context of the interTwin European Project[7] and builds on top of a background work from CMCC [20], with several changes to migrate from a CNN towards a GNN-based deep learning pipeline. Indeed, the most relevant change is about processing a dataset composed by 40x40 geographic grids, meant to be the input of a CNN, with a model belonging to a GNN class of architectures. Since TCs move in the atmosphere, variables like wind gust, pressure, and vorticity are fundamental to detect them, and the six features used for this process come from the ERA5 reanalysis dataset, hosted on Copernicus. Based on this data, the neural model can predict the position of tropical cyclones, when trained supervisedly on annotated samples coming from the International Best Track Archive for Climate Stewardship (IBTrACS). This collection provides the historical observations of tropical cyclones from multiple agencies, and it is being used as ground truth for training. This use case is directly linked to the European Open Science Cloud (EOSC), by means of the ENES Data Space [21] (ENES DS) which has been used as development platform. Concerning the creation of the image dataset, the initial code was re-adapted to use the multiprocessing library (instead of MPI) that comes natively with Python on the ENES DS. The output of this step is a Zarr dataset already divided into train, validation and test sets. In this dataset, each sample contains a timestamp, and the six atmospheric ERA5 features stacked together with a seventh one representing the density map of the eye of the cyclone, from IBTrACS. The density map is used as ground truth to guide the neural model during the supervised training. Starting from the image dataset, the training phase was performed and each image was converted into a graph data structure compatible with PyTorch Geometric. That worked by converting each pixel into a node connected to its adjacent neighbors with an edge. Taking the graph dataset as input, a Graph U-Net [22] was trained, to learn the patterns associated to tropical cyclones. This core AI module is subject to retraining over time, according to the availability of new data. After the training phase, the user can run inference either on the test set or on new CMIP simulations, measuring its performance in terms of accuracy, precision, etc. The considerable number of steps involved in this pipeline process makes it essential to collect both provenance and training information for a more profound comprehension of the system's behaviour w.r.t. the overall pipeline as well as the specific AI training task. The provenance tracking process is performed throughout the entire pipeline. Fig. 3a illustrates the progression of training loss (BCE) and GPU power consumption (W) throughout the training process. Initially, both training loss and GPU power consumption demonstrate considerable fluctuations. The training loss rapidly decreases during the initial stages, stabilising at around 0.14 after approximately 1000 training steps. Subsequently, it remains relatively constant with minor oscillations. The GPU power usage also exhibits an initial peak, followed by a gradual decline, although with larger fluctuations. In contrast, Fig. 3b illustrates a complex and oscillating pattern in GPU energy usage throughout the training process. Two phenomena are evident: firstly, the peaks observed in conjunction with validation steps, which are likely attributable to the switch between the training and validation datasets. Secondly, there appears to be a trend of decreasing GPU energy consumption with each epoch. Such results may be attributed to the caching of batches and a reduction in data movement. Embedding such metrics into the provenance documents associated to the TC workflow has resulted to be extremely valuable for

[7]https://www.intertwin.eu/



(a) Train loss in relation to GPU power used.

(b) Train loss in relation to GPU energy consumed.

Fig. 3: Relationship between training loss and energy-related metrics: GPU power and energy.

climate scientists in terms of multi-level pipeline and outcomes documentation.

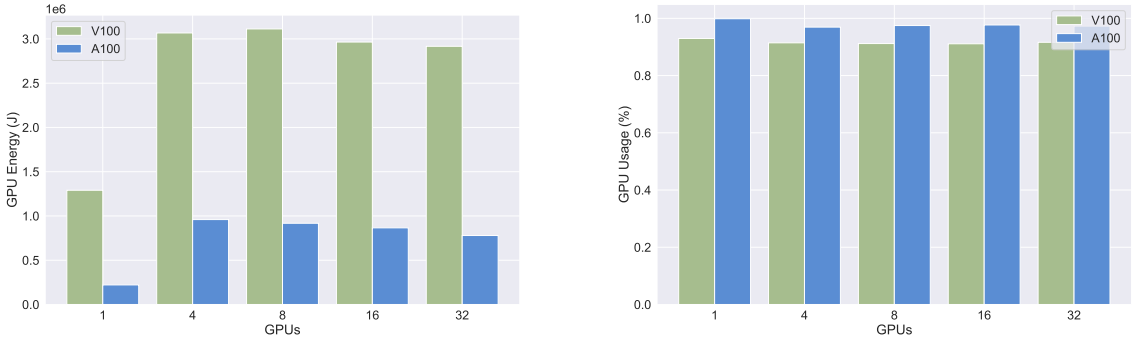
B. Fast Particle Detector Simulation with GAN

In High Energy Physics (HEP), simulating particle detectors is essential for analyzing collisions at the Large Hadron Collider (LHC). Calorimeters play a key role by measuring particle energy [23]. Traditionally, Monte Carlo (MC) techniques are used for these simulations, but they are resource-intensive [24]. To improve efficiency, the HEP community is exploring deep learning methods, including generative models that bypass step-by-step simulations [23], [25]–[27]. In particular, Generative Adversarial Networks (GANs), can directly generate realistic detector outputs [28] by modeling the observations' distribution.

CERN is leading a research project using a 3DGAN for fast particle detector simulations, treating calorimeters like 3D cameras to produce high-fidelity images of particle showers using 3D convolutions. This method improves simulation speed and accuracy [29]. The 3DGAN model is trained on data generated by MC-based simulations of specified detector's set up [30], using Geant4 (G4) toolkit [31]. Training data has been produced by a detector specific G4 application, that simulated the particle interactions. Network performance is evaluated using a loss function, consisting of the weighted sum of individual losses concerning the networks' outputs and domain-related constraints. Integrating the provenance tracking through yProv4ML allows recording all experimental steps and configurations, plus real-time metrics. This enables researchers to extract insights, track parameter impacts, and support reproducibility. Researchers can track training data versions, preprocessing steps, hyperparameters, and model checkpoints to understand how specific choices affect model's performance. The yProv4ML was integrated into *itwinai*[8], a Python library developed by CERN as part of the European Horizon Europe interTwin project, which is designed to streamline the scalability of scientific ML workflows to large-scale HPC resources. In the HEP use case presented in this section, yProv4ML is accessed through itwinai, enhancing its functionality by enabling the storage and management of provenance information for ML workflows in scientific applications. The primary objective is to gather and analyze this provenance information to understand how the number of workers used in distributed ML and different HPC infrastructures affect the efficiency and outcomes of ML training at scale. Experiments were run on two separate EuroHPC centres: Vega and Julich Supercomputer (JSC). Vega HPC is located in IZUM, Maribor, Slovenia, and is composed of a GPU partition that includes 60 nodes, each with 4 Nvidia A100 GPUs, 2 AMD Rome 7H12 CPUs, and 512 GB of memory. On JSC the experiments were run on the HDFML system, which includes 16 nodes, each with 4 Nvidia V100 GPUs, 48 Intel(R) Xeon(R) Gold 6126, and 180 GB of memory. To collect comparable results, the experiments were run on both HPC centres implementing a similar resource allocation through SLURM. Considering the goal to perform an in-depth study of the ML training task at scale, the allocation was 1 to eight 8 on the GPU partitions with 1 to 4 GPUs per node, 4 CPUs per node, and 10 GB of memory per worker. The experiments performed in this use case included 5 configurations depending on the number of GPUs used, ranging from 1, 4, 8, 16 to 32. Fig. 4 illustrates a comparison of the energy consumption of GPUs between the V100 and A100 models across a range of device counts. GPU power consumption is calculated as the average of the power consumption values recorded throughout an ML training run. In scenarios with multiple workers, this average is also computed across all workers in use. Any missing data points are replaced with the overall average. The total energy consumption is then determined by multiplying the average power consumption by the number of workers and the total training time. The GPU power consumption plot in Fig. 4a shows a linear increase in average power usage as the number of workers rises from 1 to 4, likely due to the increased intra-node collective communication managed by NCCL[9]. However, when the number of workers is a multiple of 4, GPU power consumption stabilizes, as all four GPUs per node are fully utilized, resulting in a consistent power footprint for intra-node communication. On the other hand, in runs with

[8]https://github.com/interTwin-eu/itwinai

[9]https://docs.nvidia.com/deeplearning/nccl/index.html



(a) GPU energy for different HW configurations.

(b) GPU load for different HW configurations.

Fig. 4: Impact of different HW on performance and efficiency. GPU energy consumption in Joules (left). GPU load as a percentage of total capacity (right).

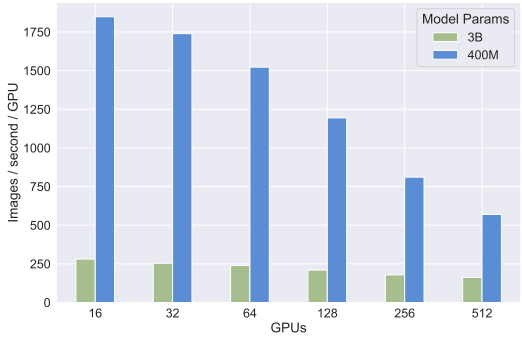
number of workers above 4, both the V100 and A100 clusters exhibited a notable decline in worker energy consumption with an increase in the number of GPUs. This is due to separation of tasks, resulting in lower execution times. When summing over all processes however, the total energy consumed by the run remains mostly constant. Comparing different hardware, V100 consistently demonstrated a higher energy consumption than the A100 clusters across all GPU counts. Fig. 4b illustrates the load for V100 and A100 GPUs. A100 GPUs consistently exhibit a higher GPU load across all configurations compared to the former group. GPU utilization and memory occupation are influenced by the task's computational complexity and the minibatch size, respectively. Since GPUs are partitioned among workers, their utilization is only loosely correlated with the number of workers involved in the distributed ML training. Overall, A100 GPUs are more energy efficient and maintain a higher and more consistent utilisation rate than V100 GPUs across different hardware configurations. As a result of the more powerful chips, A100 GPUs may be more suitable for large-scale deployments where energy efficiency and optimal resource utilisation are critical factors.

C. Vision Transformer Weak Scaling

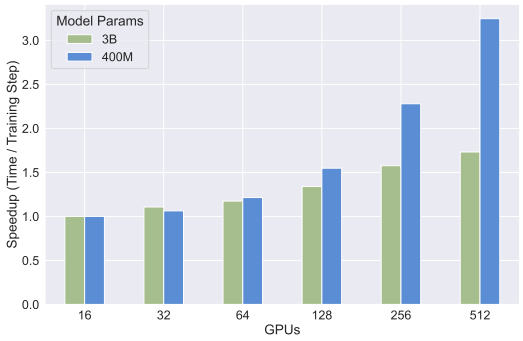
The Earth and Environmental Sciences (EES) research community has historically benefited from a vast treasure of data from a variety of sources, including earth system models, in-situ measurements and earth observations from satellites. The meteorological data archive at the European Center for Medium Range Weather Forecasts (ECMWF) contains over 500 PB of data with over 500 billion meteorological fields. Similarly, the data volume at the NASA Earth Science Data System is expected to grow to 600 PB by 2030. The research community is facing the grand challenge of harnessing the power of AI to harvest the rich information from exabytes of data using extreme scale computing toward addressing the societal needs to understand and mitigate the effects of climate change and to foster resiliency.

AI Foundation Models (FM) have rapidly gained adoption in the scientific research community. FMs are "models trained on broad data that can be adapted to a wide range of downstream tasks" [32]. Their powerful performance can be attributed to *scale*, a property emanating from large volumes of data trained with vast amounts of compute power. They are often, though not necessarily, based on the transformer deep learning architecture with self-attention mechanism that excels in next-token prediction tasks, such as autoregressive output from Large Language Models (LLM). Vision Transformers (ViT) [33], based on LLM architectures, have been adapted to train tokenized images and image-like data for GeoAI applications involving segmentation, classification, and regression tasks [34]. One of the most successful and beneficial series of earth observations has been from NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) instrument onboard the Terra and Aqua satellites. We use the MODIS products to develop a foundation model (MODIS-FM), based on a ViT backbone, for atmospheric remote sensing applications toward understanding of atmospheric, oceanic, and land characteristics, such as clouds, aerosols, water vapor, ocean color, etc. We are developing MODIS-FM using the Frontier supercomputer at the Oak Ridge Leadership Computing Facility. Frontier consists of 9402 compute nodes, each with a single 64-core AMD EPYC CPU and 8 AMD Instinct MI250X GCDs, considered a GPU for all practical purposes. For pretraining and validating the MODIS-FM the MODIS L1B calibrated radiances [35] and the Atmosphere L2 Cloud Product [36] were used. The data acquisition workflow was implemented using funcX [37] to automate the download on multi-platforms. Every image comprising 36 spectral channels covers an approximate area of 2330 km $\times$ 2030 km along the ground track. In all nearly 2 million daytime MODIS images from both Aqua and Terra satellites have been acquired via the funcX workflow. After the acquisition of the images, a Parsl [38] workflow was invoked to subsample the images to tiles of size 128×128.

The collection of provenance data is critical in the context of a weak scaling experiment, as it ensures the reproducibility and interpretability of the results. Understanding how variations in input, configurations, and resource allocations impact performance is essential, particularly in this context. Provenance offers insights into these variables, enabling researchers to



(a) Throughput for one process (in images per second) for SwinT-V2 model.



(b) Speedup for SwinT-V2 model.

Fig. 5: Scaling of the SwinTransformerV2 model across varying GPU counts. Figure (a) depicts the images per second processed per GPU. Figure (b) illustrates the speedup achieved, demonstrating the increase in training efficiency with more GPUs.

identify and account for anomalies or deviations from expected scaling behaviour. Furthermore, it facilitates the reproduction of the experiment under similar or different conditions, enhancing the validity of the conclusions drawn.

The scaling experiments utilise the Swin Transformer V2 (SwinT-V2) [39] model, which processes images at varying scales through a series of stages, and incorporates key improvements over its predecessor, making it more efficient and scalable for handling high-resolution images and large datasets. A series of experiments were conducted with model configurations of 400 million (400M) and 3 billion (3B) parameters, employing a Fully Sharded Data Parallel (FSDP) distributed training strategy. Runs were conducted with 2, 4, 8, 16, 32 and 64 Frontier nodes, where each node provides access to eight GPUs respectively. These runs are quick executions with 15 training steps each. To ensure robustness of results, each run was also repeated 5 times, and data plotted after being averaged. To validate the training process, one additional run is executed for 100 alternated training and validation steps, for the same model sizes but only on 8 and 64 nodes, due to time constraints. To calculate the throughput, as well as the speedup for each train step, the time taken for a sample to pass through the model is recorded, excluding all logging calls and including only optimizer setup, sample feed-forward, loss and gradient calculation. Fig. 4a shows weak scaling results, in terms of images per second per GPU, for the SwinT-V2 model. This weak scaling metric exhibits a decline with an increase in the number of GPUs for both model sizes. This behavior can be attributed to the communication overhead due to intra and inter-node operations. Fig. 4b, on the other hand, illustrates the speedup achieved for the SwinT-V2 model, calculated as the time required for each training steps. The speedup for both the 3B and 400M parameter models increases with the number of GPUs, indicating that the use of multiple GPUs reduces the training time per step with a corresponding increase in the global batch size. While the 400M parameter model demonstrates a pronounced increase in speed, evident at higher GPU counts, the 3B parameter model exhibits enhanced

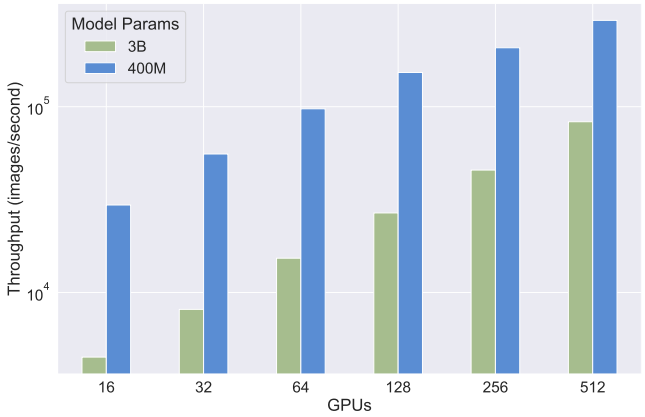


Fig. 6: Throughput for SwinT-V2 model.

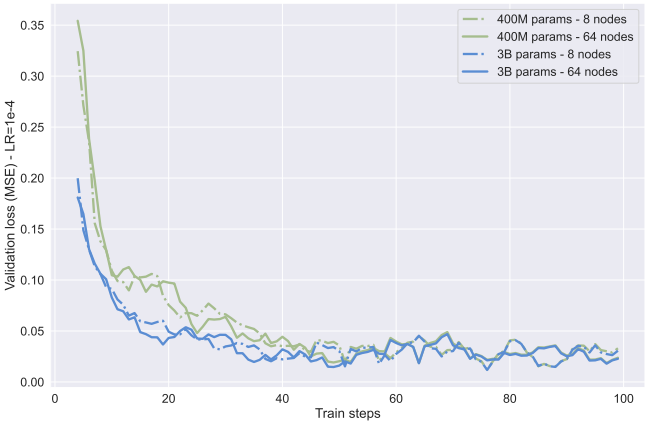


Fig. 7: Validation loss comparison for different configurations.

performance with increased GPUs, albeit to a lesser extent. It seems reasonable to assume that the edge gained by the 400M parameter model is due to its smaller size. Fig. 6 shows the throughput performance aggregated over the total number of

GPUs used, measured in images per second, of the SwinT-V2 model for the two model sizes. As expected, the throughput increases with the number of GPUs for both model sizes, with the 400M parameter version consistently outperforming its counterpart due to its smaller size and faster feed-forward phase. Fig. 7 illustrates the comparison of validation loss, measured in Mean Squared Error (MSE), over 100 training steps. The loss decreases during the validation phase since the training process is alternated with the latter one, meaning one sample of training is always followed by a validation one. Initially, all configurations exhibit a steep decline in loss, indicating rapid learning. The runs with 64 nodes, both for 400M and 3B parameters, show a slightly faster initial decrease in loss compared to their 8-node counterparts.

V. Conclusions and Future Work

This study highlights the importance of collecting provenance metadata in complex, data-driven scientific experiments in various domains. Provenance information not only facilitates the reproducibility of experiments, but also helps to diagnose performance bottlenecks and understand the interplay between model complexity and computational resources. Furthermore, the reliability and integrity of the results is ensured, which is crucial for advancing the field of large-scale ML. Future work will include support for new WfMSs, new metrics in the AI training processes, and additional exploration and analytics capabilities for scientific users.

Acknowledgment

This work was partially funded by the EU HE interTwin project (GA 101058386) and the EU HE Climateurope2 project (GA 101056933). Moreover, this work was partially funded under the NRRP, Mission 4 Component 2 Investment 1.4, by the European Union – NextGenerationEU (proj. nr. CN_00000013). This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] S. B. Davidson and J. Freire. 2008. Provenance and scientific workflows: challenges and opportunities. ACM SIGMOD 2008, USA, 1345–1350.
[2] M. Herschel et al. A survey on provenance: What for? What form? What from? The VLDB Journal 26, 6 (2017), 881–906.
[3] S. M. S. d. Cruz, et al., "Provenance Services for Distributed Workflows," CCGRID2008, Lyon, France, pp. 526-533.
[4] F. Z. Khan et al., Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv, GigaScience, Volume 8, Issue 11, Nov. 2019.
[5] Belhajjame K. et al. Using a suite of ontologies for preserving workflow-centric research objects. J WebSemantics 2015;32:16–42,10.1016/j.websem.2015.01.003.
[6] Kunze JA. et al. The BagIt File Packaging Format (V1.0). Request for Comments RFC8493. RFCEditor, 2018, 10.17487/RFC8493.
[7] Sporny M. et al. JSON-LD 1.0. W3C Recommendation 16 January 2014. http://www.w3.org/TR/2014/REC-json-ld-20140116/
[8] Kaushik G. et al. Rabix: an open-source workflow executor supporting recomputability and interoperability of workflow descriptions. Pac Symp Biocomput 2017;22:154–65, 10.1142/97898132078130016
[9] Voss K et al., Full-stack genomics pipelining with GATK4 + WDL + Cromwell. F1000Res 2017,10.7490/f1000research.1114634.1
[10] Guimera RV. bcbio-nextgen: automated, distributed next-gen sequencing pipeline. EMBnet J 2012;17(B):30, 10.14806/ej.17.B.286.
[11] M. R. Crusoe et al., The CWL Community. 2022. Methods Included: Standardizing Computational Reuse and Portability with the Common Workflow Language. Commun. ACM 65, 6 (June 2022), 54–63.
[12] S. Leo, et al. Recording provenance of workflow runs with RO-Crate, 2023, arXiv:2312.07852 https://doi.org/10.48550/arXiv.2312.07852
[13] Guha RV, Brickley D, Macbeth S. Schema.org: Evolution of Structured Data on the Web: Big data makes common schemas even more necessary. Queue 2015;13(9):10–37. doi: doi:10.1145/2857274.2857276
[14] Souza, Renan, et al. "Workflow provenance in the lifecycle of scientific machine learning." Concurrency and Computation: Practice and Experience 34.14 (2022): e6544.
[15] Cao, Yang, et al. "ProvONE: extending PROV to support the DataONE scientific community." PROV: Three Years Later (2016).
[16] S. Fiore et al., "A Graph Data Model-based Micro-Provenance Approach for Multi-level Provenance Exploration in End-to-End Climate Workflows," IEEE BigData2023, Sorrento, Italy, 2023, pp. 3332-3339.
[17] I. Colonnelli et al., "StreamFlow: cross-breeding cloud with HPC", IEEE Trans. on Emerging Topics in Computing, vol. 9, issue 4, pp.1723-1737.
[18] Oliver et al., (2018). Cylc: A Workflow Engine for Cycling Systems. JOSS, 3(27), 737, doi: https://doi.org/10.21105/joss.00737
[19] Zaharia, M. et al. (2018). Accelerating the machine learning lifecycle with MLflow. IEEE Data Eng. Bull., 41(4), 39-45.
[20] Accarino, G. et al., "An ensemble machine learning approach for tropical cyclone localization and tracking from ERA5 reanalysis data", Earth and Space Science, vol. 10, no. 11, Nov. 2023.
[21] Elia, D. et al., A Data Space for Climate Science in the European Open Science Cloud. *Computing In Science & Engineering*. 25, 7-15 (2023).
[22] Gao, Hongyang and Ji, Shuiwang, "Graph u-nets", International Conference on Machine Learning, 2019.
[23] HEP Software Foundation, J. Apostolakis et al., HEP Software Foundation Community White Paper WG - Detector Simulation, 2018, URL: https://arxiv.org/abs/1803.04165
[24] CERN, "G4FastSim", [Online]. Available: https://g4fastsim.web.cern.ch/
[25] W. Lukas, "Fast Simulation for ATLAS: Atlfast-II and ISF", Journal of Physics: Conference Series, vol. 396, no. 2, pp. 022031, Dec. 2012. DOI: https://dx.doi.org/10.1088/1742-6596/396/2/022031
[26] D. Orbaker (on behalf of the CMS collaboration), "Fast simulation of the CMS detector", J. of Physics: Conference Series, vol. 219, no. 3, pp. 032053, 2010. DOI: https://dx.doi.org/10.1088/1742-6596/219/3/032053
[27] E. Barberio, et al., "Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers", J. of Physics: Conference Series, vol. 160, no. 1, pp. 012082, Apr. 2009. DOI: https://dx.doi.org/10.1088/1742-6596/160/1/012082
[28] Ian J. et al., "Generative Adversarial Networks", 2014. arXiv: https://arxiv.org/abs/1406.2661
[29] G. R. Khattak et al., "Fast simulation of a high granularity calorimeter by generative adversarial networks", The European Physical Journal C, vol. 82, no. 4, pp. 386, Apr. 2022.
[30] M. Pierini, M. Zhang, "CLIC Calorimeter 3D images: Electron showers at Random Angle", 2020. DOI: https://doi.org/10.5281/zenodo.3603086
[31] Geant4 Collaboration, "Geant4: A toolkit for the simulation of the passage of particles through matter", https://geant4.web.cern.ch/
[32] Bommasani, Rishi et al., "On the opportunities and risks of foundation models." ArXiv Preprint. arXiv:2108.07258. 2021.
[33] Dosovitskiy, A., et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).
[34] Jakubik, Johannes, et al., "Foundation Models for Generalist Geospatial Artificial Intelligence." arXiv:2310.18660. 2023.
[35] MODIS Characterization Support Team: MODIS 1km calibrated radiances product. NASA MODIS adaptive processing system. Goddard Space Flight Center, USA, 10.5067/MODIS/MOD021KM.061.
[36] Platnick, Steven, et al. "The MODIS cloud optical and microphysical products: Collection 6 updates and examples from Terra and Aqua." IEEE Trans. on Geoscience and Remote Sensing 55.1 (2016): 502-525.
[37] Chard, Ryan, et al. "Funcx: A federated function serving fabric for science." Proc. of the 29th International symposium on HPDC. 2020.
[38] Babuji, Yadu, et al. "Parsl: Pervasive parallel programming in python." Proceedings of the 28th International Symposium on HPDC. 2019.
[39] Liu, Ze, et al. "Swin transformer v2: Scaling up capacity and resolution." Proceedings of the IEEE/CVF conference on CVPR. 2022.