Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 240 (2024) 3–12

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the First EuroHPC user day

Cross-Facility Federated Learning

Iacopo Colonnelli[a,*], Robert Birke[a], Giulio Malenza[a], Gianluca Mittone[a], Alberto Mulone[a], Jeroen Galjaard[b], Lydia Y. Chen[b], Sanzio Bassini[c], Gabriella Scipione[c], Jan Martinovič[d], Vit Vondrák[d], Marco Aldinucci[a]

[a] *University of Torino, Computer Science Dept., Corso Svizzera 185, 10149, Torino, Italy*
[b] *Delft University of Technology, Computer Science Dept., Van Mourik Broekmanweg 6, 2628 XE Delft, Netherlands*
[c] *CINECA National Supercomputing Center, Casalecchio di Reno, Bologna, Italy*
[d] *IT4Innovations, VSB – Technical University of Ostrava, Ostrava, Czech Republic*

Abstract

In a decade, AI frontier research transitioned from the researcher's workstation to thousands of high-end hardware-accelerated compute nodes. This rapid evolution shows no signs of slowing down in the foreseeable future. While top cloud providers may be able to keep pace with this growth rate, obtaining and efficiently exploiting computing resources at that scale is a daunting challenge for universities and SMEs. This work introduces the Cross-Facility Federated Learning (XFFL) framework to bridge this compute divide, extending the opportunity to efficiently exploit multiple independent data centres for extreme-scale deep learning tasks to data scientists and domain experts. XFFL relies on hybrid workflow abstractions to decouple tasks from environment-specific technicalities, reducing complexity and enhancing reusability. In addition, Federated Learning (FL) algorithms eliminate the need to move large amounts of data between different facilities, reducing time-to-solution and preserving data privacy. The XFFL approach is empirically evaluated by training a full LLaMAv2 7B instance on two facilities of the EuroHPC JU, showing how the increased computing power completely compensates for the additional overhead introduced by two data centres.

Keywords: Federated Learning; High-Performance Computing; Cross-Facility Computing; Hybrid Workflows; StreamFlow.

1. Introduction

The Big Data era is propelling machine learning experiments to unprecedented scales, outpacing even Moore's law [1]. In a decade, AI frontier research has transitioned from the researcher's workstation (AlexNet [2], 2012, trained for six days on two NVIDIA GTX580 3GB) to thousands of high-end specialised chips (PaLM [3], 2022, trained for 50 days on 6144 TPU v4 chips). The advent of foundation models [4] and physics-informed neural networks [5] has

* Corresponding author.
E-mail address: iacopo.colonnelli@unito.it

1877-0509
10.1016/j.procs.2024.07.003

catapulted Deep Neural Networks (DNNs) to trillions of parameters, requiring an entire exascale data centre to be trained from scratch [6]. This rapid evolution shows no signs of slowing down in the foreseeable future.

While top cloud providers may be able to keep pace with this growth rate [7], for universities and SMEs, the task of obtaining and efficiently exploiting computing resources at that scale is a daunting challenge. Indeed, in the last few years, this dichotomy of industry and academia has caused a reduced representation of academic-only research teams in computing-intensive research topics, especially foundation models [8].

In this scenario, HPC federations such as the one currently promoted by the EuroHPC Joint Undertaking play a crucial role. The massive amount of computing power at their disposal can compete with top-notch private data centres, potentially bridging the gap between industry and academia. However, granting researchers access to public facilities and enabling them to efficiently exploit their computing power efficiently is not trivial. Allocating an entire data centre exclusively to a single experiment for an extended period violates the principle of fair resource usage. On the other hand, efficiently exploiting multiple data centres without high-level techniques and tools to design and orchestrate cross-facility workloads is almost exclusively the prerequisite of senior computer scientists with a deep knowledge of high-performance distributed systems.

This work introduces the Cross-Facility Federated Learning (XFFL) framework to extend the opportunity to efficiently exploit multiple independent data centres for extreme-scale DNN training to data scientists and domain experts. In particular, XFFL combines two different approaches. On the one hand, hybrid workflow abstractions allow users to quickly design and orchestrate cross-facility workloads, decoupling tasks from environment-specific technical details to reduce complexity and increase reusability. On the other hand, Federated Learning (FL) algorithms eliminate the need to move large amounts of data between different facilities, reducing time-to-solution and preserving data privacy. To the best of the authors' knowledge, this work describes the first attempt to run a large-scale XFFL experiment.

The benefits brought by the XFFL approach are evaluated on a realistic scenario, i.e., training the full LLaMAv2 7B Large Language Model (LLM) [9] on top of two facilities of the EuroHPC JU, the Leonardo@CINECA and the Karolina@IT4I clusters, which occupy the 6th and 113th positions in the November 2023 Top500 ranking [10], respectively. In particular, the evaluation aims to assess whether the speedup obtained from the increased computing power is enough to compensate for the additional overhead introduced by using two distinct facilities, e.g., the model transfer time across the Internet or the misaligned execution times due to job queues. The results are promising, paving the way for further experiments with larger models and wider federations of data centres.

This article is organised as follows. Sec 2 introduces the preliminary concepts and analyses the related work. Sec. 3 introduces the proposed XFFL methodology. Sec. 4 describes the experimental environment and discusses the obtained results. Finally, Sec. 5 concludes the article and outlines future research directions.

2. Background and related work

2.1. Hybrid workflows

A hybrid workflow allows its steps to span multiple, heterogeneous, and independent computing infrastructures [11]. Each of these aspects has significant implications. Support for multiple infrastructures implies that each step can target a different execution environment. Such environments can be heterogeneous, with different and incompatible protocols for authentication, communication, resource allocation and job execution. They can also be independent of each other without the possibility of establishing direct connections to transfer data. A suitable model for hybrid workflows must then be composite, enclosing a specification of the workflow dependencies, a topology of the involved locations, and a mapping relation between steps and locations.

Hybrid Workflow Management Systems (WMSs) [12, 13, 14, 15, 16] are the software tools in charge of orchestrating hybrid workflow executions on top of complex, large-scale infrastructures, such as cross-facility and cloud-HPC environments. All the complexities of heterogeneous systems coordination, job scheduling, and data transfers are hidden behind a high-level workflow abstraction, which decouples the business code from the coordination layer. Plus, hybrid WMSs provide advanced features out of the box, such as caching, provenance tracking, and fault tolerance. Hybrid WMSs have already proved their superior flexibility and performance in diverse scientific domains: bioinformatics [17, 18], astrophysics [19], environmental science [20], and many more.

This work relies on the StreamFlow WMS [13] to orchestrate a large-scale XFFL workload, building on the initial work described in [21]. StreamFlow is a container-native hybrid WMS based on the Common Workflow Language (CWL) open standard [22]. It has been designed around two primary principles. First, it allows executing tasks requiring complex combinations of software container environments, supporting modern micro-services applications. Second, it relaxes the requirement of a single shared data space, allowing hybrid workflow executions on top of multi-cloud, multi-HPC, and mixed cloud-HPC infrastructures. Since this work requires the orchestration of a fully containerised FL workload on top of two independent HPC facilities, it represents a perfect use case for StreamFlow.

2.2. Federated Learning

Federated learning (FL) is a collaborative, distributed approach enabling the solving of a common Machine Learning (ML) problem shared between multiple institutions [23]. The core idea behind FL is sharing locally trained models instead of the local data itself. This simple yet powerful idea allows for overcoming the ML de facto standard data lake approach, in which data is harvested across different sources and accumulated by a single institution. Such a gathering process involves agreement efforts between the data-holding institutions while exposing the data to privacy attacks and misuse. FL offers a natively privacy-preserving approach, allowing multiple parties to train a DNN on their local data collaboratively and merge them into a global, shared DNN, retaining the knowledge extracted from all peers.

FL deployments involve a set of data-holding clients coordinated by a central server, even if other structures are possible [24]. The training process starts with the server communicating an initial set of parameters for the shared DNN model to each client. Each client then trains the received DNN on the local data and returns the obtained parameters to the central server. Once all (or a sufficient number) of local models are accumulated, the central server aggregates them, obtaining a global model. Such a model is then broadcast back to the clients, and the process continues iteratively until convergence. The presented process is just a general design of the FL approach, excluding many details such as which aggregation strategy is used [25, 26], how to make the execution more asynchronous [27, 28], how to speed up the client-server model communication [29, 30, 31], and more [32]. FL has been successfully applied to a variety of models ranging from deep, e.g. CNNs [23] and transformers [33], to classic [34] ML models.

FL can also be seen as a device that further increases the scalability of a distributed ML system. FL allows multiple computing infrastructures to train copies of the same model on different data partitions simultaneously, similar to a distributed data-parallel approach. Nevertheless, differently from other data-parallel strategies [35, 36], data are never shuffled among worker nodes, heavily reducing the communication overhead. Note that FL is not computationally equivalent to a centralised data lake scenario. The federation's number of clients, local and global data distribution, aggregation strategy, and many other hyperparameters heavily influence the final global model performance [37]. FL thus offers a novel trade-off between speeding up ML training and the obtained model performance; however, commercial FL frameworks [38, 39, 40, 41] do not seem to consider this aspect, preferring to focus on its privacy-preserving features rather than supporting complex, large-scale deployments.

FL of billion- and trillion-parameter foundation models is a promising approach to improve time-to-solution [42]. Indeed, FL reduces per-site memory occupancy and computing time and minimises inter-site communication compared to standard distributed training [43]. There are already few attempts to fine-tune foundation models in scientific literature [44, 45]. However, to the best of the authors' knowledge, this work describes the first attempt to train a full LLM on multiple HPC facilities through FL.

3. Cross-facility Federated Learning

By enabling researchers to harness the joint computing power of multiple clusters, we can mitigate the compute divide mentioned in Sec. 1. If a large, complex workload can be split into multiple independent sub-computations, they can run concurrently on different facilities. The partial results obtained on each machine can then be combined, leading to the final outcome of the global computation. In the case of iterative processes, these global results can be returned to the clusters, initiating a new iteration.

However, it is crucial to acknowledge that not all computations are amenable to this approach. Cluster-to-cluster communications are orders of magnitude slower than intra-cluster, node-to-node communications. Moreover, they are subject to varying bandwidths and network speeds typical of the public Internet network, making the process slow

and error-prone. For these reasons, problems that necessitate frequent and large data transfers between clusters are not ideal candidates for cross-facility executions.

To ensure more equitable resource usage, lengthy iterative processes can be divided into multiple job submissions, each handling one or more iterations. This strategy also allows for the delegation of cross-facility orchestration to a third party, such as the control plane of a hybrid WMS. This way, a group of HPC centres can be treated as a single, geographically distributed supercluster, with all the technical aspects managed by the WMS runtime. However, it is crucial to consider queue waiting times and their variability whenever a new set of tasks is submitted to different HPC centres. Queue time is generally unpredictable and can range from minutes to hours, depending on the current cluster utilisation and the requested resources. Therefore, the computing time should be sufficiently long to offset the potential delays introduced by waiting times.

For ease of understanding, consider the well-known Bulk Synchronous Programming (BSP) model [46]. A BSP computation is structured into a sequence of supersteps, each consisting of three phases: a concurrent computation phase where each component performs local computations; a communication phase where components exchange data; and a global synchronisation phase where each component waits until all other components have reached the same barrier. When submitting a BSP computation consisting of S supersteps to solve a problem of size n as a single job to an HPC cluster with p nodes, the total time T_1 can be calculated as:

$$T_1 = q + \sum_{s=1}^{S} \left[w_s(n,p) + g \cdot h_s(n,p) \right] = q + \sum_{s=1}^{S} T_s(n,p) \tag{1}$$

where q is the waiting time, w is the computing time on p nodes, h is the amount of data to communicate among p nodes, and g is the speed of the interconnection network. Note that we omitted the synchronisation overhead, as it does not change significantly in the different scenarios discussed below. Instead, if we submit each superstep as a separate job to the HPC workload manager, each superstep is subject to queueing time, increasing the total time T_2 to:

$$T_2 = \sum_{s=1}^{S} \left[q_s + w_s(n,p) + g \cdot h_s(n,p) \right] = \sum_{s=1}^{S} q_s + \sum_{s=1}^{S} T_s(n,p) > T_1 \tag{2}$$

However, note that if the queue waiting time is much shorter than the execution time, i.e., $q_s \ll T_s(n,p)$, then $T_2 \simeq \sum_{s=1}^{S} T_s(n,p) \simeq T_1$. If now we consider F facilities having the same networking technology (i.e., $g_i = g$, for any $i \in F$) but different sizes (i.e., $p_i \neq p_j$ for any $i \neq j \in F$) totalling $\sum_{i=1}^{F} p_i$ processors, the total time T_3 can be calculated as:

$$\begin{aligned} T_3 &= \sum_{s=1}^{S} \left\{ \max_{f \in F} \left[q_{s,f} + w_{s,f}(n, \sum_{i=1}^{F} p_i) + g \cdot h_{s,f}(n, \sum_{i=1}^{F} p_i) \right] + G \cdot H_s(n, p_1, \ldots, p_F) \right\} \\ &= \sum_{s=1}^{S} \left\{ \max_{f \in F} \left[q_{s,f} + T_{s,f}(n, \sum_{i=1}^{F} p_i) \right] + G \cdot H_s(n, p_1, \ldots, p_F) \right\} \end{aligned} \tag{3}$$

where G is the speed of the cluster-to-cluster network, H is the amount of data to communicate among different facilities, and $\max_{f \in F}$ models that we need to wait for the slowest facility f to finish before the global communication phase. If the problem is scalable enough, then $w_{s,f}(n, \sum_{i=1}^{F} p_i) < w_{s,f}(n, p_f)$ for any $f \in F$. However, G can be orders of magnitude higher than g and queue times can vary significantly from cluster to cluster. Therefore, a good cross-facility approach should try to:

- minimise cross-cluster communications to reduce $H_s(n, p_1, \ldots, p_F)$;

- keep $w_{s,f}(n, \sum_{i=1}^{F} p_i)$ large enough to compensate for the overhead of $q_{s,f}$;
- balance the workload of different clusters e and f such that $T_{s,e}(n, \sum_{i=1}^{F} p_i) \simeq T_{s,f}(n, \sum_{i=1}^{F} p_i)$ for any $e, f \in F$.

If all these conditions hold, Eq. 3 can be rewritten as

$$T_3 \simeq \sum_{s=1}^{S} \left[w_s(n, \sum_{i=1}^{F} p_i) + g \cdot h_s(n, \sum_{i=1}^{F} p_i) \right] = \sum_{s=1}^{S} T_s(n, \sum_{i=1}^{F} p_i) < T_2 \tag{4}$$

The classical data-parallel distributed training of a DNN is much more complex than a pure BSP computation [35]. However, each model update can be approximated with a BSP superstep that locally computes the forward pass and back-propagation on each node, communicates the gradients to all other nodes, and globally synchronises the process to compute the resulting gradients. Unfortunately, this algorithm necessitates substantial communication among all involved facilities, leading to a communication-bound problem [43]. In contrast, the FL approach simplifies this process by only requiring the exchange and aggregation of models, thereby minimising cross-cluster data transfers (each superstep models one FL round). Each cluster trains a model on the local data, which is then exchanged to update the global model before a new FL round starts. By carefully tuning the amount of data available on each cluster, we can ensure a balanced workload among different, potentially heterogeneous facilities. The key question that remains to be answered is whether the training time of foundation models is large enough to offset the queuing times variability and the increased communication overhead in cross-cluster settings, which is the primary focus of Sec. 4.

4. Evaluation

4.1. Test case

We used XFFL to continue training LLaMAv2 [9], a foundation model for natural language processing. LLaMAv2 is a family of large language models ranging up to 70B parameters based on the transformer architecture [47] and pre-trained with up to 2T tokens. The tokens stem from a mix of text corpora selected to provide a fair mix of demographic representations along five axes: religion, gender and sex, nationality, race and ethnicity, and sexual orientation. Nevertheless, the corpora are heavily centred on the English language (89.7%), with the second most represented language (German) lagging far behind (0.17%).

Inspired by this imbalance, we fine-tuned a LLaMAv2 7B model specifically for Italian and Czech using the clean_mC4_it [48] and mC4_cs [49] datasets. The mC4_cs dataset is the plain version of the mC4 corpus Czech split, while the clean_mC4_it dataset is a filtered version of the mC4 corpus Italian split. This filtering involved removing documents containing despicable words, sentences that are too short (<3 words) or too long (>1000 characters), sentences not ending with end-of-sentence punctuation, documents which are either too short (<5 sentences or 500 characters) or too long (>50000 characters), and so on. The result is a cleaned Italian corpus of 103 million documents, comprising 41 billion words.

Due to the size of the entire corpus, which was incompatible with our experiment's time requirements, we opted to use the "tiny" split. During preprocessing, the documents have been converted into fixed-length sequences of 2048 tokens through the standard LLaMAv2 tokeniser. As a result, the Italian dataset used consisted of 10 million documents comprising 4 billion words, converted into a total of 4085342 training samples and 13252 testing samples with 2048 tokens each. We retained the same quantity of data from the mC4_cs dataset to ensure balanced execution times. Each dataset occupies about 102GB of disk space, much more than the 13GB required by the half-precision representation of the LLaMAv2 7B weights. Due to limited resource availability, the total execution time required to train LLaMAv2 7B on the whole clean_mC4_it and mC4_cs datasets had to be estimated. This is possible since the training time is linear in the training dataset size, modulo having fixed hyperparameters. Once the aggregate data processing speed (usually expressed in terms of iterations/second) is stable, estimating the total execution time becomes straightforward, knowing the number of iterations needed for the dataset (1 iteration = 1 data batch = 4 data samples = 8192 tokens in our testbed).

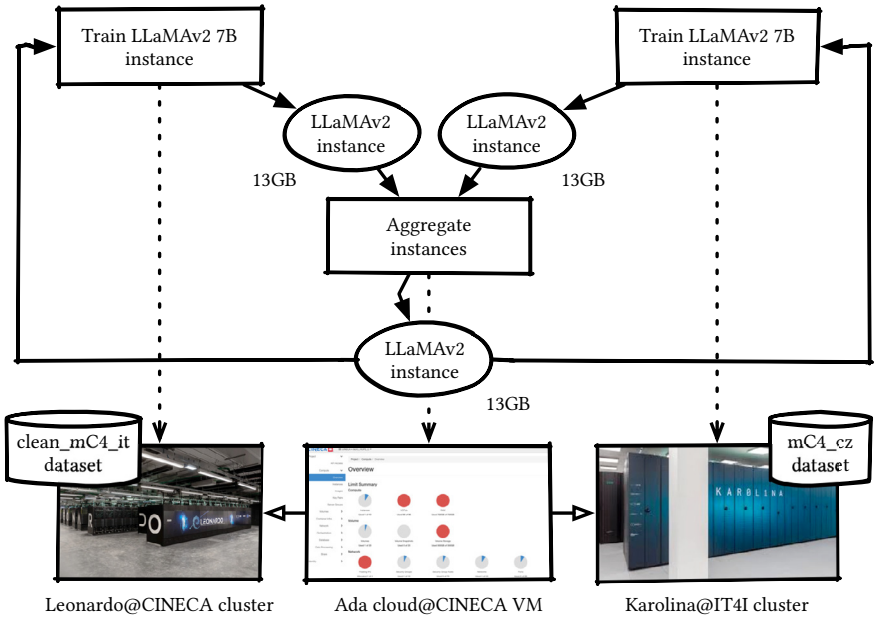

Fig. 1. Pictorial representation of the XFFL hybrid workflow. Steps are represented as rectangles, ports as circles, and dependencies as arrows with black-filled heads. Locations are represented in the lower rectangles, and communication channels between them as arrows with white-filled heads. Finally, mapping relations are expressed as dotted arrows.

4.2. Execution environment

The experiment has been performed on two HPC systems, Leonardo@CINECA and Karolina@IT4I. They occupy the 6th and 113th positions in the Top500 ranking [10] (November 2023), respectively. In particular, we trained LLaMAv2 on the reduced clean_mC4_it dataset on Leonardo@CINECA, while we relied on Karolina@IT4I to train the model on the mC4_cs data.

Leonardo is a pre-exascale system with a performance peak of 238.70 PFlop/s (LINPACK R_{max}) hosted by CINECA in Bologna, Italy. It consists of 4992 liquid-cooled compute nodes interconnected through an NVIDIA Mellanox network, with Dragon Fly+, capable of a maximum bandwidth of 200Gbit/s between each pair of nodes. Each of the (Booster) nodes used in our experiment mount a custom BullSequana X2135 "Da Vinci" blade equipped with an Intel Xeon 8358 Ice Lake CPU (32 cores, 2.6GHz), 512GB DDR4 RAM (8 x 64GB, 3200MHz), 4 NVIDIA custom A100 GPUs (64GB, HBM2e), and 2 NVIDIA InfiniBand HDR 2×100Gb/s network cards. Karolina, on the other hand, is a petascale system hosted by the IT4Innovations National Supercomputing Center at the VSB-Technical University of Ostrava, Czech Republic. Each of its GPU nodes mounts an HPE Apollo 6500 Gen10 Plus blade equipped with an AMD EPYC 7763 CPU (64 cores, 2.45GHz), 1TB DDR4 RAM (8 x 128GB, 3200MHz), 8 NVIDIA A100 GPUs (40GB, HBM2), and 4 InfiniBand HDR 200Gb/s network cards. Both facilities rely on SLURM [50] for job submission and workload management.

After each training round, the model aggregation averages the model weights through FedAVG, which has been shown effective also on transformer-type models [51]. This operation is executed on a Virtual Machine (VM) with 96 vCPUs and 720GB RAM, hosted on the OpenStack-based Ada cloud@CINECA. Despite being geographically near the Leonardo@CINECA facility, the Ada cloud@CINECA VMs do not mount any direct interconnection with the HPC computing nodes, requiring all data to be routed on the Internet to be transferred between the two environments.

Table 1. LLaMAv2 7B training performance on Leonardo@CINECA. 1 it = 1 data batch = 4 data samples = 8192 tokens

#Nodes	#GPUs	Loading time (s)	Queuing time (s)	Estimated execution time (hours)	Aggregate data processing speed (it/s)	Speedup	Efficiency
2	8	34	2	774	0.35	2.00	1.00
4	16	34	2	385	0.99	4.02	1.00
8	32	34	2	193	2.83	8.02	1.00
16	64	34	2	98	8.16	15.80	0.99
32	128	38	2	49	23.19	31.59	0.99
64	256	90	222	25	66.32	61.92	0.97
128	512	120	438	14	179.02	110.57	0.86

4.3. Implementation

The training process relies on the PyTorch 2.2.0 framework [52], manually compiled and optimised on both HPC facilities. As a future work, we want to make the experiment more portable and reproducible by relying on software containers (e.g., Singularity [53]), which have also recently shown promising results for performance portability.

First, datasets have been tokenized and manually moved to each HPC centre as described in Sec. 4.1. As assumed in standard FL practice, we assume data to be local to each facility. At each facility, the LLaMAv2 model has been distributed among the available computing nodes through the Fully Sharded Data Parallel (FSDP) methodology [54, 55], using different sharding strategies for each infrastructure to fit the local hardware characteristics better. From a high-level perspective, the LLaMAv2 model is sharded at the intra-node level, following a model parallel approach, and replicated on each node, embracing a data parallel approach. At the end of every round, models are aggregated on the Ada cloud@CINECA VM.

The cross-facility interactions have been modelled as a hybrid CWL workflow [22], ensuring reproducibility and reusability. In detail, the workflow has been specified using CWLv1.2 with the `cwltool:Loop` extension [21], as the current CWL standard does not support iterative workflows. However, this extension is currently under evaluation to be included as a native feature of CWLv1.3. At runtime, the workflow has been orchestrated by a StreamFlow [13] instance deployed on the Ada cloud@CINECA VM. From the VM, StreamFlow can communicate with each HPC centre through SSH connections, interacting with the workload managers to submit and monitor the training steps. StreamFlow also manages all the data transfers, i.e., it collects an instance of the model from each facility at the end of each training round and transfers back a copy of the updated model to each facility after the aggregation step. The aggregation step is performed directly on the VM since it has a low complexity with respect to model training. Note that the datasets are never moved, according to the FL principles. Fig. 1 depicts the whole XFFL workflow and its mapping onto the cross-facility execution environment described in Sec. 4.2. All the code is available as Open Source on GitHub[1].

4.4. Discussion

First, we executed a large-scale training of the LLaMAv2 7B model on the clean_mC4_it dataset on top of the Leonardo@CINECA facility to study the strong scaling of the training process itself. The goal was to check how much we can scale on a single facility and how the requested resources affect the queueing time. The queueing times and an estimation of the execution times ranging from 2 to 128 nodes are reported in Table 1. In our experiments, queueing times were negligible for up to 32 nodes but increased to over 7 minutes for 128 nodes. However, execution times were orders of magnitude higher, ranging from 774 (on 2 nodes) to 14 (on 128 nodes) hours. Therefore, despite

[1] https://github.com/alpha-unito/xffl

Table 2. LLaMAv2 7B cross-facility federated training overhead

	Leonardo@CINECA Transfer time (s)	Karolina@IT4I Transfer time (s)	Model aggregation time (s)
Min	138	178	118
Avg	217	242	133
Max	360	344	143

showing a very good strong scaling, training times offset queueing times even for large amounts of computing power, making it perfectly feasible to separate different training rounds into multiple job submissions without a major impact on the overall performance.

Another factor to consider is the loading time, which is the time required for all nodes to perform initial operations, such as loading the model and dataset into RAM and GPU memory and initialising the communication layer between all participating nodes. However, even considering the loading time overhead, which ranged from 34 to 120 seconds, separating iterations into different submissions remains a viable and sustainable option.

Having validated the overall approach as viable on one HPC centre, we ran the full experiment using the FL workflow described in Sec. 4.3 and measured all the additional sources of overhead, i.e., the transfer times between the Ada cloud@CINECA VM and the two HPC facilities and the time needed to aggregate the model on the Ada cloud@CINECA VM. We repeated the experiment five times and collected maximum, minimum, and average values reported in Table 2. Note that these times do not depend on the resources allocated in each HPC facility. Again, empirical measures promote the feasibility of the XFFL approach: none of the measured times exceeds hundreds of seconds, making them negligible compared to the tens of hours needed by each training round.

5. Conclusion and future work

In this work, we propose XFFL as a powerful tool to address the compute divide between industry and academia. By harnessing the capabilities of multiple independent computing centres for extreme-scale DNN training, XFFL offers a promising solution. Hybrid workflows, a key component of XFFL, allow users to abstract complex technical details and orchestration strategies of cross-facility workloads, making the training process more accessible. While this approach introduces some overhead due to additional queueing and loading times, these are insignificant compared to the time required for local training. Additionally, XFFL minimises cross-cluster interconnections, which are significantly slower than intra-cluster communications. Our empirical results demonstrate that model transfer and aggregation times are negligible compared to training round times, further highlighting the efficiency of XFFL.

Looking ahead, full-scale experimentation on a larger federation of EuroHPC facilities is already underway. We also plan to explore a diverse set of models and datasets to test the results' generality better. Another future development of this technology aims to ease workflow design by providing a customisable and parametric CWL workflow template while promoting better integration with StreamFlow and software container technologies, such as SLURM and Singularity. Indeed, other than fostering reproducibility, optimised NVIDIA PyTorch Singularity containers offer a considerable performance boost compared to bare-metal deployments.

Acknowledgements

This article describes work undertaken in the context of the SPACE Center of Excellence, "Scalable Parallel and distributed Astrophysical Codes for Exascale" which has received funding from the EuroHPC JU under grant agreement No. 101093441, and the Spoke "FutureHPC & BigData" of the ICSC – Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union – NextGenerationEU. We also thankfully acknowledge the CINECA award under the ISCRA initiative, and the Ministry of Education, Youth

and Sports of the Czech Republic through the e-INFRA CZ (ID:90254), for the availability of high-performance computing resources and support.

References

[1] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, P. Villalobos, Compute trends across three eras of machine learning, in: IEEE IJCNN, 2022, pp. 1–8. doi:10.1109/IJCNN55064.2022.9891914.

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90. doi:10.1145/3065386.

[3] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, et al., PaLM: Scaling language modeling with pathways, J. Mach. Learn. Res. 24 (2023) 240:1–240:113.

[4] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, Association for Computational Linguistics, 2019, pp. 4171–4186. doi:10.18653/V1/N19-1423.

[5] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707. doi:10.1016/J.JCP.2018.10.045.

[6] Z. Ma, J. He, J. Qiu, H. Cao, Y. Wang, et al., BaGuaLu: targeting brain scale pretrained models with over 37 million cores, in: ACM PPoPP, 2022, pp. 192–204. doi:10.1145/3503221.3508417.

[7] Meta, Introducing the AI research supercluster — Meta's cutting-edge AI supercomputer for AI research, accessed: 2024-04-10 (2022). URL https://ai.meta.com/blog/ai-rsc/

[8] T. Besiroglu, S. A. Bergerson, A. Michael, L. Heim, X. Luo, N. Thompson, The compute divide in machine learning: A threat to academic contribution and scrutiny?, CoRR abs/2401.02452 (2024). arXiv:2401.02452.

[9] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, et al., Llama 2: Open foundation and fine-tuned chat models, CoRR abs/2307.09288 (2023). arXiv:2307.09288.

[10] TOP500 Project, Top500 supercomputer sites, accessed: 2024-04-07 (2024). URL https://www.top500.org/

[11] I. Colonnelli, Workflow models for heterogeneous distributed systems, in: (ITADATA 2023), Vol. 3606 of CEUR Workshop Proceedings, 2023.

[12] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. F. da Silva, G. Papadimitriou, M. Livny, The evolution of the Pegasus workflow management software, Comput. Sci. Eng. 21 (4) (2019) 22–36. doi:10.1109/MCSE.2019.2919690.

[13] I. Colonnelli, B. Cantalupo, I. Merelli, M. Aldinucci, StreamFlow: cross-breeding cloud with HPC, IEEE Trans. Emerg. Top. Comput. 9 (4) (2021) 1723–1737. doi:10.1109/TETC.2020.3019202.

[14] D. D. Sánchez-Gallegos, D. D. Luccio, S. Kosta, J. L. G. Compeán, R. Montella, An efficient pattern-based approach for workflow supporting large-scale science: The DagOnStar experience, Future Gener. Comput. Syst. 122 (2021) 187–203. doi:10.1016/J.FUTURE.2021.03.017.

[15] I. Colonnelli, M. Aldinucci, B. Cantalupo, L. Padovani, S. Rabellino, C. Spampinato, R. Morelli, R. D. Carlo, N. Magini, C. Cavazzoni, Distributed workflows with Jupyter, Future Gener. Comput. Syst. 128 (2022) 282–298. doi:10.1016/J.FUTURE.2021.10.007.

[16] R. B. Roy, T. Patel, V. Gadepally, D. Tiwari, Mashup: making serverless computing useful for HPC workflows via hybrid execution, in: ACM PPoPP, 2022, pp. 46–60. doi:10.1145/3503221.3508407.

[17] R. F. da Silva, R. Filgueira, E. Deelman, E. Pairo-Castineira, I. M. Overton, M. P. Atkinson, Using simple PID-inspired controllers for online resilient resource management of distributed scientific workflows, Future Gener. Comput. Syst. 95 (2019) 615–628. doi:10.1016/J.FUTURE.2019.01.015.

[18] A. Mulone, S. Awad, D. Chiarugi, M. Aldinucci, Porting the variant calling pipeline for NGS data in cloud-HPC environment, in: IEEE COMPSAC, 2023, pp. 1858–1863. doi:10.1109/COMPSAC57700.2023.00288.

[19] D. A. Brown, P. R. Brady, A. Dietz, J. Cao, B. Johnson, J. McNabb, A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis, in: Workflows for e-Science, Scientific Workflows for Grids, Springer, 2007, pp. 39–59. doi:10.1007/978-1-84628-757-2_4.

[20] J. A. Barron-Lugo, J. L. G. Compeán, J. Carretero, I. López-Arévalo, R. Montella, A novel transversal processing model to build environmental big data services in the cloud, Environ. Model. Softw. 144 (2021) 105173. doi:10.1016/J.ENVSOFT.2021.105173.

[21] I. Colonnelli, B. Casella, G. Mittone, Y. Arfat, B. Cantalupo, R. Esposito, A. R. Martinelli, D. Medić, M. Aldinucci, Federated learning meets HPC and cloud, in: Astrophysics and Space Science Proceedings, Vol. 60, Springer, 2023, p. 193–199. doi:10.1007/978-3-031-34167-0_39.

[22] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanic, H. Ménager, S. Soiland-Reyes, C. A. Goble, Methods included: Standardizing computational reuse and portability with the Common Workflow Language, Communication of the ACM (2022). doi:10.1145/3486897.

[23] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: A. Singh, X. J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Vol. 54 of Proceedings of Machine Learning Research, PMLR, 2017, pp. 1273–1282.

[24] G. Mittone, N. Tonci, R. Birke, I. Colonnelli, D. Medic, A. Bartolini, R. Esposito, E. Parisi, F. Beneventi, M. Polato, M. Torquati, L. Benini, M. Aldinucci, Experimenting with emerging RISC-V systems for decentralised machine learning, in: ACM CF, 2023, pp. 73–83. doi:10.1145/3587135.3592211.

[25] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh, SCAFFOLD: stochastic controlled averaging for federated learning, in: ICML, Vol. 119, PMLR, 2020, pp. 5132–5143.

[26] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM, 2020, pp. 1698–1707.
[27] Y. Chen, Y. Ning, M. Slawski, H. Rangwala, Asynchronous online federated learning for edge devices with non-iid data, in: IEEE BigData, 2020, pp. 15–24.
[28] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, D. Huba, Federated learning with buffered asynchronous aggregation, in: AISTATS, Vol. 151, PMLR, 2022, pp. 3581–3607.
[29] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, S. Cui, UVeQFed: Universal vector quantization for federated learning, IEEE Trans. Signal Process. 69 (2021) 500–514. doi:10.1109/TSP.2020.3046971.
[30] F. Sattler, S. Wiedemann, K. Müller, W. Samek, Robust and communication-efficient federated learning from non-i.i.d. data, IEEE Trans. Neural Networks Learn. Syst. 31 (9) (2020) 3400–3413. doi:10.1109/TNNLS.2019.2944481.
[31] C. Wu, F. Wu, R. Liu, L. Lyu, Y. Huang, X. Xie, FedKD: Communication efficient federated learning via knowledge distillation, CoRR abs/2108.13323 (2021). arXiv:2108.13323.
[32] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1-2) (2021) 1–210. doi:10.1561/2200000083.
[33] B. Y. Lin, C. He, Z. Ze, H. Wang, Y. Hua, C. Dupuy, R. Gupta, M. Soltanolkotabi, X. Ren, S. Avestimehr, FedNLP: Benchmarking federated learning methods for natural language processing tasks, in: NAACL, Association for Computational Linguistics, 2022, pp. 157–175. doi:10.18653/V1/2022.FINDINGS-NAACL.13.
[34] R. Esposito, M. Polato, M. Aldinucci, Boosting methods for federated learning, in: D. Calvanese, C. Diamantini, G. Faggioli, N. Ferro, S. Marchesin, G. Silvello, L. Tanca (Eds.), SEBD, Vol. 3478 of CEUR Workshop Proceedings, CEUR-WS.org, 2023, pp. 439–448.
[35] T. Ben-Nun, T. Hoefler, Demystifying parallel and distributed deep learning: An in-depth concurrency analysis, ACM Comput. Surv. 52 (4) (2019) 65:1–65:43. doi:10.1145/3320060.
[36] P. Viviani, M. Drocco, D. Baccega, I. Colonnelli, M. Aldinucci, Deep learning at scale, in: Proc. of 27th Euromicro Intl. Conference on Parallel Distributed and network-based Processing (PDP), IEEE, Pavia, Italy, 2019, p. 124–131. doi:10.1109/EMPDP.2019.8671552.
[37] G. Mittone, F. Svoboda, M. Aldinucci, N. D. Lane, P. Lió, A federated learning benchmark for drug-target interaction, in: ACM WWW Companion, 2023, pp. 1177–1181. doi:10.1145/3543873.3587687.
[38] P. Foley, M. J. Sheller, B. Edwards, S. Pati, W. Riviera, M. Sharma, P. N. Moorthy, S. han Wang, J. Martin, P. Mirhaji, P. Shah, S. Bakas, OpenFL: the open federated learning library, Physics in Medicine & Biology 67 (21) (2022) 214001. doi:10.1088/1361-6560/ac97d9.
[39] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet, N. D. Lane, Flower: A friendly federated learning research framework, CoRR abs/2007.14390 (2020). arXiv:2007.14390.
[40] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y. Hsieh, et al., NVIDIA FLARE: federated learning from simulation to real-world, IEEE Data Eng. Bull. 46 (1) (2023) 170–184.
[41] C. He, S. Li, J. So, M. Zhang, H. Wang, et al., FedML: A research library and benchmark for federated machine learning, CoRR abs/2007.13518 (2020). arXiv:2007.13518.
[42] W. Zhuang, C. Chen, L. Lyu, When foundation model meets federated learning: Motivations, challenges, and future directions, CoRR abs/2306.15546 (2023). arXiv:2306.15546.
[43] B. Yuan, Y. He, J. Davis, T. Zhang, T. Dao, et al., Decentralized training of foundation models in heterogeneous environments, in: NeurIPS, 2022.
[44] M. Xu, D. Cai, Y. Wu, X. Li, S. Wang, FwdLLM: Efficient FedLLM using forward gradient, CoRR abs/2308.13894 (2023). arXiv:2308.138946.
[45] H. Chen, Y. Zhang, D. Krompass, J. Gu, V. Tresp, FedDAT: An approach for foundation model finetuning in multi-modal heterogeneous federated learning, in: AAAI, 2024, pp. 11285–11293. doi:10.1609/AAAI.V38I10.29007.
[46] L. G. Valiant, A bridging model for parallel computation, Commun. ACM 33 (8) (1990) 103–111. doi:10.1145/79173.79181.
[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS, 2017, pp. 5998–6008.
[48] G. Sarti, M. Nissim, IT5: large-scale text-to-text pretraining for italian language understanding and generation, CoRR abs/2203.03759 (2022). arXiv:2203.03759.
[49] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 140:1–140:67.
[50] M. A. Jette, T. Wickberg, Architecture of the slurm workload manager, in: D. Klusácek, J. Corbalán, G. P. Rodrigo (Eds.), JSSPP, Vol. 14283 of Lecture Notes in Computer Science, Springer, 2023, pp. 3–23. doi:10.1007/978-3-031-43943-8_1.
[51] Y. Tian, Y. Wan, L. Lyu, D. Yao, H. Jin, L. Sun, Fedbert: When federated learning meets pre-training, ACM Trans. Intell. Syst. Technol. 13 (4) (2022) 66:1–66:26. doi:10.1145/3510033.
[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, et al., PyTorch: An imperative style, high-performance deep learning library, in: NeurIPS, 2019, pp. 8024–8035.
[53] G. M. Kurtzer, V. Sochat, M. W. Bauer, Singularity: Scientific containers for mobility of compute, PLOS ONE 12 (5) (2017) 1–20. doi:10.1371/journal.pone.0177459.
[54] Y. Xu, H. Lee, D. Chen, H. Choi, B. A. Hechtman, S. Wang, Automatic cross-replica sharding of weight update in data-parallel training, CoRR abs/2004.13336 (2020). arXiv:2004.13336.
[55] Y. Zhao, A. Gu, R. Varma, L. Luo, C. Huang, et al., PyTorch FSDP: experiences on scaling fully sharded data parallel, Proc. VLDB Endow. 16 (12) (2023) 3848–3860. doi:10.14778/3611540.3611569.