Machine Learning Workflows in the Computing Continuum for Environmental Monitoring

Alessio Catalfamo[1], Atakan Aral[2,3], Ivona Brandic[4], Ewa Deelman[5], and Massimo Villari[1]

[1] University of Messina, Italy, {alecatalfamo,mvillari}@unime.it
[2] Umeå University, Sweden, atakan.aral@umu.se
[3] University of Vienna, Austria, atakan.aral@univie.ac.at
[4] TU Wien, Austria, ivona.brandic@tuwien.ac.at
[5] University of Southern California, USA, deelman@isi.edu

Abstract. Cloud-Edge Continuum is an innovative approach that exploits the strengths of the two paradigms: Cloud and Edge computing. This new approach gives us a holistic vision of this environment, enabling new kinds of applications that can exploit both the Edge computing advantages (e.g., real-time response, data security, and so on) and the powerful Cloud computing infrastructure for high computational requirements.
This paper proposes a Cloud-Edge computing Workflow solution for Machine Learning (ML) inference in a hydrogeological use case. Our solution is designed in a Cloud-Edge Continuum environment thanks to Pegasus Workflow Management System Tools that we use for the implementation phase. The proposed work splits the inference tasks, transparently distributing the computation performed by each layer between Cloud and Edge infrastructure. We use two models to implement a proof-of-concept of the proposed solution.

Keywords: Continuum · Worfklow · Pegasus · Cloud-Edge · Machine Learning

1 Introduction

Cloud-Edge computing is an innovative approach that emerged during the last years as a combination of Cloud computing and Edge computing paradigm. This new pattern exploits the strong computational capability of Cloud Infrastructure with the advantages of Edge computing: closing to data sources, reduced overhead, and data owners' security. This approach reduces the communication overhead and increases data security in modern applications where these requirements are crucial.

The environmental research community is witnessing a steady growth in the amount of data produced by various sources, including satellites, drones, Internet of Things (IoT) devices and remote sensors. These sources can contain essential data on climate, conservation activities, natural resource management and

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

environmental monitoring. However, the vastness of environmental data poses significant logistical and technical challenges.

This context is where Cloud-Edge computing comes into play. This innovative approach can reduce latency, enhance real-time processing, and optimize bandwidth usage by extending computing capabilities to the network's Edge, closer to the data source. It can allow for the seamless integration of Cloud resources, Edge devices, and workflows tailored to the specific needs of environmental applications. He is realized by me.

The proposed work uses workflows in Cloud-Edge computing to perform forecasting in the environment monitoring field. In particular, the research described in this paper proposes a methodology for distributing a Machine Learning (ML) model across the Cloud and Edge infrastructure, exploiting the Cloud-Edge Continuum pattern. We developed two workflows to perform the inference with two different models: a 1)Feed-Forward Neural Network and a 2)Transformer.

The work conducted here makes the following contributions:

- A design of Edge-Cloud solution able to deal with two workflows to make a distributed inference exploiting an automatic partition of the target model.
- A practical implementation of the workflows with Pegasus Workflow Management System (WMS) [11].
- An evaluation of the approach with a hydrogeological application.

The added value of this work comes from the definition of a new distributed architecture able to leverage the Continuum computing paradigm and from the concrete evaluation of two challenging ML inference models through a novel implementation in Pegasus Workflow Engine, in which an automatic model partition is performed.

The paper is structured as follows: Section 2 describes the current state of the art about the distribution of ML inference. Section 3 describes the research's motivation here. Section 4 describes the proposed solution and the workflows accomplished. Section 5 shows the practical implementation of the designed workflows. Section 6 evaluates the overhead introduced by the implemented workflow along Cloud and Edge infrastructures, and finally, Section 7 concludes the work showing future purposes.

2 State of the art

During the last few years, several research works have tried to exploit the Cloud-Edge Continuum approach in Artificial Intelligence tasks to reduce overhead and optimize the available resources.

For example, the work proposed in [18] was one of the first works that distributed deep neural network (DNN) inference over different computing infrastructures like Cloud, Edge, and end devices geographically distributed. The inference is distributed across all the paradigms used (Cloud, Edge, IoT), and the authors introduce an early-exit approach, too. This work evaluates the correctness of the inference for each layer in the inference phase, starting from the end

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

device and continuing to the Cloud. Although this solution involves the Cloud-Edge infrastructure, it applies a static configuration for ML model distribution.

The solution proposed in [6] (named Split-CNN) presents another approach to ML distribution inference. In particular, Spit-CNN consists of an ad hoc convolutional neural network (CNN) architecture that splits the input images into patches and distributes them independently on the convolutional layers. The authors show how this approach and a memory management algorithm can improve the scalability of distributed training for convolutional networks. The article depicts a relevant solution in inference distribution with good experimental results, but it does not address the distribution model problem among heterogeneous paradigms. Other similar solutions in scientific literature apply a distribution over CNN models and models for image elaboration. In particular, in [9], the authors proposed another solution to split the CNN structure into smaller parts with the aim of reducing memory usage. The simulation results performed on VGG16 and ResNet18 networks for the classification of CIFAR10 images demonstrated a reduction in both the quantity of memory consumption and the number of computational operations. Even in [12], the authors propose an algorithm to partition the CNN model during the inference phase. Unlike the previous research works, the split of CNN is based on device constraints and, in particular, on the stringent computational requirements of Edge devices, such as the memory used and the bandwidth used in transferring video streams from the Edge to the Cloud. The algorithm is based on finding the maximum level at which to partition the neural network so that the Edge device can contain the partition despite performing other computational tasks. Unlike the other solutions over CNN, here the authors apply a split for a Cloud-Edge distribution. Instead, in [10], the authors propose a partitioning of a DNN in order to reduce the complexity of big models that work on images' classification, such as DenseNet-169 [4] ResNet-152 [2], and Inception-v3 [17].

Other recent works dealt with a split of different ML models among different physical devices. For example, in [3], the authors proposed a distributed inference framework called EdgeFlow. The solution is designed to split the inference tasks among multiple Edge devices. The results showed that the proposed solution can reduce the inference time by 40.2% compared with other solutions. The work in [16] shows further progress in Edge devices ML distribution; the authors proposed a solution that not only distributes a DNN across multiple Edge devices but also optimizes computation resources and memory. The solution has been validated by simulation of six Edge devices running YOLOv2 DNN [15]. In [19], the authors proposed a solution to move the ML inference into IoT devices. The idea is to create a hierarchical structure with accuracy and prediction comparable to a Cloud-based solution. The experimental results showed that the accuracy of the hierarchical-based solution is the same as the Cloud-based one but the energy use and latency can be reduced by up to 63% and 67% respectively. Instead, The [20] realizes a work about mobile Edge computing, finding the optimal cellular network to split a DNN in order to find the best trade-off between accuracy and latency in the inference phase. The authors consider two kinds

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

of mobile network topologies defined "Chain" and "Mesh" and via a Neural Architecture Research technique, they find the optimal solution maximizing the accuracy and minimizing the latency.

The work realized in [5] introduces a split of ML involving heterogeneous devices and not only Edge devices. Indeed, the authors proposed a Spark cluster for distributing the inference in TensorFlow. In particular, they compared the performances of four different configurations: i) a single Raspberry Pi 4B; ii) a cluster composed of two Raspberry Pi 4B; iii) a cluster composed of two Virtual Machines (VMs); and iv) a desktop computer. To validate the proposed solution, they tested InceptionV3 on CIFAR10 and ImageNetV2. However, due to errors, all Spark-based clusters failed to perform the execution.

Among the most recent works, we must consider the solution described in [13], which realized an inference distributed splitting the layers of the neural network in a bootstrap phase and distribution according to specific layers. The authors proposed an architecture and a specific flow with the following tasks: Model Partitioning, Configuration, and Distributed Inference. The solution logically concatenates different nodes in order to spread the model partition and distribute the inference. The division strategy is based on splitting the network architecture into two parts: the head and the tail. The central innovative aspect of this work is to apply a knowledge distillation technique only to the head part, to which a bottleneck is further added to reduce the size of the output data. The results show that it is possible to implement complex models on Edge devices by sacrificing inference.

Although all the aforementioned works treated different aspects of the distribution of ML inference, none of these considered a framework that can dynamically distribute the ML model in a Cloud-Edge infrastructure. Moreover, none of the previously depicted solutions are considered an automatic process in which the model is split, distributed, and deployed between Cloud and Edge.

The research in [8] is one of the first solutions in which the ML distribution exploits the Cloud-Edge computing paradigm. In particular, in this work, a neural network is split into two parts spread across Cloud and Edge. The work proposes a solution to find the optimal split point considering different criteria: Cloud and Edge cost, introduced overhead, etc. Although we have a dynamical partition of the model between Cloud and Edge, the solution only considers a partition in two parts.

Our work shows generic progress in state of the art, considering also a dynamic partition of a Transformer model, never thought of in a Cloud-Edge environment at the time of writing.

3 Motivations and Use Case Description

As established earlier, the Cloud-Edge Continuum integrates two fundamental concepts for service deployment and delivery. It takes advantage of the robust computation and infrastructure offered by Cloud computing and the benefits of Edge computing. The needs of contemporary applications, where minimal

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

latency, real-time processing, and effective resource management have become critical, led to the creation of this innovative architecture. We can now take advantage of the benefits of Edge computing, such as reduced latency and improved security, along with the benefits of the Cloud, such as easy access to computing resources and services. Fundamentally, the Cloud-Edge Continuum connects the enormous processing power of the Cloud with the immediateness of Edge devices. Applications can exploit what is offered by both Cloud and Edge thanks to the smooth coordination of data flow and processing duties between them. In order to offload ML inference duties across Cloud and Edge infrastructure, we present a workflow idea in our suggested solution that operates within the Cloud-Edge Continuum context. This new method makes use of Cloud infrastructure's powerful computational power for heavy inference workloads and Edge computing's low latency for data collecting.

3.1 Hydrological use case

A region that is drained by a specific surface water or groundwater system is known as a hydrological basin or catchment. The hydrogeological catchment can play an important role in sustaining ecosystems providing drinkable water and supporting several industries. For this reason, runoff forecasting can predict any water shortages or signal any anomalous floods and it's a crucial computation in environment monitoring. Hence, the urgent need to address important water resource management and environmental concerns is driving the development of a Cloud-Edge Workflow for AI inference for the runoff level. In our solution, for the hydrogeological use case, we have exploited the LAMAH dataset [7]. This dataset, covering the Danube River's basins across three different countries, offers diverse information, from geophysical data to historical water levels. In particular, the dataset involves about 800 catchments. For each catchment, it provides a time-series data split for the day and hour of the last 35 years. The dataset covers about 60 attributes for each catchment, covering topography, climatology, hydrology, land cover, vegetation, soil and geological properties. In our case, we have considered the daily data about runoff level to design our models and distribute them between Cloud and Edge tiers. Furthermore, our solution emphasizes resource optimization, particularly in remote or resource-constrained areas. The distributed approach of the workflow maximizes resource efficiency, reducing energy consumption and costs associated with data transmission and Cloud-based processing. The proposed solution performs AI inference in order to forecast one of the columns provided by the LAMAH dataset: the runoff of the basin expressed in the dataset as *Qobs*. As of the time of writing, there is no existing solution that leverages the Cloud-Edge Continuum and Workflows for distributing AI inference processes. Through this approach, we aim to enhance the management and orchestration of AI inference across a variety of heterogeneous devices.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

4 Proposed Solution

In this section, the proposed solution will be depicted and the designed workflow will be described. In particular, we'll describe the general methodology for the implementation of a ML model in a Cloud-Edge Workflow, highlighting how the model will be partitioned and distributed. We'll describe the strategy by which the partition of the model is carried out, preliminary works computed over the model to train it and the workflow structure.

4.1 Splitting Methodology

The main strategy applied to split the inference process consists of distributing the components belonging to the model. In this work, we consider two main ML models that can partially generalize the main solutions applied to time-series data. As previously mentioned, the models considered are: 1)a Forward DNN with two hidden layer and 2) a transformer model [21] for the time-series data prediction [22]. The splitting strategy will be described for each model exploited. This step can make possible an automatic partition of the model inside the workflow we are going to realize in the Cloud-Edge Continuum environment.

DNN We can consider the typical Forward DNN constituted of homogenous layers. To design the strategy we have considered each DNN as an ensemble of Linear Layer (LL) and Activation Layer (AL). The approach consists of splitting sequentially the layer into equal parts when it's possible and in general, to minimize the difference among the number of layers of each part. In our case, the DNN has two hidden layers but the methodology can be generalized to any number of layers in a DNN Considering L the layers set and $N = |L|$ the number of total layers in the DNN. Considering N_p the number of parts we want to split the Neural Network (in our case it is equal to 4) we establish the size of each part S_k considering

$$S_k = \lfloor \frac{N}{N_p} \rfloor + 1 \tag{1}$$

for the first P parts where

$$P = N \mod N_p \tag{2}$$

and

$$S_k = \lfloor \frac{N}{N_p} \rfloor \tag{3}$$

for the remaining parts.

Transformer The splitting strategy for the Transformer model is focused on the internal architecture of the Transformer itself which is not homogenous like DNN one. According to Transformer definition [21] we can summarize it in these components that will become the partitions of our partitioned inference:

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

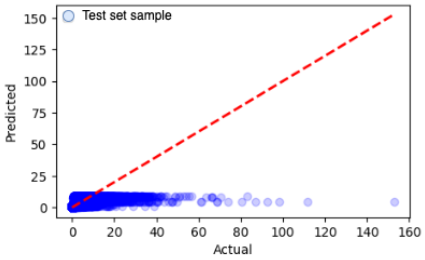

Fig. 1. Error representation for Feed-Forward model.

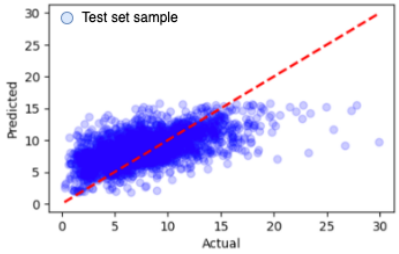

Fig. 2. Error representation for Transformer model.

- a Linear Layer for the Input Preparing of the Encoder
- a Linear Layer for the Input Preparing of the Decoder
- the positional Encoding component
- the Encoder Component
- the Decoder Component

4.2 Preliminary Works on Models

The preliminary works performed were related to the training of the two models (described in Section 4.1). The models were trained by exploiting each catchment data. In this way, a weights set is trained for each catchment to better generalize and manage the forecast for each catchment. In particular, the preliminary works on the dataset and model can be summarized to:

- Preprocessing phase in which the label (*Qobs* column in the dataset) and all the features are extracted, merged and filtered. In particular, we choose the following feature sets(as named in [7]): gauge referred attributes, meteorological variables, topographic indices, and climatic indices. Starting from the feature sets we keep only the more correlated features and finally, we have a dataset with 34 features. In this phase we remove outliers too;
- Train, Validation and Test set split;
- Train of the model;

Figure 2 and 1 show the error of the two models trained on a single basin. The graph depicts for each test set sample the error of the prediction with respect to the actual value. We can say that the Transformer model reaches a lower error on a test set after the training. The detail of training hyperparameters is out of the scope of the work presented here which is focused on the inference phase.

In the high-level architecture designed in our use case, two different levels are considered. The *lower* level is the Edge layer that interacts and collects the data victim of the inference. The *higher* layer is the Cloud Layer in which the Cloud Infrastructure is hosted.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

4.3 Workflows Design for Proof of Concept

The designed workflows split the inference processes into different chunks that can be distributed among Cloud and Edge infrastructures. To prove our solution, as mentioned before, we accomplished two different workflows that perform the inference through two different ML models. The Workflow's jobs will be depicted in detail.

Classical Feed Forward Network Inference Workflow
The workflow that realizes the Feed Forward Network inference is graphically depicted in Figure 3.
The descriptions of each jobs depicted are the following:

a. Preprocessing: This initial job involves the conversion from the raw data collected to the input of the next jobs and the input of the models
b. Preinference: Although this job doesn't explicitly involve partitioning, it could be responsible for preparing the models and data structures required for the partitioned inference. In particular, during this job, the right weight set is loaded. Indeed, as specified, different weights are trained for different basins.
c. Generate Partitions: This job applies the partitioning strategy already described in 4.1. In the case of Feed Forward Neural Network, the job split the model in homogeneous parts. It takes the trained model from the previous jobs and generates multiple partitions of the model. These partitions will be exploited for the inference in the next jobs.
d. Inference Jobs: Each of these jobs is an inference task, it performs a part of the total inference thanks to the partitioned model accomplished in the previous job. By splitting the inference into these separate jobs, it is possible to offload each job to a different infrastructure (e.g. Cloud or Edge)
e. Conversion: The final job takes the processed data and converts it into a more usable format for further analysis or reporting.

Transformer Inference Workflow
The workflow that realizes the Transformer model inference is graphically depicted in Figure 4.
The descriptions of each jobs depicted are the following:

a. The jobs related to preprocessing, preinference and conversion of the final output and generating partitions have the same logic already described for Feed Forward Network.
b. Generate Partitions: Unlike Feed Forward Network, here, the Generate Partitions job splits the model in the different components of Transformer model, as specified in 4.1.
c. Inference Decoder: This job performs the final inference that involves the Decoder of Transformer and that involves the Decoder Input and the Encoder Output.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

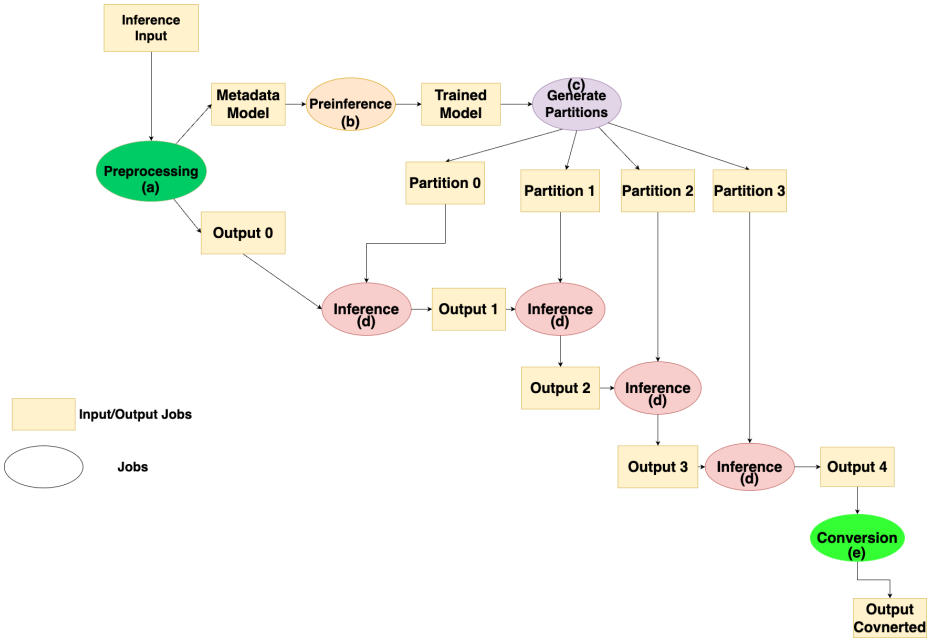

Fig. 3. Workflow for Inference with Deep Feed Forward Neural Network.

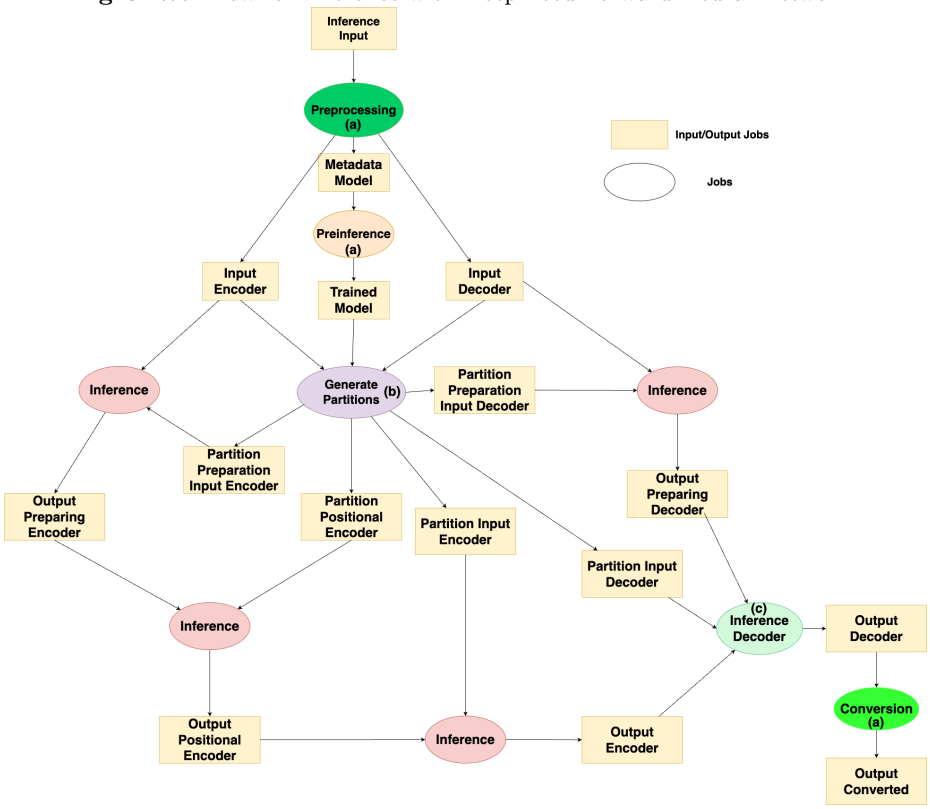

Fig. 4. Workflow for Inference Transformer Model.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

5 Implementation for a Cloud-Edge Continuum environment

In this section, the practical implementations of the designed workflows will be described. The carried-out implementation has been focused on the workflows that, as already specified, will be deployed in a Cloud-Edge Continuum environment. The workflows have been accomplished through Pegasus WMS [11].

5.1 Introduction to PEGASUS

Pegasus WMS is an open-source tool that empowers scientists to create custom abstract workflows using high-level APIs. For the execution of these workflows, Pegasus utilizes HTCondor, which is an open-source software framework designed for distributed computing. HTCondor's primary function is to manage and schedule workloads. In collaboration with HTCondor, Pegasus offers a range of functionalities that enable the creation of an abstract workflow. This transformation allows HTCondor's DAGMan to efficiently allocate resources and schedule jobs within the HTCondor cluster.

The HTCondor cluster foresee three different roles within a cluster:

1. Submit Node: The Submit Node serves as the entry point for users to submit their computational tasks to the HTCondor system. Users provide job descriptions, and the Submit Node interfaces with the HTCondor Central Manager to manage and schedule the job queue. It does not directly execute the tasks but takes charge of job submission and monitoring. To ensure seamless functionality, both Pegasus WMS and HTCondor binaries must be installed on this node, as Pegasus utilizes HTCondor utilities to deploy the workflow.
2. Central Manager: The Central Manager assumes a central role in the HTCondor. It maintains an overview of the status of all machines and oversees the matchmaking process. In the HTCondor system, each worker node periodically shares its characteristics with the central manager. The central manager gathers information from the scheduler and handles the task of matching jobs to available resources. It is instrumental in resource allocation and job distribution.
3. Execution Node: Execution Nodes represent the worker machines within the HTCondor cluster. These nodes are responsible for the actual execution of computational tasks scheduled by the Central Manager. Execution Nodes can have varying configurations, and they run user-submitted tasks in isolated environments, ensuring that tasks do not interfere with each other. HTCondor involves different architectures for the execution nodes (e.g. arm64, x8086).

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

5.2 Workflows implementation.

The practical implementation of the workflows consists of the creation of a YAML file with the following structure that represents the workflow and that will be the input of *pegasus-run* command:

1. **Metadata**:
   - `x-pegasus`: This section contains metadata about the workflow, including the creation language, creator, and timestamp.
2. **Pegasus Version**:
   - `pegasus: 5.0.4`: Specifies the version of Pegasus used for the workflow.
3. **Workflow Name**:
   - `name: qobs_prediction_timeseries`: The name of the workflow, indicating its purpose related to distributed time series prediction.
4. **Workflow Jobs**:
   - The workflow consists of several jobs, each with a type, name, ID, and a list of arguments. These jobs are organized in a sequence, serving various purposes.
   a. `Preprocessing`: Prepares data and produces output for use in subsequent jobs.
   b. `Preinference`: Preprocesses data in preparation for inference.
   c. `Generate Partitions`: Creates multiple data partitions, likely for parallel processing.
   d. `Inference Jobs`: Perform inference on different data partitions.
   e. `Inference Decoder`: May assemble results from previous inference jobs.
   f. `Conversion`: Converts processed data into a more usable format.
5. **Job Dependencies**:
   - The workflow specifies dependencies between jobs to ensure the correct order and coordination of tasks.

In practice, YAML file is never written from scratch but it's generated by the API provided by Workflow for main languages programming. In our case, we exploited Python Pegasus API to generate YAML files.

6 Performance Assessment

Table 1. Cluster's nodes characteristics

#	Tier	Model	CPU	Memory	OS
2	Cloud	Openstack VM	Intel Xeon 8x 4.0GHz	16 GB	Ubuntu 20
2	Edge	Raspberry Pi 4	ARM64 SoC 4x 1.5GHz	4 GB	Raspberry OS ARM64

This section depicts the experimental evaluations of the two provided solutions. The evaluations have focused on analysing the overhead introduced by the Workflow implementations concerning the infrastructure and the balancing across Cloud and Edge. Moreover, the test evaluated the overhead considering the model exploited in the single workflow.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

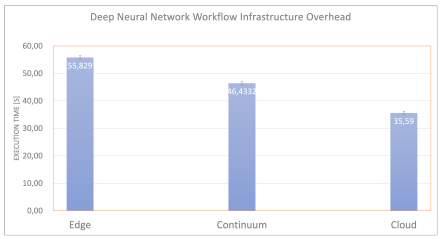

Fig. 5. Workflow for Inference with Deep Feed Forward Neural Network.

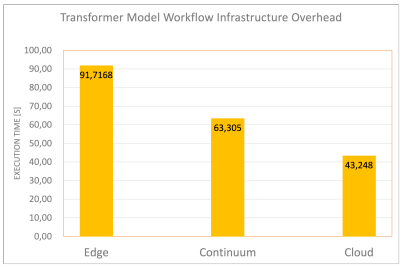

Fig. 6. Workflow for Inference with Transformer model.

6.1 System Testbed Setup

The Pegasus Workflows has been tested on a 4 Nodes Kubernetes Cluster composed of two nodes on the Cloud tier and two on the Edge tier. The System testbed involves two VM Machines deployed through OpenStack that constitute the Cloud tier. The Edge tier is realized via two Raspberry PIs. All the nodes can host each job belonging to Workflow.

6.2 Experimental Results

The test focused on overhead introduced in the distribution of ML Inference in a Cloud-Edge environment. As described, Pegasus WMS allows orchestration of the jobs involved in the workflow and with our tests we have considered three possible job distributions among Cloud and Edge tiers:

- Edge setting in which all the workflow is hosted on Edge tier
- Continuum setting in which the workflow jobs are equally distributed between Cloud and Edge. In particular, considering the jobs described in Section 4.3, we have deployed the preprocessing, preinference, *generate_partitions* and the first two jobs of the inference phase are deployed on the Edge tier. The other jobs (other parts of the inference, and final conversion) are deployed on the Cloud tier.
- Cloud setting in which all the workflow jobs are deployed on the Cloud tier.

Moreover, we compared the two models distributed through the workflow performing the average of the model concerning the infrastructure is deployed to.
The performed tests were performed through the *pegasus-statistics* tool provided by the Pegasus WMS command line interface. This tool collects all the most important statistics of the performed workflow including the cumulative wall time that consists of the sum of each workflow job time execution.
The experimental measures showed interesting results. Indeed, the infrastructure exploited for the workflow deployment is the most important parameter for overhead introduced by the execution of the whole workflow.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

As depicted in Figures 5 and 6, the Cloud infrastructure requires less time to perform the whole workflow.
Specifically, Figure 5 shows the time consumption of the workflow concerning the distribution among Cloud-Edge Continuum infrastructure of the Feed-Forward Neural Network workflow which is described in Figure 3. The image shows that the workflow fully Edge deployed is slower than the Continuum deployment in which the workflow runs across Cloud and Edge infrastructure. Moreover, the measure depicts that a fully Cloud deployed workflow is the fastest solution for the deployment. This kind of result could be considered different from the heuristic usually defined in the scientific literature in which Edge computing ensures less overhead [23] [14] [1]. The difference between Edge and Continuum and between the Continuum and Cloud jobs distribution is about 10 seconds according to performed measures. These values prove that the Continuum approach can be considered, in this case, a right trade-off to ensure that final raw data will be kept on Edge devices and to exploit Cloud infrastructure. Figure 6 shows the same comparison with Transformer workflow implementation. Even here, the trend suggests that the Cloud component saves time in the jobs distribution, and even in this case, the Continuum approach can be considered a good trade-off. Moreover, in this case, in which the model is more complex, the Continuum and the Cloud save a greater amount of time than the DNN workflow.
Figure 7 shows an average overhead comparing the two models. Since the Transformer model is more complex than a Feed-Forward Deep Neural Network, the amount of time spent to perform the inference is greater.

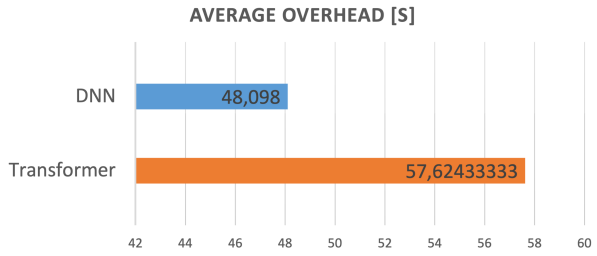

Fig. 7. Average of introduced Overhead for model.

7 Conclusions and future works

In this study, we have successfully designed two Cloud-Edge Workflows capable of processing raw data collected from Edge devices. These workflows enable distributed inference, allowing each job to be flexibly deployed on either the Edge or Cloud infrastructure. Our work includes the practical implementation

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

of these workflows, which was seamlessly managed using the WMS, Pegasus. This practical implementation demonstrates the feasibility and effectiveness of our approach in a real-world context. Looking ahead, our research opens up avenues for future investigations. An experimental evaluation of the workflow with varying job distributions between the Cloud and Edge infrastructure can provide valuable insights into performance optimization. Additionally, the next steps could involve the development and implementation of new models, considering different datasets and use-cases. These extensions will not only enhance the versatility of the workflows but also address the evolving needs of Cloud-Edge Computing applications.

Acknowledgements

This research was funded in part by the Austrian Science Fund (FWF) through following projects: *Transprecise Edge Computing (Triton)* 10.55776/P36870; *Trustworthy and Sustainable Code Offloading (Themis)* 10.55776/PAT1668223; *Sustainable Watershed Management Through IoT-Driven AI (Swain)* 10.55776/I5201, and by the Austrian Research Promotion Agency (FFG) through the following project: *Satellite-based Monitoring of Livestock in the Alpine Region (Virtual Shepherd)*, FFG Austrian Space Applications Programme ASAP 2022 #53079251. This research was also funded by the Italian Ministry of Health, Piano Operativo Salute (POS) trajectory 4 "Biotechnology, bioinformatics and pharmaceutical development", through the Pharma-HUB Project "Hub for the repositioning of drugs in rare diseases of the nervous system in children" (CUP J43C22000500006) and by Piano Operativo Salute (POS) trajectory 2 "eHealth, diagnostica avanzata, medical device e mini invasività" through the project "Rete eHealth: AI e strumenti ICT Innovativi orientati alla Diagnostica Digitale (RAIDD)"(CUP J43C22000380001). Ewa Deelman's work was funded by the U.S. National Science Foundation under grants numbers 2331153 and 2103508 and by the U.S. Department of Energy under grant number DE-SC0024387.

References

1. K. Cao, Y. Liu, G. Meng, and Q. Sun. An overview on edge computing research. *IEEE Access*, 8:85714–85728, 2020.
2. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.
3. C. Hu and B. Li. Distributed inference with deep learning models across heterogeneous edge devices. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*, pages 330–339, 2022.
4. G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks, 2018.
5. N. James, L.-Y. Ong, and M.-C. Leow. Exploring distributed deep learning inference using raspberry pi spark cluster. *Future Internet*, 14(8), 2022.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27

6. T. Jin and S. Hong. Split-cnn: Splitting window-based operations in convolutional neural networks for memory system optimization. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 835–847, New York, NY, USA, 2019. Association for Computing Machinery.
7. C. Klingler, K. Schulz, and M. Herrnegger. Lamah-ce: Large-sample data for hydrology and environmental sciences for central europe. *Earth System Science Data*, 13(9):4529–4565, 2021.
8. D. Luger, A. Aral, and I. Brandic. Cost-aware neural network splitting and dynamic rescheduling for edge intelligence. In *Proceedings of the 6th International Workshop on Edge Systems, Analytics and Networking*, EdgeSys '23, page 42–47, New York, NY, USA, 2023. Association for Computing Machinery.
9. E. MalekHosseini, M. Hajabdollahi, N. Karimi, S. Samavi, and S. Shirani. Splitting convolutional neural network structures for efficient inference, 2020.
10. Y. Matsubara, S. Baidya, D. Callegaro, M. Levorato, and S. Singh. Distilled split deep neural networks for edge-assisted real-time systems. In *Proceedings of the 2019 Workshop on Hot Topics in Video Analytics and Intelligent Edges*, 2019.
11. G. Mehta, E. Deelman, K. Vahi, and F. Silva. Pegasus Workflow Management System: Helping Applications From Earth and Space. In *AGU Fall Meeting Abstracts*, volume 2010, pages IN41B–1362, Dec. 2010.
12. R. Mehta and R. Shorey. Deepsplit: Dynamic splitting of collaborative edge-cloud convolutional neural networks. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 720–725, 2020.
13. A. Parthasarathy and B. Krishnamachari. Defer: Distributed edge inference for deep neural networks. In *2022 14th International Conference on COMmunication Systems &; NETworkS (COMSNETS)*. IEEE, Jan. 2022.
14. J. Pérez, J. Díaz, J. Berrocal, R. López-Viana, and A. González-Prieto. Edge computing: A grounded theory study. *Computing*, 104(12):2711–2747, dec 2022.
15. J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger, 2016.
16. R. Stahl, Z. Zhao, D. Mueller-Gritschneder, A. Gerstlauer, and U. Schlichtmann. Fully distributed deep learning inference on resource-constrained edge devices. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2019.
17. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015.
18. S. Teerapittayanon, B. McDanel, and H. T. Kung. Distributed deep neural networks over the cloud, the edge and end devices. *Proceedings - International Conference on Distributed Computing Systems*, pages 328–339, 7 2017.
19. A. Thomas, Y. Guo, Y. Kim, B. Aksanli, A. Kumar, and T. S. Rosing. Hierarchical and distributed machine learning inference beyond the edge. In *2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pages 18–23, 2019.
20. Y. Tian, Z. Zhang, Z. Yang, and Q. Yang. Jmsnas: Joint model split and neural architecture search for learning over mobile edge networks. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 103–108, 2022.
21. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.
22. Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey, 2023.
23. W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang. A survey on the edge computing for the internet of things. *IEEE Access*, 6:6900–6919, 2018.

ICCS Camera Ready Version 2024
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-63775-9_27