Towards Seamless Management of AI Models in High-Performance Computing

Sixing Yu[1], Murali Emani[2], Chunhua Liao[3], Pei-Hung Lin[3], Xipeng Shen[4], Ali Jannesari[1]

[1]Iowa State University, Ames, IA
[2] Argonne National Laboratory, Lemont, IL
[3] Lawrence Livermore National Laboratory, Livermore, CA
[4] North Carolina State University, Raleigh, NC
{yusx, jannesar}@iastate.edu, {liao6, lin32}@llnl.gov, memani@anl.gov, xshen5@ncsu.edu

Abstract

With the increasing prevalence of artificial intelligence (AI) in diverse science/engineering communities, AI models emerge in an unprecedented scale among various domains. However, given the complexity and diversity in the software and hardware environments, reusing AI artefacts (models and datasets) is extremely challenging, specially with AI-driven science applications. Building an ecosystem to efficiently run and reuse AI applications/datasets at scale becomes increasingly important for diverse science and engineering and high performance computing (HPC) community. In this paper, we innovate over an HPC-AI ecosystem – HPCFair, which enables the Findable, Accessible, Interoperable, and Reproducible (FAIR) principles. HPCFair enables collection of AI models/datasets allowing users to download/upload AI artifacts with authentications. Most importantly, our proposed framework provides user friendly API for users to easily run inference jobs and customize AI artifacts to their tasks as needed. Our results shows that, with HPCFair API, users irrespective of technical expertise in AI, can easily leverage AI artifacts to their tasks with minimal efforts.

Introduction

With the outstanding performance achieved by artificial intelligence (AI) and machine learning (ML), AI artifacts (AI models and datasets) being increasingly adopted to diverse science and engineering domains, such as materials discovery, ecology, cosmology, biology and wildlife conservation. However, given the complexity and diversity in the software and hardware environments, reusing AI artifacts is extremely challenging, specially with AI-driven science and engineering applications. Additionally, AI artifacts developed in various scientific domains makes it extremely challenging for scientists to fetch, reuse, and reproduce. Introducing frameworks to reasonably access, reproduce and run those AI applications at scale for diverse science and engineering communities, becomes crucial to accelerate science with high-performance computing (HPC).

We first list the key challenges for diverse scientific communities to apply AI artifacts, which need to be addressed by such AI artifact management framework. First, AI artifacts rely on complex software and hardware dependencies. Second, the dependencies vary across AI artifacts. For any given AI artifacts, we need to configure running environments for it. Third, AI artifacts supported by different backend implementations (e.g., C++ and Python) usually have interoperability challenges. Forth, applying AI artifacts requires diverse domain scientists' significant programming skills beyond science. Fifth, it is hard to find, access, interoperate, and reproduce a target AI model available in public repositories. Sixth, It is hard for scientists to find a target model that matches their needs perfectly, while customizing AI artifacts need extraordinary efforts. e.g, hundreds of hyper-parameters for tuning Lastly, lack of benchmark and standardization processes. The experimental results are hard to reproduce on the user's customized tasks.

Although the existing HPC-AI artifact management ecosystem (Wolf et al. 2019; Chard et al. 2019) significantly simplifies the threshold for applying AI artifacts, nevertheless, they are dedicated to serving computer science and software engineering domain scientists, which fails to provide solutions for diverse scientific communities. Such challenges have barely been addressed by existing HPC-AI ecosystems.

In this paper, we propose novel techniques to HPCFair (Verma et al. 2021; Nan et al. 2021) – an HPC-AI model and data management system, which enables AI artifacts Findable, Accessible, Interoperable, and Reproducible (FAIR principles) as well as provides user-friendly interfaces/APIs for diverse domain scientists adopting AI artifacts to their in-demand research tasks. Specifically, HPCFair containerized AI artifacts, where all the executing dependencies for given artifacts are built in an associate virtual machine independently. Therefore, the proposed work provides users with a friendly executing environment and bypasses the labor-costly environment established on both hardware and software. Besides that, we designed an HPC ontology to efficiently implement FAIR principles, which enables scientists to easily share and fetch target AI artifacts.

We summarised our contributions as follows:

- We proposed a novel model knowledge management system.
- Our proposed solution significantly simplified AI model deployment for domain scientists.
- Provide user friendly APIs for scientists customize AI

products to their demands.

Background

Since AI artifacts popped up on a giant scale, extensive efforts have been devoted to developing efficient AI artifact management tools. In this section, we summarized the State-of-The-Art (SoTA) AI artifacts tools.

Container Platform

A recent popular trend to improve the reproducibility of AI artifacts is containerization, which enables developers to pack the source code as well as running dependencies and provides an operation system-independent virtual environment for executing target AI artifacts. SoTA containerized platform such as Docker (Merkel 2014) and Singularity (Kurtzer, Sochat, and Bauer 2017) enables developers to integrate their codes and dependencies into containers—standardized executable components, and hence, executable in any operating system. Nowadays, great efforts are devoted to specializing containerized machine learning (ML) models and datasets, such as MLCube (Kahng, Fang, and Chau 2016). However, existing containerized platforms are targets to developers publish their works and require expert knowledge for configuration. It is hard for diverse domain scientists to specialize in it.

AI artifacts Hub

AI artifacts Hubs gather collections of AI models and datasets and provide a user-friendly interface to search and reproduce AI artifacts. For instance, Data and Learning Hub for Science (DLHub) (Chard et al. 2019), a cloud-hosted learning system, enables developers to publish their models with flexible access control. Collective Knowledge Framework (cKnowledge) (Fursin 2021) constructed a database of AI components as well as provides APIs and terminal interfaces to efficiently manage research projects for developers. Hugging Face (Wolf et al. 2019) offers NLP models and datasets, such as Transformer models, as multi-platform supportive open-source libraries that help users download, infer, optimize, and reuse AI models. Tensorflow Hub and PyTorch Hub enable developers to upload their customized model architecture and pre-trained weights in the cloud database and provide APIs to share public models. However, the AI components shared in PyTorch and Tensorflow Hub have limited their backend which hinders switching the programming frameworks as needed.

Approach

In this section, we will present how HPCFair lowered the threshold for diverse scientific communities adopt AI to their research. In essence, HPCFair introduced four components to provide scientists with a user-friendly interface. First, we proposed object converter components to enable programming framework-agnostic implementation. Then, we introduced AI artifact containerized components, which allow AI artifacts to run independently of the operating system. To allow scientists to run AI artifacts effortlessly even without a programming background, we designed a straightforward user query rule and established robust user query processing components. Additionally, to enable AI artifacts Findable, Accessible, Interoperable, and Reproducible (FAIR) principles, we designed an HPC ontology to run our proposed platform in HPC clusters.

Enable AI artifacts Collaboration

AI artifacts emerge on an unprecedented scale and have been developed by different underlying systems, such as different languages (Python, C++) and frameworks (Scikt-learn, PyTorch, TensorFlow), and AI artifacts in different frameworks are not transferable. Hence, it raises significant challenges for users inter-operate AI artifacts with distinct underlying backends. For instance, an AI model implemented in C++ is hard to integrate with an AI dataset in Python implementation. Domain scientists tend to be challenged to incorporate AI artifacts in their applications, where they have to switch back and forth between different developing backends.

Thanks for recent efforts in ONNX (Bai et al. 2019) (a community AI project for building general AI model formats), which use extensible computation graph models to represent AI models built with different frameworks. Intuitively, with its framework and platform-independent computational graph representation, AI models developed with different frameworks can be transferred to a general format, and hence, support interoperability between frameworks. However, such a great contribution has barely been used by existing HPC-AI tools. Therefore as shown in Figure 1 AI artifacts converter, HPCFair developed an online running process that any customized AI model that has been shared, uploaded, and pushed to the HPCFair database would be automatically transferred to ONNX.

Containerized AI artifacts

Since our target users are among different scientific domains and have various hardware environments, we aim to provide solutions for deploying AI artifacts among different platforms. The benefit of an AI model container image can be briefly summarized as follow: first, once the container image is built, it will provide a virtual executing environment for the associate AI model that is independent of local devices. Second, the container image can generalize the model to different software/hardware systems, and save great efforts in environment configurations. Lastly, the container image can be executed easily.

Hence, to improve and reproduce experiments with AI artifacts we aim to collect experiments run-time system and supporting metadata configuration information. Specifically, we leverage MLCube (Kahng, Fang, and Chau 2016) container storing essential runtime experimental configurations and states of AI models. A containerized object is represented by a configuration file, which contains information on the object's runtime supporting libraries and hyperparameters. Besides that, uniqueness checks are been performed to guarantee there is no duplicate uploading in the underlying database.

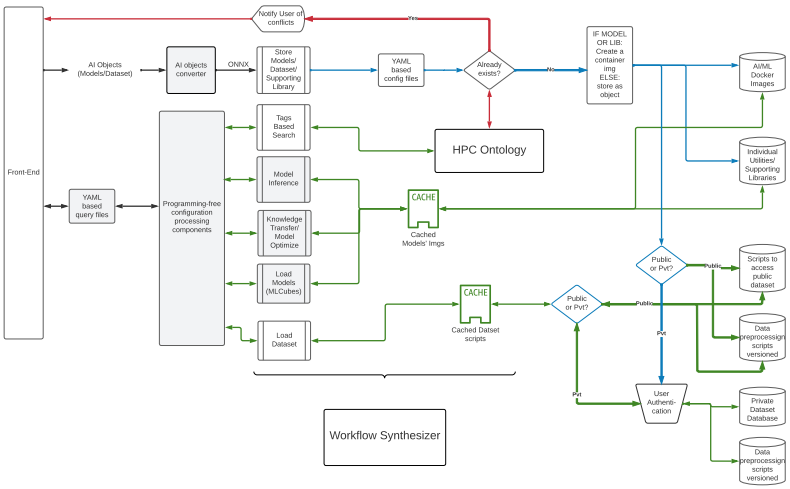

Figure 1: Designed Workflow for HPCFair.

User-Friendly Query Rule Design

As shown in Figure 1, to provide a friendly interactive interface for users, every query made by users would initialize the proposed components. Users may provide configuration files to specify tasks and parameters as needed for their tasks. In HPCFair, we designed four groups of configuration arguments to conduct main tasks (store models/datasets, tag-based search, model inference, knowledge transfer/model optimization, load models, and load dataset) provided by HPCFair APIs. Listing 1 shows the example configuration for model conversion. First configuration arguments group is general arguments, where a user specifies which task to perform, and HPCFair will initialize the corresponding components (As shown in Listing 1 lines 1-3). Then, user provides the device arguments (Listing 1 lines 5-9), user specify local device information. Next group of configuration arguments is the task arguments (Listing 1 lines 12-16), such as input, working path, etc. Lastly, the output arguments specify where HPCFair export the output (Listing 1 lines 18-19).

Designed Workflow for FAIR Principles

Our ultimate goal is to provide scientists with a friendly platform to fetch, share, and apply AI artifacts. As shown in Figure 1, we designed an efficient online workflow for HPCFair (Liao et al. 2021). First, to assist scientists in efficiently finding target AI artifacts (Findable), HPCFair registered and indexed descriptive metadata with corresponding AI artifacts together as a searchable resource. The metadata contains rich descriptive information about associated AI artifacts and is assigned a globally unique and persistent identifier, which significantly enhances searchability. Second, users can easily access AI artifacts in HPCFair database through the designed communication protocol (Accessible). Such communication protocol enables users to share or discover their target AI artifacts efficiently. Additionally, HPCFair also provides authorized credentials for users securely access AI artifacts wherever necessary. To qualify AI artifacts to interoperate among various AI frameworks (Interoperable) at the application level, the object conversion process on HPCFair server continuously transforms communicated AI models to ONNX format, equipping application users to transform models from one format to another as needed. Lastly, the scientific community oftentimes interacts among researchers to share and reuse crucial components. HPCFair provides metadata with detailed provenance to reuse the components to build an AI pipeline by plugging the data artifacts (Reproducible). The loosely coupled nature of the stored data enables efficient development.

Evaluation

In this section, we conduct comprehensive evaluations for HPCFair under different practical scenarios and use demos and examples to show the ease of scientists applying AI artifacts by using HPCFair.

AI artifacts collaborations

As AI artifacts are often implemented by diverse frameworks, enabling collaboration among AI artifacts becomes challenging. HPCFair introduces object converter components and provides APIs for a user to allow AI artifacts collaborations. To assess the HPCFair with a general use case, we experiment with interfacing two AI models implemented with PyTorch and TensorFlow respectively. We consider a popular encoder-decoder model structure, given an encoder implemented on PyTorch and a decoder developed by TensorFlow, our goal is to construct an AI model from the given encoder and decoder.

To achieve model collaboration, we first leverage HPCFair APIs to convert target AI artifacts to ONNX formats, then use HPCFair built-in inference API to run the model. To leverage functional APIs built-in HPCFair, the user provides a straightforward configuration file. In the model collaboration task, we first configure the model conversion task configuration file, as shown in Listing 1. As shown in the configuration file, the user specifies the essential AI artifacts information, such as the backend framework, and checkpoint

directory. The output file would be saved into the path user defined under out_args.

```
general_args:
  task: "conversion"
  backend: ["pt","tf"]

device_args:
  worker_num: 4
  device: "cpu"
  gpu_mapping_file: ''
  gpu_mapping_key: ''

model_args:
  model_name: ["encoder","decoder"]
  model_file: ["./ckpt/encoder.ckpt",
    "./ckpt/decoder.ckpt"]
  onnx_version: 10

out_args:
  export_file: ["encoder.onnx","decoder.
    onnx"]
```

Listing 1: Configuration for model conversion

After target model been converted to uniformed ONNX file, next step is to run the model. Similarly, HPCFair provides high-level APIs for users run AI artifacts without programming expertise knowledge. Listing 2 shows the inference configuration file.

```
general_args:
  task: "inference"
  tag: "collaboration"
  backend: "onnx"

device_args:
  worker_num: 4
  device: "cpu"
  gpu_mapping_file: ''
  gpu_mapping_key: ''

task_args:
  model_name: ["encoder","decoder"]
  model_file: ["encoder.onnx", "decoder.
    onnx"]
  onnx_version: 10
  input: "input.txt"

out_args:
  export_file: "out.txt"
```

Listing 2: Configuration for model collaboration for inference

The most exciting part of HPCFair is that it is fairly simple to call the APIs, which usually with one-line codes to finish a task. Listing 3 shows we call HPCFair-provided Python APIs to finish model collaboration tasks based on the configuration files. Model collaboration is a combined task with model conversion and model inference tasks. In the first line, we import the HPCFair python APIs. then in the main function (lines 3-6), we first create an API object (line 4). Then perform model conversion (line 5). Lastly, model collaboration (line 6). Taking the advantage of the robust high-level APIs, we finish the complex model collaboration task in 3 lines codes.

```
from hpcfair import modelAPI

if __name__ == '__main__':
    api = modelAPI()
    api.conversion(path_to_config)
    api.collaborate(path_to_config)
    api.container(path_to_config)
```

Listing 3: Call HPCFair APIs

Inference AI artifacts via HPCFair

In AI artefacts inference task, users provides an input, HPCFair run the target AI artefacts on that input and return output. As mentioned before, to support multi-framework and underlying language, HPCFair automatically transfer AI artefacts to onnx, hence, greatly simplified inference process. Inside HPCFair, we build a base container for running onnx models. The inference examples as shown in Listing 2 and Listing 3.

Run AI project via HPCFair

Different from inference AI artefatcs, which dealing with given inputs, an AI projects may involves data processing, training, fine-tuning, transferring on scaled datasets. HPCFair built a running virtual environments for AI projects by containerization. To run target AI model fetched from HPCFair, users simply provide a configuration file (As shown in Listing 4). Running a AI artefacts rely on diverse and complex environment dependencies, running AI models requires considerable efforts to satisfy both hardware and software requirements. HPCFair provides high-level APIs for users build AI artefacts to their task in one line codes (Line 7 in Listing 3).

```
general_args:
  task: "container"
  backend: "mlcube"

device_args:
  device: 'gpu'

task_args:
  work_dir: "project_dir"
  build_file: "path_to_build_file"
  build_tag: "image_name"
  volume: "/app"
out_args:
  export_file: "out.txt"
```

Listing 4: Configuration for running AI project

Conclusion

In conclusion, we proposed a novel model knowledge management system - HPCFair, which enables AI artefacts Findable, Accessible, Interoperable, and Reproducible (FAIR) principles. HPCFair provides users high-level APIs and friendly interactive interface to fetch, reproduce and retrieve AI artefacts. Most importantly, HPCFair greatly saves the labor cost for scientists to customize AI artefacts to their tasks.

Acknowledgment

This research was funded in part by and used resources at the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

Bai, J.; Lu, F.; Zhang, K.; et al. 2019. ONNX: Open Neural Network Exchange. https://github.com/onnx/onnx.

Chard, R.; Li, Z.; Chard, K.; Ward, L.; Babuji, Y.; Woodard, A.; Tuecke, S.; Blaiszik, B.; Franklin, M. J.; and Foster, I. 2019. DLHub: Model and data serving for science. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 283–292. IEEE.

Fursin, G. 2021. Collective knowledge: organizing research projects as a database of reusable components and portable workflows with common interfaces. *Philosophical Transactions of the Royal Society A*, 379(2197): 20200211.

Kahng, M.; Fang, D.; and Chau, D. H. P. 2016. Visual Exploration of Machine Learning Results Using Data Cube Analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, HILDA '16. New York, NY, USA: Association for Computing Machinery. ISBN 9781450342070.

Kurtzer, G. M.; Sochat, V.; and Bauer, M. W. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE*, 12(5): e0177459.

Liao, C.; Lin, P.-H.; Verma, G.; Vanderbruggen, T.; Emani, M.; Nan, Z.; and Shen, X. 2021. HPC Ontology: Towards a Unified Ontology for Managing Training Datasets and AI Models for High-Performance Computing. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, 69–80. IEEE.

Merkel, D. 2014. Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239): 2.

Nan, Z.; Guan, H.; Shen, X.; and Liao, C. 2021. Deep nlp-based co-evolvement for synthesizing code analysis from natural language. In *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*, 141–152.

Verma, G.; Emani, M.; Liao, C.; Lin, P.-H.; Vanderbruggen, T.; Shen, X.; and Chapman, B. 2021. HPCFAIR: Enabling FAIR AI for HPC Applications. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, 58–68. IEEE.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.