Final Report:
HPC-FAIR: A Framework Managing Data and AI Models for Analyzing and Optimizing Scientific Applications

Principal Investigator:
Chunhua Liao
Lawrence Livermore National Laboratory

Co-Investigators:
Xipeng Shen, North Carolina State University
Murali Emani, Argonne National Laboratory
Tristan Vanderbruggen, Lawrence Livermore National Laboratory
Pei-Hung Lin, Lawrence Livermore National Laboratory

Funding Agency: Office of Science, Department of Energy
Grant Number: DE-SC0021293

September, 2024

Prepared by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-TR-869537), the Argonne Leadership Computing Facility under Contract DE-AC02-06CH11357, and the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Program under Award Number DE-SC0021293.

Table of Contents

1 Executive Summary

The increasing reliance on machine learning (ML) to analyze and optimize large-scale scientific applications on supercomputers faces a significant bottleneck: the lack of readily available, high-quality training datasets and the difficulty in reusing existing AI models. This project was motivated by the urgent need to address the "FAIR" principles (Findability, Accessibility, Interoperability, Reusability) for both training datasets and AI models in the high-performance computing (HPC) domain.

The project developed HPC-FAIR, a high-performance computing data management framework designed to centralize HPC-related datasets and AI models within a unified hub. To ensure interoperability, the framework established a standardized representation and vocabulary (ontology) for both data and models. HPC-FAIR also implemented automated workflows to streamline data processing, model access, and benchmarking. Additionally, the project focused on optimizing data harnessing efficiency through advanced techniques like deep reuse and compression-based analytics.

Specifically, the project achieved the following objectives:

- Collected and generated a representative set of training datasets and AI models, particularly in performance modeling and optimization for heterogeneous architectures.
- Augmented and correlated datasets through code pattern analysis, cross-layer information mapping, and data annotation using ontologies.
- Represented HPC data and AI models using a standardized JSON-LD format (Data IR), built on top of a comprehensive HPC Ontology.
- Provided access to the data and models through user-friendly interfaces (web forms, RESTful APIs, SPARQL queries), large language models, and automated benchmarking capabilities.
- Synthesized burden-free data processing workflows based on user queries, with optimizations for reuse, caching, and compression.

The project successfully addressed the critical need for FAIR datasets and AI models in high-performance computing (HPC). Key research highlights include:

- Recognizing the scarcity of specialized HPC training data, we generated curated datasets tailored to specific research areas, such as code translation [20] and data race detection [5, 7], significantly improving the accuracy of large language models in these domains.

- To address interoperability challenges, we designed a standardized JSON-LD-based representation for data and models, informed by a carefully crafted HPC ontology [21]. We enabled RESTful APIs and SPARQL queries to provide seamless access to these resources, while automated benchmarking capabilities streamlined model evaluation and comparison.
- We developed HPCFAIR [36], a comprehensive prototype framework that serves as a central hub for HPC-related datasets, AI models, and associated research components.
- To optimize the research workflow, we created a workflow synthesizer [25, 24] capable of generating data processing pipelines based on user queries, incorporating optimizations for data reuse, caching, and compression.
- The project's impact extended beyond the framework itself: we established a practical methodology [22] for quantitatively assessing and improving FAIRness, providing actionable recommendations for researchers.
- Furthermore, we explored the potential of large language models [6, 10], demonstrating their ability to enhance data access and understanding in complex HPC scenarios.

This multifaceted approach has yielded a robust and impactful framework, poised to accelerate AI research and development in the HPC community and beyond.

2 Methodology and Results

This project employed a multi-pronged approach to achieve its goals, integrating advanced techniques from computer science, machine learning, and ontology engineering. The project is structured into several integral research thrusts including data collection and generation, data representation, data access, and workflow synthesis and optimization. We elaborate on the methods and results of each of these thrusts in this section.

2.1 Data Collection and Generation

A critical foundation for our HPC-FAIR framework is the availability of high-quality, diverse training datasets and AI models for scientific applications. This section highlights our accomplishments in data collection, generation, and augmentation, which are crucial steps in ensuring the FAIRness of our framework.

We addressed the scarcity of specialized HPC training data by creating dedicated datasets tailored to key research areas. For instance, the work in [20] presents a novel dataset designed specifically for training models that translate between OpenMP Fortran and C++ code. This dataset, sourced from diverse OpenMP benchmarks and rigorously refined through

code similarity tests, significantly improved the translation accuracy of large language models, achieving a remarkable 5.1x CodeBLEU score increase for models without prior coding knowledge.

We recognized the need for a broader perspective on data collection and generation, encompassing the entire machine learning pipeline for programming language processing (PLP). The work in [14] analyzed existing research in ML-based PLP, identifying and characterizing reusable components such as datasets, models, and tools. Figure 1 shows the taxonomy of tasks based on CodeXGLUE's categorization.

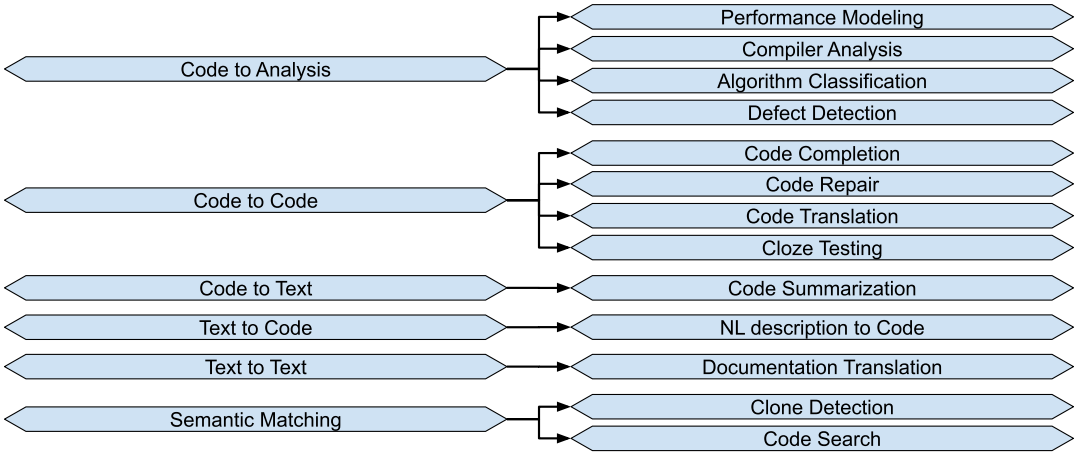


Figure 1: Our taxonomy of HPC tasks related to Programming Language Processing

As shown in Table 1, we have also collected representative tasks in HPC and the corresponding models and datasets. This research lays the groundwork for a comprehensive repository that improves the FAIRness of ML components for PLP, making them more readily accessible and reusable for new researchers and developers.

Understanding the performance implications of different programming choices is crucial in HPC. The research presented in [39] focused on generating performance data related to GPU unified memory usage. Through profiling and machine learning, we developed XUnified, a framework that guides optimal memory advice choices for various applications at runtime, achieving 94% prediction accuracy and significant kernel execution time reduction. This highlights our ability to generate data relevant to specific performance optimization tasks.

Beyond creating new datasets, we also developed methods to augment existing ones, ensuring comprehensiveness and avoiding bias. In [8], we explored transformer-based similarity analysis for the DataRaceBench suite. By adapting the CodeBERT model, we identified similar code patterns within the benchmark, aiding in the detection of duplicated kernels and guiding the addition of new ones. This work demonstrates the potential of deep learning for automated dataset augmentation and analysis.

These efforts in data collection, generation, and augmentation represent a significant step

Tasks	Models													Datasets																									
	NCC	ProgramML	SynCoBERT	PLABART	CodeT5	Code-MVP	TreeBERT	ContraCode	GraphCodeBERT	CoTexT	CodeBERT	CodeXGlue	CodeSearchNet[17]	DeepTune OpenCL	PACT	POJ-104[23]	BigCloneBench[32]	Defects4J[19]	Devign	Wang Java Datasets	LLVM IR	DeepTyper	PY150[29]	Github Java Corpus[2]	Tufano's dataset[34]	Nguyen's dataset	CodeTrans	AdvTest	CONCODE[18]	CosQA	CoNaLa	DeepCom	Python8000
Performance Modeling	✓	✓												✓	✓																							
Algorithm Classification	✓	✓														✓			✓																			
Defect Detection			✓	✓	✓	✓							✓			✓	✓	✓		✓																		
Compiler Analyses		✓																			✓	✓																
Code Completion							✓	✓					✓										✓	✓														
Code Repair					✓				✓	✓															✓													
Code Translation			✓	✓	✓				✓																✓	✓	✓											
Cloze Testing											✓		✓																									
Text-to-Code Generation			✓		✓	✓			✓	✓			✓															✓	✓	✓	✓						
Code Summarization				✓	✓		✓			✓	✓		✓																				✓				
Document Translation												✓																									
Code Search											✓		✓																									
Clone Detection					✓	✓		✓	✓								✓																					✓

Table 1: Program Language Processing (PLP) tasks and suitable models and datasets

towards building a robust and FAIR data foundation for HPC-FAIR. By combining curated datasets, automated analysis, and performance-focused data generation, we are paving the way for more effective and impactful AI-driven optimizations in the HPC domain.

2.2 Data Representation

One of the main goals of the FAIR principles is to achieve machine-actionability in order to process the large amount of scholar data generated daily. This means that data management systems should provide rich and standard information such as the type of digital objects, their formats, licensing, and appropriate operations on them. To be practical, both contextual metadata surrounding a digital object and the content of the digital object should use controlled vocabularies, which in turn are associated with controlled semantics. As a result, several refined FAIR principles shown in Table 2 explicitly mention vocabularies (I2), knowledge representation (I1), and community standards (R1.3) to improve machine-actionability.

The centerpiece of our project is the use of ontology to represent variable semantics associated with data, including both datasets and AI models. Ontologies can provide the much-needed controlled vocabularies, knowledge representation, and standards to implement the FAIR data principles. Ontology [35] is a concept originating in Philosophy, referring to the study of the nature of being, as well as the basic categories of them and their relations. In recent decades, it has become a formal way to explicitly represent knowledge in a domain. An ontology [31, 35] is a formal specification for explicitly representing knowledge about types, properties, and interrelationships of the entities in a domain. It provides a common vocabulary to represent and share domain concepts. Compared to tree-like taxonomy solely modeling the generalization-specialization relation, an ontology can form a much more complex graph with edges to model any kinds of relationships between entities (represented as graph nodes). Such graphs are often called knowledge graphs in many communities.

Table 2: FAIR Principles proposed by FORCE11 [37]

F: To be Findable	
F1	(Meta)data are assigned a globally unique and persistent identifier
F2	Data are described with rich metadata
F3	Metadata clearly and explicitly include the identifier of the data they describe
F4	(Meta)data are registered or indexed in a searchable resource
A: To be Accessible	
A1	(Meta)data are retrievable by their identifier using a standardised communications protocol
A1.1	The protocol is open, free, and universally implementable
A1.2	The protocol allows for an authentication and authorisation procedure, where necessary
A2	Metadata are accessible, even when the data are no longer available
I: To be Interoperable	
I1	(Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation
I2	(Meta)data use vocabularies that follow FAIR principles
I3	(Meta)data include qualified references to other (meta)data
R: To be Reusable	
R1	(Meta)data are richly described with a plurality of accurate and relevant attributes
R1.1	(Meta)data are released with a clear and accessible data usage license
R1.2	(Meta)data are associated with detailed provenance
R1.3	(Meta)data meet domain-relevant community standards

We have designed the HPC ontology [21] to facilitate data representation for FAIR principles. A basic scenario described by the HPC ontology is that *some people who are members of a project used some software and hardware to conduct some experiments, which in turn used some input data to generate training datasets or AI models*. The semantics can be mapped to three high-level concepts, including Agent, Activity, and Artifact. Essentially, *some Agent conducted some Activity which used some Artifact as input and generated some other Artifact*.

As shown in Figure 2, we follow a common naming convention when defining vocabularies in the HPC ontology: Singular nouns in CamelCase are used to indicate a Class. Multiword names are written without any spaces but with each word written in uppercase. Relationship (or Property) names start with lowercase letters. For example, *hpc:Project* means a class while *hpc:project* indicates a property that links some data with its associated project.

Dashed arrows in the figure indicate the *isA* relation between a subclass and its superclass. For instance, both training datasets and AI models are kind of data in this context. Solid line arrows indicate other relationships. Some arrows have only a single label to denote a single relationship. To simplify the diagram, inverse relationships are combined in a single arrow with a pair of relationship labels. For example, the label of the arrow between Person

and Project includes both *hpc:memberOf* and *hpc:member*. Not all edges are shown in the figure to avoid a cluttered figure.

We introduce the concept of Activity, which connects to many other concepts through properties such as *hpc:used*, *hpc:generated*, *hpc:wasAssociatedWithSoftware*, *hpc:usedWorkflow*, *hpc:wasConductedBy*, and so on. An activity is something that occurs over a period of time. An activity could happen after another one, linked using *hpc:wasPrecededBy*. Experiment is a subclass of Activity to represent HPC experiments. We define Data as a subclass of Artifact and further categorize it into Workflow, Training Dataset, and AI model. We have found that in practice, any combinations of mixed scripts, datasets, and AI model files are shared and reused. They are just vaguely categorized as Data currently. In the HPC software analysis and optimization domain, software itself is also a kind of artifact that can be used as input or output data. For example, many compilers, tools and workflows process software as input data to generate program analysis information. So there is a property link (*isA*) between Software and Data also.

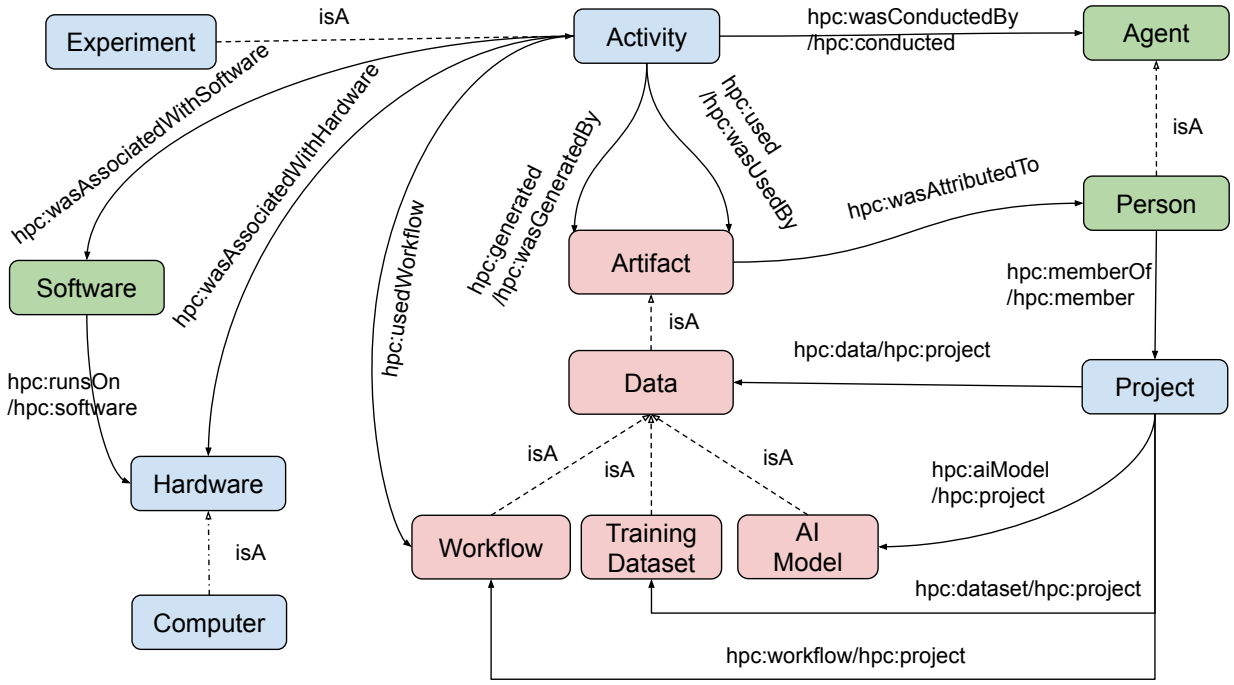


Figure 2: Major High-level Concepts and Relationships of the HPC Ontology

The HPC ontology enables us to describe semantics associated with datasets and AI models. Since an ontology can be queried using a standard RDF query language named SPARQL, we can also build various queries to find designed datasets or models for a given problem in HPC.

We anticipate some typical questions from a HPC user may include the following:

- Q1: What are the ids of datasets from a research project named "Xplacer"?
- Q2: What are the names of AI models available for a supercomputer named "lassen"?
- Q3: What AI models are available for machines with GPUs named "Nvidia V100"?
- Q4: What datasets of projects funded by NSF are available for building Performance Prediction models?
- Q5: What workflows are available for generating AI models guiding "Heterogeneous Mapping" of a benchmark (e.g. the NAS Parallel Benchmark or NPB) running on a machine using AMD GPUs?

We populated a Blazegraph RDF database with information encoding a few example datasets and AI models found in the literature, assuring diversified datasets and models across widely researched domains. We gathered data from projects like XPlacer [38] that come with the workflows defining dataset collection and offer models to perform experiments. Further, there were scenarios when researchers publish their results as a package or complete tool. The ProGraML [9] and MLGO [33] are such considered examples. Additionally, we included literature from HPC Energy Research [28] providing datasets or models alone. We also collected some datasets and refined models hosted at Kaggle. They are related to the GPU Kernel Performance [3, 27, 30].

Listing 1 shows the SPARQL queries sent to the Blazegraph database and the corresponding answers obtained, for Q1 through Q3. Listing 2 shows queries and results for Q3 and Q4. The results show that HPC Ontology is complete to support the concepts and properties expressed in these queries which in turn obtain desired results.

Listing 1: SPARQL queries to answer competency questions 1-3

```
PREFIX hpc: <https://example.org/HPC-Ontology#>
# Query for Q1: dataset ids of a project
#-------------------------------------------------------
SELECT ?ds
WHERE { ?pid rdf:type hpc:Project .
        ?pid hpc:name "Xplacer" .
        ?pid hpc:dataset ?ds }

# Results: IDs of datasets
# <http://example.org/dataset/DA000001>
# <http://example.org/dataset/DA000002>
# <http://example.org/dataset/DA000003>
# <http://example.org/dataset/DA000004>

# Query for Q2: AI models' names of a supercomputer
#-------------------------------------------------------
SELECT ?model_name
WHERE { ?model_id rdf:type hpc:AIModel .
        ?model_id hpc:targetMachine
            <http://example.org/cluster/lassen> .
        ?model_id hpc:name ?model_name }

# Results : names of AI models
# decisionTree.onnx
# randomForest.onnx

```

```
# Query for Q3: machines with NVidia V100 and their models
#-------------------------------------------------------
SELECT ?machine ?model_id
  WHERE { ?gpu rdf:type hpc:GPU .
          ?gpu hpc:name "Nvidia V100".
            ?machine hpc:coprocessorModel ?gpu .
            ?model_id hpc:targetMachine ?machine .
            ?model_id rdf:type hpc:AIModel .
            ?model_id hpc:name ?model_name  }

# Results: machine names and AI model IDs
<http://example.org/cluster/lassen> <http://example.org/AIModel/MO000001>
<http://example.org/cluster/lassen> <http://example.org/AIModel/MO000002>
```

Listing 2: SPARQL queries to answer competency questions 4-5

```
PREFIX hpc: <https://example.org/HPC-Ontology#>

# Query for Q4: NSF project's datasets for building performance prediction models
#-------------------------------------------------------
SELECT ?project_id ?ds_id
WHERE {
        ?project_id rdf:type hpc:Project .
        ?project_id hpc:fundedBy "the National Science Foundation" .
        ?ds_id hpc:project ?project_id .
        ?ds_id rdf:type hpc:Dataset .
        ?ds_id hpc:subject "Performance Prediction" }

# Results: project IDs and dataset IDs
<http://example.org/project/PR000003> <http://example.org/dataset/DA000007>
<http://example.org/project/PR000003> <http://example.org/dataset/DA000008>
<http://example.org/project/PR000003> <http://example.org/dataset/DA000009>

# Query of Q5: workflow generating AI models for NPB's heterogeneous mapping on AMD GPU
SELECT ?pid ?pname
WHERE {
        ?pid rdf:type hpc:Workflow .
        ?pid hpc:subject "Heterogeneous Mapping" .
        ?pid hpc:name ?pname .

        ?pid hpc:targetMachine ?machine_id .
        ?machine_id hpc:coProcessor ?gpu_id .
        ?gpu_id hpc:vendor "AMD" .

        ?pid hpc:targetApplication ?app_id .
        ?app_id hpc:name "NPB" .
      }

# Results
<http://example.org/workflow/WF000020>  OpenCL Heterogeneous Mapping
```

2.3 Data Access

A core objective of the HPC-FAIR project is to enable seamless access to HPC-related datasets and AI models, adhering to the FAIR principles. We have made significant progress towards this goal, starting with the initial design of a modular framework, followed by a systematic methodology for FAIRness improvement, and culminating in the exploration of large language models for enhanced data access.

2.3.1 HPCFAIR: a Modular Framework

Our initial efforts focused on designing HPCFAIR [36], a modular and extensible framework to enable the FAIR principles for AI models used in HPC applications. The framework provides a structured approach for users to search, load, save, and reuse models within their codes, addressing the challenge of duplicated efforts and promoting the reuse of existing resources. This initial design laid the foundation for a more systematic approach to FAIRness improvement.

As shown in Figure 3, HPCFair has a front-end connected to several components implementing tags-based search, user notification, load and store of models and datasets. It also contains a supportive component (HPC Ontology) to provide metadata and an advanced component to automatically synthesize workflows. These two components were developed under the research thrusts of data representation (Section 2.2) and workflow synthesis (Section 2.4).

From our detailed analysis of existing state-of-the-art frameworks, we observed that `MLCube`, in its pre-alpha stage of development, offers a perspective that is easily extensible and obeys the "plug-and-play" philosophy. Considering `MLCube` as a basis, we have further implemented enhancements to bridge gaps in achieving FAIR AI for HPC.

We have developed our framework as a Python library for a lightweight implementation. We will extend it to support other languages such as C++, Java, etc. in the future. Developers are provided with CLI support to discover and load the required components. We store detailed information about each component in Github repositories in the JSON-LD format. It helps us store a data object and associated files, thus keeping the relationship among them intact. Also, the JSON-LD format allows avenues to be converted into a more efficient search data structure which is our future direction. The requested information is provided to the developers in the dictionary format (key: value) that is easy to comprehend.

We employ a multifaceted evaluation approach to demonstrate the effectiveness of the proposed framework in adhering to the FAIR principles and enabling their benefits in scientific machine learning applications.

The evaluation focuses on five key areas:

- Support for DNN models: The framework's ability to handle deep neural networks is tested by replicating an MNIST classification experiment from MLCube, demonstrating seamless model loading, training, and inference with custom hyperparameters.
- Support for ML Libraries: HPCFAIR is evaluated for its capacity to integrate machine learning libraries as reusable components. This is tested by applying linear and logistic regression algorithms on a GPU runtime dataset, showcasing the framework's support for pipeline development using diverse ML components.
- Reproducibility of Published Research: HPCFAIR's contribution to reproducibility

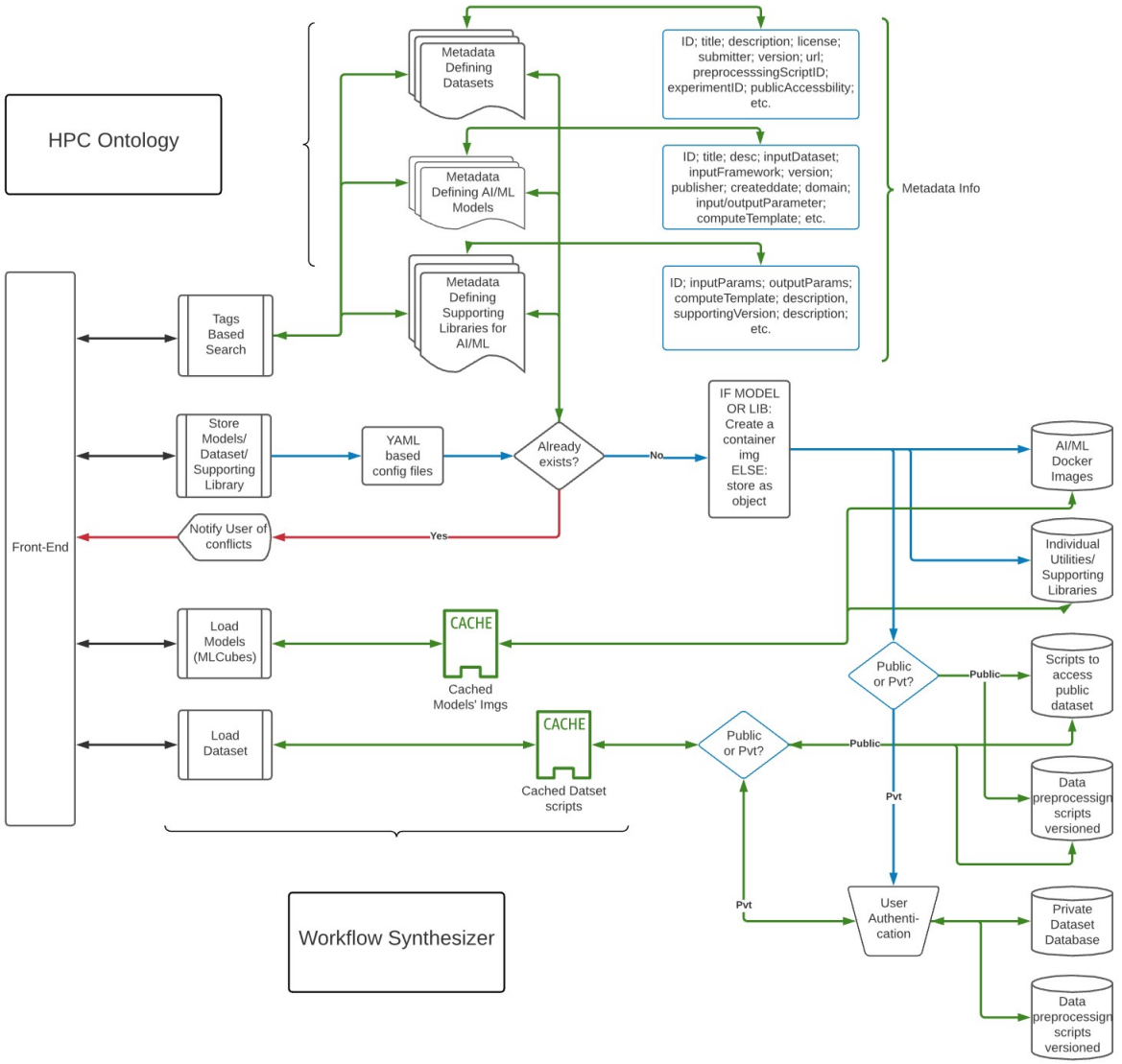


Figure 3: HPCFair: The Proposed Architecture

is demonstrated by replicating the results of a PACT'17 Best Paper Award-winning submission (DeepTune), highlighting the ease of reconstructing the experimental environment and validating the findings.

- Support for Workflows: The framework's ability to manage complex workflows is assessed by reproducing results from the XPlacer research. HPCFAIR successfully packaged the workflow steps and associated scripts, demonstrating its potential for managing multi-stage research processes.

- Support for Design Space Exploration: HPCFAIR is evaluated for its ability to facilitate design space exploration by enabling the easy swapping and comparison of different frameworks (TensorFlow Lite, TensorRT) and optimization engines for a ResNet50 in-

ference task, showcasing its support for rapid prototyping and configuration evaluation.

The evaluation demonstrates that HPCFAIR successfully achieves the following: 1) Findability: Through enriched metadata and a tags-based search mechanism, users can easily locate relevant datasets, models, and other components. 2) Accessibility: The framework provides standardized APIs and a communication protocol for accessing and retrieving data objects, enhancing interoperability. 3) Interoperability: HPCFAIR supports ONNX format, allowing model conversion between different frameworks. Additionally, metadata is encoded using JSON-LD, a formal language that facilitates data exchange and integration. 4) Reusability: The framework encourages the reuse of existing components by providing mechanisms for packaging, storing, and retrieving data objects, including models, datasets, scripts, and workflows.

Overall, the evaluation provides strong evidence that HPCFAIR effectively addresses the challenges of FAIRness in the HPC domain. It enables seamless access to and reuse of research components, promotes reproducibility, and facilitates efficient pipeline development and design space exploration, ultimately accelerating scientific discovery in HPC and machine learning applications.

2.3.2 FAIRification Methodology and a Case Study

Building upon the HPCFAIR framework, we developed a methodology to systematically assess and improve the FAIRness of HPC datasets and machine learning models [22]. This methodology involves a quantitative assessment based on the FAIR principles (shown in Table 3) and fairification workflow provided by the GO FAIR initiative. We developed a set of actionable recommendations to address common issues related to persistent identifiers, metadata descriptions, licenses, and provenance information.

Together with the FAIRification framework, the GO FAIR initiative also provides a FAIRification process aiming at addressing the conversion of raw datasets into FAIR datasets. These 7 steps and the workflow are presented in Figure 4. This FAIRification process emphasizes on FAIRification for both data and metadata, including seven steps: (1) retrieve non-FAIR data, (2) analyze the retrieved data, (3) define the semantic model, (4) make data linkable, (5) assign license, (6) define metadata for the dataset, and (7) deploy/publish FAIR data resource. The process is proposed for general purposes without considering the specific requirements and needs from the individual community. Challenges may arise when applying the process for data coming from a specific domain. For instance, without a standardized data format in the HPC community, the analyses used for the retrieved data vary based on the selected custom data formats. The development of an HPC semantic model needs iterative reviews and revisions to ensure consistency in providing accurate and unambiguous meaning of entities and relations in the HPC domain.

We demonstrated the effectiveness of our methodology through a case study using the Xplacer datasets and AI models [39], achieving a substantial increase in FAIRness score from 19.1%

Table 3: FAIR Guiding Principles: maturity indicators and assessment metrics

P[1]	Indicator	Description	S[2]
F1	RDA-F1-01M	Metadata is identified by a persistent identifier	4
	RDA-F1-01D * FsF-F1-02D	Data is identified by a persistent identifier	
	RDA-F1-02M	Metadata is identified by a globally unique identifier	
	RDA-F1-02D * FsF-F1-01D	Data is identified by a globally unique identifier	
F2	RDA-F2-01M	Rich metadata is provided to allow discovery	2
	FsF-F2-01M	Metadata includes descriptive core elements to support data findability	
F3	RDA-F3-01M * FsF-F3-01M	Metadata includes the identifier for the data	1
F4	RDA-F4-01M * FsF-F4-01M	Metadata is offered in such a way that it can be harvested and indexed	1
A1	RDA-A1-01M	Metadata contains information to enable the user to get access to the data	9
	RDA-A1-02M	Metadata can be accessed manually	
	RDA-A1-02D	Data can be accessed manually	
	RDA-A1-03M	Metadata identifier resolves to a metadata record or digital object	
	RDA-A1-03D	Data identifier resolves to a metadata record or digital object	
	RDA-A1-04M * FsF-A1-02M	Metadata is accessed through standardised protocol	
	RDA-A1-04D * FsF-A1-03D	Data is accessed through standardized protocol	
	RDA-A1-05D	Data can be accessed automatically	
	FsF-A1-01M	Metadata contains access level and access conditions of the data	
A1.1	RDA-A1.1-01M	Metadata is accessible through a free access protocol	2
	RDA-A1.1-01D	Data is accessible through a free access protocol	
A1.2	RDA-A1.2-01D	Data is accessible through an access protocol that supports authentication and authorization	1
A2	RDA-A2-01M * FsF-A2-01M	Metadata is guaranteed to remain available after data is no longer available	1
I1	RDA-I1-01M	Metadata uses knowledge representation expressed in standardized format	6
	RDA-I1-01D	(Data uses knowledge representation expressed in standardized format	
	RDA-I1-02M	Metadata uses machine-understandable knowledge representation	
	RDA-I1-02D	Data uses machine-understandable knowledge representation	
	FsF-I1-01M	Metadata is represented using a formal knowledge representation language	
	FsF-I1-02M	Metadata uses semantic resources	
I2	RDA-I2-01M	Metadata uses FAIR-compliant vocabularies	2
	RDA-I2-01D	Data uses FAIR-compliant vocabularies	
I3	RDA-I3-01M * FsF-I3-01M	Metadata includes references to other (meta)data	6
	RDA-I3-01D	Data includes references to other (meta)data	
	RDA-I3-02M	Metadata includes references to other data	
	RDA-I3-02D	Data includes references to other data	
	RDA-I3-03M	Metadata includes qualified references to other metadata	
	RDA-I3-04M	Metadata include qualified references to other data	
R1	RDA-R1-01M	Plurality of accurate and relevant attributes are provided to allow reuse	2
	FsF-R1-01MD	Metadata specifies the content of the data	
R1.1	RDA-R1.1-01M	Metadata includes information about the license under which the data can be reused	3
	RDA-R1.1-02M	Metadata refers to a standard reuse license	
	RDA-R1.1-03M * FsF-R1.1-01M	Metadata refers to a machine-understandable reuse license	
R1.2	RDA-R1.2-01M	Metadata includes provenance information according to community-specific standards	3
	RDA-R1.2-02M	Metadata includes provenance information according to a cross-community language	
	FsF-R1.2-01M	Metadata includes provenance information about data creation or generation	
R1.3	RDA-R1.3-01M * FsF-R1.3-01M	Metadata complies with a community standard	4
	RDA-R1.3-01D	Data complies with a community standard	
	RDA-R1.3-02M	Metadata is expressed in compliance with a machine-understandable community standard	
	RDA-R1.3-02D * FsF-R1.3-02D	Data is in compliance with a machine-understandable community standard	

[1] FAIR Guiding Principle ID
[2] Max score allocated to each sub-principle
* Both RDA FAIRness indicator and F-UJI metric represent the same evaluation

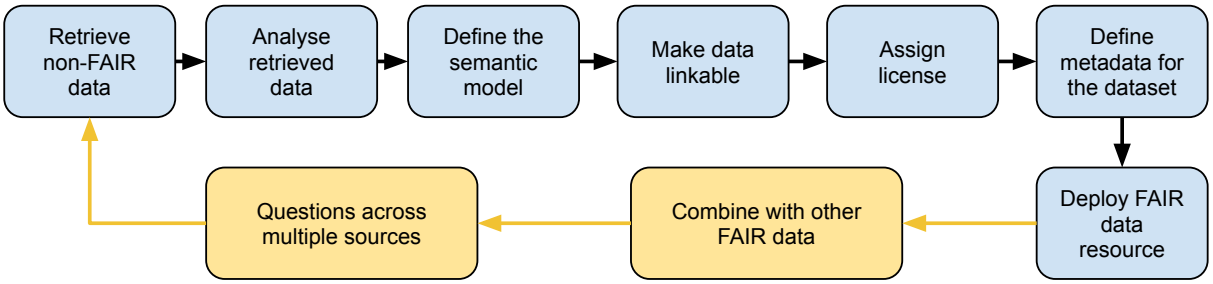
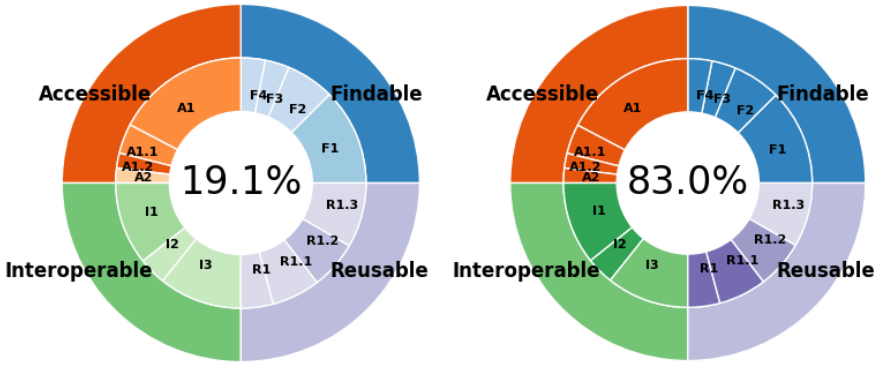


Figure 4: FAIRification Process proposed by GO FAIR initiative.

to 83.0% (shown in Figure 5). The final assessment shows that the use of the HPC ontology [21] significantly improves the metadata support for the XPlacer dataset, resulting in full fulfillment for all FAIRness indicators for 'F and A' FAIR principles (reaching 100% for both). This work provides a practical guide for researchers seeking to enhance the FAIRness of their HPC-related resources.



	Init Score	Init %	Final Score	Final %
Findable:	1 of 8	12.5%	8 of 8	100%
Accessible:	6 of 13	46.2%	13 of 13	100%
Interoperable:	1 of 14	7.1%	11 of 14	78.6%
Reusable:	1 of 12	8.3%	7 of 12	58.3%
Total:	9 of 47	19.1%	39 of 47	83.0%

Figure 5: FAIRness of XPlacer Datasets and Models: Before and After

2.3.3 Leveraging Language Language Models

We explored the potential of large language models (LLMs) to further improve data access in the HPC domain. Specifically, we developed HPC-GPT [10], an LLaMA-based model fine-tuned with HPC-specific question-answering (QA) instances. HPC-GPT demonstrated

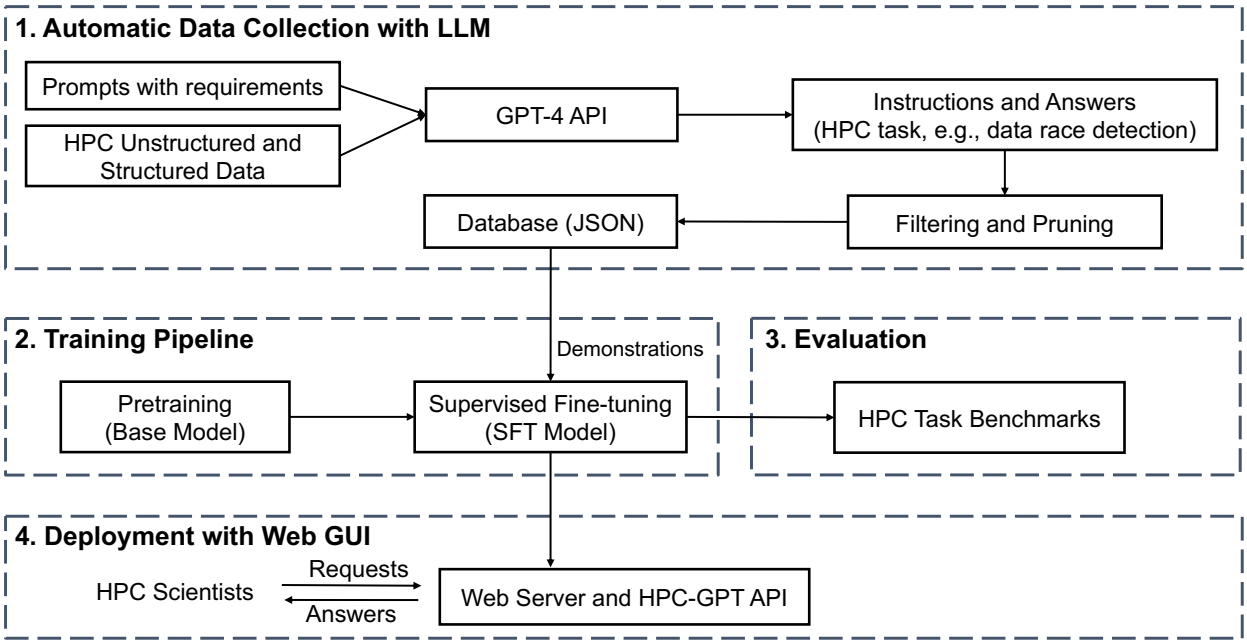


Figure 6: HPC-GPT Workflow

promising results in managing AI models and datasets for HPC, showing comparable performance with existing methods. This research highlights the potential of LLMs to provide a more intuitive and user-friendly interface for accessing and utilizing complex HPC data and models.

As shown in Figure 6, the workflow can be divided into four main stages: HPC domain data collection, training, evaluation, and deployment. In the HPC domain data collection stage, we develop a data collection method that automatically gathers the necessary training data for two specific HPC applications. This data curation process ensures that the model is trained on relevant and domain-specific information. Moving on to the training stage, we employ supervised fine-tuning on the open-source LLM using the generated instruction data. Fine-tuning helps the model adapt to the intricacies of HPC-related tasks and enhances its performance in the targeted domain. In the evaluation stage, we rigorously assess the well-trained HPC-GPT model's capabilities on a public data race detection dataset. This evaluation process validates the model's effectiveness and performance in real-world scenarios. Finally, after successful training and evaluation, we deploy the HPC-GPT model to a web server, making it readily available for HPC scientists and researchers to utilize in their work.

This work represents the first open-source LLM fine-tuned using HPC instruction data. This specialized model is specifically designed to excel in HPC tasks. Using automatic data collection with LLMs, we have integrated HPC knowledge into our framework, ensuring it possesses accurate and domain-specific information extracted from both unstructured and structured data. We released of two open-source datasets for the HPC domain: one containing HPC

models and datasets, and the other dedicated to data race detection. These datasets provide valuable resources for researchers in the HPC community, facilitating further advancements in the field. The code and datasets are publicly available.

Our work also contains a training pipeline to fine-tune base models to conduct HPC tasks. Through comparative evaluation against state-of-the-art methods, we demonstrated the model's effectiveness and its potential to outperform existing approaches in HPC-related tasks. Our research has made significant progress in enabling FAIR data access within the HPC domain. We have moved beyond the initial framework design to develop a practical methodology for FAIRness improvement and explored the potential of LLMs for enhancing the user experience. These efforts will facilitate greater collaboration, reproducibility, and knowledge sharing within the HPC community, ultimately accelerating scientific discovery.

2.4 Workflow Synthesis and Optimization

In this part, we explored *NLU-driven approach* for HPC workflow synthesis. This approach is driven by *natural language understanding (NLU)*. It is inspired by how humans develop workflows. Rather than going through thousands of examples as a *data-driven approach* does, a programmer can start developing workflows after she reads through the documentation of the language or API of interest. She may check a few examples, but the number is far less than what a *data-driven approach* usually needs. The key is in understanding the language or API, the central feature NLU-driven NL programming builds on.

More specifically, the NLU-driven approach features (1) deep processing of programmers' intentions and API documents written in natural languages via NLU, and (2) the leverage of the deep understanding rather than training examples for workflow synthesis.

Compared to data-driven approaches, the NLU-driven approach has three appealing properties. First, by avoiding the need for a large number of labeled examples, it saves users the (often heavy) burden in collecting/generating examples, and makes NL programming possible for domains where labeled examples are scarce. Second, built on understanding rather than data, it avoids the bias that training data examples bring to data-driven methods. Finally, as a "white-box" approach, its better interpretability makes the diagnosis of synthesis errors easier to do.

Our exploration leads to *HPC-HISyn* (for "human learning inspired synthesizer" for HPC), the first NLU-driven workflow synthesizing framework for HPC. HPC-HISyn is equipped with several distinctive features.

(1) Deep NL understanding for workflow synthesis. A deep natural language understanding is the key to human learning-inspired workflow synthesis. Although NLP has been used in software maintenance [40, 26, 16, 13, 42, 43, 4, 1, 11, 15], its usage in NL programming is

https://github.com/dingxianzhong/HPC-GPT
https://huggingface.co/datasets/HPC-GPT/HPC

yet to understand. Unlike prior synthesis studies that use shallow NLP just for assistance, HPC-HISyn takes modern NLP as the first-order tool. With it, HPC-HISyn automatically builds up the intermediate representation of input queries, and the knowledge base of the programming APIs, preparing the foundation for synthesis to work on. It uses *NL dependence analysis* to capture the deep relations among the different parts of the input query, and WordNet (a lexical database of English with 117,000 synsets) to capture the semantic associations of English words.

(2) Framework architecture design for cross-domain extensibility. Domain differences are inherent to natural language programming. Different domains have different terminologies, API definitions, grammar, query patterns, and so on. Previous non—data-driven studies are domain-specific; rules and elements specific to the target domain are tightly integrated with the synthesizers, making them difficult to port to other domains. HPC-HISyn strives to ensure cross-domain extensibility in design, in order to handle the large variety of scientific domains in HPC. Drawing on inspirations on how modern compilers deal with domain/language varieties, it creates architecture with the front end producing a unified intermediate representation (*NL dependence graph*) for an arbitrary domain, on which, the back end operates to generate the code in the target API. Neither the front end nor the back end requires changes across domains. The HPC-HISyn design encapsulates domain-specific elements into separate modules equipped with an easy-to-use interface. For a new domain, the developer only needs to use the interfaces to extend those domain-specific modules; no changes are needed for the HISyn framework.

(3) Graph-based algorithm for mapping. Based on the intermediate representation, HISyn employs *grammar graph-based translation* to generate code to materialize a user's intention in the target programming APIs. The novel algorithm first annotates each node in the IR with candidate APIs and then uses path finding on the reverse API grammar graph to identify the appropriate APIs, their order, and assemble them into the final code. The algorithm bridges the gap between the user's intention and the APIs by leveraging both the semantic connections at the natural language level and the syntactical constraints at the API grammar level.

We evaluate HPC-HISyn on the domain of program source code analysis. Although combing with a few examples could potentially help, to examine the potential of pure NLU-based approach, HPC-HISyn is designed to use no examples. Our experimental results show that without any training examples, HPC-HISyn can produce code as accurately as those by a representative *data-driven* NL programming framework trained on hundreds of examples. The study validates the cross-domain portability of the core of HPC-HISyn, and demonstrates the large potential of the NLU-driven approach for NL programming for the use of HPC-FAIR in HPC applications, while at the same time, saving the burden of collecting large numbers of training examples.

In addition to workflow synthesis, we have explored caching and compression for improving the efficiency of the HPC-FAIR platform. We have concentrated on graph compression for the large space cost and processing time of graphs (e.g., knowledge graphs). Many compression

algorithms have been developed to support direct processing on compressed graphs before. However, previous graph compression algorithms do not focus on leveraging redundancy in repeated neighbor sequences, so they do not save the amount of computation for graph analytics. We develop CompressGraph, an efficient rule-based graph analytics engine that leverages data redundancy in graphs to achieve both performance boost and space reduction for common graph applications. CompressGraph has three advantages over previous works. First, the rule-based abstraction of CompressGraph supports the reuse of intermediate results during graph traversal, thus saving time. Second, CompressGraph has intense expressiveness to support a wide range of graph applications. Third, CompressGraph scales well under high parallelism because the context-free rules have few dependencies. Experiments show that CompressGraph provides significant performance and space benefits on both CPUs and GPUs. On evaluating six typical graph applications, CompressGraph can achieve 1.97× speedup on the CPU, while 3.95× speedup on the GPU, compared to the state-of-the-art CPU and GPU methods, respectively. Moreover, CompressGraph can save an average of 71.27% memory savings on CPU and 70.36 on GPU.

3 Publications and Dissemination

This project has generated numerous research outputs, organized below by the key research thrusts outlined in the original proposal:

Data Collection, Generation and Augmentation

1. [20] Lei, B., Ding, C., Chen, L., Lin, P.-H., & Liao, C. (2023). Creating a Dataset Supporting Translation Between OpenMP Fortran and C++ Code. *arXiv preprint arXiv:2307.07686*.

2. [8] Chen, W., Vanderbruggen, T., Lin, P.-H., Liao, C., & Emani, M. (2022). Early Experience with Transformer-based Similarity Analysis for DataRaceBench. In *2022 IEEE/ACM Sixth International Workshop on Software Correctness for HPC Applications (Correctness)* (pp. 45–53). IEEE.

3. [39] Xu, H., Lin, P.-H., Emani, M., Hu, L., & Liao, C. (2022). XUnified: A Framework for Guiding Optimal Use of GPU Unified Memory. *IEEE Access, 10*, 82614–82625. IEEE.

4. [14] Flynn, P., Vanderbruggen, T., Liao, C., Lin, P.-H., Emani, M., & Shen, X. (2022). Finding Reusable Machine Learning Components to Build Programming Language Processing Pipelines. In *European Conference on Software Architecture* (pp. 402–417). Springer International Publishing Cham.

Data Representation

1. [12] Dorta, E., Yan, Y., & Liao, C. (2022). Generating and Analyzing Program Call Graphs using Ontology. In *2022 IEEE/ACM Workshop on Programming and Performance Visualization Tools (ProTools)* (pp. 11–20). IEEE.

2. [21] Liao, C., & Lin, P.-H. (2021). *HPC Ontology Dataset*. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

Data and Model Access

1. [10] Ding, X., Chen, L., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Xie, Z., Cerpa, A., Du, W. (2023). HPC-GPT: Integrating Large Language Model for High-Performance Computing. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (pp. 951–960).

2. [22] Lin, P.-H., Liao, C., Chen, W., Vanderbruggen, T., Emani, M., & Xu, H. (2022). Making Machine Learning Datasets and Models FAIR for HPC: A Methodology and Case Study. In *2022 Fourth International Conference on Transdisciplinary AI (TransAI)* (pp. 128–134). IEEE.

3. [36] Verma, G., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Shen, X., & Chapman, B. (2021). HPCFair: Enabling Fair AI for HPC Applications. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)* (pp. 58–68). IEEE.

Workflow Synthesis and Optimization

1. [25] Nan, Z., Guan, H., Shen, X., & 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* (pp. 141–152).

2. [24] Nan, Z., Dave, M., Shen, X., Liao, C., Vanderbruggen, T., Lin, P.-H., & Emani, M. (2022). Interactive NLU-Powered Ontology-Based Workflow Synthesis for FAIR Support of HPC. In *2022 IEEE/ACM International Workshop on HPC User Support Tools (HUST)* (pp. 29–40). IEEE.

Broader Impacts and Applications

1. [41] Yu, S., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Shen, X., & Jannesari, A. (2022). Towards Seamless Management of AI models in High-performance Computing. *arXiv preprint arXiv:2212.06352*.

2. [6] Chen, L., Lin, P.-H., Vanderbruggen, T., Liao, C., Emani, M., & De Supinski, B. (2023). LM4HPC: Towards Effective Language Model Application in High-Performance Computing. In *International Workshop on OpenMP* (pp. 18–33). Springer Nature Switzerland Cham.

3. [5] Chen, L., Ding, X., Emani, M., Vanderbruggen, T., Lin, P.-H., & Liao, C. (2023). Data Race Detection Using Large Language Models. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis* (pp. 215–223).

4 Workforce Development

This project prioritized the training and development of the next generation of researchers in AI and HPC. The project funded seven student internships across multiple institutions, providing valuable research experience and mentorship opportunities. The following students contributed to the project's success:

- Gaurav Verma: Summer Internship at ANL (2021). Computer Science Ph.D. student at Stony Brook University, New York. His research interest lies in the intersection of systems and machine learning comprising DNN Compilers, Scalable Machine Learning, High-Performance Computing, and Edge Computing.
- Sixing Yu: Summer Internship at ANL (2022). Computer Science Ph.D. student at Iowa States University, Ames, IA. His research interests is to advance state of the art in building reliable and efficient software utilizing AI/analytical methods, and help application developers to utilize the modern AI methods to develop complex software of in-demand areas.
- Winson Chen: Summer Internship at LLNL (2022). Scientific Computing and Applied Mathematics Master student at University of California, Santa Cruz. His research interests are in machine learning, computer vision, and natural language processing.
- Patrick Flynn: Summer Internship at LLNL (2022). Computer Science Master's Student at the University of North Carolina, Charlotte. His interests lie in high performance computing, compilers, language tools, and computer architecture.
- Le Chen: Summer Internship at LLNL (2023). Computer Science Ph.D. student at Iowa States University, Ames, IA. His research interests span across high performance computing, code analysis, and the innovative application of Deep Learning methodologies and Large Language Models to augment compiler efficiency and effectiveness.
- Arushi Sharma: Summer Internship at LLNL (2023). Computer Science Ph.D. student at Iowa State University, Ames, IA. Her research interests are in deep learning for software engineering and high performance computing applications, and improving the interpretability of code-trained language models.

- Xianzhong Ding: Summer Internship at ANL (2023). Computer Science Ph.D. student at University of California, Mered. His research interests are in applied Machine Learning/Deep Learning, deep reinforcement learning, AI/ML in Science.

5 Conclusions and Recommendations

This project has laid a robust foundation for enabling FAIR data and AI models within the high-performance computing (HPC) domain. Through the creation of HPC-FAIR, we have delivered a comprehensive framework and suite of tools and methodologies that empower researchers to access, reuse, and leverage essential data resources, ultimately accelerating the pace of scientific discovery. Our efforts in data collection, augmentation, representation, access, and workflow optimization have demonstrably enhanced the FAIRness of HPC-related research. The project's impact extends beyond the framework itself, as our systematic methodology for FAIRness assessment and improvement provides practical guidance for researchers across disciplines.

While our ontology-based approach has demonstrated its benefits for data interoperability and organization, the manual effort required to create and maintain these ontologies remains a significant bottleneck. The emergence of large language models (LLMs) presents a promising avenue for automating this knowledge capture process. LLMs have shown remarkable capabilities in understanding complex natural language and extracting relevant information from diverse sources.

We strongly recommend further investment to explore the application of LLMs for automating the creation, enrichment, and maintenance of ontologies within HPC-FAIR. Future research should focus on developing robust methods for:

- Domain-Specific Knowledge Extraction: Fine-tuning LLMs to effectively capture and represent knowledge relevant to the HPC domain, including concepts, relationships, and constraints related to software, hardware, datasets, and AI models.
- Ontology Population and Refinement: Leveraging LLMs to automatically populate ontologies with new information extracted from research papers, technical documentation, and community forums, as well as refine existing ontologies to reflect the evolving HPC landscape.
- Query Understanding and Response: Training LLMs to understand complex user queries that may contain subtle constraints related to desired datasets, models, and specific software/hardware configurations, ensuring accurate and relevant responses from the HPC-FAIR framework.

By harnessing the power of LLMs, we can overcome current limitations and unlock the full potential of HPC-FAIR, enabling a truly seamless and intelligent data ecosystem for scientific

discovery. This research has the potential to revolutionize not only how we manage data in HPC but also how we approach knowledge representation and information retrieval across numerous scientific domains.

6 Acknowledgments

This project gratefully acknowledges the generous support of the Office of Science, Department of Energy, under Grant Number DE-SC0021293. The project's success was made possible through the collaborative efforts of researchers from Lawrence Livermore National Laboratory, Argonne National Laboratory, and North Carolina State University. The team deeply appreciates the contributions of each institution and the dedication of the individuals involved.

7 References

References

[1] Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. A survey of machine learning for big code and naturalness. *ACM Computing Surveys (CSUR)*, 51(4):1–37, 2018.

[2] Miltiadis Allamanis and Charles Sutton. Mining Source Code Repositories at Massive Scale using Language Modeling. In *The 10th Working Conference on Mining Software Repositories*, pages 207–216. IEEE, 2013.

[3] Rafael Ballester-Ripoll, Enrique G Paredes, and Renato Pajarola. Sobol tensor trains for global sensitivity analysis. *Reliability Engineering & System Safety*, 183:311–322, 2019.

[4] Oscar Chaparro, Jing Lu, Fiorella Zampetti, Laura Moreno, Massimiliano Di Penta, Andrian Marcus, Gabriele Bavota, and Vincent Ng. Detecting missing information in bug descriptions. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 396–407, 2017.

[5] Le Chen, Xianzhong Ding, Murali Emani, Tristan Vanderbruggen, Pei-Hung Lin, and Chunhua Liao. Data race detection using large language models. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 215–223, 2023.

[6] Le Chen, Pei-Hung Lin, Tristan Vanderbruggen, Chunhua Liao, Murali Emani, and Bronis De Supinski. Lm4hpc: Towards effective language model application in high-

performance computing. In *International Workshop on OpenMP*, pages 18–33. Springer Nature Switzerland Cham, 2023.

[7] Le Chen, Wenhao Wu, Stephen F Siegel, Pei-Hung Lin, and Chunhua Liao. Dataracebench v1. 4.1 and dataracebench-ml v0. 1: Benchmark suites for data race detection. *arXiv preprint arXiv:2308.08473*, 2023.

[8] Winson Chen, Tristan Vanderbruggen, Pei-Hung Lin, Chunhua Liao, and Murali Emani. Early experience with transformer-based similarity analysis for dataracebench. In *2022 IEEE/ACM Sixth International Workshop on Software Correctness for HPC Applications (Correctness)*, pages 45–53. IEEE, 2022.

[9] Chris Cummins, Zacharias Fisches, Tal Ben-Nun, Torsten Hoefler, Michael O'Boyle, and Hugh Leather. ProGraML: A Graph-based Program Representation for Data Flow Analysis and Compiler Optimizations. In *Thirty-eighth International Conference on Machine Learning (ICML)*, 2021.

[10] Xianzhong Ding, Le Chen, Murali Emani, Chunhua Liao, Pei-Hung Lin, Tristan Vanderbruggen, Zhen Xie, Alberto Cerpa, and Wan Du. Hpc-gpt: Integrating large language model for high-performance computing. In *Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, pages 951–960, 2023.

[11] Pradeep Dogga, Karthik Narasimhan, Anirudh Sivaraman, and Ravi Netravali. A system-wide debugging assistant powered by natural language processing. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 171–177, 2019.

[12] Ethan Dorta, Yonghong Yan, and Chunhua Liao. Generating and analyzing program call graphs using ontology. In *2022 IEEE/ACM Workshop on Programming and Performance Visualization Tools (ProTools)*, pages 11–20. IEEE, 2022.

[13] Michael D Ernst. Natural language is a programming language: Applying natural language processing to software development. In *2nd Summit on Advances in Programming Languages (SNAPL 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[14] Patrick Flynn, Tristan Vanderbruggen, Chunhua Liao, Pei-Hung Lin, Murali Emani, and Xipeng Shen. Finding reusable machine learning components to build programming language processing pipelines. In *European Conference on Software Architecture*, pages 402–417. Springer International Publishing Cham, 2022.

[15] Hui Guan, Xipeng Shen, and Hamid Krim. Egeria: a framework for automatic synthesis of hpc advising tools through multi-layered natural language processing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–14, 2017.

[16] Sonia Haiduc, Venera Arnaoudova, Andrian Marcus, and Giuliano Antoniol. The use of text retrieval and natural language processing in software engineering. In *Proceedings of the 38th International Conference on Software Engineering Companion*, pages 898–899, 2016.

[17] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search, 2019.

[18] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Mapping language to code in programmatic context, 2018.

[19] René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: A database of existing faults to enable controlled testing studies for java programs. In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, ISSTA 2014, page 437–440, New York, NY, USA, 2014. Association for Computing Machinery.

[20] Bin Lei, Caiwen Ding, Le Chen, Pei-Hung Lin, and Chunhua Liao. Creating a dataset for high-performance computing code translation using llms: A bridge between openmp fortran and c++. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2023.

[21] Chunhua Liao, Pei-Hung Lin, Gaurav Verma, Tristan Vanderbruggen, Murali Emani, Zifan Nan, and Xipeng Shen. 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)*, pages 69–80. IEEE, 2021.

[22] Pei-Hung Lin, Chunhua Liao, Winson Chen, Tristan Vanderbruggen, Murali Emani, and Hailu Xu. Making machine learning datasets and models fair for hpc: A methodology and case study. In *2022 Fourth International Conference on Transdisciplinary AI (TransAI)*, pages 128–134. IEEE, 2022.

[23] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. Convolutional neural networks over tree structures for programming language processing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1287–1293, 2016.

[24] Zifan Nan, Mithil Dave, Xipeng Shen, Chunhua Liao, Tristan Vanderbruggen, Pei-Hung Lin, and Murali Emani. Interactive nlu-powered ontology-based workflow synthesis for fair support of hpc. In *2022 IEEE/ACM International Workshop on HPC User Support Tools (HUST)*, pages 29–40. IEEE, 2022.

[25] Zifan Nan, Hui Guan, Xipeng Shen, and Chunhua Liao. Deep nlp-based co-evolvement for synthesizing code analysis from natural language. In *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*, pages 141–152, 2021.

[26] Pengyu Nie, Junyi Jessy Li, Sarfraz Khurshid, Raymond Mooney, and Milos Gligoric. Natural language processing and program analysis for supporting todo comments as software evolves. In *Workshops at the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] Cedric Nugteren and Valeriu Codreanu. Cltune: A generic auto-tuner for opencl kernels. In *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, pages 195–202. IEEE, 2015.

[28] Caleb Phillipsy. Hpc energy research, 2016.

[29] Veselin Raychev, Pavol Bielik, and Martin Vechev. Probabilistic model for code with decision trees. *SIGPLAN Not.*, 51(10):731–747, oct 2016.

[30] Rupal Shrivastava. Gpu kernel performance dataset. https://www.kaggle.com, 2020.

[31] Steffen Staab and Rudi Studer. *Handbook on ontologies*. Springer Science & Business Media, 2013.

[32] Jeffrey Svajlenko, Judith F. Islam, Iman Keivanloo, Chanchal K. Roy, and Mohammad Mamun Mia. Towards a big data curated benchmark of inter-project code clones. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 476–480, 2014.

[33] Mircea Trofin, Yundi Qian, Eugene Brevdo, Zinan Lin, Krzysztof Choromanski, and David Li. Mlgo: a machine learning guided compiler optimizations framework. *arXiv preprint arXiv:2101.04808*, 2021.

[34] Michele Tufano, Cody Watson, Gabriele Bavota, Massimiliano Di Penta, Martin White, and Denys Poshyvanyk. An empirical study on learning bug-fixing patches in the wild via neural machine translation. *ACM Trans. Softw. Eng. Methodol.*, 28(4), sep 2019.

[35] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(2):93–136, 1996.

[36] Gaurav Verma, Murali Emani, Chunhua Liao, Pei-Hung Lin, Tristan Vanderbruggen, Xipeng Shen, and Barbara Chapman. HPCFair: Enabling FAIR AI for HPC Applications. In *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pages 58–68. IEEE, 2021.

[37] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[38] Hailu Xu, Murali Emani, Pei-Hung Lin, Liting Hu, and Chunhua Liao. Machine learning guided optimal use of GPU unified memory. In *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)*, pages 64–70. IEEE, 2019.

[39] Hailu Xu, Pei-Hung Lin, Murali Emani, Liting Hu, and Chunhua Liao. Xunified: A framework for guiding optimal use of gpu unified memory. *IEEE Access*, 10:82614–82625, 2022.

[40] Hao Yu, Wing Lam, Long Chen, Ge Li, Tao Xie, and Qianxiang Wang. Neural detection of semantic code clones via tree-based convolution. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pages 70–80. IEEE, 2019.

[41] Sixing Yu, Murali Emani, Chunhua Liao, Pei-Hung Lin, Tristan Vanderbruggen, Xipeng Shen, and Ali Jannesari. Towards seamless management of ai models in high-performance computing. *arXiv preprint arXiv:2212.06352*, 2022.

[42] Juan Zhai, Xiangzhe Xu, Yu Shi, Minxue Pan, Shiqing Ma, Lei Xu, Weifeng Zhang, Lin Tan, and Xiangyu Zhang. Cpc: automatically classifying and propagating natural language comments via program analysis. 2019.

[43] Yu Zhou, Ruihang Gu, Taolue Chen, Zhiqiu Huang, Sebastiano Panichella, and Harald Gall. Analyzing apis documentation and code to detect directive defects. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 27–37. IEEE, 2017.