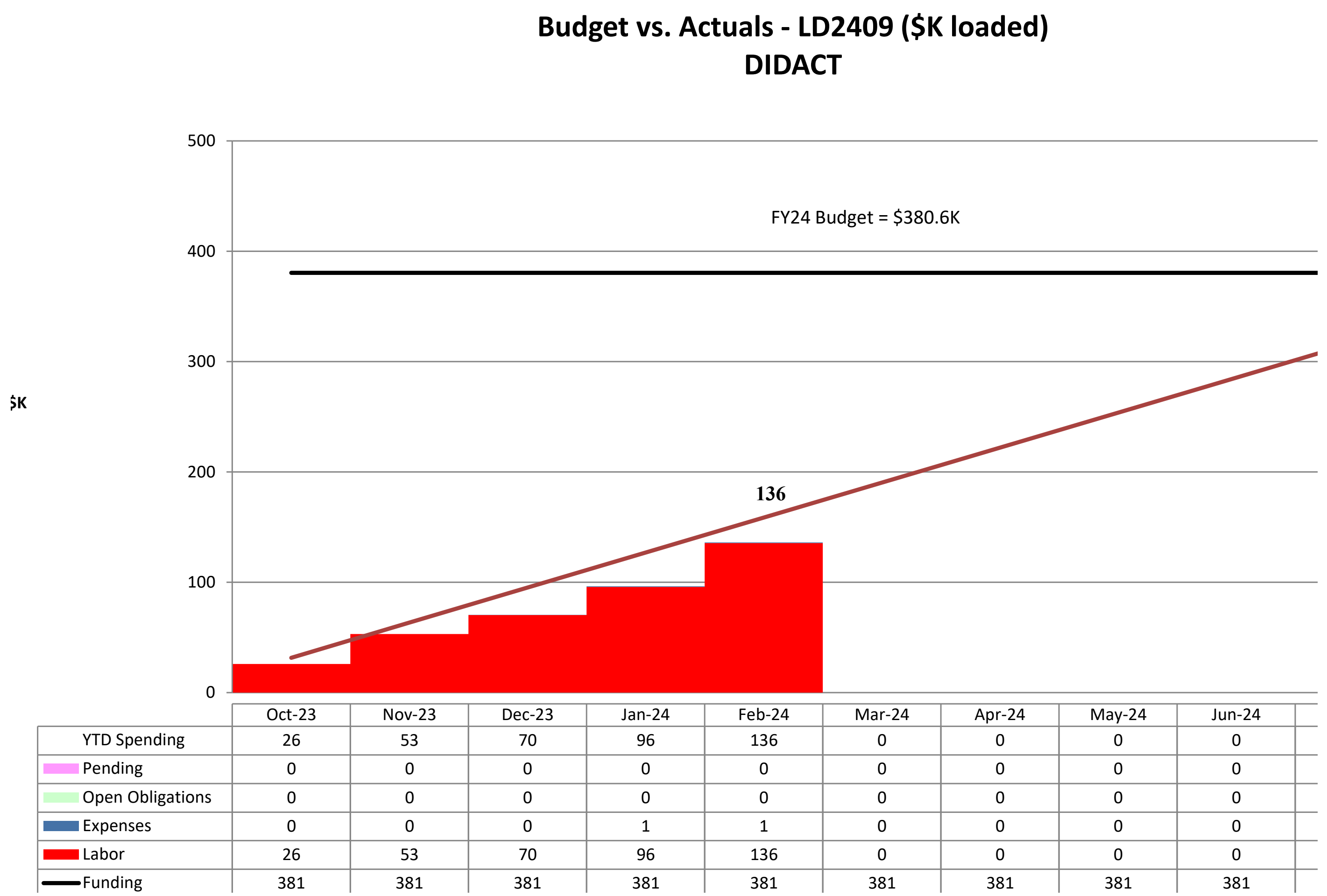
LD2409 – DIDACT Project Financials



	Oct-23	Nov-23	Dec-23	Jan-24	Feb-24	Mar-24	Apr-24	May-24	Jun-24
YTD Spending	26	53	70	96	136	0	0	0	0
🟪 Pending	0	0	0	0	0	0	0	0	0
🟩 Open Obligations	0	0	0	0	0	0	0	0	0
🟦 Expenses	0	0	0	1	1	0	0	0	0
🟥 Labor	26	53	70	96	136	0	0	0	0
— Funding	381	381	381	381	381	381	381	381	381

- Effort trailed planned rate through March
  - Operations contention for labor from other projects (ENP, LQCD)
  - One Staff member was temporarily on a part time schedule.
- Effort Changes Starting in April:
  - Dedicating more Data Science time to ML model development effort. (Additional Postdoc assignment)
  - New technician time for hardware work
  - Increased MLOps framework development time
- Contributors
  - Bryan Hess (PI), Malachi Schram (Co-PI)
  - Ops: Mark Jones (primarily), Laura Hild, Wesley Moore, Stephanie Siebor (new)
  - Data Science: Diana McSpadden, Ahmed Mohammed, Zhenyu Dai (new)

LD2409 – DIDACT Project Progress

- Good Progress on ML Operations
  - Submission to IEEE Special Issue on MLOps: *Enabling MLOps for Continual Learning in Computing Clusters*
- Challenges to modeling
  - Packaging Production Code
  - Testbed's evolving nature
  - coarse sampling rate

Theme: MLOps – Bridging the Gap Between Machine Learning and Operations

Establishing MLOps for Continual Learning in Computing Clusters

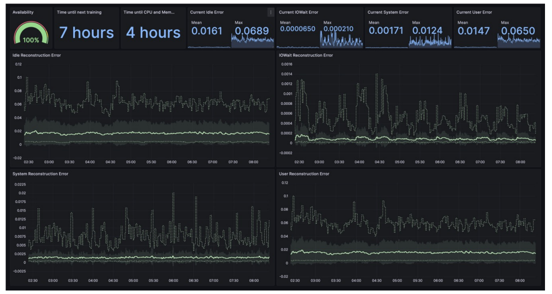
Diana McSpadden, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Mark Jones, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Ahmed Hossam Mohammed, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Bryan Hess, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Malachi Schram, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*

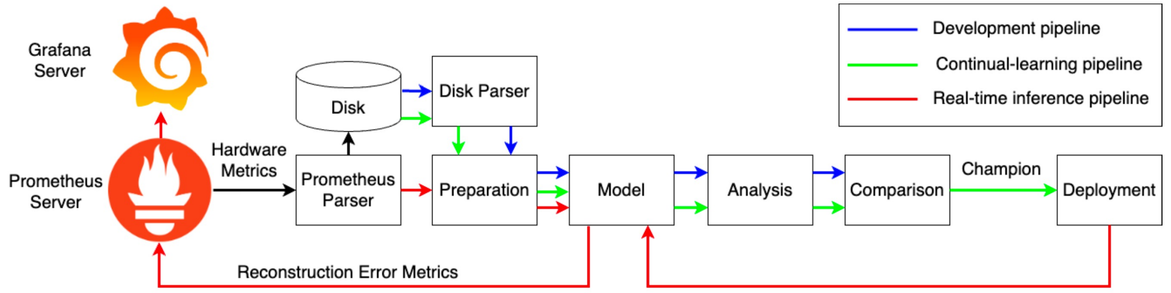
Abstract—In our exploration of the evolving behavior of a computing cluster, we focus on building a continual learning capability. This endeavor is funded through a Laboratory Directed Research and Development project at Jefferson Lab, where it was necessary to establish an MLOps capability within a basic scientific research organization. Here, we describe a composable ML workflow, a custom CGroupV2 exporter, and the implementation of Prometheus, MLFlow, and Grafana. In addition to supporting versioning, monitoring, and comparison, this integrated system also facilitates the delivery of models adapting to the dynamic nature of a computing cluster.

Thomas Jefferson National Accelerator Facility (Jefferson Lab) is a U.S. Department of Energy Office of Science National Laboratory. Scientists worldwide utilize the lab's particle accelerator to probe the basic building blocks of matter and rely on the lab's high-performance computing resources.

In 2022, a Jefferson Lab research and development project was funded for a team from the Computational Science and Technology Division. The project aims to develop automated insight into operational control of a high-performance computing system. [illegible]

erations (MLOps) to support multiple scientific applications and domains. Informally, we call our project and resulting system the Digital Data Center Twin (DIDACT) where an ML-based model would facilitate diagnostic and optimization augmentation tools. The success criteria for the DIDACT project is to demonstrate the ability to correctly diagnose sources of unexpected cluster behavior, such as memory [illegible] tation or I/O over-commitment [illegible]





Milestones and Progress – Half 1

1. ✅ Summary and detail dashboards in Grafana
2. ✅ Continuous learning process with daily updates
3. ⚠️ Human in the loop to vet out-of-distribution events
4. ⚠️ Study the performance of Variational AutoEncoder and Graph Neural Network (GNN) models
5. ✅ Measure and characterize key timing characteristics to understand continuous learning cadence

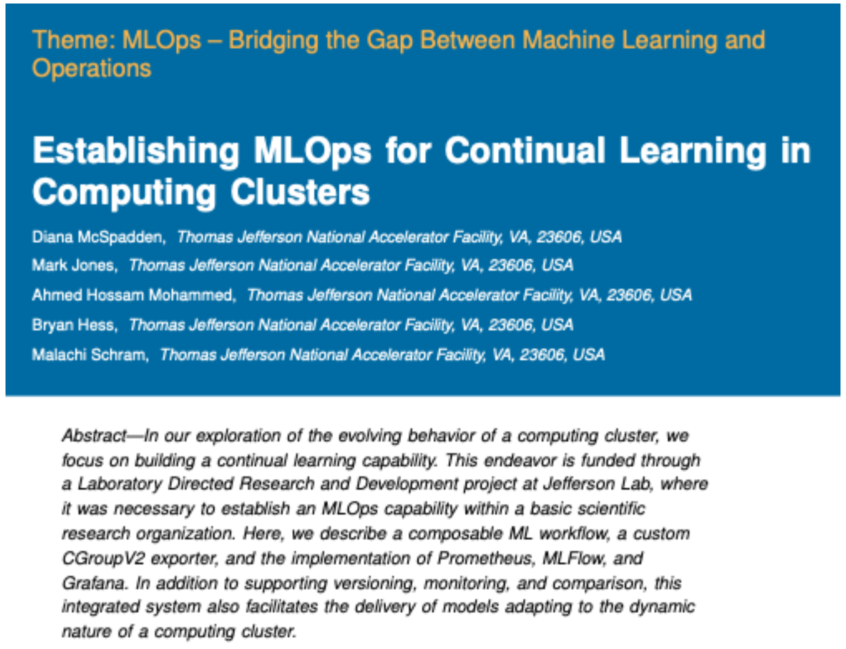
Management Changes for Q3, Q4

- Staffing increase for model development
- MLOps work to complete workflow automation
- Refactor software to fit existing architecture for pluggable changes
- Increase effort to provide diverse NP jobs. This has been more challenging than expected.

Backup

MLOps Update

- The manuscript "Establishing MLOps for Continual Learning in Computing Clusters" is under review for an IEEE Software MLOps special issue.
- **Code** for daily continual learning with competition between models has been manually tested on farm.
  - CPU and memory features
  - one model with a GNN layer and one without the GNN layer.
  - Both saved to the MLFlow model repository
  - Daily champion model is used for real-time reconstruction error results.
- Continual Integration/Continual Deployment
  - Approved git pull request results in a Container rebuild
  - Container can be run on dt100
  - Containers built on a timer on the dt100, with built containers stored on Code Gitlab instance



Theme: MLOps – Bridging the Gap Between Machine Learning and Operations

Establishing MLOps for Continual Learning in Computing Clusters

Diana McSpadden, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Mark Jones, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Ahmed Hossam Mohammed, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Bryan Hess, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*
Malachi Schram, *Thomas Jefferson National Accelerator Facility, VA, 23606, USA*

Abstract—In our exploration of the evolving behavior of a computing cluster, we focus on building a continual learning capability. This endeavor is funded through a Laboratory Directed Research and Development project at Jefferson Lab, where it was necessary to establish an MLOps capability within a basic scientific research organization. Here, we describe a composable ML workflow, a custom CGroupV2 exporter, and the implementation of Prometheus, MLFlow, and Grafana. In addition to supporting versioning, monitoring, and comparison, this integrated system also facilitates the delivery of models adapting to the dynamic nature of a computing cluster.

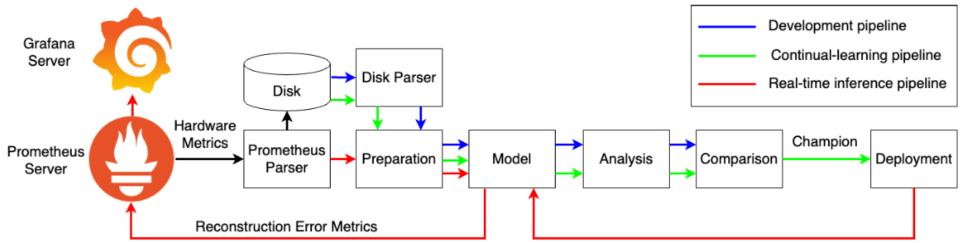
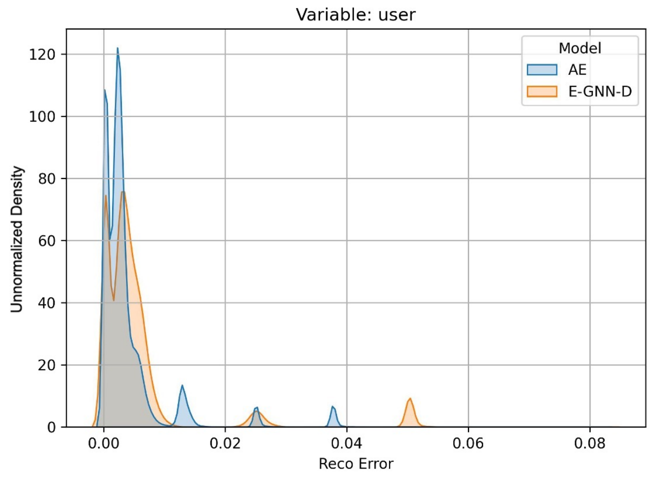
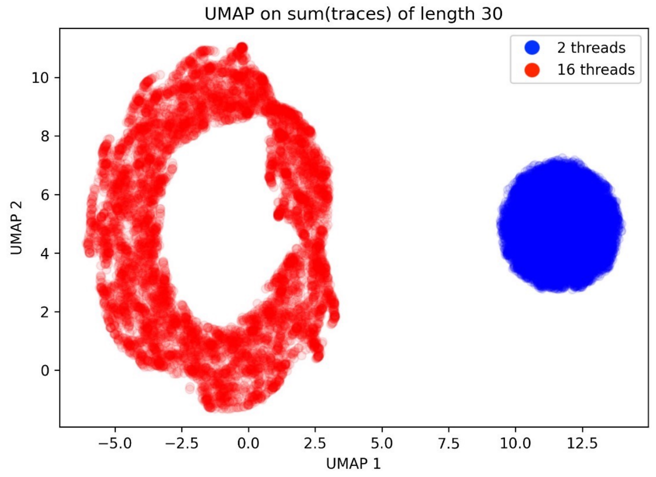


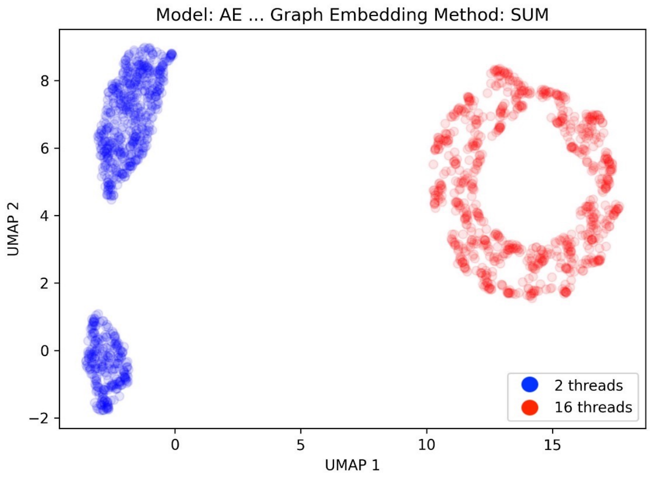
FIGURE 1: The DIDACT pipeline Python code modules, and server and database endpoints. Blue, green, and red arrows represent the development pipeline, the continual-learning pipeline, and the real-time inference pipeline, respectively.

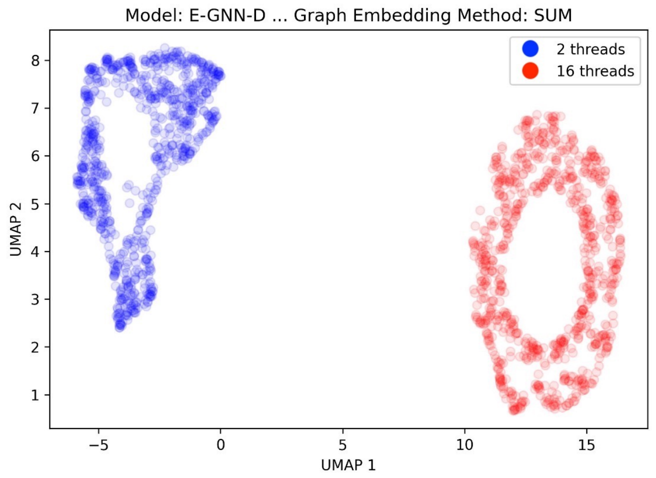
AutoEncoder (AE baseline architecture) VS GNN-assisted AE

- The model is used for two different tasks:
    1. The first task captures the salient features of the data using an AE.
    2. The second task clusters the job types based on the salient features using techniques such as UMAP.
- The baseline architecture is complete, and we will explore the performance of these models when we increasing the data volume and model parameters.
- Based on the results, we see a clear separation between to current job types. However, it is unclear that one of the models does significantly better than the other in this case
- We will study the performance of these models when we include additional job types and complexity
- Additional research directions: VAE (Variational AE): Although vanilla AE does a very good job in producing representative embeddings, VAE can serve as a useful tool to quantify the uncertainty in data.





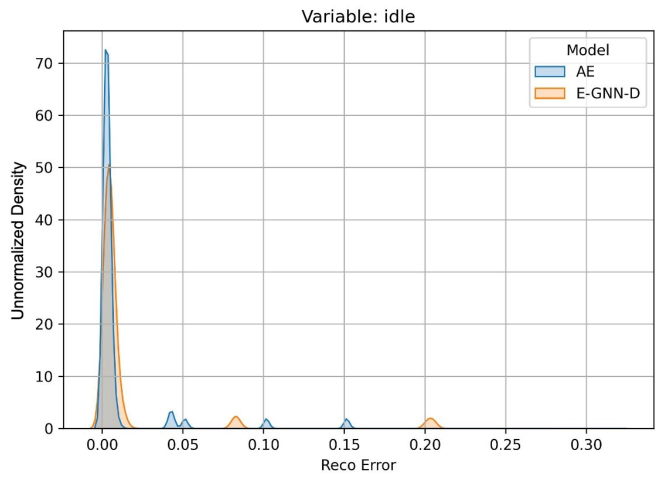


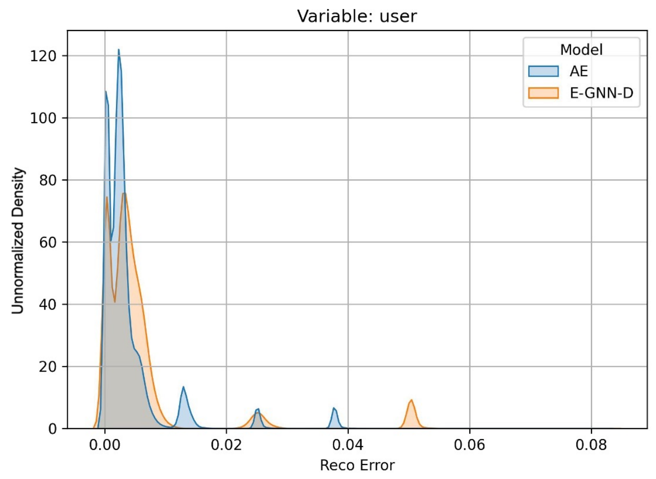


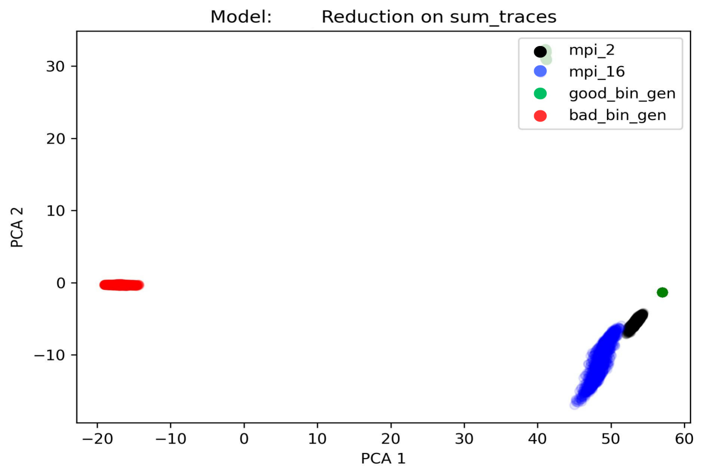
Jefferson Lab

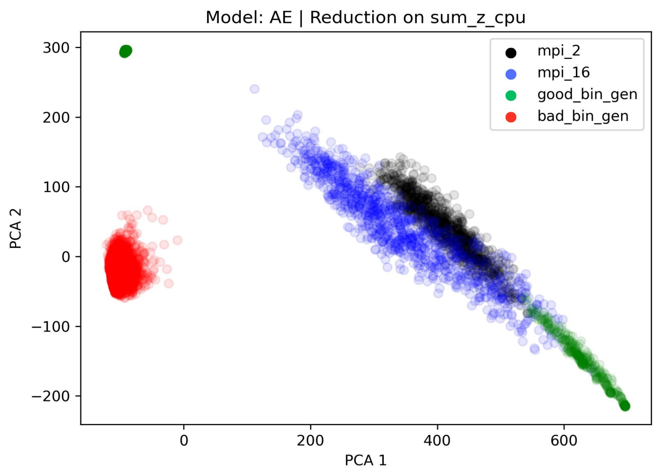
AutoEncoder (AE baseline architecture) VS GNN-assisted AE

- We are working on stages of the models:
  1. Capturing the salient features within the traces. This is quantified through the reconstruction error. Top figures compare the AE vs the GNN-assisted AE in terms of the reconstruction error of the "idle" and "user" traces of the CPU.
  2. Understanding the separation between the jobs (PCA/UMAP). A representative model would be able to reflect the jobs that are separated in the original space (bottom left figure) into a separated embedded latent space (bottom right figure).
- Based on the results, it is unclear that one of the models does significantly better than the other in this case
- Pursued research direction: Although vanilla AE does a very good job in producing representative embeddings, VAE (variational AE) can serve as a useful tool to quantify the uncertainty in data (i.e., UQ analysis).









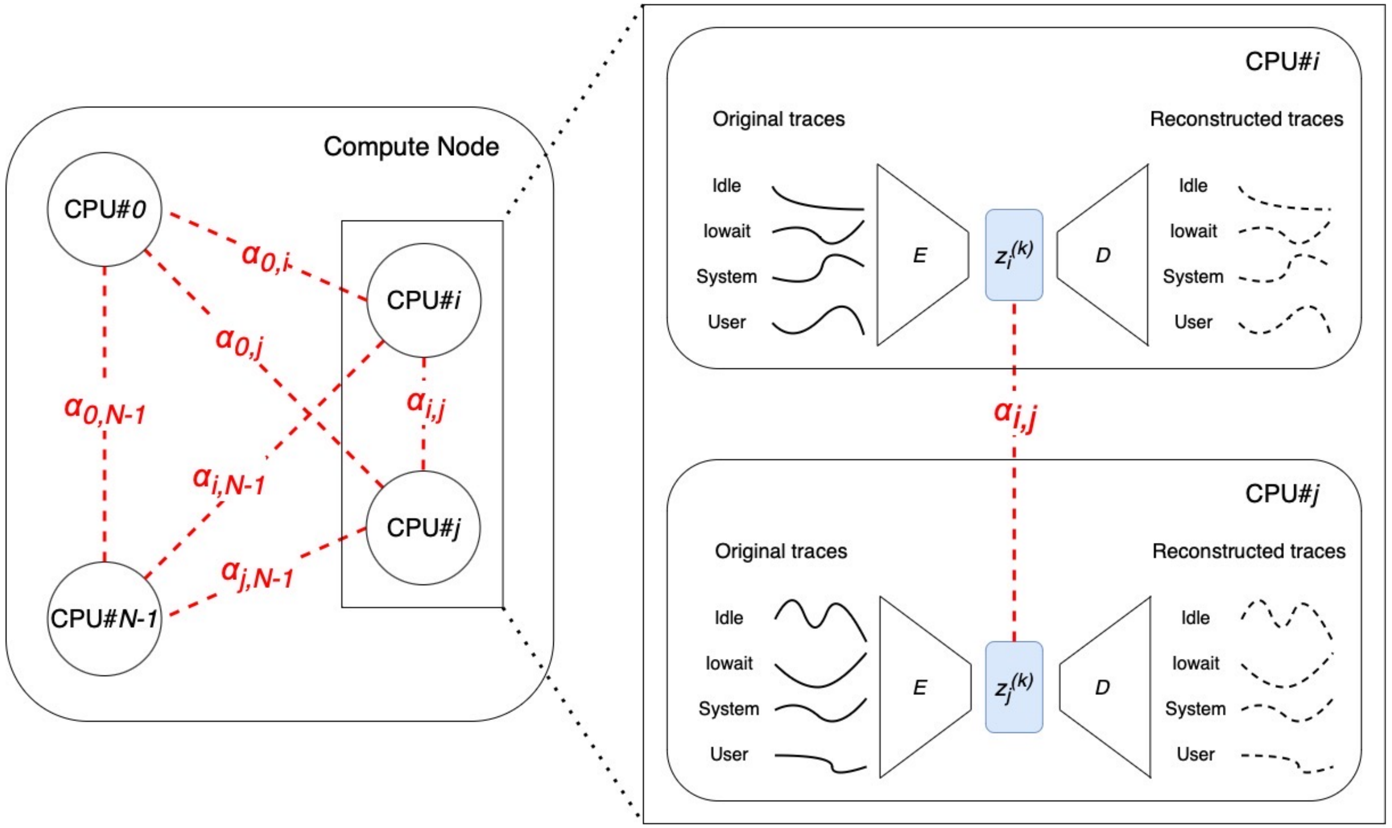
CPU Encoder-GNN-Decoder

- Encoder *E*: Input traces of CPU#*i* ➔ compressed embedding $z_i^{(0)}$.
- Decoder *D*: Tries to reconstruct original traces from z_i. Due to compression loss, perfect reconstruction is impossible.
- Hence, *E* learns the best compressed representation of the input while *D* learns the best reconstruction function.
- We can optionally introduce a GNN module between *E* and *D* such that each CPU becomes aware of its neighbors activity by sharing the latent embedding *z*:

$$z_i^{(1)} = z_i^{(0)} + f\left(z_i^{(0)} + \sum_{j=0}^{N-1} \alpha_{i,j}\, z_j^{(0)}\right),$$

where *f* and α are learnable functions.

- Due to the additional information each CPU gets, it is believed that the reconstruction error would be less especially in multi-threaded jobs where different cores affect each other.


Compute Node
CPU#0
CPU#i
CPU#j
CPU#N-1
$\alpha_{0,i}$
$\alpha_{0,j}$
$\alpha_{0,N-1}$
$\alpha_{i,j}$
$\alpha_{i,N-1}$
$\alpha_{j,N-1}$
CPU#i
Original traces
Reconstructed traces
Idle
Iowait
System
User
E
$z_i^{(k)}$
D
$\alpha_{i,j}$
CPU#j
Original traces
Reconstructed traces
Idle
Iowait
System
User
E
$z_j^{(k)}$
D