A64FX Enables Engine Decarbonization Using Deep Learning

Rodrigo Ristow Hadlich
Institute of Advanced Computational Sciences
Stony Brook, New York, USA
rodrigo.ristowhadlich@stonybrook.edu

Gaurav Verma
Institute of Advanced Computational Sciences
Stony Brook, New York, USA
gaurav.verma@stonybrook.edu

Tony Curtis
Institute of Advanced Computational Sciences
Stony Brook, New York, USA
anthony.curtis@stonybrook.edu

Eva Siegmann
Institute of Advanced Computational Sciences
Stony Brook, New York, USA
eva.siegmann@stonybrook.edu

Dimitris Assanis
Institute of Advanced Computational Sciences
Stony Brook, New York, USA
dimitris.assanis@stonybrook.edu

ABSTRACT

Decarbonization of transportation requires new deep learning models to enable improved engine control. Research and development must also be done in a computationally efficient manner, so of interest is to understand the applicability of high performance computing resources to be used for training machine learning models and to compare both the power consumption and temporal performance of new A64FX architectures to x86 architectures. This work details the development of a Multilayer Perceptron (MLP) model using the Fujitsu A64FX processor for predicting in-cylinder pressure of internal combustion engines, a critical performance parameter to develop pathways to decarbonized engine controls. A scaling analysis of up to 160 compute nodes demonstrates continuously improving performance with increasing number of nodes. A comparison of performance and power consumption between A64FX and Intel's x86 Sapphire Rapids (SPR) is also included and shows that up to 30 parallel nodes the power efficiency of A64FX is lower, but that its energy consumption is constant as opposed to SPR which increases linearly as node count increases.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Hardware** → **Power and energy**.

KEYWORDS

Deep Learning, Combustion Engine, A64FX, Energy Analysis, Performance

ACM Reference Format:
Rodrigo Ristow Hadlich, Gaurav Verma, Tony Curtis, Eva Siegmann, and Dimitris Assanis. 2024. A64FX Enables Engine Decarbonization Using Deep Learning. In *Practice and Experience in Advanced Research Computing (PEARC '24), July 21–25, 2024, Providence, RI, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626203.3670619

PEARC '24, July 21–25, 2024, Providence, RI, USA

ACM ISBN 979-8-4007-0419-2/24/07
https://doi.org/10.1145/3626203.3670619

1 INTRODUCTION

The transportation sector is responsible for approximately 22.7% of the global carbon dioxide (CO_2) emissions [8], largely stemming from fossil fuel use in internal combustion engines (ICEs). Alternative fuels and advanced combustion modes can reduce greenhouse gas (GHG) emissions by improve efficiency and reducing harmful emissions, but are hampered by implementation challenges due to increased system control volatility with existing engine control strategies.

Therefore, new strategies need to be explored, including ones based on Reinforcement Learning, which can greatly benefit from a pre-training process using a model of the engine. There are two approaches for modeling engines: physics-based models [1, 6, 11, 15] which rely on numerical representations of physical processes such that model fidelity ranges and directly increased computational cost; or data-driven models which can be trained using data collected from a physical engine [13]. Specifically, this work will focus on machine learning (ML) data-driven engine models. The benefits include (1) that the same model can be applied to any engine for which data is available, thus greatly reducing the model tuning time and eliminating the requirement of knowledge of the system, and (2) predictions are very quick to make once the model is trained, meaning it can generate data at a very fast rate. In addition, if the model is properly tuned and the quality of the data is good, the accuracy of predictions can be very high as demonstrated in [13].

In this manuscript is a simple Multilayer Perceptron (MLP) model is developed that takes as input a tuple, of length four, with crank-angle resolved values of (1) crank position and fuel injection event (2) pressure (P_{INJ}), (3) duration (τ), and (4) timing (T_{INJ}), as well as outputs a single pressure value. Model predictions are performed for each of the 720 crank angle degrees (CAD) of rotation in a four-stroke engine cycle. This pressure time-history referred to the engine cycle pressure trace is the basis to calculate many cyclic performance indicators. These derived performance indicators can then be used as inputs to control systems in real time to adjust a set of operating conditions for the subsequent cycle, and thus control engine operation on a cycle-to-cycle basis.

Research and development of such new pathways to decarbonize ICEs must also be done in an energy efficient manner. As such, we will focus on the computational energy consumption in the development of the deep learning model presented and computed

on the Ookami cluster [2, 3] located at Stony Brook University (SBU) which contains A64FX processors developed by Fujitsu for high performance and low power consumption [12]. Thus, this manuscript will focus on understanding the inter-node scalability in this chip architecture as well as power consumption compared to x86 chips for advanced ICE control application.

2 SET-UP/METHODS

2.1 Engine and Dataset Description

Experimental validation data was collected on a single-cylinder compression ignition research engine at the Advanced Combustion and Energy Systems (ACES) laboratory of SBU. The engine was operated with a single fuel injection event per cycle using research grade No.2 diesel fuel in conventional compression ignition combustion. More details are provided on the experimental set-up and testing methodology are disclosed by Ristow Hadlich et al. [14].

The experimental matrix varied operating conditions corresponding to P_{INJ}, τ, and T_{INJ}. P_{INJ} was varied between 450 and 850 bar in 100 bar increments, τ was varied to reach fuel-air equivalence ratios (ϕ) between 0.2 and 0.4 in steps of 0.05 and T_{INJ} was varied between the knock limit and misfire limit, as permitted given the values of the other variables. Knock phenomenon was defined as pressure rise rates higher than 10 bar/CAD, and misfire was defined by a coefficient of variation (COV), standard deviation divided by the mean, of the net indicated mean effective pressure ($IMEP_n$) exceeding 5%. $IMEP_n$ and ϕ were determined using the standard method disclosed in [7]. For each set of operating conditions described above, 300 consecutive cycles were recorded. In total, the dataset contains 28800 data points corresponding to 96 different operating conditions. The data was further divided into train, validation, and test subsets containing 23100, 2700, and 3000 data points, respectively. It is important to point out that all data points corresponding to a set of operating conditions were grouped together when dividing into the subsets, meaning that all the data corresponding to a given combination of operating conditions belongs exclusively to one of the data subsets.

2.2 HPC Infrastructure

The two SBU HPC clusters, Ookami & Seawulf, and the PyTorch environments are discussed. Data parallelism was accomplished using PyTorch's Distributed Data Parallel (DDP) and *mpi* backend.

The Ookami cluster contains 174 1.8GHz A64FX-FX700 nodes, each with 48 cores, 32GB of high bandwidth memory (HBM) and 512GB SSD. The chips (one per node) have four core memory groups (CMGs) with 8GB HBM each, and thus four non-uniform memory access (NUMA) regions. The communication between nodes is accomplished via an HDR 200 Gbit/s Infiniband network configured as a full fat tree. Additional details can be found in [3]. The PyTorch environment (PyTorch v2.1.0) used for the A64FX runs was built from source to accommodate for the *mpi* backend using the ARM compiler v22.0, ARMPL v2022.0.1 and Open MPI version 4.1.2. This choice of ARM compiler suite was chosen based on the results of PyTorch performance on A64FX using different compilers reported in [4]. Energy consumption was measured using the ipmi tool[9].

The Seawulf cluster contains approximately 23,000 cores distributed in over 400 compute nodes, 96 of which are Intel Sapphire Rapids (SPR). The SPR nodes contain two Intel Xeon Max 9468 CPUs with 48 cores each and operate at a base speed of 2.6 Gigahertz and 256 GB of DDR5 RAM + 128 GB of HBM2 RAM on a NDR InfiniBand network. Similarly to the A64FX chips, the SPR chips, operating in Flat mode with SNC4 clustering, have NUMA regions with HBM, but each of these Intel chips have 8 NUMA regions totalling 16 per node. The PyTorch environment (PyTorch v2.2.0) used for the SPR runs in this paper was also built from source to enable the *mpi* backend. Open MPI v5.0.2 [5], compiled with GCC v13.2.0, was chosen as the PyTorch MPI implementation due to ease of use and broad support. To optimize the PyTorch performance on x86 architecture [4] and ensure compatibility and consistency across the environment and dependent libraries, GCC v13.2.0 was employed for all compilations. This choice also supports the full implementation of C++17, a prerequisite for building PyTorch from source. Additionally, the build utilized Intel MKL libraries to optimize performance on the underlying hardware. The energy was measured the same way as for Ookami using the ipmi tool.

3 RESULTS

The main focus of this work is to analyze the benefits and limitations of the Ookami cluster A64FX architecture for use with ML models (model selection introduced first) for advanced control strategies relevant to decarbonizing ICEs. Two separate analyses were performed to understand (1) how temporal performance scales with increasing parallel computational resources and (2) how A64FX chip architecture compares to x86 in terms of execution speed versus power consumption.

3.1 Model Selection

An important step in developing a good model for a given application is to find the hyperparameters that work best for the given dataset. While the focus of this paper is not to evaluate how a model performs in terms of its predictions (the subject of previous work [13]), it is still of interest to find the best model for the application of interest and to understand the computational intensity of the task and how it can scale using HPC. Therefore, a hyperparameter optimization (HPO) was performed using a simple random sampling algorithm developed by the authors. While python libraries are available that offer many more capabilities for such purposes, their integration with PyTorch DDP was found problematic in terms of guaranteeing reliable model execution. The HPO search was performed using initial bounds based on previous knowledge, with the bounds expanded after a few hundred iterations once the results demonstrated the approximate optimal solution was outside current bounds. Custom stopping criteria were implemented to stop training of the model at either the point of model convergence or divergences.

The hyperparameters that were varied throughout the search were (1) the number of hidden layers, (2) the number of nodes per hidden layer, (3) the learning rate of the Adam optimization algorithm [10], (4) the non-linear activation function (chosen between ReLU, leaky ReLU and hyperbolic tangent ($tanh$)), and (5) the dropout probability to provide a means of mitigating overfitting. During the training of each model, the mean squared error (MSE) was selected as the loss function chosen for the update to the model

Table 1: Hyperparameter Optimization Summary

Parameter Name	Search Bounds	Optimal Combination
Number of hidden layers	[1:8]	6
Nodes per hidden layer	$2^{[4:10]}$	2^7
Learning rate	$[10^{-7}{:}10^{-4}]$ (log scale)	0.000026
Activation function	(ReLU, Leaky ReLU, *tanh*)	*tanh*
Dropout probability	int([0:5])*0.1	0

Table 2: Summary of models for scalability test

Model Labels	Number of Hidden Layers	Nodes per Hidden Layer	Number of Trainable Parameters
Small	2	16	336
Optimal	6	128	82,560
Large	8	1024	7,345,152

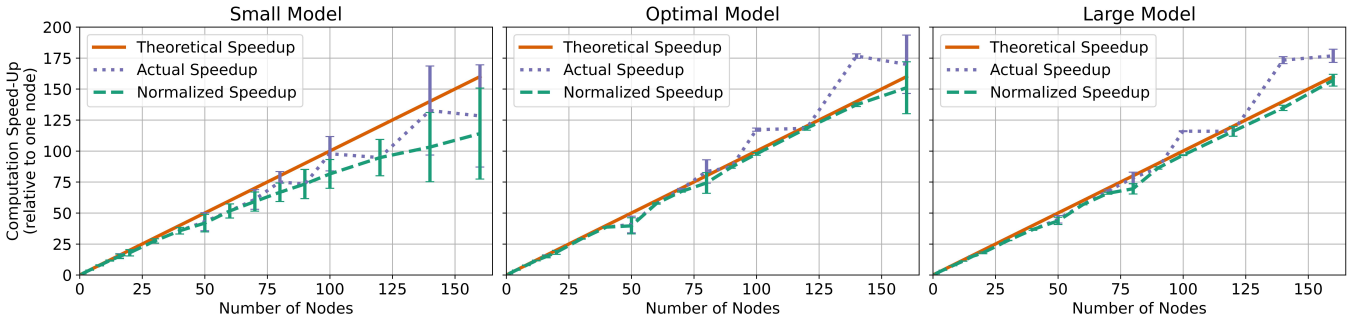

Figure 1: Computational speedup as function of number of nodes for small, optimal, and large models.

weights. To evaluate the performance of the trained models and rank between them, the mean absolute error (MAE) of the models, based on the validation dataset, was used. The best model featured an MAE of 0.105 bar (within 0.2%). Table 1 summarizes the bounds of the search and the optimal combination of hyperparameter values.

3.2 Scalability Testing on Ookami

The process and results of the scalability testing performed on the Ookami cluster are detailed below. Based on the determined HPO bounds and results, three distinct models were used throughout the scalability investigation. The smallest and largest models developed during the HPO, in terms of number of trainable parameters, were selected as the representative edge case models requiring the least and the most computational resources, respectively. The third model selected is the optimal model found in the HPO, which lies somewhere between the other two models based on number of trainable parameters. Details on the number of layers, nodes per hidden layer, and amount of trainable parameters are presented in Table 2. The learning rate, activation function, batch size, and dropout probability for all models are 10^{-6}, *tanh*, 4, and 0, respectively.

To ensure run consistency, a singular python script was developed to handle all model testing. For each evaluation considered, the same initial random state for both model and optimizer were prescribed and an average was determined of the 30 model training trials that each elapsed for a fixed 20 epochs. The only exceptions to this are the runs of 1 through 90 nodes of the large model, for which only featured 5 trials of 20 epochs due to the very extensive training time required in lower node counts. The error bars presented in Figure 1 are determined from the COV at each point as a percentage of the value. The same intra-node mapping was used in all the runs, with 16 processes per node divided evenly between the four NUMA regions of each node and two OpenMP threads per process. The reasoning behind this distribution was to use as many processes per node as possible, while ensuring that no issues were encountered in the DDP data loader. The issue would arise from having more processes spawned than there are examples in the validation dataset, which contained 2700 examples. Using 16 processes per core allowed scaling up to 160 nodes without running into this issue (2560 processes).

Figure 1 shows the results from the scalability test. The speedup as a function of n nodes is the ratio of training time using a single node to that of using n nodes. For each model, three curves are shown: (1) the theoretical speedup (N resources = $1/N$ time), (2) the actual measured speedup to train the model, and (3) the calculated normalized speedup. This normalization was necessary because as the number of total processes in a run was varied, the distributed

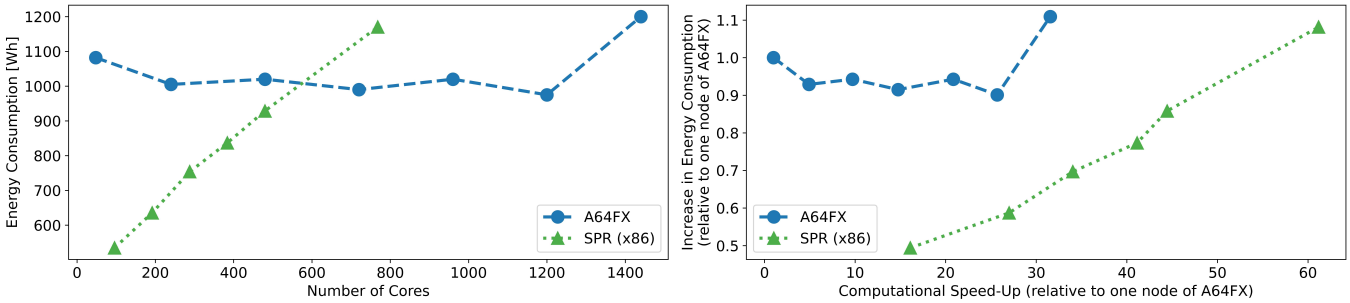

Figure 2: Comparison of performance and energy consumption between A64FX and SPR.

data sampler in DDP would prune the training dataset to distribute training examples equally among the processes (this also explains why the actual speedup is larger than the theoretical limit at times). The normalization factor used is simply a ratio of the number of examples seen in the given run compared to that of a single node (ratio is less than or equal to one).

In figure 1, it can be seen that for all models, increasing the number of nodes improves performance in the entire range of nodes explored which highlights how A64FX parallel computing is beneficial. Critically, a more nuanced observation can be made that larger models benefit more from the increased number of nodes. For the small model, as the number of nodes increased the gap between theoretical and normalized speedups grew which indicates a decrease in computational efficiency. As the model size increased, the gap became smaller with the large model showing the closest performance to the theoretical limit in the highest node count.

3.3 Comparison of A64FX and Sapphire Rapids

This subsection provides a comparison in the power consumption between A64FX and SPR nodes. The intra-node mapping was of 11 processes per NUMA with one thread each on A64FX (total of 44 processes per node) and 3 processes per NUMA with a single thread each on SPR (total of 48 processes per node). These were found to be the optimal mappings for each architecture. Figure 2 shows, for both A64FX and SPR nodes, the energy consumption as a function of the number of cores (left) and the relative increase in energy consumption as a function of computational speedup (right), both the increase in energy and speedup calculated relative to the values from a single A64FX node. The model used for this analysis was the optimal model trained for 20 epochs, the values reported being an average runtime and total energy consumption of 30 runs. It shows the energy consumption of A64FX remains constant with increasing number of nodes, while the consumption of the SPR nodes increases linearly as node count increases.

4 CONCLUSIONS

The findings of this work can be summarized by the following statements. (1) The training of models in this application greatly benefit from parallelizing the workload between multiple nodes, and using the hardware from the Ookami cluster the inter-node communication was not a limiting factor in scaling up the number of nodes up to 160. (2) The energy efficiency (run-time vs energy consumption) of the SPR nodes is higher in the range presented here, but worth noting energy consumption of A64FX remains constant while SPR increases linearly with increasing node count.

ACKNOWLEDGMENTS

We thank the SBU Research Computing and Cyberinfrastructure and the Institute for Advanced Computational Science at SBU for access to: (1) the high-performance SeaWulf computing system made possible by $1.85M in grants from the National Science Foundation (awards 1531492 and 2215987) and matching funds from the Empire State Development's Division of Science, Technology and Innovation (NYSTAR) program (contract C210148); and (2) the innovative high-performance Ookami computing system made possible by a $5M National Science Foundation grant (#1927880).

REFERENCES

[1] D. Assanis, N. Engineer, P. Neuman, and M. Wooldridge. 2016. Computational Development of a Dual Pre-Chamber Engine Concept for Lean Burn Combustion. In *SAE Technical Paper*. https://doi.org/10.4271/2016-01-2242

[2] M. A. S. Bari, B. Chapman, A. Curtis, R. J. Harrison, E. Siegmann, N. A. Simakov, and M. D. Jones. 2021. A64FX performance: Experience on Ookami. *Proceedings - IEEE International Conference on Cluster Computing, ICCC* 2021-September (2021), 711–718. https://doi.org/10.1109/CLUSTER48925.2021.00106

[3] A. Burford, A. Calder, D. Carlson, B. Chapman, F. Coskun, T. Curtis, C. Feldman, R. Harrison, Y. Kang, B. Michalowicz, E. Raut, E. Siegmann, D. Wood, R. Deleon, M. Jones, N. Simakov, J. White, and D. Oryspayev. 2021. Ookami: Deployment and Initial Experiences. *ACM Intl. Conference Proceeding Series* (7 2021). https://doi.org/10.1145/3437359.3465578

[4] S. Chheda, A. Curtis, E. Siegmann, and B. Chapman. 2023. Performance Study on CPU-based Machine Learning with PyTorch. *ACM Intl. Conference Proceeding Series* 11, 2023 (2 2023), 24–34. https://doi.org/10.1145/3581576.3581615

[5] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. 2004. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*. 97–104.

[6] G. Guleria, D. Lopez-Pintor, J. E. Dec, and D. Assanis. 2023. Development and evaluation of a skeletal mechanism for EHN additized gasoline mixtures in large Eddy simulations of HCCI combustion. *Intl. J. of Engine Research* 24, 9 (6 2023), 3863–3877. https://doi.org/10.1177/14680874231178099

[7] J. B Heywood. 2018. *Internal combustion engine fundamentals*. McGraw-Hill Education.

[8] IEA. [n. d.]. Greenhouse Gas Emissions from Energy Data Explorer – Data Tools. https://www.iea.org/data-and-statistics/data-tools/greenhouse-gas-emissions-from-energy-data-explorer

[9] ipmi. [n. d.]. GitHub - ipmitool/ipmitool: An open-source tool for controlling IPMI-enabled systems. https://github.com/ipmitool/ipmitool

[10] D. P. Kingma and J. L. Ba. 2014. Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (12 2014). https://doi.org/10.48550/arxiv.1412.6980

[11] I. Nikiforakis, G. Guleria, M. Koraiem, and D. Assanis. 2022. Understanding Pre-Chamber Combustion Performance in a Closed-Cycle Model of a Novel Rotary Engine. *SAE Technical Paper* (2022). https://doi.org/10.4271/2022-01-0396

[12] R. Okazaki, T. Tabata, S. Sakashita, K. Kitamura, N. Takagi, H. Sakata, T. Ishibashi, T. Nakamura, and Y. Ajima. 2020. Supercomputer Fugaku CPU A64FX Realizing High Performance, High-Density Packaging, and Low Power Consumption. *Fujitsu Technical Review* (2020).

[13] R. Ristow Hadlich, J. Loprete, and D. Assanis. 2024. A Deep Learning Approach to Predict In-Cylinder Pressure of a Compression Ignition Engine. *J. of Eng. for Gas Turbines and Power* (1 2024), 1–12. https://doi.org/10.1115/1.4064480

[14] R. Ristow Hadlich, Z. Ran, D. Yang, R.and Assanis, O. Mante, and D. Dayton. 2022. Experimental Investigation and Comparison of a Decalin/Butylcyclohexane Based Naphthenic Bio-Blendstock Surrogate Fuel in a Compression Ignition Engine. *SAE Intl. J. of Adv. and Cur. Practices in Mobility* 4, 5 (3 2022), 1771–1781. https://doi.org/10.4271/2022-01-0513

[15] R. Yang, Z. Ran, R. Ristow Hadlich, and D. Assanis. 2022. A Double-Wiebe Function for Reactivity Controlled Compression Ignition Combustion Using Reformate Diesel. *Journal of Energy Resources Technology, Transactions of the ASME* 144, 11 (11 2022). https://doi.org/10.1115/1.4053981/1137859