ICCS 2025
7-9 July 2025

Influence of Mixed Precision on Performance and Accuracy of DNN Training for AI-Accelerated CFD Simulations on NVIDIA Multi-GPU System

Kamil Halbiniak[1], Krzysztof Rojek[1], Roman Wyrzykowski[1],
Paweł Gepner[2] and Norbert Meyer[3]

[1] Czestochowa University of Technology, Poland
[2] Warsaw Technical University, Poland
[3] Poznan Supercomputing and Networking Center, Poland

POLI [TECH] NIKA Czestochowa University of Technology

Department of Computer Science

Agenda

- Introduction
- Floating-point data formats and mixed precision in AI-accelerated CFD simulation
- Deep neural network for AI-accelerated CFD simulation
- Distributed data parallel training of DNN
- Overview of testing platform
- Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format
- Performance-accuracy trade-off for TF32 format versus BF16 and FP32 formats
- Conclusions

Introduction

- The recent incorporation of AI into CFD has opened new prospects for faster and more reliable simulations
- In work [1], we proposed a methodology aimed at enhancing CFD simulations by integrating two main components: the AI supervisor and the AI accelerator
- This research explores the potential of mixed precision techniques with diverse data formats - BF16, TF32, and FP32 - to accelerate the distributed data-parallel training of our DNN model proposed for CFD motorBike simulations on an HPC system using multiple NVIDIA GH200 chips
- Especial emphasis is given to validating and tuning the accuracy of training concerning the impact of mixed precision methods and partitioning a large training dataset into smaller batches
- We aim to understand better how various number formats impact the performance-accuracy trade-off in training DNN models for CFD simulations on modern HPC platforms with multiple GPUs and nodes

1. *Krzysztof Rojek, Roman Wyrzykowski, and Pawel Gepner. AI-Accelerated CFD Simulation Based on OpenFOAM and CPU/GPU Computing. In Maciej Paszynski, Dieter Kranzlm¨uller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, Computational Science – ICCS 2021, pages 373–385, 2021*

Floating-point data formats and mixed precision in AI-accelerated CFD simulation

- Precision in deep learning models is a critical factor influencing performance
  - FP32 (*float32*) Format
    - Backbone of deep learning
    - High range of representable values
    - Suitable for accuracy-critical AI computations
    - Requires significant memory and computing resources
  - FP16 (*float16*) and BF16 (*bfloat16*) Formats
    - Lower precision formats
    - Reduced memory usage and computational demands
    - Attractive for large-scale AI computations (e.g., training large models)
    - Potential numerical instability in certain computations
  - TF32 (*TensorFloat-32*)
    - It uses only 19 bits with three additional bits for mantissa compared to BF16
    - Practically the same accuracy for AI computation as the FP32 format while providing radical performance compared to single-precision
  - Mixed Precision Techniques
    - Trade-offs between precision, memory constraints, and computational efficiency
    - Optimal choice depends on the specific application and precision needs

Deep Neural Network for AI-Accelerated CFD Simulation

- The AI model for predicting iteration results in motorBike CFD simulations leverages a variational autoencoder (VAE) architecture
- The AI model is trained on extensive datasets from RANS simulations, designed to infer and optimize the estimation of flow velocity *U* and pressure *p* in a motorcycle aerodynamic analysis
- The dataset is constructed through a parametric series of simulations with key physical variables, systematically varied to capture a broad range of operational conditions
- The VAE is used to predict critical CFD quantities such as pressure and velocity around a motorbike geometry
- Our approach employs an ML pipeline with four iterations of CFD simulation, which produces the input data for the VAE and generates a single predicted output
  - Each input corresponds to *q* quantities (e.g., pressure, velocity) across *n* domain cells, represented as an array of size *timesteps* × *n* × *q*
  - The output is represented as an array of size *1* × *n* × *1*, predicting the last time step's pressure distribution

Distributed Data Parallel Training of DNN

- Distributed data-parallel (DDP) training is an advanced form of data-parallel (DP) training, assuming computations across multiple nodes and allowing training to scale beyond a single machine
- Training DNN models relies on executing iteratively three steps: (i) the forward pass to compute loss, (ii) the backward pass to compute gradients, and (iii) the optimizer step to update parameters
- In DDP training, multiple processes are launched on various machines with each process usually assigned to a single GPU
  - Each process performs the forward pass independently on separate partitions of the training dataset to calculate the local gradients
  - Models for all processes have to be synchronized at each training iteration by sharing and averaging the local gradients in order to compute the global gradient used by each process to update the model
- For the DDP training, we can distinguish the batch size B_S corresponding to the size of subsets processed by each GPU and the effective batch size B_{Ef} , which corresponds to the size of the whole dataset processed by all p GPUs of the computing platform, where $B_{Ef} = pB_S$
- In this work, PyTorch is employed to implement DDP training of DNN

Testing platform

- The benchmarks are performed on the Helios supercomputer installed at ACC Cyfronet AGH
- The tests are executed on four nodes with four NVIDIA GH200 accelerators
  - ✓ NVIDIA GH200 Superchip is based on heterogenous Grace Hooper architecture
  - ➢ It combines the high-performance capabilities of the NVIDIA Hopper H100 GPU with the versatility of the NVIDIA Grace CPU based on Neoverse V2 Armv9 architecture
  - ➢ It has 72 cores and up to 480GB of LPDDR5X memory
  - ➢ The GPU chip includes 16,896 FP32 and up to 144 GB of HBM3e memory
  - ➢ GPU contains 528 fourth-generation Tensor Cores that support various precisions, including FP64, FP32, BF16, TF32, and newly introduced FP8
  - ➢ The theoretical peak performance (without sparsity) for TF32 and BF16 formats equals 494 and 990 TFlop/s, respectively
- Software installed on the platform:
  - ➢ Python 3.11.5
  - ➢ PyTorch 2.3.1
  - ➢ NVIDIA CUDA SDK 12.4.0

Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format

- The performance and accuracy of parallel DNN training with BF16 data format is evaluated in the following three scenarios:
  - *Scenario 1: Benchmarking the scalability without considering accuracy*
    - The fixed number N_E = 50 of training epochs is executed on various GPU numbers, with the batch size B_S = 16 set corresponding to the maximum portion of the dataset that ensures all computations fit into the GPU memory
  - *Scenario 2: Investigating the scalability of training with considering accuracy*
    - Unlike the previous scenario, training is performed for different numbers of epochs until the loss reaches a value equal to that obtained for a single GPU (with a tolerance of 10%)
  - *Scenario 3: Analyzing the scalability of training when batch sizes B_S and B_{Ef} are selected concerning the performance and accuracy trade-off*
- PyTorch Automatic Mixed Precision (`torch.amp`) package is employed to enable mixed-precision computations with BF16 format

Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format

- Performance and accuracy results achieved for *Scenario 1*

	1 x GH200	2 x GH200	4 x GH200	8 x GH200	16 x GH200
Time [s]	1274	657	336	175	90
Final loss	0.141	0.149	0.148	0.211	0.278
Speedup	1	1.94	3.79	7.28	14.16

- Increasing the number of GPUs significantly reduces the execution time of DNN training, with nearly linear scaling and efficiency remaining high across different GPU configurations
- Using all GPUs allows us to accelerate the training 14.6 times (89% of ideal scalability)
- However, while training performance increases significantly, the accuracy decreases when employing more GPUs
- The final loss remains relatively stable when using up to 4 GPUs, but it rises significantly for 8 and 16 GPUs (0.211 and 0.278, respectively)
- This behavior suggests that, in our case, selecting a large size B_{Ef} negatively affects the training convergence.

Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format

- Performance and accuracy results achieved for *Scenario 2*

	1 x GH200	2 x GH200	4 x GH200	8 x GH200	16 x GH200
Time [s]	-	-	-	260	172
Speedup	-	-	-	4.9	7.41

- In this scenario, training is performed only for 8 and 16 GPUs as yielding noticeably lower accuracy in *Scenario 1*
- The tests show that $N_E = 74$ and $N_E = 94$ epochs are required to reach the loss value achieved for a single GPU
- However, increasing N_E decreases the training performance significantly
- As a result, the value of speedup is equal to only 4.9 and 7.41 for configurations with 8 and 16 GPUs, respectively

Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format

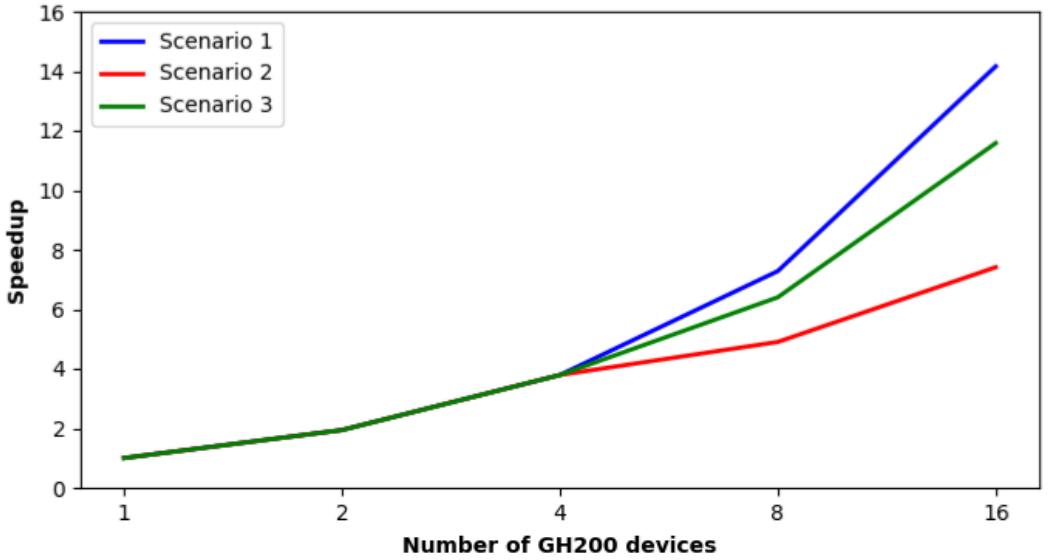
- Performance and accuracy results achieved for *Scenario 3*

		1 x GH200	2 x GH200	4 x GH200	8 x GH200	16 x GH200
$B_{Ef} = 16$	Time [s]	1274	764	396	217	129
	Final loss	0.141	0.14	0.144	0.142	0.146
	Speedup	1	1.67	3.22	5.87	9.88
$B_{Ef} = 64$	Time [s]	-	-	-	199	110
	Final loss	-	-	-	0.15	0.152
	Speedup	-	-	-	6.4	11.58

- This scenario starts with assuming the same effective batch size $B_{Ef} = 16$ as B_S for a single GPU, regardless of the number of GPUs
- The obtained results indicate better scalability than *Scenario 2* while providing the desired accuracy
- At the same time, the results achieved for *Scenario 1* show a relatively stable loss for up to 4 GPUs with the effective batch size $B_{Ef} \leq 4 * 16 = 64$
- So, it is rational to set $B_{Ef} = 64$ for 8 and 16 GPUs as well, improving scalability considerably with the speedup of 6.4 and 11.58 for 8 and 16 GPUs, respectively
- These performance gains come at slightly higher loss but within the tolerance

Performance and accuracy evaluation of multi-GPU DNN training with BF16 data format

- The comparison of scalability achieved for different scenarios with BF16 on the multi-GPU system using NVIDIA GH200 chips



- While *Scenario 1* provides the best scalability, it delivers lower accuracy for configurations with 8 and 16 GPUs
- The opposite is true for *Scenario 2*, which provides the desired accuracy but at the cost of performance
- Finally, *Scenario 3* achieves a reasonable speedup for all numbers of GPUs while maintaining the desired accuracy

Performance-accuracy trade-off for TF32 format versus BF16 and FP32 formats

- Performance and accuracy achieved for training with TF32 and FP32 formats for $N_E = 50$ epochs and $B_S = 2$

		1 x GH200	2 x GH200	4 x GH200	8 x GH200	16 x GH200
TF32	**Time [s]**	1805	938	485	250	129
	Final loss	0.118	0.114	0.12	0.138	0.139
	Speedup	1	1.92	3.72	7.22	13.99
FP32	**Time [s]**	2256	1157	603	309	161
	Final loss	0.118	0.128	0.129	0.135	0.137
	Speedup	1	1.95	3.74	7.3	14.01

- For implementing mixed precision computations with TF32 format, we use `torch.backends` module
- Besides good scalability (speedup of about 14 times for 16 GPUs), the TF32 format provides better accuracy than BF16 for all numbers of GPUs
- The performance gap between TF32 and BF16 decreases when using more GPUs
  - For 16 GPUs, TF32 is only 1.17 times slower than BF16 format, with 1.09 times better accuracy
- Using TF32 reduces computation time by approximately 1.25 times compared to FP32 across the tested GPU counts, while maintaining accuracy behavior practically identical to the full precision FP32 solution

Conclusion and future work

- This work investigates the impact of diverse floating-point data formats - BF16, TF32, and FP32 - on the performance and accuracy of training CFD AI models on multi-GPU platforms
- Leveraging mixed precision based on the BF16 format on 16 GPUs allows us to accelerate training the model by about 11.6 times, preserving the same loss value as for a single GPU
- TF32 data format provides better accuracy than BF16, but requires more computational overheads
- The accuracy behavior of the mixed precision solution with TF32 is practically the same as that of the full precision option with FP32
  - At the same time, TF32 allows us to speed up the computations 1.25 times across the considered range of GPU numbers
- In the future, we plan to study the feasible methods of increasing the effective batch size more systematically without decreasing training accuracy
- Another direction of our future work is exploiting AI accelerators with alternative architecture, such as Intel Habana Gaudi 2 and Gaudi 3 platforms

Thanks for your attention :)