Distributed Element-Local Transformer for Scalable and Consistent Mesh-Based Modeling

Shyam Sankaran[*,2], Riccardo Balin[2], Bethany Lusch[2], and Shivam Barwey[3]

[1]*Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania,* shyamss@seas.upenn.edu
[2]*Argonne Leadership Computing Facility, Argonne National Laboratory, Lemont, 60439, IL, USA,* {rbalin, blusch}@anl.gov
[3]*Department of Aerospace and Mechanical Engineering, University of Notre Dame, 46556, IN, USA,* sbarwey@nd.edu

Abstract

The transformer architecture has become the state-of-the-art across numerous domains, yet its application to large-scale, unstructured mesh-based scientific simulations remains a significant challenge. In this work, we propose a novel transformer-based methodology that operates directly on massive, high-fidelity meshes typical of leadership-class computing. Our core contribution is a **consistent, element-wise local attention** mechanism. This formulation guarantees that the model's output is arithmetically invariant to the parallel decomposition of the mesh - a critical property for scientific applications by leveraging a halo-exchange mechanism within the attention layer. We demonstrate our approach on a large-scale 3D CFD problem with over 20 million nodes distributed across 96 GPUs.

1 Introduction

Transformer architectures [31] have revolutionized machine learning, establishing state-of-the-art performance in modalities ranging from natural language [1, 30] to computer vision [11]. This success has inspired a growing interest in applying transformers to scientific domains, with notable achievements in areas like protein folding [16] and weather forecasting [19]. However, a major frontier remains largely unexplored: applying transformers directly to the massive, unstructured, high-resolution meshes that are the bedrock of high-fidelity scientific simulations, such as those in computational fluid dynamics (CFD).

In fluid dynamics, which is the scope of this work, there is notable potential in leveraging transformers to develop surrogate and/or closure models that improve upon other architectural configurations [28, 20]. This is because (a) transformer architectures have been shown to benefit significantly from data scale and token counts across many different domains [17, 8], and (b) the fluid dynamics community readily curates massive datasets [25, 9, 3, 7] in a variety of engineering-relevant geometries/configurations through validated high-fidelity mesh-based physics simulation tools [12, 13, 6, 18, 15]. However, interfacing transformers directly with the large, unstructured, high-resolution meshes that are the bedrock of these high-fidelity simulations requires careful considerations and requirements. For example, practical training datasets are 3D in space (which translates to substantially increased mesh sizes), and may contain unstructured and/or adaptive grid discretizations incompatible with previously explored structured grid approaches [32, 24]. As such, for transformer-based training workflows to be practical, these requirements must be adequately addressed.

[*]Work done while on internship at Argonne National Laboratory.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2025.

When it comes to treating larger scales in terms of grid size (i.e., when considering meshes on the order of 10M degrees of freedom and beyond), one strategy is to invoke a reduction step such that information in the larger mesh is contained in a reduced latent space [2, 33]. Although such reduction steps are indeed promising from compression and surrogate modeling standpoints, we approach this problem from a different perspective, in that a transformer-based scalable training approach is developed to be compatible with the full mesh directly (i.e., no reduction steps). Specifically, we develop a transformer-based methodology designed to learn directly on high-resolution meshes, akin to the encode-process-decode paradigm of MeshGraphNets [27]. Our primary focus is on developing a modeling framework that is both expressive and scalable on leadership-class computing facilities (particularly ALCF's Aurora).

The central contribution of this paper is the design and analysis of a **consistent, element-wise local attention** mechanism (Fig. 1) – a foundational building block for scalable scientific transformers. Core to the approach is the concept of consistency, which has been explored previously in the context of graph neural networks [4], and is now developed here using transformer architectures. Here, a single large mesh is parallelized through domain decomposition, and an element-wise local attention is made consistent through the enforcement of non-local scatter operations, such that model training and inference routine are invariant to mesh partitioning. Extensions to a multi scale hierarchical transformer architecture leveraging mesh element tokenization with global attention is also explored in the Appendix A.

2 A Consistent Transformer for Mesh-Based Data

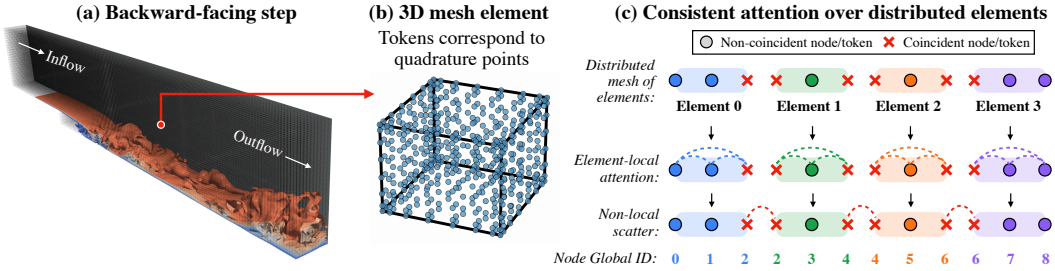

Figure 1: **The Consistent Local Attention Mechanism.** **(a)** Backward-facing step (BFS) configuration showing mesh and velocity magnitude contours. **(b)** Visualization of a single mesh element in the BFS. Each element is a 7th-order spectral element using Gauss-Lobato-Legendre (GLL) quadrature points [12] (512 total GLL points per element). These nodes correspond to tokens in our element-local attention framework. **(c)** Schematic of consistent attention using 4 elements in one dimension, and 3 total GLL points per element, for ease of illustration. In the full mesh, duplicate nodes exist at element boundaries (red markers). Our model reconciles these nodes by enforcing a consistent representation using Global IDs, achieved via on-rank averaging and cross-rank halo swaps if elements are distributed across multiple ranks.

A naive global attention mechanism over all nodes of a large-scale mesh is computationally infeasible. Our approach, as shown in Figure 1 is therefore guided by the underlying structure of the simulation data, which is generated using high-order spectral element methods. This discretization partitions the domain into a set of non-overlapping elements, but a key artifact is that adjacent elements both place nodes on their shared boundaries. This results in multiple, distinct nodes in the data structure that correspond to a single physical point in space.

Leveraging this element-based structure, we restrict the attention mechanism to be local, operating only among the nodes within each element. This locality, however, necessitates a mechanism to handle the duplicate nodes at the interfaces. Without it, nodes at the same physical point could develop different feature representations, violating physical consistency. We therefore introduce a **consistency mechanism** as a core part of our architecture. Furthermore, we deliberately depart from the conventional graph transformer paradigm, which often incorporates explicit edge features. Instead, we treat the nodes within each element as an unstructured point set, forming a **mesh-based transformer** to investigate if the model can learn the necessary geometric information guided primarily by positional embeddings. This architectural choice presents a key trade-off. On one hand,

it simplifies the data pipeline by avoiding the need to construct and store a massive edge index and its associated geometric features. On the other hand, it removes the strong geometric inductive bias that edge features provide.

Element-Wise Local Attention. The core of our model is a transformer layer where attention is computed only among the nodes belonging to the same spectral element. This captures complex local interactions and is computationally efficient. To encode the crucial geometric information without explicit edge features, we employ an Axial Rotary Position Embedding (RoPE) [29], which injects relative positional information into the attention scores while preserving translational equivariance.

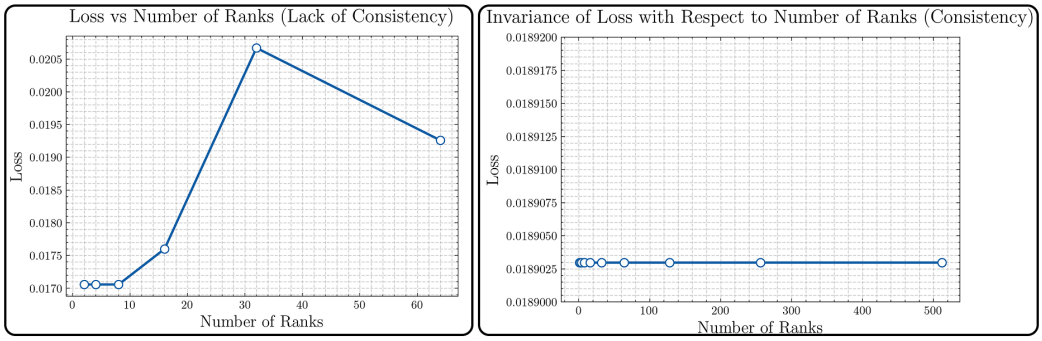

Figure 2: **Verification of Consistency. (Left):** In the lack of a consistency enforcing mechanism, the loss observed isn't invariant to the number of ranks utilized. **(Right):** The model's loss is invariant to the number of ranks when consistency is enforced.

Consistency via Reconciliation. To ensure the model is invariant to the parallel decomposition, we adapt the consistency-enforcing mechanism from Ref. [4] for our transformer architecture. Following each local attention update, a reconciliation step enforces that nodes corresponding to the same physical point have an identical feature representation. This is achieved by averaging their feature vectors through a two-stage process:

1. **On-Rank Aggregation:** Averaging features of duplicate nodes that reside on the same MPI rank.
2. **Cross-Rank Aggregation:** Using a **halo swap** mechanism to communicate and average features of duplicate nodes that are distributed across different MPI ranks.

This consistency mechanism is the key that allows a local attention model to function correctly and robustly on a large, distributed domain. To empirically verify this property, we evaluated the model's loss on an identical input snapshot, decomposed across a varying number of MPI ranks. As shown in Figure 2, the loss remains arithmetically identical up to machine precision, regardless of the number of ranks used. This result confirms that our halo-swap-based reconciliation mechanism successfully enforces invariance to the parallel decomposition.

3 Experimental Setup

Dataset and Task. Our experiments focus on surrogate modeling of the **3D backward-facing step**, a canonical configuration containing complex unsteady turbulent flow separation and reattachment physics. Data is generated using the NekRS solver [12] for a flow at a Reynolds number of $Re = 1600$; a visualization is provided in Fig. 1(a). The full simulation was run to generate 2400 total snapshots, each separated by a time step of $\Delta t = 8 \times 10^{-3}$. We utilize the first 2000 snapshots for training, and reserve the final 400 snapshots as a held-out test set. The mesh is discretized using 46,480 spectral elements with a polynomial order of $p = 7$ (512 nodes per element, see Fig. 1(b)), resulting in a graph with over 23 million nodes (Ref. [5] contains a more detailed description of the simulation setup). Here, the learning task is to predict the next-step velocity field ($\mathbf{u}_{t+1}$) given the current velocity field ($\mathbf{u}_t$) at each node. For training, all input and target velocity fields are

normalized (z-scored) using the mean and standard deviation computed *only* from the 2000 training snapshots. However, we emphasize that the concept of consistency carries to other modeling tasks, such as closure modeling.

Implementation Details. The model is implemented in PyTorch [26, 21] and trained on 96 GPUs of the Aurora supercomputer. We use the AdamW optimizer [23] with a cosine annealing learning rate schedule and a batch size of one snapshot sharded across all GPUs.

4 Results and Analysis

Our primary goal is to establish the viability of our new architecture and analyze its behavior. To this end, we compare our consistent local attention transformer against the baseline consistent MPNN of Ref. [4]. We conducted a series of studies on the transformer's performance with respect to its width, depth, and the choice of base frequency (λ) used in its rotary position embeddings (RoPE), with detailed experimental parameters provided in Appendix B.

Figure 3 summarizes our findings. The results clearly demonstrate that the transformer architecture successfully learns the underlying physical behavior, achieving a stable and low error that improves with model scale (width and depth). However, across all configurations, the model does not yet outperform the highly-specialized MPNN baseline. The most critical insight comes from the model's high sensitivity to the RoPE base frequency, λ. This hyperparameter, which governs how the model implicitly perceives the mesh geometry, has a significant impact on the final converged loss, a sensitivity not present in the MPNN which uses an explicit geometric bias via hard-coded edge features.

This finding highlights that the path to unlocking the transformer's full potential in this domain requires more than just scaling parameters. It points to a need for a more fundamental exploration of architectural choices that can better capture the underlying physics. We identify two key avenues for future work: (1) explicitly capturing global behavior through the hierarchical summary token mechanism detailed in Appendix A, and (2) improving the model's geometric reasoning by investigating stronger inductive biases, such as alternative positional encodings or hybrid architectures that re-introduce explicit edge features.

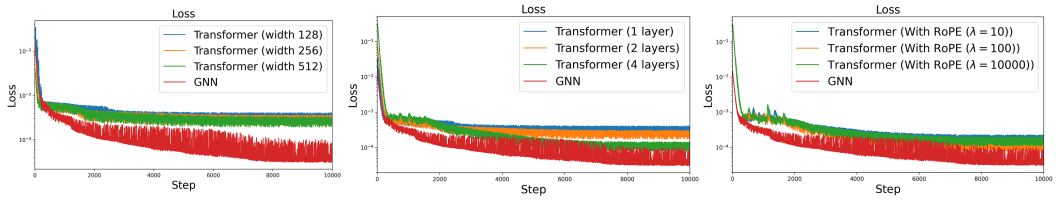

Figure 3: **Training Loss Comparison and Ablation Studies.** The training loss for the Consistent Local Attention Transformer is shown for varying width (left), depth (middle), and RoPE base frequency λ (right), compared against the baseline consistent MPNN (red). While the transformer learns effectively, it does not yet surpass the baseline and shows high sensitivity to its geometric encoding via RoPE.

4.1 Autoregressive Rollout Evaluation on Test Set

To address the models' practical stability and test performance, we conducted an autoregressive rollout evaluation. This experiment directly answers the question of how error accumulates when the model is used to generate a simulation over time, a key criticism of 1-step-ahead metrics. We used a starting snapshot from the held-out test set (comprising 400 snapshots) and tasked the models to iteratively predict the next 10 consecutive time steps. For each step, the model's prediction was fed back in as the input for the next step.

We compared our best-performing transformer configuration (4-layer, 128-width, $\lambda = 100$) against the baseline consistent MPNN. We report the MSE for 5-step rollouts and 10-step rollouts. As detailed in Table 1, we observe that both models achieve very similar performance on the held-out test set. Although, the transformer model consistently shows a marginal improvement over the baseline.

We hypothesize that this slight advantage may be attributed to the consistency mechanism, which aids in smoothing predictions during the rollout. The precise interplay between this mechanism and the model's architecture presents an interesting avenue for future investigation.

Table 1: Autoregressive rollout error (normalized MSE) on the held-out test set. Results compare the baseline MPNN against our best-performing Consistent Local Attention Transformer.

Model	**5-Step Rollout MSE**	**10-Step Rollout MSE**
Consistent MPNN (Baseline)	0.004878	0.016596
Consistent Transformer (Ours)	0.004799	0.016343

5 Conclusion and Future Work

In this work, we introduced a foundational architecture for applying transformers to massive, distributed scientific simulations: a **consistent, element-wise local attention** mechanism. We have demonstrated that this model is scalable and successfully learns complex fluid dynamics on a high-fidelity mesh, validating its viability as a new approach for scientific surrogate modeling. Our findings suggest a clear forward. The architectural avenues identified in our analysis-namely, improving geometric encoding and developing the hierarchical global attention layer detailed in Appendix A. They represent a path toward unlocking the transformer's full potential for scientific problems that have historically been challenging for local models. For instance, a hierarchical model capable of explicitly capturing long-range dependencies could enable accurate predictions over much larger simulation time steps or from sparse sensor data, scenarios where the global state is paramount. Ultimately, this work serves as a critical first step, providing both a scalable and consistent architectural building block and a roadmap for developing a new class of powerful, general-purpose transformers for scientific computing on leadership-class systems. The code for this work is publicly available at: https://github.com/argonne-lcf/nekRS-ML.

Acknowledgments

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. SB acknowledges support by the U.S. Department of Energy (DOE), Office of Science under contract DE-AC02-06CH11357

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers. *CoRR*, 2024.

[3] Riccardo Balin and Kenneth E Jansen. Direct numerical simulation of a turbulent boundary layer over a bump with strong pressure gradients. *Journal of Fluid Mechanics*, 918:A14, 2021.

[4] S. Barwey, S. Patel, R. Balin, et al. Scalable and Consistent Graph Neural Networks for Distributed Mesh-based Data-driven Modeling. In *SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2024.

[5] Shivam Barwey, Pinaki Pal, Saumil Patel, Riccardo Balin, Bethany Lusch, Venkatram Vishwanath, Romit Maulik, and Ramesh Balakrishnan. Mesh-based super-resolution of fluid flows with multiscale graph neural networks. *Computer Methods in Applied Mechanics and Engineering*, 443:118072, 2025.

[6] Ral Bielawski, Shivam Barwey, Supraj Prakash, and Venkat Raman. Highly-scalable gpu-accelerated compressible reacting flow solver for modeling high-speed flows. *Computers & Fluids*, 265:105972, 2023.

[7] Florent Bonnet, Jocelyn Mazari, Paola Cinnella, and Patrick Gallinari. Airfrans: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. *Advances in Neural Information Processing Systems*, 35:23463–23478, 2022.

[8] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2818–2829, 2023.

[9] Wai Tong Chung, Bassem Akoush, Pushan Sharma, Alex Tamkin, Ki Sung Jung, Jacqueline Chen, Jack Guo, Davy Brouzet, Mohsen Talei, Bruno Savard, et al. Turbulence in focus: Benchmarking scaling behavior of 3d volumetric super-resolution with blastnet 2.0 data. *Advances in Neural Information Processing Systems*, 36:77430–77484, 2023.

[10] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

[11] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[12] Paul Fischer, Stefan Kerkemeier, Misun Min, Yu-Hsiang Lan, Malachi Phillips, Thilina Rathnayake, Elia Merzari, Ananias Tomboulides, Ali Karakus, Noel Chalmers, et al. Nekrs, a gpu-accelerated spectral element navier–stokes solver. *Parallel Computing*, 114:102982, 2022.

[13] Marc T Henry de Frahan, Jon S Rood, Marc S Day, Hariswaran Sitaraman, Shashank Yellapantula, Bruce A Perry, Ray W Grout, Ann Almgren, Weiqun Zhang, John B Bell, et al. Pelec: An adaptive mesh refinement solver for compressible reacting flows. *The International Journal of High Performance Computing Applications*, 37(2):115–131, 2023.

[14] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[15] James R Wright III, Valeria Barra, Jed Brown, Layla Ghaffari, Kenneth Jansen, Jeremy L. Thompson. Honee: High-order navier-stokes equation evaluator.

[16] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

[17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[18] Marius Kurz, Daniel Kempf, Marcel P Blind, Patrick Kopper, Philipp Offenhäuser, Anna Schwarz, Spencer Starr, Jens Keim, and Andrea Beck. Galæxi: Solving complex compressible flows with high-order discontinuous galerkin methods on accelerator-based systems. *Computer Physics Communications*, 306:109388, 2025.

[19] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[20] Haochen Li, Jinhan Xie, Chi Zhang, Yuchen Zhang, and Yaomin Zhao. A transformer-based convolutional method to model inverse cascade in forced two-dimensional turbulence. *Journal of Computational Physics*, 520:113475, 2025.

[21] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020.

[22] Z. Liu, S. Li, Z. Wang, et al. Ring Attention with Blockwise Transformers for Near-Infinite Context. arXiv preprint arXiv:2310.01889, 2024.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[24] Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. Multiple physics pretraining for spatiotemporal surrogate models. *Advances in Neural Information Processing Systems*, 37:119301–119335, 2024.

[25] Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[27] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409, 2020.

[28] Marcial Sanchis-Agudo, Yuning Wang, Roger Arnau, Luca Guastoni, Jasmin Lim, Karthik Duraisamy, and Ricardo Vinuesa. Easy attention: A simple attention mechanism for temporal predictions with transformers. *arXiv preprint arXiv:2308.12874*, 2023.

[29] J. Su, Y. Lu, S. Pan, et al. Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864, 2021.

[30] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Sifan Wang, Jacob H Seidman, Shyam Sankaran, Hanwen Wang, George J Pappas, and Paris Perdikaris. Cvit: Continuous vision transformer for operator learning. *arXiv preprint arXiv:2405.13998*, 2024.

[33] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

[34] Junqi Yin, Mijanur Palash, M Paul Laiu, Muralikrishnan Gopalakrishnan Meena, John Gounley, Stephen M Kops, Feiyi Wang, Ramanan Sankaran, and Pei Zhang. Pixel-resolved long-context learning for turbulence at exascale: Resolving small-scale eddies toward the viscous limit. *arXiv preprint arXiv:2507.16697*, 2025.

A Appendix: Future Work with Hierarchical Attention

To better capture global correlations, we propose extending our model to a hierarchical architecture. After a local attention layer, element-level summary tokens would be generated using perceiver-style aggregation [14]. A second, global attention mechanism would then operate on these tokens. Finally, we'll have a readout to the local nodes using this updateed representation of the summary

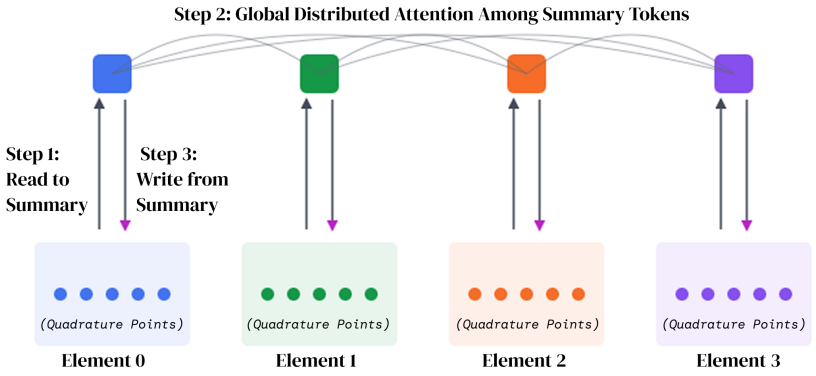

Figure 4: **Proposed Hierarchical Architecture.** The base layers perform consistent local attention on nodes. The features are then combined to create **summary tokens**. A second, global transformer performs attention on these tokens, allowing for direct, long-range communication between distant elements. Finally, we readout an update to the local representations using the updated summary tokens.

using cross-attention. While this hierarchical design, shown in Figure 4, is conceptually powerful, its implementation on distributed, leadership-class systems presents significant engineering challenges.

A naive approach would be to gather all summary tokens from every rank onto each rank using an `AllGather` collective. However, this is not scalable, as the memory required on each GPU would grow linearly with the total number of ranks, quickly becoming a bottleneck. The ideal solution is a communication-efficient algorithm like ring attention [22], which theoretically offers a path to near-infinite sequence lengths.

However, translating this theory into practice on a machine like Aurora reveals several intricacies. First, ring attention relies heavily on efficient, point-to-point communication primitives. On many HPC systems, collective operations (like `AllGather` or `AllReduce`) are highly optimized, while point-to-point communication can have higher latency and lower effective bandwidth, creating an unexpected performance bottleneck. Second, our implementation must handle an unbalanced load, where different ranks may have a varying number of elements and thus summary tokens. Finally, the peak performance of algorithms like ring attention often relies on fused kernels like flash attention [10]. Due to the current lack of a production-ready flash attention interface for Intel GPUs, our implementation must materialize the full attention matrix and use a separate log-sum-exp operation, which leads to further performance degradation compared to a fused kernel. We are also looking into alternative formulations of ring attention that are optimized for HPC systems to overcome these limitations [34].

As shown in Figure 5, our benchmarks highlight these challenges. While our Ring Attention implementation successfully scales to 96 GPUs, its performance is slower than a single-GPU, memory-bound flash attention implementation, and is significantly impacted by the overheads discussed. This analysis underscores that deploying advanced, theoretically optimal algorithms on real-world HPC systems requires careful co-design and optimization to overcome hardware and software-specific bottlenecks.

B Training Details

This section details the specific model configurations and training hyperparameters used for the experiments presented in the main text.

B.1 Common Training Hyperparameters

All experiments were conducted under a consistent training setup to ensure a fair comparison.

- **Hardware:** All models were trained on 96 GPUs of the Aurora supercomputer at Argonne National Laboratory.

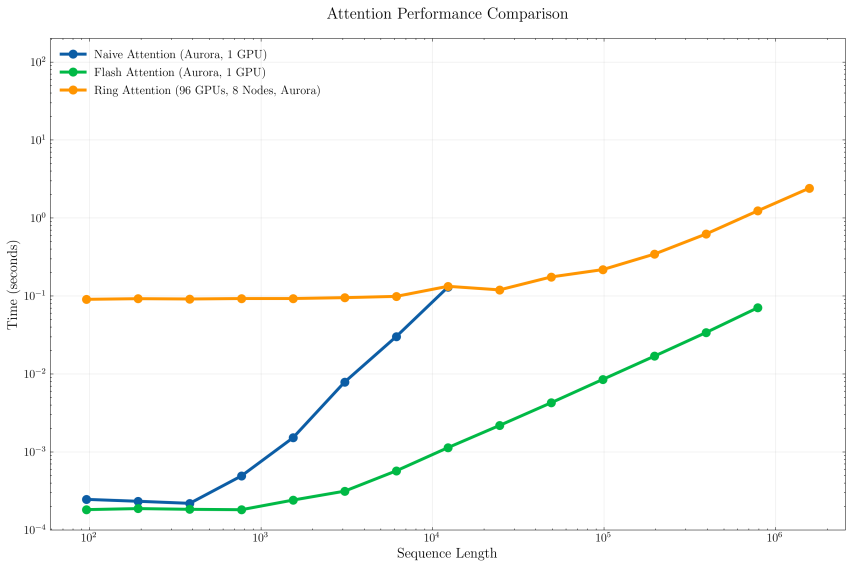

Figure 5: **Performance of Distributed Attention Mechanisms.** While our Ring Attention implementation (orange) scales across 96 GPUs, it is slower than a theoretical single-GPU Flash Attention (green) due to communication overheads and the lack of a fused kernel on our target hardware.

- **Software:** Models were implemented in PyTorch and PyTorch Geometric. Distributed training was managed using MPI and PyTorch's Distributed Data Parallel (DDP) framework.
- **Optimizer:** We used the AdamW optimizer with a weight decay of 10^{-4}.
- **Learning Rate:** We employed a custom three-phase schedule. The schedule consists of an initial phase of 1000 steps at a learning rate of 1×10^{-3}, followed by a decay phase of 7500 steps where the rate is linearly annealed down to 1×10^{-6}, and a final fine-tuning phase of 1500 steps at a constant 1×10^{-6}.
- **Batch Size:** All experiments used a global batch size of one snapshot per training step, sharded across all 96 GPUs.
- **Training Steps:** All models were trained for a total of 10,000 steps.

B.2 Model Architecture Configurations

Table 2 provides the architectural details for the baseline consistent MPNN and the various configurations of our consistent local attention transformer (T) used in our studies. The ablation studies were designed to isolate the effect of each hyperparameter. For the width study, the model depth was fixed at one layer. For the depth study, the width was fixed at 128. The sensitivity analysis of the RoPE base frequency (λ) was conducted on our largest configuration (4 layers, 128-width) to best assess its impact on a capable model.

Table 2: Model configuration details for experimental studies.

Parameter	**Baseline MPNN**	**T (Width Study)**	**T (Depth Study)**	**T (RoPE Study)**
Hidden Channels (Width)	128	128 / 256 / 512	128	128
Processing Layers (Depth)	4	1	1 / 2 / 4	4
RoPE Base Freq. (λ)	N/A	100	100	10 / 100 / 10000
Attention Heads	N/A	4	4	4