# Workload-Aware Networks for Machine Learning

Weiyang “Frank” Wang

I design network systems that integrate with machine learning (ML) workloads to enhance performance and reduce costs. Modern ML workloads consume massive computational resources at unprecedented scale: training frontier models like LLaMA-3 requires hundreds of GPU-years across sixteen thousand accelerators [1], while inference serving reaches 400 million queries daily for a single service [2]. However, existing datacenter networks waste resources: communication overhead consumes 60% of training time for Meta’s production models, while 40% of network capacity sits idle during Large Language Model (LLM) training [3, 4]. This inefficiency stems from treating ML traffic as unpredictable and bursty—assumptions inherited from legacy datacenter workloads. **My research reveals and harnesses the structure of ML workloads to design networks that improve performance and reduce cost.**

I characterize ML workload’s communication patterns and computational dependencies, revealing ML-specific optimization opportunities missed by general-purpose networking. My work spans the entire network stack through three thrusts:

- **Redesigning network architectures for ML training.** I designed the Rail-only network, which eliminates unnecessary links to **cut network costs by 37-75% without sacrificing performance** [3, 5]. Building on this, TopoOpt co-optimizes network topology with model parallelization, achieving up to **3.4× higher training throughput** than traditional datacenter networks [4].
- **Exposing network dynamics as primary entities** to empower these new architectures. I developed **the first Linux packet scheduler for direct-connect reconfigurable networks** and demonstrated the first ML training on periodically reconfigured networks for a joint DARPA-industry grant. I also co-led TDTCP, a transport protocol that adapts to dynamic network infrastructures, delivering **24-41% throughput improvement** [6].
- **Embedding application logic into datapaths** based on workload properties. I co-led Checkmate [7], which produces **5-34.5× more frequent recovery snapshots without performance overhead** for ML training by mirroring gradients in the network to dedicated clusters that replay optimizer updates. I am also mentoring a project on schedulable packet processing to maintain performance and fairness as dataplane logic grows increasingly complex.

My work has influenced both production systems and ongoing research. The Rail-only architecture has been **adapted in production settings**, such as Juniper’s Apstra 6.0 and Broadcom’s Tomahawk-6 [8, 9]. I built the **first real-world reconfigurable GPU cluster** at MIT for TopoOpt, now being evaluated for deployment at Meta [4]. Both Rail-only and TopoOpt were featured in Meta’s SIGCOMM 2025 Networks for AI Computing workshop keynote on cross-layer innovations for AI [10] and are referenced in recent work from Alibaba, Bytedance, Meta, and Google on large-scale ML training [11–14]. These production adoptions validate that workload-aware network co-design delivers practical value. As ML workload scale and diversify, my future research aims to develop principled approaches for networks and workloads to co-evolve, unlocking performance and efficiency gains that neither can achieve in isolation.

## Network Architectures for ML Training

Large-scale ML training relies on a high-speed fabric to connect tens of thousands of GPUs. Traditional network architectures, such as the Fat-Tree, dominate the landscape. Yet, these fabrics are costly and impractical: building a Fat-Tree fabric that connects 64K GPUs requires over $300 million in network equipment and consumes more than six million watts of peak power. In my work, I demonstrate that such connectivity is not only impractical but also unnecessary. I characterized three properties of ML training traffic:

1. **Predictable**: given the parallelization strategy and operator placement, the communication pattern of a training iteration will be deterministic.

2. **Repetitive**: since ML training is an iterative process, the same communication patterns will occur in each iteration for thousands or even more iterations.
3. **Mutable**: a majority of ML training traffic belongs to the class of collective communication, whose traffic pattern can change without impacting the training.

Based on these characterizations, I propose new network architectures tailored to ML's specific needs:

- **Rail-only networks [3, 5].** I discovered that optimal LLM training exhibits predictable communication patterns: LLM training clusters need high bandwidth within small GPU groups, while inter-group traffic is sparse and follows fixed patterns. This inter-group traffic does not require the expensive any-to-any connectivity provided by Fat-Tree networks, as GPU manufacturers advocate. I designed Rail-only to provide the minimum network connections needed to support these traffic patterns at full speed. By removing the top layer of switches (the "spine") and connecting only GPU pairs with significant traffic, Rail-only cuts network costs while maintaining performance. I also developed schedules for workloads such as mixture-of-experts that require frequent, small all-to-all communication.

  I formulated an analytical model to predict LLM training iteration time and empirically validated it. Using my model, I showed that Rail-only reduces network cost by 38-77% and power by 37-75% while maintaining training throughput within 12% of full Fat-Tree networks. The Rail-only network architecture has now become a standard design for LLM training clusters, with industry support from Juniper Networks and Broadcom (major networking equipment vendors) [8, 9]. It serves as a point of reference for recent network architecture proposals in top networking conferences, such as SIGCOMM, referenced by Alibaba and ByteDance [11, 12].

- **Topology-parallelization co-optimization [4, 15].** While Rail-only demonstrated the value of workload-specific design, it still lacks flexibility as workloads evolve. I took the idea of adapting the fabric to the workload further with TopoOpt, leveraging a reconfigurable network. TopoOpt draws on the predictability: it plans a Deep Neural Network (DNN) training job's parallelization strategy and an accompanying topology once, configures the network accordingly, and then keeps that plan for the job's lifetime. The insight behind TopoOpt is that the most common traffic type in ML training, collective communications, is fundamentally flexible. For example, the "AllReduce" operation (averaging gradients across GPUs) can be implemented through various traffic patterns, each stressing the network differently. TopoOpt exploits this mutability to adapt the network topology to each training job's specific communication patterns, ensuring diverse workloads all achieve high performance.

  At MIT, I built the first GPU cluster with a reconfigurable network connecting 12 GPUs via optical circuit switches and trained state-of-the-art DNN models onh this testbed. Our testbed validated that optimized topologies work with real ML frameworks and collective libraries. In large-scale simulations, TopoOpt achieved up to 3.4× faster training than Fat-Tree networks at similar cost. The approach has inspired follow-up collaborations at UW and Raytheon on collective schedule-topology co-design, establishing workload-network co-design as a new research direction [15].

# Exposing Network Dynamics to Network Stacks

The reconfigurable networks I built for ML require adaptations in the network stack to recognize network changes as first-class entities. Traditional network stacks assume static topologies and treat changes as failures to recover from. I developed systems that expose topology dynamics as a programmable resource rather than hiding them behind legacy abstractions.

- **Reconfiguration-aware kernel interface for direct-connect topologies [in submission].** While TopoOpt found one optimal topology for each ML training job, more frequent reconfiguration could adapt to instantaneous workload demands. Prior work on reconfigurable networks focuses solely on switch-level reconfiguration, but in GPU clusters where hosts connect directly to the reconfigurable fabric, the operating system also needs mechanisms to handle dynamic topology changes. To bridge this gap, I built the first Linux packet scheduler for direct-connect reconfigurable networks. My Linux packet scheduler explicitly adapts to network reconfiguration events, allowing applications like training jobs to synchronize computation with network availability. I also extended Linux with new kernel mechanisms to track and handle topology changes in real-time.

This work is a core component of the CUbiC center in the SRC JUMP 2.0 program, cofunded by the semiconductor industry and DARPA. I led one of three demos at CUbiC's 2024 annual review, demonstrating the first-ever ML training on a network that periodically reconfigures. In 2025, I co-led another demo with collaborators from Columbia University, integrating my scheduler with their spatial-wavelength-selective switch and achieving 25 Gbps data transmission and end-to-end multi-tenant DNN training. This work paves the way for reconfigurable networks to adapt to more dynamic ML workloads like inference, and is in submission to OFC, a leading conference in optical networking.

- **Congestion control for dynamic networks [6].** Reconfigurable networks also require rethinking congestion control, the mechanism that regulates sending rates to prevent network overload. Traditional congestion control protocols react to network changes by probing for available bandwidth; in reconfigurable networks, however, paths change every few microseconds. This reactive approach means senders are always behind, constantly converging to a bandwidth that is already outdated.

  In a project I co-led with collaborators from CMU and UCSD, we designed TDTCP to address congestion control in dynamic bandwidth environments. Our framework leverages a priori knowledge of network topology schedules, eliminating the need for rapid bandwidth convergence. TDTCP keeps a separate congestion state for each recurring path and switches to the correct state the instant the network reports a path change with top-of-rack switches. The effect is to turn a rapidly changing channel into a sequence of short, predictable regimes that the transport layer models explicitly, resulting in a throughput improvement of 24-41%.

# Embedding Workload Logic into Datapaths

A crucial approach to workload-aware networking is to embed application logic directly into network datapaths, enabling networks to actively participate in computation. I leverage ML workloads' computational dependencies to embed training functions into the network, and design new mechanisms to maintain performance and fairness as dataplane logic becomes increasingly complex.

- **Checkmate: network-assisted fault tolerance [7].** Checkpointing refers to saving training states for recovery after a failure, which occurs frequently at scale—Meta reported 419 interruptions during LLaMA 3's 54-day pre-training [1]. Traditional checkpointing pauses training to copy state to storage, forcing a tradeoff: frequent checkpoints waste GPU cycles, while infrequent ones risk massive recomputation. I co-led Checkmate, a checkpointing system that eliminates this tradeoff by replicating gradients as they traverse the network, enabling iteration-level checkpoints without performance overhead.

  I recognized that the network already has everything needed to maintain a checkpoint: ML algorithms follow deterministic dependency graphs in which gradients must be aggregated, giving the network complete visibility into the reduced gradients that determine the next model state. Checkmate maintains a live model replica by mirroring gradient traffic to a shadow cluster that applies identical optimizer updates every training iteration. This transforms checkpointing from an expensive interruption into a zero-performance-overhead background operation. Checkmate achieves iteration-level recovery granularity while matching the throughput of systems with checkpointing disabled, achieving the theoretical maximum throughput and recoverability at the same time.

- **Schedulable ingress packet processing [work-in-progress].** eXpress Data Path (XDP) embeds workload logic into datapaths by enabling programmable packet processing directly in the Linux kernel, immediately upon packet arrival. Datacenters increasingly rely on XDP for complex tasks like load balancing and DDoS mitigation. However, Linux treats all packet processing as uniform, invisible overhead—an assumption that breaks when XDP programs perform heavy computation, causing head-of-line blocking and unfairness.

  I am mentoring a student at MIT who is leading a project to make XDP programs schedulable by introducing preemption points into their execution. This new mechanism allows the kernel to account for computational costs and interrupt processing when necessary, maintaining fairness and low-latency guarantees. Our preliminary implementation preserves XDP's performance while reducing tail latencies by orders of magnitude. We plan to submit this work to OSDI 2026, a top venue in operating systems.

# Future Work

My research has demonstrated that harnessing workload properties, from traffic patterns to computational dependencies, enables performance gains with less cost in AI infrastructure. Building on this foundation, my research group will advance network and workload co-design across three timescales: deploying immediate optimizations for emerging AI applications (1-2 years), developing full-stack orchestrators for systematic co-optimization (3-5 years), and creating autonomous network infrastructures that self-architect through reasoning AI models (5-10 years).

## Short-Term Projects: Network for Emerging AI Applications

My group's immediate focus will address critical gaps in current AI infrastructure that traditional networks fail to address. For instance, modern inference serving exhibits volatile patterns, including fluctuating request rates, dynamic attention sparsity, and cascading dependencies in agentic systems. Another popular workload, multi-tenant training, suffers from destructive interference between hundreds of concurrent training jobs. My group will build upon my past work, leveraging the workload structure to address challenging scenarios. Specifically, we will develop reconfigurable networks that pre-adapt to predicted inference patterns, and create coordination mechanisms that expose and compose network capabilities across multiple training jobs. The key insight remains consistent with my prior work: by exploiting predictable phases within requests and repetitive patterns across jobs, we will eliminate the inefficiencies that limit today's one-size-fits-all networks. These problems directly impact the industry's ability to scale AI services cost-effectively, and I will pursue industry funding partnerships to accelerate the deployment of these solutions in production environments.

## Mid-Term Plan: Orchestrator for Network–Application Co-optimization

My past research has demonstrated the power of harnessing workload structure across the network stack, yet translating these insights to new workloads and adapting new hardware technologies remains labor-intensive. Our research group will aim to develop an orchestrator that provides a unified interface through domain-specific languages (DSLs). Through this interface, workloads can express their computation and communication requirements (e.g., collective patterns, dependency graphs), while networks can advertise their capabilities (e.g., reconfigurability, programmability). For instance, a multi-model retrieval system could declare its query types, while future switches could advertise in-network caching capabilities. The orchestrator would then automatically match these properties using model- or learning-based optimization and configure caching at the switch level, without manual end-to-end implementation. Key technical challenges include semantic matching between high-level workload specifications and low-level network primitives, and designing abstractions that are general enough to express evolving workload patterns yet specific enough to unlock concrete optimization strategies. By treating workload requirements and network capabilities as well-defined declarations, we enable composing optimizations across different pairings rather than rebuilding from scratch each time. To support this five-year vision, I plan to pursue NSF and CNS funding, in addition to industry funding, and leverage the strong connections I've built with the SRC research community through my prior work, ensuring sustained support for this research agenda.

## Long-Term Vision: Autonomous Network Infrastructures

As ML models acquire increasingly strong reasoning capabilities, networks will take this advantage and evolve from passive infrastructure into autonomous intelligent systems that co-evolve with the workloads they serve. This represents the natural culmination of my research trajectory: from exploiting known patterns, to enabling bidirectional workload-network exchange, to ultimately creating self-architecting networks that discover optimization opportunities. If successful, my research will transform networks to employ agentic systems as their control plane, directly ingesting high-level objectives (e.g., "minimize training time within power budget") and automatically deriving optimal topologies, protocols, and scheduling strategies through continuous learning. The network becomes truly autonomous, observing performance, hypothesizing improvements, implementing changes, and learning from outcomes at speeds that surpass those of human operators. The technical challenges are substantial, such as ensuring stability during self-modification and debugging distributed protocols. Addressing them will define the next generation of AI infrastructure, positioning my research at the forefront of making intelligent, self-managing systems a reality.

# References

[1] A. Grattafiori *et al.*, "The Llama 3 Herd of Models," *arXiv*, 2024.

[2] D. Cheney, "How GitHub Copilot Serves 400 Million Completion Requests a Day." `https://www.infoq.com/presentations/github-copilot/`, 2025. QCon San Francisco 2024, InfoQ.

[3] **W. Wang**, M. Ghobadi, K. Shakeri, Y. Zhang, and N. Hasani, "Rail-only: A Low-Cost High-Performance Network for Training LLMs with Trillion Parameters," in *2024 IEEE Symposium on High-Performance Interconnects (HOTI)*, IEEE, 2024.

[4] **W. Wang**, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, and A. Kewitsch, "TopoOpt: Co-optimizing Network Topology and Parallelization Strategy for Distributed Training Jobs," in *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23)*, pp. 739–767, 2023.

[5] **W. Wang** and M. Ghobadi, "Spine-free networks for large language model training," *IEEE Micro*, vol. 45, no. 2, pp. 18–25, 2025.

[6] S. S. Chen*, **W. Wang***, C. Canel, S. Seshan, A. C. Snoeren, and P. Steenkiste, "Time-Division TCP for Reconfigurable Data Center Networks," in *Proceedings of the 2022 ACM SIGCOMM Conference (SIGCOMM '22)*, (New York, NY, USA), pp. 19–35, ACM, 2022. *Equal contribution.

[7] A. Bhardwaj*, **W. Wang***, J. Carin, A. Belay, and M. Ghobadi, "Checkmate: Zero Performance Overhead Model Checkpointing via Network Gradient Replication," in *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '26)*, 2026. to appear. *Equal contribution.

[8] Juniper Networks, "Templates — Apstra 6.0 — Rail-Collapsed (Rail-Only) AI Fabric Design." `https://www.juniper.net/documentation/us/en/software/apstra6.0/apstra-user-guide/topics/concept/templates.html`, 2025. Product documentation.

[9] Broadcom Inc., "Broadcom Ships Tomahawk 6: World's First 102.4 Tbps Switch." `https://investors.broadcom.com/news-releases/news-release-details/broadcom-ships-tomahawk-6-worlds-first-1024-tbps-switch`, June 2025. Press release.

[10] Y. Zhang, "Cross-Layer Innovations in Network Design for AI at Meta." `https://www.youtube.com/watch?v=4Dg006cZqHQ`. ACM SIGCOMM Second workshop on Networks for AI Computing (NAIC) 2025.

[11] K. Qian, Y. Xi, J. Cao, J. Gao, Y. Xu, Y. Guan, B. Fu, X. Shi, F. Zhu, R. Miao, C. Wang, P. Wang, P. Zhang, X. Zeng, E. Ruan, Z. Yao, E. Zhai, and D. Cai, "Alibaba HPN: A Data Center Network for Large Language Model Training," in *Proceedings of the ACM SIGCOMM 2024 Conference*, (Sydney, NSW, Australia), ACM, 2024.

[12] Z. Yan, D. Li, L. Chen, D. Xiong, K. Gao, Y. Zhang, R. Yan, M. Zhang, B. Zhang, Z. Jiang, J. Ye, and H. Lin, "From atop to zcube: Automated topology optimization pipeline and a highly cost-effective network topology for large model training," in *Proceedings of the ACM SIGCOMM 2025 Conference*, SIGCOMM '25, (New York, NY, USA), p. 861–881, Association for Computing Machinery, 2025.

[13] A. Gangidi, R. Miao, S. Zheng, S. J. Bondu, G. Goes, H. Morsy, R. Puri, M. Riftadi, A. J. Shetty, J. Yang, S. Zhang, M. J. Fernandez, S. Gandham, and H. Zeng, "Rdma over ethernet for distributed training at meta scale," in *Proceedings of the ACM SIGCOMM 2024 Conference*, ACM SIGCOMM '24, (New York, NY, USA), p. 57–70, Association for Computing Machinery, 2024.

[14] N. Jouppi, G. Kurian, S. Li, P. Ma, R. Nagarajan, L. Nai, N. Patil, S. Subramanian, A. Swing, B. Towles, C. Young, X. Zhou, Z. Zhou, and D. A. Patterson, "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, (New York, NY, USA), Association for Computing Machinery, 2023.

[15] L. Zhao, S. Pal, T. Chugh, **W. Wang**, J. Fantl, P. Basu, J. Khoury, and A. Krishnamurthy, "Efficient Direct-Connect Topologies for Collective Communications," in *Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI '25)*, pp. 705–737, 2025.