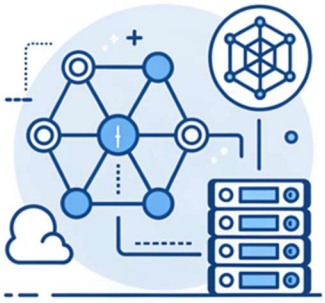
SCALA COMPUTING



Beyond GPU Speed: Understanding Network Bottlenecks in Distributed LLM Training

As Large Language Models (LLMs) become central to AI innovation, organizations face unprecedented demands on their data center infrastructure. Scala Computing provides cutting-edge network simulation tools to model, analyze, and optimize the network behavior of LLM workloads at scale. Our platform supports high-fidelity simulations using Chakra trace files which capture representation of real-world AI workloads including the detailed dependencies within and between GPUs for computation and communication activities.

These simulations allow stakeholders to **test new architectures**, validate **hardware designs**, and fine-tune **transport protocols** like **RoCEv2** or **Ultra Ethernet (UET)** before physical deployment, dramatically reducing cost, risk, and time to insight. They can help stakeholders understand where the resources are being spent and where they are being wasted.

Why Simulate LLM Workloads?

ML Researchers	Infrastructure Teams	Executives and Planners
Gain deep visibility into LLM execution behavior, including GPU compute and communication dependencies	Identify network bottlenecks and congestion and predict large scale performance	Reduce capital risk in hardware investments
Optimize execution plans while minimizing consumption of GPU hours	Tune ECN thresholds, routing strategies, and buffer management	Predict ROI from new infrastructure

Why Scala?

Scala's platform integrates **Chakra trace files,** directed acyclic graphs (DAGs), that encode compute and communication events on each GPU across time. Scala supports both **synthetic** Chakra workloads and traces **captured** during real LLM training or inferencing.

- Scala Provides the ability to **upload your own Chakra files** or chose from a **library of expert Chakra files**
- Traces for each GPU, in the form of a Chakra Directed Acyclic Graph (DAG), preserve inter-GPU dependency flows via SEND/RECEIVE flows and intra-GPU dependencies of communication and computation tasks
- Represent collectives (AllReduce, Broadcast, etc.) with accurate timing
- **Synthetic Chakra trace generator**
    - Support synthetic trace generation for clean, tunable experiments
    - Synthetic traces can be generated for any size network
    - Synthetic traces are complemented with GPU specific profiling information that is used to simulate computation tasks
    - Synthetic traces mimic real training behavior without exposing IP
- Validate new topologies, algorithms, and protocol stacks
- Scala **Supports RoCEv2 and UEC Transports**
- **Scales to 64k+ nodes with full GPU trace fidelity**
- Access **validated network models** and **workloads** from **hardware vendors and Hyperscalers** along with an advanced set of **generic models** provided by Scala Computing or use the provided **SDK** to develop your own custom models.

Case Study

LLM Workload Simulation

We simulated an **LLM training** workload with the following characteristics:

- 256 GPUs (64 DP × 2 TP × 2 PP x 1SP)
    - GPU specific profiling information is used to simulate the computational performance of the NVIDA A10G GPU.
- RoCEv2 transport on a fully subscribed 2-tier Clos topology
- Each GPU trace mapped to a node using Chakra DAGs
- Network was fully subscribed and built using validated device models
- Simulation ran on Scala Computing's MPI-enhanced version of ns-3, deployed to elastic cloud resources

Study Insights

Simulation results revealed critical network behaviors:

- **Packet Volume**: Over 4.5 billion packets simulated representing 1.1 second of simulation time
- **PFC Events**: Pause frames indicated network congestion at fabric/rack switch layers—especially during **incast** events
- **NIC RX Utilization**: At 0.3047s, only **6 NICs** were active, revealing workload-induced under-utilization of network resources
- **Throughput Drops**: Bottlenecks aligned with unmet dependencies in the Chakra DAG, causing traffic stalls

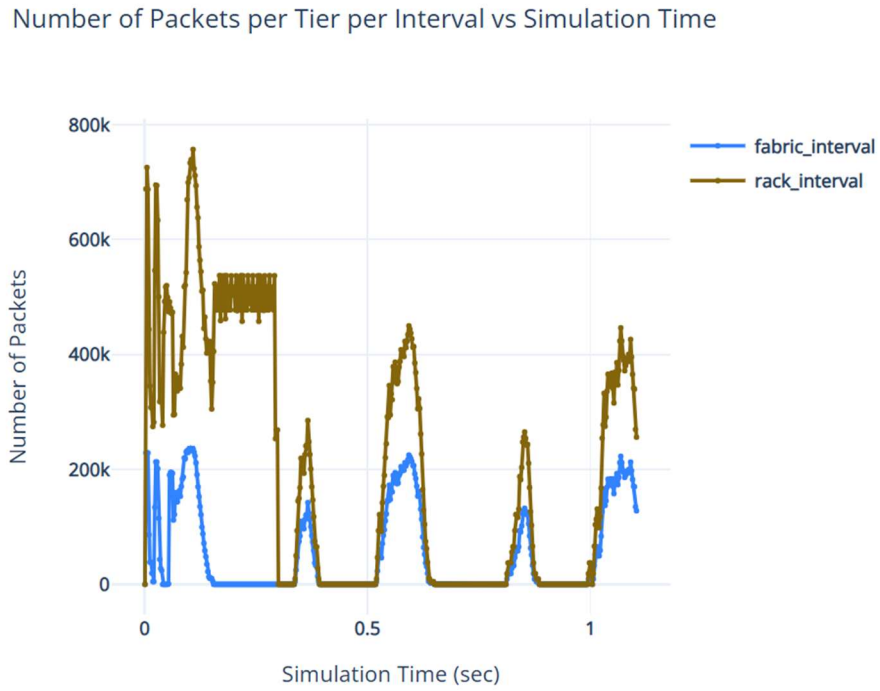
These insights help engineers co-optimize workloads and networks to improve performance **before physical training begins**.

1. Packets and PFCs

Network Interface Statistics

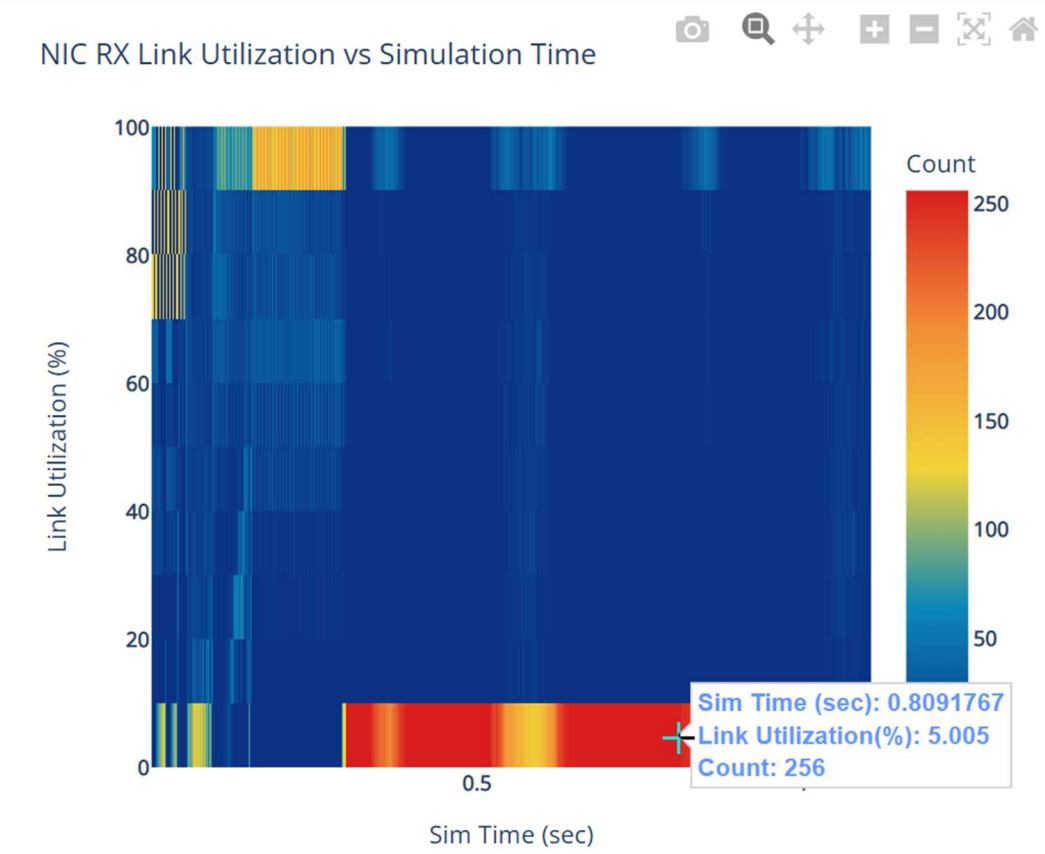
Tier	Packets RX	Packets TX	Bytes RX (GB)	Bytes TX (GB)	Ingress Drops (RX)	Egres
Global	4547995937	4547998603	9731.295	9679.678	0	0
Spine Switch	0	0	0	0	0	0
Spine Switch	0	0	0	0	0	0
Fabric Switch	0	0	0	0	0	0
Fabric Switch	593441014	593475126	1269.692	1259.004	0	0
Rack Switch	593474471	593441669	1269.689	1259.008	0	0
Rack Switch	1680346530	1680734599	3595.973	3565.684	0	0
NIC	1680733922	1680347209	3595.944	3595.974	0	0
NIC	0	0	0	0	0	0

This table shows the volume of packets and bytes transmitted and received across network tiers. Over 4.5 billion packets were processed globally representing 1.1 second of simulation time. No packet drops were observed, as expected with ROCEv2 with PFCs.



Looking at the number of packets per tier per interval, we see that there are periods when communication drops to zero. These are periods dominated by computation and suggest computation and communication are not fully concurrent.

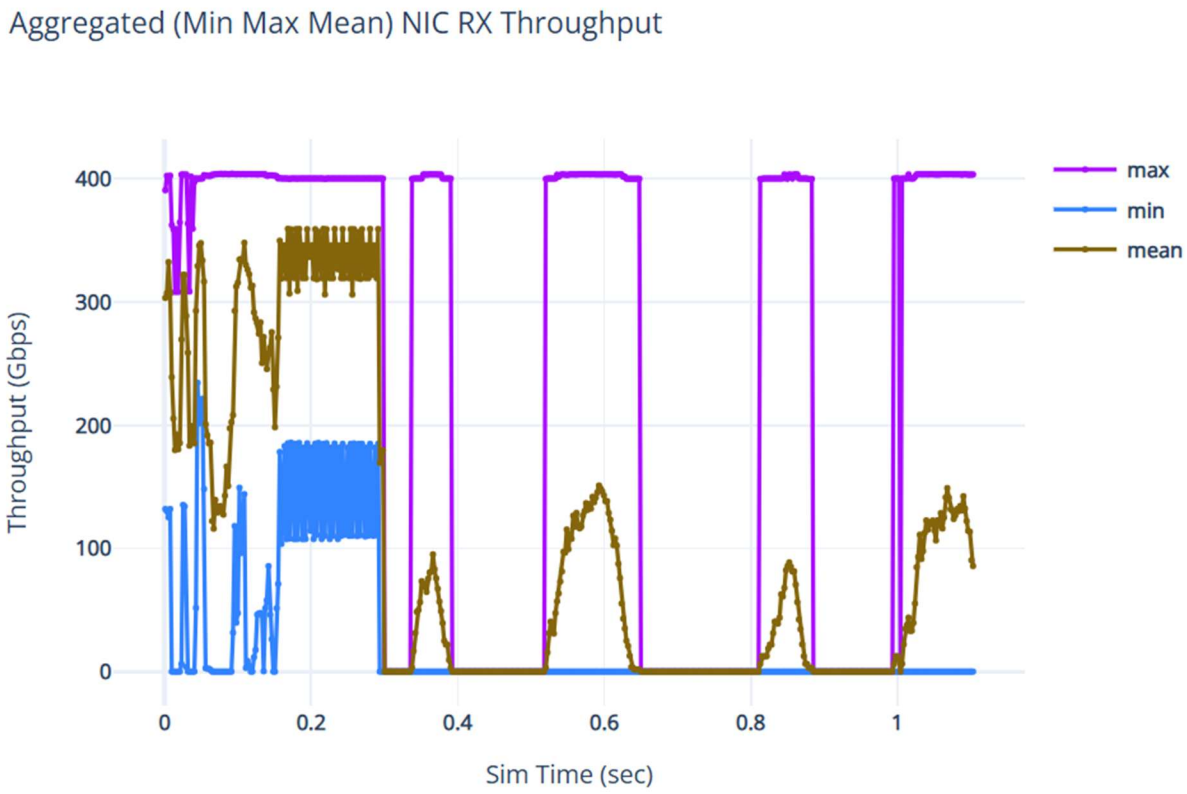
2. NIC RX Link Utilization



🔎 *An interesting thing to note is the complex behavior displayed by LLM workloads. With basic workloads, such as a single-collective all2all, the network may look like things are running well with high or even 100% utilization. By using a real LLM workload, the simulation is able to reveal realistic network bottlenecks and identify opportunities for optimization.*

This heatmap shows link activity over time. You can see the warmup in the first 0.1 seconds, then utilization is high for a brief period before it drops off significantly at around 0.3047 seconds. The network is mostly underutilized after the initial warmup. At times, the GPU utilization drops to zero because all the GPUs are involved with computation or are simply waiting on each other. This is workload-induced underutilization and suggests an opportunity for workload-network co-optimization.

3. NIC RX Throughput (Gbps)



The graph shows mean, min, and max throughput. Interestingly, around 0.3047 sec, throughput drops sharply, followed by periodic drops in communication. This is more evidence of a communication dependency stall, indicating a model-structure-induced bottleneck.

4. PFC Statistics

PFC Statistics

Tier	XOff RX	XOff TX	XOn RX	XOn TX	Total Time Paused (usec)	Direction (UL/DL)
Global	422040	422040	286455	286455	4920784.8	
Spine Switch	0	0	0	0	0	ul
Spine Switch	0	0	0	0	0	dl
Fabric Switch	0	0	0	0	0	ul
Fabric Switch	76013	104745	58862	65491	667751	dl
Rack Switch	104745	76013	65491	58862	808628	ul
Rack Switch	0	241282	0	162102	0	dl
NIC	241282	0	162102	0	3444404.7	ul
NIC	0	0	0	0	0	dl

PFC counters show congestion at rack and fabric switches, with significant pause frame activity.

The presence of PFCs in this RoCEv2 example indicates congestion in some network pathways, while the other charts indicate overall underutilization in most of the network. This apparent “paradox” is partially due to the structure of the LLM workload and partially due to characteristics of the ROCEv2 transport layer. Relevant considerations include:

- The proportion of GPUs that are sending traffic at a given time can be small; a larger fraction of GPUs are involved in performing compute tasks or waiting on receiving flows from other GPUs before they can progress their work.
- The extent of concurrency between compute and communications tasks may be limited. Full compute/communication concurrency is not always achieved.
- RoCEv2 lacks advanced features such as packet sprays allowing path diversification, load balancing, or strict control of queueing delays via congestion control. This can result in under-utilization of network resources and slower execution of highly interdependent distributed LLM workloads that are sensitive to queueing delays and influenced by other transport layer characteristics.

Comparison

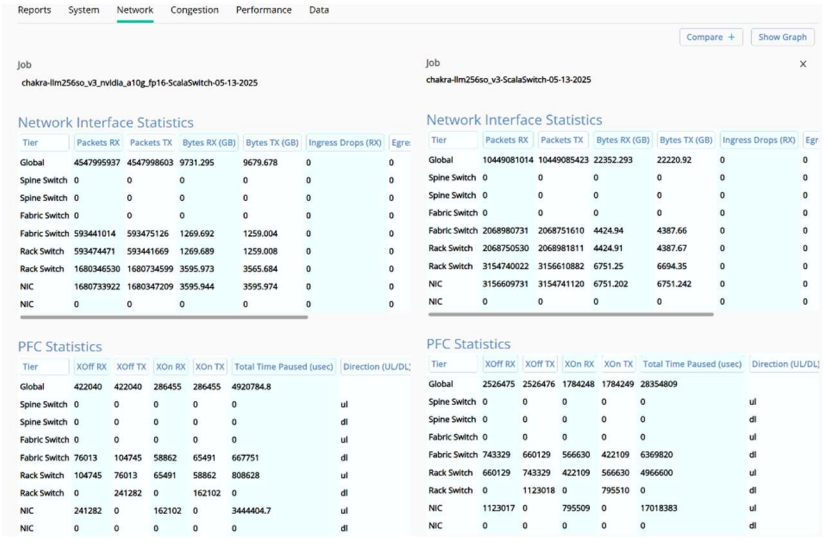
Previous charts have identified periods of low network utilization and throughput bottlenecks as many GPUs are involved with computation or are simply waiting on other GPUs to complete computation and communication tasks.

We could speculate that this is because we are using relatively low cost A10G GPUs and that we could improve that computation time with a faster and more expensive GPU. For instance an A100 costs about 5x the price of an A10G and would run at 2 or 3x speed due to the HBM2 & tensor cores. An H100 would be 8 to 12x more expensive, and would run 4 to 8x times faster, reducing the computation time significantly.

We could run a series of simulations that would test these faster GPUs to see if spending that extra money would significantly improve our network performance. Or we could reduce the simulated computation time to zero, simulating an infinitely fast computation, which would allow us to focus on optimizing the network efficiency independent of the GPU capabilities.

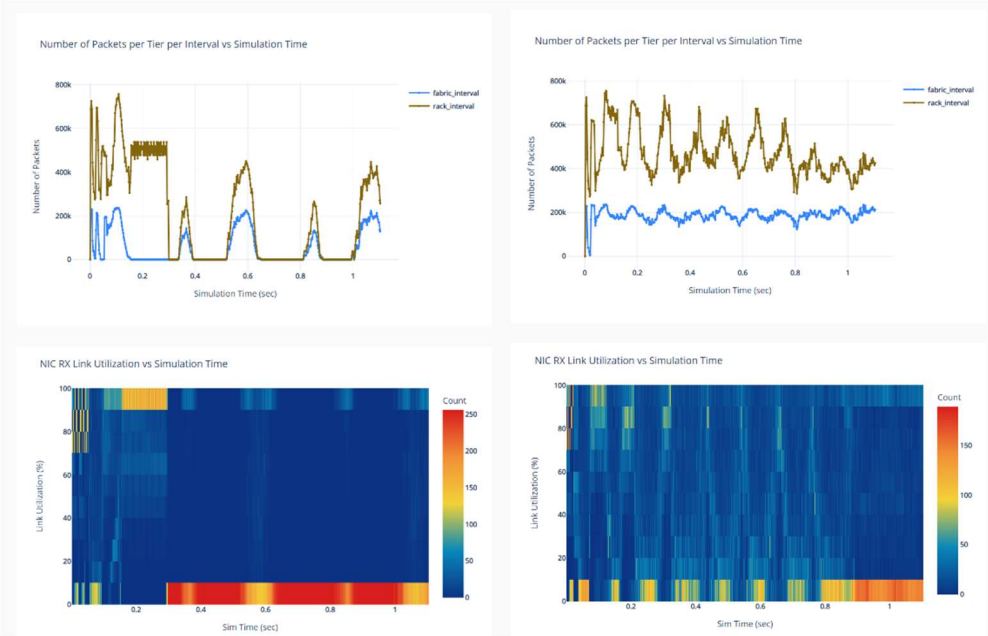
We re-ran the same simulation, but set computation time to zero for the Chakra DAG. Our real-world max potential lies between the performance shown with the A10Gs and the zero-computation time scenario. So we can also use this simulation to evaluate the max potential performance improvement we could get by spending a lot more money on GPUs.

Let's chart these next to each other and consider the differences. The first simulation, with the A10Gs, is on the left and the 2nd simulation, with the instant compute, is on the right.

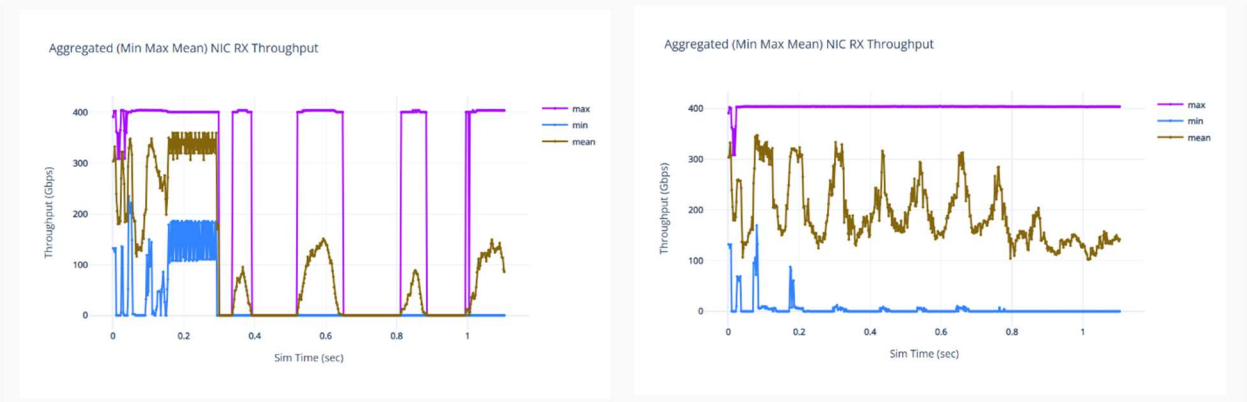


Looking at the Packets for the zero-compute time simulation, we see that more than twice as many packets were transferred during the simulation time, indicating a substantial improvement in network throughput performance because the GPUs were not waiting on computation. However, the number of PFCs went up by an order of magnitude, indicating congestion and potential to improve the LLM and the Network.

There are many other charts to compare, but let's look at Packets per tier per interval and the NIC RC Link Utilization for both scenarios...



Without Computation time, the dead periods do not appear at all. So this confirms the theory that those were due to computation time. Link utilization is much higher and more uniform, but still operates well below network capacity.



Throughput also indicates significant improvement, and some are operating at 400Gbps, but the mean throughput is still less than 50% of capacity.

This new simulation indicates the highest possible performance that can be gained by simply spending more on GPUs. Instead, AI developers and Network Engineers should be working together to develop AI algorithms that have a better fit with the Networks they are running on. These designs can be simulated and compared to understand the tradeoffs.

Conclusion

By enabling accurate simulation of complex distributed training patterns, Scala Computing offers an efficient, insightful, and secure path to help our partners gain deeper insights into network performance, and reduce risk, before deployment. The understanding that comes from simulation helps to develop more efficient, scalable solutions for modern networking challenges. The cost of training AI models can be in the tens of millions of dollars, so even small performance improvements in ideal ratios can translate to significant recurring savings. This approach is essential for ensuring that next-generation network architectures can support the growing demands of AI and high-performance computing applications

With Chakra-based LLM workload simulation, as implemented by Scala Computing, engineers, and executives are empowered to understand and optimize network performance in the age of AI/ML, at data center scale. The combination of an accurate model of a customer's real-world workload (through Chakra) and an accurate representation of their network, using vendor validated device models, enables network engineers to identify areas of sub-optimized coupling between the workload and the network. This opens the possibility of engineering the workloads specifically to couple better with the network or configuring the network to optimize the performance of the workloads.