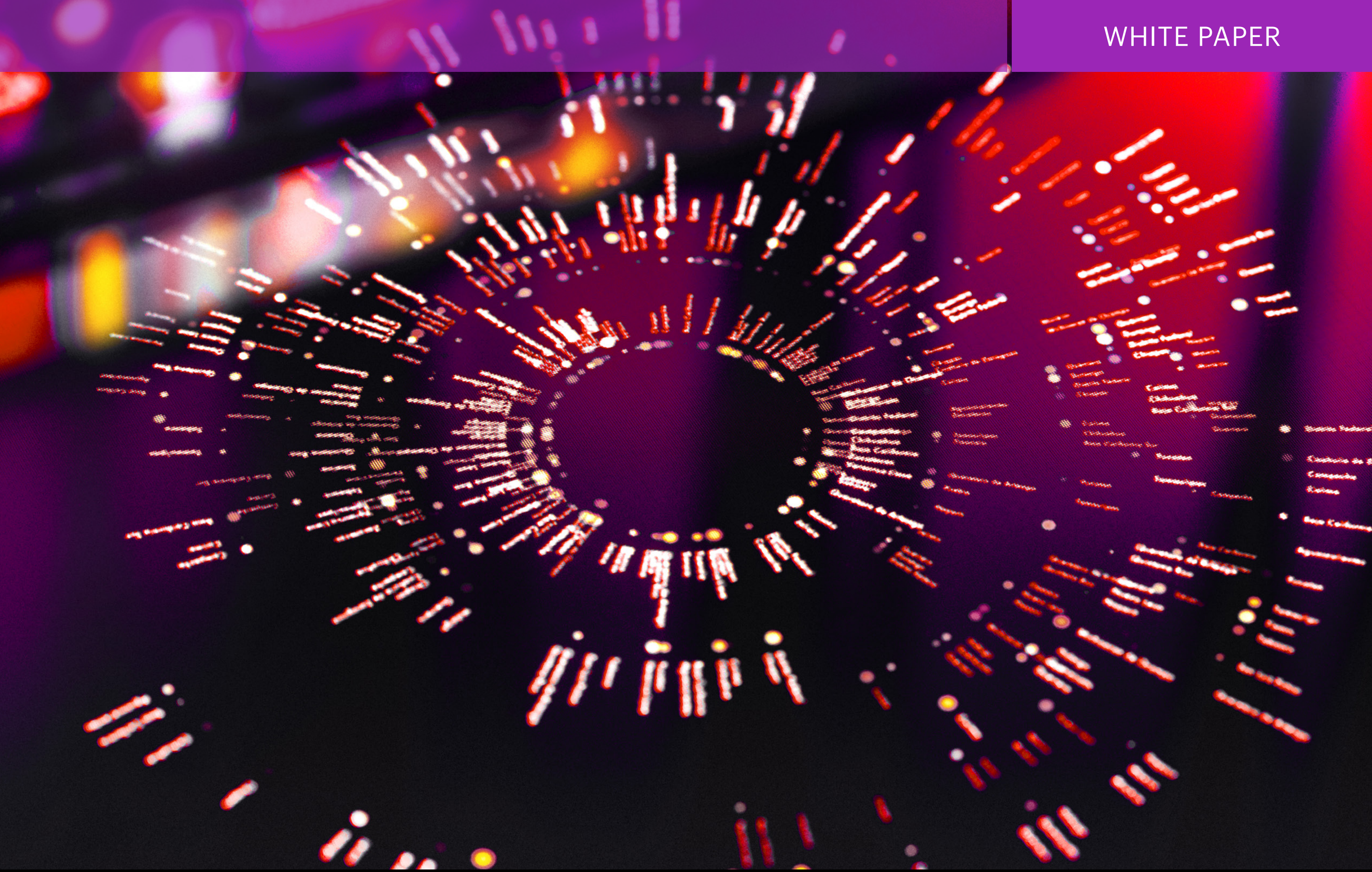
WHITE PAPER



How AI / ML Networks Differ from Traditional Networks

KEYSIGHT

The increasing adoption of artificial intelligence (AI) and machine learning (ML) demands more robust and efficient data center networks. This white paper looks at the new requirements for AI networks, the distinct AI traffic patterns, the technology available to make Ethernet networks suitable to run high-performance AI workloads, and how Keysight solutions can help optimize AI networks.

New Network Requirements

Networks that support AI and ML have different requirements and traffic patterns than traditional data center networks. The type of data, volume, and traffic patterns are radically different between traditional data centers and AI data centers. Larger AI clusters have hardware investments worth hundreds of millions of dollars, and optimizations can significantly reduce the time required to create learning models.

Traditional data centers

In traditional data center networks, individual queries or scheduled jobs — including overnight jobs — are common. These workloads vary widely, and the traffic is distributed across different connections. The overall network load evens out across individual links, growing proportionally with the number of users. Delayed or dropped packets do not typically cause significant problems. Examples of these traditional enterprise workflows are a web request for a banking back-end system to pull an individual’s account balance or an overnight job that calculates interest.

AI data centers

An AI cluster in a data center, on the other hand, must behave more like a supercomputer with thousands of graphical processing units (GPUs) and hundreds of CPUs and switches. In an AI cluster, the GPUs all work on the same problem, and building a large language model (LLM) can take days or weeks.

Interconnected with the fastest network links, these GPUs move massive amounts of data and cannot drop packets or encounter congestion on any link. Because the GPUs are all working on the same problem, they complete a task when the last GPU finishes processing. Once built, the LLM can move to a smaller GPU or CPU-based front-end computer system. Then, users can query the model to see how well it applies the information learned during training. This process is known as inferencing. For the sake of this paper, we are talking only about the back-end LLM training.

Built to scale

When scaling a traditional data center, optimization is primarily determined by comparing the service-level agreements (SLA) for a query response with the actual result. The result could be milliseconds for retrieving the balance of a checking account or hours for large overnight jobs. If the results do not meet the expected timing, then operators can adjust the number of servers and network speeds and feeds.

Scaling an AI cluster, however, requires optimizing the time it takes to build the learning model. Building a new model can take weeks or months. Reducing this time by even a few days can free up millions of dollars' worth of GPUs in an AI data center to work on the next job. Adding GPUs is expensive, and they have limited availability. So the logical first optimization is improving the GPU idle time and removing any potential network congestion before adding capacity.

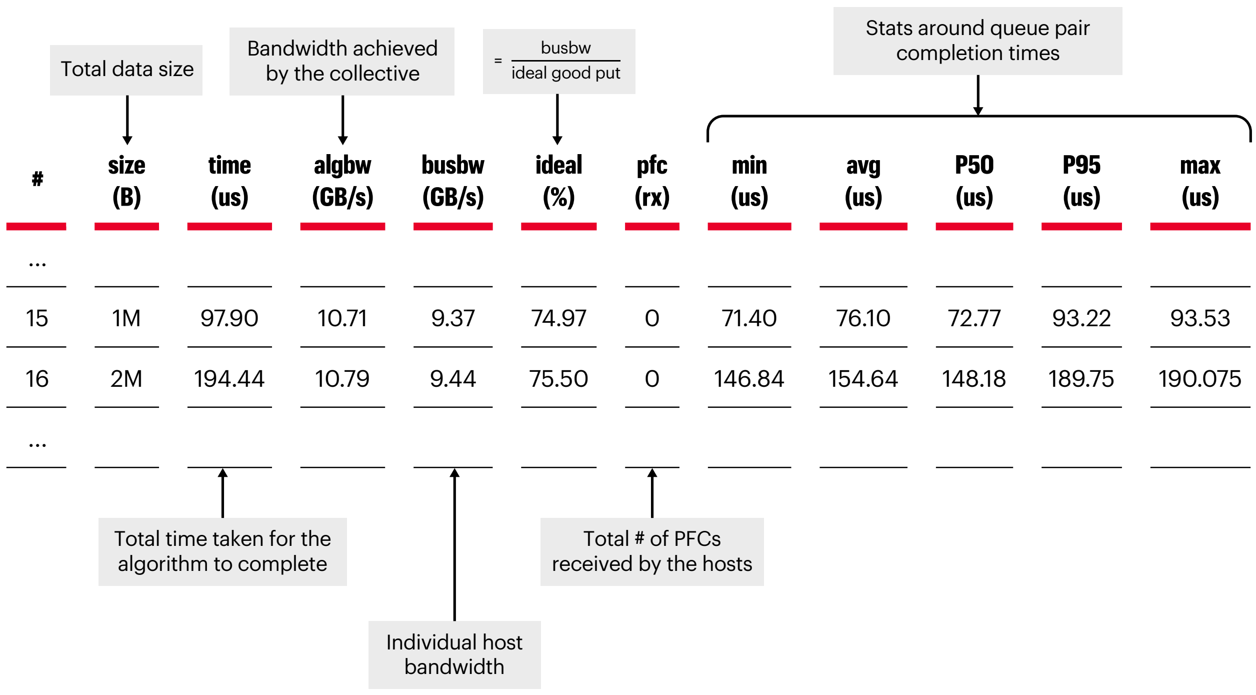
In an AI cluster, GPUs work together to train the model through learning. Any prolonged packet latency or packet loss affecting even one GPU can significantly increase the job completion time as the other GPUs sit idle. High-speed network links, while required, are not enough. The key goal is configuring the AI network to avoid congestion using various techniques that are part of modern Ethernet networks.

New Traffic Patterns

The nature of network traffic patterns in AI data centers differs from traditional data center traffic. The workloads are distributed among hundreds or thousands of GPUs, with massive data sets being sent and received. AI data set sizes exhibit limited randomness, unlike variably sized internet traffic. The AI cluster experiences rapid, high-frequency shifts between GPU computations and the sharing of computation results among GPUs. When a GPU sends or waits for information, it is idle. Traffic can also be bursty and exhibit specific patterns, such as all-to-all, with many GPUs trying to send data to one another, causing in-cast congestion.

Long tail

AI network performance is a measurement of the flows with the longest completion time, not average bandwidth. These long tails significantly affect the job completion times and, thus, GPU utilization. If the average flow completion time is 150 ms but the longest completion time on one GPU is 190 ms, then the actual overall collective completion time for all GPUs (the time required for the algorithm to complete the workload) is 190 ms. See Figure 1 for details.


Total data size
Bandwidth achieved by the collective
= busbw / ideal good put
Stats around queue pair completion times
Total time taken for the algorithm to complete
Individual host bandwidth
Total # of PFCs received by the hosts

#	size (B)	time (us)	algbw (GB/s)	busbw (GB/s)	ideal (%)	pfc (rx)	min (us)	avg (us)	P50 (us)	P95 (us)	max (us)
...											
15	1M	97.90	10.71	9.37	74.97	0	71.40	76.10	72.77	93.22	93.53
16	2M	194.44	10.79	9.44	75.50	0	146.84	154.64	148.18	189.75	190.075
...											

Figure 1. Key measurements metrics example

Balance is important in network optimization

In this example, some GPUs get their data much faster than other GPUs. The optimization goal is not achieving the fastest possible data movement to a particular GPU but rather balancing the network to ensure that all GPUs receive the data at around the same time so they don't sit idle. In effect, this process involves speeding up the slow flows and slowing down the fast flows. Once the GPUs receive data from one other, they can kick off another compute cycle. This optimized network maximizes GPU utilization.

The analogy here is 100 marbles suspended over a net with holes that are just slightly larger than the marbles. If you drop all the marbles into the net, some will fall through very quickly, but many will bunch up, and it will take some time for the last one to fall through. If you were to direct the marbles via some sort of lanes to the holes, even if it takes longer for the first marble to get through, all the marbles would get through more quickly. The holes here are the network links, and the marbles are the flows from the GPUs.

In comparison, traditional data center traffic consists of many randomly sized flows occurring at different times and connecting to many clients. Balancing this type of traffic network link is relatively straightforward, and in some cases, it balances itself. AI traffic, on the other hand, involves massive flows all the time to all nodes and is much more challenging to balance.

When to upgrade an AI network? The paradigm has changed for AI

From an operational perspective, in a traditional data center, if link utilization approaches 50%, discussions start about upgrades. In an AI data center, link utilization can reach 90%. If the speed of all the links magically doubled, the link utilization would still be very high.

New Ethernet Network Configurations

Ethernet networks are prevalent and well-established in today's data centers, and companies can optimize and configure them to support AI networks. The skills needed to build, deploy, manage, and troubleshoot these networks are often available using internal company resources or contractors and consultants.

Companies can use these existing skill sets to configure Ethernet networks for AI to avoid congestion that could affect GPU utilization.

Modern Ethernet protocols manage flow and congestion in data center networks using capabilities such as priority flow control (PFC), explicit congestion notification (ECN), data center quantized congestion notification (DCQCN), and packet spraying. Let's take a quick look at each of these technologies.

Start tuning with PFC and ECN

PFC enables a switch to send a pause frame to the upstream device when its buffer reaches a certain threshold, stopping traffic for that queue. While this approach prevents packet dropping, it is not a great solution on its own. The network would run slowly, with queues starting and stopping.

ECN provides congestion notification between devices so that the sending device reduces the rate of traffic.

DCQCN coordinates the work of ECN and PFC. DCQCN is an algorithm that enables ECN to manage flow control by decreasing the transmission rate when congestion starts, thereby minimizing the duration of PFC. Tuning DCQCN is tricky, and other paths for improving AI network configurations exist.

Further options for optimization

Equal cost multipath (ECMP), a routing strategy traditional data centers use, balances the network with flows. But that is challenging when a single AI flow can saturate a link. Balancing the network at the packet level is more effective for an AI network. Packet spraying and other forms of load balancing, such as dynamic load balancing, cell-based routing, and cognitive routing, send packets across the available network links. Packets are small compared to flows in the AI collective, dramatically improving link utilization.

At the hardware level, remote direct memory access (RDMA) allows applications across two servers to exchange data directly without the use of the processors, operating system, cache, or networking kernel. That is, an application can read / write data on a remote server’s memory without using the processor of either server, so data moves faster with lower latency. RDMA over Converged Ethernet (RoCE) provides this mechanism on Ethernet networks.

The case for the lossless Ethernet network

Creating a lossless Ethernet network is possible using a combination of these technologies and the right settings for each.

The protocols for lossless Ethernet networks exist, as do the tools to benchmark results, the required management applications, and the institutional knowledge of network engineers and architects.

Industry experts are developing new Ethernet capabilities and innovations for AI. The Ultra Ethernet Consortium is working to standardize high-performance Ethernet capabilities and simplify configuration and management as part of its road map for AI networking growth.

The challenge is how to validate the design and objectives before deployment.

New Ways to Optimize an AI Network

Benchmarking an AI network requires creating traffic patterns seen during AI training and sending that data through a network traffic generator that can emulate a GPU and RDMA network interface card (NIC). GPUs support RDMA NICs, which enable fast data access between GPUs.

Types of traffic to emulate

The system should be able to repeatably create scenarios with different data patterns and sizes that result from collective communications in an AI cluster. The traffic includes emulating queue-pair (Q-pair) connections and flows, generating congestion notifications, performing DCQCN-based dynamic rate control, and providing flexibility to test throughput, buffer management, and ECMP hashing.

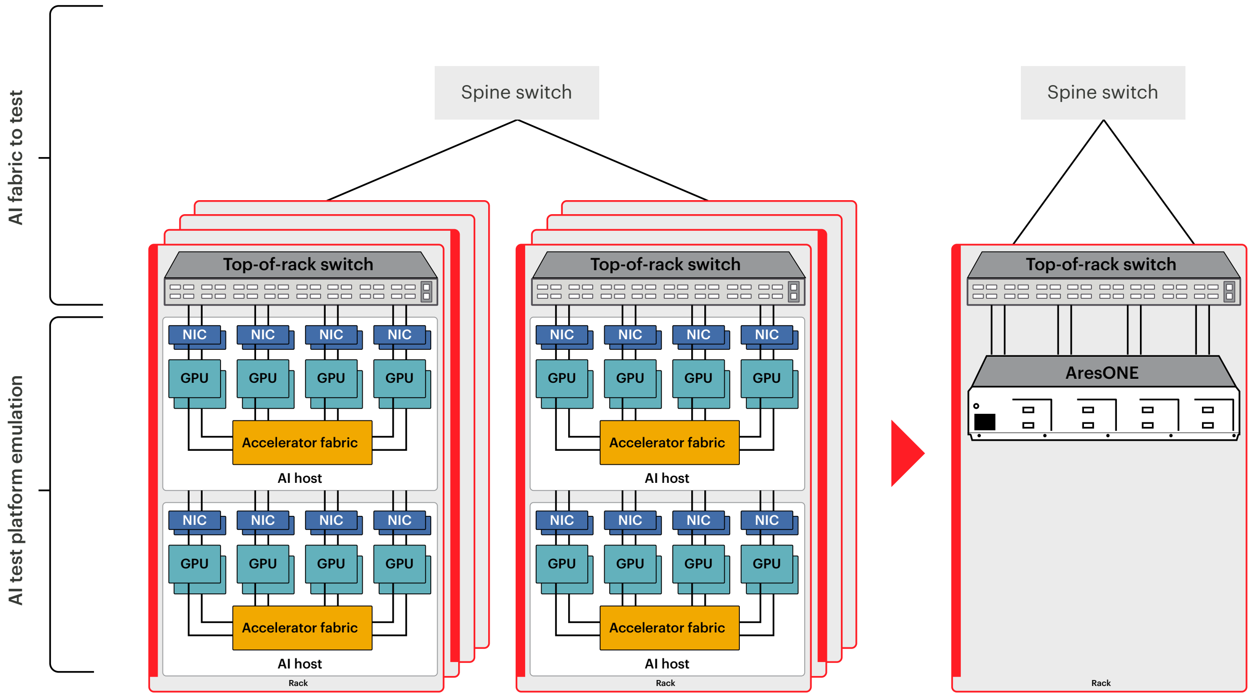
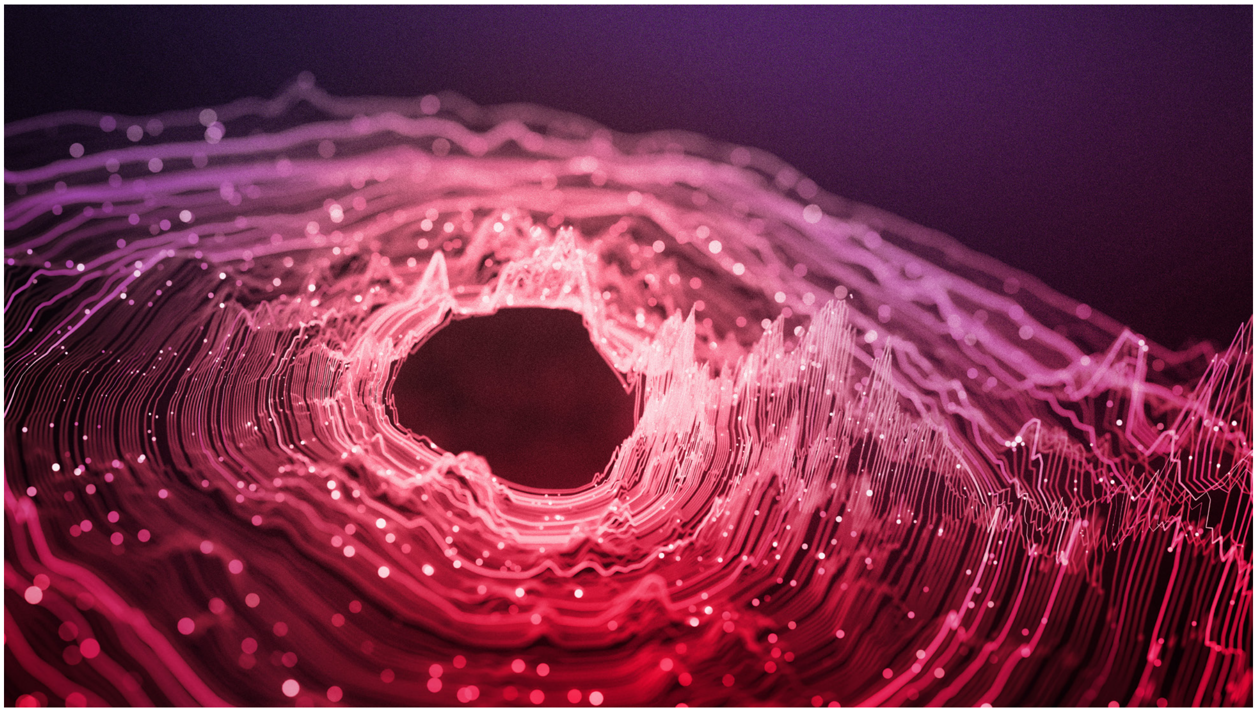


Figure 2. AI cluster compared to Keysight AI Data Center Builder

Engineering teams can use network traffic generators that support RoCE v2 / RDMA to make design improvements in a lab or staging environment based on performance measurements of the fabric, without relying on GPU accelerators.

An effective solution for optimizing AI networks should provide the flexibility to define AI system configurations for workload emulation. This includes the number of GPUs, NICs, congestion control settings (such as PFC and DCQCN), data sizes, Q-pair characteristics, and the configuration of the emulated NIC. This flexibility enables the benchmarking of different configurations in an efficient and repeatable manner.

It is important to conduct runs of different data sizes, providing results for key performance indicators such as completion time, algorithm, and bus bandwidth. Insights into statistical metrics distribution among individual RoCEv2 Q-pairs are also crucial.



Conclusion

AI data center network requirements and traffic patterns are significantly different from traditional data center networks. Paradigms used to optimize an AI network are different, and there is an expectation that the network will run near capacity and in a lossless manner. One key strategy is to optimize the network for GPU utilization. While there are numerous ways to accomplish this with Ethernet networks, it is not necessarily obvious or trivial to do.

To avoid manual, time-consuming work, Keysight's tools to benchmark and optimize AI networks leverage existing data center engineering skills and institutional knowledge and processes. With that, network architects can use the Keysight AI Data Center Builder to emulate network loads and GPU behavior to proactively pinpoint bottlenecks and optimize network performance. In combination with load test modules, this solution optimizes the AI network, which can result in an improvement of GPU utilization — minimizing wasted resources and slashing network GPU expenses.