Cisco Data Center Networking Solutions: Addressing the Challenges of AI/ML Infrastructure

White Paper

Contents

This document explains how Cisco data center networking solutions address the key challenges in interconnecting the AI/ML compute accelerators, commonly known as GPUs. These solutions are based on the Cisco Nexus 9000 Series Switches for data-plane connectivity and the Cisco Nexus Dashboard platform for management, automation, and visibility features.

AI/ML Infrastructure Networks – An Overview

Artificial Intelligence and Machine Learning (AI/ML) applications require substantial computing resources to identify patterns in vast data sets. The patterns in the existing data are calculated as parameters of a model, which is called training the model. The model can then be used to make predictions on new data, which is called inferencing from the model. These massive computations take extremely long to complete on Central Processing Units (CPUs), but they can be significantly accelerated using the parallel processing capabilities of Graphics Processing Units (GPUs).

One challenge in training a model is that the data sets are so large that even a GPU may take months to complete a job. Another hurdle is that the model size is so large that it can't fit within the memory of a single GPU. These challenges can be overcome by distributing a job among many GPUs using parallelization strategies. For example, the data parallelization strategy divides a large data set into smaller batches to be processed on each GPU. Model parallelism is another strategy that divides the model into smaller batches for each GPU.

Regardless of the parallelization strategy, one common phenomenon is the frequent sync of the states of the GPUs running a distributed job. This sync is crucial for ensuring all GPUs calculate a unified model. Essentially, the GPUs in a cluster calculate the result of each step. But, before going to the next step, they must learn the result of a similar step from all other GPUs running the same distributed job. Syncing the data among all the GPUs is therefore called collective communication. Many algorithms, such as AllReduce, ReduceScatter, and AllGather, have been invented to make collective communication more efficient by reducing the delay and overhead.

While collective communication operates at a higher layer, the actual data exchange among the GPUs happens by direct point-to-point memory access of one GPU by another. In simpler terms, a GPU directly accesses the memory of another GPU to read or write to it. This is called Direct Memory Access (DMA) and it happens when the GPUs are within the same machine or physical enclosure. The same concept is extended to achieve Remote Direct Memory Access (RDMA) when the GPUs are in separate machines.

GPUs connected by a network initiate RDMA operations using InfiniBand (IB) verb APIs. Next, data to be sent to another GPU is divided into multiple payloads. Then, each payload is encapsulated within User Datagram Protocol (UDP), Internet Protocol (IP), and Ethernet headers, and is transferred using the standard forwarding mechanism of the network. This exchange of RDMA operations through an Ethernet network using UDP/IP encapsulation is called RDMA over Converged Ethernet version 2 (RoCEv2). Note that RoCEv2 does not have a dedicated header. It is just the name of an RDMA-capable protocol using UDP/IP packets through an Ethernet network.

To bring it together:

- All GPUs running a distributed training job using one of the parallelization strategies are known to be in a cluster.
- All GPUs within a cluster frequently sync their states using collective communication.

- Collective communication exchanges data using RDMA operations initiated by IB verb APIs.
- These RDMA operations result in standard UDP/IP packets on an Ethernet network interconnecting the GPUs.

The complexity of the machine learning algorithm, collective communication, RDMA operations, and even the IB verb APIs remain within the end devices hosting the GPUs. The network remains unaware of these details and transfers the UDP/IP packets to the destination using its standard forwarding mechanism. Although this simplifies the networks for AI/ML infrastructure, the frequent sync of the GPU states while calculating patterns in huge data sets causes unprecedented traffic patterns. The GPUs not only send and receive traffic at the full capacity or line rate, but the collective communication makes all GPUs in a cluster initiate RDMA operations simultaneously, leading to burst patterns significantly different from those seen in a typical data center network.

The following sections explain these challenges and how Cisco Data Center Networking solutions based on Nexus 9000 Series switches and the Nexus Dashboard platform address them. The examples in this document are based on the Cisco Nexus 9332D-GX2B switch providing 32 400 Gigabit Ethernet (GbE) ports in the 1 rack unit (RU) form factor, and the Cisco Nexus 9364D-GX2A switch providing 64 400 GbE ports in the 2 RU form factor. The high-speed ports in a compact form factor and the rich features of the NX-OS operating systems make these models the apt choice for AI/ML infrastructure networks.

Multiple Networks of an AI/ML Infrastructure

The most important component of the AI/ML infrastructure is the GPU node, which can be a host or server with typically eight GPUs. These nodes have many ports. Some ports are dedicated for special purposes, such as inter-GPU connectivity. Other ports are dedicated to connecting to a typical data center network or accessing a remote storage device.

Based on the port types, these GPU nodes are connected through multiple networks, each serving a special function. Figure 1 illustrates how the nodes are connected.

These are the functions of the networks:

- **Inter-GPU backend network**: An Inter-GPU backend network connects the dedicated GPU ports for running distributed training. This network is also known as the back-end network, compute fabric, or scale-out network.
- **Front-end network**: A front-end network connects the GPU nodes to the data center network for inferencing, logging, managing in-band devices, and so on.
- **Storage network**: A storage network connects the GPU nodes to the shared storage devices providing parallel file system access to all the nodes for loading (reading) the data sets for training, and checkpointing (writing) the model parameters as they are learned. Some users may share the front-end network to connect storage devices, eliminating a dedicated storage network.
- **Management network**: A management network provides out-of-band connectivity to the devices of the AI/ML infrastructure, such as GPU nodes, network switches, and storage devices.

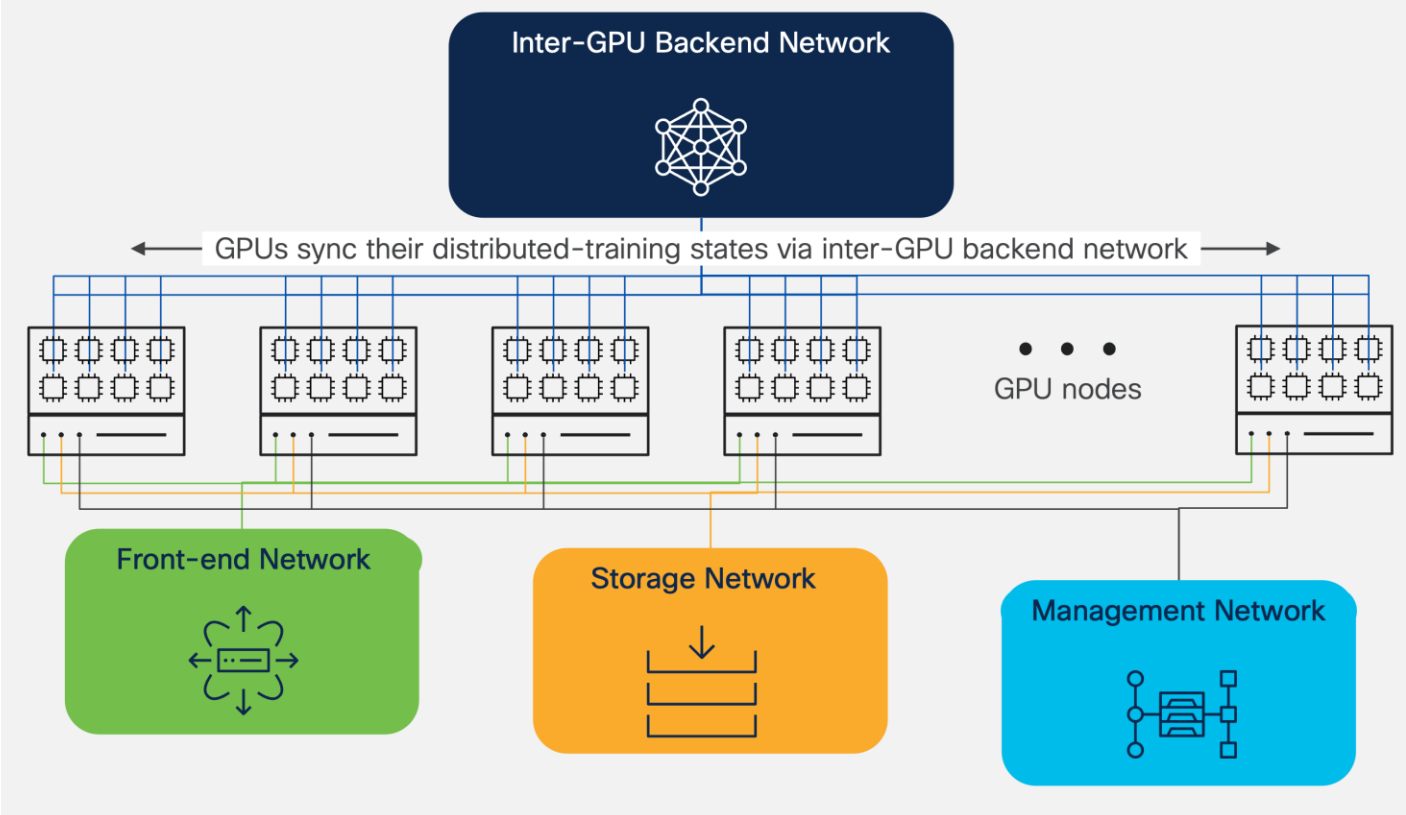

Figure 1.
Multiple networks interconnecting the GPU nodes

The primary focus of this document is on the inter-GPU backend networks. For details about the other network types, refer to the references section.

Designing an Inter-GPU Backend Network

Smaller GPU clusters can use a single-switch network. For example, up to 64 GPUs can be interconnected using the 2 RU, 64-port 400 GbE, Cisco Nexus 9364D-GX2A switch (see Figure 2).

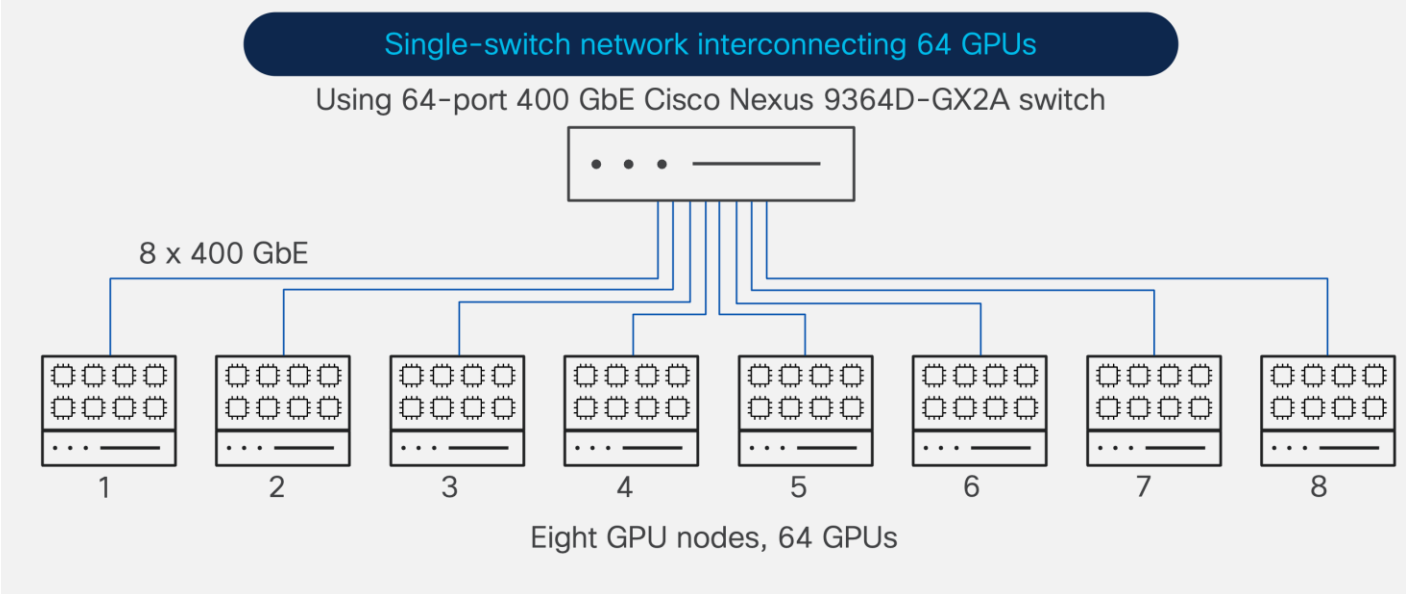

Figure 2.
Single-switch backend network for interconnecting up to 64 GPUs

For a larger GPU cluster, a spine-leaf network design is the best option because of its consistent and predictable performance and ability to scale. The edge switch ports that connect to the GPU ports should operate at the fastest supported speeds, such as 400 GbE. The core switch ports between the leaf switches and spine switches should match or exceed the speed at which the GPUs connect to the network.

Non-Blocking Network Design

The inter-GPU backend networks should be non-blocking with no oversubscription. For example, on a 64-port leaf switch, if 32 ports connect to the GPUs, the other 32 ports should connect to the spine switches. This non-oversubscribed design provides enough capacity when all the GPUs in a cluster send and receive traffic at full capacity simultaneously.

Rails-Optimized or 3-Ply Network Design

A rails-optimized or 3-ply network design improves inter-GPU collective communication performance by allowing single-hop forwarding through the leaf switches without the traffic going to the spine switches. These network designs and their variants are based on the traffic patterns among the GPUs in different nodes. The rails-optimized design is recommended for NVIDIA GPUs, whereas a 3-ply design is recommended for Intel Gaudi accelerators.

Figure 3 shows a rails-optimized network design using 64-port 400 GbE Cisco Nexus 9364D-GX2A switches. This design has the following attributes:

- It connects 256 GPUs in 32 nodes, each with eight GPUs.
- It is rails-optimized by connecting port 1 on all the GPU nodes to the first leaf switch, port 2 on all the GPU nodes to the second leaf switch, and so on.
- It has eight leaf switches and four spine switches connected using a non-blocking design.

To create a larger cluster, the design of Figure 4 can be expanded with multiple such networks interconnected between leaf and spine switches in a non-blocking way.

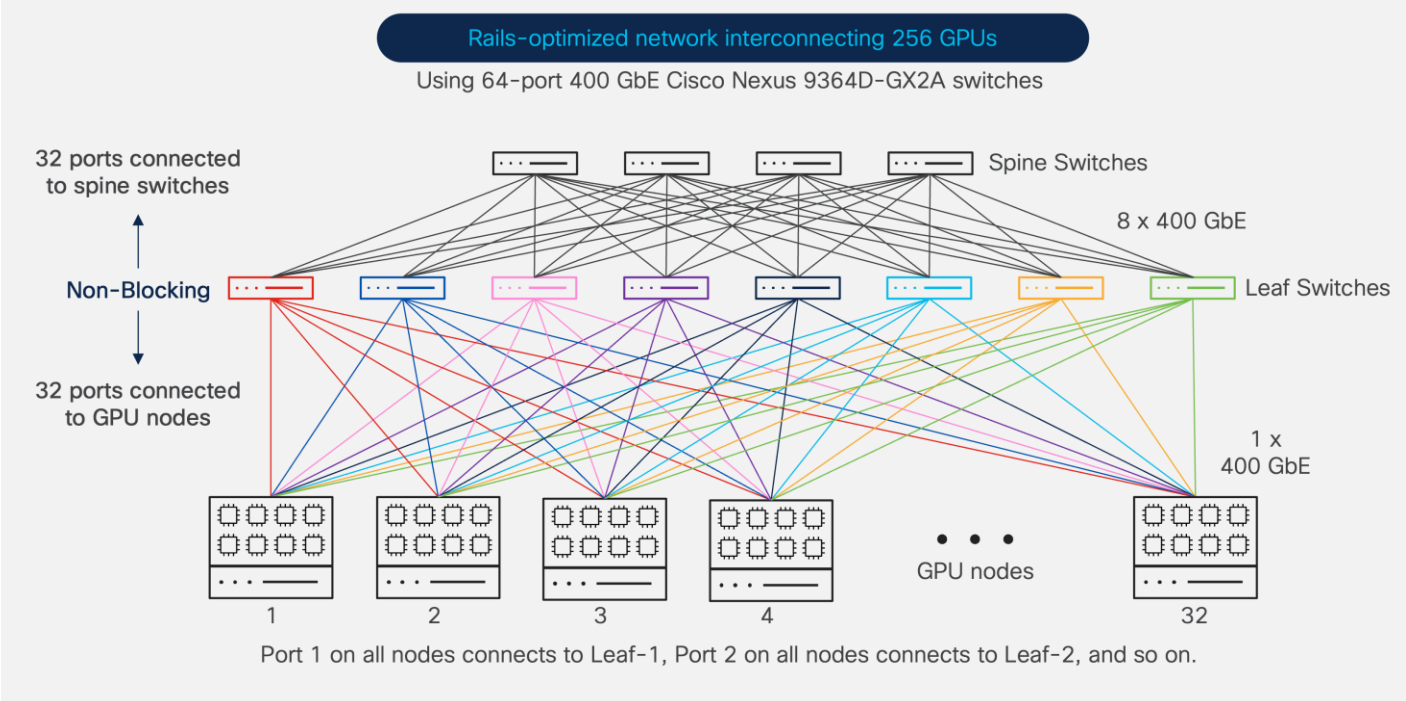

Figure 3.
A non-blocking rails-optimized network design based on 64-port 400 GbE Cisco Nexus 9364D-GX2A switches for interconnecting 256 GPUs

Figure 4 shows a 3-ply network design using 32-port 400 GbE Cisco Nexus 9332D-GX2B switches. This design has the following attributes:

- It connects 128 Intel Gaudi2 accelerators in 16 nodes. Each node provides six ports for the inter-Gaudi2 network.
- On each node, the first and fourth ports are connected to the first ply switch, the second and fifth ports are connected to the second ply switch, and the third and sixth ports are connected to the third ply switch. This connection scheme creates a 3-ply network design.
- It has three switches.

To create a larger cluster, the design of Figure 4 can be expanded by using the 64-port 400 GbE Cisco Nexus 9364D-GX2A switches and connecting the network plies using spine switches in a non-blocking way.

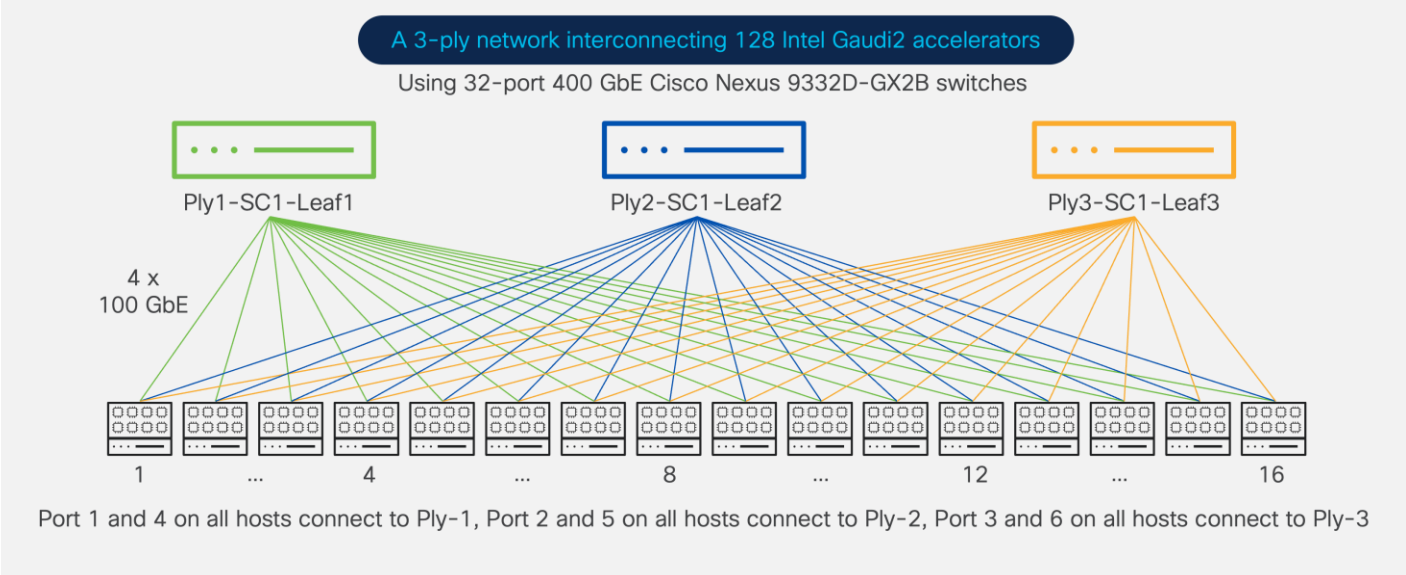

Figure 4.
A 3-ply network design using 32-port 400 GbE Cisco Nexus 9332D-GX2B switches for interconnecting 128 Intel Gaudi2 accelerators

The Importance of an Optimal Network Design

An optimal network design is the foundation of a high-performance inter-GPU backend network. As mentioned, a non-blocking network provides ample capacity when all GPUs in a cluster send and receive traffic at their full capacity simultaneously. Likewise, the rails-optimized or 3-ply network design reduces the network delay. Moreover, it also eliminates or lowers the amount of traffic between the leaf and spine switches. Network delay and high traffic volume are the two key challenges in inter-GPU networks, as explained in detail in the following sections. Both these challenges can be avoided, or their adverse effects can be lowered with an optimal network design.

The Challenges with the Inter-GPU Backend Networks

In addition to robust data plane connectivity, the inter-GPU backend network must also be capable of faster and simpler configuration, and proactive detection of anomalies while the GPUs send and receive traffic during a distributed training job. As mentioned earlier, all GPUs in a cluster frequently sync their states and must wait for this sync to complete before moving to the next step. This means that even if one GPU is degraded or disconnected from the network, the entire training job is halted, thereby increasing the model training time and delaying the return on investments. The following sections explain these challenges and how Cisco Data Center Networking solutions based on Nexus 9000 Series switches and the Nexus Dashboard platform address them.

Challenge 1: Packet Loss in an Inter-GPU Backend Network Degrades the Training Performance

Why this is a challenge

The effect of lost packets is much more severe in an inter-GPU backend network than in a typical data center network. This is because most traffic in a typical data center network uses Transmission Control Protocol (TCP), which has a built-in capability to retransmit selectively only the lost packets. However, an inter-GPU backend network does not use TCP. As explained earlier, GPUs transfer data using RDMA operations encapsulated within UDP, IP, and Ethernet headers (RoCEv2). Unlike TCP, UDP cannot detect

and retransmit a lost packet. As a result, the upper layer, which is the RDMA layer in this case, on the GPU waits for a timeout value leading to retransmitting the entire RDMA operation or retransmitting all packets after the lost packet in the sequence. This retransmission not only degrades the distributed training performance, but also creates additional load on the network by transmitting the same packets again, even when only one out of hundreds of packets of an operation is lost.

What causes packet loss in a network

The two common causes of packet loss in a network are congestion and bit errors:

- **Network congestion**: A network becomes congested when the ingress traffic volume exceeds the egress bandwidth capacity. Excess traffic may be temporarily stored in the buffers of the network devices. But, buffers are a finite resource and they eventually fill up if the ingress traffic rate does not become equal to or lower than the egress capacity. When the buffers are full, the only option for a network port is to drop new incoming traffic.
- **Bit errors**: When a bit stream is exchanged over a network, some bits may be altered, resulting in bit errors. Bits may be altered because of faulty cables or transceivers, loose connections, accumulated dust on cable end points, transceivers, or patch panels, and environmental issues such as temperature and humidity. When bit errors are within a frame and these errors exceed the number of bits that can be recovered by Forward Error Correction (FEC), the frame fails the Cyclic Redundancy Check (CRC) and is dropped by the receiver.

How Cisco helps avoid packet loss due to network congestion

Cisco Nexus switches avoid packet loss due to network congestion by pacing the ingress traffic rate with the egress traffic rate. The traffic pacing is achieved by Priority-based Flow Control (PFC), which uses special-purpose pause frames to inform the directly attached sender to pause or un-pause the traffic. This avoids buffer overrun on the receiver port, and as a result, packet drops are avoided.

Networks that use flow control between directly-connected devices are called lossless networks.

A key point to understand is that PFC does not guarantee that traffic will not be dropped. Traffic is still dropped in some conditions, such as when packets are held up in the buffers for a long time during severe congestion and when frames are corrupted due to bit errors. In other words, lossless networks do not provide a guarantee of lossless packet delivery, although they end up achieving it quite well.

Cisco Nexus Dashboard simplifies configuring the lossless networks by enabling PFC on all network ports using a single check box. It has built-in templates with optimized buffer thresholds for sending the pause frames. Further optimizations with refined thresholds can be uniformly applied across the network by changing these templates.

How Cisco helps in handling bit errors

Cisco Nexus switches use standard-based Forward Error Correction (FEC) mechanisms to recover automatically a limited number of bit errors, thereby reducing CRC-errored frames and retransmission of all or many packets of an RDMA operation.

But, the ultimate solution to bit errors is to solve the root cause, such as faulty devices, loose connections, dust, and environmental issues mentioned earlier. For a faster resolution, Cisco Nexus switches help pinpoint the source of bit errors using metrics such as CRC and stomped CRC. FEC-corrected and uncorrected counters can even predict the likeliness of CRC errors on a port, thereby helping proactively resolve the root cause of bit errors before they degrade the collective communication in an inter-GPU network.

Cisco Nexus Dashboard pinpoints the exact location of CRC errors (see Figure 5) and tries to find the root cause based on automatic correlation with the transceiver health.

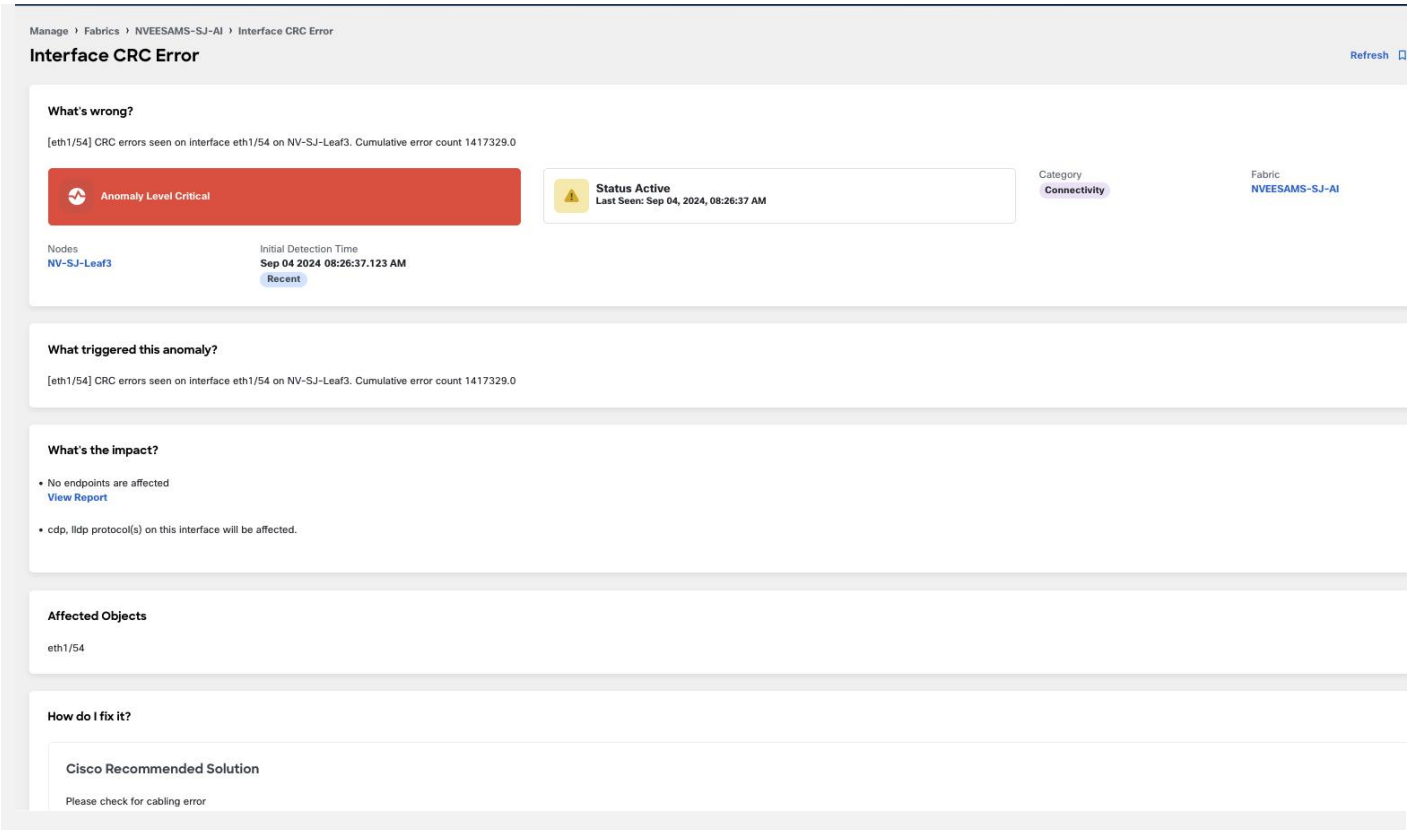

Figure 5.
Nexus Dashboard pinpoints the time and the location of CRC errors

Challenge 2: Network Delay for Inter-GPU Communication

Why this is a challenge

Time spent by the packets in the network contributes to the completion time of an RDMA operation and the collective communication. Not just the network delay, but the variance in the delay is another challenge for inter-GPU networks. This is because even if only one RDMA operation takes longer to complete, the entire collective communication among all the GPUs in a cluster must wait, thereby degrading the training performance. This increased delay caused by only a small percentage of operations is known as tail latency or 99th percentile latency.

What causes network delay

The following factors cause network delay:

- **Forwarding delay**: Forwarding delay is the time taken to forward a packet from an ingress port to an egress port on a switch.
- **Propagation delay**: Propagation delay is the time a signal takes to travel the length of a link depending on the speed of light in the media. Typically, a 1 meter cable adds 5 nanoseconds of propagation delay.

- **Serialization delay**: Serialization delay is the time a network port takes to transmit a packet on the wire. It depends on the link speed and packet size. For example, a 400 GbE port takes 80 nanoseconds to transmit a 4 kilobyte (KB) frame.
- **Queuing delay**: Queuing delay is the time a packet spends in the queue of a network port, waiting for its turn while the other packets ahead of it are transmitted.

The following things are the key attributes of these delay factors:

- After a network is operational, the forwarding delay, propagation delay, and serialization delay remain fixed and cannot be changed by a user. Only the queuing delay varies and therefore needs special attention to minimize it.
- Forwarding delay, propagation delay, and serialization delay are small compared to the end-to-end completion times of the RDMA operations that are exchanged through the inter-GPU backend network.
- The port-to-port switching latency of a switch typically includes the forwarding delay and the queuing delay because both are caused within a switch, but as mentioned, only the queuing delay varies.
- The variance in the queuing delay increases the tail latency for RDMA operations. As mentioned earlier, a 400 GbE port takes 80 nanoseconds to transmit a 4 KB frame. If the packets of an RDMA operation are delayed due to waiting in the queue behind 10 similar-sized packets, the completion time of this RDMA operation increases (degrades) by 800 nanoseconds. Even if just one RDMA operation takes longer to complete, the entire collective communication to which this RDMA operation belongs is adversely affected.

How Cisco Addresses Network Delay

An optimal design should be the first step for reducing network delay. As mentioned, rails-optimized or a 3-ply network design lowers the network delay by allowing single-hop forwarding through the leaf switches without the traffic going to the spine switches.

Moreover, the cut-through switching architecture of Cisco Nexus switches helps minimize the forwarding delay even for large-sized packets. Besides the low latency, consistent and predictable performance on all the switch ports ensures that all the GPUs connected to different ports are subjected to the same network delays. This consistency helps to keep the tail latency to a minimal value, thereby increasing the overall performance of the collective communication within a GPU cluster.

Rich Quality of Service (QoS) features of Cisco Nexus switches allow prioritizing or using dedicated queues for small-sized response or acknowledgment packets so that they are not delayed while waiting behind the large-sized data packets in the same queue. Multiple no-drop queues or priority queues can also be used based on the inter-GPU traffic profile.

Cisco Nexus Dashboard automatically detects unusual spikes in network delay and flags them as anomalies. This latency information is available at a flow granularity and correlated with other relevant events such as bursts and errors. Getting real-time visibility into the network hot spots gives enough insights to a user to take corrective actions such as changing the QoS thresholds, increasing network capacity, or using an improved load-balancing scheme. To apply these actions uniformly across the network, Nexus Dashboard offers customizable templates (see Figure 6) and full-featured APIs.

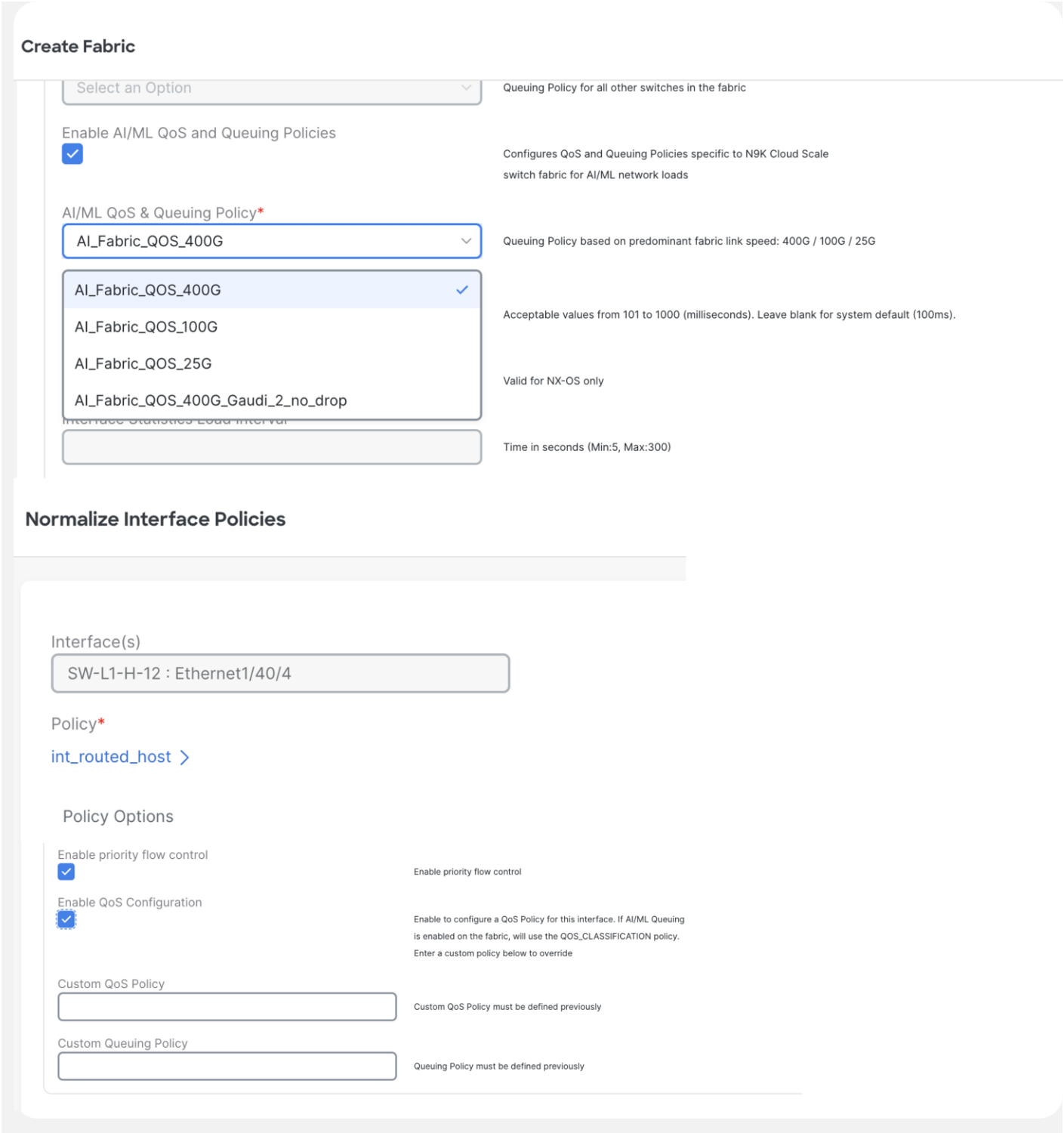

Figure 6.
Nexus Dashboard templates apply consistent traffic policies across the network

Challenge 3: Network Congestion

Why this is a challenge

Under congestion, the time a packet spends within a switch is much longer than the typical port-to-port switching latency, thereby increasing the completion time of the RDMA operations. If PFC is enabled to create a lossless network, congestion may even spread to the upstream traffic direction, victimizing many other GPUs exchanging traffic using the same paths. This network congestion delays collective communication, wastes the expensive compute cycles of the GPUs, and degrades the model training time.

What causes network congestion

The two common causes of congestion in a lossless network are slow or stuck end devices and overutilization of a link.

- **Congestion due to a slow or stuck GPU NIC port**: A GPU NIC port with a slower processing rate than the rate at which traffic is being delivered to it is called a slow NIC port. As mentioned earlier, GPUs can send and receive traffic at the full capacity of their NIC ports. But, sometimes due to faults, their traffic processing capacity may degrade. An example is if ports can only process 350 gigabits per second (Gbps), even when connected at 400 GbE. This is called a slow NIC port. Another example is the ports may stop the processing completely for a longer duration, reducing their traffic rate to zero. This is called a stuck NIC port. When PFC is enabled to create a lossless network, instead of dropping the excess traffic, the GPU NIC port sends pause frames to its directly-connected switch to reduce the traffic rate. The switch may absorb excess traffic within its buffers if this is a transient condition. But, if the GPU NIC port is stuck, the switch eventually reaches its buffering threshold, which sends the pause frames in the upstream traffic direction, typically to the spine switches. This congestion ultimately spreads to the traffic sources, which are the other GPU NICs trying to sync their states with this slow or stuck GPU. At this condition, PFC between directly attached devices achieves its goal of avoiding packet drops under congestion. However, the side effect of congestion spreading leads to adversely affecting the flows sharing the same network path even when they are not destined to the slow or stuck GPU NIC.
- **Congestion due to over-utilization of a link**: Congestion due to over-utilization happens when a switch port is transmitting at the maximum speed but has more packets than can be transmitted on that link. This condition can be caused by speed mismatch, bandwidth mismatch, or similar variants such as inadequate over-subscription and lack of network capacity. This condition causes congestion spreading similar to that caused by a slow or stuck NIC port. The congestion spreads toward the traffic source, victimizing many other GPUs that are using the same paths. Congestion due to over-utilization on the edge link may be observed when GPU NIC ports operate at mixed speeds, such as some operating at 100 GbE and others operating at 400 GbE. Avoid such speed mismatch conditions while designing the networks. Congestion due to over-utilization on the core links may be observed even with links with the same speed because of traffic imbalance issues. The next section explains traffic imbalance in detail.

How Cisco addresses congestion in inter-GPU backend networks

Cisco Nexus switches and the Nexus Dashboard platform provide rich features to detect, troubleshoot, and prevent congestion in an inter-GPU network.

- Cisco Nexus switches isolate a faulty GPU NIC port that is stuck and unable to receive traffic continuously for a longer duration, such as 100 milliseconds. After some time, if the GPU NIC recovers from its stuck state, the switch detects this change and allows this NIC port to

communicate through the network by de-isolating it. Both the isolation of a stuck NIC port and the de-isolation of a healthy NIC port are automatically performed by the Nexus switches. This feature, called PFC Watchdog, eliminates the source of congestion from the network, thereby avoiding the other GPUs from becoming victims of congestion spreading.

- Cisco Nexus switches can detect network congestion and then notify the GPU NIC ports using the Explicit Congestion Notification (ECN) field in the IP header. After learning about congestion in the network, GPU NIC ports may adjust their transmission rates to lower the severity of congestion.
  - To increase the effectiveness of this mechanism, known as Data Center Quantized Congestion Notification (DCQCN), Cisco Nexus switches allow fine-tuning of the ECN thresholds using weighted random early detection (WRED).
  - In addition, a unique feature called Approximate Fair Detection (AFD) allows ECN to have per-flow granularity instead of the per-traffic class granularity used by WRED.
  - A GPU NIC port, after learning about the network congestion from the ECN field in the IP header, may send a special-purpose Congestion Notification Packet (CNP) to the traffic source. The flexible QoS architecture of the Cisco Nexus switches allows the CNP to be transported using priority queues while continuing to transfer other inter-GPU traffic using no-drop queues.
- GPU NICs may implement congestion control mechanisms based on the real-time values of Round-Trip Times (RTT) for the inter-GPU RDMA operations. These mechanisms differ from DCQCN and therefore do not rely on ECN from the network. The accuracy of these RTT-based congestion control mechanisms depends not only on how fast an RDMA operation initiates, but also on how quickly the last response or acknowledgment packet arrives, completing that RDMA operation. For this purpose, Cisco Nexus switches can categorize the traffic into multiple classes to provide them with round-robin or priority forwarding through the network. This reduces the queuing delay of the small-sized response or acknowledgment packets caused by waiting behind the large-sized data packets in the same queue, thereby allowing the already-initiated RDMA operations to be completed quickly.
- Cisco Nexus Dashboard simplifies the interpretation of various congestion symptoms such as the number of pause frames sent or received, the number of ECN marked packets, and traffic sent and received on a link, by calculating a congestion score and an automatic assignment to mild, moderate, or severe categories (see Figure 7). The automatic correlations in Nexus Dashboard eliminate long troubleshooting cycles by accurately detecting the source and cause of congestion. This real-time visibility simplifies the buffer fine-tuning by showing fabric-wide congestion trends and providing the ability to consistently apply the fine-tuned thresholds across the network.

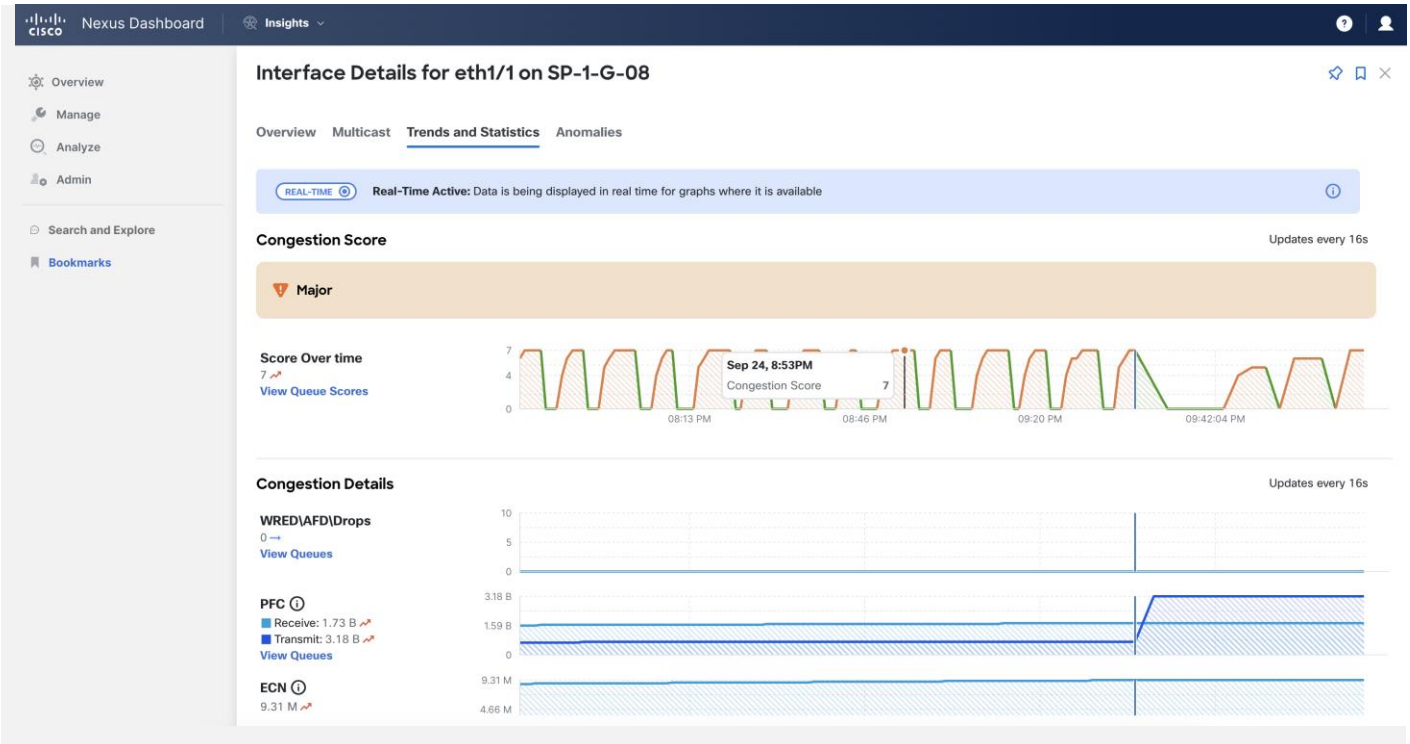

Figure 7.
Nexus Dashboard identifies the severity of network congestion by calculating a congestion score

Challenge 4: Non-uniform Utilization of Inter-Switch Links leading to Congestion due to Over-Utilization

Why this is a challenge

Non-uniform utilization of the network links, such as Inter-Switch Links (ISLs) can lead to congestion due to over-utilization. In lossless networks, this congestion spreads to the traffic source, victimizing many GPUs that share the same network paths. Consider a simple example of a spine switch receiving 800 Gbps to be transmitted on two 400 GbE links that go toward the leaf switches. Even a minor traffic imbalance leads to a condition where one link transmits at 98%, whereas the other link transmits at 100%. The excess 2% of traffic continues to occupy the switch buffers, but is not dropped in a lossless network.

What causes non-uniform utilization of ISLs

The default load-balancing scheme in the Ethernet switches is based on flows identified by five tuples in the packet headers. These tuples are source IP address, destination IP address, Layer 4 protocol, Layer 4 source port, and Layer 4 destination port. The switch feeds the five tuples of a packet to a hash function and uses its output to select one of the equal cost paths to transmit this packet. This scheme is called Equal-Cost Multipath (ECMP) load balancing. The benefit of this scheme is that the hash result for the 5-tuple is always the same. As a result, all packets of a flow are always transmitted on the same path avoiding out-of-order delivery of the packets to the destination.

One characteristic of the ECMP load balancing scheme is that it may assign more flows to a path than other paths leading to a non-uniform distribution of the number of flows.

Another limitation is that even if all paths have the same number of flows, the throughput of each flow may be different. As a result, the collective throughput of flows on a path may be different than that of the other

paths. This leads to non-uniform utilization of the equal-cost paths. In simple words, this scheme is unaware of the real-time link utilization.

Traffic imbalance between equal-cost links is common in a typical data center network, but generally it is not a major issue due to the following reasons:

1. A typical data center network has a large number of flows. Statistically, as the number of flows increases, their distribution among the links becomes more uniform.
2. Most flows have a low throughput, which are called mouse flows. Only a small subset have a high throughput, which are called elephant flows. As a result, even if the paths have a non-uniform distribution of the flows, the collective throughput of the flows on each path may not be very different.
3. A typical data center network rarely has links operating at 100% utilization. In other words, having some links at 60% utilization while other links are at 80% utilization can be acceptable because no link is congested and all links can still accommodate more traffic.

In contrast, the traffic patterns in an inter-GPU backend network are significantly different due to the following reasons:

1. The number of five-tuple flows is fewer. This is because each GPU NIC is assigned an IP address used as the source and destination for the RDMA operations. The number of IP addresses in inter-GPU networks is significantly smaller than in a typical data center network because of no virtual NICs adding to the number of IP addresses. Also, the size of the inter-GPU backend network itself is smaller, such as having only 512 IP addresses for a 512 GPU cluster. The other values of the five tuples also lack variation because UDP is the only Layer 4 protocol for RoCEv2 protocol and the Layer 4 destination port is always the same (4791). However, the Layer 4 source port may change depending on the implementation of the GPU NIC.
2. Most inter-GPU flows are large-throughput elephant flows.
3. GPU NICs can send and receive traffic at full capacity, such as 400 Gbps on a 400 GbE link. Not only is there 100% link utilization on one link, but all GPU links can witness full utilization simultaneously because of the basic nature of collective communication.

Overall, fewer elephant flows that collectively need full network capacity are much more likely to cause congestion in inter-GPU networks than a mix of many mouse flows and a few elephant flows in a typical data center network.

How Cisco addresses non-uniform utilization of ISLs

An adequate network design should be the first step in reducing ISL utilization. As mentioned, rails-optimized or 3-ply network design allows single-hop forwarding through the leaf switches. These designs eliminate traffic between leaf and spine switches for a training job distributed among the GPUs connected to the same leaf switches, such as 256 GPUs in Figure 4. When a job runs on the GPUs connected to different leaf switches, the traffic on ISLs is much lower than in a network not optimized using rails or 3-plys. This is because only a portion of traffic passes through the spine switches to reach the GPUs connected to different leaf switches, whereas the other traffic remains local to the leaf switches.

However, as GPU cluster size increases, the traffic passing through spine switches also increases. In these cases, traffic imbalance continues to be a prominent cause of congestion due to over-utilization of the ISLs. This section explains how this challenge can be addressed using the features of Cisco Nexus switches and the Nexus Dashboard platform.

Dynamic Load Balancing (DLB)

Cisco Nexus switches can dynamically load balance (DLB) the traffic by estimating the utilization of the equal-cost paths and then sending traffic on the least utilized paths (see Figure 8). DLB identifies flowlets based on the gap between the bursts of packets in a five-tuple flow. Each flowlet can be transmitted on a different equal-cost path based on the transmission rates of the paths estimated in real time. To avoid the reordering of the packets of an RDMA operation, the flowlet identification logic has been carefully designed based on enough gap between the bursts.

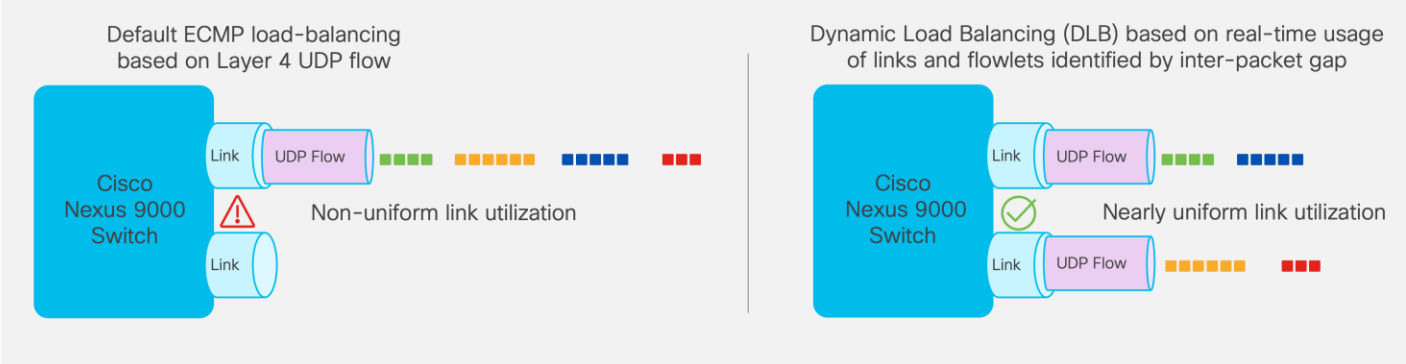

Figure 8.
Dynamic load balancing on Cisco Nexus 9000 Series switches improves uniform utilization of equal-cost paths

Static Pinning

Another feature of Cisco Nexus switches is the ability to pin the traffic statically from the edge ports that are connected to GPUs to the core ports that are connected to spine switches (see Figure 9). This pinning is created one-to-one with the basic premise of a non-blocking network design. For example, all traffic from a 400 GbE switch port connected to the GPU NIC port is switched only to a statically-defined 400 GbE port connected to a spine switch. Static pinning makes the load balancing completely deterministic. All links can be utilized at their full capacity and no individual link is subjected to more traffic than can be sent on it.

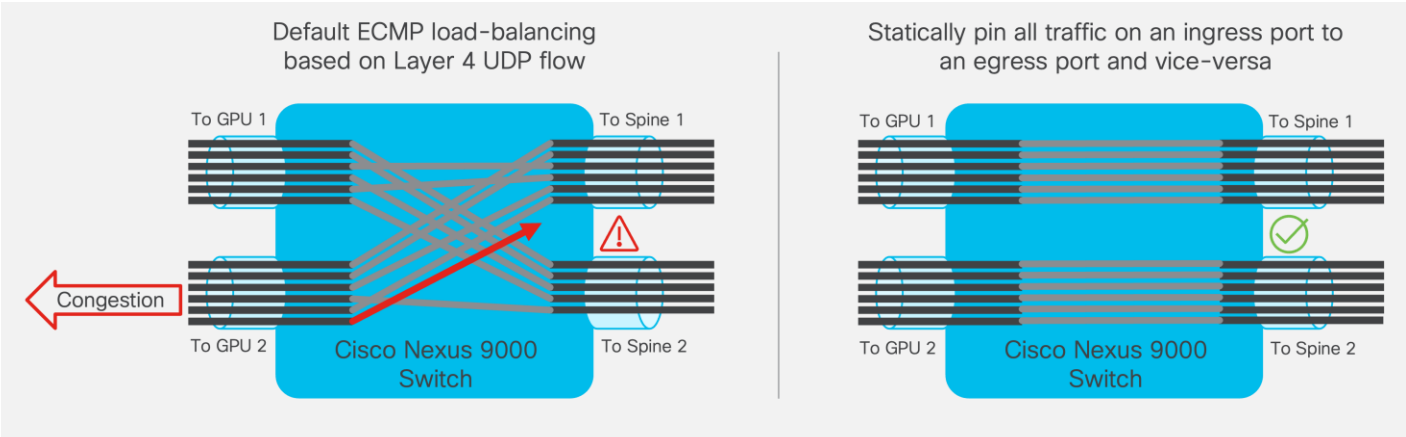

Figure 9.
Static pinning on Cisco Nexus 9000 Series Switches makes load balancing on equal-cost paths completely deterministic

Ability to identify and load-balance based on smaller-throughput flows

Cisco Nexus Switches can detect multiple smaller-throughput flows within a single large-throughput flow, which can then be used for load-balancing across equal-cost paths leading to improved uniform utilization (see Figure 10).

The logic to detect smaller flows is flexible based on a user-defined field (UDF). Instead of identifying flows based on the traditional five-tuples, Cisco Nexus Switches can identify flows based on InfiniBand destination queue pair to load-balance the traffic in equal cost paths. Consider two GPUs exchanging traffic at 400 Gbps, with all packets identified with the same UDP flow. The default 5-tuple load-balancing scheme would transmit the entire 400 Gbps of traffic on just one of the many equal-cost paths. Assume that the GPUs use eight IB queue pairs, each serving equal throughput. By load-balancing the traffic based on these queue pairs, the switches can send traffic on up to 8 links. This would make link utilization more uniform, thereby reducing the likeliness of congestion due to over-utilization of one of the links while other links remain under-utilized.

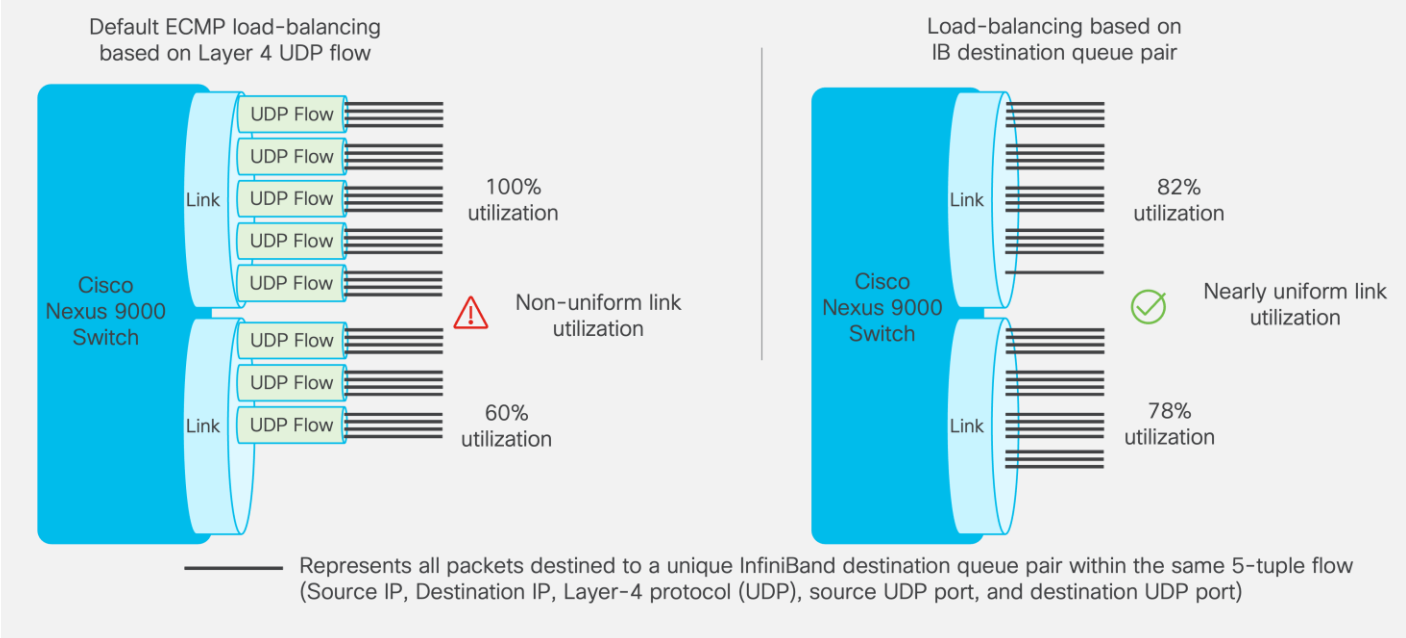

Figure 10.
Cisco Nexus 9000 Series switches can identify flows based on InfiniBand destination queue pair to load-balance the traffic on equal-cost paths

Cisco Nexus Dashboard

Cisco Nexus Dashboard simplifies changing one load balancing scheme to another and detecting the scheme that gives the best results. The configurations can be consistently applied on all switches or even a different scheme can be configured on leaf switches and spine switches. Nexus Dashboard shows the real-time usage of the links that can be used to verify the outcome of these load balancing schemes.

Challenge 5: Simultaneous Elephant Flows with Large Bursts

Why this is a challenge

The switches in the inter-GPU networks must have an intelligent buffering architecture to handle frequent large bursts on all switch ports simultaneously. Even minor inefficiencies in managing the buffer space or an unfair allocation of buffers among the switchports degrades the performance of the RDMA operations, thereby adversely affecting the collective communication between the GPUs and the model training time.

What causes simultaneous large bursts

Simultaneous large bursts occur when GPUs sync their states while running a distributed training job. These simultaneous bursts are observed repeatedly because the GPUs complete their calculations simultaneously, and they must learn the calculated results of the other GPUs before moving to the next step. These bursts are observed on all switch ports connected to the GPU NIC ports.

How Cisco addresses simultaneous large bursts

Cisco Nexus switches use a shared-memory egress-buffering architecture. Large bursts on a port can be absorbed by buffers shared among a set of ports. An intrinsic feature of this architecture is Dynamic Buffer Protection (DPB), which achieves fair buffer usage across all ports and queues preventing an elephant flow from starving other flows for buffers. This dynamic scheme delivers the best performance for the front-end networks and the storage networks for the AI/ML infrastructure.

To boost the performance of the inter-GPU networks, Cisco Nexus switches can also dedicate buffers per port. This feature makes buffer allocation even more efficient, thereby improving the handling of frequent large bursts on all the switch ports simultaneously. This dedicated buffering scheme has shown up to 17% performance improvement in the AllReduce and ReduceScatter collective communications in an inter-GPU backend network (See Figure 11).

Figure 11.
Cisco Nexus 9000 Series switches innovations show up to 17% performance improvement for inter-GPU collective communication

Conclusion

Cisco Nexus switches have a proven track record of successful deployments exceeding the requirements of the front-end network and storage network use cases. The inter-GPU backend networks have additional challenges such as simultaneous large bursts, traffic imbalance leading to congestion, and all GPUs in a cluster sending and receiving traffic simultaneously. Cisco Nexus switches address these challenges by purposefully designed features, such as Dynamic Load Balancing, the ability to identify and load balance on smaller-throughput flows, and dedicating buffer per port for lossless traffic.

The configuration can be automated using the full-featured APIs for Cisco Nexus switches and Nexus Dashboard. The real-time telemetry from the Nexus switches can be consumed within Nexus Dashboard or exported to other tools that may already be in use.

Moreover, Nexus Dashboard complements the data plane features of the Nexus switches by simplifying their configuration using built-in templates, which can be customized for special use cases. It detects network health issues, such as congestion, bit errors, and traffic bursts in real time and automatically flags them as anomalies. These issues can be resolved faster using integrations with commonly used tools, such

as ServiceNow and Ansible, allowing the inter-GPU networks to be aligned with the existing workflows of an organization.

The fully integrated Data Center Networking solution of Cisco switching ASICs, NX-OS operating system, and Nexus Dashboard platform improves the network uptime and the performance of the inter-GPU communication. These benefits speed up the model training time, thereby increasing the return on investments from the AI/ML infrastructure.

References

1. Cisco Validated Design for Data Center Networking Blueprint for AI/ML Applications
2. Cisco Data Center Networking Blueprint for AI/ML Applications
3. RoCE Storage Implementation over NX-OS VXLAN Fabrics – Cisco White Paper
4. NextGen DCI with VXLAN EVPN Multi-Site Using vPC Border Gateways White Paper
5. Cisco's Massively Scalable Data Center Network Fabric White Paper
6. Meeting the Network Requirements of Non-Volatile Memory Express (NVMe) Storage with Cisco Nexus 9000 and Cisco ACI White Paper
7. Intelligent Buffer Management on Cisco Nexus 9000 Series Switches White Paper
8. Getting Started with Model-Driven Programmability on Cisco Nexus 9000 Series Switches White Paper
9. Benefits of Remote Direct Memory Access Over Routed Fabrics White Paper

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA Cpg-dcnaiml-01 10/24