DRIVENETS

Fabric-Scheduled Ethernet as an Effective Backend Interconnect for Large AI Compute Clusters

WHITE PAPER

Contents

Introduction 3
AI Compute Server Architecture 3
  GPU Dominance 3
  GPU Server Architecture Challenges 4
Networking Challenges in AI Compute Server Arrays 5
  Two Types of DC networking 5
  Third DC Networking Type: Infrastructure for AI 6
  The GPU Allocation Challenge 7
  The Workload Behavior Challenge 7
  The Versatility Challenge 7
Networking Solutions for AI Clusters 8
  Naïve Ethernet 8
  InfiniBand 10
  Distorted Ethernet 12
  Single Chassis 13
  Utilizing Disaggregated Distributed Chassis for AI/HPC Backend Networking Fabric 14
Internal Fabric 15
  Black Box Functionality 15
  Vendor Lock Concerns 15
Benefits of a Scheduled Fabric 16
  Maximum Scale 17
  High Bandwidth and Reduced CBB 17
  Fast Failure Recovery 18
  Consistent Latency 19
  Flow Agnostic 19
  Lossless Behavior 20
  Telemetry Collection 21
  Managing Different NOS 21
Examples: AI Backend networking Scheduled Fabric in Action 22
  Mean JCT Time of Multiple Jobs 22
  Checking the Impact of Failure Scenarios 23
Comparison Table of AI Cluster Solutions 25
AI/HPC Scheduled Fabric Meets the Challenges facing an AI-enabled Data Center Infrastructure 26

Introduction

Artificial intelligence (AI) has evolved in recent years to meet a larger array of use cases and applications, as AI applications reach a wider audience. While the user experience becomes more simplified and intuitive, the computation model required to generate this logic becomes larger and more complicated. Companies providing large cloud infrastructure, such as Microsoft, Google, Meta, and ByteDance, are developing new AI models and introducing AI capabilities into their infrastructure to expand the array of services offered to their users. While the monetization from these newly deployed services is still unclear, even less clear is what future applications will join this line-up and what infrastructure will be required to enable them.

The required infrastructure for handling the computing needs in these newly developed models, and those to come, are driving market momentum toward substantial compute capacity. This dramatic growth in market volume has triggered an "arms race" with Nvidia, acting as the main 'weapons' supplier, while alternative suppliers are gradually entering this space.

This paper will cover the market landscape on the compute side, focusing on the networking solutions that support connecting these large arrays of servers. Exploring these solutions' attributes, advantages and caveats, this document will also take a deep dive into the Distributed Disaggregated Chassis (DDC) architecture and how it fits the AI backend networking fabric use case.

AI Compute Server Architecture

Since the introduction of the graphical processing unit (GPU) at the turn of the century, the compute world has been gradually drifting from central processing unit (CPU) dominance, led by the x86 architecture, towards GPU dominated systems. In such systems, the CPU plays the role of managing the GPU resource, not acting as the compute resource by itself.

GPU Dominance

The high-performance computing (HPC) industry keeps track of the world's top 500 supercomputers. In 2018, the industry registered the world's fastest computer named "Summit", based on GPUs from Nvidia and CPUs from IBM, that used its later deprecated "Power" architecture. Marking the end of CPU dominance of the HPC space in March 2014, the U.S. Department of Energy announced a $425 million budget towards research and construction of supercomputers. "Summit" was planned to be built to work at almost triple the speed of China's Tianhe-2 supercomputer, at performing at 150 petaflops.

Today's top compute servers comprise multiple GPUs connected in a type of a mesh between them and other such servers. The most common architecture coming from Nvidia is the DGX/HGX, which hosts 8 GPUs and a proprietary fabric between them called NVLink. Other architectures also incorporate GPUs coming from AMD and Intel, while other technologies have been developed with the capability to target certain compute tasks, though not necessarily "general purpose" capable. The most widely deployed example is Google's TPU portfolio, which is now at its 5th "generation" - with each generation focused on different workload types.

While Google's TPU is the most well-known (or most communicated) acceleration hardware (HW), other hyperscale companies are also developing self-built accelerators, some already in deployment.

GPU Server Architecture Challenges

In parallel to the takeover by the GPU of the HPC space, the AI world has evolved and outgrown the HPC domain in terms of volume and compute capacity. This triggered another race between startups aiming to take substantial market share in the AI-compute space. Companies such as Cerebras, Groq, SambaNova, Graphcore, and more all aim to build some of the larger compute infrastructures, expected to come soon.

Power Consumption

Arrays of GPU servers are typically built into standard 19” racks. With the power consumption of the high-capacity GPUs reaching higher levels with every generation, the number of servers hosted in a single rack are dictated by the power feed to that rack.

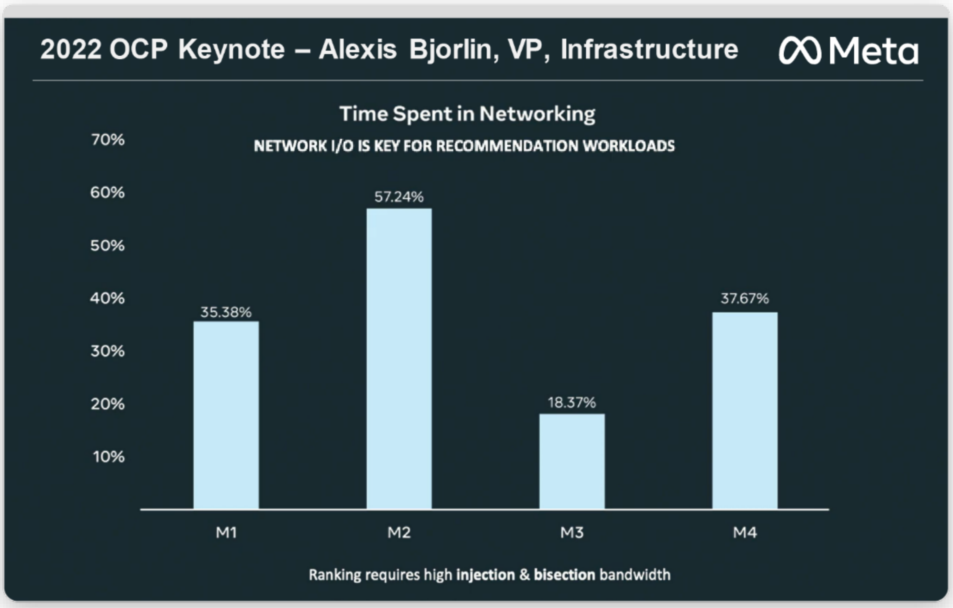
Top-of-rack (ToR) Topology

A typical data center has a top-of-rack (ToR) switch/router directly connected to all the servers hosted in that rack, connecting them to other ToR devices via a multi-tiered network.

With the variety of different server architectures appearing with the multitude of compute engines, the standard ToR topology is what allows the flexibility to connect different types of devices with a unified architecture.

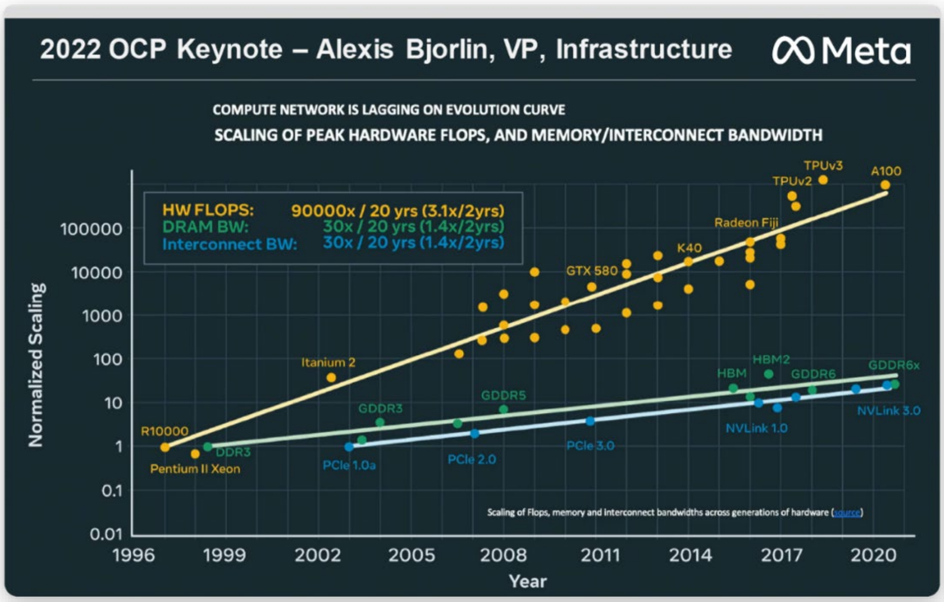
GPU Allocation for Processing Tasks

Another challenge evident with the different types of compute engines is the allocation of the GPU to process certain tasks. With parallel compute tasks spanning up to tens and hundreds of processors, the allocation of compute engines is a task that, when done properly, can raise the utilization level of these compute engines to high levels. This is a crucial factor for making an efficient compute array, due to the large expense involved in building such infrastructure, with the most expensive element in these arrays being the GPU server itself (80-90% of the expense).



Bar chart showing time wasted while waiting for the network to deliver a needed payload. Time wasted means GPU idle cycles, translating to wasted money that was already spent. Source: Meta Keynote, OCP Summit 2022

While allocating compute resources properly increases GPU utilization, it assumes that the network directly connects all GPUs in the array (which scales to many thousands) and promises lossless high-bandwidth (BW) and congestion-free network connectivity. While this is a fair assumption for job schedulers when properly allocating GPU per tasks, it is quite a challenge to establish such a high performing network.



Graph showing the gap created between compute growth and network capacity of a single GPU. This puts more emphasis on the network side for properly scaling out an array of GPUs. Source: Meta Keynote, OCP Summit 2022

Networking Challenges in AI Compute Server Arrays

The world of networking has distinct groups. Telecom, unlike enterprises, differs from campuses, which differs from data centers - just to name a few differing network types. Within these categories there are also subcategories, and within data center networking there are typically two types.

Two Types of DC networking

One network type is the naïve Ethernet. It is “naïve” in the sense that it implements a “best effort” of a lossy technology for scaling tens of thousands of servers connected between themselves as well as towards external domains - like the internet. This technology has many vendors and a healthy, competitive and fairly large market.

Another network type within the data center is often referred to as a backend network. This network is active in the non-interoperable side of the data center, connecting between storage nodes or compute nodes, and is disconnected from outside the data center. The term backend indicates location but it differs from the Naïve Ethernet because of its attributes. A backend typically has more stringent demands from its network, expecting:

- lossless behavior
- predictability of throughput and latency
- high bandwidth
- high density of the deployment environment

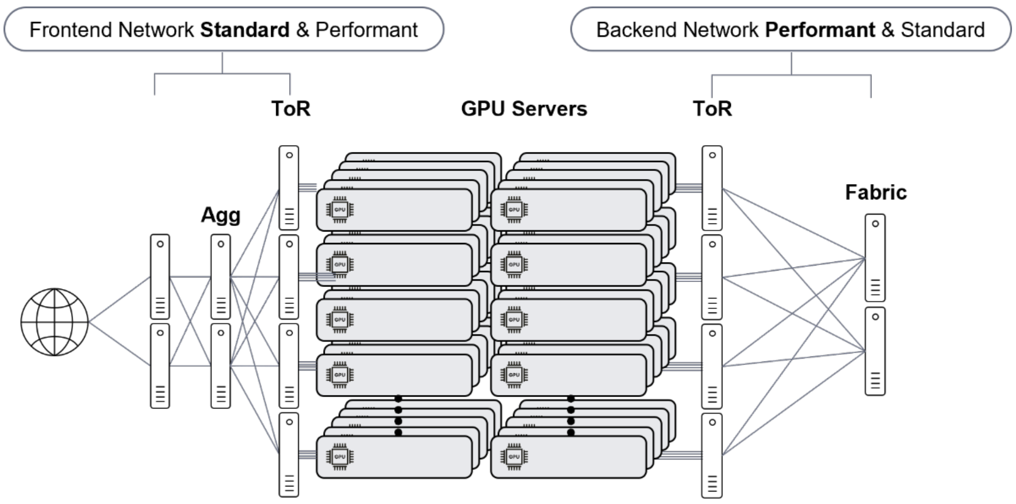
It is these same attributes that negated use of the naïve Ethernet, allowing for niche standard and proprietary technologies to evolve into these data center domains. The HPC domain is one such example of a use case where a backend network technology (e.g. Slingshot, InfiniBand) flourished.

The impact of a lossless network has a triple effect. To increase GPU utilization, one needs to look at where GPU cycles are wasted.

- By poorly allocating GPUs per job, meaning that an available GPU is not allocated and stays idle or that the wrong GPU is allocated (wrong GPU in this sense means that there exists an allocation that would result in a shorter job completion time (JCT).); this is the task of the job scheduler which "assumes" that traffic between GPUs always arrives at its destination and this is outside the scope of this paper.
- Within the network itself, clearly increasing cross bisectional bandwidth (CBB), which will be discussed in detail throughout this paper.
- The preparation of the datasets for the GPU to work on, where this can be improved with a better interconnect on the GPU server itself. A network interface card (NIC) with better offload capabilities and remote direct memory access (RDMA) would have the ability to prepare the payloads carried by the network into datasets for the GPU to work on in a faster way. This frees the GPU from dealing with such tasks (wasting compute cycles) and/or waiting for the NIC to prepare new datasets to process. From the NIC perspective, in order for RDMA to work, the network is mandated to be lossless. Whether implemented over InfiniBand or RDMA over Converged Ethernet (RoCE), RDMA realizes its efficiency from bypassing software (SW) procedures for moving data between servers; some of these SW procedures also include the checks that the traffic has arrived and is intact. A faulty network (lossy or otherwise noisy) would terminate the advantage of using RDMA.

Third DC Networking Type: Infrastructure for AI

With the rise of AI-induced installations, although quite different from HPC, the compute array still requires the same attributes from the network as found in a backend. However, this AI infrastructure should be part of a cloud and enable native connectivity to domains outside the data center. This combination creates a third flavor of data center networking that should be both performant and based on a widely used standard.


Frontend Network **Standard** & Performant
Backend Network **Performant** & Standard
ToR
GPU Servers
ToR
Agg
Fabric

Typical data center array formation showing two separate networks while the introduction of AI will push both networks towards each other into a third type or perhaps convergence of the two types. Source: DriveNets

There is some debate whether this third type of DC networking will consume either of the existing two domains or perhaps even both domains into itself. This paper will not discuss this debate and instead focus on this newly formed type and its attributes.

The GPU Allocation Challenge

GPU servers are evolving fast, and GPU capacity can reach a network feed of 400G, while expected to grow to 800G soon. When running a heavy workload, a group of GPUs can all utilize their network links to their maximal capacity and create a maximal throughput within the network. At a scale of thousands of GPUs, clearly it is not an option to connect all GPUs to a single device, and so a second tier to the network exists. The BW carried by the network's second tier is referred to as cross bisectional bandwidth (CBB). When multiple GPUs transmit/receive large workloads simultaneously, the CBB spikes. This is a rather common scenario resulting from how GPUs are allocated in a parallel compute algorithm.

The Workload Behavior Challenge

Another challenge for the network is the time it takes for messages to run through the network between endpoints – not only flat latency but also the variation of delays between messages. This is due to the fact that GPUs are kept hanging until they receive full acknowledgement that a compute cycle has concluded. A long tail latency would eliminate any value of minimal head latency.

The Scale Challenge

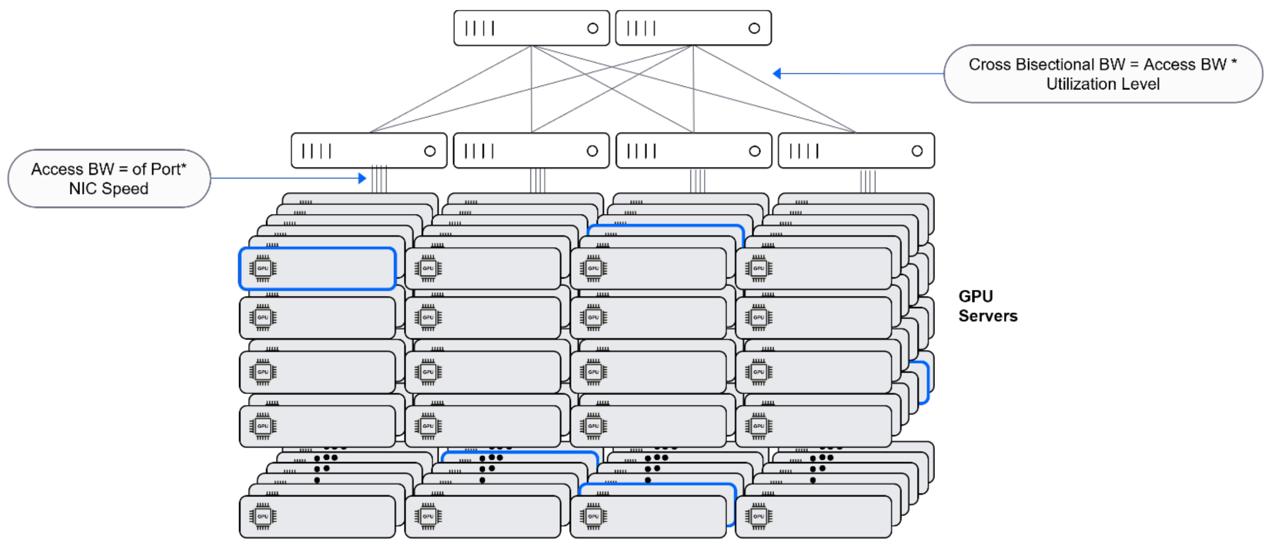
AI workloads are large and getting larger. Model accuracy and the race to outpace the competition is calling for larger infrastructure to run on. Whatever challenges or attributes of the networks which are noticeable in moderate scale are multiplied when scale increases.

The Versatility Challenge

Another major challenge expected from the network is dealing with different traffic patterns and workload characteristics. The various AI models run differently, and often use different communication algorithms such as:

- All-to-all
- Many-to-one
- One-to-many
- All reduce

Different packet sizes and flow sizes represent some of the different characteristics of the traffic that the network needs to propagate. The AI network cannot be fine-tuned to one specific traffic pattern but must be satisfying them all.


Cross Bisectional BW = Access BW * Utilization Level
Access BW = of Port* NIC Speed
GPU Servers

The marked GPU represent GPUs allocated for the same job which translate to high traffic between these endpoints. The network needs to fulfill any traffic patterns without discrimination in order to increase its utilization level. Source: DriveNets

These challenges have created hurdles for network solutions from both types of data center networking. Each of these solutions, however, have drawbacks, holding them back from maximizing network utilization.

Networking Solutions for AI Clusters

There are multiple solutions for data center networking, all of which can be considered as candidates for a networking solution for AI clusters. This paper will investigate the pros and cons of some leading alternatives.

Naïve Ethernet

The most deployed network type within data centers is naïve Ethernet, and there are several attributes that have made it such a strong leader. This solution consists of:

- ToR switch/router connected to servers
- Aggregation layer (typically 1 or 2 layers) switches connected to ToR switches
- Routing uplinks toward the aggregation layer in a Clos-3 or Clos-5 topology

It introduces the following attributes:

- Very large scale and high bandwidth towards the GPU network ports
- Standard behavior and common connectivity with devices outside the data center

Implementations of such networks exist from many silicon vendors, with the main ones mentioned below:

- Broadcom (Tomahawk, Trident ASICs)
- Nvidia (Spectrum ASIC)
- Marvell (including the Xpliant and Innovium products)
- Cisco (multiple self-built ASICs)

- Intel (Barefoot, QLogic, as well as self-built)

Systems and software include vertically integrated solutions from systems vendors as well as open source solutions like Open Compute Project (OCP) for HW and SONiC for software.

While the naïve Ethernet solution scores high on openness and standardization, it fails on the performance side.

- Although the bandwidth towards the GPU is high, the CBB deteriorates rapidly when the overall throughput of the network is expected to rise.
- While flat latency of a single packet via a single switch can be very low (employing cut-through latency), the different paths for a packet to take between endpoints results in a high tail latency and impacted GPU performance.
- Because traffic is carried over from the first tier of the network to the higher tiers via a hash-based distribution algorithm (like ECMP), it is, by definition, sensitive to flow characteristics and therefore not balanced.

These caveats mean that in a large environment, many GPUs will be left idle through long durations due to the network's naïve behavior.

Attempts to improve the naïve Ethernet Clos

As noted earlier, naïve (AKA "best effort") assumption of a perfect network in fact means "worst effort" and with this being the worst effort, it can clearly be improved. Several attempts to improve the naïve Ethernet Clos (spine-leaf) topology have been defined and implemented.

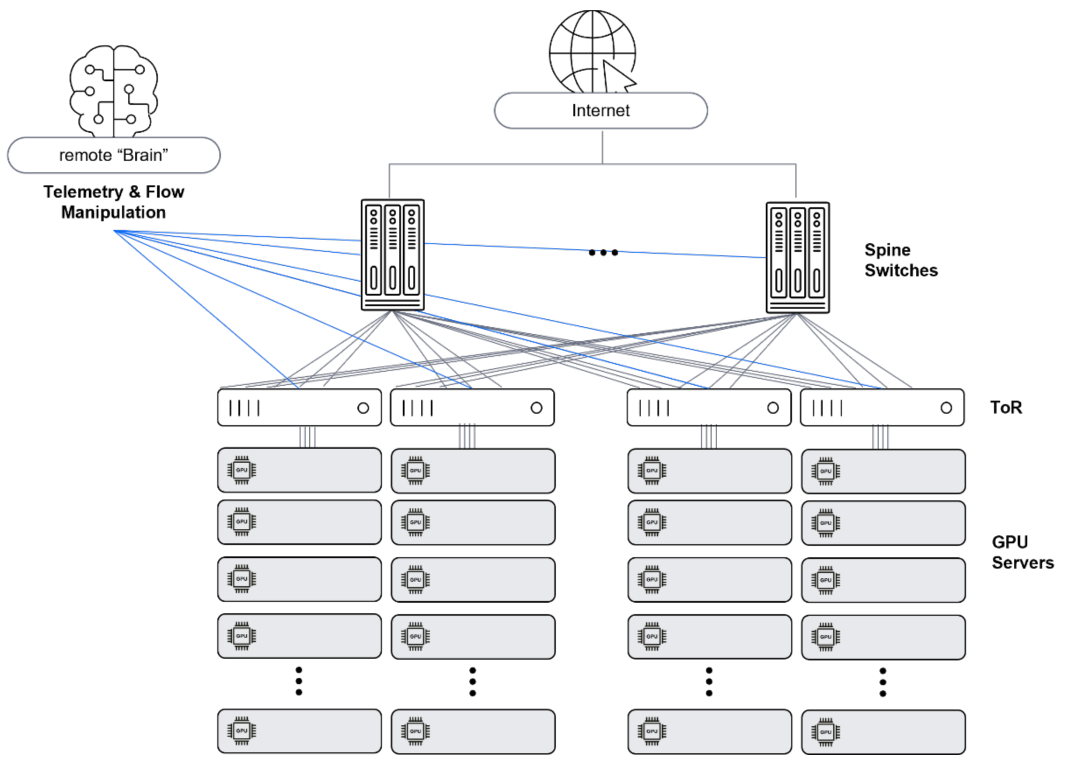
Some Examples include:

- Fine granularity flow-control mechanisms with forward and backward notifications
- Noise-induced hashing functions to force distribution of large flows over multiple tier #2 (spine) devices
- Adaptive routing to enforce alternative paths for certain flows that are outside the standard routing protocol
- And more

These additions (some of which are proprietary) have made the naïve Clos solution less standard while their introduced improvement in performance has been marginal.

Job allocation of GPUs

Another approach for dealing with Clos topologies is the job allocation of GPUs from within the same branch of the Clos. This means traffic between GPUs will not bypass the higher layers of the network and will not require any CBB at all. While not really resolving but avoiding (some of) the inherent problems of Clos, it enables the use of Clos in some specific AI use cases. This approach moves the burden of making this work towards the job scheduler, which needs to consider the physical location of GPUs. In some cases, this means that although the cluster has sufficient compute to run a job, the job will not run because these are the "wrong" GPUs to allocate.



Typical naïve Ethernet Clos with "enhancements" – a "best effort" attempt to force lossless behavior on a lossy technology. Source: DriveNets

InfiniBand

Introduced in 1999 (the same year that Nvidia introduced its first GPU) as a solution for connecting servers in a high-capacity performance network, InfiniBand has all the characteristics for a successful backend solution.

As a newly formed standard, it included functions that Ethernet did not originally possess, including:

- Loss preventing flow control giving InfiniBand lossless behavior
- Adaptive routing avoiding the use of hash algorithms
- Careful flow handling giving the network the ability to cope with unique traffic characteristics

Dominant standard in HPC domain

InfiniBand grew to become the predominant technology in the HPC domain where the compute clusters were rarely (if at all) connected to devices outside the compute cluster. While InfiniBand grew to develop dominance with HPC, it failed to make any meaningful penetration into other data center domains (where Ethernet stayed as the primary option). This led most (all but one) of the InfiniBand vendors to step out of this technology domain, abandoning their product portfolios. The sole vendor developing InfiniBand solutions was Mellanox Technologies, which was acquired by Nvidia in 2020.

Nvidia is the only InfiniBand vendor

While InfiniBand is officially a standard, its single supplier model largely negates the idea of deploying a protocol. Connecting an InfiniBand network to devices outside it's compute network is cumbersome and a performance obstacle. Single vendor technology is a risk but it is rather safe to assume that Nvidia will not terminate this business any time soon (or collapse which will have a much larger effect than a network impact); having said that, it is still a risk from the perspective of whoever is building the AI cluster to source too many components from the same vendor. Strong vendors are good, as they give customers a good night's sleep. Dominant vendors are bad, as they govern customers' infrastructure, leaving them without control.

AI Clusters introduce more endpoints

Several more caveats concerning InfiniBand have to do with the difference between HPC and AI compute clusters, which have similarities but are not exactly alike.

AI clusters can grow much larger than HPC clusters, and they are expected to get even larger. InfiniBand is a software-defined networking (SDN) technology that was designed for hundreds up to a few thousands of endpoints. AI clusters are already targeting tens of thousands of endpoints per single cluster. While in theory InfiniBand can reach these numbers, in practice there are the familiar echoing sounds of failure when SDN has attempted to scale significantly.

Not ready to scale-up for faster GPU servers

Another scale-related issue is the scale-up factor of the GPU server. While it is at 400G today (NDR in InfiniBand terminology, which was just recently announced), 800G is just around the corner and the cadence between InfiniBand speeds is typically 4-5 years. There is the possibility of Nvidia delaying their push towards an 800G-capable GPU and risking their advantage as a GPU vendor to allow NDR InfiniBand a larger window of opportunity.

Lacks tuning

A final caveat for InfiniBand comes from the tuning it enables for specific applications and workflow attributes. In the HPC space, this was an advantage that improved performance and made InfiniBand better than the alternatives. In the AI space, however, with its multitude of changing workflows and traffic patterns, the ability to fine tune is not practical and the lack of tuning generates poor performance.

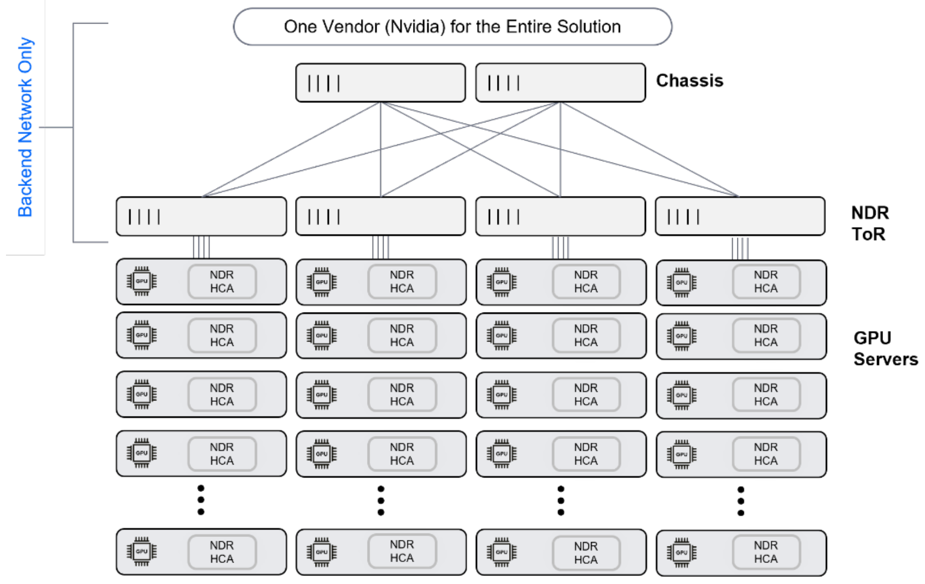
No test results for large AI clusters implementations

Considering there are very limited or no known test results available for InfiniBand in large AI clusters, it is a safe assumption that there is reason to hide such results. It is, however, a known that large language models (LLMs) such as ChatGPT were trained over an InfiniBand network. So clearly the solution is capable of some performance level, although apparently lower than what Nvidia would like the market to think.

Nvidia vendor lock puts performance at risk

There is another claim made by Nvidia that is a concern: The full alignment with SerDes speeds from network to GPU grants a performance boost. Is it fair to assume that replacing the interconnect will degrade the performance of the GPU itself? Is it fair to assume that InfiniBand (or any other Nvidia network) will not perform as well when the GPU is not from Nvidia?

So, InfiniBand doesn't shine on scale aspects and workload variation, as well as introduces a bothersome vendor lock with the main GPU provider.



Typical HPC Clos topology InfiniBand backend (only) network at the max speed of NDR (400G) provided from a single vendor. Source: DriveNets

Distorted Ethernet

There are several levels of Ethernet as a standard. While implementations can be aligned with the Ethernet frame format, they also can introduce non-standard traffic-handling behavior. Let's call these options "distorted Ethernet."

Such solutions include those that:

- perform well in the HPC space (like HPE's Slingshot) with their AI behavior left as TBD
- collapsed into larger companies (Like Qlogic's and Barefoot's collapse into Intel) and might be wearing some different name
- are well marketed and yet-to-be-seen technologies (like Cisco's Silicon One cluster and Nvidia's Spectrum-X solution)

Varying traffic flows

None of these solutions have a proper description publicly available, nor a communicated deployment within this domain, so the analysis here is partial at best. Some solutions claim best scale (Spectrum-X and Silicon One), some claim great SDN control and packet flow manipulation (Barefoot), and some claim great performance in HPC (Slingshot). Yet it is worth noting that all these solutions are aware of their carried traffic flows; with the varying traffic patterns, and potentially repeating characteristics between flows, this could result in poor CBB and severe performance degradation.

Untested claims for AI workloads

Also worth noting is that the newly announced solutions (Spectrum-X and Silicon One) claim to be designed for AI workloads; so while there are no details on how they handle flows, there is the possibility that there is some hidden mechanism. Cisco claims that packets are "sprayed" over the spines, which yields TBD performance. Nvidia claims that its solution dictates the path of a flow from the endpoint and applies adaptive routing (much like InfiniBand), which make its solution proprietary and mandates a complete end-to-end Nvidia solution (again, much like InfiniBand).

Single Chassis

A potential solution of some validity would be to avoid all the potential mishaps of a network and connect all GPUs to a single network element. Anything to do with interfering flows, traffic bursts, predictability, and so on would all be removed if all GPUs would connect to the same chassis-based device.

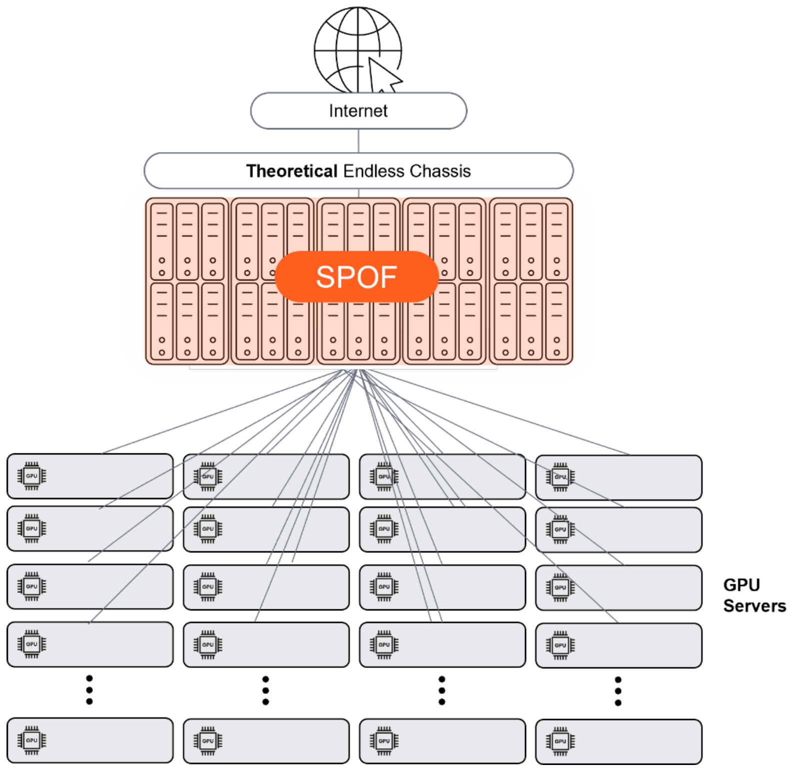
Unrealistic solution for AI cluster

While some solutions of this topology do exist, this is mainly in areas where network know-how is scarce and money is abundant – but not for performance reasons and surely not at scale. Considering an AI cluster of 2000 ports, which is by today's standards a small compute array, a chassis of the same size would be almost impossible to build. There is a claim of such an InfiniBand chassis, but it is only available on paper. Using such a machine would require logistics and operational overhead to power and cool it as well as to host it in any standard rack.

Operational nightmare

Once in place, pulling in 2000 optical cables from ~100 neighboring racks to connect to the chassis would be an operational nightmare for installation and even more so for maintenance of these optical links. Using copper (passive or active), which is far more reliable and less prone to failures, would not reach the range needed to connect so many different racks together.

While a single chassis in theory could be a great solution, in practice its operational challenges could not be overcome, and it would constitute a single point of failure (SPOF).



A potential single "unlimited scale" chassis formation creates a clutter of cabling and poses a single point of failure for the entire compute array. Source: DriveNets

Utilizing Disaggregated Distributed Chassis for AI/HPC Backend Networking Fabric

Disaggregated Distributed Chassis (DDC) scheduled fabric is an emerging technology defined under the telco project of the Open Compute Project (OCP). Defined by AT&T to use forwarding ASICs from Broadcom, it has seen implementations of its white box hardware by UfiSpace, Delta and Edgecore, and uses software from DriveNets and Cisco. AT&T also has been the first to implement the DDC model into their network to run both core MPLS and peering routers.

Same advantages as Clos topology

Like chassis construction, DDC scheduled fabric utilizes Clos topology, with the advantage of chassis such as reliability and performance, and the flexibility of Clos topology using small white boxes. Most components from the chassis have been implemented as standalone devices that can be "reconstructed" by software to act as a unified network element.

Software-based

The one component of the chassis that did not get a dedicated device to represent it in DDC scheduled fabric is the chassis device itself. The metal enclosure that holds it all together and guarantees a lot of the control, management and data planes attributes of a chassis has been replaced with software. This software orchestrates all the various components for this distributed chassis.

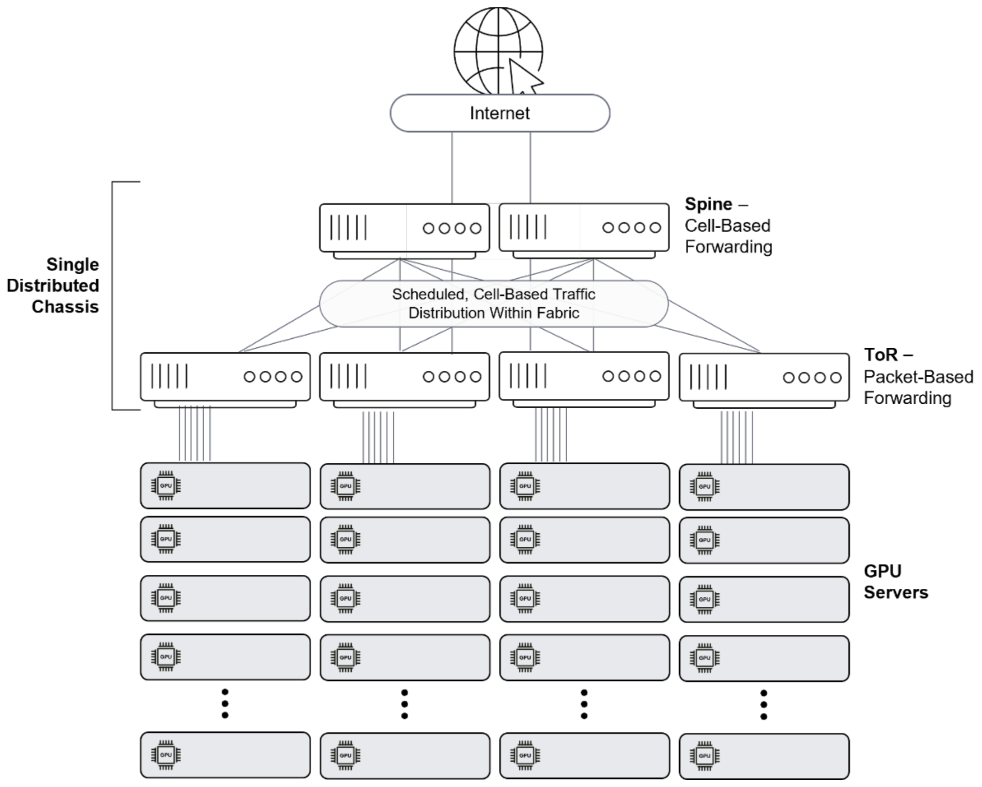
Scalable

Besides its advantages, the classic chassis enclosure is also a limitation. It physically limits the size of the deployment, the number of hosted ports, the power it consumes, and the amount of heat it dissipates. Implementing a DDC as an AI backend networking fabric means that the backplane breaks free of these limitations and can grow to the max radix (and even beyond) of the ASIC. Power and heat are distributed between various standalone devices, each of which can be placed in a different rack where it is closer to its directly connected endpoints.

DDC as Data Center Interconnect (DCI) solution

While originally built to be used as a large and versatile multiservice router, the DDC model can be used as a data center interconnect (DCI) with DriveNets' Network Cloud Packet Forwarder (NCP), where the NCP acts as a ToR and DriveNets' Network Cloud Fabric (NCF) acts as a spine. This solution enjoys all the attributes of a carrier-grade router and the almost unlimited scale of the ASIC's density.

How DDC copes with the demands of an AI backend cluster and how scheduled fabric solutions are differentiated will be discussed in the next chapter.


Internet
Spine – Cell-Based Forwarding
Single Distributed Chassis
Scheduled, Cell-Based Traffic Distribution Within Fabric
ToR – Packet-Based Forwarding
GPU Servers
GPU

DriveNets AI backend networking fabric topology is operated and looks physically like Clos and connects to external devices with standard Ethernet, while its inside operates like a scheduled fabric. Source: DriveNets

Internal Fabric

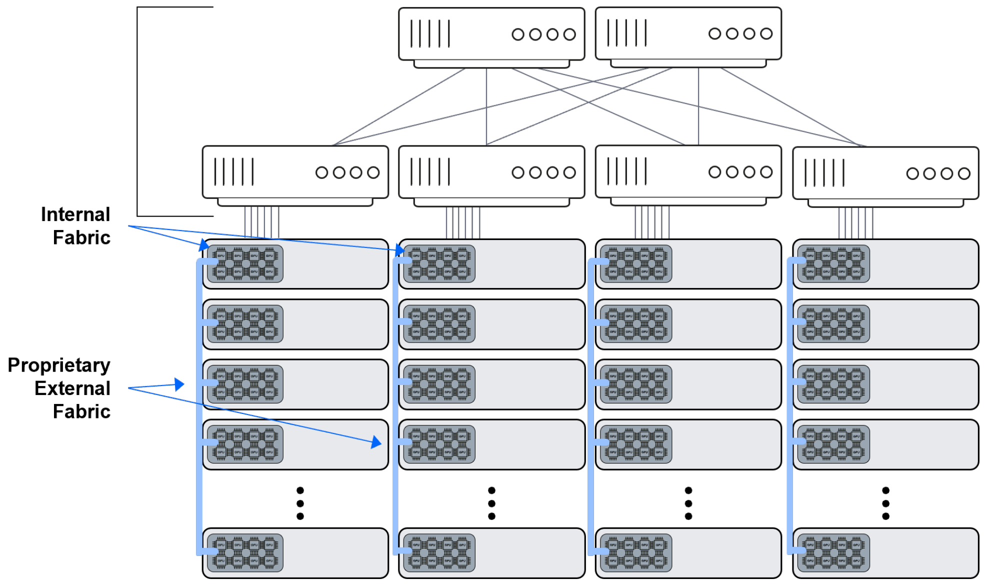
One very common method for connecting GPUs is an internal fabric. It is very common due to the dominance of Nvidia's GPU in the AI market and the fact that the company promotes its NVLink implementation through its DGX/HGX server designs. Equivalent solutions are available from AMD with its Infinity Fabric and from Cerebras with its inherent data plane fabric mechanism that interconnects areas within its wafer solution.

Black Box Functionality

Internal fabric solutions are a bit mysterious. While they do work, it is not very clear how since they are all proprietary and protected IP of the various vendors. They do provide seamless connectivity between the compute engines and high effective throughput between the fabric-connected compute engines. To some extent, the internal fabric also offers some balancing between the compute engines within the server, so the scheduler's task becomes easier and the written application code is easier to implement.

Vendor Lock Concerns

As long as the internal fabric type of the solution stays internal, this is tolerable from the user's perspective. Whether a single compute engine, a pair, an octet or a whole wafer, this is still viewed as a single building block bought "off the shelf" and its proprietary nature is deemed acceptable. Expanding this "unknown" fabric onto a network of servers at a large scale becomes more of a vendor-lock liability (in addition to its advantage of easier job placement). It brings implications to scheduler behavior and application code architecture, and prevents procurement teams from mixing and matching various compute solutions.



Pod of 8 GPUS in a server, connected via proprietary internal fabric. The external backend network is not negated by this existence of internal fabric. Expanding the internal fabric into cross-server proprietary fabric mandates that all GPUs come from the same vendor and that SW be written accordingly. Source: DriveNets

Benefits of a Scheduled Fabric

To recap, the attributes of a network suitable for an AI cluster include:

- High bandwidth
- High CBB
- Predictable and consistent latency throughout multiple workflows with high variance
- Lossless behavior (with very high penalty for outages)
- Connectivity to the internet via common standard interfaces

DDC for AI backend networking fabric was originally built as a standard telco-grade router, with its HW and SW designed to maintain the 99.999% availability criteria often referred to as "carrier grade". A full testing cycle by AT&T prior to entrusting its nationwide core network to this new technology, confirms that anything on the outside of a DDC scheduled fabric is standard since it was implemented into a live network, interfacing to any existing equipment in that network.

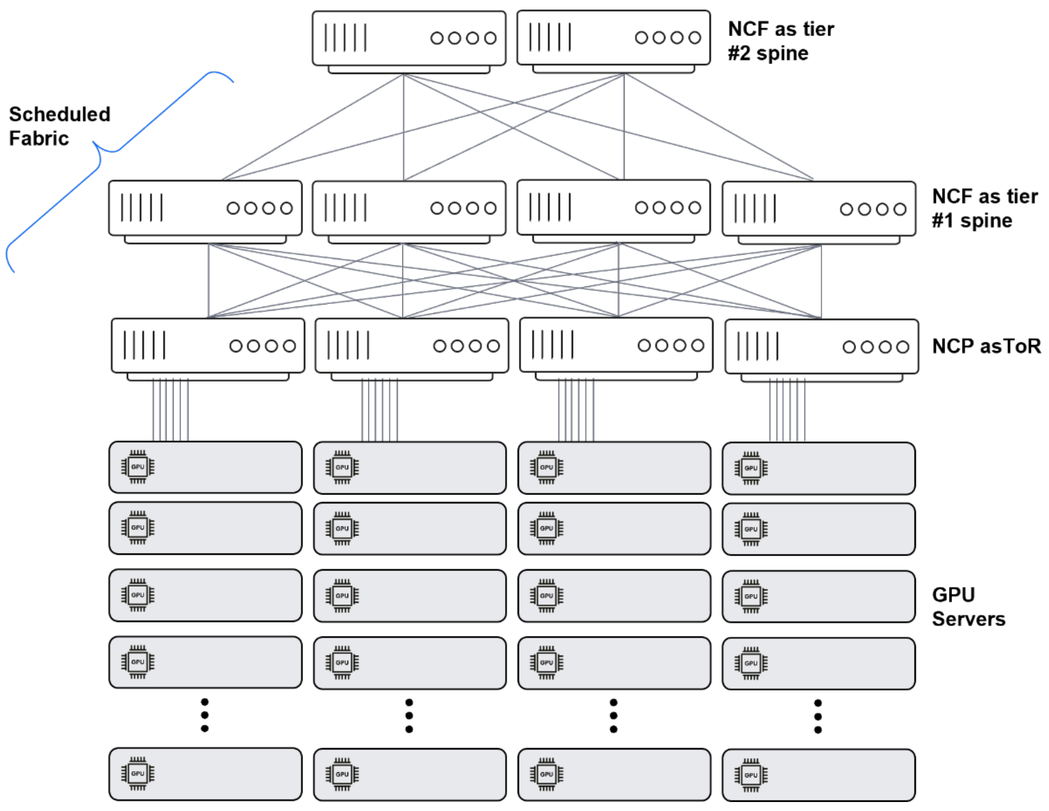
While an AI cluster carries financial penalties (expensive GPUs left idle) for network outages, the penalty for an outage in the telco space is no less significant. Between actual fines paid by operators for lack of service availability and government-mandated emergency services like 911, failures cost telcos dearly. This is what drove the development of the DDC architecture.

With that background for DDC inception, it is clearly not a coincidence to find these AI/HPC network fabric attributes in DDC solutions. Let's look at how DDC's scheduled fabric architecture addresses key challenges.

Maximum Scale

DDC scheduled fabric is implemented with DriveNets' Network Cloud Packet Forwarder (NCP) and DriveNets' Network Cloud Fabric (NCF) built as standalone elements. NCP connects to NCF in a Clos topology, which enables the maximal fanout of the ASIC radix and scale that is impossible in a metal enclosure chassis.

A second level of scale comes from the ability to further connect the NCF as a mid-layer between NCP and 2nd-tier NCF in a Clos-5 topology. This increases the fanout of the scheduled fabric from the radix of a single ASIC (N) to a power of that radix [N^2/2]. Clos topology guarantees flat latency for every network flow.

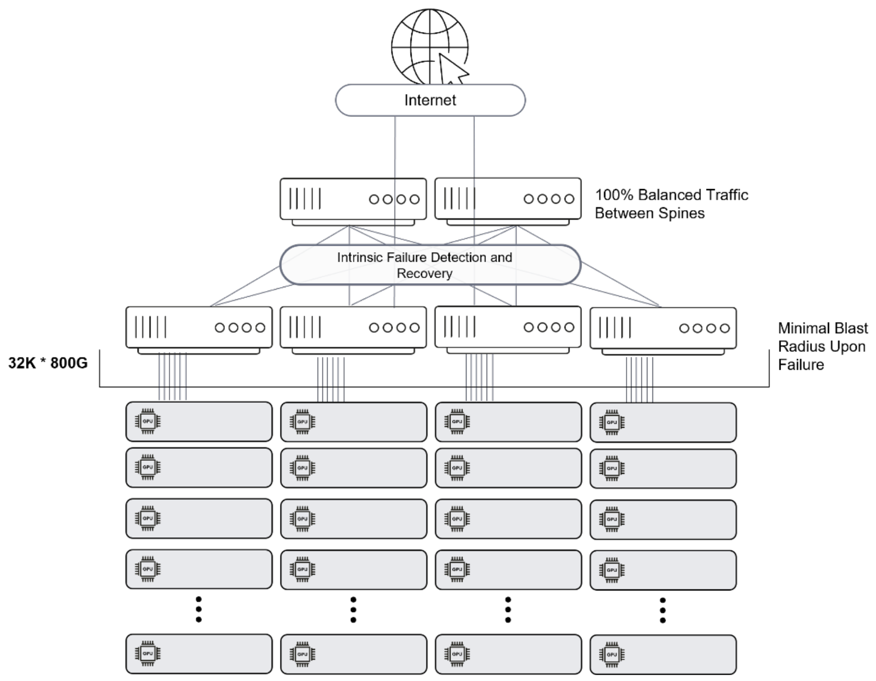


NCP and NCF positioned in a Clos topology building a two-tier scheduled fabric infrastructure

High Bandwidth and Reduced CBB

DDC scheduled fabric was originally designed with Broadcom's Jericho2 ASIC, carrying 100G and 400G interfaces and connected via a 400G fabric operated by the Ramon ASIC. The next generation of Jericho introduces 100G SerDes and 800G interfaces and includes backwards compatibility so it can be added to existing AI networking fabric clusters or used to build new ones. This enables connectivity of today's most recent GPU servers and readiness for next-gen servers expected within 2024. With the cadence of NIC and switch speeds aligned, one can assume that the next generation of 1.6T speeds also will be aligned from a timeline and interoperability perspective.

A packet received on an AI networking fabric is looked at to analyze the destination port within the scheduled fabric where this packet should egress. The packet then goes through segmentation into uniform cells and the spraying of these cells through all the fabric devices where they are forwarded to the egress port for reassembly. This cell distribution of every packet guarantees the balanced utilization of all fabric elements. This means that when, on average, the utilization level of the fabric is 90%, the actual utilization of each fabric is also 90%. Any other method of distributing the traffic between different spine devices will suffer from utilization variance between spine devices, which effectively reduces the CBB of the entire network.


Internet
100% Balanced Traffic Between Spines
Intrinsic Failure Detection and Recovery
Minimal Blast Radius Upon Failure
32K * 800G
GPU

AI backend networking topology shows scale of up to 32K GPU connected into a fabric with 99.999% reliability and close to 100% network utilization levels. Source: DriveNets

Fast Failure Recovery

Any network with connected devices carries the ability to detect physical failure and trigger an outcome. The scheduled fabric in this respect is no different, but with a subtle difference in that *any* failure within the scheduled fabric is considered an internal failure. AI scheduled fabric is a well-known topology and operates as such. Any failure is captured by hardware indicators, with the implication of such failure precalculated locally per every element. The impact on the data plane and reactions needed are implemented at nanosecond-level recovery. Moreover, the propagation of the failure indication is done actively, so the entire AI scheduled fabric system becomes aware of the failure and how to minimize the packet loss resulting from it.

This compares favorably with the sampling nature of traditional networks in which the implication of slow failure recovery carries a heavy impact on the measured GPU utilization as well as its effective utilization. As an AI job is run on multiple GPUs, when a network failure happens that job cannot run to its completion. The result is that the job needs to rerun either in its entirety or from the most recent checkpoint where it arrived. This means that although the GPUs running that job were active, the work they ran would need to run again.

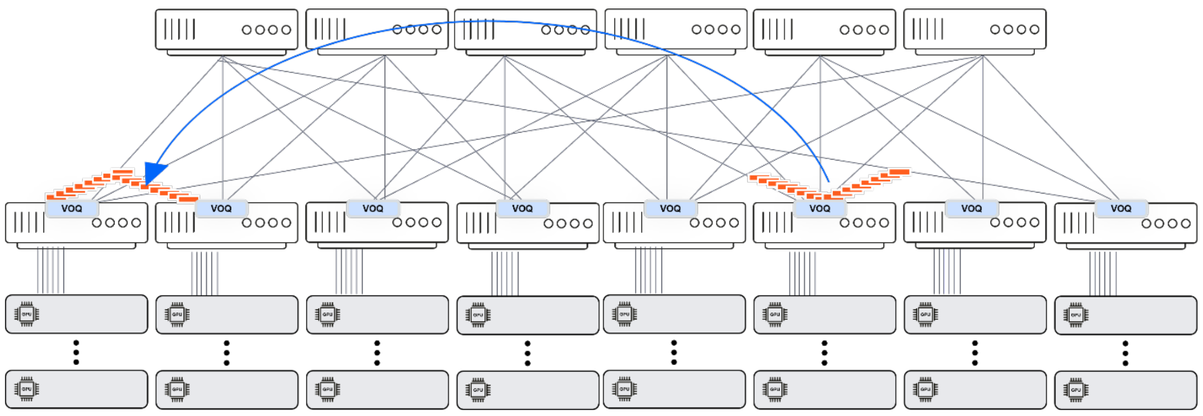
This means that although measured as actual utilization, the effective utilization is impacted. Considering how modern AI clusters can reach sizes of tens of thousands GPUs, the number of elements (>1000) used for interconnecting all these GPUs means that a failure scenario becomes a common routine.

A faster failure recovery at nanosecond level vs. millisecond level aligns the actual and the effective GPU utilization ratios, resulting in shorter job completion time (JCT).

Consistent Latency

The segmentation and reassembly (SAR) layer of the AI scheduled fabric guarantees that all packets go through the same path from source to destination, preventing variance in the latency of a packet. However, because the system has multiple inputs and multiple outputs with an any-to-any traffic pattern, a head-of-line (HOL) blocking scenario can still cause severe damage to both throughput and latency. A HOL scenario indicates a flow that cannot be forwarded (for actual network reasons like congestion or incast on the egress port). While this flow at the head of the forwarding line awaits the availability of "its" egress port, it holds up other flows behind it that otherwise could be forwarded because "their" parts of the network are free. An actual network would suffer dramatic degradation from HOL scenarios, becoming very common as network utilization grows, and impacting packet latency.

The AI scheduled fabric implements a virtual output queue (VOQ) associated with the egress port. Other flows destined to other egress ports do not wait in that same queue and are not held up by the HOL packet, which is waiting for a credit from the egress port to signal it into the SAR layer and into the fabric. The only way for a packet to be delayed in a scheduled fabric is when there is an egress port (directly connected to a GPU) that is being utilized to a higher capacity than that of the GPU to which it is connected. This scenario is not normal, and in fact indicates flawed behavior of the AI workload scheduler.



AI scheduled fabric topology's internal traffic flow relies on credit-based forwarding that triggers packet spread into cells from the VOQ and a reassembly on the receiving end. Any traffic flow is treated similarly. Source: DriveNets

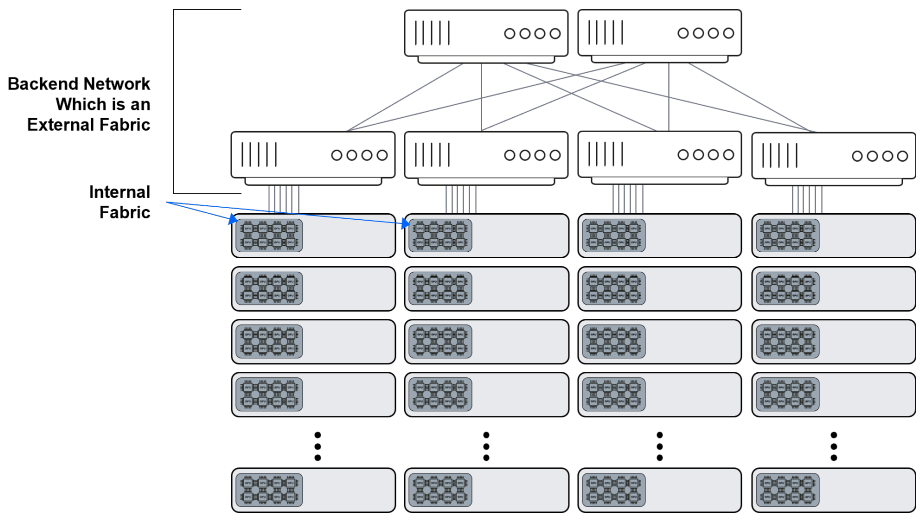
Flow Agnostic

Looking at every specific workload, the network can be educated to pinpoint the needs of the resulting workflow. Yet it is very difficult to do such tuning and requires specific expertise. It is true that the tuned network needs to be one that is capable of such external tweaking of traffic flows; a general example of such a network is SDN. The problem arises when the workflows run on an array of GPUs that are not constant and continuously changing, resulting in changing traffic patterns and workload characteristics.

The allocation of a scattered group of GPUs to run a specific job is a task handled by a job scheduler, which considers the availability and capacity of each GPU. The whereabouts of GPUs and the existing workload of the network that connects these GPUs is unknown to both the GPUs and the scheduler. This results in a scenario where tuning the network for a specific flow is useless and tuning for every flow is impossible.

AI scheduled fabric architecture avoids the selection of the most optimal path for a specific flow; every flow assumes every possible path between ingress and egress points in a well-balanced manner due to the cell distribution of the scheduled fabric. This also solves the huge, and unresolved, challenge of network-aware GPU allocation per jobs.

Another angle for GPU job allocation relates to the previously mentioned internal fabric and the option to extend that fabric outside of the servers. The advantages of a fabric are apparent to the industry. Extending the fabric advantage of seamless GPU allocation and ignoring the physical location of GPUs in the cluster outside of a single server on a full cluster is exactly the benefit of the scheduled fabric. In this case, the extended fabric and the backend network are the same element granting external high BW and reliable connectivity to/from the cluster even to the outside world. This element also provides internal any-to-any GPU connectivity agnostic of physical placement of the GPU.



An AI cluster showing the use of internal fabric (NVLink/Infinity/other) within servers while inter-server connectivity is over a standard scheduled fabric. Source: DriveNets

Lossless Behavior

As described, AI scheduled fabric architecture handles queuing and fragmentation of packets to maximize network utilization and increase cross bisectional bandwidth (CBB).

By definition Ethernet carries lossy network behavior, forwarding packets towards the destination with an assumption that they will pass every network junction and egress on their end destination. In case packets hit any network issue, meaning a problem has occurred, it signals the alarm with the hope that this will be sufficient to avoid packet loss. If the packet is still dropped, Ethernet assumes higher layers will take care of retransmission of that lost packet. This behavior is referred to as "best effort."

AI scheduled fabric, however, implements three mechanisms to avoid packet loss, guaranteeing lossless behavior.

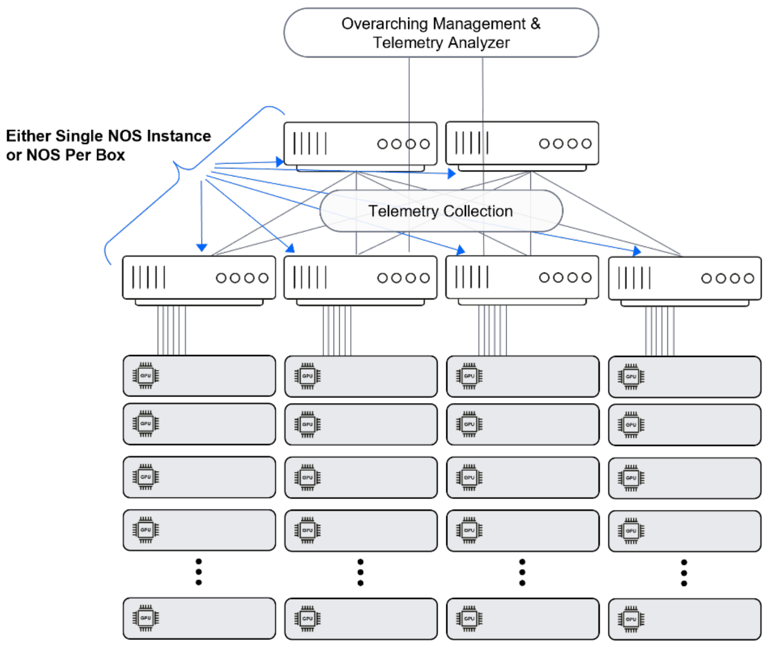
1. The virtual output queue (VOQ), held in the ingress port, prevents HOL blocking as described above
2. The SAR layer maximizes network CBB and in fact prevents these so called "network issues"
3. A credit-based forwarding mechanism. The egress port signals the ingress port with a credit that a packet can be forwarded. This means that there are no packets sent through the fabric that end up in a capacity-lacking scenario. This gives scheduled fabric architecture an advantage of better absorption of microbursts when new jobs are launched or in between jobs, avoiding a scenario where traffic consumes fabric resources and packets eventually end up dropped.

Telemetry Collection

When building a large network for carrying data center traffic, there are many indication points within the network known as telemetry. These indications are collected and are used to trigger conclusions and actions regarding the behavior of the network. The most well-known cases are for measuring latency and/or for calculating the likelihood of packet drops. While these two attributes are inherently resolved in an AI scheduled fabric, the ability to collect telemetry is still there, and it can be used for more profound analysis of network health and for failure prevention.

Managing Different NOS

Large AI arrays connect via a large network. This is a given. Large routing networks are a managerial headache when not using scheduled fabric. From managing protocols and routing tables to managing different network operating system (NOS) types and NOS versions and the supported or non-supported interoperability between them all, these are tasks that are simply avoided when scheduled fabric is used. AI scheduled fabric can run a single centralized NOS instance as a distributed function throughout all scheduled fabric devices or a multitude of NOS instances each operating on a standalone device. While the recommendations and best practice is for a centralized NOS, the option of multiple NOSs is left for users to decide.


Overarching Management & Telemetry Analyzer
Either Single NOS Instance or NOS Per Box
Telemetry Collection

AI scheduled fabric carries telemetry from throughout the cluster into an analyzer. Containerized NOS functions or full NOS per box gives provides operational flexibility. Source: DriveNets

Telemetry and Centralized Management

Two other features of scheduled fabric that are relatively unknown are telemetry and centralized management.

Examples: AI Backend networking Scheduled Fabric in Action

The next chapter will explore some results from actual tests scenarios.

A “noisy neighbor” scenario is apparent when an impairment or other outstanding behavior of a single traffic flow impacts other traffic flows within that same network despite no relationship between the flows.

Mean JCT Time of Multiple Jobs

The following tables show the mean JCT time of multiple jobs in random placement in a 2K GPU cluster.

In the first table, the performance pre-introduction of any impairment shows only a 4% deviation from the modeled theoretical perfect network, which is an outstanding result.

The next table shows some impairment that degrades a link to 70% capacity; job #1 suffers a 24% performance degradation while all other jobs are not impacted at all.

The final table shows a further increase of impairment to a 50% capacity level, which translates to 73% degradation of the directly impacted job; in this case as well there is no impact on other jobs running within that same network. Other tests taken over Clos topology have shown that the 4% deviation from optimum has grown to 5.5% in the 70% scenario and to 6.5% in the 50% scenario.

DNX 2048 Nodes, Random Placement, No Impairments

Configurations	Job	Mean	100th
2k nodes DNX, multiple jobs (8x256), random placement, no impairments	6	1.040	1.041
	2	1.040	1.041
	8	1.040	1.042
	4	1.040	1.041
	7	1.040	1.042
	5	1.040	1.041
	3	1.040	1.042
	1	1.040	1.042

DNX 2048 Nodes, Random Placement, One NIC Impaired by 70%

Configurations	Job	Mean	100th
2k node DNX, 256Nx8, random, RX impairments at 70% of the PCIe peak bandwidth, w/single job impairment	1	1.238	1.240
	7	1.040	1.041
	3	1.040	1.042
	5	1.040	1.041
	8	1.040	1.042
	6	1.040	1.041
	2	1.040	1.042
	4	1.040	1.042

DNX 2048 Nodes, Random Placement, One NIC Impaired by 50%

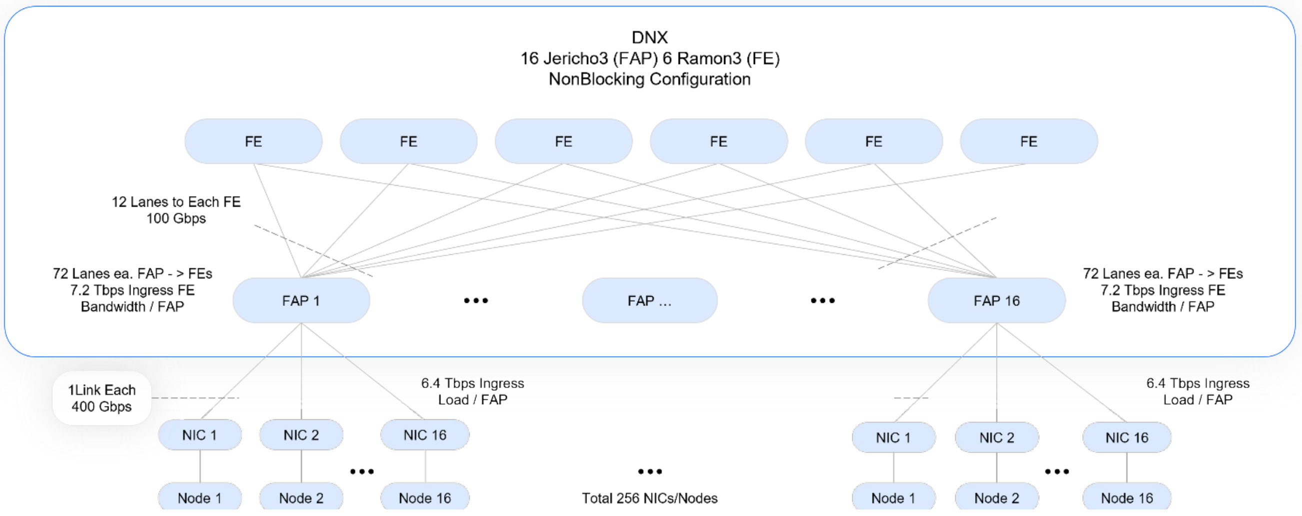
Configurations	Job	Mean	100th
2k node DNX, 256Nx8, random, RX impairments at 50% of the PCIe peak bandwidth, w/single job impairment	1	1.730	1.733
	7	1.040	1.041
	3	1.040	1.042
	5	1.040	1.042
	8	1.040	1.042
	6	1.040	1.042
	2	1.040	1.042
	4	1.040	1.042

Such results emphasize the need to monitor GPU allocation decisions in a steady state and collateral damage impact in failure scenarios.

Another conclusion from these results is the ability of the AI scheduled fabric solution to scale the number of run jobs “linearly.” This compares favorably with a flow-aware network topology where scaling the number of jobs carries an impact on existing jobs running through the network.

Checking the Impact of Failure Scenarios

The next example shows a scenario of 256 GPUs used to check the impact of failure scenarios on an AI scheduled fabric (DDC topology). Note that the DNX architecture is architecturally defined with inherent oversubscription capacity on the links between fabric access processors (FAPs) and fabric elements (FEs).



The measured JCT in a single-spine and a dual-spine failure scenario are shown in the table below. Results show the effective BW degradation is aligned to the percentage of BW lost as a result of the failing FE.

This leads to two conclusions:

- The traffic was evenly balanced over the remaining uplinks after the failure, resulting from the cell-based traffic distribution.
- The minimal delta between the mean results and the 100th result, showing the negligible variance between different measurements.

DNX 256 Nodes Single Job with Impaired Switch/es

Configurations	JCT/BW	Mean	100th
256 node DNX, one FE switch down	1.249	1.095	1.097
256 node DNX, two FE switches down	1.562	1.362	1.365

The AI scheduled fabric solution shows consistent and independent test results in scenarios where flow-based technologies, like ECMP, suffer both cross interference and great variance in the predicted results.

Comparison Table of AI Cluster Solutions

	Proprietary	Ethernet			
	InfiniBand & others	Clos Topology [Naïve Ethernet]	Distorted Ethernet	Single Chassis	Scheduled Fabric (DDC)
Scale	Theoretically possible however yet to be used in high scale	High but not with sufficient performance	Yet to be used in high scale	Limited to a chassis footprint	High
BW & CBB	BW cadence is slow, and CBB is tuned per application	Interface speeds are of latest generation, but CBB is poor	Interface speeds are cutting edge, but CBB at large scale is an unknown	Interface speeds are a step behind, while CBB is close to 100%	Interface speeds are cutting edge, while CBB is close to 100%
Latency	Ultra-low latency and jitter when tuned and preconfigured properly	Low latency without any control on latency variation	Low latency without any control on latency variation	Medium latency with minimal variation	Medium latency with minimal variation
Flexible Flow Handling	Poor	Sub-par regardless of single or multiple flows	Unknown	Agnostic to flows, good although limited in scale	Agnostic to flows
Failure Recovery	Failure reaction by centralized SDN controller is inherently slower	Localized action and slow propagation via protocols	Localized action and slow propagation via protocols	Immediate HW-based	Immediate HW-based
Lossless Behavior	Credit-based lossless protocol	Lossy with attempts to emulate lossless on top of it	Lossy with attempts to emulate lossless via endpoint management	Lossless device	Lossless fabric
Telemetry & Centralized Management	Managed as an island. All proprietary and centralized	Initial use of telemetry for improving flows behavior	More intensive use of non-standard telemetry	No telemetry for inner-chassis stats	Rich use of telemetry as inherent functionality of the Scheduled Fabric (DDC)

AI/HPC Scheduled Fabric Meets the Challenges facing an AI-enabled Data Center Infrastructure

The introduction of AI into data centers poses many challenges on the network layer. The many network alternatives offered for interconnecting AI clusters all have pros and cons, forcing tradeoffs for the end user.

Utilizing DDC for AI Backend Networking Fabric appears to meet all the challenges of the AI-enabled data center infrastructure despite, and perhaps because of, being primarily designed for a completely different networking use case. The one main architectural variance between AI scheduled fabric and all other proposed network solutions alternatives is that scheduled fabric is not a network, but a network element.

When brought into the data center space, scheduled fabric scales up to be a very large network element. It is also still one logical element due to its management and control software. Outside the AI backend networking are network interfaces, while inside is a fabric. The fabric behavior is exactly what is required to meet the needs of a compute AI server array.

AI scheduled fabric guarantees lossless connectivity for a large-scale server array running high-bandwidth workloads free of flow discrimination and with minimal impact from network failures, all in a traditional and easy-to-manage Clos-like topology.