Functional Architecture for Optically-Accelerated AI Interconnect

Classification: APPROVED Reference Document
Confidentiality: PUBLIC
Version: 1
[AI Interconnect FA]
March 2026

Legal

Contents

List of Figures

List of Tables

1. Introduction

1.1. Background

Recently, the demand for on-demand access to high-performance computing resources, such as GPUs and TPUs (collectively referred to as XPUs), has been rapidly increasing, driven by the growing computational requirements of large-scale AI applications [1-2]. As AI applications continue to grow in complexity, large-scale distributed computation across thousands or even tens of thousands of XPUs has become indispensable [3-4]. Consequently, large-scale computing clusters (referred to as AI clusters) require a scalable, cost-effective, and energy-efficient interconnects (referred to as AI Interconnects) for inter-XPU collective communication [5].

However, conventional AI Interconnects, comprising both Scale-Up (SU) and Scale-Out (SO) Interconnects, face several challenges. For Scale-Up Interconnects, implementations rely on electrical interconnect technologies and are therefore confined to a single rack or Universal Baseboard (UBB). This limitation restricts the number of supported XPUs within a single SU network domain [6–7]. On one hand, Scale-Out Interconnects are built on electrical packet switches (EPSs) with conventional optical pluggable transceivers and so suffer from limited scalability and high bandwidth cost [8-9]. Furthermore, in both Scale-Up and Scale-Out Interconnects, efficient multi-tenant support, which is an essential requirement for XPU-as-a-Service (XPUaaS) platforms [10], is constrained by limited per-XPU radix and vendor-proprietary interconnect technologies. These limitations restrict flexible per-tenant XPU allocation and isolation, and raise security concerns when resources including memory are shared across tenants.

1.2. Purpose

The purpose of this document is to guide AI cluster implementers in building scalable optically-accelerated AI Interconnects based on its functional architecture, enabling extended communication reach and XPU scaling beyond the limits of conventional AI Interconnects.

Another purpose is to bring the following benefits to suppliers (system component vendors):

- Enhanced Market Opportunity:
    - Provision of clear user requirements for systems and components to enable targeted product development.
    - Expanded use cases for these products, opening up new market segments.
- Fostering a Healthy Ecosystem:
    - Encouraging healthy competition, which stimulates innovation and investment.

1.3. Objective

The objective of this document is to define a functional architecture for optically-accelerated AI Interconnects that enable scalable inter-XPU communication through optical connectivity and overcome the limitations of conventional AI Interconnects.

1.4. Scope

The scope of this document is as follows:

- Definition of the overall functional architecture of the optically-accelerated AI Interconnects, including functional components and their roles.
- Illustration of implementation options for each functional component.

Note that this document focuses on the design of optically-accelerated AI Interconnects within data centers (DCs) while another project [11], Remote GPU over APN, aims at connecting users and green DCs for reductions in training time, power consumption, and cost for generating Large Language Models (LLM), regardless of the structure of AI clusters.

2. Use Cases

This section describes the AI Interconnect and the use cases it should support, which inform its functional architecture. In this document, AI parallelization strategies used in major AI workloads, such as training and inference, are treated as primary use cases (See Annex A for more details).

An AI Interconnect is a subsystem of an AI cluster that interconnects XPUs for inter-XPU communication. It typically encompasses XPUs, switches, and cabling infrastructure, and should be designed to deliver high-bandwidth, low-latency communication to efficiently support collective communications while meeting requirements of scalability, cost efficiency, and reliability.

Table 1 summarizes the key characteristics of inter-XPU communication across major AI parallelization strategies. It shows that communication patterns vary in terms of operation type, message sizes, and operation frequency, ranging from frequent, latency-sensitive small messages to infrequent, bandwidth-intensive data transfers. This diversity in AI workloads highlights the need for an AI Interconnect that allows a single cluster to efficiently support various communication patterns [5-6].

Table 1: Characteristics of inter-XPU communication patterns in AI parallelization strategies.

Category	Operation type	Message Size Distribution	Operation Frequency
TP (Tensor Parallelism)	Collective (All-Reduce, Reduce-Scatter, All-Gather)	Large (MB–hundreds of MB, relatively uniform)	High (per layer per training iteration; sometimes per token in inference)
PP (Pipeline Parallelism)	Point-to-Point (activation / gradient transfer between stages)	Medium–Large (several MB–tens of MB)	Medium–High (per micro-batch)
DP (Data Parallelism)	Collective (All-Reduce for gradient synchronization)	Very Large (tens of MB–GBs)	Low–Medium (once or a few times per training iteration)
EP (Expert Parallelism)	All-to-All (token dispatch / gather)	Small–Medium (KB–several MB, highly variable)	Very High (per layer / per token, bursty)
P-D (Prefill–Decode)	Point-to-Point (state / intermediate representation transfer)	Small–Medium (KB–several MB)	High for the gradual KV transfer from P to D.

To support these diverse communication patterns, AI clusters typically adopt an interconnect comprising Scale-Up Interconnect and Scale-Out Interconnect as introduced in Section 1 [6]. As AI models grow in size and complexity, including LLMs and multi-modal AI, both training and inference rely on large numbers of XPUs [12-13]. A Scale-Up Interconnect must therefore support hundreds of tightly coupled XPUs with extremely high per-XPU bandwidth, high-radix connectivity, and ultra-low latency to enable frequent collective, all-to-all, and memory-semantic operations [14-16]. At the same time, achieving sufficient scalability requires extending computation beyond a single SU domain*. Consequently, the Scale-Out Interconnect must support interconnecting tens of thousands of XPUs by providing sufficient aggregate and per-XPU bandwidth, flexible topology support to accommodate diverse communication patterns [17-18], and high system-level reliability to sustain long-running distributed workloads [19-20].

In some deployments such as multi-tenant operation (Fig. 1), the AI Interconnect may need to be logically and/or physically partitioned into multiple XPU pools, while sharing the same underlying interconnect infrastructure across multiple tenants.

To effectively support AI parallelization use cases discussed above, this document identifies the need for an optically-accelerated AI Interconnect. The subsequent sections provide a detailed discussion of the motivations and technical justifications for the optically-accelerated AI Interconnect.

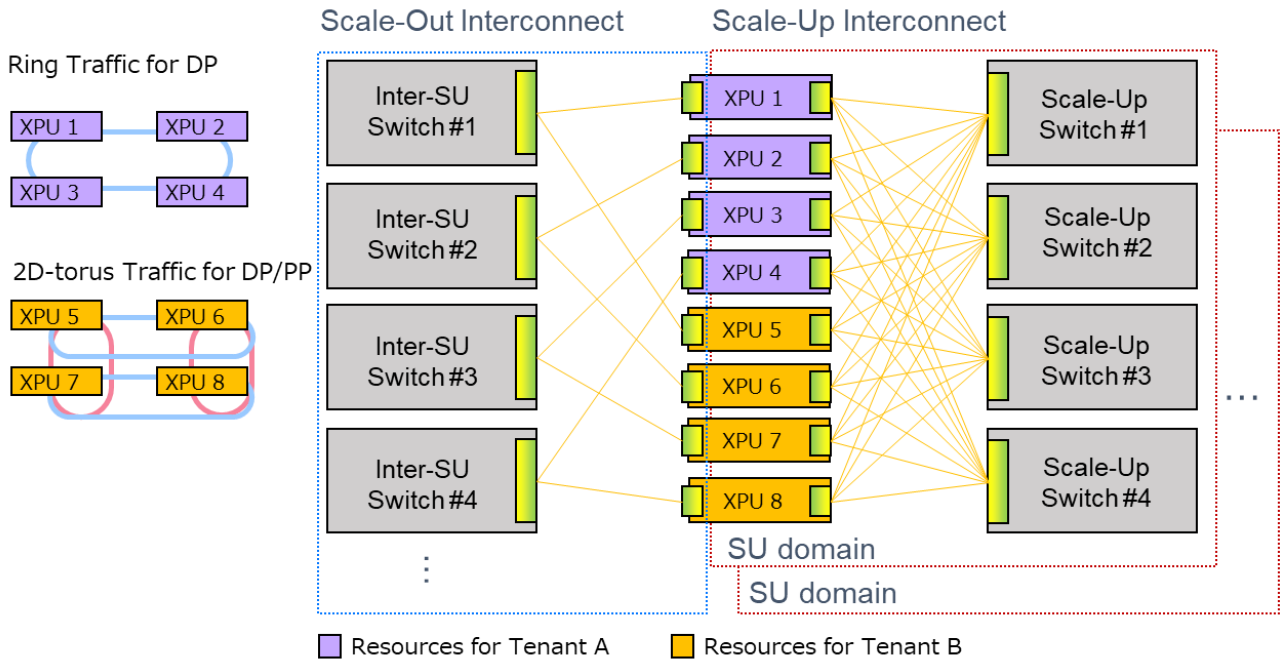

Figure 1: AI Interconnects comprising Scale-Up and Scale-Out Interconnects. Network resources (switches and links) are shared in multi-tenant use case.

* This document defines a SU domain as a set of XPUs tightly coupled through ultra-low-latency, high-bandwidth interconnects for tightly synchronized collective communications and memory-semantic operations. In contrast, the Scale-Up Interconnect refers to the interconnect service used to construct the SU domain.

3. Services

This section defines optically-accelerated AI Interconnect Services. To well serve the use cases in Section 2, this document adopts a two-layer interconnect architecture consisting of an optically-accelerated Scale-Up and a Scale-Out Interconnect Services. The optical Scale-Up Interconnect provides high-bandwidth, low-latency communication among XPUs within an SU domain, while the optical Scale-Out Interconnect extends connectivity across multiple SU domains to enable system-level scalability.

3.1. Scale-Up Interconnect Service and Its Requirements

The Scale-Up Interconnect Service shall provide high-bandwidth, low-latency communication among tightly coupled XPUs within a single SU domain. It is intended to support fine-grained, latency-sensitive parallelisms (e.g., TP and EP) as well as memory-semantic operations that are essential for efficient distributed computation [5, 16].

Based on the use case described in Section 2, this document identifies the following requirements for Scale-Up Interconnect Service.

- **(Requirement #1-1) The number of XPUs**
  - Hundreds of XPUs must be installed within a SU domain for sufficient scalability.
- **(Requirement #1-2) Bandwidth**
  - Per-XPU bandwidth must be at least several Tbps (ideally approaching 10 Tbps or higher).
- **(Requirement #1-3) Network structure**
  - The physical network of a SU domain must provide high-radix all-to-all connectivity among XPUs.
- **(Requirement #1-4) Latency**
  - The one-way latency among any XPU pair in a SU domain must be less than 1 μs [14-16].

3.2. Scale-Out Interconnect Service and Its Requirements

The Scale-Out Interconnect Service shall provide scalable inter-XPU communication across multiple SU domains for large-scale distributed training and inference. It is intended to support computation-intensive and topology-aware communication patterns (e.g., DP and PP) over a very large number of XPUs, while maintaining system-level reliability for long-running workloads.

Based on the use cases described in Section 2, this document identifies the following requirements for Scale-Out Interconnect Service.

- **(Requirement #2-1) The number of XPUs**
  - At least 10K XPUs must be accommodated for sufficient scalability and advanced large-scale AI applications.
- **(Requirement #2-2) Bandwidth**
  - Per-XPU bandwidth must be hundreds of Gbps to a few Tbps to enable efficient communication across any XPU.
- **(Requirement #2-3) Network structure**
  - The physical topology must be designed to efficiently support the realization of multi-dimensional parallelisms.

3.3. Non-Functional Requirements

Based on the use cases in Section 2, this document identifies the following non-functional requirements for AI Interconnect Services.

- AI cluster user perspective
  - **High inter-XPU network performance**: High performance of the inter-XPU network is essential for efficient distributed computations.
  - **Reliability**: The interconnect services shall support high availability for AI workloads through redundancy mechanisms, enabling continued operation under recoverable component-level failures that do not compromise application state.
- AI cluster operator perspective
  - **Scalability**: The AI cluster architecture shall support scalable expansion within SU domains and across SO domains, consistent with the quantitative XPU requirements defined in Requirement #1-1 and Requirement #2-1.
  - **Efficient and robust resource allocation and topology optimization for workload**: XPU resource allocation, together with the underlying physical or logical network topology, must be optimized on a workload or per-tenant basis to maximize resource utilization, prevent workload fragmentation, and enable fault-tolerant operation.

Note that the requirements related to tolerable XPU unreachable time due to topology reconfiguration during workload execution are not considered here, as topology reconfiguration is assumed to occur at the time of XPU allocation for workloads or tenants, prior to the initiation of workload execution. XPU unreachability caused by network hardware failures is considered under reliability aspects; however, quantitative requirements on tolerable unreachable time are not specified in this document, as they depend on the XPU implementation or the service-level agreement defined by each AI cluster owner.

4. Gap Analysis and Design Concepts

4.1. Gap Analysis

This subsection identifies the limitations of existing Scale-Up and Scale-Out Interconnects. They can interconnect only a limited number of XPUs for the following reasons:

- For the Scale-Up Interconnect, the number of deployable XPUs is limited because they are typically confined in a rack or Universal Baseboard (UBB) with specialized electric-wires [6-7, 16]. This limitation arises from the substantial propagation loss experienced by broadband signals traversing electrical interconnects, which restricts the cable length and consequently imposes a severe limit on the number of XPUs deployable in a single SU domain [8]. The constraint on the SU domain size tends to lower XPU utilization. For example, if one AI model replica requires 16 XPUs, only four replicas can be deployed within a 72-XPU SU domain, leaving eight XPUs unused. This issue will become more pronounced as AI model sizes are continuing to grow.
- For the Scale-Out Interconnect constructed using conventional EPSs, the number of deployable XPUs is constrained by the switching capacity and radix limitations [9, 22]. These are fundamentally constrained by semiconductor chip size and Input/Output (I/O) density, including Serializer/De-serializer (SerDes) area, package pin count, and signal integrity [23]. Although higher per-lane electrical data rates are used to increase capacity and radix, severe attenuation and noise at such speeds significantly reduce link reach, making it difficult to realize the high-capacity, high-radix EPSs required for large-scale networks [8].

In addition, in existing scale-up and scale-out networks based on conventional pluggable transceivers for XPUs and L2 Switches, achieving extremely high aggregate bandwidth and multi-dimensional torus topologies is challenging. The limited port density and power efficiency of pluggable transceivers constrain scalable multi-dimensional or multi-rail designs for AI workloads, making efficient large-scale realization difficult.

4.2. Design Concepts

This document proposes an optically-accelerated AI Interconnect that addresses the limitations of existing AI Interconnects described in Subsection 4.1.

4.2.1. Scale-Up Interconnect

This document proposes an optically-accelerated Scale-Up Interconnect by replacing conventional electrical interconnects, which are typically vendor-proprietary links used for communication among densely placed XPUs, with optical interconnects, as well as enabling optical I/O at both XPUs and switches (e.g., via co-packaged optics (CPO) and optical circuit

switches (OCSs)) (Fig. 2). This design concept aims to realize Scale-Up Interconnects that satisfy the requirements derived in Subsection 3.1 as follows.

Firstly, leveraging the extended reach and high bandwidth and port density of optical interconnects enables a floor-scale Scale-Up Interconnect applicable to both general-purpose packet-switched fabrics (e.g., Ethernet) and emerging accelerator interconnects (e.g., UALink). This allows hundreds of XPUs to be accommodated within a single SU domain, satisfying **Requirement #1-1**.

Second, this design enables per-XPU bandwidth of at least several Tbps by aggregating multiple high-bandwidth channels provided by optical I/O, thereby satisfying **Requirement #1-2.** The required bandwidth is achieved by appropriately determining the number of integrated optical engines (OEs) and channels in XPUs or L2 Switches, together with the corresponding SerDes lane counts and per-lane data rates.

Third, a physical network configuration optimized for all-to-all operations should be realized to support latency-sensitive, communication-intensive parallelisms such as TP and EP. To this end, every XPU must be physically connected in a full connectivity manner through a SU domain. Such full connectivity among XPUs is enabled by dense optical I/O made possible by the multi-fiber connectivity of advanced optical technologies such as CPO, thereby satisfying **Requirement #1-3**.

Finally, one-way latency of less than 1 μs among any XPU pair (**Requirement #1-4**) can be achieved by carefully determining the length of each cable and Forward Error Correction (FEC) configuration [24] for transceivers (TRx).

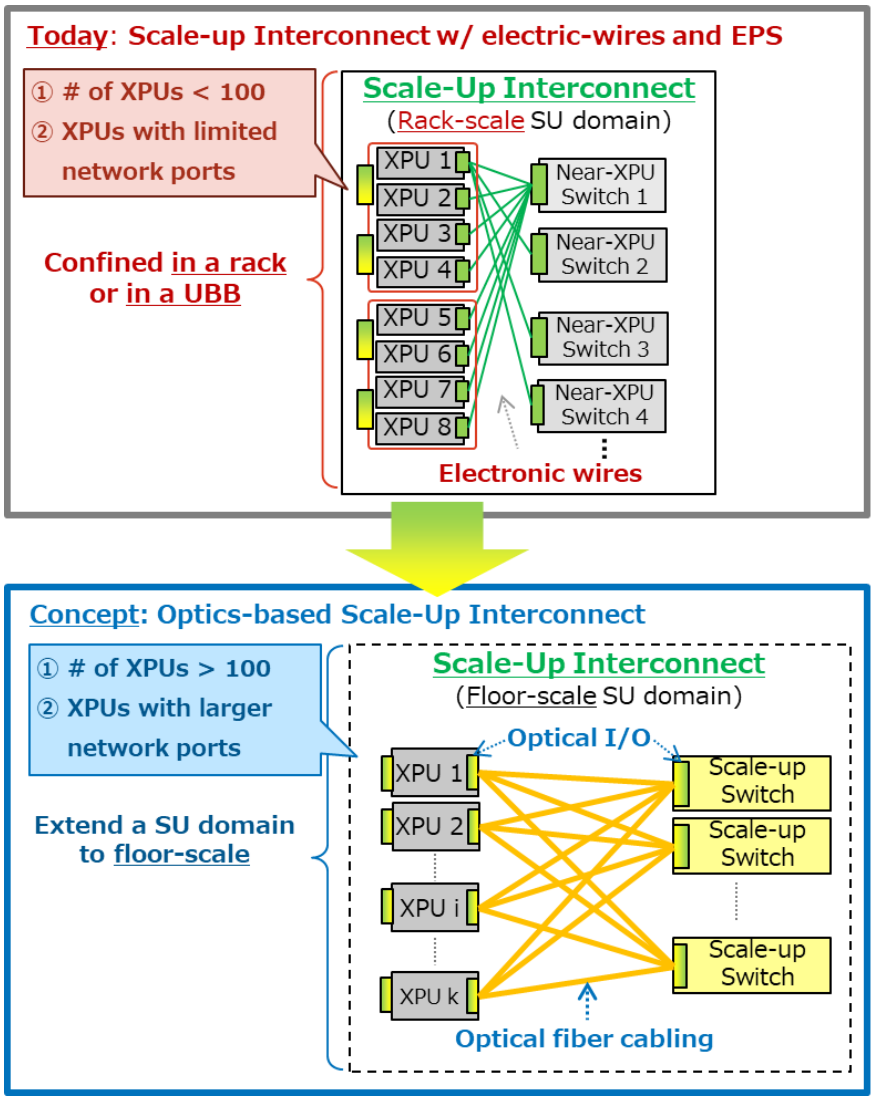

Figure 2: Design Concept #1: Optical Scale-Up Interconnect.

4.2.2. Scale-Out Interconnect

Similar to Subsection 4.2.1, this document proposes an optically-accelerated Scale-Out Interconnect (Fig. 3) to address the high CAPEX/OPEX inherent in existing scale-out networks. By integrating advanced optical technologies such as CPO, the per-XPU radix is increased through higher bandwidth and port density, enabling AI workload–optimized interconnect topologies. In addition, introducing OCSs removes the need for optical pluggable transceivers in the switching fabric, reducing hardware costs, while allowing OCS reuse across multiple TRx generations as long as physical-layer parameters remain unchanged [22]. This design concept aims to realize Scale-Out Interconnects that satisfy the requirements derived in Subsection 3.2 as follows.

First, this design concept allows to accommodate at least 10K XPUs. OCS-based optical interconnects inherently could support virtually unbounded scalability, while high-radix L2 Switches with optical I/O can also satisfy this requirement. For example, a rail-only topology composed of 128 L2 Switches, each with a radix of 256, can interconnect more than 30K XPUs [15]. Thus, this approach satisfies **Requirement #2-1**.

Second, this design concept aims to provide per-XPU bandwidth of hundreds of Gbps to a few Tbps. As in the Scale-Up Interconnects (Subsection 4.2.1), optical I/O enables high aggregate bandwidth in Scale-Out Interconnects by appropriately selecting the number of OEs per XPU or L2 Switch, along with their SerDes lane counts and per-lane data rates. This approach readily satisfies **Requirement #2-2**.

Finally, this design concept aims to support scalable XPU interconnection topologies optimized for distributed computation. Such topologies can be flexibly realized by high-radix XPUs equipped with high-bandwidth, high-port-density optical I/O (e.g., [20], [21]), thereby satisfying **Requirement #2-3**.

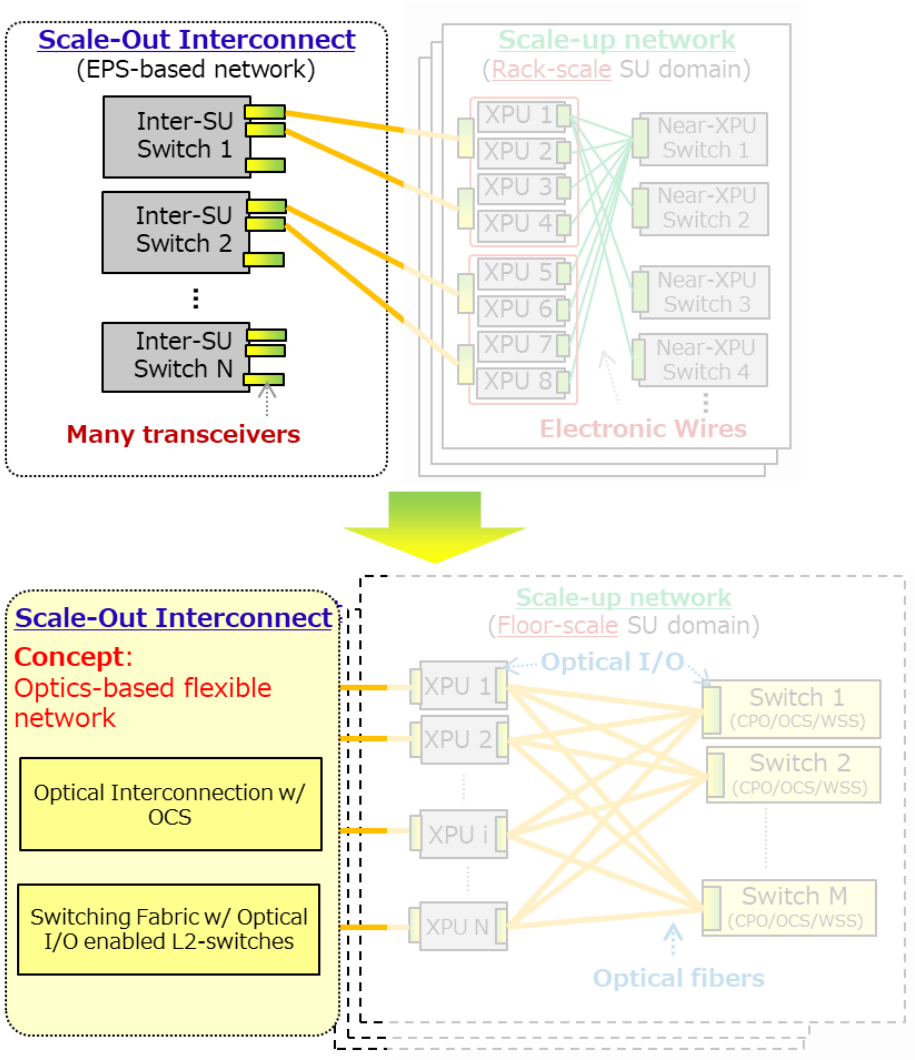

Figure 3: Design Concept #2. Optical Scale-Out Interconnect.

5. Functional Architecture

This section details the functional architecture of an AI cluster with optically-accelerated Scale-Up and Scale-Out Interconnects.

5.1. Overall Functional Architecture

Figs. 4 and 5 show functional architectures of optically-accelerated AI clusters whose system parameters are summarized in Table 2. It comprises the following XPU components and functional blocks, and the corresponding implementation options are summarized in Table 3.

- XPUs
- Optical Cabling Infrastructure (OCI) for Scale-Up and Scale-Out Interconnects
- Dynamic Switching Fabric (DSF) for Scale-Up and Scale-Out Interconnects

This document does not impose any constraints on the placement or arrangement of DSF and OCI blocks. Implementers may determine the arrangement based on physical layout, scalability, and deployment requirements. For example, one implementer may adopt an architecture in which OCI blocks are located between the XPU and DSF blocks, while another may adopt an AI cluster architecture in which the SO-OCI accommodates the SO-DSF.

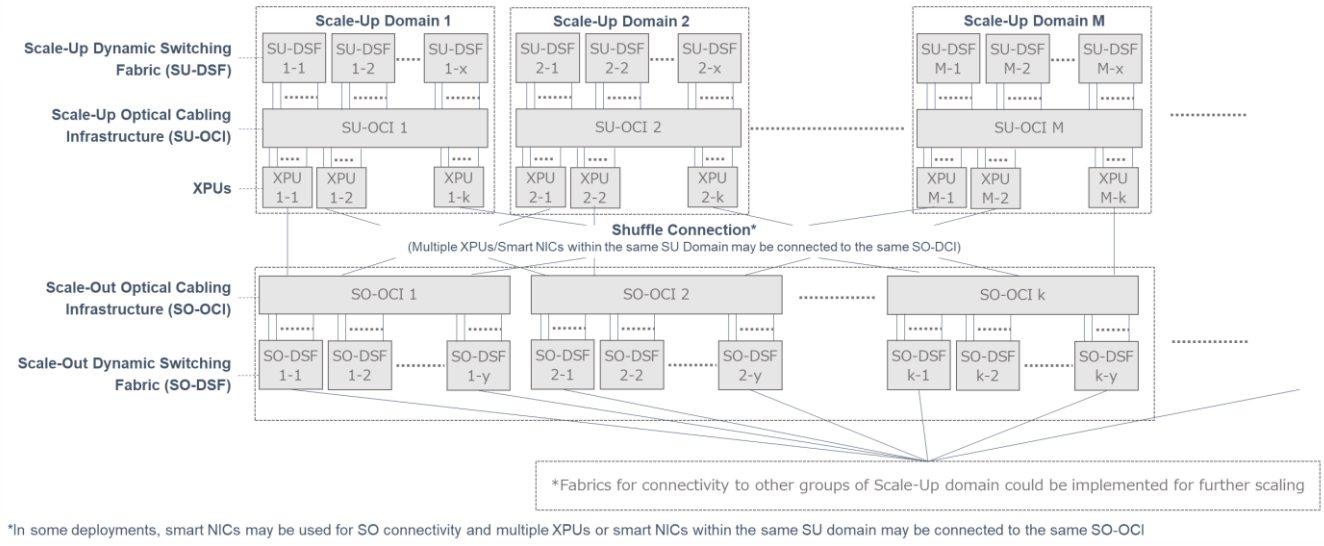

Figure 4: Functional architecture illustrating OCI blocks located between XPU and DSF blocks (Example 1).

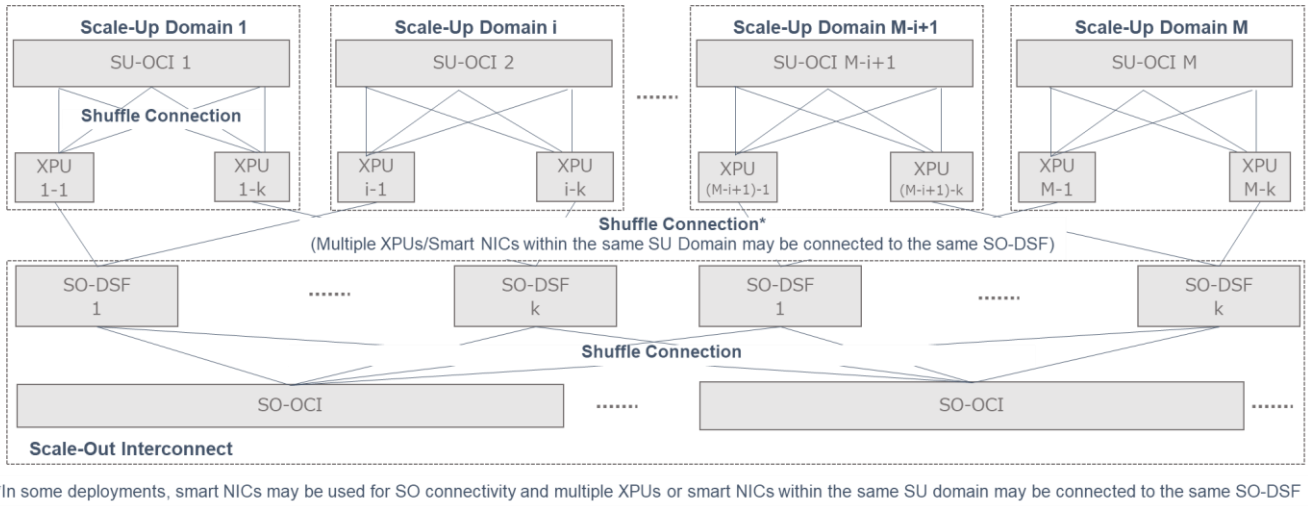

Figure 5: Functional architecture illustrating SO-DSF blocks located between XPU and SO-OCI (Example 2).

Table 2: System parameter list for AI clusters.

System Parameters	Notations
Number of SU domains	M
Number of XPUs in a SU domain	k
Number of ports allocated to each XPU for connectivity to the Scale-Up Interconnect	$P_{xpu\text{-}up}$
Number of channels allocated to each XPU for connectivity to the Scale-Up Interconnect	ℓ_{up}
Number of ports allocated to each XPU for connectivity to the Scale-Out Interconnect	$P_{xpu\text{-}out}$
Number of channels allocated to each XPU for connectivity to the Scale-Out Interconnect	ℓ_{out}
Number of SU-DSFs	x
Number of SO-DSFs	y
Number of network ports on switches	P_{sw}
Cable length in SU domain	L_{up}
Cable length in scale-out domain	L_{out}

Table 3: Components, functional blocks, and implementation options

Components/ functional blocks	Implementation/ component options
• XPUs	• XPU with optical transceivers such as CPO, NPO, LPO and other optical transceiver technologies. • XPU and smart NIC on the same PCIe/CXL interconnect **Note:** Different implementation options may be selected depending for scale-up and scale-out connectivity, respectively.
• SU-OCI • SO-OCI	• Static optical interconnection • Dynamic OCS-based optical interconnection **Note**: Different implementation options may be selected for scale-up and scale-out connectivity, respectively. **Note**: Static SU-OCI typically assumes a fully connected topology; OCS-based implementations may support arbitrary configurations, with switching speed potentially imposing more stringent requirements. **Note:** Static SO-OCI typically assumes structured topologies (e.g., torus or rail); OCS-based implementations may support arbitrary configurations, with switching speed potentially imposing less stringent requirements than those in SU domains.
• SU-DSF • SO-DSF	• Packet switches, including L2 Switches, equipped with optical transceivers such as CPO, NPO, LPO, and other optical transceiver technologies. • NULL (no DSF) **Note:** Different options may be selected for scale-up and scale-out connectivity, respectively

XPUs are the primary components of an AI cluster. This document assumes that the AI cluster consists of M SU domains, each with k XPUs (i.e., k × M XPUs are installed in a SU domain). Each XPU is equipped with Pxpu-up ports and ℓup channels for the Scale-Up Interconnect, and Pxpu-out ports and ℓout channels for Scale-Out Interconnect. In this document, the terms “Scale-Up ports or channels” and “Scale-Out ports or channels” refer to the network ports or channels used to connect to the Scale-Up and Scale-Out Interconnects, respectively. While Figs. 4 and 5 assume

that XPUs provide scale-out ports, this architecture does not preclude deployments where smart Network Interface Cards (NICs) are used to provide scale-out connectivity instead of XPU Ethernet ports. This option enables early deployment of the architecture and is not expected to result in a significant performance disadvantage for scale-out communication.

OCI, is an optical cabling infrastructure that physically and optically interconnects XPUs or switches. One approach for OCI is a static optical interconnect, in which optical interconnection is statically configured (e.g., via manual cabling). Another approach is optical interconnection with OCSs, which enables reconfigurability. In this document, Scale-Up Optical Cabling Infrastructure (SU-OCI) refers to the OCI for the Scale-Up Interconnect. In contrast, Scale-Out Optical Cabling Infrastructure (SO-OCI) refers to the OCI for the Scale-Out Interconnect. The cable lengths in SU-OCI and SO-OCI are denoted as Lup and Lout, respectively. While Figs. 4 and 5 illustrate a case where the XPUs of one SU domain are connected to different SO-OCIs, this functional architecture does not preclude deployments in which multiple XPUs within a single SU domain are connected to the same SO-OCI or SO-DSF. This may enhance fault tolerance.

DSF, is a switching block that performs signal switching in Scale-Up and Scale-Out Interconnects using L2 Switches. In this document, Scale-Up Dynamic Switching Fabric (SU-DSF) refers to the DSF for the Scale-Up Interconnect, while Scale-out Dynamic Switching Fabric (SO-DSF) refers to the DSF for the Scale-Out Interconnect. The number of SU-DSF and SO-DSF is denoted as x and y, respectively. Note that the implementation of DSF is not mandatory; for example, a scalable Scale-Out Interconnect can be realized solely with SO-OCI.

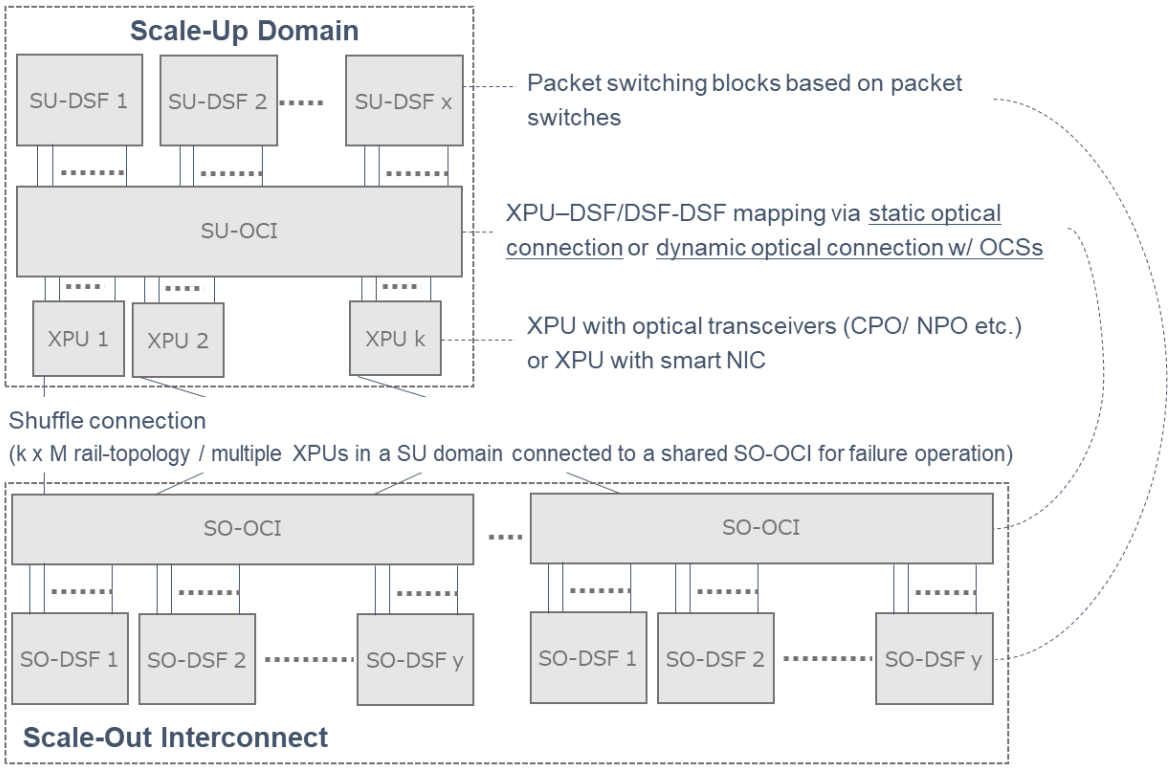

Figure 6: Example of implementation options for each functional block.

5.2. Scale-Up Interconnect

This subsection describes two representative variants for implementing the Scale-Up Interconnect, depending on required flexibility and performance:

1. **Static OCI with DSF**: Optical interconnection is statically configured and combined with a DSF to enable signal routing within the SU domain.
2. **Dynamic OCS-based OCI for XPU-DSF mapping**: Dynamic optical interconnection using OCSs for XPU-DSF mapping, enabling partitioning XPU resource pools for some use cases (e.g., multi-tenancy).
3. **Dynamic OCS-based OCI for DSF-DSF mapping**: Dynamic optical interconnection using OCSs for DSF-DSF mapping, enabling physical topology reconfiguration among DSF.

5.3. Scale-Out Interconnect

This subsection describes representative variants for implementing the Scale-Out Interconnect:

1. **Static OCI with DSF**: Statically configured optical interconnection (e.g., by manual or patch panel based interconnection) paired with DSF for routing across SU domains.
2. **Dynamic OCS-based OCI for XPU-DSF mapping**: Dynamic optical interconnection using OCSs for XPU-DSF mapping, enabling partitioning XPU resource pools for some use cases (e.g., multi-tenancy).
3. **Dynamic OCS-based OCI for DSF-DSF mapping**: Dynamic optical interconnection using OCSs for DSF-DSF mapping, enabling physical topology reconfiguration among DSF.
4. **Dynamic OCS-based OCI without DSF**: DSF-free implementation where OCS-based optical interconnection alone provides scalable, flexible connectivity across tens of thousands of XPUs, reducing hardware complexity and cost.

These variations illustrate how the AI Interconnect can be adapted to different requirements, from high-performance SU domains to flexible, large-scale scale-out deployments.

[Annexes]

A. Use Case Analysis in Detail

This Annex describes representative use cases executed on AI clusters in more detail.

A.1. Parallelization Strategies for AI use cases

The lifecycle of an AI application comprises the following three phases (Fig. 7):

1. Training phase
2. Fine-tuning phase
3. Inferencing phase

Where distributed computation, employing parallelisms such as DP, PP, TP, and EP, are often executed in each phase (Fig. 8). A brief summary of each parallelism is provided below.

- DP replicates the full model across XPUs and aggregates gradients using ring all-reduce to synchronize updates without model partitioning. Consequently, communication is dominated by ring-based all-reduce operations.
- PP partitions the model into sequential stages across XPUs, where activations and gradients are transferred between adjacent devices using point-to-point Send/Recv operations rather than collective primitives. Consequently, communication is dominated by these point-to-point transfers between adjacent XPUs.
- TP splits individual matrix operations across XPUs and requires frequent synchronization via reduce-scatter, all-gather, or all-to-all operations to exchange intermediate tensors for attention and Multi-Layer Perceptron partitioning. Consequently, communication is dominated by these collective synchronization operations.
- EP distributes Mixture-of-Experts (MoE) experts across XPUs and performs token dispatch and combination using all-to-all operations. Consequently, communication is dominated by all-to-all traffic exchanges among XPUs.

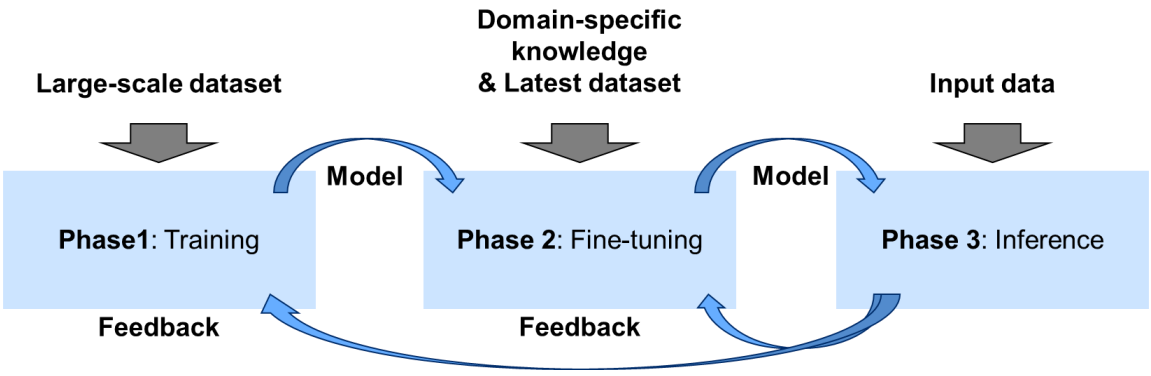

Figure 7: AI lifecycle.

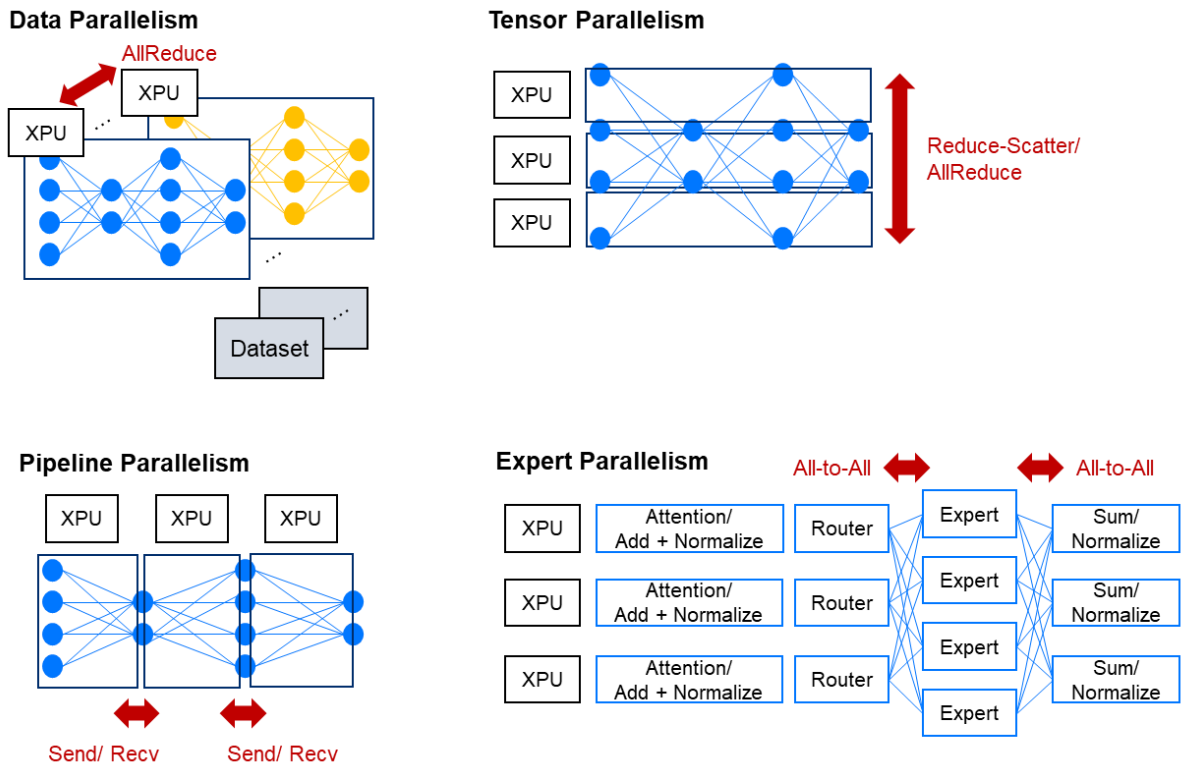

Figure 8: Representative parallelisms performed in distributed computation.

In addition, the Sequence Parallelism (SP) and Context Parallelism (CP) are increasingly used in LLMs, driven by these architectural characteristics and scalability requirements.

- SP partitions the input sequence across XPUs while replicating the model. During attention and normalization, partial results are exchanged to compute global statistics and complete attention outputs. Consequently, communication is dominated by sequence-wise collectives such as all-gather and reduce-scatter.
- CP distributes long-context workloads across XPUs to enable processing beyond the memory capacity of a single device. During attention, KV states or intermediate activations are exchanged to maintain global context consistency. Consequently, communication is dominated by context-wise collectives such as all-gather or all-to-all.

Some parallelisms allow partial overlap between communication and computation, improving the efficiency of distributed computation. A representative topology that can support this overlap is the multi-dimensional torus network [18-19], which is particularly well suited for strategies such as DP and PP. It can also be applied to TP or SP under certain conditions, with TP overlap typically limited by frequent synchronization points, and SP overlap depending on implementation and integration of sequence-parallel collectives. However, torus networks are generally less suitable for EP due to its sparse all-to-all communication pattern.

Since each phase of AI use cases (Fig. 7) has different objectives and demand different requirements for the network, their workload characteristics are more deeply examined by analyzing representative AI application in the following subsections.

A.2. LLMs for text generation, question answering, and translation

This subsection analyzes the distributed computation for LLMs. LLMs are employed for AI applications such as text generation, question answering, and translation, enabling context-aware and natural language outputs across diverse domains.

During the first phase (LLM training), distributed computation employing multi-dimensional parallelisms such as DP, PP, TP, and EP are executed on AI clusters [18-19]. Among these parallelization strategies, TP and EP are communication-bound, as they require fine-grained and high-frequency synchronization of large intermediate tensors or token expert routing, thereby imposing stringent bandwidth and inter-XPU latency requirements [6, 14-16]; accordingly, this document refers to these parallelisms as communication-intensive parallelisms. On the other hand, DP and PP are computation-bound, as their performance is primarily limited by the computational throughput of the underlying hardware and demand less communication overhead [6]; accordingly, this document refers to these parallelisms as computation-intensive parallelisms. This phase takes tens of hours to days because state-of-the-art LLM training requires processing extremely large datasets, typically consisting of millions to billions of training samples or tokens, using large batch sizes (hundreds to thousands), and performing a vast number of optimization steps across a massive number of XPUs. During this phase, system failures due to XPU and transceiver failures, must be prevented; however, such failures arise on a timescale of a few to several tens of hours for a single resource unit which could lead to entire system failures [19-20].

In the second phase (fine-tuning), pre-trained models are adapted to specialized domains or tasks through distributed computation such as parameter updates across large models while maintaining the integrity of their pre-trained linguistic and generative knowledge. Similar to the training phase, fine-tuning employs distributed computation, typically involving DP, PP and TP, to update model parameters. However, the computational and communication demands in this phase are significantly lower than those of training phase because the dataset size is orders of magnitude smaller (ranging from thousands to millions of samples rather than billions), batch sizes are reduced, and the number of optimization steps is fewer than training phase. Consequently, the required number of XPU is smaller, and the execution time of this phase is shorter than the training phase (hours to days).

Finally, in the inference phase, the LLMs process input prompts using pre-trained model parameters to generate context-aware and domain-/task-specific outputs. The inference relies on model parallelisms such as PP and TP. In addition, it involves not only processing large AI models using the model parallelisms, but also two distinct processes with different computational and

communication demands: understanding the prompt (Prefill) and generating the response (Decode). Prefill processes multiple tokens together, making it computationally intensive, while Decode processes one token at a time, resulting in frequent KV-cache data transfers and high communication overhead. Since KV-cache transfers are dependent and often chained across multiple pipeline stages or devices during distributed inference, a low-latency interconnect is essential to prevent cumulative communication delays from violating these real-time requirements [25][26]. Whereas training and fine-tuning require tens of hours to days due to large datasets and extensive optimization steps, the execution time of inference must be milliseconds to seconds per request to satisfy real-time service-level user-experience.

A.3. Multi-modal AI for text-to-video generation, image captioning, and visual question answering

This subsection analyzes the distributed computation for multi-modal AI applications. Multi-modal AI integrates multiple data modalities including not only text as LLMs but also image, audio, and video to perform tasks that require cross-modal understanding or generation. Such AI applications include text-to-video generation, image captioning, and visual question answering.

As in the case of LLMs, the training phase of multi-modal AI applications relies on large-scale distributed computation using multi-dimensional parallelisms such as DP, PP, TP, and EP (Fig. 4). However, multi-modal datasets are typically substantially larger and more heterogeneous than those used for LLMs because they include multi-frame videos, high-resolution images, and long-duration audio streams, often paired or aligned with textual annotations. This heterogeneity amplifies both data volume and data-processing complexity; for example, a single training instance may include thousands of video frames or high-frequency audio segments in addition to text tokens. As a result, future multi-modal AI training is expected to require tens of thousands of XPUs to meet computational throughput requirements and to process datasets that may reach exabyte-scale over time [27-28]. In addition, the training time remains tens of hours to days, similar to or potentially longer than LLM training, and reliability is even more critical because failures during long multi-modal training runs may invalidate progress across multiple modality pipelines.

Following the training phase, multi-modal models undergo a fine-tuning phase in which both modality-specific modules (e.g., vision encoders, audio encoders) and cross-modal fusion layers (e.g., multi-head cross-attention) are adapted to specialized tasks or domains. Compared with training, the dataset scale is smaller (i.e., often thousands to millions of multi-modal samples), and the number of steps is significantly reduced. Thus, fine-tuning is executed on fewer XPUs, with lower communication intensity than training phase as in the LLM case. Nevertheless, fine-tuning multi-modal models remain more complex than fine-tuning LLMs because:

- Updates must preserve alignment across heterogeneous modalities,
- Different modalities may have different temporal or spatial granularities, and

- Cross-modal fusion layers may still require TP-level synchronization across large intermediate tensors.

Consequently, execution time typically spans hours to days, depending on modality resolution, task complexity, and the extent of parameter updates.

Finally, in the inference phase, cross-modal processing could be performed on AI clusters for real-time generation or analysis of multi-modal outputs. Similar to LLM inference, this process consists of Prefill and Decode stages where Prefill processes multi-modal inputs such as video frames, audio segments, and text often performing cross-modal embedding, temporal aggregation, and fusion. This stage is compute-intensive, especially for high-resolution images or long video clips, and may involve substantial tensor exchange across XPUs for cross-modal attention while Decode generates outputs autoregressively (e.g., text tokens, video frames, audio samples). As with LLMs, Decode becomes communication-bound, as each generation step requires fine-grained synchronization of KV-caches, attention states, and cross-modal embeddings. However, in multi-modal models, this effect is amplified:

- Cross-modal attention paths increase KV-cache size,
- Video or audio generative models require higher-dimensional representations, and
- Fusion modules amplify synchronization frequency.

Even tens of microseconds of interconnect latency accumulate across multiple attention layers and decoding steps, resulting in degraded alignment between modalities and inconsistent user experience during real-time interactions [29-30]. Therefore, future multi-modal inference systems demand inter-XPU latency on the order of a few microseconds, in addition to high-bandwidth scale-up fabrics, to sustain real-time responsiveness and seamless cross-modal coherence. Execution time must remain milliseconds to seconds per request, depending on content type, to support interactive applications such as live video captioning or conversational visual agents.

B. Abbreviations and Acronyms

Abbreviation	Description
AI	Artificial Intelligence
ASIC	Application Specific Integrated Circuit
Bi-Di	Bidirectional
CAPEX	Capital Expenditure
CP	Context Parallelism
CPO	Co-Packaged Optics
CXL	Compute Express Link
DC	Data Center
DP	Data Parallelism
EP	Expert Parallelism
EPS	Electrical Packet Switch
GPU	Graphics Processing Unit
I/O	Input/Output
KV	Key-Value
LLM	Large Language Model
LPO	Linear-drive Pluggable Optics
MoE	Mixture-of-Expert
NPO	Near-Packaged Optics
NIC	Network Interface Card
OCI	Optical Cabling Infrastructure
OCS	Optical Circuit Switch
OE	Optical Engine
OPEX	Operational Expenditure
PCIe	Peripheral Component Interconnect Express
PP	Pipeline Parallelism
P-D	Prefill-Decode

Abbreviation	Description
SerDes	Serializer/De-serializer
SO	Scale-Out
SP	Sequence Parallelism
SU	Scale-Up
TP	Tensor Parallelism
TPU	Tensor Processing Unit
TRx	Transceiver
UBB	Universal BaseBoard
XPU	General term for Processing Unit (such as GPU and TPU)
XPUaaS	XPU-as-a-Service

C. References

[1] DriveNets, Neocloud Providers: The Future of GPUaaS for AI Workloads, 2025. [Online]. Available: https://drivenets.com/resources/education-center/what-are-neocloud-providers/

[2] WWT, What is GPU-as-a-Service (GPUaaS) or GPU Cloud?, 2024. [Online]. Available: https://www.wwt.com/article/what-is-gpu-as-a-service-gpuaas-or-gpu-cloud

[3] Z. Jiang et al, "MegaScale: Scaling Largeban Language Model Training to More Than 10,000 GPUs," in Proc. 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI '24), 2024, pp. 745–758.

[4] D. Narayanan et al., “Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM,” in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC ’21), Nov. 2021.

[5] S. Yeluri, “GPU Fabrics for GenAI Workloads,” Juniper Community Blog, Jan. 2, 2024. [Online]. Available: https://community.juniper.net/blogs/sharada-yeluri/2024/01/02/gpu-fabrics-for-genai-workloads

[6] C. Shou, et al., "InfiniteHBD: Building datacenter-scale high-bandwidth domain for LLM with optical circuit switching transceivers," Proc. ACM SIGCOMM, 2025.

[7] NVIDIA, GB200 NVL72: Grace + Blackwell rack-scale NVLink domain for real-time trillion-parameter LLM inference, NVIDIA Corporation, 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/gb200-nvl72/

[8] S. Priyadarshi, "Advancing AI Scalability and Performance with Optical Interconnects.," in IEEE Communication Magazine, 2025.

[9] W. Wang et al., “Rail-only: A low-cost high-performance network for training LLMs with trillion parameters,” in Proc. IEEE Symp. High-Performance Interconnects (HOTI), Aug. 2024, pp. 1–10.

[10] “Multi-Tenant HPC and AI: How the Network Can Make or Break the System,” HPCwire, Jul. 24, 2025. [Online]. Available: https://www.hpcwire.com/2025/07/24/multi-tenant-hpc-and-ai-how-the-network-can-make-or-break-the-system/

[11] Innovative Optical and Wireless Network Global Forum (IOWN GF), "Remote GPU Use Case: RIM PoC Reference 1.1," IOWN GF, Mar. 2025. [Online]. Available: https://iowngf.org/wp-content/uploads/2025/03/IOWN-GF-RD-Remote-GPU-Use-Case-RIM-PoC-Ref-1.1-1.pdf

[12] D. Huang et al., “From Large Language Models to Large multi-modal Models: A Literature Review,” Applied Sciences, vol. 14, no. 12, p. 5068, 2024.

[13] D. Zhang et al., “MM-LLMs: Recent Advances in multi-modal Large Language Models,” arXiv preprint arXiv:2401.13601, 2024.

[14] Open Compute Project, Scale Up Ethernet (SUE), Version 1.0.0, Open Compute Project, Sep. 5, 2025. [Online]. Available: https://www.opencompute.org/documents/ocp-sue-spec-final-pdf-1

[15] Ultra Ethernet Consortium, Ultra Ethernet Specification, version 1.0.1, Ultra Ethernet Consortium, Sep. 5, 2025.

[16] X. Song et al., "Survey of Intra-Node GPU Interconnection in Scale-Up Network: Challenges, Status, Insights, and Future Directions," Future Internet, vol. 17, no. 12, p. 537, 2025.

[17] S. Chen, et al., "Toward Co-adapting Machine Learning Job Shape and Cluster Topology," arXiv e-prints, arXiv:2510.03891, 2025.

[18] Y. Zu, et al, “Resiliency at Scale: Managing Google’s TPUv4 Machine Learning Supercomputer,” in 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pp. 761-774, 2024.

[19] G. Patronas et al., “Optical switching for data centers and advanced computing systems,” J. Opt. Commun. Netw., vol. 17, no. 1, pp. A87–A95, 2024.

[20] R. Singh, et al., "Surviving switch failures in cloud datacenters," ACM SIGCOMM Comput. Commun. Rev., vol. 51, pp. 2–9, 2021.

[21] Z. Jiang et al., “MegaScale: Scaling Large Language Model Training to More Than 10,000 GPUs,” arXiv preprint arXiv:2402.15627, Feb. 23 2024.

[22] L. Poutievski et al., “Jupiter evolving: transforming Google's datacenter network via optical circuit switches and software-defined networking,” in Proceedings of the ACM SIGCOMM 2022 Conference, 2022.

[23] H. Esmaeilzadeh et al., “Power challenges may end the multicore era,” Commun. ACM, vol. 56, no. 2, pp. 93–102, Feb. 2013, doi:10.1145/2408776.2408797.

[24] LINK-PP Official, “Forward Error Correction (FEC) in Optical Networks | 100G, 400G & 800G Ethernet Explained,” Link-PP, Sep. 16, 2025. [Online]. Available: https://www.link-pp.com/knowledge/forward-error-correction-fec-optical-networks.html

[25] Z. Zhong et al., “DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving,” in Proc. 18th USENIX Symp. Operating Systems Design and Implementation (OSDI), 2024.

[26] S. Rajbhandari et al., “DeepSpeed Inference: Enabling Efficient Inference of Transformer Models at Unprecedented Scale,” in Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis (SC), 2022.

[27] D. Zhang et al., “MM-LLMs: Recent Advances in multi-modal Large Language Models,” arXiv preprint arXiv:2401.13601, 2024.

[28] D. Huang et al., “From Large Language Models to Large multi-modal Models: A Literature Review,” Applied Sciences, vol. 14, no. 12, p. 5068, 2024.

[29] M. Tsimpoukelli, et al., “Multi-modal Few-Shot Learning with Frozen Language Models,” in Advances in Neural Information Processing Systems (NeurIPS), 2021.

[30] J. B. Alayrac, et al., “Flamingo: A Visual Language Model for Few-Shot Learning,” in Advances in Neural Information Processing Systems (NeurIPS), 2022.

History

Revision	Release Date	Summary of Changes
1	March 2026	Initial Release