Benchmarking Methodology Working Group
Internet-Draft
Updates: 2544 (if approved)
Intended status: Informational
Expires: 28 August 2026

F. Calabria
Cisco
C. Pignataro
Blue Fern Consulting
Q. Wu
G. Fioccola
Huawei

24 February 2026

Benchmarking Methodology for AI Inference Serving Network Fabrics
draft-calabria-bmwg-ai-fabric-inference-bench-01

Abstract

This document defines benchmarking terminology, methodologies, and Key Performance Indicators (KPIs) for evaluating Ethernet-based AI inference serving network fabrics. As Large Language Model (LLM) inference deployments scale to disaggregated prefill/decode architectures spanning hundreds or thousands of accelerators (GPUs/XPUs), the interconnect fabric becomes the critical bottleneck determining Time to First Token (TTFT), Inter-Token Latency (ITL), and aggregate throughput in tokens per second (TPS). This document establishes vendor-independent, reproducible test procedures for benchmarking fabric-level performance under realistic AI inference workloads.

Coverage includes RDMA-based KV cache transfer between disaggregated prefill and decode workers, Mixture-of-Experts (MoE) expert parallelism AllToAll communication, request routing and load balancing for inference serving, congestion management under bursty inference traffic patterns, and scale/soak testing. The methodology enables direct, equivalent comparison across implementations, NIC transport stacks (RoCEv2, UET), and fabric architectures.

This document is a companion to [TRAINING-BENCH], which addresses training workloads.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://fcalabri.github.io/bmwg-ai-fabric-inference-bench/draft-calabria-bmwg-ai-fabric-inference-bench.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-calabria-bmwg-ai-fabric-inference-bench/.

Discussion of this document takes place on the BMWG Working Group mailing list (mailto:bmwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/bmwg/. Subscribe at https://www.ietf.org/mailman/listinfo/bmwg/.

Source for this draft and an issue tracker can be found at https://github.com/fcalabri/bmwg-ai-fabric-inference-bench.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Large Language Model (LLM) inference serving has emerged as a dominant consumer of datacenter network capacity, with fundamentally different fabric requirements compared to training workloads. While training workloads are characterized by bulk synchronous collective operations (AllReduce, AllGather) with predictable periodicity, inference workloads exhibit bursty, latency-sensitive request/response patterns with strict Service Level Objectives (SLOs) on per-token latency and time-to-first-token.

The advent of disaggregated serving architectures, where the computationally intensive prefill phase (prompt processing) is physically separated from the memory-bound decode phase (token generation), introduces a new class of fabric-critical data movement: KV cache transfer. A single large prompt processed by a typical large-scale model generates multiple gigabytes of KV cache state that must be transferred from prefill workers to decode workers within a fraction of the target TTFT SLO.

As clusters scale with thousands of concurrent requests, this creates sustained multi-terabyte-per-second aggregate transfer demands on the fabric. Simultaneously, Mixture-of-Experts (MoE) architectures introduce expert parallelism (EP), which distributes expert sub-networks across GPUs and requires AllToAll communication for token-to-expert routing. Wide EP configurations (e.g., 96-way EP across 12 nodes of 8 GPUs each) generate fine-grained, latency-sensitive inter-node traffic that contends with KV cache transfers on shared fabric links.

This document defines vendor-independent benchmarking methodologies for evaluating how well a network fabric supports these inference-specific traffic patterns. All tests are designed for controlled laboratory environments using either hardware traffic generators or software workload emulators capable of reproducing inference serving traffic profiles.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Scope and Applicability

The scope covers Layer 2/3 fabric performance (switch forwarding, link utilization, congestion management), RDMA transport performance (one-sided PUT/GET operations for KV cache transfer, two-sided SEND/RECV for expert parallelism dispatch), and the interaction between fabric behavior and application-level inference metrics (TTFT, ITL, TPS).

The DUT boundary for all measurements in this document is defined as the NIC-to-NIC Ethernet fabric segment — specifically, the path from the point of packet transmission by the source NIC Ethernet port to the point of packet reception at the destination NIC Ethernet port.

Intra-node transfer segments (NVLink GPU-to-GPU, PCIe/CXL GPU-to-NIC) are explicitly OUT OF SCOPE as primary benchmarked entities. Where intra-node transfer contributes measurably to an end-to-end latency measurement (e.g., TTFT decomposition in Section 6.1), implementers MUST report intra-node transfer time as a separately labelled component so that the fabric contribution can be isolated. See Section 3.2 for DUT boundary diagram.

The document does NOT address benchmarking of individual accelerator (GPU/XPU) compute performance, model accuracy or quality metrics benchmarking of the inference serving software stack in isolation from the fabric.

All methodologies assume controlled laboratory conditions per BMWG convention.

1.3. Relationship to Existing BMWG Work

This document builds upon the foundational BMWG benchmarking framework established by [RFC1242], [RFC2544], [RFC2889], and [RFC6349].

The test structure follows RFC 2544 conventions for trial duration (minimum 60 seconds), statistical repetition (minimum 20 trials for latency, 50 for burst), and reporting format (graphical and tabular).

The methodologies extend RFC 2544 Section 26 benchmarks (throughput, latency, frame loss rate, back-to-back frames, system recovery, reset) to inference-specific scenarios including KV cache transfer, expert parallelism dispatch, and disaggregated serving request routing.

1.4. Relationship to Companion Documents

This document is a companion to [TRAINING-BENCH], which defines benchmarking methodologies for AI training network fabrics. Both documents share common terminology (Section 2), test topology conventions (Section 3), and reporting formats (Section 14). Both documents use the terminology defined in [TERMINOLOGY], which provides the common terminology base for AI fabric benchmarking.

Where training workloads are dominated by bulk synchronous collective communication (AllReduce, AllGather) with high bandwidth utilization and periodic synchronization barriers, inference workloads are dominated by bursty, latency-sensitive point-to-point transfers (KV cache) and fine-grained AllToAll dispatch (MoE expert parallelism). Implementers deploying converged fabrics that serve both training and inference workloads SHOULD run both test suites.

2. Terminology and Definitions

The following terms are used throughout this document. Terms defined in the companion training document are referenced but not redefined unless the inference context introduces substantive differences.

TTFT: Time to First Token. The elapsed time from receipt of an

inference request by the serving system to emission of the first output token. Includes prompt processing (prefill), KV cache generation, optional KV cache transfer (in disaggregated architectures), and initial decode step. Target: < 500 ms for interactive serving.

ITL: Inter-Token Latency. The elapsed time between successive output tokens during the autoregressive decode phase. Measured at P50, P95, P99, and P99.9 percentiles. Target: < 50 ms P99 for interactive serving.

TPS: Tokens Per Second. Aggregate throughput of the serving system measured as the total number of output tokens generated per second across all concurrent requests. Reported separately for input (prefill) TPS and output (decode) TPS.

KV Cache: Key-Value Cache. The intermediate attention state (key and value projection matrices) computed during the prefill phase and reused during each decode step. Size scales with model dimension, number of layers, number of attention heads, sequence length, and numerical precision. For a 70B parameter model at FP16 with 4K context: approximately 1.34 GB per request.

Prefill Phase: The compute-bound phase of inference in which the entire input prompt is processed in parallel to generate the KV cache and the first output token. Characterized by high arithmetic intensity (200-400 ops/byte), high GPU utilization (90-95%), and large activation tensors.

Decode Phase: The memory-bound phase of inference in which output tokens are generated autoregressively, one token per forward pass. Characterized by low arithmetic intensity (60-80 ops/byte), lower GPU utilization (20-40%), and memory-bandwidth-limited KV cache reads.

Disaggregated Serving: An inference serving architecture in which prefill and decode computations are executed on physically separate groups of accelerators (workers), connected by a network fabric. The KV cache generated by prefill workers are transferred over the fabric to decode workers.

xPyD Ratio: The allocation ratio of prefill (x) to decode (y) resources in a disaggregated serving cluster. For example, 3P9D indicates 3 prefill nodes and 9 decode nodes. The optimal ratio depends on model size, prompt length distribution, output length distribution, and SLO targets.

EP: Expert Parallelism. A parallelism strategy for Mixture-of-

Experts (MoE) models in which expert sub-networks are distributed across multiple GPUs. Token routing to the appropriate experts requires AllToAll communication.

Wide EP: Expert Parallelism spanning many GPUs (e.g., 96-way EP across 12 nodes), requiring inter-node AllToAll communication for every MoE layer forward pass.

DP Attention: Data Parallelism applied to the attention computation, where the KV cache is partitioned across data-parallel ranks. Each rank holds 1/DP_SIZE of the KV cache, and AllToAll communication is used to exchange attention outputs.

MoE: Mixture of Experts. A model architecture that activates only a subset of expert sub-networks for each token, enabling larger model capacity with sub-linear compute scaling.

Normal Dispatch: A communication mode for AllToAll MoE dispatch optimized for the prefill phase. Maximizes throughput for long input sequences but generates dynamic (symbolic) shapes incompatible with CUDA Graph.

Low-Latency Dispatch: A communication mode for AllToAll MoE dispatch optimized for the decode phase. Uses fixed input shapes compatible with CUDA Graph, reducing kernel launch overhead at the cost of slightly lower peak throughput.

RDMA: Remote Direct Memory Access. A transport mechanism enabling direct memory-to-memory data transfer between hosts without CPU involvement. Implementations include InfiniBand Verbs and RoCEv2 (RDMA over Converged Ethernet v2).

RoCEv2: RDMA over Converged Ethernet version 2. An RDMA transport that encapsulates InfiniBand transport over UDP/IP, enabling RDMA semantics on standard Ethernet fabrics.

UET: Ultra Ethernet Transport. A transport protocol defined by the Ultra Ethernet Consortium (UEC) Specification 1.0, offering ordered/unordered reliable delivery, multipath packet spraying, and integrated congestion control for AI/HPC workloads.

KVCXL: KV Cache Transfer Library. A library providing standardized point-to-point data transfer primitives (register, transfer, notify) for inference engines, abstracting underlying transports (intra-node interconnect, RDMA, PCIe, and storage interfaces). Multiple open-source and vendor implementations exist.

GIN: GPU-Initiated Networking. A communication paradigm where GPU

threads directly initiate network operations (RDMA sends, one-sided puts) without CPU involvement, reducing latency by eliminating CPU-GPU synchronization.

PagedAttention: A memory management technique for KV caches that stores attention keys and values in fixed-size pages (typically, 16-64 KB), enabling non-contiguous allocation and reducing memory fragmentation.

Continuous Batching: A scheduling technique that dynamically adds new requests to an active inference batch as decode slots become available, improving GPU utilization compared to static batching.

Prefix Caching: Reuse of previously computed KV cache segments for prompts that share a common prefix (e.g., system prompt), avoiding redundant prefill computation.

DUT: Device Under Test. In this document, the DUT is one or more network fabric elements (switches, NICs, or the complete fabric) whose performance impact on inference serving is being characterized.

SUT: System Under Test. The complete inference serving system including accelerators, NICs, fabric, and serving software, when end-to-end metrics are being measured.

RT: Router Tester / Traffic Generator. Test equipment capable of generating and receiving network traffic at specified rates with timestamping accuracy sufficient for the measurements defined herein.

S_KV: S_KV (KV Cache Transfer Size) The total size in bytes of the KV cache state generated by a single inference request across all transformer layers and all context tokens, computed as:

S_KV = 2 x L x H_kv x D x C x P_bytes

Where:

L = number of transformer layers

```
H_kv     = number of KV attention heads per layer
             (H_kv <= H_total for GQA/MQA; see note below)

D        = per-head key/value dimension (head_dim)
             Typically: head_dim = model_dim / H_total

C        = context length in tokens (prompt tokens + generated tokens)

P_bytes  = precision in bytes per element
             (FP16 / BF16 = 2,  FP8 / INT8 = 1)

Factor 2 = accounts for both the K (key) and V (value) tensors,
           each of shape \[H_kv, D\] per layer per token

Attention variant mapping for  H_kv:

MHA (Multi-Head Attention):    H_kv = H_total

GQA (Grouped-Query Attention): H_kv = H_total / GQA_ratio
                               (e.g., H_total=64, GQA_ratio=8 -> H_kv=8)

MQA (Multi-Query Attention):   H_kv = 1
```

This formula yields the total KV cache bytes for one complete inference request. The per-layer, per-token contribution is:

s_kv_unit = 2 x H_kv x D x P_bytes (bytes per layer per token)

and S_KV = s_kv_unit x L x C.

Assumption: all layers share identical H_kv and D values. Hybrid architectures (e.g., sliding-window + full-attention layers) MUST substitute per-layer values and sum across layers.

3. Test Topology and Architecture

3.1. Reference Fabric Topologies

The reference topologies from the companion training document (2-Tier Clos, 3-Tier Clos, Rail-Optimized) remain applicable. Inference serving introduces additional topology considerations related to disaggregated prefill/decode placement and MoE expert distribution.

3.1.1. Topology A: 2-Tier Clos (Leaf-Spine)

Applicable to inference clusters up to approximately 2,048 accelerators. Prefill and decode worker groups SHOULD be placed on separate leaf switches (or separate leaf switch groups) to isolate KV cache transfer traffic from decode-to-client response traffic. Expert parallelism (EP) traffic within a single MoE dispatch group SHOULD be confined to a single leaf switch or a minimal number of leaf switches to minimize spine-hop latency.

3.1.2. Topology B: 3-Tier Clos (Leaf-Spine-Superspine)

Required for inference clusters exceeding 2,048 accelerators or for multi-model serving deployments where different model instances occupy different fabric pods. KV cache transfer traffic between prefill and decode workers in different pods traverses the superspine tier, making superspine bandwidth and latency critical.

3.1.3. Topology C: Disaggregated Prefill/Decode Placement

A topology variant specific to inference serving in which prefill workers and decode workers are placed in distinct physical locations within the fabric, connected by a dedicated KV cache transfer network segment. This topology enables independent scaling of prefill and decode resources and allows heterogeneous hardware (e.g., high-compute GPUs for prefill, high-memory-bandwidth GPUs for decode).

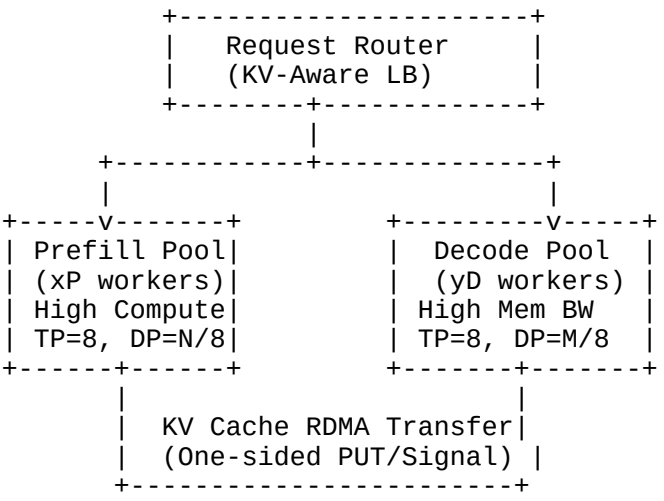

Figure 1: Disaggregated Prefill/Decode Inference Topology

3.2. Disaggregated Prefill/Decode Topology

The disaggregated topology separates the inference pipeline into physically distinct pools connected by the fabric. The test topology MUST include the following components:

* *Prefill Worker Pool:* N Prefill nodes, each containing G accelerators with high-compute capability. These workers execute the prefill phase and generate KV cache state. Tensor Parallelism (TP) is applied within each node; Data Parallelism (DP) is applied across nodes. Each prefill worker communicates with one or more decode workers via RDMA-based KV cache transfer.

* *Decode Worker Pool:* M Decode nodes, each containing G accelerators with high memory bandwidth. These workers receive KV cache state from prefill workers and execute the autoregressive decode phase. DP Attention may partition the KV cache across DP ranks within the decode pool, requiring AllToAll communication during decode.

* *KV Cache Transfer Network:* The Ethernet fabric segment connecting prefill and decode worker pools. This segment carries one-sided RDMA PUT operations (or PUT-with-signal) transferring KV cache blocks from prefill GPU memory to decode GPU memory via RDMA over Converged Ethernet (RoCEv2) or Ultra Ethernet Transport (UET).

  NOTE ON TRANSFER PATH DECOMPOSITION: The end-to-end transfer from GPU memory to remote GPU memory traverses three segments: (1) GPU-to-NIC: PCIe/CXL (intra-node, out of scope as DUT); (2) NIC-to-NIC: Ethernet fabric (THE DUT — in scope); (3) NIC-to-GPU: PCIe/CXL at destination (intra-node, out of scope as DUT). Benchmarking procedures in Sections 5 and 6 measure fabric-segment latency and throughput exclusively. When end-to-end measurements are reported (e.g., TTFT decomposition), the intra-node segments MUST be labelled separately.

The end-to-end transfer from GPU memory to remote GPU memory traverses three segments: (1) GPU-to-NIC: PCIe/CXL (intra-node, out of scope as DUT); (2) NIC-to-NIC: Ethernet fabric (the DUT, in scope); (3) NIC-to-GPU: PCIe/CXL at destination (intra-node, out of scope as DUT). Benchmarking procedures in Sections 5 and 6 measure fabric-segment latency and throughput exclusively. When end-to-end measurements are reported (e.g., TTFT decomposition), the intra-node segments MUST be labelled separately.

~~~~ GPU Memory --> [PCIe/CXL] --> NIC --> [ETHERNET FABRIC] --> NIC --> [PCIe/CXL] --> GPU Memory <---intra-node (out of scope)--->|<------DUT (in scope)------->|<---intra-node (out of scope)---> ~~~~

* *Request Router:* A network-layer or application-layer load balancer that assigns incoming inference requests to prefill workers and subsequently routes KV cache to the appropriate decode workers. KV-aware routing and prefix-aware caching policies are under test.

3.3. Device Under Test (DUT) Identification

The following table defines the DUT configurations tested in this document:
~~~~

DUT ID	Description	Components Under Test
DUT-S	Single Switch	Individual leaf or spine switch forwarding inference traffic. Measures per-hop latency, buffer absorption, ECN marking accuracy.
DUT-F	Complete Fabric	End-to-end fabric from prefill NIC egress to decode NIC ingress. Measures fabric-level KV cache transfer latency, throughput, and congestion behavior.
DUT-N	NIC Transport	NIC RDMA transport stack processing KV cache transfer operations. Measures RDMA verb completion latency, one-sided PUT bandwidth, QP scaling.
DUT-PD	Prefill-Decode Path	The complete data path from prefill GPU memory through NIC, fabric, NIC, to decode GPU memory. Measures end-to-end KV cache transfer including NVLink, PCIe, and fabric segments.
SUT-E	End-to-End System	Complete inference serving system including inference serving software, RDMA transfer libraries, fabric, and accelerators. Measures TTFT, ITL, TPS as functions of fabric performance.

Table 1: DUT Configuration Definitions

3.4. Traffic Generator and Workload Emulator Requirements

Tests in this document require one or both of the following traffic generation modes. The mode used MUST be documented in all test reports.

3.4.1. Hardware Traffic Generator (RT) - Minimum Requirements

The hardware traffic generator MUST satisfy all of the following:

* RDMA traffic generation supporting RoCEv2 and, where tested, UET transport; configurable RDMA verb types (one-sided PUT, PUT-with-signal, two-sided SEND/RECV).

* Configurable message sizes from 4 KB (minimum KV cache page) to 256 MB (large KV cache block).

* Configurable QP counts from 1 QP to a minimum of 256 QPs per source-destination port pair.

3.4.2. Software Workload Emulator (WE) - Minimum Requirements

A software workload emulator runs on actual accelerators and generates realistic inference workloads. The WE MUST support all of the following:

* Configurable prompt length distributions: uniform, Zipf, and trace-replay modes.

* Configurable output length distributions and configurable request arrival rates: Poisson, bursty, and trace-replay.

* Disaggregated prefill/decode execution with actual RDMA-based KV cache transferring between prefill and decode worker pools.

* MoE expert parallelism with actual AllToAll dispatch where MoE-specific tests (Section 7) are performed.

* Measurement instrumentation providing per-request TTFT and ITL with timestamp accuracy <= 1 millisecond.

When a software workload emulator is used, the complete software configuration MUST be documented per Section 3.3, as framework version, RDMA library version, and GPU driver version materially affect results.

4. KPI Framework and Metrics Taxonomy

This section defines the Key Performance Indicators measured across all test categories. KPIs are organized into four tiers: Primary Latency KPIs (end-user-facing response time metrics), Primary Throughput KPIs (system-level capacity metrics), Fabric-Level KPIs (network-specific measurements), and Fabric Health Indicators (operational monitoring metrics).

> NOTE: Where numerical reference values appear in the Target column of the KPI tables below (including TTFT, ITL, and other latency targets), these values are non-normative informational reference points reflecting current industry observations for interactive inference workloads as of 2025-2026. They do NOT constitute benchmarking acceptance criteria or performance requirements. Per the BMWG charter, the definition of acceptance criteria or

performance requirements is explicitly outside the scope of this Working Group. Implementers MAY use these values as contextual references when interpreting results; they MUST NOT be used as pass/fail thresholds in vendor evaluations. Deployment-specific SLOs will vary by application, model architecture, and operator requirements.

4.1. Primary Latency KPIs

KPI	Unit	Definition	Target (Interactive)	Measurement Point
TTFT	ms	Time from request arrival to first output token emission	< 500 ms P99	SUT-E request/ response boundary
ITL	ms	Time between successive output tokens	< 50 ms P99	SUT-E token emission timestamps
TTFT_fabric	ms	Fabric contribution to TTFT (KV cache transfer latency)	< 300 ms P99	DUT-PD NIC-to-NIC measurement
ITL_fabric	ms	Fabric contribution to ITL (EP dispatch latency per decode step)	< 5 ms P99	DUT-F EP dispatch round-trip
E2E_latency	ms	End-to-end request latency from arrival to completion of all output tokens	Varies by output length	SUT-E request/ response boundary

Table 2: Primary Latency KPIs

4.2. Primary Throughput KPIs

KPI	Unit	Definition	Measurement Point
TPS_input	tokens/s	Aggregate input (prefill) tokens processed per second across all workers	SUT-E prefill completion events
TPS_output	tokens/s	Aggregate output (decode) tokens generated per second across all workers	SUT-E token emission events
TPS_per_GPU	tokens/s/ GPU	Output tokens per second normalized by number of decode GPUs	SUT-E per-worker counters
Goodput	GB/s or tokens/s	See the Goodput definition in [TERMINOLOGY] Reports MUST use Inference_Goodput for token-rate measurements and Fabric_Goodput for byte-rate fabric measurements	SUT-E successful completion events
KV_BW	GB/s	Aggregate KV cache transfer bandwidth between prefill and decode pools	DUT-PD RDMA counters
Request_Rate	req/s	Maximum sustained request arrival rate meeting all latency SLOs	SUT-E admission control boundary

Table 3: Primary Throughput KPIs

4.3. Fabric-Level KPIs

KPI	Unit	Definition	DUT
KV_xfer_latency	us	One-sided RDMA PUT completion time for a single KV cache block transfer	DUT-N
KV_xfer_bandwidth	GB/s	Sustained unidirectional KV cache transfer throughput per NIC port	DUT-N
EP_alltoall_latency	us	Round-trip time for a complete MoE expert parallelism AllToAll dispatch	DUT-F
EP_alltoall_bandwidth	GB/s	Aggregate AllToAll bandwidth across all EP ranks during dispatch	DUT-F
Fabric_FCT	us	Flow completion time for a KV cache transfer flow through the fabric	DUT-F
Buffer_utilization	%	Peak switch buffer utilization during KV cache transfer bursts	DUT-S
ECN_marking_rate	%	Fraction of packets marked with ECN-CE during inference traffic	DUT-S
PFC_frame_count	frames	Number of PFC PAUSE frames generated per unit time	DUT-S
Link_utilization	%	Average and peak link utilization on fabric links carrying inference traffic	DUT-F
Packet_drop_rate	ppm	Packets dropped per million due to buffer	DUT-F

			overflow or transport error	

Table 4: Fabric-Level KPIs

4.4. Fabric Health Indicators

Indicator	Threshold	Description
CPU Utilization (switch)	< 30%	Control plane CPU usage on switches under inference traffic load
Memory Usage (switch)	< 70%	TCAM, buffer, and control plane memory usage
FEC Error Rate	< 1e-12 post-FEC BER	Forward Error Correction effectiveness on fabric links
CRC Error Count	0	Layer 2 CRC errors on any fabric link
BGP/OSPF Stability	0 flaps	Routing protocol adjacency stability under inference load
NIC QP State	100% active	All RDMA Queue Pairs in active state (no error/ reset)
GPU-NIC PCIe BW	> 90% of theoretical	PCIe Gen5 x16 bandwidth utilization between GPU and NIC

Table 5: Fabric Health Indicators

5. Test Category 1: RDMA KV Cache Transfer Benchmarks

KV cache transfer between disaggregated prefill and decode workers is the defining fabric workload for inference serving. Unlike training collectives (AllReduce, AllGather) which are periodic and predictable, KV cache transfers are event-driven (triggered by prefill completion) and bursty.

5.1. Point-to-Point KV Cache Transfer Throughput

Objective: To determine the maximum sustained KV cache transfer throughput between a single prefill worker NIC and a single decode worker NIC across the DUT fabric.

Procedure: Configure a single RDMA connection (QP) between the prefill and decode endpoints. Send a sequence of one-sided RDMA PUT operations with message sizes corresponding to KV cache block sizes. The message size sequence MUST include: 64 KB (single attention page), 256 KB, 1 MB, 4 MB, 16 MB, 64 MB, 256 MB (large prompt KV cache), and 1 GB. For each message size, transmit at the maximum rate sustainable by the NIC for a minimum of 60 seconds per trial. Repeat for 1, 4, 8, 16, 32, 64, and 128 concurrent QPs. The DUT is the fabric path from NIC to NIC.

Measurement: Record throughput (GB/s), CPU utilization on both endpoints, GPU memory-to-NIC transfer overhead, and NIC hardware offload utilization. The test MUST be repeated a minimum of 20 times per configuration and the average reported.

Reporting Format: Results SHOULD be reported as a multi-line graph with message size (log scale) on the X axis and throughput (GB/s) on the Y axis. Separate lines for each QP count. A reference line showing theoretical NIC line rate MUST be included.

5.2. KV Cache Transfer Latency

Objective: To determine the latency of individual KV cache block transfers across the DUT fabric under varying load conditions.

Procedure: Using the same endpoint configuration as Test 5.1, measure the completion time of individual RDMA PUT operations. Latency is measured from the initiation of the PUT verb on the prefill NIC to receipt of the completion signal on the decode NIC (for PUT-with-signal) or to polling of the remote completion queue. Measure latency under unloaded conditions (single outstanding operation) and under loaded conditions (background traffic at 25%, 50%, 75%, and 90% of fabric capacity). Message sizes MUST include 64 KB, 1 MB, 16 MB, and 256 MB.

Measurement: Report latency at P50, P95, P99, and P99.9 percentiles. The test MUST be repeated a minimum of 20 trials of at least 120 seconds each per configuration. The difference between P99 and P50 (tail latency spread) SHOULD be reported as a derived metric.

Reporting Format: Results SHOULD be reported as a table with columns for message size, background load level, and latency at each percentile. A complementary CDF plot of latency distribution for selected configurations SHOULD be included.

5.3. Concurrent KV Cache Transfer Scaling

Objective: To characterize how aggregate KV cache transfer performance scales as the number of concurrent prefill-to-decode transfer pairs increases.

Procedure: Configure N concurrent prefill-decode endpoint pairs, where N ranges from 1 to the maximum supported by the fabric (e.g., 1, 2, 4, 8, 16, 32, 64, 128 pairs). Each pair executes continuous KV cache transfers of 16 MB messages (representative of a medium-length prompt). Measure aggregate throughput and per-pair latency as N increases.

Measurement: Report aggregate throughput (GB/s), per-pair median latency (us), per-pair P99 latency (us), Jain Fairness Index across pairs, and maximum fabric link utilization observed. The test MUST be repeated a minimum of 20 times per value of N.

Reporting Format: Results SHOULD be reported as a dual-axis graph with N (concurrent pairs) on the X axis, aggregate throughput on the left Y axis, and P99 latency on the right Y axis. The JFI value for each N SHOULD be annotated.

5.4. Multi-Tier Storage Transfer Characterization

Objective: To characterize KV cache transfer performance across the memory/storage hierarchy: GPU HBM to GPU HBM (inter-node RDMA), GPU HBM to remote CPU DRAM (offload), CPU DRAM to GPU HBM (reload), and GPU HBM to NVMe/SSD (persistent cache).

Procedure: For each tier pair, measure unidirectional transfer throughput and latency for message sizes of 1 MB, 16 MB, and 256 MB. Use zero-copy transfers where supported (GDS for NVMe, GPUDirect RDMA for inter-node).

Measurement: Report throughput (GB/s) and latency (P50, P99) for each tier pair and message size. Report the tier throughput ratio relative to GPU-to-GPU RDMA as a derived metric.

Reporting Format: Results SHOULD be reported as a table with rows for each tier pair and columns for throughput and latency at each message size.

6. Test Category 2: Prefill/Decode Disaggregation Benchmarks

Disaggregated prefill/decode serving separates the two phases onto distinct hardware pools to enable independent optimization and scaling. This section benchmarks the fabric's ability to support the resulting KV cache transfer traffic patterns and their impact on end-to-end inference metrics.

6.1. End-to-End Disaggregated TTFT

Objective: To measure TTFT as a function of prompt length in a disaggregated serving configuration, isolating the fabric contribution.

Procedure: Configure a disaggregated serving system (SUT-E) with a specified xPyD ratio (e.g., 3P9D for a 12-node cluster). Submit inference requests with prompt lengths of 128, 512, 1024, 2048, 4096, 8192, and 16384 tokens. For each prompt length, measure the total TTFT and decompose it into: T_prefill (prefill compute time), T_transfer (KV cache fabric transfer time, measured at DUT-PD), and T_decode_init (first decode step time).

Measurement: Report TTFT (ms) and its decomposition at P50, P95, and P99 percentiles. The ratio T_transfer/TTFT (fabric fraction) SHOULD be reported as a derived metric. The test MUST be repeated a minimum of 20 trials per prompt length.

Reporting Format: Results SHOULD be reported as a stacked bar chart with prompt length on the X axis and TTFT (ms) on the Y axis, with bars decomposed into T_prefill, T_transfer, and T_decode_init. A table of numerical values MUST accompany the chart.

6.2. xPyD Ratio Optimization

Objective: To determine the optimal prefill-to-decode resource ratio for a given model, prompt distribution, and latency SLO, as limited by fabric transfer capacity.

Procedure: For a fixed total number of nodes N (e.g., 12), iterate over xPyD ratios: 1P11D, 2P10D, 3P9D, 4P8D, 6P6D, 8P4D, 10P2D, 11P1D. For each ratio, submit a sustained request stream matching a target request rate with a specified prompt length distribution (e.g., Zipf with alpha=1.0 over [128, 8192] tokens). Measure TTFT P99, ITL P99, TPS_output, and Goodput for each configuration.

Measurement: Report all four metrics for each xPyD ratio and request rate. Identify the Pareto-optimal ratio(s) that maximize TPS_output while meeting TTFT P99 < 500 ms and ITL P99 < 50 ms.

Reporting Format: Results SHOULD be reported as a multi-panel figure with one panel per request rate, each showing xPyD ratio on the X axis and metrics on dual Y axes (TTFT/ITL on left, TPS on right). The Pareto frontier SHOULD be highlighted.

6.3. Heterogeneous Parallelism Configuration

Objective: To evaluate the fabric impact of using different parallelism strategies on prefill vs. decode pools in a disaggregated configuration.

Procedure: Test the following parallelism configurations:

* Prefill TP=8, Decode TP=8 (baseline, same parallelism)

* Prefill TP=8, Decode TP=4 with DP_Attention=2 (reduced TP, added DP)

* Prefill TP=4 with DP=2, Decode TP=2 with DP_Attention=4 (aggressive DP)

Measurement: Report the number of concurrent RDMA flows, aggregate bandwidth (GB/s), TTFT (ms), and ITL (ms) at P50 and P99 for each configuration.

6.4. Prefill Queue Depth Impact on Transfer Latency

Objective: To measure how queuing of prefill requests (due to compute contention) affects KV cache transfer burstiness and fabric congestion.

Procedure: Oversubscribe the prefill pool by submitting requests at a rate exceeding prefill capacity. Measure the resulting KV cache transfer burst characteristics: burst size, burst duration, inter-burst gap, and peak fabric bandwidth demand. Vary the oversubscription ratio from 1.0x (saturated) to 2.0x in 0.25x increments.

Measurement: Report burst size distribution, peak and average fabric bandwidth, KV transfer latency P99, and ECN/PFC event counts as functions of oversubscription ratio.

7. Test Category 3: MoE Expert Parallelism Benchmarks

Mixture-of-Experts models distribute expert sub-networks across GPUs and route tokens to the appropriate experts via AllToAll communication. This section benchmarks the fabric's ability to support the resulting fine-grained, latency-sensitive inter-GPU traffic patterns.

7.1. AllToAll Dispatch Throughput

Objective: To determine the maximum AllToAll dispatch throughput for MoE expert parallelism across the DUT fabric.

Procedure: Generate a synthetic MoE dispatch workload where each GPU sends token embeddings to the experts selected by a top-k routing function. The dispatch payload per GPU per MoE layer is:

T_dispatch = (B * k * H_model * P_bytes) / N. where B = batch size (tokens), k = top-k routing count, H_model = hidden dimension, P_bytes = precision bytes (BF16=2), N = EP group size

Canonical MoE Test Matrix

Config	E (experts)	k (top-k)	H_model	T_dispatch (B=128, BF16, N=96)
M1	8	2	4096	2.1 MB/GPU
M2	64	4	7168	29 MB/GPU
M3	256	2	7168	14 MB/GPU
M4	256	8	7168	58 MB/GPU
M5	(implementer-defined — report all parameters)			

Table 6: Canonical MoE Test Matrix

Measurement: Report aggregate bandwidth (GB/s), per-dispatch latency (us) at P50 and P99, and GPU idle time waiting for dispatch completion. The test MUST be repeated a minimum of 20 times per configuration.

Reporting Format: Results SHOULD be reported as a heatmap with EP group size on the Y axis, batch size on the X axis, and throughput (GB/s) as the color dimension. A companion latency table MUST be included. Reports MUST state which config row(s) were used. M5 MUST include E, k, H_model, P_bytes, and N in the results table.

NOTE: If per-accelerator normalized throughput (BusBW) is reported alongside EP_alltoall_bandwidth, the algo_factor for AllToAll is (n-1)/n where n is the number of EP ranks. See the BusBW definition in [TERMINOLOGY].

7.2. Routing Mode and Dispatch Mode Comparison.

Objective: To compare fabric performance across dispatch modes and routing policies. Tests MUST cover Normal Dispatch and Low-Latency Dispatch. Tests SHOULD additionally cover at least one alternative routing mode from Table 7.

Routing Mode Taxonomy

Mode	Description	Traffic Impact
Standard Top-k	Each token routed to k. highest-scoring experts	Fixed, uniform AllToAll dispatch volume
Expert Choice (EC)	Experts select tokens; ensures load balance	Non-uniform message sizes; tests HOL-blocking resilience
Top-k with Token Drop	Overloaded experts drop excess tokens	Lower peak traffic; unpredictable under load
Auxiliary Loss Top-k	Load-balanced top-k via training loss	Near-uniform AllToAll; lower hot-spot risk

Table 7: MoE Routing Mode Taxonomy

Measurement: Measure dispatch latency, fabric bandwidth, and routing mode impact on AllToAll traffic distribution and fabric congestion per Table 7 . Results from different routing modes MUST be reported in separate result tables with the routing mode labelled.

7.3. Wide Expert Parallelism Scaling

Objective: To characterize AllToAll dispatch performance as EP group size scales beyond a single node (wide EP), requiring inter-node fabric communication.

Procedure: Scale the EP group from intra-node only (EP=8) to wide EP (EP=16, 32, 48, 64, 96 spanning 2, 4, 6, 8, 12 nodes). Use a fixed batch size of 128 tokens and at least one configuration from the canonical MoE test matrix Table 6. The selected config row MUST be identified in the results.

Measurement: Report total dispatch latency (us), inter-node bandwidth (GB/s), and latency decomposition (intra-node vs. inter-node fraction). Report the scaling efficiency: (EP=8 latency) / (EP=N latency) * (N/8).

7.4. Expert Parallelism and KV Cache Transfer Contention

Objective: To measure the mutual interference between EP AllToAll dispatch traffic and KV cache transfer traffic when both share the same fabric links.

Procedure: On a shared fabric, simultaneously execute: (a) continuous KV cache transfers at a sustained rate (e.g., 50%, 75% of fabric capacity), and (b) periodic EP AllToAll dispatches (one per MoE layer forward pass).

Measurement: Report KV_xfer_latency P99 (us) and EP_alltoall_latency P99 (us) for the isolated and contended cases. Report the contention penalty as the ratio of contended P99 to isolated P99 for each traffic class. Report ECN/PFC event counts during contention.

8. Test Category 4: Congestion Management Benchmarks

Inference traffic patterns differ from training in their burstiness, heterogeneity (mixed KV cache transfers and EP dispatches), and latency sensitivity.

8.1. ECN Marking Under Inference Incast

Objective: To verify that ECN marking thresholds are correctly applied when multiple prefill workers simultaneously transfer KV cache blocks to a single decode worker (incast pattern).

Procedure: Configure M prefill workers (M = 2, 4, 8, 16, 32) to simultaneously transfer 16 MB KV cache blocks to a single decode worker port. Repeat for ECN marking thresholds of 100 KB, 500 KB, 1 MB, and 5 MB. The DUT is the individual leaf switch (DUT-S).

Measurement: Report the ECN marking rate (fraction of marked packets), the onset of marking, queue depth at marking onset, and aggregate throughput achieved. Repeat a minimum of 20 times per configuration.

8.2. PFC Behavior Under Bursty KV Cache Transfers

Objective: To characterize PFC PAUSE frame generation and propagation under bursty KV cache transfer patterns typical of disaggregated serving.

Procedure: Generate KV cache transfer bursts: N_burst concurrent transfers (N_burst = 4, 8, 16, 32), each of size 16 MB, arriving within a window of T_arrival (100 us, 1 ms, 10 ms). Vary the PFC threshold from 10 KB to 1 MB.

Measurement: Report PFC frame count, total PAUSE duration (us), head-of-line blocking delay imposed on other traffic classes (us), and KV cache transfer completion time.

8.3. Congestion Control Convergence for Mixed Traffic

Objective: To measure the convergence time of DCQCN (or UET congestion control) when KV cache transfer traffic and EP AllToAll dispatch traffic share fabric capacity.

Procedure: Establish a sustained KV cache transfer at 80% of fabric capacity. Introduce EP AllToAll dispatch traffic on the same fabric links. Measure the convergence time to stable rate allocation. Repeat with the roles reversed.

Measurement: Report convergence time (ms) to within 5% of steady-state rates, steady-state bandwidth allocation between traffic classes, packet loss during convergence, and Jain Fairness Index of the steady-state allocation.

8.4. PFC Storm and Deadlock Resilience

Objective: To verify that the fabric does not enter a PFC storm or deadlock condition under adversarial inference traffic patterns.

Procedure: Per the companion training document, generate a PFC storm scenario by creating circular buffer dependency across multiple switches. Simultaneously inject KV cache transfer traffic on all affected paths. Monitor for PFC storm propagation, deadlock, and recovery time. The test duration MUST be at least 300 seconds.

Measurement: Report whether PFC storm occurred (yes/no), deadlock occurred (yes/no), maximum PAUSE propagation depth (number of hops), maximum zero-throughput duration (ms), and recovery time (ms).

9. Test Category 5: Request Routing and Load Balancing

Inference serving introduces application-layer routing decisions that interact with fabric-layer load balancing (ECMP, flowlet, packet spray).

9.1. KV-Aware Request Routing Efficacy

Objective: To measure the effectiveness of KV-aware request routing, where the request router considers decode worker KV cache memory occupancy and fabric path congestion when assigning requests.

Procedure: Configure a request router with KV-aware routing enabled. Submit a sustained request stream at rates of 10, 50, 100, and 200 req/s. Compare against round-robin routing (baseline).

Measurement: Report the coefficient of variation (CV) of decode worker memory utilization, P99 TTFT, P99 ITL, KV cache eviction rate, and Goodput for both KV-aware and round-robin routing.

9.2. Prefix-Aware Cache Hit Rate

Objective: To measure the fabric bandwidth savings achieved by prefix-aware caching, where requests with common prefixes are routed to workers that already hold the corresponding KV cache segment.

Procedure: Generate a request workload where P% of requests share a common prefix of L tokens (P = 25%, 50%, 75%, 90%; L = 256, 512, 1024, 2048). Compare against non-prefix-aware routing.

Measurement: Report cache hit rate (%), fabric bandwidth reduction (%), TTFT reduction (ms), and TPS improvement (%) for each (P, L) combination.

9.3. ECMP and Dynamic Load Balancing Under Inference Traffic

Objective: To evaluate fabric-layer load balancing effectiveness under inference traffic patterns characterized by a mix of large KV cache flows and small EP dispatch flows.

Procedure: Measure link utilization uniformity under: (a) KV cache transfers only (large flows, 16 MB+), (b) EP AllToAll dispatches only (small flows, < 1 MB), (c) mixed KV cache and EP traffic.

Measurement: Report JFI, maximum link utilization (%), minimum link utilization (%), and the oversubscription ratio for each scenario and load balancing algorithm.

9.4. Jain Fairness Index for Decode Worker Utilization

Objective: To measure how evenly the fabric distributes KV cache transfer load across decode workers.

Procedure: With N_D decode workers (N_D = 8, 16, 32, 64), submit a sustained request stream and measure per-worker KV cache receive rate, GPU utilization, and output TPS.

Measurement: Report JFI for KV cache receive rate, GPU utilization, and output TPS. Report the max/min ratio for each metric.

10. Test Category 6: Latency Benchmarks

Inference latency is the primary user-facing quality metric. This section defines benchmarks that isolate the fabric’s contribution to end-to-end inference latency.

10.1. TTFT Under Varying Prompt Lengths

Objective: To characterize TTFT as a function of prompt length, isolating the fabric-dependent KV cache transfer component.

Procedure: Submit single requests (no concurrent load) with prompt lengths of 128, 256, 512, 1024, 2048, 4096, 8192, and 16384 tokens. Measure TTFT and decompose into T_prefill, T_transfer, and T_decode_init. As a refernce the following table is provided.

Config ID	Model Profile	S_KV @ 4K ctx	S_KV @ 32K ctx	S_KV @ 128K ctx
CFG-A	Small: L=32, H_kv=8 (GQA), D=128, BF16	0.25 GB	2.0 GB	8.0 GB
CFG-B	Mid: L=80, H_kv=8 (GQA), D=128, BF16 (e.g., Llama-3 70B)	1.3 GB	10.5 GB	42.0 GB
CFG-C	Large MHA: L=96, H_kv=64 (MHA), D=128, BF16	12.3 GB	98.6 GB	>300 GB
CFG-D	Mid INT8: L=80, H_kv=8 (GQA), D=128, INT8 (quantized)	0.67 GB	5.4 GB	21.5 GB
CFG-E (custom)	Implementer-defined: L=*_, H_kv=_*, D=*_, P=_*	Computed	Computed	Computed

Table 8: Reference Configuration Matrix

Measurement: Report TTFT, T_transfer, and T_transfer/TTFT at P50, P95, P99 for each prompt length. The test MUST be repeated a minimum of 100 times per prompt length

Reporting Format: Results MUST specify the configuration ID (CFG-A through CFG-E) or provide complete values for L, H_kv, D, C, and P_bytes for any test that specifies KV cache message sizes. Results SHOULD be reported as a line graph with prompt length on the X axis and TTFT (ms) on the Y axis, with separate lines for P50 and P99. The T_transfer component SHOULD be shown as a shaded region.

10.2. ITL Characterization and Tail Latency

Objective: To characterize inter-token latency distribution and identify fabric-induced tail latency during the decode phase.

Procedure: Submit a single long-output request (e.g., 2048 output tokens) and record the timestamp of each emitted token. Repeat under: (a) unloaded fabric, (b) loaded fabric (50% of capacity), and (c) heavily loaded fabric (90% of capacity plus concurrent EP dispatches).

Measurement: Report ITL at P50, P95, P99, P99.9, and maximum for each load condition. Report the number of tokens exhibiting ITL > 100 ms (stall events). The test MUST generate at least 10,000 ITL samples per condition.

10.3. End-to-End Latency Under Multi-Tenant Load

Objective: To measure inference latency when multiple models or model instances share the same fabric.

Procedure: Deploy two or more model instances on separate worker pools sharing the same fabric. Submit requests to both instances concurrently.

Measurement: Report per-instance TTFT P99, ITL P99, and the interference penalty: (multi-tenant metric - single-tenant metric) / single-tenant metric * 100%.

10.4. Latency Sensitivity to Fabric Congestion

Objective: To establish the relationship between fabric congestion level and inference latency degradation.

Procedure: Inject controlled background traffic on the fabric at levels from 0% to 95% of capacity in 5% increments. At each level, submit inference requests at a fixed rate and measure TTFT and ITL.

Measurement: Report TTFT P99 and ITL P99 as functions of background traffic level. Identify the inflection point at which latency begins to degrade significantly. Report the latency degradation factor at 50%, 75%, and 90% background load.

11. Test Category 7: Throughput Benchmarks

Inference throughput determines the cost-effectiveness of the serving deployment.

11.1. Aggregate Tokens Per Second

Objective: To determine the maximum sustained aggregate TPS achievable while meeting latency SLOs.

Procedure: Increase the request arrival rate from 1 req/s to the point where either TTFT P99 exceeds 500 ms or ITL P99 exceeds 50 ms. At each rate, measure TPS_output, TPS_input, Goodput, and all latency KPIs.

Measurement: Report TPS_output, TPS_input, Goodput, TTFT P99, ITL P99, and fabric utilization at the SLO-bounded throughput. Report the fabric utilization at the SLO boundary as a key efficiency metric.

11.2. Batch Size Scaling and Continuous Batching Impact

Objective: To measure the interaction between inference batch size, continuous batching, and fabric transfer patterns.

Procedure: Configure the serving system with varying maximum batch sizes (1, 4, 8, 16, 32, 64, 128). For each batch size, measure: (a) the number of concurrent KV cache transfers, (b) aggregate fabric bandwidth consumed, (c) TPS_output, and (d) TTFT P99. Enable continuous batching and repeat.

Measurement: Report TPS_output, TTFT P99, fabric bandwidth (GB/s), and peak concurrent transfers for each batch size, with and without continuous batching.

11.3. Goodput Under Preemption and Eviction

Objective: To measure the Goodput loss when fabric congestion forces KV cache eviction or request preemption.

Procedure: Oversubscribe the system beyond the SLO-bounded throughput (at 110%, 125%, 150%, and 200% of the rate found in Test 11.1). Measure the rate of KV cache evictions, request preemptions, and the resulting Goodput reduction.

Measurement: Report Goodput, eviction rate (evictions/s), preemption rate (preemptions/s), wasted fabric bandwidth (GB/s), and the Goodput/TPS_output ratio (efficiency).

12. Test Category 8: Scale and Autoscaling

Inference serving clusters must scale dynamically to match request demand.

12.1. Fabric Scale Limits for Inference Clusters

Objective: To determine the maximum inference cluster size supportable by the DUT fabric while meeting performance requirements.

Procedure: Progressively scale the cluster from a minimal configuration (e.g., 2 nodes, 16 GPUs) to the fabric's capacity (e.g., 1024 nodes, 8192 GPUs). At each scale point (following powers of two), measure KV cache transfer throughput and latency, EP AllToAll dispatch latency, fabric control plane convergence time, routing table size, and end-to-end TTFT and TPS.

Measurement: Report all KPIs at each scale point. Identify the scale limit as the point where any KPI degrades by more than 10% from the minimal-configuration baseline.

12.2. Dynamic Autoscaling Response Time

Objective: To measure the time required for the fabric to accommodate dynamic scaling of inference worker pools (adding/removing prefill or decode workers).

Procedure: Starting from a stable serving state, trigger a scale-up event (e.g., adding 4 decode nodes). Measure: (a) fabric convergence time, (b) time from fabric convergence to first KV cache transfer on new nodes, (c) time to reach steady-state throughput. Repeat for scale-down events.

Measurement: Report fabric convergence time (ms), first-transfer time (ms), and time to steady-state (ms) for scale-up and scale-down events. Report any packet loss or latency spikes during the scaling transition.

12.3. Link Failure Convergence Impact on Serving

Objective: To measure the impact of fabric link failures on inference serving performance and the convergence time to restore full service.

Procedure: During sustained inference serving at 80% of SLO-bounded throughput, fail a single fabric link on: (a) a leaf-spine link carrying KV cache traffic, (b) a spine-spine link, (c) a link on the decode worker's leaf switch. Measure traffic disruption and recovery time. Repeat for dual link failures.

Measurement: Report traffic disruption duration (ms), convergence time (ms), TTFT degradation during convergence (ms above baseline P99), TPS reduction during convergence (%), and time to full recovery (ms). The test MUST be repeated a minimum of 20 times per failure scenario.

13. Test Category 9: Soak and Stability

Long-running inference serving deployments must maintain performance without degradation over time.

13.1. 24-Hour Sustained Inference Load

Objective: To verify that the fabric maintains performance under continuous inference serving load for 24 hours.

Procedure: Configure the SUT-E at 80% of the SLO-bounded throughput determined in Test 11.1. Run a continuous request stream for 24 hours with a realistic prompt length distribution. Sample the following metrics every 15 minutes: TTFT P99, ITL P99, TPS_output, KV_xfer_latency P99, fabric link utilization, switch CPU/memory usage, NIC counters (RDMA retransmissions, QP errors), and PFC/ECN event counts.

Measurement: Report the trend of all sampled metrics over the 24-hour period. There SHOULD be zero NIC QP errors, zero routing flaps, and less than 1% variation in TTFT P99 over the test duration.

13.2. KV Cache Memory Leak Detection

Objective: To detect memory leaks in the KV cache management subsystem that may manifest as fabric performance degradation over time.

Procedure: Monitor GPU memory, CPU memory, NIC registered memory regions, and RDMA memory region counts on all prefill and decode workers during the 24-hour soak test. Record the number of active KV cache pages, RDMA memory registrations, and pinned memory at each sampling interval.

Measurement: Report the trend of each monitored metric. Flag any monotonic increase as a potential leak. Report the maximum observed memory usage and the usage at the end of the 24-hour period.

13.3. Long-Running Serving Stability

Objective: To verify that fabric-dependent components remain stable under continuous inference serving.

Procedure: During the 24-hour soak test, monitor: NIC QP state transitions, switch buffer utilization trend, FEC error rate trend, BGP/OSPF adjacency stability, and RDMA retransmission rate. At the 12-hour mark, trigger a controlled perturbation (single link flap) and verify recovery.

Measurement: Report the count of any QP state transitions, maximum switch buffer utilization, FEC error trend, adjacency flap count, and RDMA retransmission count. Report the recovery time from the 12-hour link flap perturbation.

14. Reporting Format

All test results MUST be reported following the conventions established in RFC 2544 Section 26. In addition, the following inference-specific reporting requirements apply:

* *System Configuration Report:* The report MUST include: model name and parameter count, parallelism strategy (TP, DP, EP, PP configuration for both prefill and decode pools), xPyD ratio, inference serving framework name and version, KV cache transfer library name and version, accelerator type and count, NIC type and firmware version, switch ASIC and software version, fabric topology, and link speeds.

* *Workload Characterization Report:* The report MUST include: prompt length distribution (mean, P50, P99, distribution type), output length distribution, request arrival rate and distribution, number of concurrent requests, and prefix sharing percentage.

* *Results Reporting:* For each test, results MUST include: the specific test identifier (e.g., Test 5.1), the DUT/SUT configuration tested, the number of trials, all measured KPI values with confidence intervals, and any anomalies observed.

Report Element	Format	Required?
System Configuration	Structured table per above	Yes (required)
Workload Parameters	Structured table per above	Yes (required)
KPI Summary Table	Table with all measured KPIs	Yes (required)
Latency Distribution Plots	CDF or histogram per test section	Recommended
Throughput vs. Scale Graphs	Line chart per test section	Recommended
Fabric Health Indicators	Table per Section 4.4	Yes (required)
Raw Data Appendix	Machine-readable format (CSV, JSON)	Optional

Table 9: Reporting Format Requirements

15. Security Considerations

This document defines benchmarking methodologies for controlled laboratory testing. All tests MUST be conducted in isolated test environments that are not connected to production networks or the public Internet. The security considerations from [RFC2544] and [RFC6815] apply.

Additionally, implementers SHOULD be aware that RDMA-based KV cache transfer provides direct memory access between hosts; all RDMA connections in the test environment MUST use authenticated QPs where supported. The test results themselves may reveal performance characteristics that could inform denial-of-service attack vectors; results SHOULD be treated as sensitive when applicable.

16. IANA Considerations

This memo includes no request to IANA.

17. References

17.1. Normative References

[RFC1242] Bradner, S., "Benchmarking Terminology for Network Interconnection Devices", RFC 1242, DOI 10.17487/RFC1242, July 1991, <https://www.rfc-editor.org/rfc/rfc1242>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC2544] Bradner, S. and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices", RFC 2544, DOI 10.17487/RFC2544, March 1999, <https://www.rfc-editor.org/rfc/rfc2544>.

[RFC2889] Mandeville, R. and J. Perser, "Benchmarking Methodology for LAN Switching Devices", RFC 2889, DOI 10.17487/RFC2889, August 2000, <https://www.rfc-editor.org/rfc/rfc2889>.

[RFC6349] Constantine, B., Forget, G., Geib, R., and R. Schrage, "Framework for TCP Throughput Testing", RFC 6349, DOI 10.17487/RFC6349, August 2011, <https://www.rfc-editor.org/rfc/rfc6349>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[TERMINOLOGY] Calabria, F., Pignataro, C., Wu, Q., and G. Fioccola, "Benchmarking Terminology for AI Network Fabrics", Work in Progress, Internet-Draft, draft-calabria-bmwg-ai-fabric-terminology-01, 21 April 2026, <https://datatracker.ietf.org/doc/html/draft-calabria-bmwg-ai-fabric-terminology-01>.

17.2. Informative References

[DISTSERVE] Zhong, Y., "DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving", OSDI 2024, 2024.

[EP-COMM] "DeepEP: An Efficient Expert-Parallel Communication Library", 2025.

[K8S-INF] "llm-d: Kubernetes-Native Distributed LLM Inference", 2025.

[LMCACHE] LMCache Project, "LMCache: Hierarchical KV Cache Management for Inference", 2025.

[RFC6815] Bradner, S., Dubray, K., McQuaid, J., and A. Morton, "Applicability Statement for RFC 2544: Use on Production Networks Considered Harmful", RFC 6815, DOI 10.17487/RFC6815, November 2012, <https://www.rfc-editor.org/rfc/rfc6815>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/rfc/rfc7432>.

[SGLANG] "SGLang: Efficient Execution of Structured Language Model Programs", 2024.

[TRAINING-BENCH] "Benchmarking Methodology for AI Training Network Fabrics", Work in Progress, Internet-Draft, draft-calabria-bmwg-ai-fabric-training-bench, 2026, <https://datatracker.ietf.org/doc/html/draft-calabria-bmwg-ai-fabric-training-bench>.

[UEC-SPEC] Ultra Ethernet Consortium, "UEC Specification 1.0", 2024.

[VLLM] Kwon, W., "Efficient Memory Management for Large Language Model Serving with PagedAttention", Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles, 2023.

Appendix A. KPI-to-Test Mapping Summary

The following table provides a cross-reference from each KPI defined in Section 4 to the test(s) in which it is measured.

KPI	Primary Test(s)	DUT/SUT
TTFT	6.1, 6.2, 10.1, 10.3	SUT-E
ITL	10.2, 10.3, 10.4	SUT-E
TPS_output	6.2, 11.1, 11.2, 11.3	SUT-E
TPS_input	11.1	SUT-E
Goodput	11.1, 11.3	SUT-E
KV_xfer_latency	5.2, 5.3, 6.1, 6.4	DUT-N, DUT-PD
KV_xfer_bandwidth	5.1, 5.3, 5.4	DUT-N, DUT-PD
EP_alltoall_latency	7.1, 7.2, 7.3, 7.4	DUT-F
EP_alltoall_bandwidth	7.1, 7.3	DUT-F
Fabric_FCT	5.2, 5.3	DUT-F
Buffer_utilization	8.1, 8.2	DUT-S
ECN_marking_rate	8.1, 8.3	DUT-S
PFC_frame_count	8.2, 8.4	DUT-S
Link_utilization	5.3, 9.3, 12.1	DUT-F
Packet_drop_rate	8.1, 8.2, 12.3	DUT-F
Request_Rate	11.1	SUT-E
Prefix Cache Hit Rate	9.2	SUT-E
JFI (Decode Worker)	9.4	SUT-E

Table 10: KPI-to-Test Mapping

Appendix B. Inference Serving Framework Capability Categories (Informational)

This appendix describes the inference serving framework capability categories relevant to AI fabric benchmarking. This appendix is intended to guide documentation of SUT-E configurations and is NOT normative. Implementers using a Software Workload Emulator (SUT-E tests) SHOULD document which of the following capabilities their serving framework supports.

Capability Category	Description	Relevance to Fabric Benchmarking
Disaggregated Prefill/ Decode (PD)	Physical separation of prefill and decode execution across different accelerator pools	Determines whether DUT-PD topology tests apply (Section 6)
KV Cache Transfer Protocol	Protocol and library used for prefill-to-decode KV state transfer (one-sided PUT, two-sided SEND/RECV, GPU-initiated)	Determines RDMA verb types under test and applicable frame formats (Appendix C)
MoE Expert Parallelism (EP) Support	Distribution of MoE expert sub-networks across GPUs and AllToAll dispatch mode support	Determines whether MoE EP tests apply (Section 7)
Continuous Batching	Dynamic request admission to active inference batches	Affects request arrival rate distributions and load balancing tests (Section 9)
Prefix / KV Cache Sharing	Reuse of KV cache segments for requests with common prefixes	Determines applicability of prefix cache hit rate test (Section 9.2)
RDMA Transport Support	Underlying transport(s) supported: RoCEv2, UET, or other	Must be documented; affects congestion management test interpretation (Section 8)

GPU-Initiated Networking (GIN) Support	Ability for GPU threads to directly initiate RDMA operations without CPU involvement	Affects RDMA primitive choice in MoE dispatch tests (Section 7)
Kubernetes / Orchestration Integration	Native support for container-based deployment and horizontal scaling	Relevant for autoscaling tests (Section 12.2)
Maximum Reported Scale	Maximum cluster scale at which the framework has been validated	Documents applicability of fabric scale tests

Table 11: Framework Capability Categories

NOTE: Implementers MUST document the specific framework name, version, and configuration in all test reports. Results obtained with different frameworks are not directly comparable; framework identity is a required reporting parameter per Section 14.

Appendix C. KV Cache Transfer Frame Format

This appendix defines the reference frame formats for KV cache transfer benchmarking over RoCEv2. The frame format follows the standard RoCEv2 encapsulation with one-sided RDMA WRITE (PUT) operations.

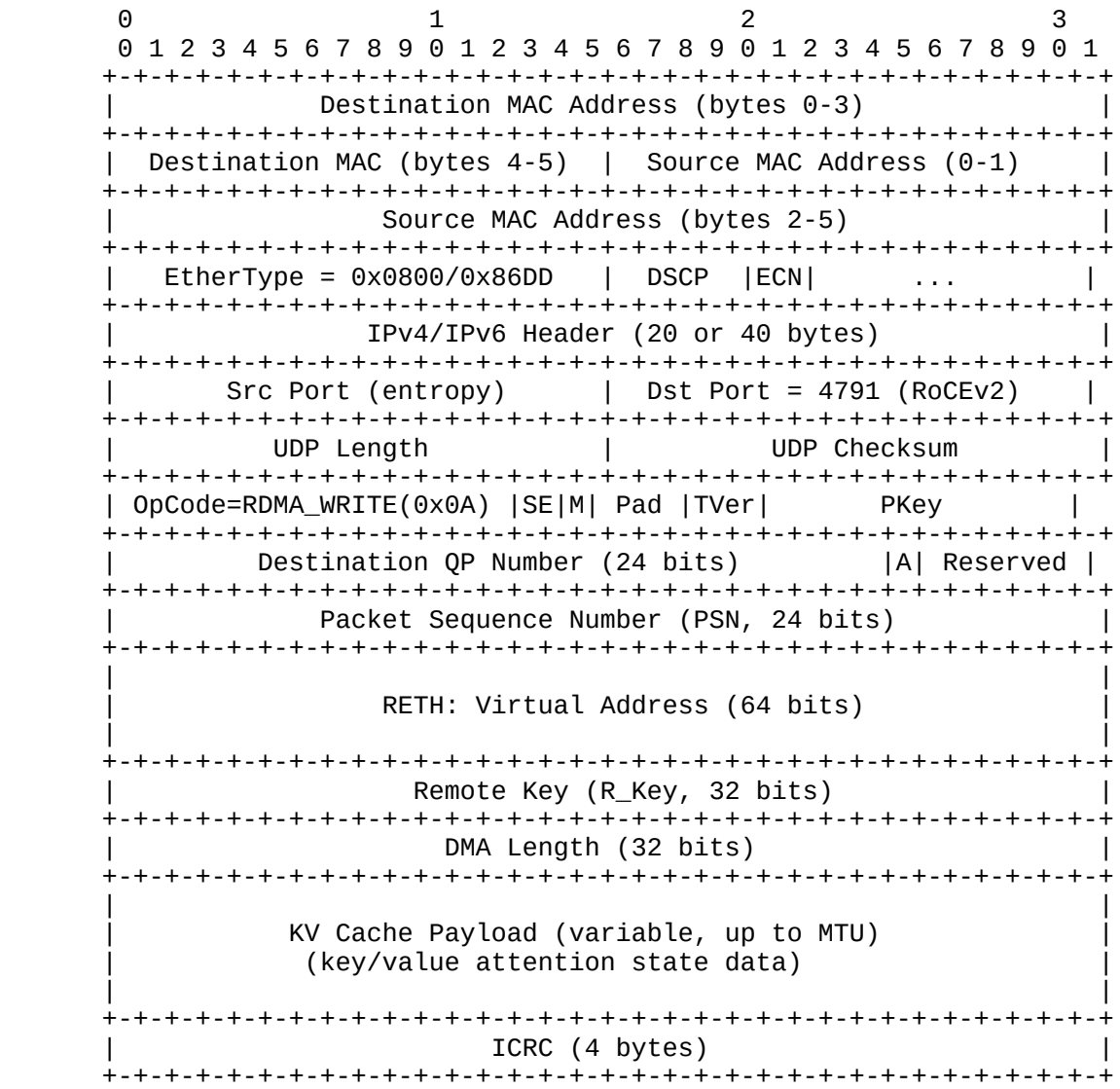

Figure 2: RoCEv2 KV Cache Transfer Frame (One-Sided RDMA WRITE)

Notes: The UDP Source Port SHOULD use entropy-based values for ECMP load distribution. The RETH (RDMA Extended Transport Header) carries the remote virtual address, remote key, and DMA length for the one-sided WRITE operation. For KV cache transfers, the DMA Length field indicates the size of the KV cache block being transferred. Typical MTU for RoCEv2 is 4096 bytes; larger KV cache blocks (e.g., 64 KB pages) are segmented into multiple packets by the NIC. For PUT-with-

signal operations, the last packet in the transfer includes an RDMA WRITE with Immediate Data (OpCode 0x0B) to signal completion to the decode worker.

Appendix D. MoE AllToAll Communication Pattern

This appendix describes the AllToAll communication pattern used for MoE expert parallelism dispatch and its fabric-level traffic characteristics. In a Mixture-of-Experts model with M total experts distributed across N GPUs (each GPU holds M/N experts), a single MoE layer forward pass generates an AllToAll communication pattern where each GPU sends a variable-size payload to every other GPU.

Parameter	Normal Dispatch (Prefill)	Low-Latency Dispatch (Decode)
Batch Size	128 - 512 tokens	1 - 16 tokens
Payload per GPU pair	Variable (depends on routing)	Fixed (padded to max)
Shape Compatibility	Dynamic (symbolic)	Static (CUDA Graph)
QP Parallelism	24 QPs per connection	8 - 16 QPs per connection
RDMA Primitive	Two-sided SEND/RECV or one-sided PUT	One-sided PUT (GPU-direct RDMA, GIN)
GPU Initiation	CPU-initiated or GIN	GIN (device-initiated, GPU-to-NIC direct)
Typical per-dispatch size	1 - 10 MB aggregate	10 KB - 1 MB aggregate
Dispatch Frequency	Once per MoE layer (prefill)	Once per MoE layer per token (decode)
Latency Target	< 1 ms per dispatch	< 200 us per dispatch

Table 12: MoE Dispatch Traffic Characteristics by Mode

For a representative dense MoE configuration (M3: E=256, k=2, H_model=7168, EP=96 across 12 nodes, BF16), the inter-node traffic per MoE layer dispatch using T_dispatch = (B * k * H_model * 2) / N is approximately

* Normal Dispatch (prefill, batch=256): 256 * 2 * 7168 * 2 bytes / 96 GPUs = ~76 KB per GPU pair, ~870 MB aggregate across all pairs.

* Low-Latency Dispatch (decode, batch=8): 8 * 2 * 7168 * 2 bytes / 96 GPUs = ~2.4 KB per GPU pair, ~27 MB aggregate.

With 61 MoE layers and a decode iteration time target of ~30 ms, the decode phase requires 61 AllToAll dispatches within 30 ms, yielding ~2,000 dispatches per second per decode step, consuming approximately 54 GB/s aggregate inter-node bandwidth for the Low-Latency Dispatch path.

Appendix E. Model Architecture Parameters

This appendix provides a sample calculation for the S_KV formula already provided. It's based on a '70B parameter model at FP16 with 4K context' model

Parameter	Symbol	Value	Source
Transformer layers	L	80	Published architecture
KV attention heads (GQA-8)	H_kv	8	H_total=64 / GQA_ratio=8
Per-head dimension	D	128	model_dim(8192) / H_total(64)
Context length	C	4,096	Given
Precision	P_bytes	2	FP16 = 2 bytes/element

Step-by-Step Calculation

S_KV = 2 × L × H_kv × D × C × P_bytes

= 2 × 80 × 8 × 128 × 4,096 × 2

Step 1: 2 × 80 = 160 (K + V tensors × layers)

Step 2: 160 × 8 = 1,280 (× KV heads)

Step 3: 1,280 × 128 = 163,840 (× head dimension)

Step 4: 163,840 × 4,096 = 671,088,640 (× context tokens)

Step 5: 671,088,640 × 2 = 1,342,177,280 bytes

Acknowledgements

Contributions and review are solicited from the BMWG mailing list (bmwg@ietf.org) and the broader AI networking community. The BMWG chairs and Area Director are identified at https://datatracker.ietf.org/group/bmwg/about/.

Authors' Addresses

Fernando Calabria
Cisco
Email: fcalabri@cisco.com

Carlos Pignataro
Blue Fern Consulting
Email: carlos@bluefern.consulting

Qin Wu
Huawei
Email: bill.wu@huawei.com

Giuseppe Fioccola
Huawei
Email: giuseppe.fioccola@huawei.com