

Breaking the Interconnect Bottleneck in LLM Serving via Congestion-Aware Scheduling over Heterogeneous Interconnects

Dong Liu¹, Yanxuan Yu², Chang Liu³, Eric Jiang¹, Tong Geng⁴, Ying Nian Wu¹

¹UCLA, Los Angeles, CA, USA ²Columbia University, New York, NY, USA

³UIUC, Urbana, IL, USA ⁴Rice University, Houston, TX, USA

pikeliu@ucla.edu, yy3523@columbia.edu, tony.geng@rice.edu, ywu@stat.ucla.edu

Abstract—Modern LLM serving clusters expose a stratified fabric of interconnects—NVLink, PCIe, host-memory staging, and InfiniBand/RoCE—that differ not only in bandwidth and latency, but also in reachability, transfer semantics, and shared-resource coupling. Production frameworks nevertheless bind transfers with launch-time, class- and load-oblivious policies, treating these fabrics as interchangeable wires. We instrument a 64-GPU cluster and find that 71% of tail TTFT requests (those at or above the empirical P99) coincide with interconnect congestion while an admissible alternative decision—another RDMA rail, an equivalent decode replica, or host staging—remains underutilized (median residual-capacity ratio $2.4\times$; mean $3.1\times$). We present HICS (Heterogeneous Interconnect Congestion-aware Scheduler), a runtime layer that jointly selects *fabric*, *transport mode*, *placement*, and *path* for slack-tolerant KV transfers and request placement, while coordinating with preconfigured NCCL communicators for latency-critical TP/PP traffic. HICS integrates with vLLM/SGLang via a thin shim (0.73 μ s scheduling latency and $<2\mu$ s including chunk dispatch). On Llama-3-70B and Mixtral-8 $\times$ 7B under realistic traces, it reduces P99 TTFT by 41% and raises sustained output-token throughput by 28% over the best static baseline—gains that shrink under homogeneous multipath alone. Code: <https://github.com/FastLM/CongShed>.

I. INTRODUCTION

Large language models (LLMs) have become load-bearing infrastructure for cloud services ranging from code completion to retrieval-augmented generation (RAG). Meeting service-level objectives (SLOs) at scale requires batching thousands of concurrent sequences across tens to hundreds of GPUs, connected by a rich hierarchy of interconnects. A single H100 SXM node exposes a hierarchy of fabrics: NVLink 4.0 (up to 900 GB/s *aggregate bidirectional* per GPU via NVSwitch), PCIe Gen5 $\times 16$ (≈ 64 GB/s per direction / 128 GB/s *aggregate bidirectional*), host DRAM as a staging tier (and, when present, CXL-attached memory), and InfiniBand/RoCE (400 Gb/s NDR ≈ 50 GB/s peak line rate per direction; measured application goodput is lower—Table I). Heterogeneity is not merely a bandwidth ladder: fabrics differ in *who* they can reach, *how* they move data (GPU-direct vs. host staging vs. memory placement), and *which* hidden resources they share (NVSwitch, PCIe root complex, NIC queues, memory controllers).

Yet today’s LLM serving stacks under-use this hierarchy at runtime. vLLM [1], SGLang [2], and TensorRT-LLM [3] rely on launch-time NCCL topology selection and static endpoint bindings: collectives are not topology-blind, but they are *class- and load-oblivious* once communicators are formed. Three traffic classes compete across the hierarchy: (1) *tensor-parallel (TP) all-reduces* on the decode critical path, (2) *pipeline-parallel (PP) activation sends* between stages, and (3) *KV-cache migrations* under disaggregated prefill/decode [4]. These classes have different deadlines, sizes, destination flexibility, and legal fabrics—yet frameworks flatten them onto a single static route per (src, dst) pair.

Observation 1. Let $\mathcal{T}_{\text{tail}} = \{r : \text{TTFT}_r \geq Q_{0.99}(\text{TTFT})\}$. On our 64-GPU cluster, 71% of requests in $\mathcal{T}_{\text{tail}}$ coincide with at least one inter-GPU link exceeding 85% utilization in the preceding 20ms, while an admissible alternative decision (another rail, replica, or staging tier) carries $<30\%$ load.

Prior congestion control (DCQCN [5], HPCC [6]) sees NIC queues but not LLM traffic classes or NVLink/PCIe coupling. Framework schedulers see request deadlines but not instantaneous fabric load. NCCL/UCX expose topology at startup but lack a runtime policy that reasons about class admissibility and shared-resource contention.

We bridge this gap with HICS, making three contributions:

- 1) **Semantic and resource asymmetry.** We characterize not only load imbalance across links, but also reachability, transfer-mode, and hidden shared-resource asymmetry across NVLink, PCIe, host staging, and RDMA—showing that nominally disjoint transfers often contend on PCIe root complexes and memory controllers (§III–§IV).
- 2) **Fabric-aware transfer abstraction.** We define per-class admissibility, destination/placement flexibility, staging cost, and cross-fabric coupling. Unlike homogeneous multipath routing, HICS jointly selects destination (when replicas exist), placement, transport mode, and physical path (§V).
- 3) **SLO-aware fabric orchestrator.** We implement a userspace shim that controls KV-transfer placement, RDMA rail selec-

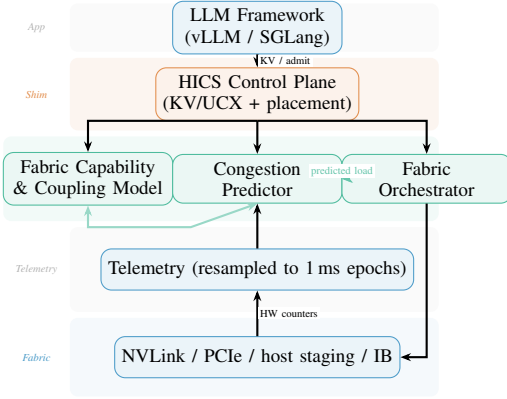


Fig. 1: HICS architecture. The shim controls framework-visible KV transfers, RDMA rail selection, and request/replica placement; latency-critical TP/PP remain on preconfigured NCCL communicators that HICS isolates via admission and KV preemption ($0.73\mu\text{s}$ scheduling latency; $<2\mu\text{s}$ including chunk dispatch).

tion, and request/replica admission, while coordinating with preconfigured NCCL communicators (priority / isolation) for TP/PP—guided by a pluggable short-horizon congestion predictor (§V–§VI).

Our evaluation shows substantial improvements in tail TTFT and sustained output-token throughput with negligible overhead.

II. BACKGROUND

A. LLM Serving Architecture

Autoregressive LLM inference proceeds in two phases. The *prefill* phase processes the entire prompt in a single forward pass, producing a KV cache of shape $[L \times 2 \times H_{\text{kv}} \times S \times D]$ (layers $\times$ K/V $\times$ KV-heads $\times$ sequence $\times$ head-dim). The *decode* phase then generates one token per step, reusing the accumulated cache. Disaggregated serving [4], [7] places prefill on dedicated *prefill nodes* and decode on *decode nodes*, transferring the KV cache across the network after each prefill completes. For Llama-3-70B (80 layers, 8 GQA KV-heads, 128 head-dim, FP16) a single 4096-token sequence produces a $\approx 1.25\text{ GiB}$ (1.34 GB) KV cache ($80 \times 2 \times 8 \times 128 \times 4096 \times 2\text{ bytes}$), demanding low-latency inter-node transfer at the end of every prefill.

Tensor parallelism (TP) shards each weight matrix across T GPUs, requiring an all-reduce after every attention and feed-forward block—up to $2L$ collectives per forward pass. Pipeline parallelism (PP) places consecutive layers on different GPUs, requiring activation tensors of size $[B \times S \times D]$ to be sent between stages at micro-batch boundaries.

B. Heterogeneous Interconnect Hierarchy

Modern GPU servers expose a four-tier interconnect hierarchy:

NVLink (intra-node, GPU–GPU). Each H100 exposes up to 900 GB/s *aggregate bidirectional* NVLink bandwidth through NVSwitch ($\sim 1\mu\text{s}$ switch latency).

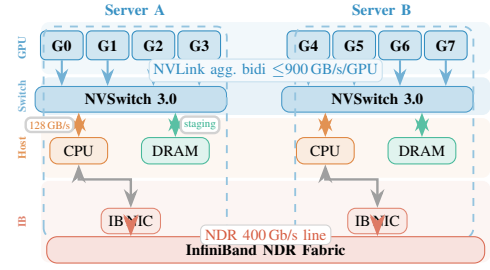


Fig. 2: Interconnect hierarchy on our H100 cluster. Fabrics differ in reachability and semantics: NVLink (GPU-direct collectives), PCIe (shared root / staging), host DRAM (KV placement tier), IB (cross-node RDMA). HICS matches TP/PP/KV to admissible (fabric, mode, placement) decisions—not interchangeable wires.

TABLE I: Fabric rates on our H100 SXM5 cluster (peak vs. measured). NVLink peak is *per-GPU aggregate bidirectional*; measured bulk is unidirectional GPU $\leftrightarrow$ GPU via NVSwitch. IB peak is port line rate; measured is application goodput with GPUDirect RDMA. Host-DRAM staging uses cudaMemcpyAsync over PCIe (not GPU-coherent CXL).

Fabric	Peak	Meas. bulk	Small lat.	Access mode
NVLink 4.0	900 bidi/GPU	~ 420 unidir	$\sim 1\mu\text{s}$	GPU P2P
PCIe Gen5 $\times 16$	64/dir	~ 52 /dir	$\sim 3\mu\text{s}$	host DMA
Host DRAM stage	—	~ 48 /dir	$\sim 8\mu\text{s}^\dagger$	PCIe copy
IB NDR 400G	50/dir	~ 42 /dir	$\sim 5\mu\text{s}$	GPUDirect

[†]DMA setup + first 64 KB; not CXL load latency.

PCIe (intra-node, CPU–GPU / NIC). PCIe Gen5 $\times 16$ provides $\approx 64\text{ GB/s}$ per direction (128 GB/s aggregate bidirectional). Host-staged RDMA, GPU $\leftrightarrow$ CPU copies, and any memory-tier staging share this root complex.

Host / CXL staging tier. We treat host DRAM (and, when available, CXL-attached memory) as a *placement* tier for KV offload/staging, not as a GPU–GPU parallel wire. Our cluster evaluation uses host-DRAM staging with measured DMA paths (Table I); we do *not* claim a production-wide GPU-coherent CXL 3.0 memory pool on stock H100 SXM nodes.

InfiniBand / RoCE (inter-node). NDR 400 Gb/s ports have $\approx 50\text{ GB/s}$ peak line rate per direction; measured bulk goodput on our fabric is lower (Table I). Cross-node KV migrations and multi-node TP/PP groups use this fabric.

Figure 2 depicts this hierarchy. Table I separates peak line rates from measured bulk goodput and small-transfer latency on our platform.

These fabrics are *not* interchangeable weighted edges. NVLink is GPU-direct and shares NVSwitch with TP collectives; IB is cross-node and still traverses PCIe root / NIC queues for many paths; host DRAM is a *placement* substrate, not a GPU–GPU parallel wire; PCIe is both a first-class path and a hidden transit resource. A static launch-time binding therefore fails along two axes: (i) traffic–fabric mismatch (slack-tolerant KV contending with TP), and (ii) cross-fabric coupling (an IB “bypass” that still saturates the same PCIe root).

TABLE II: Representative 60 s window: utilization of two NVSwitch directional ports and one IB port (100 ms buckets). When one resource is hot, an admissible alternative decision is often cool.

Resource	Mean (%)	Peak (%)	Time > 85%
NVLink dir. port 0	53.6	99.4	14.2%
NVLink dir. port 1	48.1	96.8	9.1%
IB port / rail 0	50.7	98.1	11.6%

III. MOTIVATION AND MEASUREMENT

A. Measurement Setup

We instrument a 64-GPU cluster (8 nodes $\times$ 8 H100 SXM5 GPUs, NDR InfiniBand, NVLink 4.0/NVSwitch) with hardware counters at their supported rates—NVLink via `nvmlDeviceGetNvLinkUtilizationCounter` (≈ 1 ms hardware update granularity; polled at 1 kHz), PCIe via `pcm-iio` (1 kHz), and IB via `perfquery` (500 Hz)—then *resample all series into aligned 1 ms scheduling epochs* for the orchestrator. The telemetry daemon uses 0.14 CPU cores with $<0.5\%$ missed epochs at steady state. We serve Llama-3-70B in a TP=4, PP=2 configuration with disaggregated prefill/decode (4 prefill nodes, 4 decode nodes) under a Poisson arrival process with mean rate 8 req/s and prompt lengths from a ShareGPT-derived trace [8].

B. Congestion Is Pervasive and Avoidable

Table II shows that individual resources frequently approach saturation despite moderate average utilization. Across the full 7-day trace:

Observation 2. *When any monitored resource exceeds 85% utilization, the best admissible alternative provides 18.4 GB/s median residual bandwidth versus 7.7 GB/s on the congested resource, corresponding to a $2.4\times$ median ratio ($3.1\times$ mean).*

Observation 3. *Median TTFT is 312 ms; P99 TTFT is 1,840 ms—a $5.9\times$ gap. Of requests in $\mathcal{T}_{\text{tail}}$ ($\text{TTFT} \geq Q_{0.99}$), 71% coincide with link saturation in a 20 ms lookback, 18% with compute queuing only, and 11% with host PCIe contention (categories are assigned by priority when multiple signals fire; we do not claim controlled causal attribution).*

C. Case Study: Anatomy of a P99 Spike

Table III traces one tail request R ($\text{TTFT} = 4,210$ ms vs. P50 = 312 ms) at $\lambda = 8$ req/s under vLLM continuous batching (no chunked prefill; decode step waits for all sequences in the running batch to finish the current iteration). At $t = 142$ ms, prefill completes and the ≈ 1.25 GiB KV is issued to the pinned decode replica over IB rail 0. Seven other prefills finishing within 20 ms create a ~ 10 GB burst on that rail while PP activations sustain ~ 14 GB/s on the same port. Rail 0 saturates at $t = 148$ ms; PP steps S1–S4 stall 210/165/140/105 ms and the decode iteration completes only at $t = 768$ ms. The remaining ≈ 3.4 s is *queue amplification*: R waits behind three full batch drains whose members were themselves delayed by the same stall—not pure wire time.

TABLE III: Case-study timeline for request R (ms). Network stall ends by 768 ms static / 290 ms HICS; most of the static TTFT is decode-queue amplification under continuous batching.

Event	Static	HICS
Prefill complete / KV issued	142	142
IB rail saturates	148	—
PP iteration unblocked	768	290
Decode admission	2,410	620
First token (TTFT)	4,210	1,073

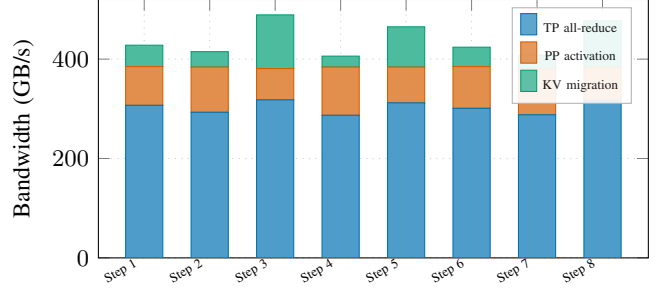


Fig. 3: Per-step bandwidth by traffic class on one GPU's NVLink domain (8 consecutive decode steps). Demand is plotted against the measured ≈ 420 GB/s unidirectional aggregate goodput for that domain (Table I). KV bursts (green) coincide with TP (blue) in steps 3, 5, and 8.

Under HICS, 5 ms lookahead predicts rail 0 saturation; PP stays on the critical rail, four slack-tolerant KV transfers use rail 1 (same-destination multipath) or an alternate decode replica, and remaining KV on shared resources is preempted. PP completes by $t = 290$ ms; R is admitted one batch earlier and reaches first token at 1,073 ms.

IV. PROBLEM CHARACTERIZATION

A. Transfer Taxonomy and Structural Imbalance

Figure 3 shows that KV bursts frequently coincide with TP all-reduces on the same NVLink domain (NCCL operation tagging [9]). TP/PP are fixed and latency-critical, whereas KV is slack-tolerant and destination-flexible; in 37% of observed steps at least one class admitted an alternative *decision*—same-destination multi-rail IB, replica-aware decode destination, or host-DRAM staging—that would have kept demand within capacity, had the runtime reasoned about admissibility and shared-resource load rather than a single static binding.

We classify each transfer t by slack, destination flexibility, and fabric admissibility. Fabrics are not parallel edges on the same (s, d) ; Table IV lists legal (fabric, mode) pairs, and illegal ones are excluded before cost evaluation.

Definition 1 (Admissible Decision Set). $\mathcal{A}(t) = \{(f, m, d', p)\}$ satisfying Table IV and reachability, via (i) same-destination multipath, (ii) replica-aware redirection, or (iii) staged host-DRAM placement—never treating IB as a same-GPU substitute for intra-node NVLink.

TABLE IV: Fabric-mode admissibility (P=preferred, V=valid, S=staging, —=invalid). Intra-node TP stays on NCCL/NVLink.

Traffic class	NVLink	IB/RoCE	PCIe	Host DRAM
Intra-node TP collective	P	—	—	—
Cross-node TP collective	—	P	transit	—
PP activation	P (intra)	V (cross)	S	—
KV migration	V (intra)*	V (cross)	S	S
KV eviction / offload	limited	remote	host DMA	P

*Same-node only; not an IB substitute for G0→G1.

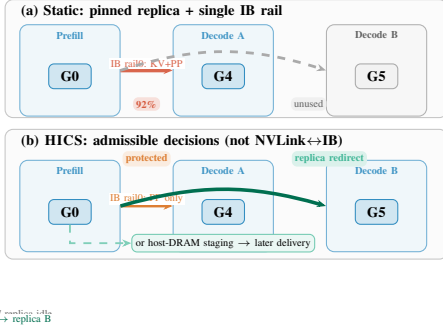


Fig. 4: Static vs. HICS under disaggregated prefill/decode. (a) KV+PP pinned to one rail/replica. (b) Admissible KV decisions (multi-rail, replica redirect, or staging) while PP stays protected—not NVLink G0→G1 over IB.

Static bindings pin KV and PP traffic to one IB rail or replica while other admissible decisions remain idle (Figure 4).

B. Spatial Skew and Congestion Predictability

Load is highly skewed: among 128 logical directional NVLink counters (8×8 GPUs $\times$ 2 aggregated NVSwitch port groups), 12 hot ports (CoV > 1.2) carry 43% of counted NVLink traffic and dominate tail spikes; IB rails are more variable, while PCIe/host-staging paths remain comparatively stable (CoV < 0.5). At ~ 5 ms horizons, KV-TP co-bursts are bimodal and partially predictable (Table V): a compact LSTM halves EWMA MAE on the rising edge, motivating proactive orchestrator inputs as a *pluggable* module—not the core research claim.

V. HICS: SYSTEM DESIGN

A. Architecture Overview

As depicted in Figure 1, HICS is a *heterogeneous fabric orchestrator*, not a generic shortest-path router. HICS directly controls KV migration, RDMA rail selection, decode-replica placement, host staging, and KV preemption. TP/PP remain on preconfigured NCCL communicators; HICS protects them through admission control and by removing competing KV traffic rather than dynamically rewriting intra-node NCCL paths. The orchestrator returns $x_t = (f_t, m_t, d'_t, p_t, q_t)$ for controllable transfers with $0.73 \mu\text{s}$ scheduling latency ($< 2 \mu\text{s}$ including chunk dispatch).

TABLE V: Congestion predictor MAE (held-out traces)

Predictor	MAE (%)	95th-pct Err (%)	Latency
EWMA ($\alpha=0.3$)	9.1	21.4	$< 1 \mu\text{s}$
AR(8)	7.3	17.1	$2 \mu\text{s}$
LSTM (32 units)	4.3	10.7	$8 \mu\text{s}$ (async)
LSTM (128 units)	3.9	9.3	$31 \mu\text{s}$ (async)

B. Fabric Capability and Coupling Model

Endpoints (GPU, CPU socket, host-memory region, IB port) form a digraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, but edges are typed by fabric $f \in \{\text{NVL}, \text{PCIe}, \text{DRAM}, \text{IB}\}$ and are *not* scored with a single scalar latency. Each candidate decision consumes a set of shared resources

$$R(f, p) \subseteq \{\text{NVSwitch}, \text{PCIe root}, \text{NIC}, \text{mem ch}\}. \quad (1)$$

Base fabric costs (serialization + fabric-specific overheads) are:

$$C_{\text{NVL}} = \ell_{\text{NVL}} + \frac{\sigma}{B_{\text{NVL}}^{\text{eff}}} + Q_{\text{NVL}} + \phi_{\text{coll}}, \quad (2)$$

$$C_{\text{IB}} = \ell_{\text{NIC}} + \ell_{\text{fab}} + \frac{\sigma}{B_{\text{IB}}^{\text{eff}}} + Q_{\text{NIC}} + Q_{\text{sw}} + T_{\text{reg}}, \quad (3)$$

$$C_{\text{PCIe}} = \ell_{\text{PCIe}} + \frac{\sigma}{B_{\text{PCIe}}^{\text{eff}}} + Q_{\text{root}} + T_{\text{copy}}, \quad (4)$$

$$C_{\text{DRAM}} = T_{\text{place}} + T_{\text{wr}} + T_{\text{rdbk}} + P_{\text{cap}}. \quad (5)$$

Here ϕ_{coll} penalizes NVLink-domain load that contends with in-flight TP; $T_{\text{reg}}/T_{\text{copy}}$ capture GPUDirect vs. host-staged RDMA; C_{DRAM} is a *placement* cost (PCIe DMA write + future readback + capacity pressure), not a GPU-GPU wire cost. (A future CXL Type-3 pool would plug into the same placement term with device-specific T_{place} .) Effective bandwidth B^{eff} uses measured utilization with a fabric-calibrated contention curve from microbenchmarks (sweep injected load; fit residual goodput): NVLink is near-linear until NVSwitch saturation, while IB/PCIe show heavier tail inflation—we encode this as $B^{\text{eff}} = B(1 - \hat{u})^{\alpha_f}$ with $\alpha_{\text{NVL}}=1.4$, $\alpha_{\text{IB}}=1.8$, $\alpha_{\text{PCIe}}=2.1$ on our H100/NDR cluster.

Cross-fabric coupling closes the model. Selecting IB does not escape a congested PCIe root that the NIC still traverses; host-DRAM staging may unload IB but tax PCIe/memory later at decode. The full cost is

$$C(t, x) = C_f(t, x) + \sum_{r \in R(f, p)} \psi_r(L_r + \Delta L_r(t, x)) + \gamma \cdot \text{lateness}(t, x), \quad (6)$$

where L_r is current load on resource r , ΔL_r is the incremental load of decision x , ψ_r is an empirical penalty (piecewise-linear from the same microbenchmarks), and lateness weights TP/PP far above KV. The orchestrator minimizes C over $\mathcal{A}(t)$, not over all topological paths.

C. Prediction-Guided Orchestration

HICS uses a pluggable 5 ms load predictor; the default is a 32-unit LSTM evaluated against EWMA and AR in Table V. Algorithm 1 scores ≤ 6 admissible decisions for controllable KV/placement transfers, applies KV-only striping, and preempts largest-slack KV when TP/PP need a shared resource.

Algorithm 1 HICS Fabric Orchestration

Require: Transfer t : src s , class c , size σ , slack h , dest set $\mathcal{D}$, staging flag; loads L , $\hat{u}$

Ensure: Decision $x^* = (f^*, m^*, d^*, p^*, q^*)$

```
1:  $\mathcal{A} \leftarrow \text{ENUMERATEADMISSIBLE}(t)$   $\{\leq k \text{ candidates}\}$ 
2: if  $|\mathcal{A}| = 1$  then
3:   return the sole decision
4: end if
5:  $x^* \leftarrow \arg \min_{x \in \mathcal{A}} C(t, x)$  {Eq. 6}
6: if  $c = \text{KV}$  and multi-rail in  $\mathcal{A}$  then
7:   Enable striping / chunk QoS on  $x^*$ 
8: end if
9: if critical TP/PP needs shared  $r$  held by KV then
10:   $\text{PREEMPTKVBYSLACK}(r)$ 
11: end if
12: return  $x^*$ 
```

D. Integration with vLLM and SGLang

HICS integrates as (i) a UCX transport / framework KV-transfer hook for migrations and staging, (ii) a request placement sidecar that chooses decode replicas, and (iii) optional `ncclNet_v8` interposition *only for cross-node net transfers* where the framework already uses the net plugin—not as a means to retarget intra-node NVLink collectives. At startup, a 200 ms profiling sweep populates fabric baselines, $R(f, p)$, and admissibility tables. Telemetry runs as a `SCHED_FIFO` thread.

VI. IMPLEMENTATION

HICS comprises 5K+ lines of C++ and 1K+ lines of Rust. Counters are read at device-supported rates and resampled into 1 ms epochs in a NUMA-local lock-free ring buffer (§III). The congestion predictor runs asynchronously (INT8 LSTM, 8 μs /pass); the control plane reads a 1 ms-invalidated cache. KV path control uses a custom UCX `uct_iface` (GPUDirect RDMA vs. host-staged PCIe copy; host-DRAM placement). Cross-node net hooks via `ncclNet_v8` observe/pace overlapping transfers but do not rewrite NVLink rings/trees. Admissible candidates ($k \leq 6$) and $R(f, p)$ are built at startup (~ 200 ms); scoring Equation 6 takes $< 0.3 \mu\text{s}$. Slack-tolerant KV migrations stripe in 256 MB chunks (rebalance every 50 ms or on predicted congestion), with at most four outstanding chunks.

VII. EVALUATION

A. Experimental Setup

We evaluate on 64 H100 SXM5 GPUs across 8 nodes with NDR 400 Gb/s InfiniBand fat-tree (2:1 leaf oversubscription) and NVLink 4.0/NVSwitch within each node, serving Llama-3-70B (TP=4, PP=2, FP16) and Mixtral-8 $\times$ 7B [10] (TP=8, PP=1, FP8). Host DRAM provides a 128 GB/node staging/offload tier (PCIe DMA). Workloads are *ShareGPT* (mean prompt/output 512/256 tokens) and *LongCtx* (prompts up to 16K, outputs up to 2K), with Poisson arrivals $\lambda \in \{4, 8, 12, 16\}$ req/s. Each configuration: 5 runs $\times$ 30 min after 2 min warmup (≥ 12 K completed requests/run at $\lambda=8$); we report the mean and note run-to-run P99 TTFT CoV $< 3.2\%$ at $\lambda=8$.

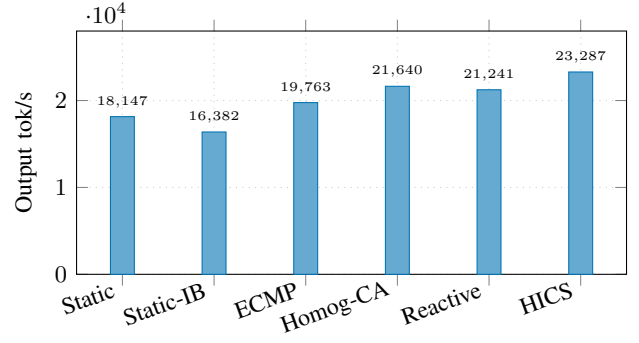


Fig. 5: Maximum sustained *output-token* throughput while measured P99 TTFT stays ≤ 2 s SLO (Llama-3-70B, ShareGPT; cluster aggregate). HICS: 23,287 tok/s (+28% vs. Static); Homogeneous-CA: 21,640 tok/s.

Baselines isolate *path diversity* from *fabric orchestration*: **Static** (framework default bindings), **Static-IB**, **ECMP** (switch ECMP on IB rails), **Homogeneous-CA** (congestion-aware multi-rail IB only—no replica redirect or staging), and **Reactive** (threshold reroute at 80% util, no prediction, no coupling term).

B. End-to-End Latency

HICS outperforms all baselines on P99 TTFT. At $\lambda = 8$ req/s, it achieves 1,073 ms vs. 1,836 ms for Static (**41%** reduction) and 1,417 ms for Reactive (**24%**). Homogeneous-CA reaches only 1,291 ms: multi-rail helps, but most residual gap is closed by replica-aware KV placement, host staging under pressure, and PCIe-root-aware rail choice—i.e., heterogeneous decisions, not extra IB paths alone.

C. Throughput

Figure 5 reports the maximum load point at which the measured P99 TTFT remains ≤ 2 s (the serving SLO), counting *output tokens only* across the cluster (prompt tokens excluded; rejected requests excluded from goodput). HICS achieves 23,287 output-tok/s, **28%** above Static (≈ 364 tok/s/GPU). Static-IB performs worst: forcing traffic through host/NIC paths pays PCIe staging and hurts TP.

D. Ablation: Components and Fabric Heterogeneity

Figure 7 separates mechanism ablations from the heterogeneity claim. Preemption (+31% P99 when removed) protects TP on NVLink/IB critical paths; the predictor (+19%) avoids committing large KV onto fabrics about to saturate; striping (+9%) helps long KV only. Removing the coupling term $\sum_r \psi_r(\cdot)$ (+16%) is telling: IB detours that ignore PCIe-root load recreate congestion on the hidden shared resource. Homogeneous-CA (+20% vs. full HICS) confirms that same-fabric multipath cannot substitute for traffic-fabric matching and staged placement.

Fabric / decision contribution. Disabling host-DRAM staging (KV offload/placement only) raises rejection under memory pressure (LongCtx, $\lambda=12$) from 3.8% to 17.6%, with mild mean-TTFT impact when GPU memory is ample—staging

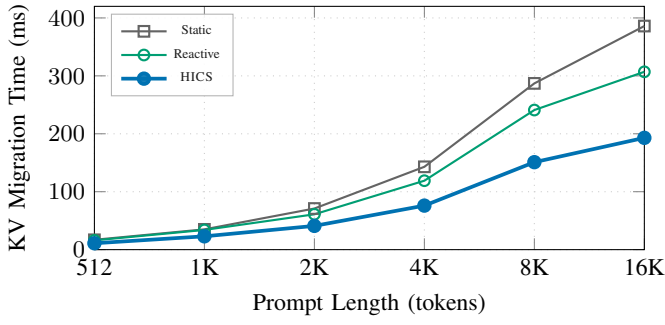


Fig. 6: KV cache migration time vs. prompt length (LongCtx trace). Multi-rail IB striping in HICS halves migration time for prompts $\geq 4K$ tokens (same-destination multipath—not NVLink+IB to one GPU).

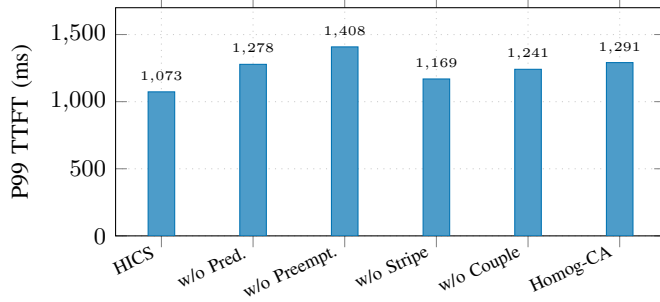


Fig. 7: Ablations at $\lambda = 8$ (Llama-3-70B, ShareGPT). Predictor, preemption, striping, and the coupling term each matter; Homogeneous-CA shows path diversity alone is insufficient.

helps capacity, not as a parallel NVLink path. Under disaggregation, IB remains mandatory for cross-node KV; orchestration then reduces to rail choice, replica selection, staging, and preemption.

E. Overhead Analysis

TABLE VI: HICS Runtime Overhead

Component	Latency	CPU Overhead
Shim intercept + path lookup	0.73 μ s	0.07 cores
Predictor (cached)	0 μ s	0.12 cores
Telemetry daemon	—	0.14 cores
Multi-path chunk dispatch	1.18 μ s	0.05 cores
Total	<2 μ s/op	0.38 cores

Table VI shows 0.73 μ s scheduling latency (<2 μ s/op including chunk dispatch) and 0.38 CPU cores on a 96-core host; GPU compute utilization drops <0.4%.

F. When Heterogeneity Does Not Help

HICS is not uniformly beneficial; we document the failure modes. (1) *No admissible alternative*: single-rail IB, singleton decode replica, no staging—orchestration degenerates to KV preemption only; P99 gain shrinks to 9–12%. (2) *Low load*

($\lambda \leq 4$): Static already meets the 2 s SLO; HICS is within 3% on P99 (overhead-dominated). (3) *Predictor miss*: oracle future utilization recovers an extra $\sim 4\%$ P99; EWMA loses 19% (Figure 7). (4) *Trade-offs*: replica/rail steering increases bytes on cool IB rails by 11–18% while cutting hot-rail time-above-85% by 2.3 $\times$; host staging raises PCIe traffic and can hurt if the coupling term is disabled—hence Equation 6 is mandatory.

VIII. RELATED WORK

Network congestion control and multipath. DCQCN [5] and HPCC [6] regulate NIC rates without LLM class semantics or GPU-fabric state. Bolt [11] provides sub-RTT congestion control but remains unaware of LLM traffic semantics and GPU-fabric state. Homogeneous multipath (ECMP, multi-rail) increases path count; it does not encode host-DRAM placement, NVLink-collective coupling, or PCIe-root sharing. HICS targets that cross-fabric decision space.

KV cache management. FlexGen [12] and InfiniGen [13] offload KV to CPU/SSD under memory pressure. TinyServe [14] selects query-relevant KV pages at decode time. These work studied how to decide *which* or *where* KV caches to be kept or evicted; while HICS studies an orthogonal direction of *how* to transfer them across heterogeneous fabrics with efficiency.

Disaggregated LLM serving. DistServe [4] and Mooncake [15] optimize prefill-decode split for throughput, assuming static high-bandwidth routes. CXL-SpecKV [16] disaggregates speculative KV onto CXL/FPGA memory; KVLearn [17] learns which KV to retain under disaggregation. Neither selects fabric or path at runtime. HICS closes the remaining gap: class-aware transfer under heterogeneous interconnect congestion.

IX. CONCLUSION

We presented HICS, a fabric orchestrator that treats NVLink, PCIe, host-DRAM staging, and RDMA as semantically distinct resources—differing in reachability, transfer mode, placement, and shared-resource coupling—rather than as weighted edges in a homogeneous multipath graph. Measurements show most tail TTFT requests coincide with avoidable congestion under static, class-oblivious binding. HICS controls KV placement/rails and request admission while coordinating with preconfigured NCCL TP/PP communicators. On production-scale models and traces, it cuts P99 TTFT by up to 41% and raises output-token throughput by up to 28%; ablations show the gains require heterogeneous decisions, not path diversity alone.

As LLMs scale, congestion-aware scheduling over heterogeneous interconnects will be critical for sustainable AI infrastructure, and our work delivers a cost-efficient, class-aware fabric orchestrator for next-generation datacenter LLM serving.

ACKNOWLEDGMENTS

Y. W. is partially supported by NSF DMS-2415226, DARPA W912CG25CA007 and research gift funds from Amazon and Qualcomm.

REFERENCES

- [1] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. E. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with PagedAttention," in *Proc. 29th ACM Symposium on Operating Systems Principles (SOSP)*, 2023, pp. 611–626.
- [2] L. Zheng, L. Yin, Z. Xie, C. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez *et al.*, "Sglang: Efficient execution of structured language model programs," *Advances in neural information processing systems*, vol. 37, pp. 62 557–62 583, 2024.
- [3] NVIDIA Corporation, "TensorRT-LLM: Accelerating large language model inference," <https://github.com/NVIDIA/TensorRT-LLM>, 2024.
- [4] Y. Zhong, S. Liu, J. Chen, J. Hu, Y. Zhu, X. Liu, X. Jin, and H. Zhang, "DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving," in *Proc. 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024, pp. 193–210.
- [5] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale RDMA deployments," in *Proc. ACM SIGCOMM*, 2015, pp. 523–536.
- [6] Y. Li, R. Miao, H. H. Liu, Y. Zhuang, F. Feng, L. Tang, Z. Cao, M. Zhang, F. Kelly, M. Alizadeh *et al.*, "Hpcc: High precision congestion control," in *Proceedings of the ACM special interest group on data communication*, 2019, pp. 44–58.
- [7] P. Patel, E. Choukse, C. Zhang, A. Shah, Í. Goiri, S. Maleki, and R. Bianchini, "Splitwise: Efficient generative LLM inference using phase splitting," in *Proc. 51st International Symposium on Computer Architecture (ISCA)*, 2024, pp. 118–132.
- [8] ShareGPT, "ShareGPT conversation dataset," Hugging Face dataset, 2023, accessed July 2026. Available: https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered.
- [9] NVIDIA Corporation, "NCCL: NVIDIA collective communication library," <https://github.com/NVIDIA/nccl>, 2023.
- [10] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M.-A. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mixtral of experts," *arXiv:2401.04088*, 2024.
- [11] S. Arslan, Y. Li, G. Kumar, and N. Dukkupati, "Bolt: Sub-RTT congestion control for ultra-low latency," in *Proc. 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023, pp. 219–236.
- [12] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré, I. Stoica, and C. Zhang, "FlexGen: High-throughput generative inference of large language models with a single GPU," in *Proc. 40th International Conference on Machine Learning (ICML)*, 2023, pp. 31 094–31 116.
- [13] W. Lee, J. Lee, J. Seo, and J. Sim, "InfiniGen: Efficient generative inference of large language models with dynamic KV cache management," in *Proc. 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024, pp. 155–172.
- [14] D. Liu and Y. Yu, "Tinyserve: Query-aware cache selection for efficient llm serving," in *Proceedings of the 33rd ACM International Conference on Multimedia*, ser. MM '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 12 529–12 537. [Online]. Available: <https://doi.org/10.1145/3746027.3758181>
- [15] R. Qin, Z. Li, W. He, M. Zhang, Y. Wu, W. Zheng, and X. Xu, "Mooncake: A KVCache-centric disaggregated architecture for LLM serving," *arXiv:2407.00079*, 2024.
- [16] D. Liu and Y. Yu, "CXL-SpecKV: A disaggregated FPGA speculative KV-cache for datacenter LLM serving," in *Proceedings of the 2026 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '26)*. Seaside, CA, USA: Association for Computing Machinery, Feb. 2026, pp. 56–66, 11 pages. [Online]. Available: <https://doi.org/10.1145/3748173.3779188>
- [17] D. Liu, Y. Yu, E. Jiang, S. Wang, and Y. N. Wu, "To keep or not to keep: Learning KV cache retention in disaggregated LLM serving systems," in *Proceedings of the 19th ACM International Systems and Storage Conference (SYSTOR '26)*. Raanana, Israel: Association for Computing Machinery, Sep. 2026.