When the LLM-Tuned Stack Misses: An Infrastructure View of Biological Foundation Model Inference Across NVIDIA and AMD

Manpreet Singh
Embedded LLM
Singapore

Abstract

AI inference infrastructure is increasingly tuned for one workload shape: dense, large-batch, NVIDIA-resident transformer decoding for LLMs. Biological foundation models, including protein language models, long-context genomic transformers, and structure-conditioned binders, do not fit that shape. They run at small batch sizes over short alphabets, attend over 100K+ token sequences, mix transformer blocks with geometric ops, and are increasingly deployed on whatever GPU is available, including AMD MI300X. We report a measurement study of six widely used biological models (ESM-2, ProtBERT, ProteinMPNN, AlphaGenome, Enformer, DualBind) on three GPUs spanning two vendors (NVIDIA L4, H100; AMD MI300X). We show that 60–80% of reference-implementation runtime is spent outside dense compute (in memory-bound element-wise ops, fragmented launches, and in one case a host–device round-trip the upstream code never removed), and that a single Triton kernel source tree closes most of the gap on both vendors without per-vendor forks. The results surface three infrastructure observations: (i) the dominant bottlenecks on bio workloads are different from those LLM stacks are tuned for; (ii) vendor-stack asymmetries amplify trivial systems bugs into order-of-magnitude regressions; and (iii) write-once cross-vendor portability for non-LLM AI workloads is achievable today with off-the-shelf compiler infrastructure.

1 Introduction

The dominant assumption shaping AI infrastructure today is that the workload is an LLM. Serving stacks, kernel libraries, scheduler heuristics, and vendor performance teams all optimize for dense decoder transformers running at large batch sizes on NVIDIA GPUs [3, 8]. That assumption is increasingly wrong for an important and rapidly growing class of deployments: *biological foundation models*.

These workloads, which include protein language models such as ESM-2 and ProtBERT [5, 6], structure-to-sequence generators like ProteinMPNN [4], long-context genomic transformers such as Enformer and AlphaGenome [1, 2], and structure-conditioned binders like DualBind [7], are the AlphaFold-era backbone of drug discovery and protein engineering pipelines. They are also, from an infrastructure point of view, *not LLMs*: alphabet sizes are 4–20 tokens rather than 100K+; sequence lengths reach 196K nucleotides; deployments run at small batch sizes because each sample is biologically distinct; and the compute is interleaved with non-transformer ops (radial basis functions, virtual-atom geometry, distance matrices). The deployment hardware is also more diverse: bio labs increasingly run on whatever accelerator is available, including AMD MI300X for its 192 GB of unified HBM3.

This paper asks a simple infrastructure question: *what does it cost to serve bio foundation models on today's stack, and where does the cost come from?* We answer it by porting six widely used bio models to fused Triton kernels and measuring them on NVIDIA L4, H100, and AMD MI300X. Our findings reframe the bio-inference problem as a systems-infrastructure problem with three concrete observations.

Observation 1: bio workloads stress different bottlenecks than LLM-tuned stacks were built for. We measure 60–80% of reference-implementation runtime in memory-bound element-wise ops, fragmented kernel launches, and small-batch overhead, not in dense matmul. Vendor LLM kernels do not target these paths because LLM serving amortizes them over much larger batches and longer decode loops.

Observation 2: vendor-stack asymmetry amplifies trivial systems bugs. Our most dramatic measured speedup (38.8× on DualBind, AMD MI300X) is *not* a kernel-engineering win. It comes from removing a per-sample host–device round-trip in the upstream reference implementation. The same fix on H100 yields only 4.2×, because NVIDIA's PCIe path and prefetcher hide the overhead. This is not a bio anomaly: it is what happens when the reference implementation was written and tuned on one vendor's stack and deployed on another's.

Observation 3: write-once cross-vendor portability for non-LLM AI workloads is achievable today. We write a single Triton kernel source tree and run it unmodified on L4, H100, and MI300X. Across six bio models we measure speedups of 1.51×–43.1× over the reference implementations, with cosine similarity ≥ 0.99999 to the baseline outputs.

We frame this as an infrastructure paper rather than an ML one. The contribution is not new models or new fused-attention algorithms; it is the measurement that today's AI

The 3rd Workshop on Hot Topics in System Infrastructure (HotInfra'26), co-located with ISCA'26, June 28, 2026, Raleigh, NC, USA

infrastructure has a workload shape it was not built for, that the gap is large, and that the cost to close it is much lower than the field assumes.

2 Why Bio Workloads Are Different

We profile the reference PyTorch implementations of all six models with the PyTorch profiler and `rocprof`/`nsys` on the respective vendor stacks. Across the six, three workload properties recur and together explain why LLM-tuned infrastructure underserves them.

Small alphabets, large activations. Protein models tokenize over 20 amino acids; genomic models over 4–5 nucleotides. Embedding and unembedding kernels in stock implementations are written as general GEMMs over the full vocabulary. For ESM-2 this means embedding lookup and the final projection together consume a measurable fraction of forward-pass time despite the trivial vocabulary. LLM-tuned kernels are specialized for 50K+ vocabularies in the opposite direction (large matmul, custom sampling); they leave the small-vocabulary case unoptimized.

Very long sequences at batch size one. Enformer accepts 196K-base genomic inputs; AlphaGenome attends across genomic windows that exceed typical LLM context. Bio deployments rarely batch these because the next sample is a different gene. The result: attention is memory-bandwidth-bound rather than compute-bound, and the standard NVIDIA FlashAttention tile sizes (tuned for shorter contexts and larger batches) do not fit the activation footprint. We measured the default attention path stalling on shared-memory pressure on H100 at 196K tokens.

Non-transformer compute on the critical path. ProteinMPNN's runtime is dominated not by attention but by radial-basis-function distance encoding, virtual-C_β computation, and pairwise distance matrices over residues. DualBind interleaves a binding-energy computation between transformer passes. These ops are not present in LLM serving stacks at all, so vendor optimization effort never reaches them. They show up in our profiles as long sequences of small launches, exactly the launch-overhead-dominated regime LLM kernels are designed to avoid by having no such gaps in the first place.

Together, these three properties move the bottleneck off the dense GEMM path that vendor libraries are tuned for, and onto a combination of memory-bound elementwise work, atypical attention shapes, and non-transformer compute. This is the workload shape mismatch we quantify next.

3 Measurement Setup

Hardware. NVIDIA L4 (24 GB GDDR6), NVIDIA H100 SXM 80 GB HBM3, AMD Instinct MI300X (192 GB HBM3, VF partition with 205.8 GB visible). All three are widely deployed in bio-ML clusters today.

Baselines. For each model, the baseline is the official Hugging Face or vendor-released PyTorch implementation, run on the respective vendor's current stack (CUDA 12.x with cuDNN/PyTorch 2.x on NVIDIA; ROCm 6.x with PyTorch 2.x on AMD). For Enformer we use the widely cited Lucidrains PyTorch port at the full 196K-base length.

Kernel implementation. A single Triton [8] source tree, no per-vendor forks. Mixed precision (FP16/BF16) with selective FP32 accumulators for softmax and LayerNorm statistics.

Fidelity gate. Every kernel must achieve either (i) cosine similarity ≥ 0.99999 to the baseline output, (ii) exact-match top-1 discrete predictions, or (iii) zero numerical difference to 6 decimal places. Kernels that fail this gate are excluded.

4 Findings

Table 1. Per-model headline measurements. Speedup is over the reference PyTorch implementation on the listed GPU. Fidelity is the metric used to gate kernel inclusion (see §3).

Model	Hardware	Speedup	Fidelity
ESM-2 (8M)	NVIDIA L4	43.1×	cos. >0.99999
ProtBERT	AMD MI300X	3.6×	top-1 exact (12/12)
ProteinMPNN	NVIDIA L4	1.51×	exact sequence
AlphaGenome	NVIDIA H100	5.05×	cos. 0.99999997
Enformer	NVIDIA H100	1.82×	cos. 0.99999825
Enformer	AMD MI300X	1.43×	cos. 0.99999967
DualBind	AMD MI300X	38.8×	exact (6-dec)
DualBind	NVIDIA H100	4.22×	exact (6-dec)

Headline numbers. Table 1 reports per-model results. The range (1.51×–43.1×) is wide and the source of the wins differs by regime; we unpack three.

4.1 Regime A: memory-bound, fusion-shaped (Meta ESM-2, ProtBERT, Google DeepMind AlphaGenome)

For the three transformer-heavy models, the wins (43.1×, 3.6×, 5.05×) come from fusing QKV projection, bias, and post-attention LayerNorm into a single kernel, plus a FlashAttention-style attention path. The largest ESM-2 win (74.7× on long sequences) coincides with the regime where the baseline activation no longer fits in cache and IO-aware tiling pays off most. The pattern is well known in LLM serving; the infrastructure finding is that on bio inputs (short alphabet, small batch, mixed sequence lengths) it is *not on by default* in the reference implementations practitioners run, and the gap is order-of-magnitude.

4.2 Regime B: non-transformer geometric compute (ProteinMPNN)

ProteinMPNN's 1.51× is the smallest win in Table 1 and the most informative one. The model spends most of its time in

RBF distance encoding and virtual-C_β geometry: ops with no vendor library equivalent, so the ceiling is set by what a hand-written fused kernel can do. The infrastructure implication: as bio models move further from the transformer template, the headroom available to general-purpose AI infrastructure shrinks sharply.

4.3 Regime C: systems bugs amplified by vendor asymmetry (Nvidia DualBind)

The DualBind result is the most important one for this audience, and the one most likely to be misread without care. The 38.8× on MI300X is *not* driven by a more aggressive kernel than the H100 path: the kernel files are identical between vendors. During profiling we found the upstream DualBind reference was not GPU-resident end-to-end. It performed a per-sample host–device round-trip between the transformer pass and the binding-energy computation. On H100, NVIDIA's PCIe controller, unified-memory prefetcher, and CUDA runtime mask much of this cost (the same baseline runs at 4.22× headroom). On MI300X the round-trip is exposed, and 800 samples take 41.3 s instead of 1.06 s.

We flag this explicitly rather than burying it because it is the central infrastructure lesson. *The same upstream code can have order-of-magnitude different performance on different vendors not because the hardware differs by 10× but because the host–device path differs by 10× in how forgivingly it hides bad code.* Bio reference implementations are overwhelmingly developed on NVIDIA; when deployed on AMD, latent systems bugs become first-order. A fused, GPU-resident kernel masks this on both vendors and brings them within ~1.4× of each other in absolute throughput.

4.4 Cross-vendor portability, measured

The Enformer row pair in Table 1 is the cleanest portability evidence. We run the same Triton source files on H100 and MI300X at the full 196K-base sequence length, 25 trials per platform. Absolute baseline times are 0.933 s (H100) and 1.611 s (AMD); optimized times are 0.513 s and 1.125 s. Same source files, no ifdefs, no per-vendor tiles. The speedup asymmetry (1.82× vs. 1.43×) is a property of the baseline: H100 is slower in absolute terms but had less of the unfused overhead our kernel removes, leaving less relative headroom. We treat this as a representative result for what write-once cross-vendor performance looks like on a non-LLM AI workload today: not identical speedups, but the same kernel running, with measurable wins, on both vendors.

5 Discussion

Implications for AI infrastructure. The dominant optimization effort in AI infrastructure today targets a narrow slice: large-batch LLM decoding on NVIDIA. The measurements in §4 suggest the non-LLM tail leaves large headroom on the table (orders of magnitude in our worst case) at low closure cost (a researcher-engineer-month per model in Triton, no per-vendor forks). Infrastructure planning that assumes "vendor libraries handle it" silently overpays on a growing fraction of real deployments.

Implications for AMD-as-a-second-source. The DualBind result cuts both ways. The 38.8× AMD speedup is not evidence MI300X underperforms: it is evidence that a baseline written on NVIDIA had a latent bug NVIDIA's stack happened to hide. But the bug is real, and teams adopting AMD for bio workloads should expect to find more like it. A Triton-based kernel layer brings cross-vendor parity much closer than reference-implementation numbers suggest.

Limitations. Our six-model selection is biased toward transformer-derived architectures; structure-prediction models with iterative refinement (AlphaFold, ESMFold) and diffusion-based designers are absent and likely have different bottleneck profiles. We benchmark single-GPU inference; multi-GPU sharding and tensor parallelism, which dominate production bio pipelines at scale, are out of scope. All measurements are inference-only; training-time portability across vendors remains open.

Open questions for the workshop. Should bio inference be a first-class workload class in serving stacks, or is the right layer a portable kernel IR? Are vendor-asymmetric systems bugs of the DualBind kind common across other non-LLM AI workloads (climate, materials, medical imaging)?

6 Related Work

FlashAttention [3] established IO-aware attention on NVIDIA, but it is not on by default in the bio reference implementations we measured. Triton [8] provides the cross-vendor kernel substrate we rely on; recent ROCm releases extend Triton to AMD. Bio acceleration work to date has targeted single models [4, 6]; we contribute a cross-model, cross-vendor measurement study framed as a systems infrastructure question. Vendor performance asymmetry has been studied for HPC training; the inference-time asymmetry we observe on real bio code has received less attention.

7 Conclusion

We measured six biological foundation models on three GPUs across two vendors using a single Triton source tree. Bio workloads stress parts of the AI stack that LLM-tuned libraries leave alone; vendor-stack asymmetries turn latent systems bugs into order-of-magnitude regressions when bio code crosses vendors; write-once cross-vendor portability for non-LLM AI is achievable today with off-the-shelf compiler infrastructure.

References

[1] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. 2021. Effective gene expression

prediction from sequence by integrating long-range interactions. *Nature Methods* 18, 10 (2021), 1196–1203.

[2] Žiga Avsec, Natasha Latysheva, Jun Cheng, Guido Novati, Kyle R. Taylor, Tom Ward, Clare Bycroft, Lauren Nicolaisen, Eirini Arvaniti, Joshua Pan, Raina Thomas, Vincent Dutordoir, Matteo Perino, Soham De, Alexander Karollus, Adam Gayoso, Toby Sargeant, Anne Mottram, Lai Hong Wong, Pavol Drotár, Adam Kosiorek, Andrew Senior, Richard Tanburn, Taylor Applebaum, Souradeep Basu, Demis Hassabis, and Pushmeet Kohli. 2026. Advancing regulatory variant effect prediction with AlphaGenome. *Nature* 649, 8099 (2026), 1206–1218. https://doi.org/10.1038/s41586-025-10014-0

[3] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135 [cs.LG] https://arxiv.org/abs/2205.14135

[4] J Dauparas, I Anishchenko, N Bennett, H Bai, RJ Ragotte, LF Milles, BIM Wicky, A Courbet, RJ de Haas, N Bethel, PJY Leung, TF Huddy, S Pellock, D Tischer, F Chan, B Koepnick, H Nguyen, A Kang, B Sankaran, AK Bera, NP King, and D Baker. 2022. Robust deep learning-based protein sequence design using ProteinMPNN. *Science* 378, 6615 (oct 2022), 49–56. https://doi.org/10.1126/science.add2187 Epub 2022 Sep 15.

[5] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. 2021. ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing. https://doi.org/10.1038/s41592-021-01252-x arXiv:2007.06225 [cs.LG] Published 2021 Oct 04.

[6] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan Dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. 2023. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science* 379, 6637 (mar 2023), 1123–1130. https://doi.org/10.1126/science.ade2574 Epub 2023 Mar 16.

[7] Meng Liu and Saee Gopal Paliwal. 2024. DualBind: A Dual-Loss Framework for Protein-Ligand Binding Affinity Prediction. arXiv:2406.07770 [cs.LG] https://arxiv.org/abs/2406.07770

[8] Philippe Tillet, H. T. Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages* (Phoenix, AZ, USA) *(MAPL 2019)*. Association for Computing Machinery, New York, NY, USA, 10–19. https://doi.org/10.1145/3315508.3329973