Tolaga Research
Harness the Power of Intelligence

Analyst Report
November 2026

Optimizing Memory for Fast Inference

Matching inference infrastructure to the workload

Phil Marshall PhD, Chief Research Officer, Tolaga Research



November, 2026

Background

This report uses Tolaga's AI Infrastructure Simulator to investigate how memory capacity, bandwidth, data movement, and computational efficiency affect AI inference performance. Through a series of simulation studies, it explores the benefits and trade-offs of different optimization techniques, highlighting the importance of matching infrastructure architecture to workload requirements.

KEY TAKEAWAYS

- **Faster memory doesn't always mean better inference.** Although the importance of high-performance memory is on the increase, it must be targeted to areas that are memory-bound.
- **Inference prefill and decode commonly have different performance bottlenecks.** In large-scale systems, prefill is typically compute-bound and decode is memory-bound, driving efforts to disaggregate prefill and decode accelerator resources.
- **Uneven data precision and quantization benefits.** An inference workload with approximately one million input context tokens was modeled. Latency reductions were predicted to reach 42.7% for decode, versus just 1.6% for prefill. However, data conversion overheads could erase these gains.

BOTTOM LINE

Fast inference is increasingly about matching architecture to workload, rather than simply deploying faster hardware. The growing diversity of inference workloads makes this an increasingly complex challenge for infrastructure designers and operators.

Matching inference infrastructure to the workload

A technique that doubles the speed of one AI application can leave another almost unchanged. The difference often comes down to where performance is constrained and whether it is constrained by memory, compute or connectivity. For memory its about: how much data must be stored, how quickly it can be accessed, and how far it must travel.

As AI adoption accelerates, operators face growing request volumes, longer contexts, and increasingly complex workloads. Improving performance and infrastructure efficiency requires matching optimization techniques to these different demands.

This report uses Tolaga's AI Infrastructure Simulator to examine how memory capacity, bandwidth, and data movement affect AI inference performance, identifying the benefits and trade-offs of different optimization approaches.

© 2026 Tolaga Research. All rights reserved.



A tale of two phases

Large language model (LLM) inference has two phases – prefill and decode. During prefill, the model processes the input prompt and produces the first output token. During decode, it generates subsequent tokens sequentially. Prefill often places greater demands on compute, while decode is more sensitive to memory bandwidth and data movement.



Two metrics capture these differences: **time to first token (TTFT)** measures how quickly a response begins, while **time per output token (TPOT)** measures how quickly subsequent tokens are generated.

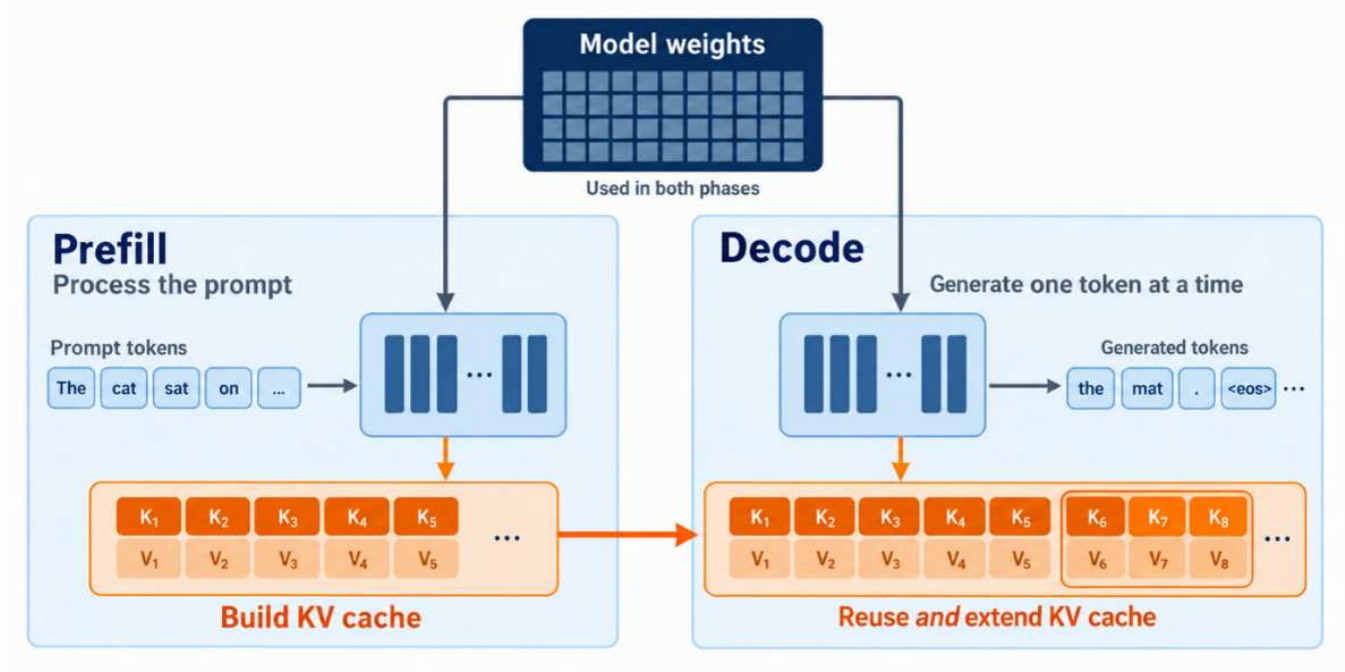
Model weights and the KV cache cale

Two types of data dominate inference memory requirements: model weights and the key/value (KV) cache.

Model weights are the parameters learned during training.

The **KV cache** is created during prefill and extended during decode, and grows with output context length and concurrent requests.

© 2026 Tolaga Research. All rights reserved.



The inference optimization maze

There is no silver bullet for inference optimization. The diversity of AI models and workloads, combined with the growing array of optimization techniques, creates a complex landscape of performance tradeoffs. Navigating this landscape requires understanding the interplay between compute, memory and data movement, as models grow, workloads become more complex, computational, memory and connectivity capabilities become increasingly important determinants of inference performance and reliability.

To better understand data management and memory design trade-offs, this report examines several key design factors through four controlled studies that investigate the following:

1. **Prefill versus decode:** How do computational and memory requirements differ across inference operations?
2. **Data Precision:** How do lower-precision model weights and KV caches affect inference performance?
3. **Arithmetic Precision:** How do arithmetic precision, numerical conversion overhead, and computational efficiency affect inference latency, and why does faster arithmetic not necessarily translate into faster inference?

The studies evaluate memory requirements, time to first token, decode latency, and the architectural trade-offs influencing inference performance.

A note on methodology

The results come from Tolaga's AI Infrastructure Simulator, an equation and flow based analytical model rather than measurements from physical hardware. The simulations isolate architectural and workload effects to provide directional guidance, not predictions of specific commercial platform performance. Validation would require measurements using representative production workloads.

© 2026 Tolaga Research. All rights reserved.

The baseline model

The analysis uses a synthetic one-trillion-parameter Mixture-of-Experts (MoE) model (see our MoE tutorial). The expert model is broadly comparable to Moonshot AI's Kimi K2 in scale and expert configuration, but with several technical differences. In particular, it uses a different layer structure and grouped-query attention (GQA) instead of Multi-head Latent Attention (MLA). The model comprises 96 transformer layers and 386 experts per layer, with 16 activated per token (see our tutorial on model basics).

The model runs on 64 active accelerators within a 72-accelerator scale-up domain, using tensor and expert parallelism. Each accelerator provides 186 GB of high-bandwidth memory and 8 TB/s of nominal memory bandwidth.

Want to know why only 64 instead of 72? See our powers of 2 Research Note.

Simulations cover input contexts from 2K to 1M tokens, batch sizes of 1, 8, and 32, and 512 output tokens per sequence.

Baseline Synthetic Model and System Configuration

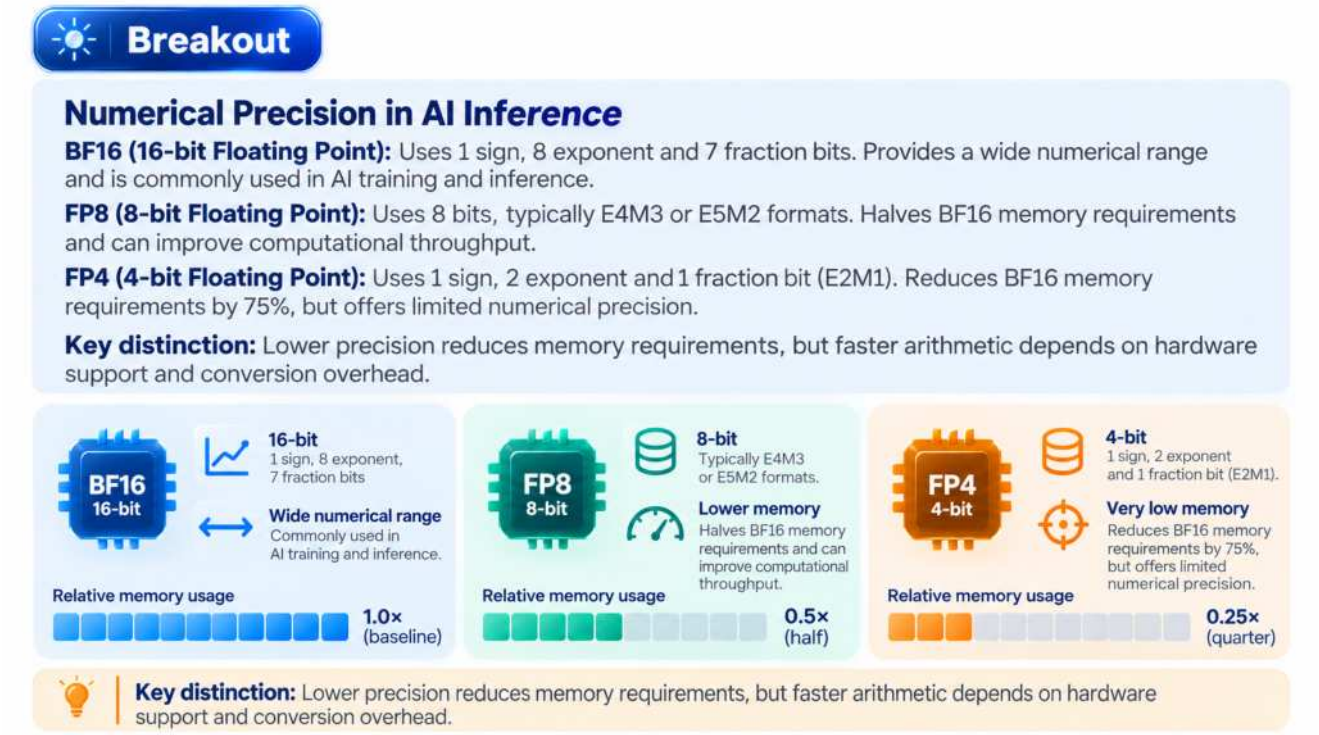
Model Parameter	Configuration
Model Parameters	~1-trillion
Configuration	96 layers; hidden width 7,168
Architecture	386; top-16 routing; expert hidden width 2,432
Experts	64 query heads; eight key/value heads
Attention	Tensor parallelism 8; expert parallelism 8; pipeline/context parallelism 1
Accelerator placement	64 active accelerators in one 72-accelerator local domain; eight unused
HBM capacity per accelerator	186 GB nominal; 167.4 GB usable
HBM bandwidth per accelerator	8,000 GB/s nominal; 6,400 GB/s effective central assumption
Local-link bandwidth	900 GB/s per accelerator, one direction
Nominal compute peaks	BF16: 2,500 TFLOPS/accelerator; FP8: 5,000 TFLOPS/accelerator
Central compute efficiency	50% of the selected nominal peak
Prompt lengths	2K, 32K, 256K and 1M tokens
Concurrent sequences	Batches of 1, 8 and 32
Output length	512 tokens per sequence
Prefill policies	Whole prompt or sequential 1,024-token chunks per sequence

Token counts use binary conventions: K denotes 1,024 tokens and 1M denotes 1,048,576 tokens.
Memory capacity in GB uses decimal bytes. A multiply-add operation is counted as two floating-point operations.

© 2026 Tolaga Research. All rights reserved.

Study 1: Data precision, quantization and inference performance

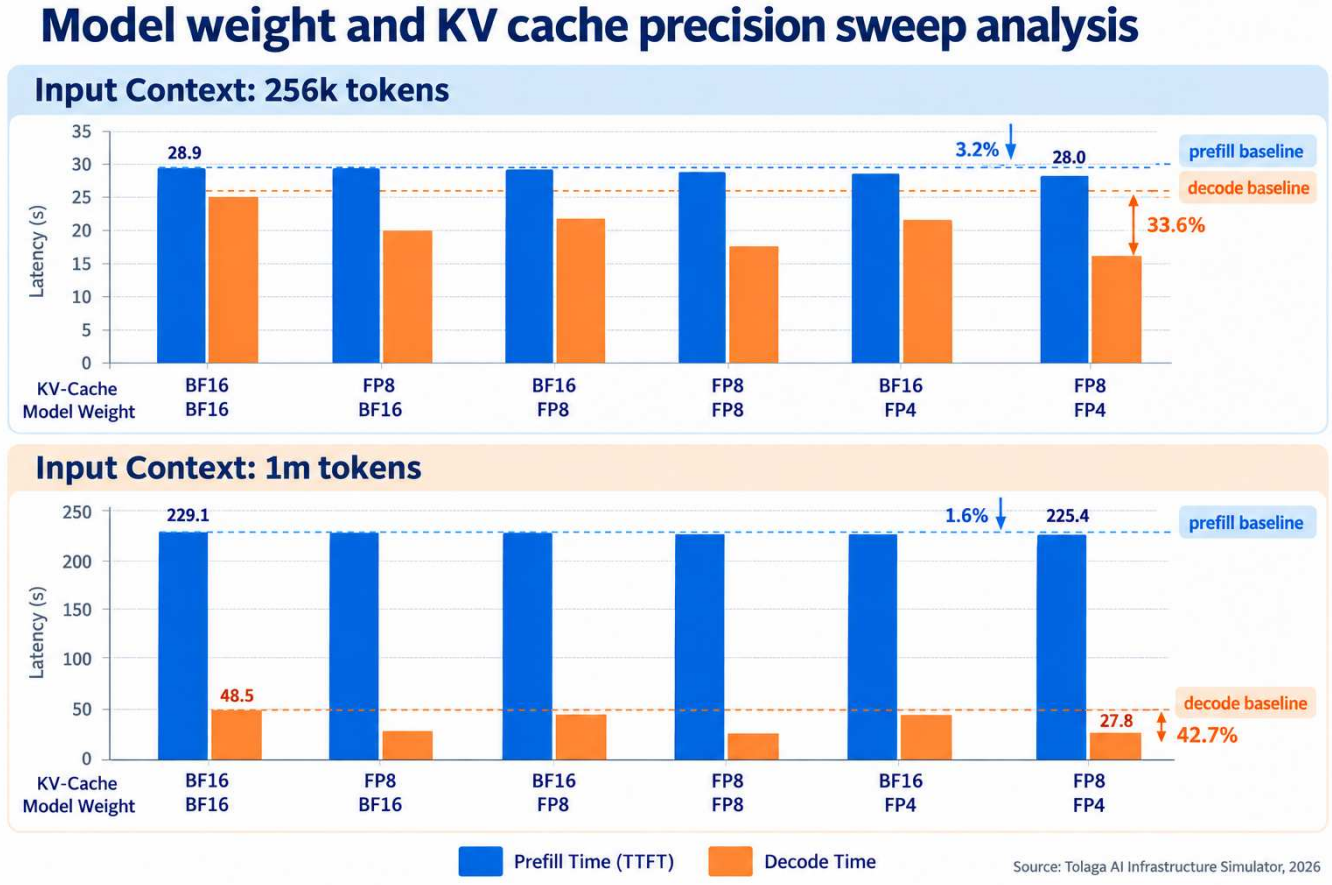
The study examines how model-weight and KV-cache precision between BF16 and FP4 affects inference performance.



Workloads with 256k and 1 million input context tokens and 512 output context tokens were simulated. In both cases, inference time was dominated by compute bound prefill processes and only saw up to 3.2% latency improvement for prefill. By contrast, latency improvements of up to 42.7% for decode were observed.

These findings support optimization strategies that treat prefill and decode separately, providing a technical rationale for disaggregated inference architectures such as NVIDIA Dynamo and Moonshot AI's Mooncake. However, the benefits depend on workload characteristics and interconnect performance, particularly the overhead of transferring KV-cache data between resources. For shorter prompts or lighter workloads, collocated prefill/decode execution remains preferable.

© 2026 Tolaga Research. All rights reserved.

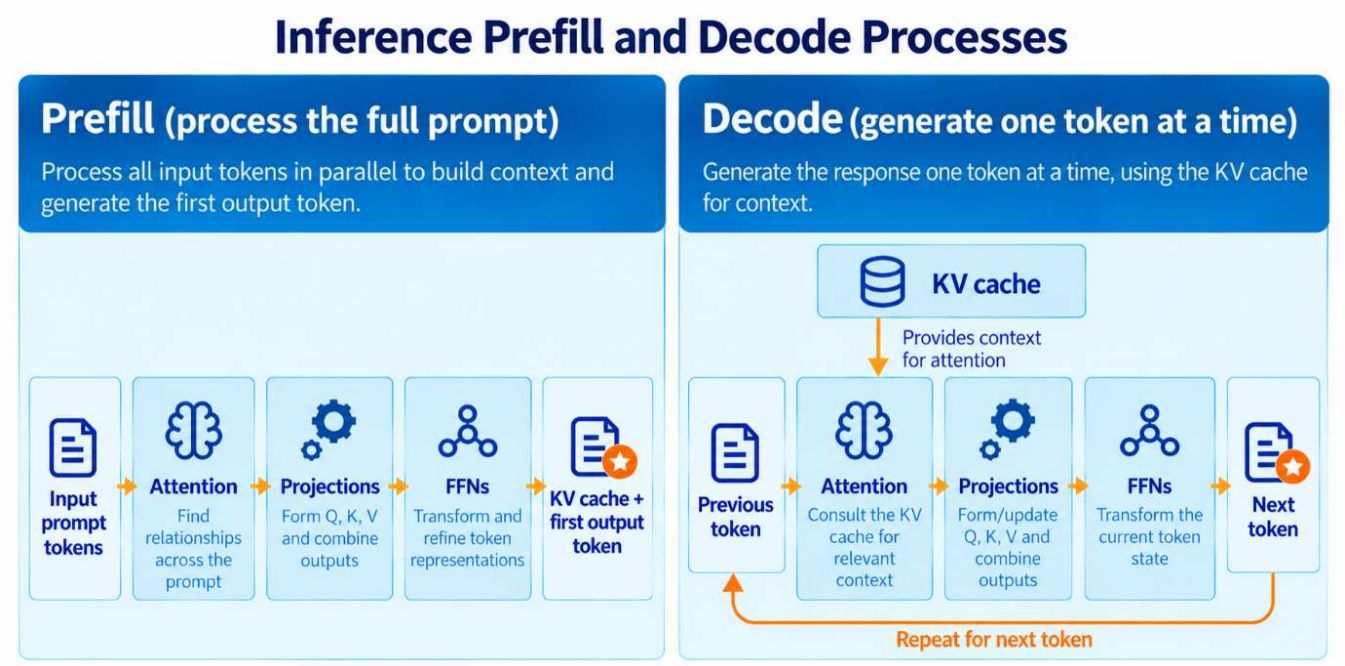


Study 2: Getting under the hood of prefill and decode

The It is instructive to investigate the prefill and decode procedures in the context of the three core operations they perform, which include:

- **Attention:** Identifies relationships between tokens and determines which parts of the context are most relevant.
- **Projections:** Transform token representations into queries (Q), keys (K), and values (V), which determine what information is sought, matched, and retrieved.
- **Feed-Forward Networks (FFNs):** Apply learned (trained) transformations to refine token representations.

© 2026 Tolaga Research. All rights reserved.



Attention, projections, and feed-forward networks (FFNs) have different computational and memory requirements. This study examines these differences by assessing operational intensity, raw compute time, and raw memory transfer time for a workload comprising 1 million input tokens and 512 output tokens.

Operational intensity measures computational work relative to memory traffic, expressed as floating-point operations (FLOPs) per byte transferred to or from high-bandwidth memory (HBM). Raw compute time represents the time required to perform the calculations, while raw memory transfer time represents the time required to move data between HBM and the processor. These times are evaluated independently, without accounting for overlap between computation and memory transfers.

The simulation results, presented below, highlight three key findings:

- **Prefill dominates overall inference latency**. The exceptionally long input context makes prefill substantially more computationally demanding than decode, with greater raw compute and memory transfer times.
- **Decode is predominantly memory-bound.** Memory transfers account for substantially more time than computation, making memory bandwidth and data movement critical determinants of decode performance.
- **Attention is the dominant performance contributor.** Attention accounts for the largest share of both computational work and memory transfer time across prefill and decode, reflecting the significant processing and memory demands of long-context workloads.
-

© 2026 Tolaga Research. All rights reserved.

Inference compute processing and memory transfer time analysis

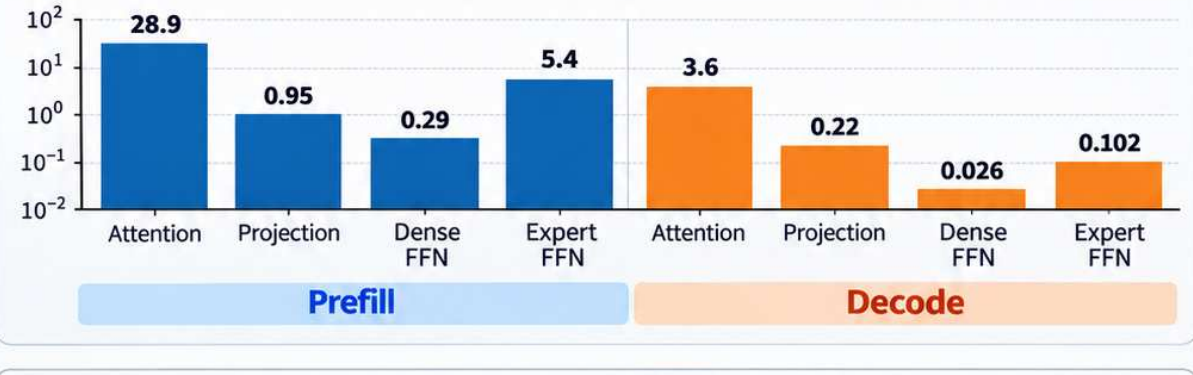
Operational Intensity (FLOPs/HBM Byte)



Raw Compute Time (seconds)



Raw Transfer Time (seconds)



Prefill (context processing) | Decode (token generation)

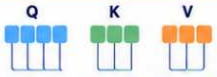
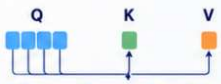
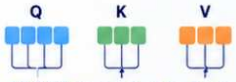
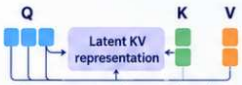
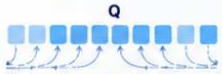
Batch size = 1 | 1M-token prompt | 511 decode steps

Source: Tolaga AI Infrastructure Simulator, 2026

© 2026 Tolaga Research. All rights reserved.

Since attention design plays a key role in long-context inference efficiency, commercial AI models employ different attention mechanisms to balance performance and memory requirements, as summarized below. The synthetic model uses Grouped-Query Attention (GQA), which reduces KV-cache requirements by sharing key and value heads across multiple query heads.

Attention Architectures in Commercial AI Models

Attention technique	How it works (conceptual)	Companies and example models	Primary benefit	Implications for inference
Multi-Head Attention (MHA)	 Each query head has its own key and value heads.	Google, OpenAI Original Transformer; earlier Google and OpenAI models	High modeling flexibility	Highest KV-cache memory requirements, increasing the potential benefit of quantization.
Multi-Query Attention (MQA)	 All query heads share a single key and value head.	TII Technology Innovation Institute Falcon 7B/40B	Substantially reduces KV-cache storage and bandwidth	Lower memory requirements and reduced memory traffic during inference.
Grouped Query Attention (GQA)	 Multiple query heads share key and value heads.	Meta Llama 3, Google Gemma 3, Alibaba Qwen3, Mistral AI Mistral 7B, OpenAI gpt-oss	Reduces KV-cache requirements while maintaining modeling flexibility	Lower KV-cache storage and bandwidth compared with MHA, while preserving performance.
Multi-Head Latent Attention (MLA)	 Compresses key and value information into latent representations.	Moonshot AI Kimi K2/K2.5 DeepSeek V3/R1	Reduces KV-cache footprint through compressed representations	Changes the memory baseline and potential gains from KV quantization, especially for long-context inference.
Sliding-Window Attention (SWA)	 Each token attends only to a limited window of nearby tokens.	Mistral AI Mistral 7B, Google Gemma 3, OpenAI gpt-oss	Reduces attention computation and possible KV-cache growth	Lower compute and memory demands for long sequences, with limited local context per layer.
Hybrid Local/ Global Attention	 Combines local attention layers with full-context attention layers.	Google Gemma 3, OpenAI gpt-oss	Balances long-range context access with memory and compute efficiency	Improves scalability for long-context workloads while preserving global context where needed.

Study 3: The benefits and cost of arithmetic precision

Rather than reducing the precision of stored model weights and KV-cache data, this study examines the impact of changing arithmetic precision during inference. It assesses how arithmetic precision, numerical conversion overhead, and computational efficiency affect inference latency, comparing BF16 with FP8 arithmetic for projections and feed-forward networks (FFNs).

Under the BF16 baseline, time to first token (TTFT) is 229.1 seconds and decode-step latency is 9.5 milliseconds (ms). Doubling projection and FFN computational throughput using FP8 delivers negligible TTFT improvement and leaves decode latency unchanged. Introducing conversion overhead of 0.01 ns per value increases TTFT to 441 seconds and decode latency to 31.4 ms. Reducing BF16 computational efficiency to 25% increases TTFT to 383 seconds, while decode latency remains unchanged.

© 2026 Tolaga Research. All rights reserved.

The results demonstrate that faster arithmetic does not necessarily translate into faster inference. Prefill performance is heavily influenced by attention computation and computational efficiency, while decode is predominantly constrained by memory access. Moreover, conversion overhead can outweigh the benefits of faster arithmetic, emphasizing the importance of considering the entire computational process rather than arithmetic throughput alone.

Impact of Arithmetic Precision, Conversion Overhead, and Compute Efficiency on Inference Latency

Scenario	TTFT (s)	Decode step (ms)
BF16 central assumptions	229.1	9.5
FP8 projections/FFNs, zero conversion cost	228.0	9.5
FP8 projections/FFNs, 0.01ns latency/value conversion	441.0	31.4
BF16, 25% compute efficiency	383.0	9.5

Source: *Tolaga Research, 2026*

The results highlight that **faster arithmetic does not guarantee faster inference**. Prefill remains sensitive to attention computation and compute efficiency, while decode is predominantly constrained by memory access. Conversion overhead can more than offset the gains from faster arithmetic.

The Right Architecture for the Right Workload

Faster compute does not guarantee faster inference, and faster memory only helps when it addresses the performance bottleneck. The four studies demonstrate that memory capacity, bandwidth, computational efficiency, and data movement must be optimized together, with their relative importance changing across workloads.

Several key findings emerge:

Prefill and decode demand different optimizations. Compute-intensive prefill and memory-intensive decode provide a strong technical rationale for disaggregated inference, allowing each phase to run on infrastructure suited to its requirements.

Quantization benefits decode more than prefill. Reducing weight and KV-cache precision delivers modeled decode-latency improvements exceeding 40% for long contexts, while prefill gains remain modest.

Faster arithmetic does not guarantee faster inference. Computational gains can be limited by other bottlenecks or negated by conversion overhead.

© 2026 Tolaga Research. All rights reserved.

Hybrid architectures offer a compelling direction. Combining compute-optimized prefill with memory-optimized decode resources can improve performance, provided the benefits outweigh communication overhead, infrastructure costs, and complexity.

Inference performance is increasingly an architectural optimization challenge, not simply a race for faster processors. As models grow and workloads become more complex, success will depend on matching infrastructure to workload requirements while minimizing unnecessary data movement.

About the Author: Phil Marshall, PhD

Phil Marshall's work in AI began more than 30 years ago with Kohonen maps, discrete neural networks, machine learning, and fuzzy logic. More recently, his work has targeted AIenabled solutions for the automotive, healthcare, manufacturing, and smart city sectors. Over the past decade, he has applied natural language processing and large language models to analyze online content and generate distinctive, datadriven industry research insights.

Marshall also brings deep experience in telecom, wireless, mobile, and networking technologies, fields he has worked in since the early 1990s. More recently, he has combined this infrastructure background with advanced software and modeling capabilities to develop Tolaga's AI Infrastructure Simulator, which is used to analyze the performance, cost, and scalability of emerging AI architectures.

Marshall is a dual citizen of the United States and New Zealand and currently resides in Auckland, New Zealand.

Contact: philip.marshall@tolaga.com

© 2026 Tolaga Research. All rights reserved.