

Decode at the edge: the role of LPDRAM in accelerating AI inference

Executive summary

Edge AI is now a core driver of innovation, delivering low-latency intelligence directly on smartphones and laptops. These edge devices are now capable of running intelligent, context-aware workloads locally, without relying on the cloud. However, as AI shifts from cloud to edge, memory becomes a critical bottleneck — placing it at the center of system performance. From centralized AI models to distributed, agentic system, three technologies are driving this transformation: small language models (SLMs), high-performance AI accelerators, and advanced memory architectures.

Micron's leadership in low-power DRAM (LPDRAM) for mobile and client positions us to meet the growing demands of edge AI. Our memory solutions deliver the bandwidth and latency required for real-time, secure, and intelligent interactions across smartphones, laptops, and emerging edge platforms, laying the foundation for a fast, efficient and smooth user experience.

This paper explores how memory performance, especially during the decode phase of inference, directly impacts the responsiveness of agentic AI systems — and why Micron is uniquely equipped to lead this evolution.

Introduction

The smartphone ecosystem is rapidly evolving to support the adoption of edge AI, transforming how users interact with their devices. Unlike cloud-based AI inferencing, which relies on servers with vast compute and memory resources to run large language models (LLMs), edge AI performs optimized inferencing tasks directly on the device. This shift eliminates network latency and enables a more private context-aware experience.

The transition to edge AI is driven by three key enablers:

- Small language models (SLMs)
- High-performance edge AI accelerators
- Advanced memory technologies like mobile LPDRAM

As edge AI capabilities expand, they increasingly support agentic AI systems that place even greater demands on memory performance, especially during the decode phase of inference. This paper explores how memory, particularly mobile LPDRAM, underpins the performance of edge AI workloads. We examine the inference pipeline and highlight how memory bandwidth directly impacts the user experience.

Small language models (SLMs): Specialized, efficient and local

SLMs are compact language models optimized to perform tasks efficiently on resource-constrained edge AI devices. Instead of relying on trillions of trained weights common in today's best LLMs, SLMs are trained using a fraction of the total parameters. For example, Llama 3.1 is a 405 billion parameter LLM model that exceeds 800GB [1], this model is far too large to run using edge hardware. In contrast, Qwen2.5 SLM, with 3 billion parameters, is approximately 6.2GB [2]. Techniques like [quantization](#) further reduce model size by lowering numerical precision for example, converting weights from 16-bit floating point to 4-bit integers. This can shrink the model footprint to around 2GB, enabling faster load times, lower power consumption and improved responsiveness.[3].

Reducing the number of parameters and the precision of the model's weights impacts the general knowledge and accuracy of the model. To address this problem, SLMs are trained and **fine-tuned for specific domains**. This specialization allows an SLM to excel at tasks like voice command recognition, summarization, translation, calendar management, etc. Multiple SLMs can run in parallel on-device, each tailored to a specific domain.

Edge AI offers speed, privacy and offline functionality, but cloud-based LLM inferencing remains essential for accessing broad external real-time knowledge and for performing complex and accurate reasoning. To capitalize on the benefits of both edge and cloud inferencing, a hybrid approach is used. The requirements of an AI use determine

whether an edge-based SLM is used for low-latency specialized tasks or a cloud-based LLM will be used to access broader external intelligence and reasoning.

Local agentic AI in action

Consider a user in the not-so-distant future asking their phone's AI assistant:

“Am I free to meet with John tomorrow, or the day after, at 3 p.m. to discuss the upcoming business trip?”

The AI assistant responds:

“Tomorrow at 3 you have your dentist appointment, but Thursday you are available. I sent a meeting invite to John for Thursday at 3 to see if that time will work for him. Would you like me to arrange transportation to the dentist since your car will still be in the shop?”

Behind this simple exchange, a coordinated pipeline of specialized models is activated, including intent recognition, named entity recognition, calendar and messaging integration, and dialogue generation. The autonomous nature of agentic AI requires multiple processes and models to execute in parallel to provide the user with a meaningful output. Agentic AI seeks to identify the user's intent and autonomously execute actions to achieve a task. By executing models locally, the agent avoids the overhead of network latency and preserves user privacy.

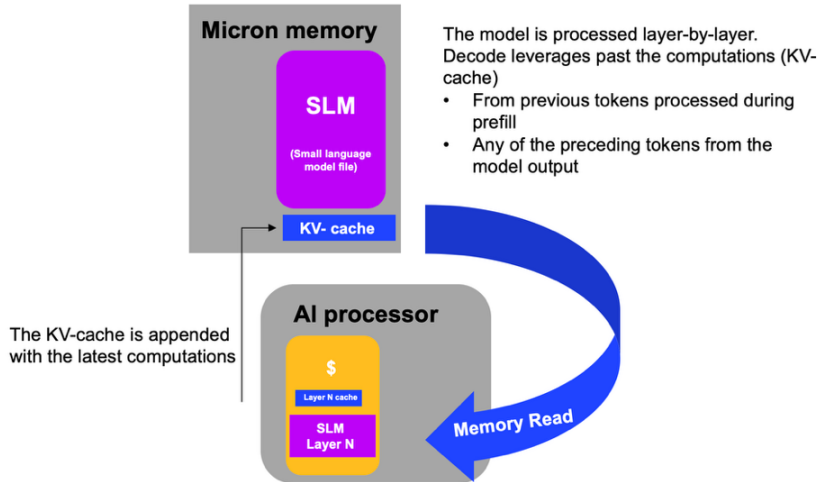
Within this pipeline SLMs are used to process and generate natural language. If the pipeline requires different SLMs, each SLM will require a separate instance of inference to be performed, generating high demands on the system hardware. SLM inference can be simplified into three phases: embedding the input, performing prefill on the embeddings to develop a relational understanding of the embeddings, and finally, executing the decode phase, which is the repeated processing of the model to predict the next output. During these steps, the model also generates hidden tokens — intermediate representations that capture reasoning and context but are never shown to the user. These hidden tokens are essential for planning and decision-making within the agentic pipeline, yet they significantly increase memory traffic and storage requirements. Each of phases of inference interacts with memory differently and can place significant demands on the memory bandwidth and free capacity.

Maintaining a copy of the SLM in memory

AI inference is highly processor-intensive and requires model data to reside close to the compute cores to achieve optimal performance. Instead of repeatedly accessing model content from persistent storage, the model's tensors and associated data are first read from the device's UFS (universal flash storage) and then loaded into LPDDR memory. While SLMs are significantly smaller than traditional LLMs, they can still exceed 4GB in size, making them among the largest files stored on a mobile device. To minimize the effects of storage read latency and reduce TTFT (time to first token), SLMs can be prefetched into memory ahead of user interaction. This proactive loading can occur during app launch or other preparatory phases, ensuring faster response times when the model is queried.

Inference phases and the role of memory

Embedding: Converting words into numerical representation



When performing inference of a model, the first step is to convert the user’s prompt into tokens — sub-word units from model-specific vocabularies that the model can process. Each token is then provided with additional meaning by being mapped to a multi-dimensional vector space using an internal embedding layer specific to the model (e.g., Llama or Qwen). This phase requires minimal memory resources.

For example:

Our prompt is first broken into individual tokens

["Am", "I", "free", "to", "meet", "with", "John", "tomorrow", ",", "or", "the", ...].

Using the embedding layer, our tokens are converted into multi-dimensional vector representations, which can now be used by the model.

```
[
  [0.01, -0.02, ..., 0.03], # 'Am'
  [0.04, 0.05, ..., -0.06], # 'I'
  ...
]
```

Prefill: Building context with attention

During the prefill phase of AI inference, the model processes all token embeddings in parallel to build contextual understanding. This is achieved through attention mechanisms, which evaluate relationships between tokens across the input sequence. The phase is computationally intensive and well-suited for the parallel processing capabilities of a smartphone’s neural processing unit (NPU).

In our example, the model must understand: “John” is a person, “Tomorrow at 3 pm” is a time reference, and “business trip” is the topic.

As part of this process, a key-value (KV) cache — per-layer, per-head buffers that store the attention “keys” and “values” from previously processed tokens so the model can reuse context without recomputing — is generated and kept in memory for each layer, attention head, and token. After computing the keys and values for all attention heads in a given layer, the model proceeds to the next layer, repeating the process. This layer-by-layer approach enables efficient attention computation across the entire input context. By reusing the model layer’s parameters for all tokens in the input, the system minimizes redundant memory access and reduces overall memory bandwidth consumption.

Decode: How it works and why it's so demanding on memory

The decode phase is the most memory-intensive stage of the inference pipeline. Unlike the prefill phase, which processes all input tokens in parallel, decode operates sequentially, generating one token at a time. For each new

token, the model computes attention only for the last generated token. The attention from previous tokens is maintained in the KV cache to eliminate the redundant processing of past tokens when generating future tokens. To determine the next token, attention is computed only for the last token generated. The KV cache holds the previously computed attention for every other token already generated and for the tokens provided in the prompt. During each decode step, the model must access both the KV cache and relevant portions of the model content from memory. This repeated access pattern creates significant memory traffic. For example, if a 2.3GB SLM generates a 20-token response in one second, the model content may be read up to 20 times, resulting in over 46GB of read activity. Additional memory bandwidth is consumed by repeated KV cache access. This intense data movement makes memory latency a dominant factor in decode performance for autoregressive transformer-based models.

From the earlier example, after the prefill processing of all input embeddings, the model will start to generate the output:

- Using its parameters and the KV cache, the model generates “tomorrow.”
- The keys and values for “tomorrow” are added to the KV cache.
- To generate the next token, “at,” the model must again access both the updated KV cache and as well as the model content.

Analogy: It’s like a GPS that must reload the entire map and recalculate your route every time it gives you the next turn.

Agentic AI: Hidden tokens, hidden costs

The decode phase becomes even more memory-intensive in the context of agentic AI. Unlike traditional interactions where only visible output tokens are generated, agentic systems must maintain high accuracy while orchestrating multiple internal processes and datasets across the user’s environment. This interconnectedness results in the generation of many additional, “hidden” tokens that are never seen by the user.

These hidden tokens may:

- Be used for internal reasoning and planning.
- Serve as inputs to specialized models (e.g., calendar access, task execution).
- Support multi-step response pipelines that require intermediate outputs.

Each of these hidden tokens must pass through the decode phase, adding to the already high memory load of an SLM. This intense data movement underscores the critical importance of maximizing memory bandwidth and minimizing latency to sustain responsive, edge AI performance, especially as agentic AI systems scale in complexity and autonomy.

Faster memory speed provides faster token generation performance

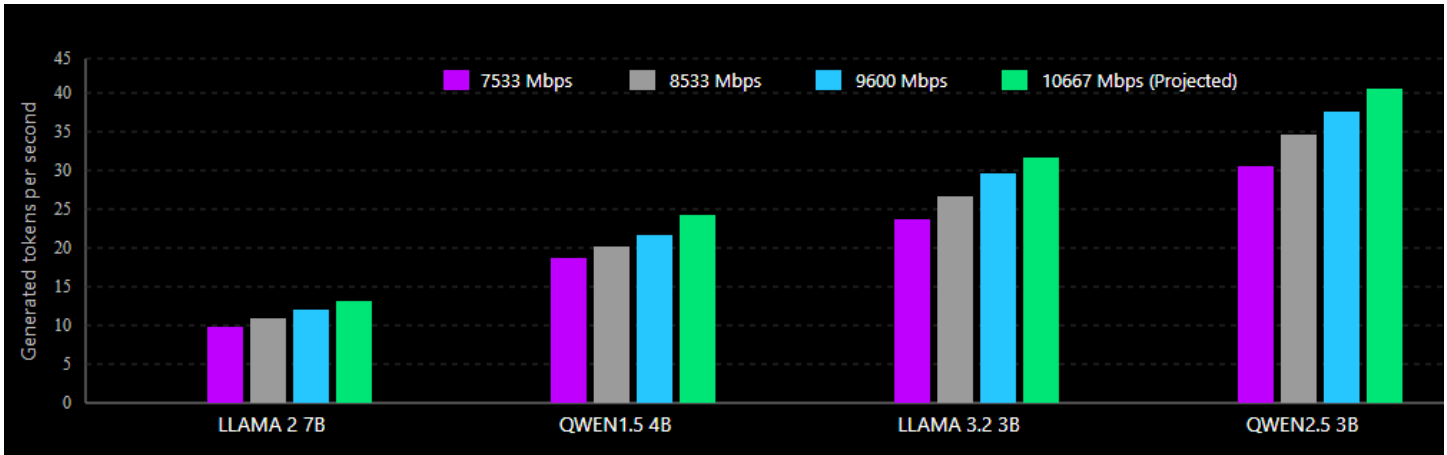


Figure 1. Measured decode performance (tokens/second) for different models and parameter sizes. Increasing the max supported LPDDR5X frequency provides meaningful decode improvements for all measured configurations. Increasing the max supported LPDDR5X frequency provides meaningful decode improvements for all measured configurations.

Micron LPDDR5X: Unlocking decode performance

Memory bandwidth is the key to overcoming decode latency for SLMs. Micron’s LPDDR5X offers industry-leading speeds that directly reduce token generation time. Increasing memory speed can provide a nearly proportionate performance increase in SLM token generation. Micron’s LPDDR5X runs at the fastest available speed grades:

- 7.5 Gbps
- 8.5 Gbps (a theoretical 13% improvement from 7.5 Gbps)
- 9.6 Gbps (a theoretical 13% improvement from 8.5 Gbps)
- 10.7 Gbps (a theoretical 11% improvement from 9.6 Gbps)

Micron’s increasing memory speed grades provide meaningful gains to the edge decode performance of SLMs. Across various models and decoding techniques, each speed grade increase resulted in **average decode performance gains of more than 9%** (see Figure 1), translating directly into faster and more responsive AI interactions.

The data shows increased memory speeds remain impactful as new decoding strategies are adopted to incorporate parallelism during decode:

- **Lookahead decoding** simultaneously predicts multiple token sequences at once, allowing parallel computation across those predictions rather than waiting for each token to be confirmed sequentially.
- **Self-speculative decoding** accelerates token generation by skipping or approximating intermediate layers during speculative steps, enabling faster inference through reduced computation and increased parallel execution paths.

These methods reduce the number of times the model is read from memory during decode by improving the utilization of the NPU. But the improved efficiency of these methods maintains a high demand on the memory to continually read the next layer of the model. As a result, the demand for higher memory bandwidths persists. Faster memory enables these parallel decoding strategies to operate at full efficiency, sustaining low decode latency and increasing the responsiveness of the model (see Figure 2).

New decode techniques benefit from faster LPDDR5X speeds

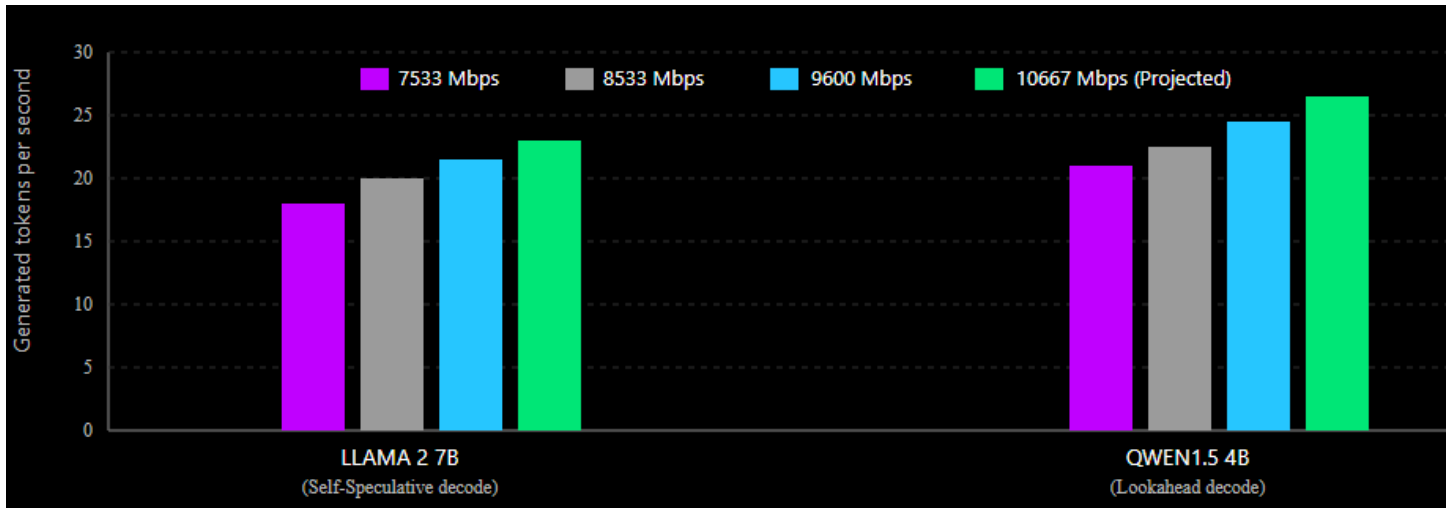


Figure 2. Measured decode performance (tokens/second) for new decode techniques. Increasing the max supported LPDDR5X frequency provides meaningful decode improvements for all measured configurations.

Conclusion

Edge AI is transforming how users interact with their devices, enabling faster, more private and context-aware experiences. But behind this transformation lies a critical foundational piece: **high-performance memory**.

From tokenization to token generation, every phase of inference depends on the speed and efficiency of LPDRAM. As AI assistants become more autonomous and multi-modal, the performance demands on memory will continue to grow.

- During edge inference, a copy of the language model must be maintained in memory to provide the AI accelerator quick access to the model weights.
- **LPDDR5X memory bandwidth** governs how fast tokens can be generated, especially during the decode phase, where memory becomes the primary bottleneck.

Micron's leadership in **LPDDR technology** positions it at the forefront of this evolution. By delivering the fastest available memory solutions, Micron is not just supporting edge AI, it's enabling the future.

The next generation of intelligent, responsive and secure edge experiences will be built on memory. Micron is building the foundation.

1. [meta-llama/Llama-3.1-405B · Hugging Face](https://huggingface.co/meta-llama/Llama-3.1-405B)
2. <https://huggingface.co/Qwen/Qwen2.5-3B>
3. <https://huggingface.co/Qwen/Qwen2.5-3B-Instruct-GPTQ-Int4>

micron.com

©2026 Micron Technology, Inc. All rights reserved. All information herein is provided on an "AS IS" basis without warranties of any kind, including any implied warranties, warranties of merchantability or warranties of fitness for a particular purpose. Micron, the Micron logo, and all other Micron trademarks are the property of Micron Technology, Inc. All other trademarks are the property of their respective owners. No hardware, software or system can provide absolute security and protection of data under all conditions. Micron assumes no liability for lost, stolen or corrupted data arising from the use of any Micron product, including those products that incorporate any of the mentioned security features. Products are warranted only to meet Micron's production data sheet specifications. Products, programs and specifications are subject to change without notice. Rev. A 01/2026 CCM004-1681249710-11848