



aiDAPTIV™ Dynamic MoE

Running Larger MoE Models with Less Memory



PHISON

Mixture of Experts (MoE)

Large language models, or LLMs, are deep neural networks that drive all things AI. It also contains broad information about anything and everything. But what if greater expertise is required for certain AI use cases? That's where Mixture of Experts, or MoE comes into play.

An MoE is a neural network architecture that splits a model into multiple specialized sub-networks, which we'll refer to as experts. This approach depends on a routing or gating network that dynamically selects which expert is ideal to process each input.

Only a subset of the model is active per token in MoE use, unlike dense models, which require every parameter for every input. This enables large model capacity without the required proportionate compute cost.

Conditional computation, or sparsity, is the concept used by MoE models, and enables a model to scale to more parameters and higher capacity without requiring linear scaling of compute costs. The result is better performance at similar or potentially lower compute demands.

There is a trade-off for greater compute efficiency, however. While only a subset of experts are active per token, all experts must still reside in memory, requiring memory capacity to scale accordingly, but Phison's Pascari aiDAPTIV technology addresses those needs more cost-effectively.

The Architecture Behind It

Each expert in an MoE is typically a feed-forward neural network (FFN), specialized sub-network. The expert specializes in patterns such as code, math, or language structures, which emerge during training.

There's a gating network, or router, that's also a small neural network that determines which experts to consult per token. The output of this gating network is a probability distribution over experts. Top-K routing is a common distribution method. So, if K was 2, then the router would choose the top two experts to send the token to.

Each token is routed to only a few experts in a sparse routing mechanism. For example, in Mixtral 8x7B, there's eight experts and only two are active per token. The outputs from experts are combined in a weighted sum.

How a MoE Gets Its Groove

Experts and the router are trained together during the training phase. The router learns which expert performs best for which inputs while experts gradually specialize based on routed data.

The inference phase is where the experts are determined, with the router selecting top-K experts for each token during this phase, and only those experts execute. Chosen experts process the token independently, then the outputs are combined using a weighted sum based on the router's probabilities.

This produces a single dense hidden representation, or vector, that is passed to the next layer. The result requires much lower compute per token compared to dense models and sometimes delivers better performance.

Why MoE?

MoE delivers the best combination of LLM capability and efficiency. It enables higher model capacity, but each token only uses a fraction of the total model parameters. This leads to greater efficiency because the compute cost is like a smaller model.

There's also the model specialization that improves the quality of the results since the experts become highly optimized for specific token types or patterns. Lastly, MoE models can reach target quantity with less compute compared to denser models. Ultimately, MoE provides big model target quality with small model compute costs and enables scaling to significantly larger models without proportionally increasing inference costs.

Tradeoffs and challenges

While MoE sounds great in theory, there's still some trade-offs and challenges. One challenge is that gating networks must learn effective routing, because poor routing results in poor performance. There's also the need for high memory capacity since all experts are still loaded in memory, even if not used per token.

Lastly, traditional MoE requires all experts to reside in GPU memory. Nevertheless, Phison has a solution for these challenges with aiDAPTIV Dynamic MoE.

Meet aiDAPTIV Dynamic MoE

Mixture of Experts models can bring greater efficiency to compute with effective routing but still have high GPU memory demands. There's no way around it – the entire model must reside in memory, somewhere.

Pascari aiDAPTIV is Phison's memory tiering technology that extends AI inference capacity across GPU memory, system memory, and NVMe SSD-based cache memory. That's where it comes to the rescue with Dynamic MoE.

aiDAPTIV's Dynamic MoE feature enables models larger than the GPU memory allows by extending the capacity using a solid-state drive. This approach distributes experts across a multi-tier memory hierarchy instead of requiring all the experts to reside in GPU memory.

Three tiers are used for aiDAPTIV Dynamic MoE – GPU memory, system memory, and cache memory. Active experts and data required for compute always reside in GPU memory – there's no way around that. All experts can reside in cache memory, as SSD storage is more cost-efficient when it comes to capacity.

However, since the cache memory isn't as fast as GPU memory, the system memory serves as an intermediate cache between the two. This memory arrangement enables only a subset of experts present in GPU memory at any given time, but the required experts are dynamically moved between the three memory tiers during inference.

Expert Placement and Initialization

A configurable number of experts are loaded into GPU memory at initialization, controlled through system parameters. If the GPU memory can hold 9 experts, the system initially loads experts 0 through 8, for example. The remaining experts are stored in the aiDAPTIV cache memory.

The system memory can serve as a cache for a portion of experts depending on configuration, optionally. This creates a working set of experts in GPU memory and system memory.

Expert Selection During Inference

The model's routing network selects the top-k experts per layer for each token. A model's architecture defines the number of selected experts (K) – GPT OSS 20B and 120B always has a k of 4, or a model may have 32 experts and

selects 4 experts per layer, for example. Pascari aiDAPTIV does not change the k value. Only the selected experts are required for computation.

Hierarchical Expert Lookup

The system performs a hierarchical lookup when an expert is required. The search begins with a check to see if the required expert is present in GPU memory, then moves onto the system memory if not found. When the required expert is in neither location, it's loaded from the cache memory via system memory, then transferred to GPU memory for compute.

Key Concepts

- Total parameters are all of the experts combined
- Activate parameters is the number of experts used per token
- An MoE model can have 47B total parameters but only 12B active per token
- Top-K routing means instead of using all experts, usually 1 or 2 are selected as the top K, and this controls the compute vs performance tradeoff
- Some MoE models can suffer from uneven expert utilization where a small subset of experts receives most tokens during training
- This can reduce effective model capacity and limit expert specialization
- Training techniques such Load Balancing and routing regularization are performed to distribute tokens more evenly across experts as a fix

Expert Caching and Eviction

A Least Recently Used, or LRU policy manages experts across memory tiers. Frequently used experts remain in GPU memory while less frequently used experts are moved to system memory or cache memory.

When a new expert is required but there's not enough space in GPU memory available, the least recently used expert is evicted. Expert selection typically follows temporal locality. Nearby tokens usually have very similar experts, leaving routing decisions to experts already located in the GPU from previous tokens.

LRU keeps a hot set of experts in GPU memory while replacing the not hot experts, making it an effective eviction policy. This caching strategy maintains a working set of active experts in GPU memory.

On-Demand Expert Loading

Experts are only loaded when required by the router in the current system. While predictive prefetching experts would be ideal for this, a predictive prefetcher is unavailable at a production-level quality, thus experts are loaded on-demand. If a required expert is not in GPU memory or system memory, it's loaded from cache memory before computation can proceed.

Compute and Execution Behavior

The system can only begin computations when all required experts for a layer are present in GPU memory. It must load it from system memory or cache memory if any required expert is missing, which impacts latency. The GPU simply idles during the loading process. This creates a dependency between data movement and compute, resulting in increased latency that directly impacts inference performance.

System Memory as Intermediate Buffer

GPU memory sits at the top of the performance hierarchy in the aiDAPTIV three tier memory architecture. That performance comes with scarce capacity and a monetary cost, because it's always attached to a GPU. SSDs have the cheapest cost-per-capacity available, but even the fastest NAND flash only delivers a fraction of the performance of GPU memory.

System memory performance sits between the two but is much closer to GPU memory performance than cache memory, which makes it the ideal intermediate buffer. Experts stored in system memory load into GPU memory faster than it would from the cache memory.

The effectiveness of system memory reduces the frequency of cache memory access for lower latency. However, if system memory caching is disabled, the experts are loaded directly into GPU memory from the cache memory, resulting in higher latency and longer GPU idle times.

Performance Characteristics

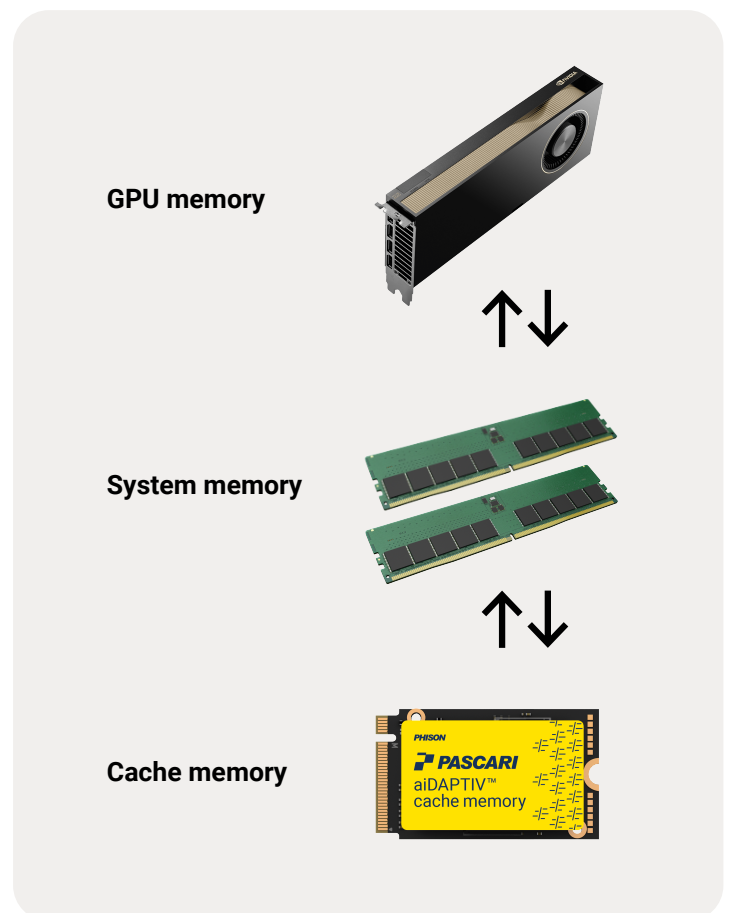
Pascari aiDAPTIV is optimized for memory capacity rather than raw latency and performance. Overall performance highly depends on GPU memory capacity, system memory size, and cache memory throughput. The GPU memory capacity dictates the number of experts that can be cached while the system memory capacity determines the effectiveness of it as a cache.

The main bottleneck for performance is the available bandwidth for the cache memory and the data transfer latency. While performance is dependent on workload characteristics, GPU compute performance is substantially faster than the cache memory can transfer data, which may cause stalling while waiting for an expert to load.

Workload Sensitivity

Input patterns of workloads greatly influence performance. Similar or repetitive queries that reuse the same experts, resulting in higher cache hit rates and fewer cache memory to system memory transfers, is the best-case scenario for performance.

Workloads where queries shift domains, such as going from math to history topics, and complex tasks requiring diverse reasoning, such as code analysis, are worst-case scenarios. These cases depend heavily on loading new experts from cache memory, and the worst of the worst may need up to approximately 80-percent of the experts loaded.



aiDAPTIV Helps Fit 32GB-Class Workloads into 16GB

Without aiDAPTIV
Requires 23.5GB Memory

Gemma4 26B A4B (15GB)

KV Cache (1.5GB)

Runtime (1GB)

OS + App (6GB)

Workload won't fit

With aiDAPTIV
Requires 23.5GB Memory

Gemma4 26B A4B (7.4GB)

KV Cache (1.5GB)

Runtime (1GB)

OS + App (6GB)

Fits with room to spare

offload



aiDAPTIV
cache memory

Tradeoffs

There're tradeoffs between memory and performance when it comes to Dynamic MoE. Increasing GPU memory capacity enables more experts to remain resident, improving the cache hit rate, and reducing dependency on loading experts from cache memory to system memory. Lower GPU memory capacity results in more frequent expert swapping, which increases latency due to frequent cache memory access.

Systems with a higher capacity of GPU memory simply perform better for MoE. However, performance isn't the complete story. Pascari aiDAPTIV Dynamic MoE expands the system's overall capability that was previously impossible. Leveraging SSD storage as cache memory enables inference of larger models that won't completely fit in GPU memory. While there's a tradeoff between model capacity and inference speed, aiDAPTIV prioritizes scalability and model size capability over consistent latency.

Gemma4 26B A4B has 26 billion total parameters with only 4 billion active per token. Without aiDAPTIV, running it requires 23.5GB DRAM. With aiDAPTIV offloading model weights to cache memory, that drops to 15.9GB, bringing a workload that wouldn't fit into a 16GB system well within reach.

Key Idea

Dynamic MoE enables large Mixture of Experts models to run on systems with limited GPU memory by treating experts as dynamically cached units across GPU memory, system memory, and cache memory. The experts are loaded on demand based on model routing decisions, letting systems do more with less GPU memory.

Deploy aiDAPTIV Memory Management Middleware, aiDAPTIV Cache Memory, take advantage of dynamic MoE with ease, and put advanced AI to work today!