www.pipelinepub.com
Volume 22, Issue 2

Edge AI: Changing GenAI Balance Between Edge and Data Center

By: Brian Case, Mark Cummings, Ph.D.

Recently, a significant technical advancement has allowed computers at the network edge to run very large LLMs (Large Language Models, which are the engines that drive generative AI systems). The result has important implications for privacy, IP protection, and availability.

Background

This capability comes from a new AI inference application called simply Inferencer. It allows even modest Apple Mac computers with as little as the base amount of DRAM memory (working memory sometimes called scratch pad memory) to run the largest open-source LLMs (Large Language Models, which are the "brain" of generative AI systems). This ability is not without compromise, of course: the rate of token generation can be very slow. Where LLMs in data centers typically generate at least high tens or even hundreds of tokens per second, Inferencer will generate a few tokens per minute when running the largest models available, such as the full Deepseek R1 at 671B parameters, GLM-4.6 at 357B parameters, or even Kimi K2 at 1Trillion parameters.

Last year, we wrote about the ability to run interactive inference of modest but useful LLMs on laptops. Interactive means entering a prompt and getting a generated response within seconds to minutes. These modest models we covered could range from low billions to thirty billion parameters. Back then, the majority of open-source models were "dense" models, which means simply that all of the billions of parameters in the model are evaluated for every token that is generated. Since then, almost all top-ranked models—regardless of their total size or their status as open or proprietary—are "sparse" models, also called mixture-of-experts or MoE models. In an MoE model, only a small subset of the total parameters are activated (evaluated) for any given token that is generated. For example, even though Kimi K2 has 1Trillion parameters, only 32B are evaluated for any generated token. Thus, this MoE model architecture dramatically increases the speed of token generation during inference, even though it does not reduce memory requirements.

In last year's article about LLM inference on laptops, low quantization (low number of bits used to represent a value) of model weights was already popular to accomplish two things: reduce the AI's memory requirements and speed up inference. With quantization, the range of model sizes that could be usefully run on laptops is increased. For example, Meta's LLaMa 70B would not fit on an Apple Mac with

64GB of memory—it couldn't run at all—but with 4-bit quantization, the memory requirement is reduced to about 40GB, which is well within the capabilities of a 64GB laptop. Quantization in this case has the additional benefit of increasing token generation speed to a comfortable level.

In addition to the industry moving to MoE models with quantization, researchers in all the AI labs have made significant improvements in pre-training (which now refers to what used to be called simply "training") and post-training. The result is models of all sizes having more intelligence and capabilities. In addition, training is now often done with quantization in mind; that is, the model builders conduct all aspects of training knowing that the final model that will be run will be in a low-quantized form, whether in a data center or at the Edge. At the Edge, users are concerned with a small computer being able to run inference at all and with acceptable speed; in the data center, everyone is concerned with speed, but also size. Because small, quick models consume the minimum of expensive and often scarce data-center resources. In summary, the goals of data-center inference and Edge inference are very aligned.

When the AI race got started with the introduction of ChatGPT back in late 2022, model weights (parameters) were typically represented with 32 or 16 bits. Now, it is widely recognized that a properly trained and quantized model can achieve very nearly the same quality of output with only 4 bits per weight. This fact is so accepted that Nvidia and other hardware vendors have added hardware support for multiple 8-bit and 4-bit representations. In doing so, they have cast quantization into the 'stone' of their hardware.

Nonetheless, the largest models were still out of reach for Edge users because these trends were not powerful enough to overcome the sheer size of the biggest models.

For example, Kimi K2 at 4 bits per weight quantization still requires 500GB just to hold the model weights, even though only 16GB is needed for any specific token generation. The MoE architecture reduces the size of the set of active weights but not the size of the model that has to be stored.

One solution to the sheer size requirement is to simply increase the DRAM memory in a laptop. This will happen over time, but today, a half-terabyte or even a full-terabyte of DRAM has very challenging cost impacts for Edge systems.

SSD Streaming

The Inferencer solution to this problem is to leave all the model weights on the laptop's SSD (Solid State Disk - also known as the "hard disk") and stream them into DRAM as needed. This can work, but it results in token generation being limited by the read speed of the SSD. To understand the tradeoffs of SSD streaming, it is helpful to understand a few details about the structure and operation of LLMs.

An LLM is built from a stack of layers; each layer implements some very large matrix operations (chiefly multiplies). Large LLMs, such as Deepseek R1 or Kimi K2, might consist of as many as a hundred or two hundred layers. Each layer computes some results that are passed on to the next layer in the stack. These intermediate results are not huge; they are on the order of the context (background data used by the LLM in producing the desired output) size (the number of tokens in the context window) times the embedding (how tokens are encoded) dimension. This could be as many as ten thousand numbers (embedding dimension) for each of a hundred thousand positions (tokens) in the context. That is, it might be on the order of 10,000 x 100,000 or 1,000,000,000 (one billion).

With SSD streaming, the Inferencer app streams (loads) model weights from the SSD into working memory, where they are used once to compute the results of the current layer in the model. This is repeated for each layer until the final layer, which computes the probabilities for the next token to be generated.

This has the desirable properties of: (1) allowing nearly any size of model to be run by the laptop (or other Edge hardware) and (2) allowing the model to use any quantization level up to full-precision (16-bit)

representation for the model weights. Thus, with Inferencer's SSD streaming, it is possible for a laptop to run the full-size, unquantized Deepseek R1 model. But, at full 16-bit precision, just storing the model requires 1.3TB (1,342,000,000 bytes). So, a laptop or other hardware does still need a hefty amount of free SSD storage. These days, however, 8-bit quantization is considered to deliver full quality from LLMs, so in actuality, at most, only 671GB (671,000,000,000 bytes) of free storage would be required. With 4-bit quantization, storage requirements are reduced further to a "mere" 336GB. As discussed above, 4-bit quantization can deliver excellent inference quality when training and quantization have been done with low quantization in mind.

Since Deepseek R1 is an MoE model with 37B active parameters, "only" 37 billion out of the total of 671 billion weights need to be streamed in from storage to generate any given next token. Further, at 4-bit quantization, only ~18.5 billion bytes need to be streamed in from the SSD.

SSD Streaming Latency

The performance of running an LLM with Inferencer will be limited by one of two things: (1) the speed of the streaming weights from SSD into the laptop's working memory, or (2) the speed of the matrix-multiply compute in the processor. Compute, however, for a modern laptop will not be the bottleneck because even an entry-level laptop CPU can execute hundreds of billions of multiply-accumulate operations per second.

For a modern laptop, the internal SSD can stream at least 3GB/s (some will be significantly faster, some slower). Given such an SSD, the speed of inference on Deepseek R1 with 37B active parameters can be approximated as follows: at 4-bit quantization, the size of the set of active weights is ~19GB (37GB times one-half since 4 bits is a half byte). We then divide 19GB by 3GB/s to get about 6 seconds per trip through the set of active weights; that is, ~6 seconds per token. So, that's about 10 tokens per minute, or 0.16 tokens/second inference speed.

For a text answer to a question, this kind of response time may be challenging. For example, ~3 minutes for a one-sentence response. For a one paragraph response, ~30 minutes. Two paragraphs ~1 hour. Etc. For one, the model shows its reasoning process for many hours.

One way to think about the potential of extremely slow but extremely intelligent AI inference is to ask the following question: Would an extremely slow Einstein or Steven Hawking be valuable to a user? If a user could ask this virtual Hawking an important question, go to sleep, and wake up to a genius answer, how much value could that bring?

For an intelligent agent application that needs only five tokens per transaction and a 1-minute transaction time, this latency may not be a problem. While the improved quality from these large frontier models may be very valuable.

It's important to remember that even modest AIs today are already more knowledgeable and much more capable in some ways than even the smartest among us. Over time, AIs of all sizes are increasing in capability.

So, while Edge hardware will be able to run increasingly good models with interactive speed (say, 15 tokens/second and faster), with SSD streaming, it is also possible to run the best open-weights models at sub-interactive speed. One possible use case is to use a speedy, interactive model to refine the prompt. That is, a user can refine the phrasing that gives the best results, so that the model understands the best. Then submit the refined prompt to the extremely slow but extremely intelligent "thinks overnight" model.

Impact

Running GenAI on commodity hardware means that it can be run at the network Edge. Operation at the Edge has some intrinsic advantages that include: reliability, privacy/IP protection, and network latency. There may also be financial drivers as well.

The recent Amazon outage is a good example of what can happen when people or organizations depend on a data center network-accessible AI. Having an Edge implementation, either as a standalone or a backup, can overcome these outage problems.

Working with vendor-provided data center AI has some inherent privacy and IP (Intellectual Property) exposures. For some applications, these exposures can be quite important. Many people think of nation-state organizations as being the most protective of secrets. But experience shows that automated factory, Pharma, automotive, etc., industries can be more paranoid. For them, the fact that their data can be used in training LLMs, in the context windows of other users, etc., may be too great a concern. The data may not go to others in exactly the form received by the data center. But it may be used in training. Thus, it is part of the reasoning data that the GenAI system uses. Resulting in what is termed 'IP Leakage'.

Some organizations may meet this concern by implementing their own private data center. However, this still has a data exposure risk on the network that accesses the data center. Plus, Edge systems may be more cost-effective, have more predictable expense profiles, etc. Finally, there may be latency issues. Especially for intelligent agents, time can be critical. Just the round-trip network communication time to and back from the data center may be problematic. For these reasons, and possibly just convenience, users may prefer running GenAI locally on edge systems

Likely Future Developments

One way to think about the future of AI is to look at the pattern experienced in the introduction of the microprocessor. When the first microprocessors appeared, they were used simply to build less expensive mini-computers. At that time, few could envisage that there would be greeting cards with a small computer used to reproduce an audio greeting message. So, it is hard to say exactly in what forms GenAI will emerge, but it is clear that the move to the Edge is well underway. This doesn't mean that data center GenAI will disappear, but it does mean that there will be a balance of Edge and data center usage.

Conclusion

Today, GenAI is dominated by large data center deployments. AI inference at the edge has started. It is driven by requirements for reliability, privacy, IP protection, latency, convenience, and financial concerns. As time goes on, there are likely to be further significant advancements in Edge systems.