www.pipelinepub.com
Volume 20, Issue 8

LLMs on Commodity Hardware: What, Where, Why AI PCs Are Already Here

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

Generative AI (GenAI) Large Language Models (LLMs) can run with okay single-user performance on today's notebook computers. This has very important implications. First, it promises to give individual users a great increase in personal power for both good and for bad. Second, it foreshadows a world where GenAI is everywhere and in everything including routers, switches, home controllers, industrial systems, the electrical grid, water systems, etc. Given all the press about Nvidia's very expensive specialized chips that companies are fighting to buy, this assertion may be a little surprising. To help understand this sea change, we provide a little background, the results of some real-world tests, and a beginning of a discussion on implications.

Background

Before LLMs, AI applications were built on a variety of Machine Learning (ML) model architectures, each focused on one specific task. With the advent of LLMs, GenAI has been shown to be able to do all the things that those single purpose AI systems could do and more.

LLM performance is typically measured along two axes: 1) the speed of output, called inference performance (how fast it generates a response to a prompt); and 2) the quality of their output, referred to as accuracy (the "intelligence" of the response). Model accuracy is typically a function of characteristics that are fixed during training, which is before inference happens. These characteristics include model size, nature of the training data, goals of fine tuning (including alignment), and so on. These characteristics affect, in general, the amount of knowledge stored in the LLM. Here we are interested in the inference speed achievable with commodity hardware running trained open-source LLMs.

What is an LLM?

The LLM revolution was triggered by a key invention known as attention, which allows a model to understand the relationships of words in sentences, elements of pictures, sounds in audio, and so forth. This invention resulted in the transformer architecture, described in the seminal paper **Attention is All You Need**; this architecture is used in modern LLMs. A transformer, like most other modern AI algorithms, is a machine-learning model implemented as a DNN (Deep Neural Network). LLM capabilities scale with model size, which is determined by so-called hyper-parameters such as the number of transformer layers, token embedding size (dimensionality of semantics for each token), number of attention heads, etc. For inference, these hyper-parameters are interesting, but they are "baked in" to the model during training and are not, generally speaking, modifiable.

To improve inference speed, a process known as post-training quantization reduces the number of bits that represent each weight (a weight is also called a parameter), which is typically a floating-point value. These weights are a result of training, fine tuning, etc. Since even a small LLM has over a billion weights, a reduction in the number of bits per weight has a big impact on storage requirements and performance. Up to a point, reducing the number of bits (the degree of quantization) usually increases the speed of inference with only minor effects on accuracy.

In contrast to inference, LLM training is a very resource intensive task and is not considered here. Rather, our focus is on the use of LLMs, which means inference.

Where Can we Run LLMs and How Well do they Run?

Here, we show that LLM inference is already feasible on commodity hardware, specifically laptops. This is important because history shows that what runs on today's laptops will run on tomorrow's smartphones, tablets, IoT devices, etc. To illustrate our finding, we use software created by the open source llama.cpp project. We use a few sizes of some open-source LLMs. We run them on a few laptops and one server. The laptops range from today's fastest, high end, back to old ones that may be still in use. The server is from yesterday's generation of high-end data-center systems. Table 1 shows the hardware used.

Platform (CPU + GPU)	CPU P-Cores	GPU Cores
Server (Xeon Gold 2P (2023) + A100 80GB (2020))	32	108
High-end Contemporary Laptop (M3 Max in 16" MacBook Pro, 2023)	12	40
Low-end Contemporary Laptop (M1 Pro in 14" MacBook Pro, 2020)	8	16
Legacy Laptop (i7 8550 in HP x360 Laptop, 1997)	4	n/a

Table 1. Hardware platforms used in this work.

The fastest hardware we used is a system comprised of a 2P Xeon Gold server with a single Nvidia A100 80GB GPU. The A100 is still very useful for inference, but as of Nvidia's latest announcements, it is now two generations behind their leading-edge AI hardware. In addition, we ran llama.cpp on Apple MacBook Pro laptops with the M-series chips. These chips integrate a multi-core CPU with a capable GPU in the same SoC (system-on-chip). The same microarchitecture is implemented in Apple's iPhone handsets — though with fewer cores — so we can postulate that today's low-end MacBook performance is tomorrow's iPhone performance. We expect other smartphone platforms to keep pace due to competitive pressures.

To put a bound on the low end, we employ one past-generation Intel x86 microprocessor in an HP laptop; as with the sever platform, it's running Linux.

We will compare a small selection of models and sizes to get a sense for what is possible currently: two LLaMA-2 models — 7B-chat (small at seven-billion weights) and 70B-chat (largest at seventy-billion); the state-of-the-art Mixtral-8x7B-instruct mixture-of-experts (MoE) model (large at fifty-six billion total parameters with fourteen-billion used at any one time); and the two small models — phi-2-2.7B (small at 2.7-billion) and tiny-vicuna-1B (smallest at one-billion). All models will be run with 4-bit quantization in the format known as GGUF, which is the native quantized format for llama.cpp. We show results for five models on the memory-rich platforms and three models on the memory-constrained. For high-end GPU-assisted results, we compare the largest models separately.

Consult Tables 2, 3, and 4 to see the results. Each row is for a single model on a single platform, and there are columns for raw speed measured as tokens generated per second (tokens/second), power dissipation (the sum of CPU and GPU when applicable), and efficiency measured as speed/Watt (tokens/second/Watt). Tables 2 and 3 are for inference with the combination of CPU and GPU (the GPU is used to speed up matrix multiplication; a CPU is always needed to coordinate overall execution of the model). Table 4 shows results for CPU-only inference. With modern multicore CPUs, LLM inference can still achieve practical, useful speed. As is to be expected, smaller models inference more quickly.

Hardware	LLM	tokens/second	Power	tokens/second/Watt
Xeon Gold 2P + A100 80GB	LLaMA-70B-chat	27 t/s	660 W	0.04
	Mixtral-8x7B-instruct	65 t/s	560 W	0.12
M3 Max 16" MacBook	LLaMA-70B-chat	8.5 t/s	39 W	0.23
	Mixtral-8x7B-instruct	37 t/s	32 W	1.2

Table 2. CPU plus GPU inference speed and performance-per-Watt for large models on server and high-end laptop. The MoE Mixtral-8x7B has both better accuracy (response quality; not shown) and better speed than the large LLaMA model.

Hardware	LLM	tokens/second	Power	tokens/second/Watt
Xeon Gold 2P + A100 80GB	LLaMA-7B-chat	145 t/s	552 W	0.26
	phi-2-2.7B	175 t/s	487 W	0.36
	tiny-vicuna-1B	270 t/s	414 W	0.65
M3 Max 16" MacBook	LLaMA-7B-chat	68 t/s	39 W	1.7
	phi-2-2.7B	120 t/s	23 W	5.2
	tiny-vicuna-1B	218 t/s	15 W	15
M1 Pro 14" MacBook	LLaMA-2 7B-chat	37 t/s	20 W	1.9
	phi-2-2.7B	65 t/s	15 W	4.3
	tiny-vicuna-1B	128 t/s	16 W	8.0

Table 3. CPU plus GPU inference speed and performance-per-Watt for smaller models.

on average. As shown in Tables 2, 3, and 4, we measure inference speed in tokens-per-second (tokens/sec). A token is, in concept, a word; in practice, however, it's actually about three-quarters of a word on average.

The results show, as expected, that the Nvidia GPU platform yields the best performance. If the A100 were combined with a lower-power desktop chip, the efficiency, tokens/second/Watt, would be better, but it would still significantly lag the Apple M-series platforms. Specifically, we expect a desktop CPU would dissipate on the order of 100W instead of the data-center configuration's 300W. The Nvidia A100 (or the rough consumer equivalent, the GeForce 4090) would dissipate the same power regardless.

The general trend of results shows that smaller but still capable models, e.g., the llama-2-7B-chat with seven billion parameters, run with very good performance on most consumer hardware even without the help of a GPU. Despite being smaller than the LLaMA-2-70B-chat model, Mixtral-8x7B-instruct exceeds it on objective evaluations of accuracy (response quality, not shown). Mixtral has faster inference than LLaMA-2-70B because it uses the modern MoE architecture, which engages only one-fourth of the model's parameters at any one time. The Xeon is a good stand-in for modern consumer x86 CPUs since the core micro-architectures are similar.

As shown in Tables 2, 3, and 4, we measure inference speed in tokens-per-second (tokens/sec). A token is, in concept, a word; in practice, however, it's actually about three-quarters of a word

The phi-2-2.7B model is an example of a smaller model that nonetheless produces high-quality results. Microsoft created this model with the explicit goal of training on higher-quality data to see if that could endow a smaller model with higher-quality output, and results show that it does. Phi-2-2.7B produces accuracy on par or exceeding that of LLaMA-2-7B-chat, which is roughly twice its size. Phi-2-2.7B and the tiny-vicuna-1B model show that in the case where a small model is known to produce sufficient inference quality, even CPU-only execution results in adequate inference speed. Even the low-power i7 8550U can produce double-digit tokens-per-second with these small models, and even its worst efficiency rating competes well with that of the Xeon. It's a slightly mis-matched comparison since the design points of the laptop and server chips are very different, but the comparison is nonetheless Revealing.

Hardware	LLM	tokens/second	Power	tokens/second/Watt
Xeon Gold 2P	LLaMA-70B-chat	3.0 t/s	420 W	0.01
	Mixtral-8x7B-instruct	11 t/s	420 W	0.03
	LLaMA-7B-chat	21 t/s	420 W	0.05
	phi-2-2.7B	31 t/s	420 W	0.07
	tiny-vicuna-1B	60 t/s	260 W	0.23
M3 Max 16" MacBook Pro	LLaMA-70B-chat	3.0 t/s	43 W	0.07
	Mixtral-8x7B-instruct	15 t/s	37 W	0.41
	LLaMA-7B-chat	29 t/s	40 W	0.72
	phi-2-2.7B	57 t/s	32 W	1.8
	tiny-vicuna-1B	138 t/s	28 W	4.9
M1 Pro 14" MacBook Pro	LLaMA-2 7B-chat	23 t/s	33 W	0.70
	phi-2-2.7B	45 t/s	28 W	1.6
	tiny-vicuna-1B	105 t/s	25 W	4.2
I7 8550U HP 13" Laptop	LLaMA-2 7B-chat	5.0 t/s	15 W	0.33
	phi-2-2.7B	10 t/s	15 W	0.67
	tiny-vicuna-1B	24 t/s	15 W	1.6

Table 4. CPU-only inference speed and performance-per-Watt for all model sizes that can run on each platform (limitation due to memory capacity on the low-end laptops). Note the high-end laptop outperforms the server on all but the largest model.

The main conclusion from this data is that, as shown, ubiquitous LLM deployment is possible today. Apple already has announced that its next notebooks will be more specialized to provide even better performance. Both AMD and Intel have announced that forthcoming x86 chips will have enhanced AI capabilities. We can expect these developments to be followed by improved chips for smartphones and tablets.

Implications

The implications of end users running LLMs on their own machines are quite extensive. So extensive that they deserve their own article. Here are some brief summary considerations.

Individual users working with personal computers running LLMs that they can choose and tailor to their specific needs may mean the user will have more options than merely going online to a multi-user public GenAI system. There are also significant advantages to using local LLMs. On the ethical use side, running in this mode can avoid IP leakage. It can also allow for safe and ethical cybersecurity testing. Both are serious concerns now on multi-user public systems. These Personal LLMs can also be tailored for the specific needs of the user by improving performance, lowering power consumption, using LLMs tailored for specific disciplines, etc.

These Personal LLMs can, however, also remove the guard rails against creating cyber attacks, pornography, social engineering, fake news, etc.

Then there are the implications of embedding LLMs in our networks, infrastructure, appliances, etc. This can have very beneficial effects, but, on the other hand, it can have negative side effects such as hallucinations, which are the creation of undetected false responses. One of the central questions surrounding LLM deployment concerns so-called alignment: Can LLMs be made to avoid hallucinations, refuse to produce dangerous output, be empathetic to human needs and so on to avoid these negative side effects? Here again, this is a big question that will require significant community efforts and resources. This is a subject deserving an article of its own.

One thing that is clear: the social impact of personal LLMs will be significant. The impacts need to be studied and reported on.

Conclusion

The fact that GenAI LLMs can run with okay single user performance on today's laptop computers has been shown. Soon to be released laptops will provide even better LLM performance. Smartphone, tablet, and IoT LLM platforms will soon follow. The implications for both good and bad are apparent but not totally clear. More study and analysis of the likely future path and consequences are called for.