Bringing LLMs back to your local machine

LLMs like ChatGPT are all the hype. Using them as they are or as the key part of a RAG (Retrieval-Augmented Generation) system stretches the limits of what is possible in software development today. Unfortunately, those models typically run in the cloud either because vendors just don’t want to share their models or because there simply is no hardware you could buy in large numbers to make them run in the first place. There are, however, reasons why you would want an LLM to run on machines managed by yourself, like:

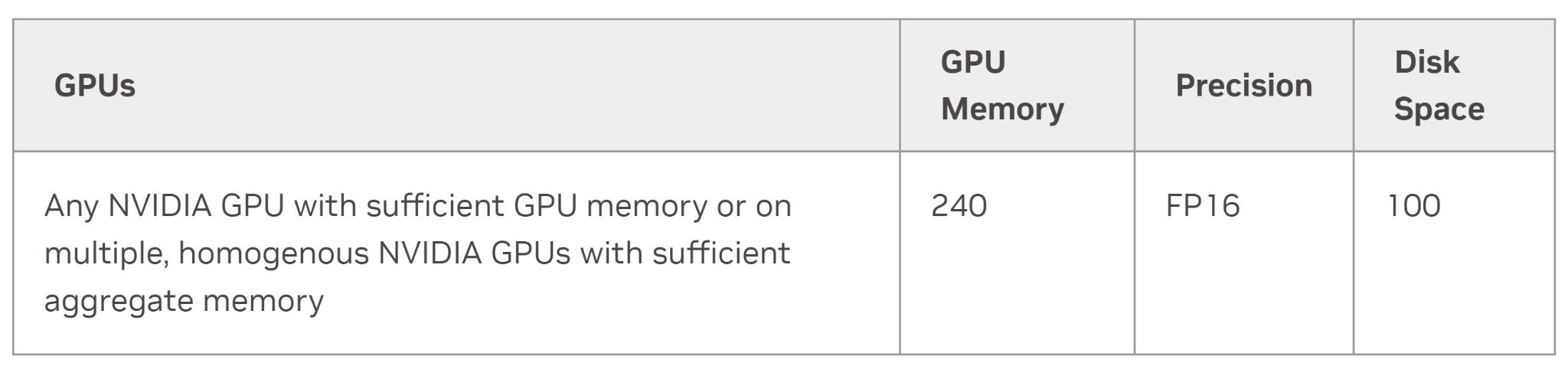
1. Privacy & data protection - think of health or legal data which must remain in your local networks
2. Full control of operation
3. Cost of operation
4. Ecological footprint

However, hosting LLMs yourself requires you to have access to affordable and readily available GPUs. This requires them to be small enough to fit or at least reduce computational burden on GPUs.

# The Challenge

Let’s look at *Llama 3 70B Instruct*, a model that might perform similar to GPT-3.5.

Looking at chart in figure 1 coming directly from the NVIDIA documentation shows us that even a single instance of this model would need 240 GB of GPU memory, so you would need three H100 - H100 being the current flagship GPU of NVIDIA having a price tag as high as 30k €. Almost 100k € to run a decent LLM with a single concurrent request sounds steep. So steep, that reportedly [NVIDIA ships H100 to data centers in armored cars](https://www.wsj.com/articles/armored-cars-and-trillion-dollar-price-tags-how-some-tech-leaders-want-to-solve-the-chip-shortage-d7c75039).

[](https://docs.nvidia.com/nim/large-language-models/24.05.rc15/support-matrix.html#id2)

*Figure 1: who has three H100 GPUs (needed to load the full version of Llama 3 instruct)*

Quoting the same source: optimizing such a model for latency would even require the full compute power of 8 H100. So, even when you have that many H100, how would you even scale that to a higher load?

So, for a few people this might be doable, but not for the majority of us where even access to a good number of smaller GPUs like an L4 (<https://www.nvidia.com/en-us/data-center/l4/>) or maybe the more powerful L40s (<https://www.nvidia.com/en-us/data-center/l40s/>) - which are comparable to the most recent consumer RTX 40 series) - can be a challenge. Additionally, if you just want to try out one of the newer models how would you even get temporary access to such a number of high end GPUs?

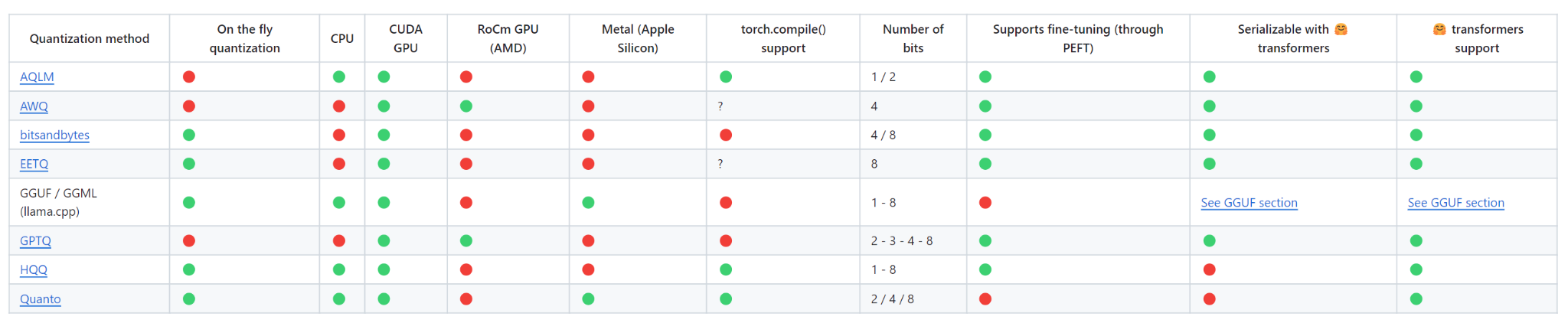
# The Remedy

There is hope, however, and the remedy lies in making the LLMs work on these GPUs or even less powerful ones by

1. using a smaller model to begin with
2. reducing the precision of the parameters to 8, 4 or even less bits
3. using a sparse mixture of experts that at least reduces the required computational power (see <https://huggingface.co/blog/moe>)

At least for experiments this allows you to run a Llama 3 Instruct model on a GPU as basic as a T4 with 16GB (comparable to the outdated consumer RTX 20 series). Services like [Google Colab](https://colab.research.google.com) offer such a GPU even in their free tier. And if you are ready to pay a few Euros per month, you get an L4 or even an A100 which will allow for less compressed versions of the Llama model or faster execution.

[HuggingFace](https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct) hosts a version of the Llama 3 Instruct model that is already reduced to 8 billion parameters which will already make it fit in 16GB of GPU (T4). When we also need memory for the context (up to 8k), we would need at least an L4 with 24GB. So for the free Colab version we can apply a precision reduction from 16 to 8 bits, and we are good to go. Reducing the bit length is called quantization. There are different ways of quantizing the model’s parameters to a lower resolution as summarized in figure 2 described here <https://huggingface.co/docs/transformers/v4.42.0/quantization/overview>.



*Figure 2: Options to reduce the size of parameters in the GPU’s memory*

# Make it run on Colab

With <https://huggingface.co/docs/transformers/v4.42.0/quantization/bitsandbytes> we choose the easiest option that supports at least 8 and 4 bits. For us 8 bit is small enough, so this is what we do, and yes, it is as easy as this to load the model in the reduced resolution:

from transformers import AutoModelForCausalLM, BitsAndBytesConfig

quantization\_config = BitsAndBytesConfig(load\_in\_8bit=True)

model = AutoModelForCausalLM.from\_pretrained(

"meta-llama/Meta-Llama-3-8B-Instruct",

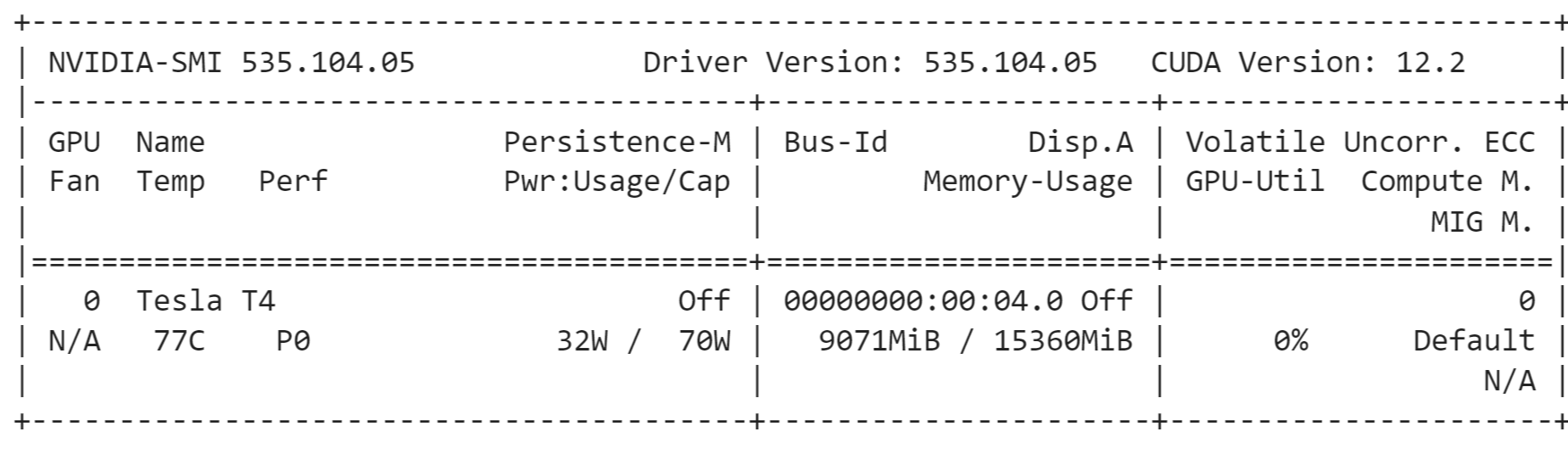
quantization\_config=quantization\_config

)

You can directly make this code run in this Colab notebook we prepared for you:

<https://github.com/DJCordhose/transformers/blob/main/notebooks/Llama_3_8B_Instruct_8bit.ipynb>

We have configured it to use a T4 GPU and after running the first generation you see that the quantized model very easily fits into memory:



*Figure 3: Even after using a bit of memory for context, the quantized model fits easily*

# Finally

To be fair it should be mentioned that even though LLama 3 does run on a T4, it is really slow. Quantization reduces memory footprint, but does not mean the model is faster, it can even be a bit slower. Sparse Mixture of Experts (SMoE) models like Mixtral [8x7B](https://mistral.ai/news/mixtral-of-experts/) additionally reduce the number of active parameters. This allows to execute even such a model - that has 8 times more parameters than the Llama we used - on a T4 as you can see there <https://github.com/DJCordhose/transformers/blob/main/notebooks/Mixtral_8x7B_Instruct_HQQ_T4.ipynb>.

While it might be fast enough for offline use, you certainly would not want an online chat with a system that makes you wait for minutes to answer. In that case you would want to use a faster GPU that will easily bring the latency down to just a few seconds.

Finally, reducing the size of a model comes at a price. Smaller models will likely be less accurate, so using good evaluations and validating answers is even more crucial.

# Our Workshop at ODSC Europe 2024 in London

Our training "How to make LLMs fit into commodity hardware again: A Practical Guide" covers these topics in more depth and will elaborate on evaluation and validation. It will be held in-person at [ODSC Europe this September in London](https://odsc.com/europe/europe-schedule/).

Oliver Zeigermann



Oliver works as an AI engineer from Hamburg, Germany. He has been developing software with different approaches and programming languages for more than 3 decades. In the past decade he has been focusing on Machine Learning and its interactions with humans.

Christian Hidber



Christian lives in Zurich, Switzerland and works as a Consultant focusing on real-world machine learning applications. He earned his PhD in mathematics from ETH Zurich. Christian has been developing and architecting IT solutions for the last 20+ years. Currently he’s applying artificial intelligence to Geberit’s planning software ProPlanner.