Navigating the High Cost of AI Compute

by Guido Appenzeller, Matt Bornstein, and Martin Casado

The generative AI boom is compute-bound. It has the unique property that adding more compute directly results in a better product. Usually, R&D investment is more directly tied to how valuable a product was, and that relationship is markedly sublinear. But this is not currently so with artificial intelligence and, as a result, a predominant factor driving the industry today is simply the cost of training and inference.

While we don't know the true numbers, we've heard from reputable sources that the supply of compute is so constrained, demand outstrips it by a factor of 10(!) So we think it's fair to say that, right now, **access to compute resources — at the lowest total cost — has become a determining factor for the success of AI companies**.

In fact, we've seen many companies spend more than 80% of their total capital raised on compute resources!

In this post, we try to break down the cost factors for an AI company. The absolute numbers will of course change over time, but we don't see immediate relief from AI companies being bound by their access to compute resources. So, hopefully, this is a helpful framework for thinking through the landscape.

Why are AI models so computationally expensive?

There is a wide variety of generative AI models, and inference and training costs depend on the size and type of the model. Fortunately, the most popular models today are mostly transformer-based architectures, which include popular large language models (LLMs) such as GPT-3, GPT-J, or BERT. While the exact number of operations for inference and learning of transformers is model-specific (see this paper), there is a fairly accurate rule of thumb that depends only on the number of parameters (i.e., the weights of the neural networks) of the model and the number of input and output tokens.

Tokens are essentially short sequences of a few characters. They correspond to words or parts of words. The best way to get an intuition for tokens is to try out tokenization with publicly available online tokenizers (e.g., OpenAI). For GPT-3, the average length of a token is 4 characters.

The rule of thumb for transformers is that a forward pass (i.e., inference) for a model with p parameters for an input and an output sequence of length n tokens each,

takes approximately *2*n*p* floating point operations (FLOPs)[1]. Training for the same model takes approximately *6*p* FLOPs per token (i.e., the additional backward pass requires four more operations[2]). You can approximate total training cost by multiplying this by the amount of tokens in the training data.

Memory requirements for transformers also depend on model size. For inference, we need the *p* model parameters to fit into memory. For learning (i.e., back-propagation), we need to store additional intermediate values per parameter between the forward and backward pass. Assuming we use 32-bit floating point numbers, this is an additional 8 bytes per parameter. For training a 175-billion-parameter model, we would need to keep over a terabyte of data in memory — this exceeds any GPU in existence today and requires us to split the model up across cards. Memory requirements for inference and training can be optimized by using floating point values of shorter lengths, with 16-bit becoming common and 8-bit anticipated in the near future.

Compute requirements for LLMs

Model	Parameters	Training Data	Training	Inference
BERT	340 million	3.3 million words	6.73^10^18 FLOPs	680 GFLOPs
GPT-J	6 billion	402 billion tokens	1.45*10^22 FLOPs	12 TFLOPs
GPT-3	175 billion	300 billion tokens	3.14*10^23 FLOPs	350 TFLOPs

Computational effort required for training (all training data) and inference (1024 tokens) for a number of popular text models.

a16z Enterprise

The table above has sizes and compute costs for several popular models. GPT-3 has approximately 175 billion parameters, which for an input and output of 1,024 tokens, results in a computational cost of approximately 350 trillion floating point operations (i.e., Teraflops or TFLOPs). Training a model like GPT-3 takes about 3.14*10^23 floating point operations. Other models like Meta's LLaMA have even higher compute requirements. Training such a model is one of the more computationally intensive tasks mankind has undertaken so far.

To summarize: AI infrastructure is expensive because the underlying algorithmic problems are extremely computationally hard. The algorithmic complexity of sorting a database table with a million entries is insignificant compared with the complexity of generating a single word with GPT-3. This means you want to pick the smallest model that solves your use case.

The good news is that, for transformers, we can easily estimate how much compute and memory a model of a certain size will consume. And, so, picking the right hardware becomes the next consideration.

The time and cost argument for GPUs

How does computational complexity translate to time? A processor core can typically execute 1-2 instructions per cycle, and processor clock rates have been stable around 3 GHz for the past 15 years due to the end of Dennard Scaling. Executing a single GPT-3 inference operation without exploiting any parallel architecture would take on the order of 350 TFLOPs/(3 GHz*1 FLOP) or 116,000 seconds, or 32 hours. This is wildly impractical; instead we need specialized chips that accelerate this task.

In practice, all AI models today run on cards that use a very large number of specialized cores. For example, an NVIDIA A100 GPU has 512 “tensor cores” that can perform a 4×4 matrix multiplication (which is equivalent to 64 multiplications and additions, or 128 FLOPs) in a single cycle. AI accelerator cards are often referred to as GPUs (graphics processing units), as the architecture was originally developed for desktop gaming. In the future we expect AI to increasingly become a distinct product family.

The A100 has a nominal performance of 312 TFLOPS which in theory would reduce the inference for GPT-3 to about 1 second. However this is an oversimplified calculation for several reasons. First, for most use cases, the bottleneck is not the compute power of the GPU but the ability to get data from the specialized graphics memory to the tensor cores. Second, the 175 billion weights would take up 700GB and won’t fit into the graphics memory of any GPU. Techniques such as partitioning and weight streaming need to be used. And, third, there are a number of optimizations (e.g., using shorter floating point representations, such as FP16, FP8, or sparse matrices) that are being used to accelerate computation. But, overall, the above math gives us an intuition of the overall computation cost of today’s LLMs.

Training a transformer model takes about three times as long per token as doing inference. However, given that the training data set is about 300 million times larger than an inference prompt, training takes longer by a factor of 1 billion. On a single GPU, training would take decades; in practice this is done on large compute clusters in dedicated data centers or, more likely, in the cloud. Training is also harder to parallelize than inference, as updated weights have to be exchanged between nodes. Memory and bandwidth between GPUs often becomes a far more important factor, with high-speed interconnects and dedicated fabrics being common. For training very large models, creating a suitable network setup can be the primary challenge. Looking into the future, AI accelerators will have networking capabilities on the card or even on the chip.

How does this computational complexity translate to cost? A GPT-3 inference, which, as we saw above, takes approximately 1 second on an A100 would have a raw compute cost between $0.0002 and $0.0014 for 1,000 tokens (this compares to OpenAI's pricing of $0.002/1000 tokens). A user generating 100 inference requests a day would cost in the order of dollars per year. This is a very low price point and makes most use cases of text-based AI by humans financially viable.

Training GPT-3, on the other hand, is much more expensive. Again calculating only the compute cost for 3.14*10^23 FLOPs at the above rates gives us an estimate of $560,000 on A100 cards for a *single training run*. In practice, for training we will not get nearly 100% efficiency in the GPU; however we can also use optimizations to reduce the training time. Other estimates of GPT-3 training cost range from $500,000 to $4.6 million, depending on hardware assumptions. Note that this is the cost of a single run and not overall cost. Multiple runs will likely be required and cloud providers will want long-term commitments (more on this below). **Training top-of-the-line models remains expensive, but within reach of a well-funded start-up.**

To summarize, generative AI requires massive investments in AI infrastructure today. There is no reason to believe that this will change in the near future. Training a model like GPT-3 is one of the most computationally intensive tasks mankind has ever undertaken. And while GPUs are getting faster, and we find ways to optimize training, the rapid expansion of AI negates both of these effects.

Considerations for AI infrastructure

To this point, we've tried to give you some intuition for the scale required to do training and inference of AI models, and what underlying parameters drive them. With that context, we now want to provide some practical guidance on how to decide which AI infrastructure to use.

External vs. in-house infrastructure

Let's face it: GPUs are cool. Many engineers and engineering-minded founders have a bias toward provisioning their own AI hardware, not only because it gives fine-grained control over model training, but because there's just something fun about harnessing large amounts of computing power (exhibit A).

The reality, however, is that **many startups — especially app companies — don't need to build their own AI infrastructure** on Day 1. Instead, hosted model services like OpenAI or Hugging Face (for language) and Replicate (for image generation) allow founders to search rapidly for product-market fit without the need to manage the underlying infrastructure or models.

These services have gotten so good that many companies never graduate from them. Developers can achieve meaningful control over model performance through prompt engineering and higher-order fine-tuning abstractions (i.e., fine tuning through API calls). Pricing for these services is consumption-based, so it's also often cheaper than running separate infrastructure. We've seen app companies generating more than $50 million of ARR, and valued over $1 billion, that run hosted model services under the hood.

On the flip side, some startups — especially **those training new foundation models or building vertically integrated AI applications — can't avoid running their own models directly** on GPUs. Either because the model is effectively the product and the team is searching for "model-market fit," or because fine-grained control over training and/or inference is required to achieve certain capabilities or reduce marginal cost at large scale. Either way, managing the infrastructure can become a source of competitive advantage.

The cloud vs. data center build out

In most cases, the cloud is the right place for your AI infrastructure. Less up-front cost, the ability to scale up and down, regional availability, and less distraction from building your own data center are compelling for most startups and larger companies.

But there are a few exceptions to this rule:

- If you are operating at a very large scale, it may become more cost effective to run your own data center. The exact price point varies based on geographic location and setup, but it typically requires infrastructure spend of more than $50 million per year.
- You need very specific hardware that you can't obtain from a cloud provider. For example, GPU types that are not widely available, as well as unusual memory, storage, or networking requirements.
- You cannot find a cloud that is acceptable for geopolitical considerations.

If you do want to build your own data center, there have been comprehensive price/performance analysis of GPUs for your own setup (e.g., Tim Dettmer's analysis). In addition to the cost and performance of the card itself, hardware selection also depends on power, space, and cooling. For example, two RTX 3080 Ti cards together have similar raw compute capacity to an A100, but respective power consumption is 700W vs. 300W. The 3,500 kWh power difference at market rates of $0.10/kWh over a three-year life cycle increases the cost of the RTX3080 Ti by nearly 2x (approximately $1,000).

All of this said, we expect the vast majority of startups to use cloud computing.

Comparing the cloud service providers

Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) all offer GPU instances, but new providers are also appearing to focus on AI workloads specifically. Here's a framework we've seen many founders use to choose a cloud provider:

Price: The table below shows pricing for a number of major and smaller specialty clouds as of April 7, 2023. This data is only indicative, as the instances vary considerably in terms of network bandwidth, data egress costs, additional cost from CPU and network, available discounts, and other factors. For example, Google requires an A2 accelerated-optimized instance for an A100 40GB, which can increase cost by 25%.

Available GPUs (by cloud provider)

Cloud Service Provider (CSP)	Available GPUs	Pricing Indicator (A100 40gb, on-demand)
AWS	A100, V100, M60, T4, A10, Trainium, Inferentia, Gaudi, V520	$4.10/h (p4d.24xlarge)
Azure	H100, A100, V100, P100, K80, M60, P40, T4, A10, MI25	$3.40/h (ND96asr A100 v4)
Oracle	A100, V100, P100, A10	$3.05/h (BM.GPU4.8)
Google	A100, V100, P100, K80, T4, P4, TPUv4	$2.93/h (NVIDIA A100 40GB)
Coreweave	H100, A100, V100, A40, A6000, A5000, A4000, Quadro RTX 4k/5k	$2.06/h (A100 40GB NVLINK)
Lambda Labs	A100, V100, A6000, A10, Quadro RTX 6k	$1.10/h (1x NVIDIA A100)
FluidStack	A100, V100, A40, A6000, A5000, QUADRO RTX 5K/4K, RTX 3090, RTX 3080, RTX 2080Ti	$1.73/h (A100 40 GB)

Source: Cloud provider websites

a16z Enterprise

Compute capacity on specific hardware is a commodity. Naively, we would expect fairly uniform prices, but this is not the case. And while substantial feature differences between the clouds exist, they are insufficient to explain that the pricing for an on-demand NVIDIA A100 varies by a factor of almost 4x among providers.

At the top end of the price scale, the big public clouds charge a premium based on brand reputation, proven reliability, and the need to manage a wide range of workloads. Smaller specialty AI providers offer lower prices, either by running purpose-built data centers (e.g., Coreweave) or arbitraging other clouds (e.g., Lambda Labs).

Practically speaking, most larger buyers negotiate prices directly with the cloud providers, often committing to some minimum spending requirement as well as minimum time commitments (we have seen 1-3 years). The price differences between clouds shrink somewhat after negotiation, but we've seen the ranking in the table above remain relatively stable. It's also important to note that smaller companies can get aggressive pricing from speciality clouds without large spending commitments.

Availability: The most powerful GPUs (e.g., Nvidia A100s) have been consistently in short supply for the past 12-plus months.

It would be logical to think the top three cloud providers have the best availability, given their large purchasing power and pool of resources. But, somewhat surprisingly, many startups haven't found that to be true. The big clouds have a lot of hardware but also have big customer needs to satisfy — e.g., Azure is the primary host for ChatGPT — and are constantly adding/leasing capacity to meet demand. Meanwhile, Nvidia has committed to making hardware available broadly across the industry, including allocations for new specialty providers. (They do this both to be fair and to reduce their dependence on a few large customers who also compete with them.)

As a result, many startups find more available chips, including the cutting-edge Nvidia H100s, at smaller cloud providers. If you're willing to work with a newer infrastructure company, you may be able to reduce wait times for hardware and possibly save money in the process.

Compute delivery model: The large clouds today only offer instances with dedicated GPUs, the reason being that GPU virtualization is still an unsolved problem. Specialized AI clouds offer other models, such as containers or batch jobs, that can handle individual tasks without incurring the start-up and tear-down cost of an instance. If you are comfortable with this model, it can substantially reduce cost.

Network interconnects: For training, specifically, network bandwidth is a major factor in provider selection. Clusters with dedicated fabrics between nodes, such as NVLink, are needed to train certain large models. For image generation, egress traffic fees can also be a major cost driver.

Customer support: Large cloud providers serve a huge pool of customers across thousands of product SKUs. It can be hard to get the attention of customer support, or get a problem fixed, unless you're a big customer. Many specialized AI clouds, on the other hand, offer fast and responsive support even for small customers. This is partly because they are operating at a smaller scale, but also because their

workloads are more homogenous — so they are more incentivized to focus on AI-specific features and bugs.

Comparing GPUs

All else being equal, the top-end GPUs will perform best on nearly all workloads. However, as you can see in the table below, the best hardware is also substantially more expensive. Picking the right type of GPU for your specific application can substantially reduce cost and may make the difference between a viable and nonviable business model.

Prices for common GPU types (by cloud provider)

A100 80 GB (per hour)			
CSP	**On Demand**	**Reserved (1y)**	**Spot/Preempt.**
AWS	$5.12	$3.00	n/a
Azure	$4.10	$3.53	$2.25
Oracle	$4.00	n/a	$2.00
Google	$3.93	n/a	$1.25
CoreWeave	$2.21	$1.80	n/a
Lambda Labs	$1.50	$1.36	n/a
FluidStack	$1.49	$1.19	n/a
Average	*$3.19*	*$2.27*	*$1.83*

A100 40 GB (per hour)			
CSP	**On Demand**	**Reserved (1y)**	**Spot/Preempt.**
AWS	$4.10	$2.40	$1.22
Azure	$3.40	$2.82	$2.04
Oracle	$3.05	n/a	$1.52
Google	$2.93	$1.03	$0.88
CoreWeave	$2.06	$1.77	n/a
FluidStack	$1.73	$1.44	n/a
Lambda Labs	$1.10	$1.05	n/a
Average	*$2.62*	*$1.75*	*$1.42*

A10 24 GB (per hour)*			
CSP	**On Demand**	**Reserved (1y)**	**Spot/Preempt.**
Azure	$3.20	$2.66	$1.28
Oracle	$2.00	n/a	$1.00
AWS	$1.00	$0.60	$0.49
Lambda Labs	$0.60	n/a	n/a
Average	*[N/A, small sample size]*		

Source: Cloud provider websites

a16z Enterprise

Deciding how far down the list to go — that is, determining the most cost-effective GPU choices for your application — is largely a technical decision that is beyond the scope of this article. But we'll share below some of the selection criteria we've seen are most important:

Training vs. inference: As we saw in the first section above, training a Transformer model requires us to store 8 bytes of data for training in addition to the model weights. This means a typical high-end consumer GPU with 12GB of memory could barely be used to train a 4-billion-parameter model. In practice, training large models is done on clusters of machines with preferably many GPUs per server, lots of VRAM, and high bandwidth connections between the servers (i.e., clusters built using top-end data center GPUs).

Specifically, many models will be most cost effective on the NVIDIA H100, but as of today it is hard to find and usually requires a long-term commitment of more than a year. The NVIDIA A100 runs most model-training today; it's easier to find but, for large clusters, may also require a long-term commitment.

Memory requirements: Large LLMs have parameter counts that are too high to fit in any card. They need to be split across multiple cards and require a setup similar to training. In other words, you probably need H100s or A100s even for LLM inference. But smaller models (e.g., Stable Diffusion) require much less VRAM. While the A100 is still popular, we have seen startups use the A10, A40, A4000, A5000 and A6000, or even RTX cards.

Hardware support: While the vast majority of workloads in companies that we have talked to run on NVIDIA, a few have started experimenting with other vendors. Most common is the Google TPU, but Intel's Gaudi 2 appears to be getting some traction, as well. The challenge with these vendors is that performance of your model is often highly dependent on the availability of software optimizations for these chips. You will likely have to do a PoC in order to understand performance.

Latency requirements: In general, less latency sensitive workloads (e.g., batch data processing or applications that don't require interactive UI responses) can use less-powerful GPUs. This can reduce compute cost by as much as 3-4x (e.g., comparing A100s to A10s on AWS). User-facing apps, on the other hand, often need top-end cards to deliver an engaging, real-time user experience. Optimizing models is often necessary to bring costs to a manageable range.

Spikiness: Generative AI companies often see dramatic spikes in demand since the technology is so new and exciting. It's not unusual to see request volumes increase by 10x in a day, based on a new product release, or grow 50% per week consistently. Handling these spikes is often easier on lower-end GPUs, since more compute nodes are likely available on demand. It often makes sense, too, to serve this kind of traffic with lower-cost resources — at the expense of performance — if it comes from less engaged or less retentive users.

Optimizing and scheduling models

Software optimizations can hugely affect the running time of models — and 10x gains are not uncommon. However, you'll need to determine which methods will be most effective with your particular model and system.

Some techniques work with a fairly broad range of models. Using shorter floating point representations (i.e., FP16 or FP8 vs. the original FP32) or quantization (INT8, INT4, INT2) achieve a speedup that is often linear with the reduction of bits. This sometimes requires modifying the model, but there are, increasingly, technologies available that automate working with mixed or shorter precision. Pruning neural networks reduces the number of weights by ignoring weights with low values. Together with efficient sparse matrix multiplication, this can achieve a substantial speedup on modern GPUs. Another set of optimization techniques addresses the memory bandwidth bottleneck (e.g., by streaming model weights).

Other optimizations are highly model-specific. For example, Stable Diffusion has made major advances in the amount of VRAM required for inference. Yet another class of optimizations is hardware-specific. NVIDIA's TensorRT includes a number of optimizations, but will only work on NVIDIA hardware. Last, but not least, scheduling of AI tasks can create huge performance bottlenecks or improvements. Allocating models to GPUs in a way to minimize swapping of weights, picking the best GPU for a task if multiple ones are available, and minimizing downtime by batching workloads in advance are common techniques.

In the end, model optimization is still a bit of a black art, and the majority of startups that we talk to work with third parties to help with some of these software aspects. Often, these are not traditional MLops vendors, but instead are companies that specialize in optimizations for specific generative models (e.g., OctoML or SegMind).

How will AI infrastructure cost evolve?

Over the last few years, we have seen exponential growth of both model parameters and GPU compute power. It is unclear if this trend will continue.

Today, it is widely accepted that there is a relationship between optimal number of parameters and the size of the training data set (see Deepmind's Chinchilla work for more on this). The best LLMs today are trained on the Common Crawl (a collection of 4.5 billion web pages, or about 10% of all web pages in existence). The training corpus also includes Wikipedia and a collection of books, although both are much smaller (the total number of books in existence is estimated to be only around 100 million). Other ideas, such as transcribing video or audio content, have been

suggested, but none of these come close in size. It is not clear if we could obtain a non-synthetic training dataset that is 10x larger than what has already been used.

GPU performance will continue to increase, but also at a slower rate. Moore's Law is still intact allowing for more transistors and more cores, but power and I/O are becoming limiting factors. Additionally, many of the low-hanging fruit for optimizations have been picked.

However, none of this means we don't expect an increase in demand for compute capacity. Even if model and training set growth slows, the growth of the AI industry and increase in the number of AI developers will fuel a demand for more and faster GPUs. A large fraction of GPU capacity is used for testing by developers during the development phase of a model, and this demand scales linearly with headcount. **There is no sign that the GPU shortage we have today will abate in the near future.**

Will this continued high cost of AI infrastructure create a moat that makes it impossible for new entrants to catch up with well-funded incumbents? We don't know the answer to this question yet. The training cost of an LLM may look like a moat today, but open source models such as Alpaca or Stable Diffusion have shown that these markets are still early and may change quickly. Over time, the cost structure of the emerging AI software stack (see our previous post) may start looking more like the traditional software industry.

Ultimately, this would be a good thing: History has shown that this leads to vibrant ecosystems with rapid innovation and lots of opportunities for entrepreneurial founders.