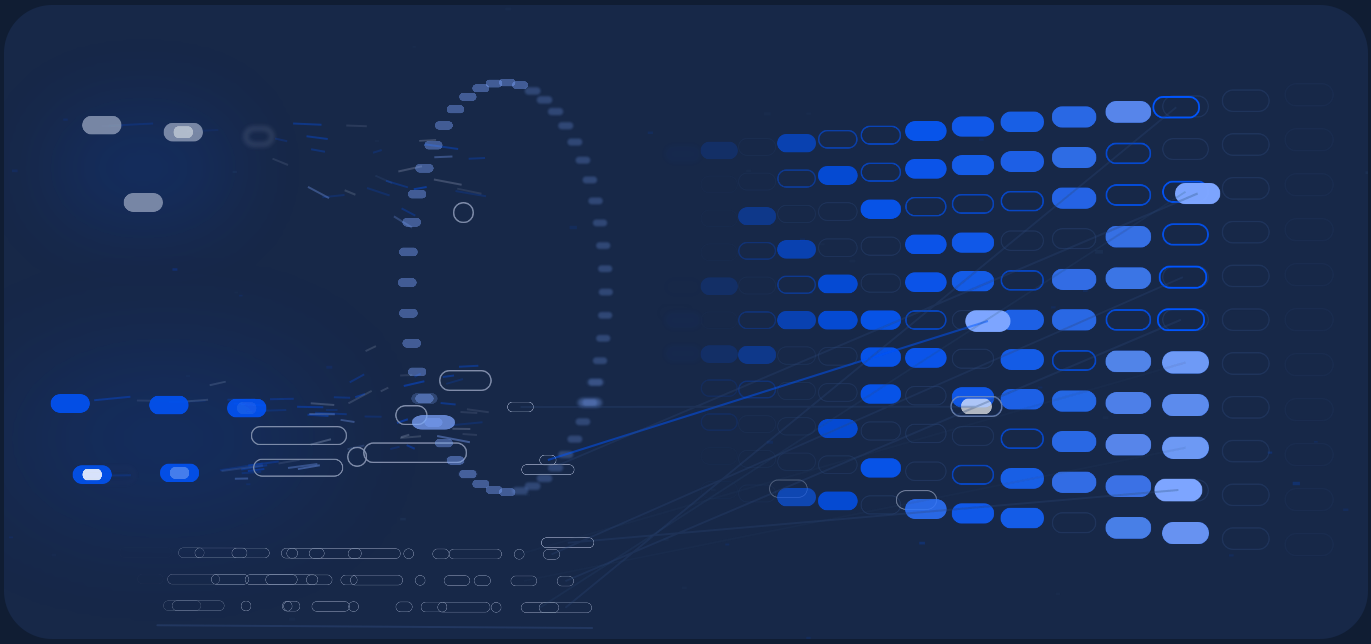
Rillor

METHODS NOTE

From GPU-hour price to cost per training token

How to calculate LLM training cost from a GPU-hour price: compute per token, dense peak FLOPS, utilization in published runs and a worked example with dated public prices.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

A GPU-hour price becomes a cost per training token through four inputs: the compute each token needs, a dense peak rate, the utilization a comparable run reached and the time lost to interruptions.

- Training a dense transformer takes about 6 floating-point operations per parameter for each training token, 2 in the forward pass and 4 in the backward pass [1, 3, 19]. A fuller count by the Chinchilla authors came within 10% of that estimate for models of 73 million to 6.8 billion parameters [2].
- Cost per million training tokens is the price per GPU-hour divided by the millions of tokens one GPU trains in an hour. DeepSeek-V3's report of 180K H800 GPU hours per trillion tokens, at the $2 per GPU hour it assumes, gives $0.36 per million training tokens (this note's arithmetic) [11].
- NVIDIA's H100 page prints its BF16, FP16 and FP8 tensor rates with sparsity, and the dense rate is half of each [20, 22, 23, 24]. Training papers report the share of peak a run reaches against dense peaks [3, 5, 6], so dividing by the sparse figure halves an estimated cost per token with no change in the hardware (this note's arithmetic).
- Model FLOPs utilization (MFU), the share of a system's peak rate that a run turns into model computation, was 38% to 43% for a 405-billion-parameter model on up to 16,384 H100 GPUs [7] and 54.2% for a 530-billion-parameter model on 2,240 A100 GPUs [5]. One A100 run reads as about 40% of peak in its own paper (this note's arithmetic) and as 30.2% MFU in another paper's count [3, 6].
- For a 70-billion-parameter dense model trained on 15 trillion tokens at NVIDIA's measured FP8 rate of 1,664 tokens per second per H100, dated public prices give $0.98 per million tokens at a Lambda 64-GPU cluster rate and $1.15 at AWS on-demand rates, about $14.6 million and $17.2 million before lost time (this note's arithmetic) [26, 30, 31, 32].
- The GPU-hour price leaves out interruptions, experiments and staff. Llama 3 405B pre-training had 466 job interruptions in 54 days and kept effective training time above 90% [7], and Cottier and colleagues sample total development compute at 1.2 to 4 times the final run [15].
- In one NVIDIA benchmark of a mixture-of-experts model, B200 trained 2.33 times as many tokens per second per GPU as H100. At Lambda's listed rates that made B200's cost per token 28% to 31% lower, and at AWS on-demand rates 11% lower (this note's arithmetic) [26, 30, 31, 32].

Background

A GPU-hour price is the cost of holding one GPU for one hour. A training budget needs the cost of the work done in that hour, which depends on the compute each token needs, the share of the GPU's peak the software reaches and the hours lost to failures. This note works the conversion for ML leads, research engineers and finance teams who estimate a pre-training budget or compare GPU offers by the cost of useful work.

Two ways to price a run. Cottier and colleagues estimate training costs with a cloud method, "Total cost = Price per chip-hour × Training chip-hours", and an amortized method that spreads a chip's starting value over the training hours and adds energy [15]. The two "produced consistent estimates of the growth rate", but "the cloud costs were twice as large on average" [15]. The authors expect the cloud method to overestimate frontier model costs, "since model developers usually either own or have private rental agreements for their training hardware" [15]. This note follows the cloud method because it starts from a posted price. An owner replaces the price with an amortized hourly cost and keeps every other step.

What training benchmarks measure. MLPerf® Training records "the wall clock time required to train a model on the specified dataset to achieve the specified quality target" [27]. Round v5.0, published 4 June 2025, added a Llama 3.1 405B pretraining benchmark in place of the GPT-3-based one [28]. A time to a quality target compares systems on a fixed piece of work, which differs from the cost of a full pre-training run.

Terms. A GPU-hour is one GPU for one hour, so an 8-GPU instance price is divided by 8 [32]. Tokens are counted by the model's own tokenizer [7]. MFU, defined in Finding 3, is a share of peak arithmetic. It differs from the utilization that rental analyses use for the share of hours a GPU is rented. Inference cost is out of scope.

Method

What was examined. The formulas that turn parameters and tokens into training compute, NVIDIA's peak rates, the definitions of model and hardware FLOPs utilization, the throughput and GPU-hours reported for published runs, and dated public prices for H100 and B200 capacity.

Sources. Thirty-two sources, each opened on 8 November 2026 and [31] again on 12 November 2026: fifteen papers or preprints, one model card, three publications by research groups, five NVIDIA documents, two pages of NVIDIA software documentation, two MLCommons pages, one cloud provider blog post and three price and specification sources. The throughput tables come from NVIDIA's own recipes on its own DGX systems [25, 26].

Steps.

1. Training FLOPs are about 6 × N × D, with N the parameter count (the active count for a mixture-of-experts model) and D the training tokens [1, 19].
2. Tokens per second per GPU are about MFU × dense peak FLOP/s ÷ 6N [3].
3. Tokens per GPU-hour are 3,600 × tokens per second.
4. Cost per million tokens is the price per GPU-hour ÷ millions of tokens per GPU-hour, the cloud method of [15] taken per token.
5. GPU-hours are D ÷ tokens per GPU-hour, divided by the share of time spent on useful training [7, 16].

A measured tokens-per-second figure, where one exists, replaces steps 1 and 2, and the note shows both routes. Figures marked "this note's arithmetic" are calculations on the cited numbers.

Prices. Lambda's page, viewed 8 November 2026, states under each instance table that "These prices take effect November 16, 2026, at 00:00 UTC", and adds sales tax, VAT or GST [30]. Its 1-Click Cluster rates are used as listed that day. AWS prices come from its on-demand file for US East (N. Virginia) and Linux, published 10 November 2026 at 17:38:52 UTC [31], divided by the 8 GPUs each instance holds [32]. No provider is ranked, and no figure is a forecast.

Where sources disagree. BLOOM's paper reports 1,082,990 compute hours and up to 156 TFLOPs per GPU [10]. Cottier and colleagues report 1,161,261 A100-hours and 150 TFLOP/GPU/s at 48% MFU [15]. Llama 3's paper gives 15.6 trillion tokens for the 405B model [7], and the Llama 3.1 card "15T+" [8]. Each figure is cited to its own source.

Findings

1. Compute per training token: 6N, 6ND and the attention term

The 6N rule. Kaplan and colleagues count a forward pass as about 2N operations, since "the factor of two comes from the multiply-accumulate operation", and "Accounting for the backwards pass (approximately twice the compute as the forwards pass)" estimate about 6N floating-point operations per training token, with N the non-embedding parameters [1]. They drop a context term that is small when the model width exceeds one-twelfth of the context length [1]. A run of D tokens needs about 6ND.

How close it comes. The Chinchilla authors compared their full count, embeddings included, with C = 6DN. The ratio ran from 0.99 at 6.8 billion parameters to 1.10 at 305 million, and they "find the differences in FLOP calculation to be very small" [2]. EleutherAI's guide writes the same budget as $C \approx \tau T = 6PD$, with τ the GPUs times the actual FLOPs per GPU and T the seconds [19]. Epoch AI's method multiplies one forward pass by 3 and the passes over the data, and states that "This formula is a heuristic" [17]. Its companion study finds backward-forward ratios "close to 2:1" in classic settings and from 1:1 to 3:1 by layer type, batch size and depth [18].

Attention. PaLM's authors count "6N matmul FLOPs per token seen (2N for forward and 4N for backward)" plus 6LH(2QT) for dense self-attention, for L layers, H heads, head size Q and sequence length T, so peak tokens per second are peak FLOP/s ÷ (6N + 12LHQT) [3]. Llama 3 405B has 126 layers, a model dimension of 16,384 and 128 heads [7], a head size of 128. At 8,192 tokens the attention term is 2.03 × 10^11 FLOPs per token, 8.4% on top of 6N's 2.43 × 10^12. At 131,072 tokens it is 3.25 × 10^12, 134% on top (this note's arithmetic from [3, 7]), counting full attention as PaLM's formula does. Llama 3's 131,072-token stage ran at 380 TFLOPs per GPU, against 400 for 8,192-token sequences on the same 16,384 GPUs [7].

Megatron-LM's count. Narayanan and colleagues count matrix multiplications only, $F = 96Bslh^2(1 + s/6h + V/16lh)$ per iteration, "a lower bound for the true FLOP count" that "assumes activation recomputation" and so includes an extra forward pass [4]. Rates on this count are hardware rates, above MFU for the same run.

Mixture of experts. Such models select "different parameters for each incoming example", for "a constant computational cost" per token [12]. DeepSeek-V3 has "671B total parameters with 37B activated for each token" [11], and Mixtral uses 13 billion of its 47 billion parameters per token during inference [13]. 6N on the active count is an approximation that leaves routing work uncounted [11, 12].

Tokens and D. Llama 3's tokenizer raised compression on English data from 3.17 to 3.94 characters per token [7], so a cost per token compares runs only when tokenizers are comparable. The Chinchilla authors found that "for every doubling of model size the number of training tokens should also be doubled", with 1.5 trillion tokens for a 67-billion-parameter model [2]. Llama 3 405B used 15.6 trillion tokens and 3.8 × 10^25 FLOPs [7], and 6 × 405 billion × 15.6 trillion is 3.79 × 10^25 (this note's arithmetic). The cost scales with D.

2. Peak rates on spec sheets, dense and sparse

What the sheets print. NVIDIA's H100 page lists the SXM part at "BFLOAT16 Tensor Core * 1,979 teraFLOPS", 1,979 for FP16 and 3,958 for FP8, with the footnote "* With sparsity", 80GB of memory and a TDP of up to 700W [20]. The A100 datasheet of June 2021 prints both figures, "BFLOAT16 312 TFLOPS | 624 TFLOPS*" [21]. The Blackwell datasheet lists HGX B200 at 4.5 PFLOPS for FP16 and BF16 and 9 for FP8 per GPU, and states: "Specifications in sparse. Dense is one-half of the sparse spec shown" [22]. NVIDIA's preliminary H100 table at launch listed "Peak BF16 Tensor Core 1000 TFLOPS | 2000 TFLOPS", the second with sparsity [24].

What sparsity requires. "In each contiguous block of four values, two values must be zero", and the hardware can then "complete the same effective calculation in half the time" [23]. NVIDIA's sparse ResNeXt-101 test on A100 found "performance improvements up to 20% at larger batch sizes" for inference [23]. The training papers below measure against dense peaks, so this note uses 989.5 TFLOP/s for H100 SXM at BF16, 1,979 at FP8 and 2,250 for B200 at BF16 (this note's arithmetic, half of [20, 22]).

What papers divide by. Korthikanti and colleagues state that "For NVIDIA A100 GPUs, the peak theoretical FLOPs per second is 312 teraFLOPs/sec" [5], and the MT-NLG authors use the same 312 at 16-bit precision [6]. PaLM's authors use 312 for A100 and 275 for TPU v4 [3], and BLOOM's authors aimed at half of "312 TFLOPs (in float32 or bfloat16)" [10]. Llama 3's Table 4 pairs 430, 400 and 380 TFLOPs per GPU with 43%, 41% and 38% MFU [7], which gives 1,000, 976 and 1,000 TFLOP/s, the dense BF16 rate (this note's arithmetic).

The size of the trap. At 40% MFU, a 70-billion-parameter model on the dense 989.5 TFLOP/s trains 942 tokens per second per GPU, $1.72 per million tokens at $5.85 per GPU-hour. On 1,979 the same 40% gives 1,885 tokens per second and $0.86 per million, half the cost with no change in the hardware (this note's arithmetic from [20, 30]).

FP8 has no single denominator. TorchTitan's authors "do not report Model FLOPS Utilization (MFU)" with Float8 on, because BF16 and FP8 Tensor Cores both take part "but they have different peak FLOPS and the definition of MFU under such scenario is not well-defined" [14]. NVIDIA's FP8 recipe for Llama 3 70B on 64 H100s reports 731 model TFLOP/s per GPU [26], which is 36.9% of the dense FP8 peak and 73.9% of the dense BF16 peak (this note's arithmetic from [20, 26]).

3. Model FLOPs utilization in published runs

Definitions. PaLM's authors define MFU as "the ratio of the observed throughput (tokens-per-second) relative to the theoretical maximum throughput of a system operating at peak FLOPs", a maximum that "only accounts for the required operations to compute the forward+backward passes, and not rematerialization" [3]. Hardware FLOPs utilization (HFU) also counts recomputation, and they call it "not a consistent and meaningful metric" because it depends on the implementation [3]. Korthikanti and colleagues put the ratio of hardware to model FLOPs at "approximately 1 + s/6h" [5]. Google Cloud's goodput post says its Program Goodput is "also referred to as Model Flop Utilization" [29].

Run and source	Hardware and scale	Reported
Llama 3 405B, 8,192-token sequences [7]	8,192 H100	430 TFLOPs per GPU, 43% BF16 MFU
Llama 3 405B, 8,192-token sequences [7]	16,384 H100	400 TFLOPs, 41%
Llama 3 405B, 131,072-token sequences [7]	16,384 H100	380 TFLOPs, 38%
530B model, Korthikanti et al. [5]	2,240 A100	54.2% MFU, 42.1% with recomputation
1T model, Megatron-LM [4]	3,072 A100	163 TFLOP/s per GPU, 52% of peak, hardware count
MT-NLG 530B [6]	2,240 A100	126 TFLOP/s per GPU, a count with recomputation (this note's arithmetic from [4, 6])
BLOOM 176B [10]	384 A100	156 TFLOPs in its fastest configuration, half of 312
Llama 3.1 8B, TorchTitan [14]	8 or 128 H100, no Float8	33% to 42% MFU
PaLM 540B (TPU, not GPU) [3]	6,144 TPU v4	46.2% MFU, 57.8% HFU

Scale and batch. Llama 3's drop from 43% to 41% "is due to the lower batch size per DP group", at 16M tokens per batch in each configuration [7]. Korthikanti's 530B model fell from 56.0% to 54.2% MFU at 8-way data parallelism, with iteration time rising from 37.83 to 39.15 seconds [5]. Without data parallelism, their MFU rose with model size, from 41.5% at 22 billion parameters to 56.3% at 1 trillion [5]. Megatron-LM found that recomputation "leads to up to 33% lower throughput" at small batch sizes, while at large ones throughput with it was up to 2 times that of runs without it [4].

One run, two numbers. MT-NLG's iteration time of 60.1 seconds on 280 DGX A100 servers corresponds to 126 teraFLOP/s per GPU [6], 40.4% of 312 (this note's arithmetic). PaLM's authors list the same model at 30.2% MFU, from 65.43K tokens per second on 2,240 GPUs, and compute "(65.43 × 6 × 530)/(312 × 2240)" as 29.7% without attention [3]. The gap is the counting rule.

PaLM on TPU. PaLM 540B reached 238.3K tokens per second on 6,144 TPU v4 chips, 45.7% MFU without self-attention and 46.2% with it [3], and 238.3K × 6 × 540 billion ÷ (275 TFLOP/s × 6,144) is 45.7% (this note's arithmetic). These are TPU figures. The same table lists GPT-3 175B on V100 at 21.3% [3].

Rules of thumb. Epoch AI suggested in January 2022 "a utilization rate of 0.3 for large language models, and a utilization rate of 0.4 for other networks" [17]. EleutherAI wrote in April 2023 that "you should always be able to achieve approximately 120 TFLOP/s/A100", with GPT-NeoX at 150 and, with Flash Attention, 180 [19]. Each is a dated figure for one generation of software.

Measured throughput. NVIDIA's Megatron Bridge pages report "Tokens/sec/GPU" and "Model TFLOP/sec/GPU" from its performance recipes, and balance tokens among experts in mixture-of-experts runs [25]. From the 26.04.01 NeMo container section, pre-training [26]:

Model, system, GPUs, precision	Tokens/s/GPU	Model TFLOP/s/GPU
Llama 3 70B, DGX-H100, 64, FP8	1,664	731
Llama 3 70B, DGX-GB200, 64, FP8	4,224	1,892
Llama 3.1 405B, DGX-H100, 1,024, FP8	328	827
GPT OSS 120B, DGX-H100, 64, BF16	5,824	158
GPT OSS 120B, DGX-B200, 64, BF16	13,568	369

6N gives 699 TFLOP/s for the 70B row and 797 for the 405B row, and NVIDIA's model figures are 4.6% and 3.8% higher (this note's arithmetic from [26]). Tokens per second fall with model size while model TFLOP/s per GPU stays near 731 to 827 [26], so cost per token tracks the parameter count. At $5.85 per GPU-hour, the 405B row is $4.95 per million tokens (this note's arithmetic from [26, 30]).

TorchTitan's H100s are not standard: "The actual peak TFLOPs should be between SXM and NVL, and we don't know the exact value" [14]. Read "at the (arbitrarily determined) 90th iteration", its Llama 3.1 8B run on 128 GPUs went from 5,645 tokens per second per GPU with FSDP to 9,319 with torch.compile and Float8 [14].

4. GPU-hours and tokens from published runs

Published GPU-hours give a route with no MFU at all: training FLOPs divided by reported GPU-seconds give an average rate over whatever the count includes.

Run	Tokens	GPU-hours	Average rate (this note's arithmetic)
Llama 3 405B	15.6T [7]	30.84M H100 [8]	342 TFLOP/s, 34.6% of 989.5
Llama 3.1 70B	"15T+" [8]	7.0M H100 [8]	at least 250 TFLOP/s
Llama 3.1 8B	"15T+" [8]	1.46M H100 [8]	at least 137 TFLOP/s
Llama 2 70B	2.0T [9]	1,720,320 A100 [9]	136 TFLOP/s, 43.5% of 312
BLOOM 176B	366B [10]	1,082,990 [10]	99 TFLOP/s, 31.8% of 312
DeepSeek-V3, 37B active	1T [11]	180K H800 [11]	343 TFLOP/s by 6N on active parameters

Llama. The 405B rate divides the paper's 3.8 × 10^25 FLOPs [7] by the model card's hours, which are "the total GPU time required for training each model", 39.3M GPU hours in all on H100-80GB [8]. The gap to the 380 to 430 TFLOPs reported during the run [7] includes whatever that total covers beyond pre-training steps. The 70B and 8B rates are lower bounds because the card gives "15T+" tokens [8]. Llama 2's table lists pretraining GPU time on A100-80GB, 3,311,616 hours across four sizes, all trained on 2.0 trillion tokens [9]. For 70B that is 1.16 million tokens per A100-hour (this note's arithmetic).

BLOOM. Training "took about 3.5 months to complete and consumed 1,082,990 compute hours" on 48 nodes of 8 A100 80GB GPUs [10]. Its average of 99 TFLOP/s sits against 156 in its fastest configuration [10] and 150 reported for the run [15], and with Cottier's 1,161,261 A100-hours it is 92.5 (this note's arithmetic from [10, 15]).

DeepSeek-V3. Its report states that "training DeepSeek-V3 on each trillion tokens requires only 180K H800 GPU hours, i.e., 3.7 days on our cluster with 2048 H800 GPUs" [11]. Table 1 lists 2,664K GPU hours for pre-training, 119K for context extension and 5K for post-training, 2,788K and $5.576M in all, "assuming the rental price of H800 is $2 per GPU hour" [11]. Pre-training used 14.8 trillion tokens in FP8 mixed precision [11], and 14.8 × 180K is the table's 2,664K. That is 5.56 million tokens per H800-hour and $0.36 per million at the assumed price (this note's arithmetic from [11]).

Megatron-LM's time formula. "End-to-end training time ≈ 8TP/(nX)", the 8 counting recomputation [4]. For GPT-3 175B on 300 billion tokens on 1,024 A100s at 140 teraFLOP/s per GPU it gives 34 days [4], or 33.9 days and 0.83 million GPU-hours (this note's arithmetic).

5. A worked example with dated public prices

Inputs. A 70-billion-parameter dense model trained on 15 trillion tokens, the scale of the Llama 3.1 card's "~15 trillion" [8], on H100 GPUs at NVIDIA's measured 1,664 tokens per second per GPU for Llama 3 70B at FP8 on 64 H100s [26], or 5.99 million tokens per GPU-hour (this note's arithmetic).

Offer	Per GPU-hour
AWS p5.48xlarge, 8 x H100, on demand, US East (N. Virginia), $55.04 an hour, file of 10 November 2026 [31, 32]	$6.88
Lambda 1-Click Cluster, 64 H100 GPUs, terms of 2 weeks to 1 year [30]	$5.85
Lambda 1-Click Cluster, 256 H100 GPUs [30]	$5.54
Lambda 8x NVIDIA HGX H100 80 GB instance, rates from 16 November 2026 [30]	$4.59

For terms of a year or more the page says "Talk to our team" and shows no price [30].

Price per GPU-hour	Per million tokens	15T tokens	At 90% effective training time
$6.88, AWS on demand	$1.15	$17.2M	$19.1M
$5.85, Lambda 64-GPU cluster	$0.98	$14.6M	$16.3M
$5.54, Lambda 256-GPU cluster	$0.92	$13.9M	$15.4M
$4.59, Lambda 8x instance, from 16 November 2026	$0.77	$11.5M	$12.8M

Arithmetic: this note's, from [26, 30, 31, 32]. The run takes 2.50 million GPU-hours. The last column divides by Llama 3's "higher than 90%" effective training time [7], close to the "around 0.9" that Kokolis and colleagues estimate for runs of two days or more on 2,048 to 4,096 GPUs of one cluster, with assumed checkpoint and restart overheads [16].

Wall time. On 64 GPUs the run would take about 1,630 days, and on 256 GPUs about 408 days, past the listed term of up to 1 year (this note's arithmetic from [30]). A real run uses more GPUs, where per-GPU throughput can fall, as Llama 3's MFU did from 43% to 41% when its GPU count doubled [7].

Formula route. At BF16 on the dense 989.5 TFLOP/s, Llama 3's reported 38% and 43% MFU [7] and a middle value of 40% give 895, 942 and 1,013 tokens per second per GPU: $1.82, $1.72 and $1.60 per million tokens at $5.85, and $2.13, $2.03 and $1.89 at $6.88. At 40%, 15 trillion tokens take 4.42 million GPU-hours, $25.9 million at $5.85 before lost time (this note's arithmetic from [7, 20, 30, 31]). The FP8 measurement is about 1.8 times the 40% BF16 rate, more than the 1.5 spread between the four prices, so precision moves the estimate more than the choice of price does (this note's arithmetic from [20, 26, 30, 31]).

Whole project. Cottier and colleagues sample a multiplier "with a 90% CI of 1.2x to 4x, meaning that total compute for model development is 1.2x to 4x larger than the final training run", median 2.2 times [15]. A final-run estimate is the low end of a project budget.

6. What the formula leaves out

Interruptions. "During a 54-day snapshot period of pre-training, we experienced a total of 466 job interruptions", 419 of them unexpected, and about 78% of the unexpected interruptions were "attributed to confirmed or suspected hardware issues" [7]. The team "achieved higher than 90% effective training time", "the time spent on useful training over the elapsed time" [7].

Reliability at scale. Kokolis and colleagues define effective training time ratio (ETTR) as "the ratio of productive runtime to available wallclock time of a job run", with losses from catching up after the last checkpoint, restarts and checkpoint writes [16]. Over "11 months of data from two state-of-the-art ML environments with 4 million jobs and over 150 million A100 GPU hours", the mean time to failure was 7.9 hours for 1,024-GPU jobs and 47.7 days for 8-GPU jobs [16]. For about 12,000 GPUs to reach an ETTR of 0.9, checkpoint writes need to take about 10 seconds or failures need to fall sharply [16].

Goodput. Google Cloud splits ML Productivity Goodput into Scheduling Goodput, below 100% in on-demand or preemptible models "because of potential stockouts", Runtime Goodput and Program Goodput [29]. The post is a vendor's, dated 10 April 2024 [29].

Storage and transfer. Llama 3's storage "offers 240 PB of storage out of 7,500 servers equipped with SSDs", with checkpoints of "from 1 MB to 4 GB per GPU" [7]. A cloud bills storage and transfer apart from the GPU-hour, and this note prices neither.

Experiments. Cottier and colleagues estimate that for GPT-3 "at least 14% of compute was spent outside the main training run", and that BLOOM's final run "used 37.24% of the energy of the BLOOM project" [15]. DeepSeek-V3's costs exclude "prior research and ablation experiments on architectures, algorithms, or data" [11].

Staff and the cluster. Across four models' full development, R&D staff including equity made up 29% to 49% of total amortized cost, computing hardware 47% to 64% and energy 2% to 6%, and the hardware figure includes "23% overhead added for cluster-level networking" [15].

Taxes. Lambda's prices carry "applicable sales tax/VAT/GST" [30], and the AWS figures are list prices for one region and operating system [31].

7. Comparing GPU classes by cost of work

Inputs. For the GPT OSS 120B mixture-of-experts model at BF16 on 64 GPUs, with a global batch of 1,280 and 4,096-token sequences, NVIDIA reports 13,568 tokens per second per GPU on DGX-B200 (micro-batch 4, pipeline parallelism 1, expert parallelism 8) and 5,824 on DGX-H100 (micro-batch 1, pipeline parallelism 4, expert parallelism 8) [26]. The measured ratio is 2.33, and the dense BF16 peak ratio, 2,250 against 989.5 TFLOP/s, is 2.27 (this note's arithmetic from [20, 22, 26]).

Prices. Lambda lists 8x H100 at $4.59 and 8x HGX B200 180 GB at $7.69 per GPU-hour from 16 November 2026, a ratio of 1.675, and 64-GPU clusters at $5.85 and $9.36, a ratio of 1.60 [30]. AWS lists p5.48xlarge at $55.04 and p6-b200.48xlarge, with 8 B200 GPUs, at $113.9328 an hour, $6.88 and $14.24 per GPU-hour, a ratio of 2.07 [31, 32].

Price basis	H100 per million tokens	B200 per million tokens	B200 difference
Lambda 8x instances, from 16 November 2026	$0.219	$0.157	28% lower
Lambda 64-GPU clusters	$0.279	$0.192	31% lower
AWS on demand, 10 November 2026	$0.328	$0.292	11% lower

Arithmetic: this note's, from [26, 30, 31, 32].

Reading the table. On this recipe, B200 costs less per token than H100 wherever its price per GPU-hour is below 2.33 times H100's (this note's arithmetic). The same throughput gap gives 28% to 31% on one provider's prices and 11% on another's, so the price ratio decides as much as the hardware does. For Llama 3 70B at FP8, GB200 trained 2.54 times H100's tokens per second per GPU (this note's arithmetic from [26]).

Implications

For ML leads and research engineers. Start from a measured tokens-per-second figure for the same model size, precision and GPU count [25, 26], and check it with the formula route [3]. Name the peak used, its precision and whether it is dense. Add the attention term for long sequences [3, 7], use active parameters for mixture-of-experts models [11, 12], and plan for about a tenth of the time lost, as in published large runs [7, 16].

For finance and procurement teams. A budget built from a GPU-hour price covers the final run's compute. Cottier and colleagues sample development compute at 1.2 to 4 times the final run, put R&D staff, including equity, at 29% to 49% of total amortized cost, and found cloud-method costs about twice the amortized estimates [15]. Every price carries a date, and some an effective date, as Lambda's instance rates from 16 November 2026 do [30].

For comparing GPU offers. Compare cost per million training tokens on one recipe, setting the price ratio against the measured throughput ratio [26, 30, 31]. A sparse peak in the denominator halves the cost on paper [20, 22], and an FP8 utilization figure needs its peak named [14].

Rillor's GPU and neocloud pricing method brings each instance or term price to U.S. dollars per GPU-hour, one GPU for one hour, and keeps the price exactly as the provider states it on record beside it.

Limits of this analysis

- The throughput tables are NVIDIA's own recipes on its own DGX systems [25, 26], and TorchTitan's H100s have an unknown peak [14]. None is a measurement on rented cloud capacity.
- The worked example covers one model size and one recipe measured on 64 GPUs. Per-GPU throughput at full-run scale can differ [5, 7].
- 6N on active parameters is an approximation for mixture-of-experts models [11, 12].
- The attention term counts full attention as PaLM's formula does [3]. No source opened for this note gives the effect of causal masking or fused attention kernels.
- Prices come from two providers: AWS on-demand list prices for one region and operating system [31], and Lambda's listed rates, whose instance rates take effect on 16 November 2026 [30]. Spot, reserved and other providers' prices are outside this note, and GB200 is compared on throughput only.
- The model card's GPU-hours are not split by training stage [8].
- No MLPerf® Training result was used. Energy, storage and staff costs per token are not computed.

Sources

1. Scaling Laws for Neural Language Models. Jared Kaplan, Sam McCandlish, Tom Henighan and others. arXiv 2001.08361, 23 January 2020, viewed 8 November 2026. https://arxiv.org/abs/2001.08361

2. Training Compute-Optimal Large Language Models. Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch and others. arXiv 2203.15556, 29 March 2022, viewed 8 November 2026. https://arxiv.org/abs/2203.15556

3. PaLM: Scaling Language Modeling with Pathways. Aakanksha Chowdhery, Sharan Narang, Jacob Devlin and others. arXiv 2204.02311, version 5, 5 October 2022, viewed 8 November 2026. https://arxiv.org/abs/2204.02311

4. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. Deepak Narayanan, Mohammad Shoeybi, Jared Casper and others. SC 2021, arXiv 2104.04473, version 5, 23 August 2021, viewed 8 November 2026. https://arxiv.org/abs/2104.04473

5. Reducing Activation Recomputation in Large Transformer Models. Vijay Korthikanti, Jared Casper, Sangkug Lym and others. arXiv 2205.05198, 10 May 2022, viewed 8 November 2026. https://arxiv.org/abs/2205.05198

6. Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model. Shaden Smith, Mostofa Patwary, Brandon Norick and others. arXiv 2201.11990, version 3, 4 February 2022, viewed 8 November 2026. https://arxiv.org/abs/2201.11990

7. The Llama 3 Herd of Models. Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri and others. arXiv 2407.21783, version 3, 23 November 2024, viewed 8 November 2026. https://arxiv.org/html/2407.21783v3

8. Llama 3.1 model card (MODEL_CARD.md). Meta, meta-llama/llama-models repository on GitHub, last changed 30 September 2024, viewed 8 November 2026. https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md

9. Llama 2: Open Foundation and Fine-Tuned Chat Models. Hugo Touvron, Louis Martin, Kevin Stone and others. arXiv 2307.09288, version 2, 19 July 2023, viewed 8 November 2026. https://arxiv.org/abs/2307.09288

10. BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. BigScience Workshop. arXiv 2211.05100, version 4, 27 June 2023, viewed 8 November 2026. https://arxiv.org/abs/2211.05100

11. DeepSeek-V3 Technical Report. DeepSeek-AI. arXiv 2412.19437, version 2, 18 February 2025, viewed 8 November 2026. https://arxiv.org/abs/2412.19437

12. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. William Fedus, Barret Zoph and Noam Shazeer. arXiv 2101.03961, version 3, 16 June 2022, viewed 8 November 2026. https://arxiv.org/abs/2101.03961

13. Mixtral of Experts. Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux and others. arXiv 2401.04088, 8 January 2024, viewed 8 November 2026. https://arxiv.org/abs/2401.04088

14. TorchTitan: One-stop PyTorch native solution for production ready LLM pre-training. Wanchao Liang, Tianyu Liu, Less Wright and others. arXiv 2410.06511, version 3, 7 June 2025, viewed 8 November 2026. https://arxiv.org/abs/2410.06511

15. The rising costs of training frontier AI models. Ben Cottier, Robi Rahman, Loredana Fattorini and others. arXiv 2405.21015, version 2, 7 February 2025, viewed 8 November 2026. https://arxiv.org/abs/2405.21015

16. Revisiting Reliability in Large-Scale Machine Learning Research Clusters. Apostolos Kokolis, Michael Kuchnik, John Hoffman and others. HPCA 2025, arXiv 2410.21680, version 2, 6 February 2025, viewed 8 November 2026. https://arxiv.org/abs/2410.21680

17. Estimating training compute of deep learning models. Jaime Sevilla, Lennart Heim, Marius Hobbhahn, Tamay Besiroglu, Anson Ho and Pablo Villalobos. Epoch AI, 20 January 2022, viewed 8 November 2026. https://epoch.ai/publications/estimating-training-compute

18. What's the backward-forward FLOP ratio for neural networks? Marius Hobbhahn and Jaime Sevilla. Epoch AI, 13 December 2021, viewed 8 November 2026. https://epoch.ai/publications/backward-forward-FLOP-ratio

19. Transformer Math 101. Quentin Anthony, Stella Biderman and Hailey Schoelkopf. EleutherAI blog, 18 April 2023, viewed 8 November 2026. https://blog.eleuther.ai/transformer-math/

20. NVIDIA H100 GPU. NVIDIA, viewed 8 November 2026. https://www.nvidia.com/en-us/data-center/h100/

21. NVIDIA A100 Tensor Core GPU datasheet. NVIDIA, June 2021, viewed 8 November 2026. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf

22. Datasheet for NVIDIA Blackwell Architecture. NVIDIA, October 2025, viewed 8 November 2026. https://dam-cdn.nvd.orangelogic.com/AssetLink/y441155802qub41q118b2852i557jem5.pdf

23. Accelerating Inference with Sparsity Using the NVIDIA Ampere Architecture and NVIDIA TensorRT. Jeff Pool, Abhishek Sawarkar and Jay Rodge. NVIDIA Technical Blog, 20 July 2021, viewed 8 November 2026. https://developer.nvidia.com/blog/accelerating-inference-with-sparsity-using-ampere-and-tensorrt/

24. NVIDIA Hopper Architecture In-Depth. NVIDIA Technical Blog, 22 March 2022, viewed 8 November 2026. https://developer.nvidia.com/blog/nvidia-hopper-architecture-in-depth/

25. Performance. Megatron Bridge documentation, 26.08.01 NeMo container. NVIDIA, viewed 8 November 2026. https://docs.nvidia.com/nemo/megatron-bridge/latest/performance-summary.html

26. Performance Archive. Megatron Bridge documentation, 26.04.01 NeMo container section. NVIDIA, viewed 8 November 2026. https://docs.nvidia.com/nemo/megatron-bridge/latest/performance-summary-archive.html

27. MLPerf Training Benchmark. MLCommons, page modified 15 October 2026, viewed 8 November 2026. https://mlcommons.org/benchmarks/training/

28. New MLCommons MLPerf Training v5.0 Benchmark Results Reflect Rapid Growth and Evolution of the Field of AI. MLCommons, 4 June 2025, viewed 8 November 2026. https://mlcommons.org/2025/06/mlperf-training-v5-0-results/

29. Introducing ML Productivity Goodput: a metric to measure AI system efficiency. Vaibhav Singh and Daniel Herrington. Google Cloud Blog, 10 April 2024, viewed 8 November 2026. https://cloud.google.com/blog/products/ai-machine-learning/goodput-metric-as-measure-of-ml-productivity

30. GPU cloud pricing: rent NVIDIA H100, H200, and B200. Lambda, viewed 8 November 2026. https://lambda.com/pricing

31. Amazon EC2 On-Demand price data, US East (N. Virginia), Linux. Amazon Web Services, published 10 November 2026, viewed 12 November 2026. https://b0.p.awsstatic.com/pricing/2.0/meteredUnitMaps/ec2/USD/current/ec2-ondemand-without-sec-sel/US%20East%20%28N.%20Virginia%29/Linux/index.json

32. Specifications for Amazon EC2 accelerated computing instances. Amazon EC2 documentation. Amazon Web Services, viewed 8 November 2026. https://docs.aws.amazon.com/ec2/latest/instancetypes/ac.html

About this report

Published by Rillor on 13 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. From GPU-hour price to cost per training token. 13 November 2026. https://rillor.com/insights/gpu-hour-price-to-cost-per-training-token

Notices

Rillor is not a registered investment adviser or commodity trading advisor and does not provide investment or trading advice. Research, forecasts, data and software described on this site are for research and engineering use. Nothing here is an offer or recommendation to buy or sell any security, commodity interest or digital asset. Past or simulated results do not indicate future results.

NVIDIA, AMD, Ceph, Kubernetes and related product names are trademarks of their owners and are used only to identify products. No affiliation or endorsement is implied.

Work with Rillor

Send the GPU classes, regions and pricing types you need, and how often you need them. Contact us at rillor.com/contact?type=index or contact@rillor.com. More on GPU and neocloud pricing: rillor.com/markets/gpu-pricing