Chair for Decentralized Information Systems and Data Management
TUM School of Computation, Information and Technology
Technical University of Munich

Cloud Information Systems

Exercise 8

9th December 2024

Prof. Dr. Viktor Leis, M.Sc. Till Steinert, M.Sc. Jana Vatter

WILL KNIGHT BUSINESS DEC 3, 2024 1:13 PM

Amazon Is Building a Mega AI Supercomputer With Anthropic

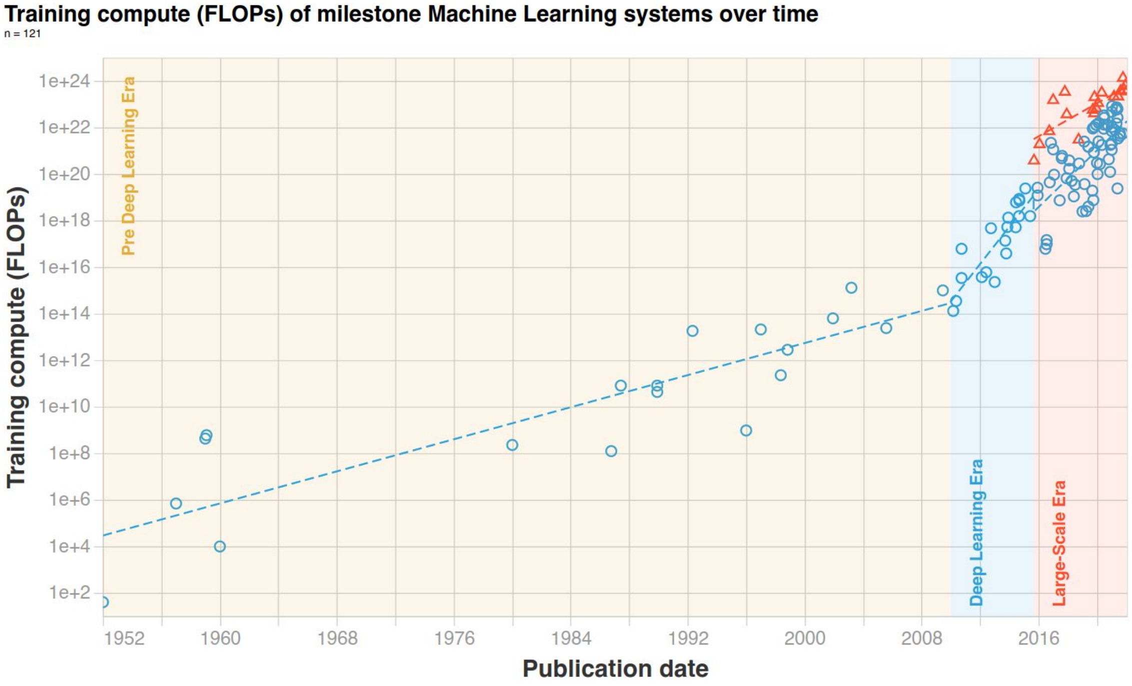
At its Re:Invent conference, Amazon also announced new tools to help customers build generative AI programs, including one that checks whether a chatbot's outputs are accurate or not.

AWS & Anthropic Announcement

1. Recap: Hardware Trends

- **Past:** Moore’s Law
  - Number of Transistor per Chip doubles every ~ two years
  - Incentive to focus R&D on fast improving CPUs
  - Specialized Hardware not economically viable due to high R&D cost and fast improving CPUs
- **Present:** CPU Stagnation
  - Emergence of Workloads (e.g., ML, cryptocurrencies) that profit from specialized Hardware
  - Lower Adoption Threshold due to the Cloud (Renting the Hardware vs developing/buying custom chips)

1. Recap: Training Compute of ML Models



[Interactive Visualization]

1. Recap: Machine Learning on AWS

- **Option 1**: Graphics Processing Unit (GPU)
  - two orders of magnitude more floating point operations per seconds (FLOPS) than CPUs
  - Several GPU-equipped instances: g2, g3, g3s, g4ad, g4dn, g5, g5g, p2, p3, p3dn, p4d, p4de
  - p4de.24xlarge: 8x NVIDIA A100 (≈ 600 16-bit TFLOPs), 640 GB GPU memory
- **Option 2**: Machine Learning Accelerators
  - Purpose built for Machine Learning
  - AWS Trainium trn1.32xlarge: 3 PFLOPS (5x FLOPs of p4de.24xlarge)
  - Only available at a specific Cloud Vendor (you can't purchase them!)

2. Train a large ML model in the Cloud

2. Train a large ML model in the Cloud?

Accelerated Computing

Accelerated computing instances use hardware accelerators, or co-processors, to perform functions, such as floating point number calculations, graphics processing, or data pattern matching, more efficiently than is possible in software running on CPUs.

P4 | P3 | P2 | DL1 | Trn1 | Inf2 | Inf1 | G5 | G5g | G4dn | G4ad | G3 | F1 | VT1

Amazon EC2 P4 instances are the latest generation of GPU-based instances and provide highest performance for machine learning training and high performance computing in the cloud.

Features:

- 3.0 GHz 2nd Generation Intel Xeon Scalable processors (Cascade Lake P-8275CL)
- Up to 8 NVIDIA A100 Tensor Core GPUs
- 400 Gbps instance networking with support for Elastic Fabric Adapter (EFA) and NVIDIA GPUDirect RDMA (remote direct memory access)
- 600 GB/s peer to peer GPU communication with NVIDIA NVSwitch
- Deployed in EC2 UltraClusters consisting of more than 4,000 NVIDIA A100 Tensor Core GPUs, Petabit-scale networking, and scalable low latency storage with Amazon FSx for Lustre

Instance	GPUs	vCPUs	Instance Memory (GiB)	GPU Memory	Network Bandwidth	GPUDirect RDMA	GPU Peer to Peer	Instance Storage (GB)	EBS Bandwidth (Gbps)
p4d.24xlarge	8	96	1152	320 GB HBM2	400 ENA and EFA	Yes	600 GB/s NVSwitch	8 x 1000 NVMe SSD	19
p4de.24xlarge (In preview)	8	96	1152	640 GB HBM2e	400 ENA and EFA	Yes	600 GB/s NVSwitch	8 x 1000 NVMe SSD	19

P4d instances have the following specs:

- 3.0 GHz 2nd Generation Intel Xeon Scalable processors
- Intel AVX, Intel AVX2, Intel AVX-512, and Intel Turbo
- EBS Optimized
- Enhanced Networking†
- Elastic Fabric Adapter (EFA)

EC2 instance types

EC2 instance type	GPU model
P4	NVIDIA A100
P3	NVIDIA Tesla V100
G5g	NVIDIA T4G Tensor Core
G4ad	AMD Radeon Pro V520
…	…

2. Train a large ML model in the Cloud?

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*
Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam
Girish Sastry Amanda Askell Sandhini Agarwal Ariel Herbert-Voss
Gretchen Krueger Tom Henighan Rewon Child Aditya Ramesh
Daniel M. Ziegler Jeffrey Wu Clemens Winter
Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray
Benjamin Chess Jack Clark Christopher Berner
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

Abstract

We demonstrate that scaling up language models greatly improves task-agnostic, few-shot performance, sometimes even becoming competitive with prior state-of-the-art fine-tuning approaches. Specifically, we train GPT-3, an autoregressive language model with 175 billion parameters, 10x more than any previous non-sparse language model, and test its performance in the few-shot setting. For all tasks, GPT-3 is applied without any gradient updates or fine-tuning, with tasks and few-shot demonstrations specified purely via text interaction with the model. GPT-3 achieves strong performance on many NLP datasets, including translation, question-answering, and cloze tasks. We also identify some datasets where GPT-3's few-shot learning still struggles, as well as some datasets where GPT-3 faces methodological issues related to training on large web corpora.

1 Introduction

NLP has shifted from learning task-specific representations and designing task-specific architectures to using task-agnostic pre-training and task-agnostic architectures. This shift has led to substantial progress on many challenging NLP tasks such as reading comprehension, question answering, textual entailment, among others. Even though the architecture and initial representations are now task-agnostic, a final task-specific step remains: fine-tuning on a large dataset of examples to adapt a task agnostic model to perform a desired task.

Recent work [RWC+19] suggested this final step may not be necessary. [RWC+19] demonstrated that a single pretrained language model can be zero-shot transferred to perform standard NLP tasks

*Equal contribution
†Johns Hopkins University, OpenAI

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Model	Total train compute (PF-days)	Total train compute (flops)	Params (M)	Training tokens (billions)	Flops per param per token	Mult for bwd pass	Fwd-pass flops per active param per token	Frac of params active for each token
T5-Small	2.08E+00	1.80E+20	60	1,000	3	3	1	0.5
T5-Base	7.64E+00	6.60E+20	220	1,000	3	3	1	0.5
T5-Large	2.67E+01	2.31E+21	770	1,000	3	3	1	0.5
T5-3B	1.04E+02	9.00E+21	3,000	1,000	3	3	1	0.5
T5-11B	3.82E+02	3.30E+22	11,000	1,000	3	3	1	0.5
BERT-Base	1.89E+00	1.64E+20	109	250	6	3	2	1.0
BERT-Large	6.16E+00	5.33E+20	355	250	6	3	2	1.0
RoBERTa-Base	1.74E+01	1.50E+21	125	2,000	6	3	2	1.0
RoBERTa-Large	4.93E+01	4.26E+21	355	2,000	6	3	2	1.0
GPT-3 Small	2.60E+00	2.25E+20	125	300	6	3	2	1.0
GPT-3 Medium	7.42E+00	6.41E+20	356	300	6	3	2	1.0
GPT-3 Large	1.58E+01	1.37E+21	760	300	6	3	2	1.0
GPT-3 XL	2.75E+01	2.38E+21	1,320	300	6	3	2	1.0
GPT-3 2.7B	5.52E+01	4.77E+21	2,650	300	6	3	2	1.0
GPT-3 6.7B	1.39E+02	1.20E+22	6,660	300	6	3	2	1.0
GPT-3 13B	2.68E+02	2.31E+22	12,850	300	6	3	2	1.0
GPT-3 175B	3.64E+03	3.14E+23	174,600	300	6	3	2	1.0

Table D.1: Starting from the right hand side and moving left, we begin with the number of training tokens that each model was trained with. Next we note that since T5 uses an encoder-decoder model, only half of the parameters are active for each token during a forward or backwards pass. We then note that each token is involved in a single addition and a single multiply for each active parameter in the forward pass (ignoring attention). Then we add a multiplier of 3x to account for the backwards pass (as computing both $\frac{\partial params}{\partial loss}$ and $\frac{\partial acts}{\partial loss}$ use a similar amount of compute as the forwards pass. Combining the previous two numbers, we get the total flops per parameter per token. We multiply this value by the total training tokens and the total parameters to yield the number of total flops used during training. We report both flops and petaflop/s-day (each of which are 8.64e+19 flops).

Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.
[Link to Paper]

2. Train a large ML model in the Cloud?

How many seconds does it take one V100 GPU to train GPT-3?
→ 3.14e+23 FLOPs / 28e+12 FLOPs
≈ 1.12e+10 seconds

How many years are that?
→ 1.12e+10 secs / 3.156e+7 secs ≈ 355 years

How much would the training cost when using a p3.2xlarge instance?
→ 1.12e+10 secs × $0.000274 per sec
= $3,068,800

What do we know?	
GPT-3 175B	3.14e+23 FLOPs in total
NVIDIA Tesla V100	~**28** TFLOPs
Tera	10^12
1 year	31,536,000 seconds
p3.2xlarge (1 GPU)	**$0.9853** per hour reserved 3 years **$0.000274** per second reserved 3 years

2. Train a large ML model in the Cloud?

How many seconds do it take 8 A100 GPUs to train GPT-3?
→ 3.14e+23 FLOPs / ((8 × 78)e+12) FLOPs
≈ 5.03e+8 seconds

How many years are that?
→ 5.03e+8 secs / 3.156e+7 secs ≈ 16 years

How much would the training cost when using a p4d.24xlarge instance?
→ 5.03e+8 secs × $0.003022 per sec
= $1,520,686

What do we know?	
GPT-3 175B	3.14e+23 FLOPs in total
NVIDIA Tesla A100	~**78** TFLOPs
Tera	10^12
1 year	31,536,000 seconds
p4d.24xlarge (8 GPUs)	**$10.8787** per hour reserved 3 years **$0.003022** per second reserved 3 years

2. Train a large ML model in the Cloud?

How many seconds does it take one trn1.32xlarge instance to train GPT-3?
→ 3.14e+23 FLOPs / ((16 × 190)e+12) FLOPs
≈ 1.03e+8 seconds

How many years are that?
→ 1.03e+8 secs / 3.156e+7 secs ≈ 3.26 years

How much would the training cost when using a trn1.32xlarge instance?
→ 1.03e+8 secs × $0.001982 per sec
= $204,146

What do we know?	
GPT-3 175B	3.14e+23 FLOPs in total
AWS Trainium	**190** TFLOPs
Tera	10^12
1 year	31,536,000 seconds
trn1.32xlarge (16 Trainium devices)	**$7.1341** per hour reserved 3 years **$0.001982** per second reserved 3 years

3. Reverse Engineering the EC2 Pricing Model

3. Reverse Engineering the EC2 Pricing Model

- So far we have seen different Pricing Structures:
  - Lambda: Pricing of vCPU/RAM is coupled ($0.0000166667 per-GB-per-sec)
  - Fargate: Separately pay for vCPU-per-hour ($0.04048) and GB-per-hour ($0.004445)
- For EC2 Instances, it is not so clear:
  - c5.large (4GiB, 2 vCPUs) -> c5.xlarge (8GiB, 4 vCPUs) for 2x Price
  - How does this translate to a GB-per-hour / vCPU-per-hour Price ?

3. Reverse Engineering the EC2 Pricing Model

Instances only differ in one Attribute (and Price)

Name	**Memory**	vCPUs	Processor	Storage	Network	**Price**
r7g.16xlarge	**512**	64	AWS Graviton3	EBS only	30 Gbit	**3.4272 $**
m7g.16xlarge	**256**	64	AWS Graviton3	EBS only	30 Gbit	**2.6112 $**
c7g.16xlarge	**128**	64	AWS Graviton3	EBS only	30 Gbit	**2.32 $**

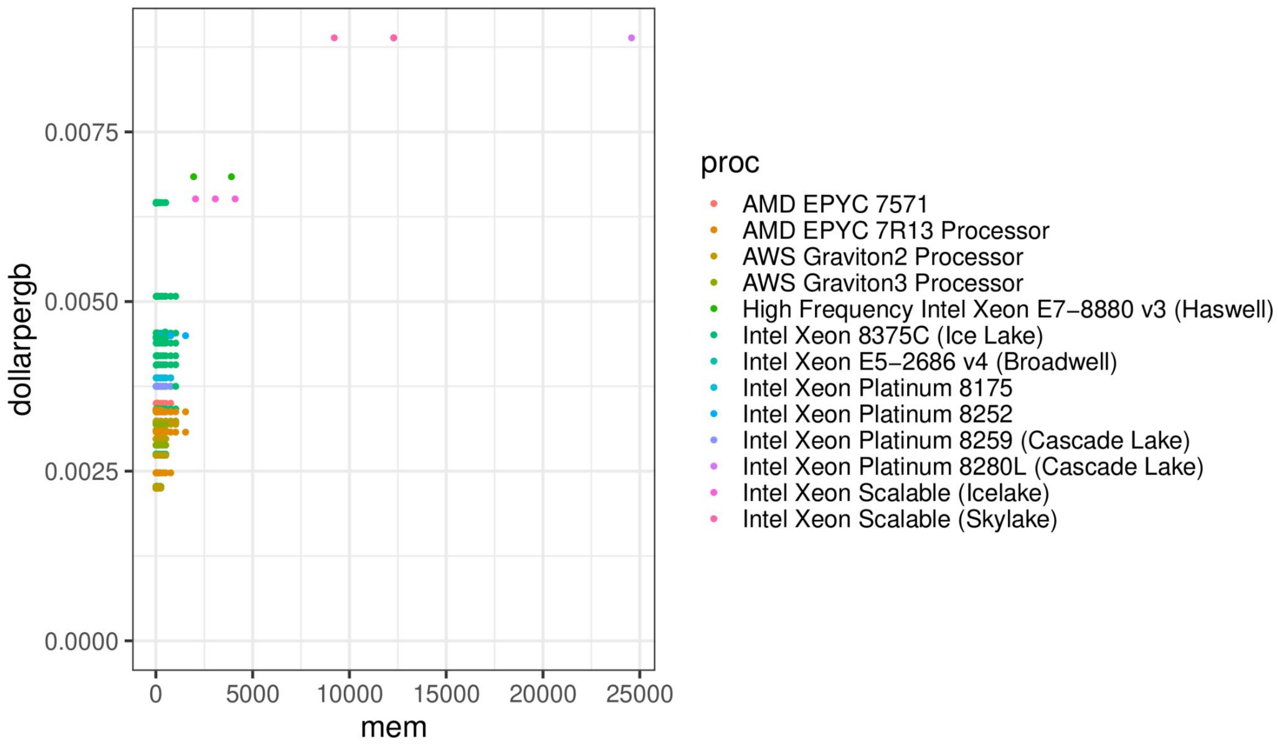
Pair-wise Comparison of Instances

m7g.16xlarge c7g.16xlarge

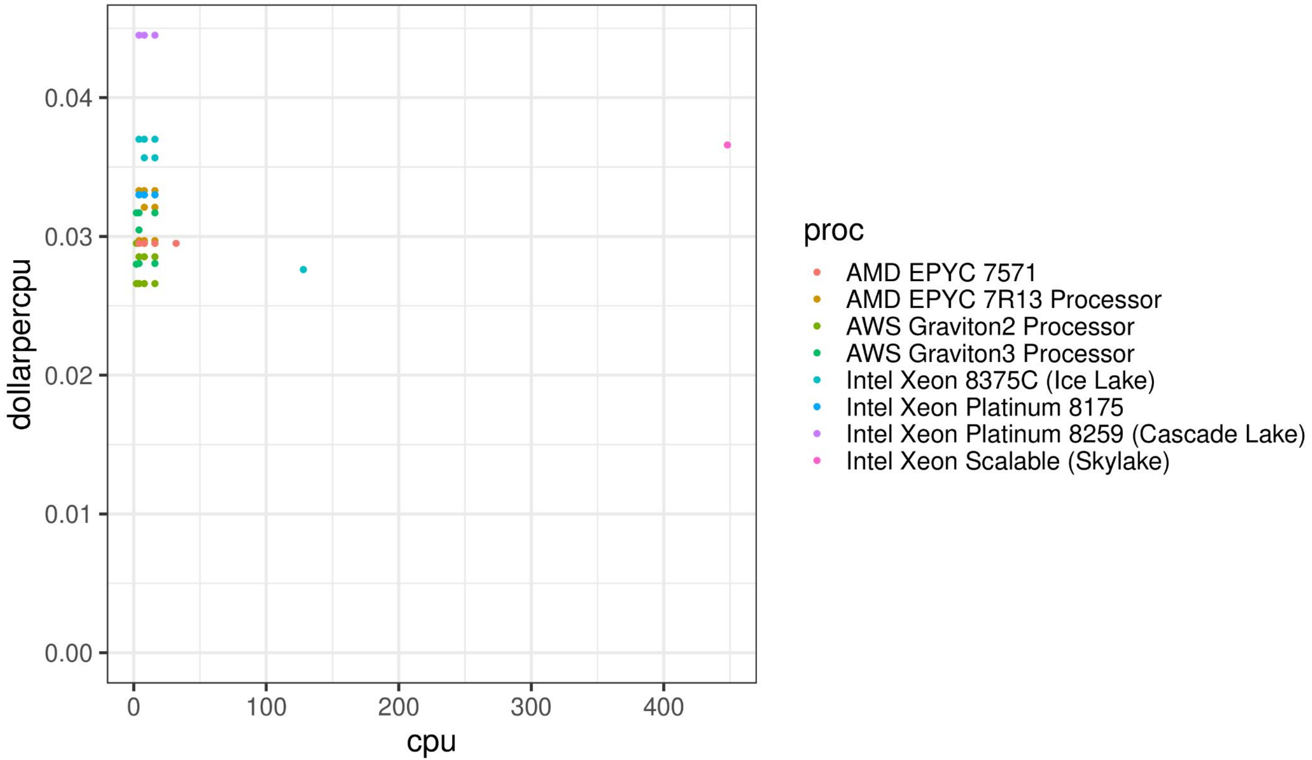
$$price\_per\_gb = \frac{\boxed{2.6112\$} - \boxed{2.32\$}}{\boxed{256GiB} - \boxed{128GiB}} = 0.002275$$

Repeat for other Pairs (m7g.16xlarge, r7g.16xlarge) and (c7g.16xlarge, r7g.16xlarge)

Dollar per GiB of RAM



Dollar per vCPU



How much do you pay for a vCPU ?

Processor	Avg($/vCPU)	Min($/vCPU)	Max($/vCPU)
AWS Graviton2	0.02818182	0.0266	0.0295
AMD EPYC 7571	0.0295	0.0295	0.0295
AWS Graviton3	0.02995238	0.028	0.0317
AMD EPYC 7R13	0.03165	0.0297	0.0333
Intel Xeon Platinum 8175	0.033	0.033	0.033
Intel Xeon 8375C	0.0343272	0.02761146	0.037
Intel Xeon Scalable	0.03658973	0.03658973	0.03658973
Intel Xeon Platinum 8259	0.0445	0.0445	0.0445

4. Questions

References

[1] https://www.reddit.com/r/GPT3/comments/p1xf10/how_many_days_did_it_take_to_train_gpt3_is/

[2] https://lambdalabs.com/blog/demystifying-gpt-3