COMPARING DATA PRECISION ON LOW-RANK ADAPTATION FOR FINE-TUNING LARGE LANGUAGE MODELS

Bagus Hanindhito[1], Bhavesh Patel[2], and Lizy K. John[3]

[1] **hanindhito@bagus.my.id**, The University of Texas at Austin, Austin, TX, USA
[2] **bhavesh_a_patel@dell.com**, Dell Technologies, Round Rock, TX, USA
[3] **ljohn@ece.utexas.edu**, The University of Texas at Austin, Austin, TX, USA

ARC-LG Workshop on Large Language Models and Graph Neural Networks @ ISCA 2024
June 30, 2024, Buenos Aires, Argentina.

Agenda

Agenda

Introduction

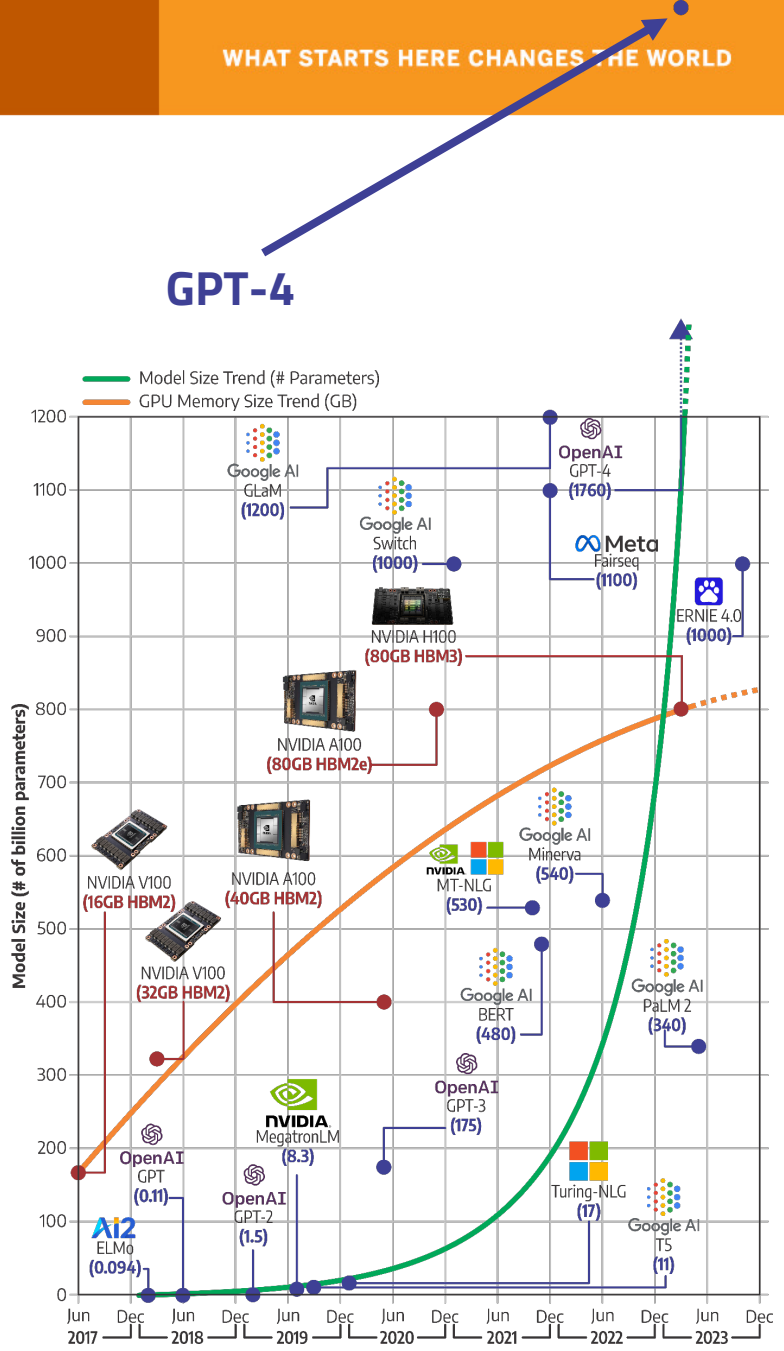
The size of **Large Language Models** grows exponentially.
- Factor of **1000×** between 2018 (Elmo) to 2020 (GPT-3).
- The **generative AI** era demands even larger models.

Graphics Processing Units are popular **accelerators** for **AI/ML.**
- Only see **5× increase** in **memory capacity** between 2018 (V100) to 2020 (A100).

State-of-the-art models can no longer fit inside single GPU.
Hundreds/thousands GPUs are required to train them.
- **Expensive** infrastructure, out-of-reach for most researchers.
- Significant **environmental impact**.

Fine-tuning has become famous for efficiently adapting pre-trained models for specific tasks, compared to training from scratch.

As the model size increases, fine-tuning demand even more resources.



Agenda

Research Objectives

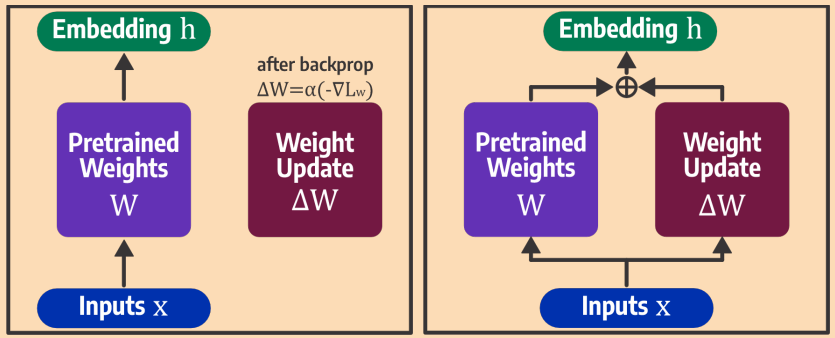
obj 1	**Summarize and compare** standard, LoRA, and QLoRA fine-tuning methods.
obj 2	**Estimate** the **GPU memory requirement** for each **fine-tuning method** using different **data precision**.
obj 3	**Measure fine-tuning performance** on three **GPUs**: NVIDIA A100, H100, and L40.
obj 4	**Measure resource utilization** (CPU memory, GPU memory) for each fine-tuning method.
obj 5	**Evaluate the model quality** from each **fine-tuning method** with different **data precision**.

Agenda

Fine-tuning Large Language Models

Standard Fine-tuning

- **Adapting pre-trained models** for specific tasks by exposing them to more specific datasets.
- Save time, resources, and energy compared to training from scratch.

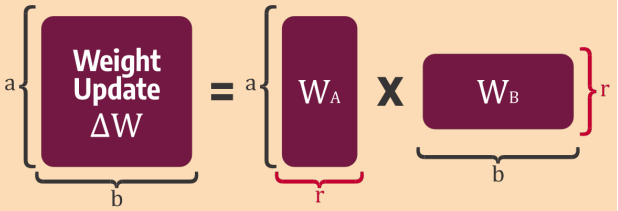


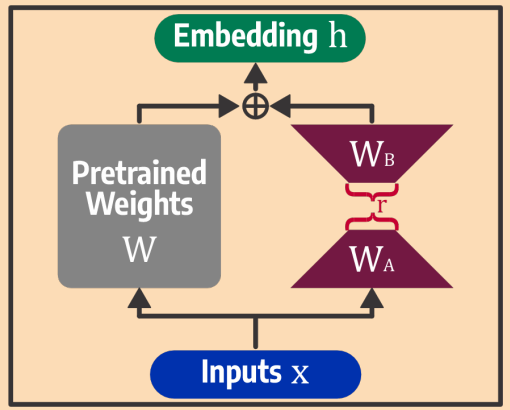
Fine-tuning flow Alternative flow

- **Fine-tuning** is **more demanding** as the size of **models grows**.
- Needs to find more efficient way.

Low-Rank Adaptation (LoRA)

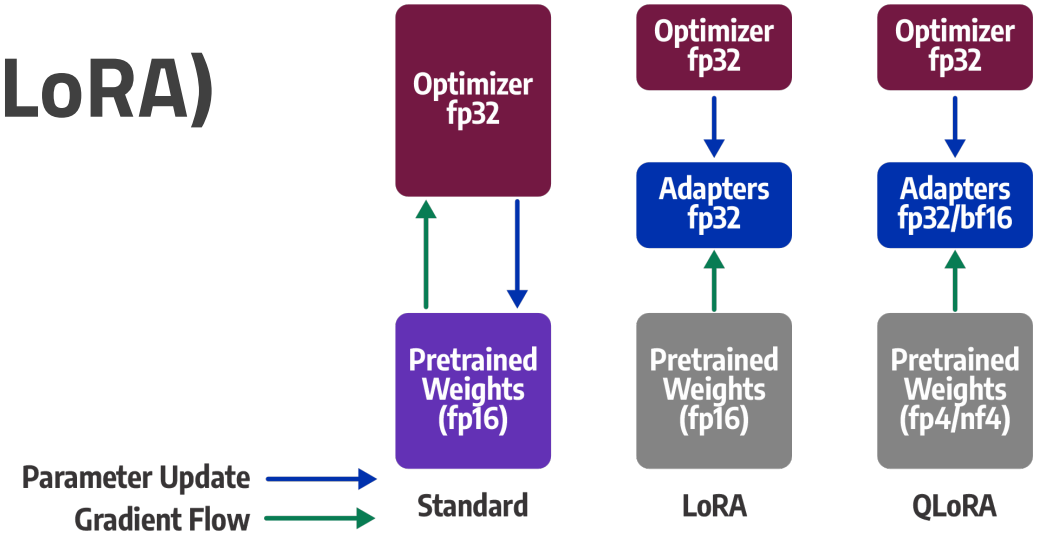
- Aghajanyan et. al. showed that **pre-trained models** have significantly **lower intrinsic dimensions** during fine-tuning.
  - There exists a lower dimension representation of the model.
- Microsoft proposed efficient fine-tuning method called **Low-Rank Adaptation (LoRA)**.
- Replaces Weight Update matrix with two smaller matrices (**LoRA Adapter**).
  - **Rank r** is **hyperparameter** that controls how large **LoRA matrices**.
  - Provide **trade-off** between model **complexity**, **adaptation** ability, fine-tuning **costs**.
- Pre-trained **weights** are **frozen** (not updated) during fine-tuning.
  - Only **adapters** are **updated**.





Quantized Low-Rank Adaptation (QLoRA)

- Even with LoRA, the **frozen weight matrix** can still be **very large** and beyond the capacity of single GPU.
- QLoRA was introduced by Dettmers et. Al. in 2023 to **significantly reduce memory requirements of LoRA**.
- Improved LoRA with **three major innovations:**



4-Bit Weight Quantization

- Introduced two **4-bit floating-point data types** for **storing** pre-trained model's **weight in memory**:
  - 4-bit floating-point format (**FP4**)
  - 4-bit **normalized** floating-point format (**NF4**)
    - **Information-theoretically optimal data type**.
    - Each quantization bin has equal values based on empirical cumulative distribution.
- **Computation** is still done in **FP32/BF16** ("**dequantize**")

Double Quantization

- **Quantize the quantization constant.**
- **Two levels of quantization**:
  - Level 1: One FP32 constant per 64 values.
  - Level 2: One FP8 constant per 256 constant.
- Quantization **overhead**: 0.127 bits/parameter

Paged Optimizer

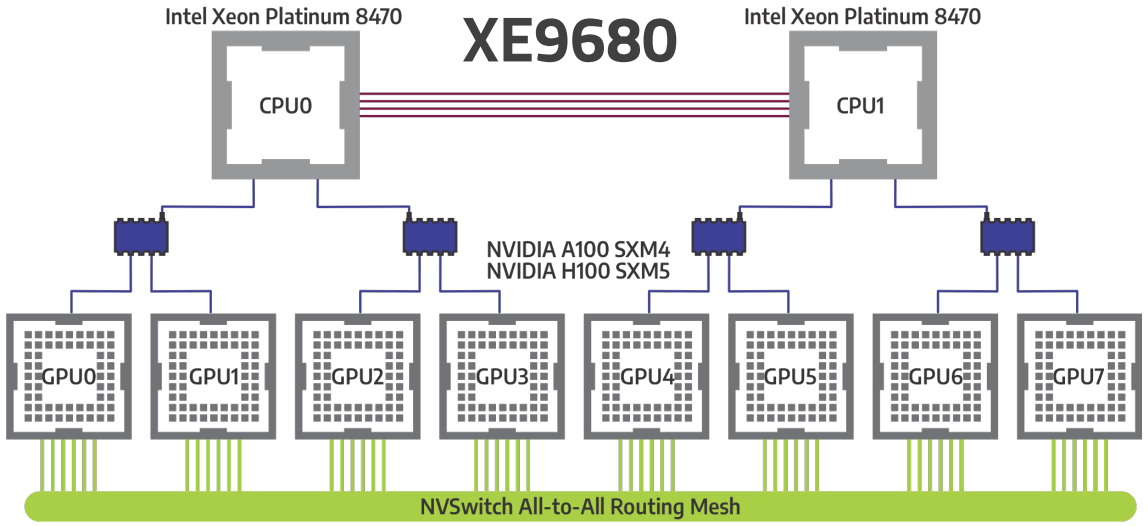
- Use NVIDIA Unified Virtual Memory to use both CPU and GPU memory to store optimizer states.
- Degrade performance due to data movement.

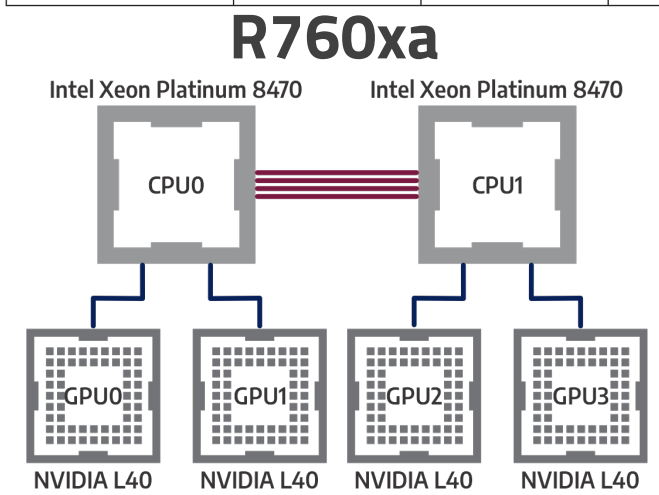
Agenda

Hardware Setup

- Three state-of-the-art NVIDIA GPUs:
  - NVIDIA **A100 (Ampere)** 80GB SXM4 with NVLink
    - Housed inside Dell XE9680 8-way GPU chassis
  - NVIDIA **H100 (Hopper)** 80GB with NVLink
    - Housed inside Dell XE9680 8-way GPU chassis
  - NVIDIA **L40 (Ada Lovelace)** 48GB PCIe.
    - Housed inside Dell R760xa chassis

Platform	XE9680	XE9680	R760xa
GPU			
Model	NVIDIA A100	NVIDIA H100	NVIDIA L40
# GPUs	8	8	4
Form Factor	SXM4	SXM5	PCIe
# CUDA Cores	6912	16896	18176
# Tensor Cores	432	528	568
Memory Size	80 GB	80 GB	48 GB
Memory Type	HBM2E	HBM3	GDDR6
Bandwidth	2,039 GBps	3,350 GBps	864 GBps
Typical Power	500 W	700 W	300 W
CPU			
Model (#Sockets)	Xeon 8470 (2)	Xeon 8470 (2)	Xeon 6430 (2)
Base/Turbo Clock	2.00/3.80 GHz	2.00/3.80 GHz	2.10/3.40 GHz
#Cores/#Threads	104/208	104/208	64/128
Memory Size	2048 GB	2048 GB	512 GB
Memory Type	DDR5-4400	DDR5-4400	DDR5-4400
Bandwidth	281 GBps	281 GBps	281 GBps
Typical Power	350 W	350 W	270 W
Interfaces			
CPU-to-CPU	UPI 16 GT/s	UPI 16 GT/s	UPI 16 GT/s
CPU-to-GPU	PCIe 4.0 x16	PCIe 5.0 x16	PCIe 4.0 x16
GPU-to-GPU	NVLink 3.0	NVLink 4.0	None





Software Setup

Large Language Model

- Four different models, each with two sizes:
  - **small** (7 billion parameters)
  - **large** (> 40 billion parameters)

Name	Developer	# Parameters	HuggingFace Hub Link
Llama	Meta AI	7 billion	huggyllama/llama-7b
Llama	Meta AI	65 billion	huggyllama/llama-65b
Llama2	Meta AI	7 billion	meta-llama/Llama-2-7b-hf
Llama2	Meta AI	70 billion	meta-llama/Llama-2-70b-hf
Falcon	TII UAE	7 billion	tiiuae/falcon-7b
Falcon	TII UAE	40 billion	tiiuae/falcon-40b
WizardLM	Microsoft	7 billion	WizardLM/WizardLM-7B-V1.0
WizardLM	Microsoft	70 billion	WizardLM/WizardLM-70B-V1.0

Dataset

OpenAssistant Conversation Dataset (OASST1)

Model Quality Evaluation

Massive Multitask Language Understanding (MMLU)

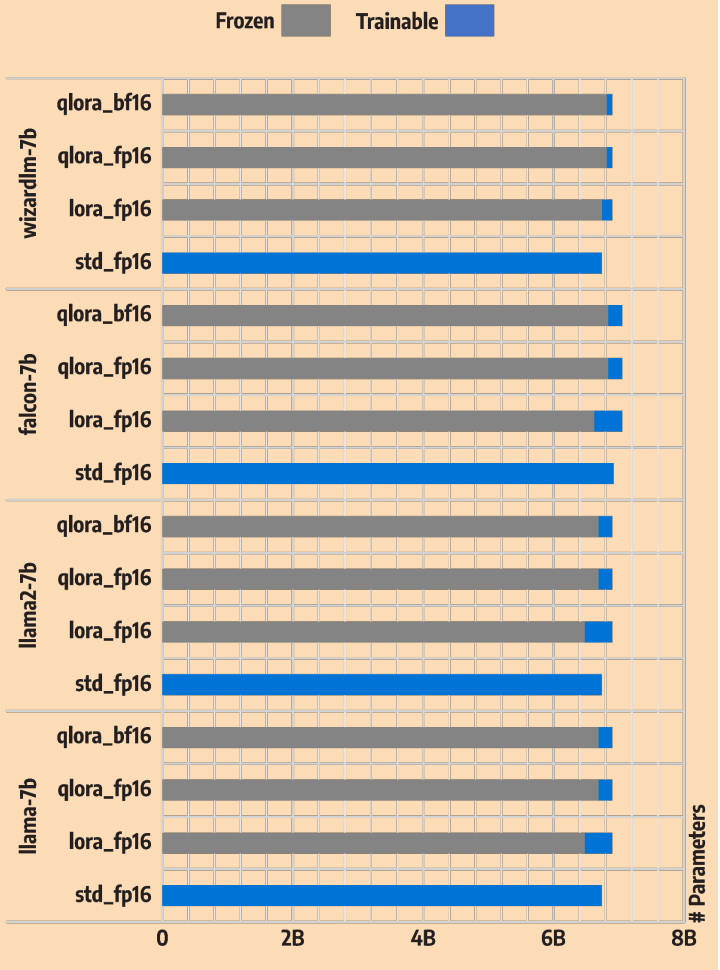
Software Stack

- NVIDIA Driver 535.86 with CUDA 12.2
- PyTorch 2.2.0 build from source.
- HuggingFace **Transformers**
  - Provides APIs for quickly **interacting** with **pre-trained models**.
- HuggingFace **Evaluate**
  - Provides tools for **evaluating** and comparing model's performance.
- HuggingFace **Accelerate**
  - Provides abstraction to run PyTorch in any device, including **Multi-GPU**.
- **DeepSpeed ZeRO**
  - Alternative to HuggingFace Accelerate, providing **multi-GPU with data-parallelism.**
- **BitsandBytes**
  - **4-bit** and **8-bit quantization** for QLoRA.

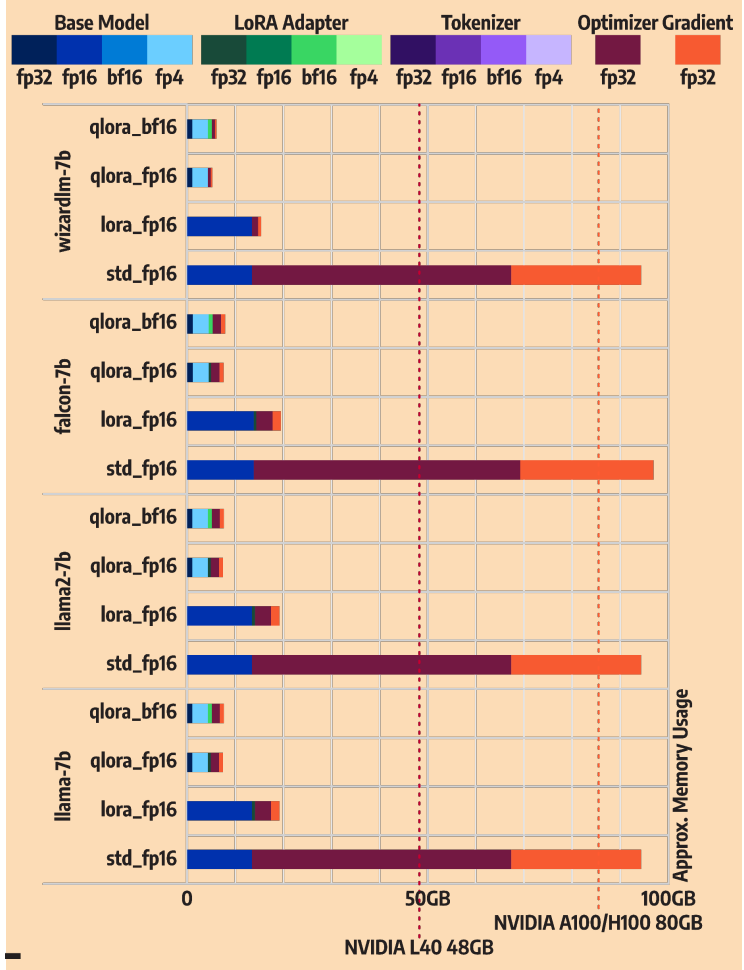
Agenda

Estimating Memory Usage (~7 Billion Parameter Models)

Parameters (Trainable / Frozen)



Approximate Memory Usage

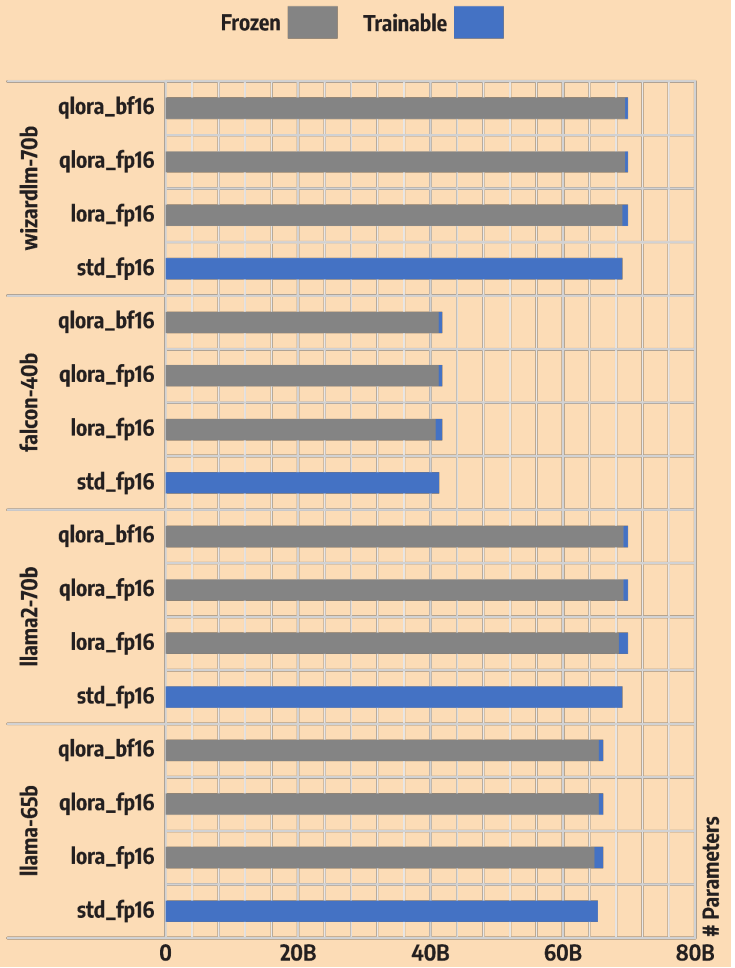


Remarks

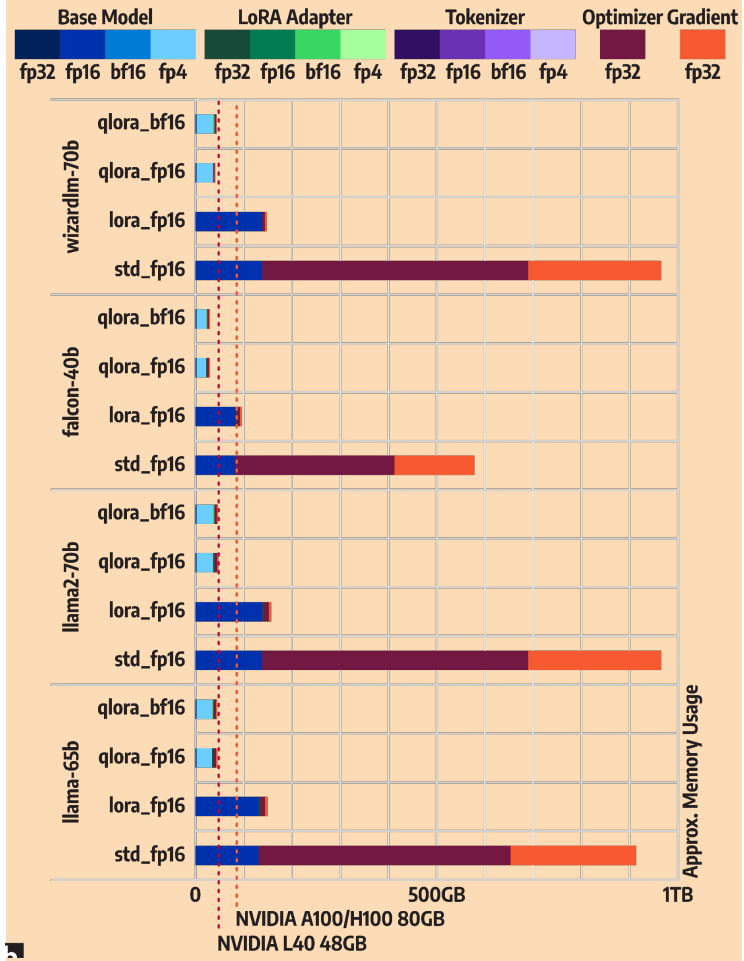
- For **LoRA** and **QLoRA**, **only the adapters are trainable**; the pre-trained weight is frozen.
  - **Frozen parameters do not need optimizers & gradients**.
- For QLoRA, **two 4-bit parameters are packed into 1 byte**.
  - PyTorch counts two quantized parameters as one.
- **Standard fine-tuning** requires memory larger than single A100/H100 80GB.
  - **Needs multiple GPUs**.
- LoRA allows single L40 48GB GPU to fine-tune these models.
- QLoRA further reduce memory requirements by 75%.

Estimating Memory Usage (>40 Billion Parameter Models)

Parameters (Trainable / Frozen)



Approximate Memory Usage



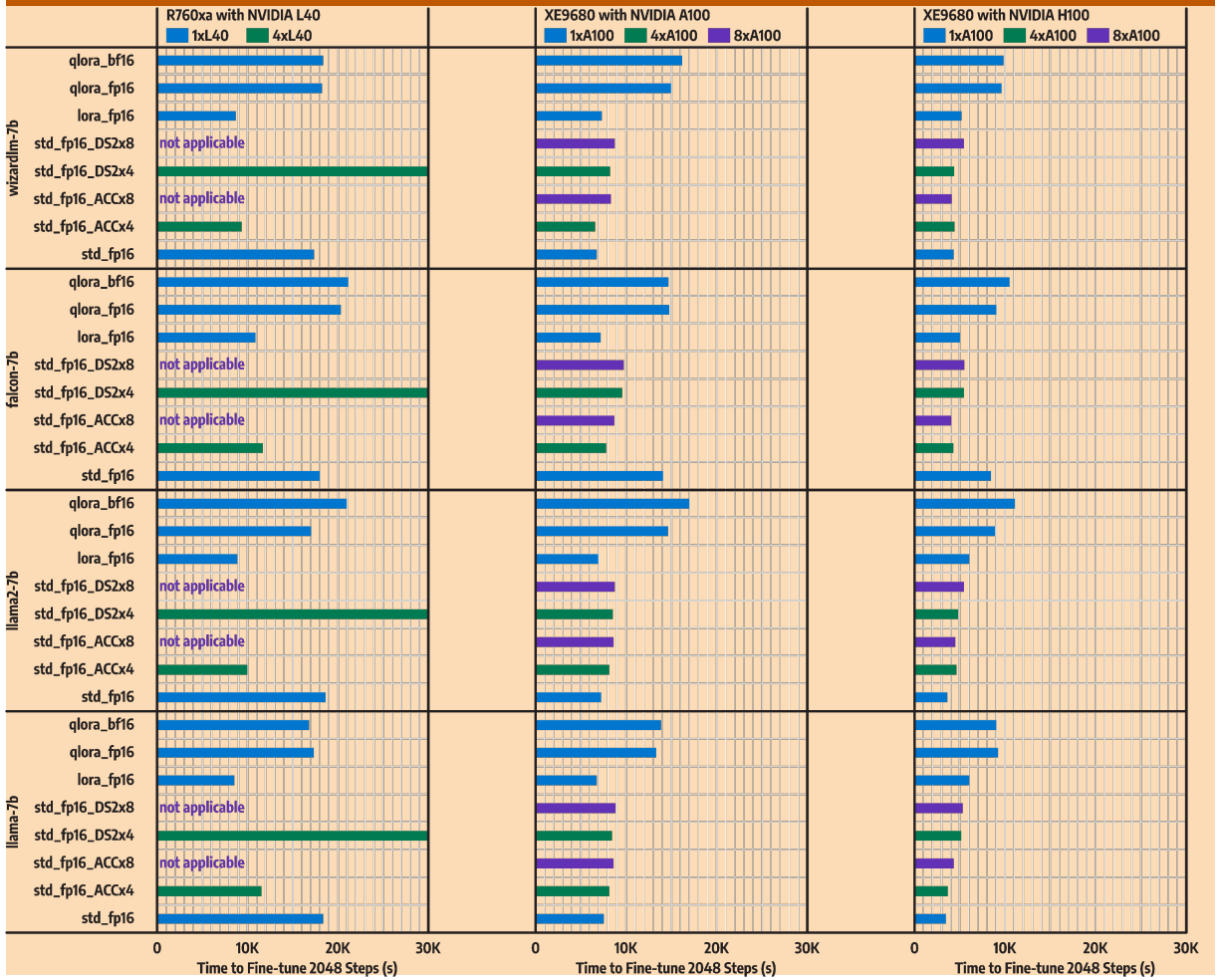
Remarks

- **Standard Fine-tuning** will require **~16 NVIDIA A100/H100 80GB** GPUs.
- **LoRA could no longer fit** the model into single A100/H100 80GB GPU for fine-tuning.
- **QLoRA is able to fit** the model into single A100/H100 80GB GPU.
  - It even gives a chance to use single L40 48GB GPU for fine-tuning.

Agenda

Fine-tuning Performance (~7 Billion Parameter Models)

Time to Fine-tune 2048 Steps (s)

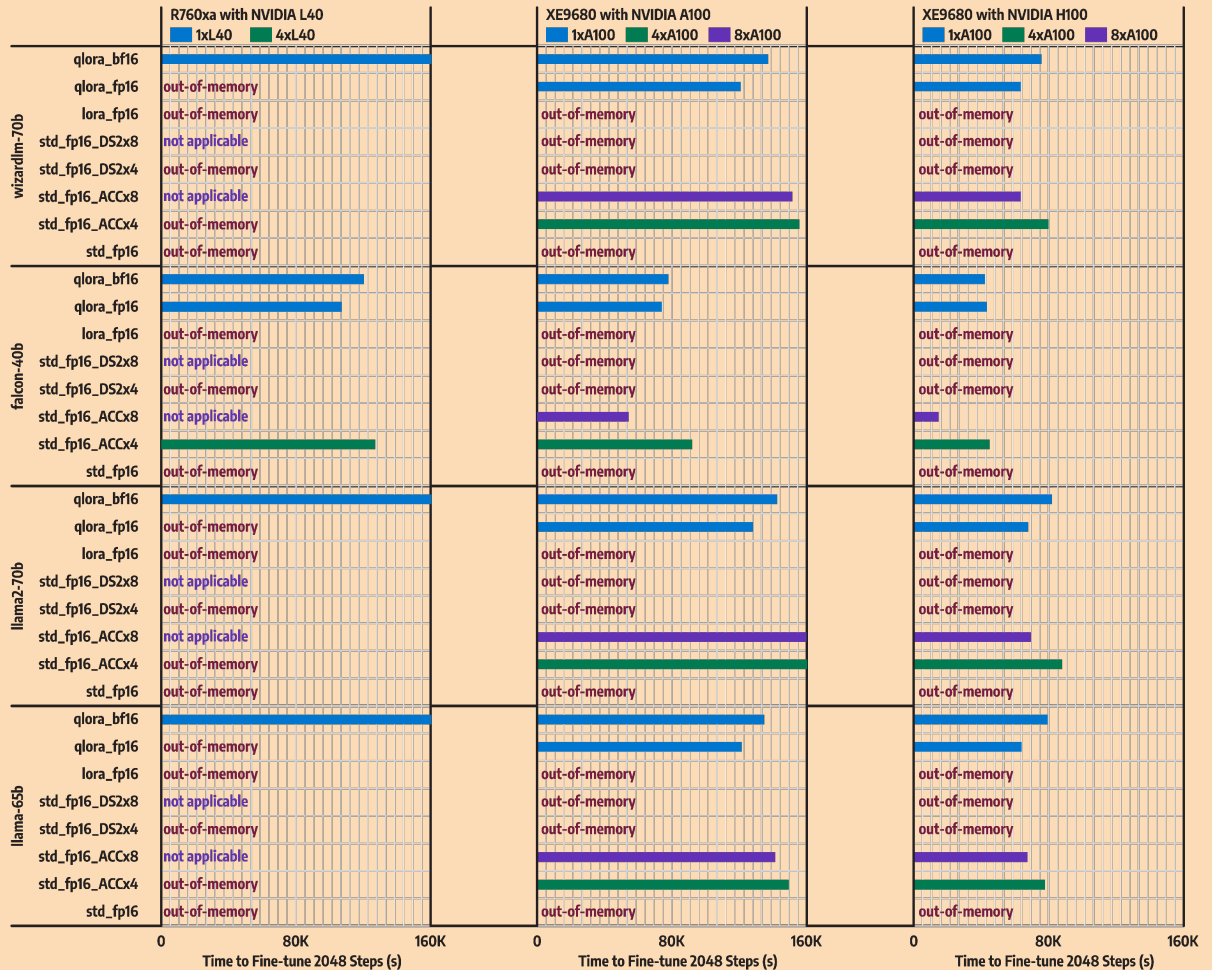


Remarks

- Standard Fine-tuning performance:
  - Single GPU is not sufficient, optimizer states must **spill over to CPU memory**.
  - **L40 48GB** got the worst impact of excessive CPU-GPU data movement due to small memory.
  - Multi-GPU fine-tuning with **HF Accelerate**:
    - Four L40 48GB GPUs get the highest benefit.
    - Using four and eight A100/H100 80GB GPUs do not give improvement due to inter-GPU communication overhead.
  - Multi-GPU fine-tuning with **DeepSpeed ZeRO**:
    - Worse performance than HF Accelerate.
- **LoRA gives better performance** than standard fine-tuning, especially for L40 48GB.
- **QLoRA gives slightly worse performance than LoRA** due to the overhead of quantization.
- **QLoRA** with **BF16** gives a slightly **lower performance** than **QLoRA FP16**.

Fine-tuning Performance (>40 Billion Parameter Models)

Time to Fine-tune 2048 Steps (s)



Remarks

- Standard Fine-tuning performance:
  - Single GPU no longer sufficient, even with paged optimizer.
  - Multi-GPU fine-tuning with **HF Accelerate**:
    - Allows standard fine-tuning on four and eight A100/H100 80GB GPUs.
  - Multi-GPU fine-tuning with **DeepSpeed ZeRO**:
    - ZeRO-2 is not sufficient; need to use ZeRO-3 with offload.
- **LoRA** with FP16 is still too big for single GPU, even for A100/H100 80GB.
- **QLoRA** allows fine-tuning in FP16/BF16 with single A100/H100 80GB GPU.
  - Only QLoRA with BF16 can fit the model on L40 48GB.

Agenda

Model Quality Evaluation using MMLU Benchmark

Average Accuracy (standard vs. LoRA vs. QLoRA)

- **For Llama2 7B**:
  - Standard gives average accuracy **0.24.**
  - LoRA & QLoRA gives average **0.42** and **0.49.**
- For WizardLM 7B:
  - LoRA and QLoRA gives **on-par accuracy** with standard fine-tuning.
- Other models **show the same trends**.

QLoRA FP4 vs. NF4 Quantization

- **NF4 achieves slightly better accuracy** score than FP4: up-to 4% higher average score, depending on the model.
- **NF4 introduces more computation overhead** due to empirical distribution quantization, resulting in 4%-5% longer fine-tuning time.

QLoRA Mixed Precision FP16 vs. BF16

- Note that QLoRA with FP16 is actually run on FP32 precision.
  - Underlying CUDA library uses TF32 instead of FP32 to take advantages of Tensor Cores.
- **QLoRA FP16** is slightly **faster** than BF16.
- **QLoRA BF16** has slightly **lower memory** than FP16.

QLoRA Single vs. Double Quantization

- Single and Double quantization achieves roughly **the same accuracy score.**
- **Double quantization saves memory usage** by up-to 5%.

Conclusion

- **Model Fine-tuning:**
  - Standard fine-tuning is no longer viable for handling state of the art models; finding efficient fine-tuning method is required.
- **Low-Rank Adaptation (LoRA) and Quantized LoRA (QLoRA):**
  - Since models have low intrinsic dimensions during fine-tuning, one can represent the model with smaller matrices ("LoRA Adapter") without losing too much information.
  - Even with LoRA, large models may still need huge memory requirements exceeding single GPU memory capacity.
  - QLoRA improves LoRA with three innovations: Four-bit quantization, Double Quantization, and Paged Optimizers.
- **Fine-tuning Performance**
  - QLoRA allows fine-tuning models with >40 billion parameters using single GPU at the expense of more computation due to quantization compared to LoRA.
- **Model Quality**
  - QLoRA delivers model quality that is on par or exceed the standard fine-tuning.

☺ Thank You ☺

Laboratory for Computer Architecture

The Laboratory for Computer Architecture (LCA) is a research group within the Department of Electrical and Computer Engineering at The University of Texas at Austin. The lab is directed by Dr. Lizy Kurian John and is part of the Computer Engineering Research Center (CERC).

The members of the Laboratory for Computer Architecture are investigating several avenues in computer architecture. Some of our current research interests include:

- Cloud and Big Data Architecture
- Memory Systems for Multicore and Many- core Architectures
- Workload Characterization
- Proxies for Computer Performance/ Power Evaluation
- Low Power Architectures
- Development of Energy-efficient, High-Performance Codes
- Compiler Support for Innovative Micro-architectures

UT ECE