Course Review

COMP6211J

Binhang Yuan

The Path Towards a Foundation Model

RELAXED SYSTEM LAB

Preparing a massive corpus for pretraining

A multi-stage pipeline:
- Acquisition (e.g. OCR)
- Cleaning;
- Filtering;
- De-duplication;
- Sampling;
- Shuffling.

Training a foundation model by fitting the large corpus by auto-regression

Core infra:
a large-scale parallel training system running over thousands of GPUs

Tune the model so it can follow user instructions effectively and safely.

Various methods:

- SFT
- RLHF
- RLAIF

Evaluate the model from different aspects:

- Comprehension
- Instruction following
- Expression
- Reasoning
- Coding
- Interaction
- Safety

Deploy the model to serve users under different settings:
- Cloud
- Local devices

Require system efforts to optimize the performance, cost, and latency.

What We Have Explored.

RELAXED SYSTEM LAB

Date	Topic
W1 - 09/03, 09/05	• Introduction and Logistics • Stochastic Gradient Descent
W2 - 09/10, 09/12	• Auto-Differentiation • Nvidia GPU Computation and Communication
W3 – 09/17, 09/19	• LLM Pretraining • Data Parallelism, Pipeline Parallelism
W4 - 09/24, 09/26	• Tensor Model Parallelism, Optimizer Parallelism • LLM Tuning and Utilization
W5 - 10/03	• Generative Inference Overview
W6 - 10/08, 10/10	• Alogirhtm Optimizations for Generative Inference • System Optimizations for Generative Inference
W7 - 10/15, 10/17	• RAG and Domain-Specific LLM Agent • Review

ML System Basics

ML System Basics

RELAXED
SYSTEM LAB

- Stochastic gradient descent:
  - Define the empirical risk;
  - Optimizing the empirical loss.
- Automatic differentiation:
  - Forward mode AD;
  - Reverse Mode AD;
  - Compute the gradient of a Linear Layer.

Summary of a Linear Layer Computation

RELAXED SYSTEM LAB

- Forward computation of a linear layer: $\boldsymbol{Y} = \boldsymbol{XW}$
  - Given input: $\boldsymbol{X} \in \mathbb{R}^{B \times D_1}$
  - Given weight matrix: $\boldsymbol{W} \in \mathbb{R}^{D_1 \times D_2}$
  - Compute output: $\boldsymbol{Y} \in \mathbb{R}^{B \times D_2}$
- Backward computation of a linear layer:
  - Given gradients w.r.t output: $\frac{\partial L}{\partial \boldsymbol{Y}} \in \mathbb{R}^{B \times H_2}$
  - Compute gradients w.r.t weight matrix: $\frac{\partial L}{\partial \boldsymbol{W}} = \boldsymbol{X^T} \frac{\partial L}{\partial \boldsymbol{Y}} \in \mathbb{R}^{B \times H_2}$
  - Compute gradients w.r.t input: $\frac{\partial L}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}} \boldsymbol{W}^T \in \mathbb{R}^{B \times H_2}$

Relevant Presentation Topics

RELAXED
SYSTEM LAB

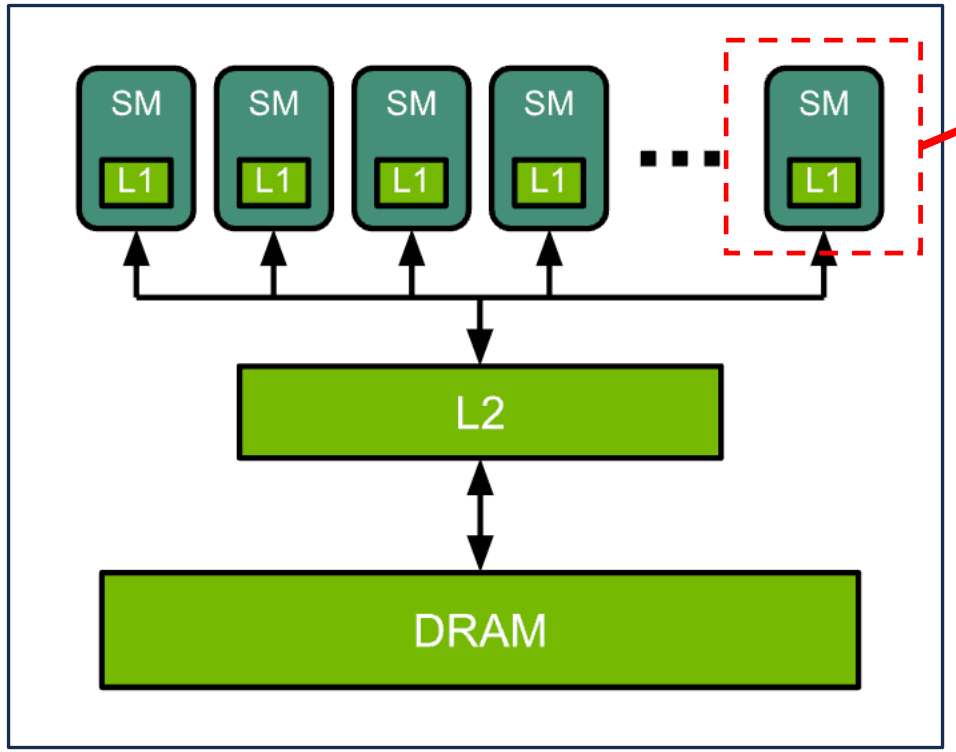
- 1 - Tensor Program Optimization.

RELAXED
SYSTEM LAB

GPU Computation and Communication

Ampere GPU Architecture

RELAXED SYSTEM LAB

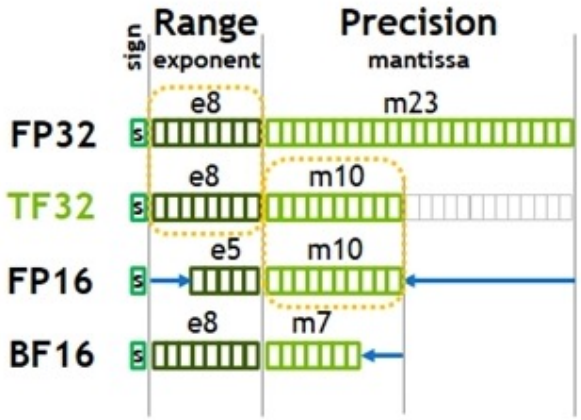


108 SM in a A100 GPU

A100 GPU Tensor Core Computation

RELAXED SYSTEM LAB

- Multiply-add is the most frequent operation in modern neural networks. This is known as the fused multiply-add (FMA) operation.
- Includes one multiply operation and one add operation, counted as two float operations.
- A100 GPU has 1.41 GHz clock rate.
- The Ampere A100 GPU Tensor Cores multiply-add operations per clock:



Ampere A100 GPU FMA per clock on a SM					
FP64	TF32	FP16	INT8	INT4	INT1
64	512	1024	2048	4096	16384

Arithmetic Intensity

RELAXED
SYSTEM LAB

- Thus, on a given processor a given algorithm is math limited if:
  - $T_{math} > T_{mem}$
  - $\frac{\#op}{BW_{math}} > \frac{\#bytes}{BW_{mem}}$

- By simple algebra, the inequality can be rearranged to:
  - $\frac{\#op}{\#bytes} > \frac{BW_{math}}{BW_{mem}}$

- The left-hand side: the algorithm's *arithmetic intensity*.
- The right-hand side: *ops:byte ratio*.

Relevant Presentation Topics

RELAXED
SYSTEM LAB

- 3 - NPU Architecture.
- 4 - TPU Architecture.

NCCL Operators

RELAXED SYSTEM LAB

- Optimized collective communication library from Nvidia to enable high-performance communication between CUDA devices.
- NCCL Implements:
  - **AllReduce**;
  - **Broadcast;**
  - **Reduce;**
  - **AllGather;**
  - **Scatter;**
  - **Gather;**
  - **ReduceScatter;**
  - **AlltoAll.**
- Easy to integrate into DL framework (e.g., PyTorch).

Relevant Presentation Topics

RELAXED
SYSTEM LAB

- 5 - Collective Communication Optimization.
- 6 - In Network Aggregation.

RELAXED
SYSTEM LAB

Language Model

Autoregressive Language Models

RELAXED SYSTEM LAB

- A common way to write the joint distribution $p(x_{1:L})$ of a sequence to $x_{1:L}$ is using the *chain rule of probablity*:

$$p(x_{1:L}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_L|x_{1:L-1}) = \prod_{i=1}^{L} p(x_i|x_{1:i-1})$$

- In particular, $p(x_i|x_{1:i-1})$ is a conditional probability distribution of the next token x_i given the previous tokens $x_{1:i-1}$.
- An autoregressive language model is one where each conditional distribution $p(x_i|x_{1:i-1})$ can be computed efficiently (e.g., using a feedforward neural network).

TransformerBlocks$(x_{1:L} \in R^{L\times D}) \rightarrow \mathbb{R}^{L\times D}$

RELAXED SYSTEM LAB

- B is the batch size;
- L is the sequence length;
- D is the model dimension;
- Multi-head attention: $D = n_H \times H$
- H is the head dimension;
- n_h is the number of heads.

Computation	Input	Output
$Q = xW^Q$	$x \in \mathbb{R}^{L\times D}, W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$
$K = xW^K$	$x \in \mathbb{R}^{L\times D}, W^K \in \mathbb{R}^{D\times D}$	$K \in \mathbb{R}^{L\times D}$
$V = xW^V$	$x \in \mathbb{R}^{L\times D}, W^V \in \mathbb{R}^{D\times D}$	$V \in \mathbb{R}^{L\times D}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[K_1, K_2 \dots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{L\times D}$	$K_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[V_1, V_2 \dots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{L\times D}$	$V_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$\text{Score}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i, K_i \in \mathbb{R}^{L\times H}$	$\text{score}_i \in \mathbb{R}^{L\times L}$
$Z_i = \text{score}_i\ V_i, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{L\times L}, V_i \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$
$Z = \text{Merge}_{-1}([Z_1, Z_2 \dots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$	$Z \in \mathbb{R}^{L\times D}$
$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{L\times D}, W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{L\times D}$
$A = \text{Out}\ W^1$	$\text{Out} \in \mathbb{R}^{L\times D}, W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{L\times 4D}$
$A' = \text{relu}(A)$	$A \in \mathbb{R}^{L\times 4D}$	$A' \in \mathbb{R}^{L\times 4D}$
$x' = A'W^2$	$A' \in \mathbb{R}^{L\times 4D}, W^2 \in \mathbb{R}^{4D\times D}$	$x' \in \mathbb{R}^{L\times D}$

Scaling Law

RELAXED SYSTEM LAB

- Intuitively:
  - Increase parameter # N → better performance
  - Increase dataset scale D → better performance
- But we have a fixed computational budget on $C \approx 6ND$
- ***To maximize model performance, how should we allocate C to N and D?***

DeepMind

Training Compute-Optimal Large Language Models

Jordan Hoffmann*, Sebastian Borgeaud*, Arthur Mensch*, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals and Laurent Sifre*
*Equal contributions

We investigate the optimal model size and number of tokens for training a transformer language model under a given compute budget. We find that current large language models are significantly under-trained, a consequence of the recent focus on scaling language models whilst keeping the amount of training data constant. By training over 400 language models ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens, we find that for compute-optimal training, the model size and the number of training tokens should be scaled equally: for every doubling of model size the number of training tokens should also be doubled. We test this hypothesis by training a predicted compute-optimal model, *Chinchilla*, that uses the same compute budget as *Gopher* but with 70B parameters and 4× more more data. *Chinchilla* uniformly and significantly outperforms *Gopher* (280B), GPT-3 (175B), Jurassic-1 (178B), and Megatron-Turing NLG (530B) on a large range of downstream evaluation tasks. This also means that *Chinchilla* uses substantially less compute for fine-tuning and inference, greatly facilitating downstream usage. As a highlight, *Chinchilla* reaches a state-of-the-art average accuracy of 67.5% on the MMLU benchmark, greater than a 7% improvement over *Gopher*.

Evaluating Distributed Computation

RELAXED SYSTEM LAB

- Scaling law tells us given a fixed computation budget, how should we decide the model scale and data corpus.
- The computation budget is formulated by the total FLOPs demanded during the computation.
- But the GPU cannot usually work at its peak FLOPs.
- *How can we evaluate the performance of a distributed training workflow?*
  - Training throughput (token per second);
  - Scalability;
  - Model FLOPs Utilization.

Relevant Presentation Topics

RELAXED
SYSTEM LAB

- 2 - Beyond Transformer Architecture.
- 35 - Text Image Generation.
- 36 - Text Video Generation.
- 37 - Text-to-3D Asset Generation.

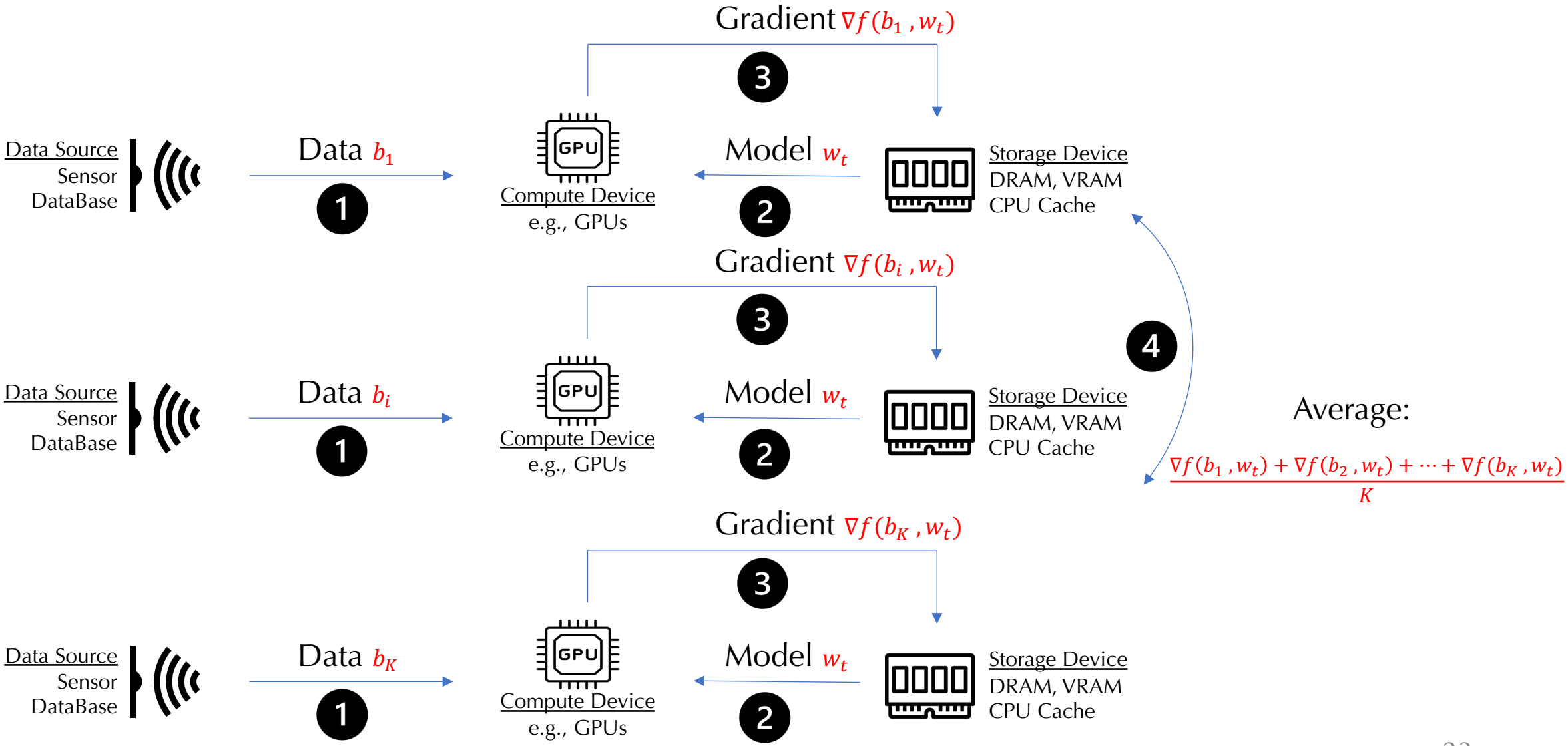
Parallel Training

Parallel Training

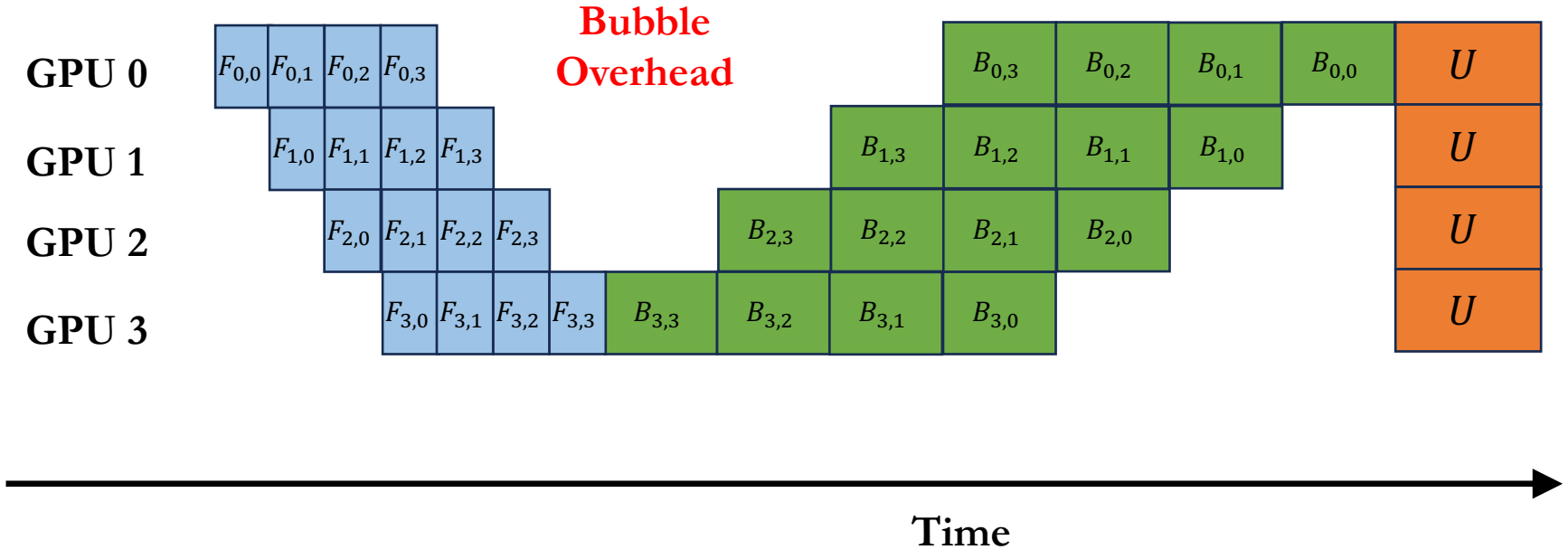
RELAXED
SYSTEM LAB

- Categories:
  - Data parallelism;
  - Pipeline parallelism;
  - Tensor model parallelism;
  - Optimizer parallelism.
- What are their communication paradigms?
  - Communication targets;
  - Communication volume.
- What are their advantages and disadvantages?

Data Parallel Training



Pipeline Parallel Training - GPipe

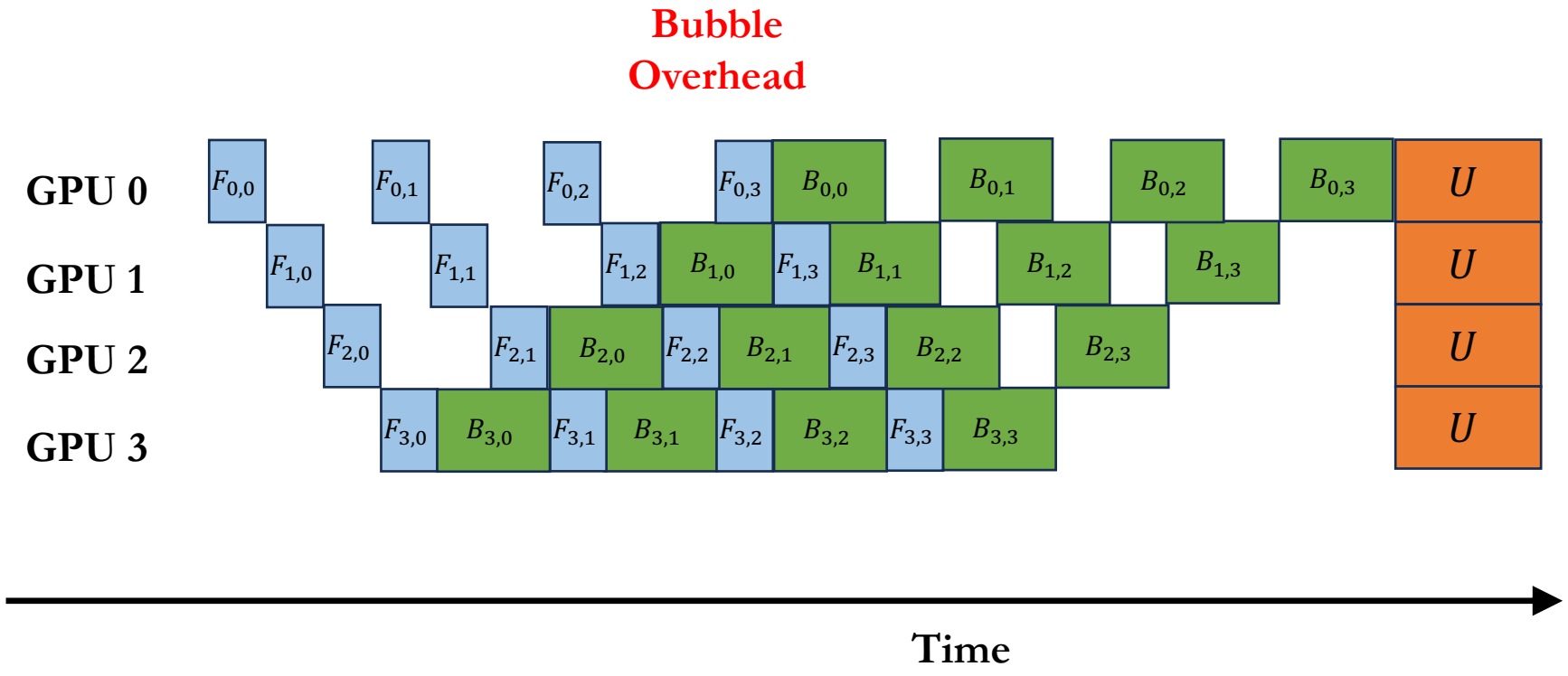


- If we ignore the computation time of optimizer updates.
- Suppose:
  - K is the number of GPUs;
  - M is the number of micro-batches;
- What is the percentage of bubble overhead? $\frac{K-1}{M+K-1}$

The number on each block represents the stage index and the micro-batch index.

Pipeline Parallel Training - 1F1B

RELAXED SYSTEM LAB

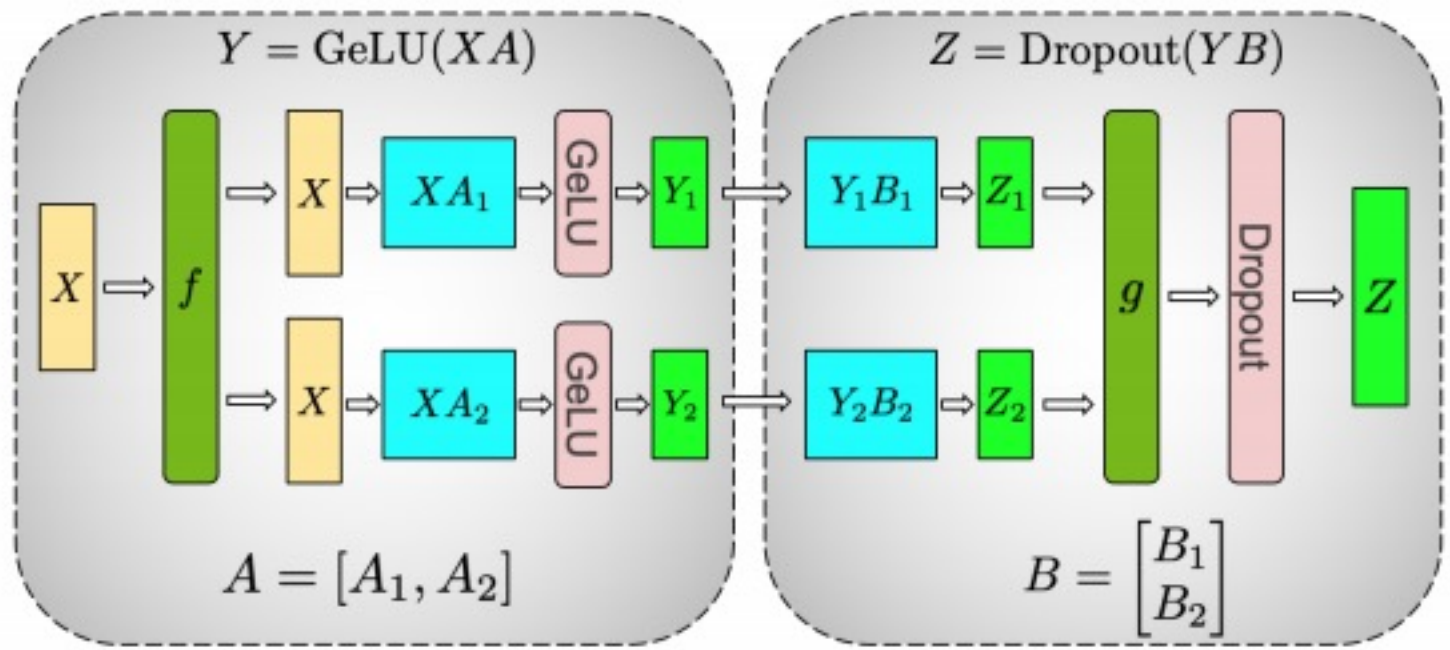


- If we ignore the computation time of optimizer updates.
- Suppose:
  - K is the number of GPUs;
  - M is the number of micro-batches;
- What is the percentage of bubble overhead? $\frac{K-1}{M+K-1}$

The number on each block represents the stage index and the micro-batch index.

Tensor Model Parallelism - MLP

RELAXED SYSTEM LAB

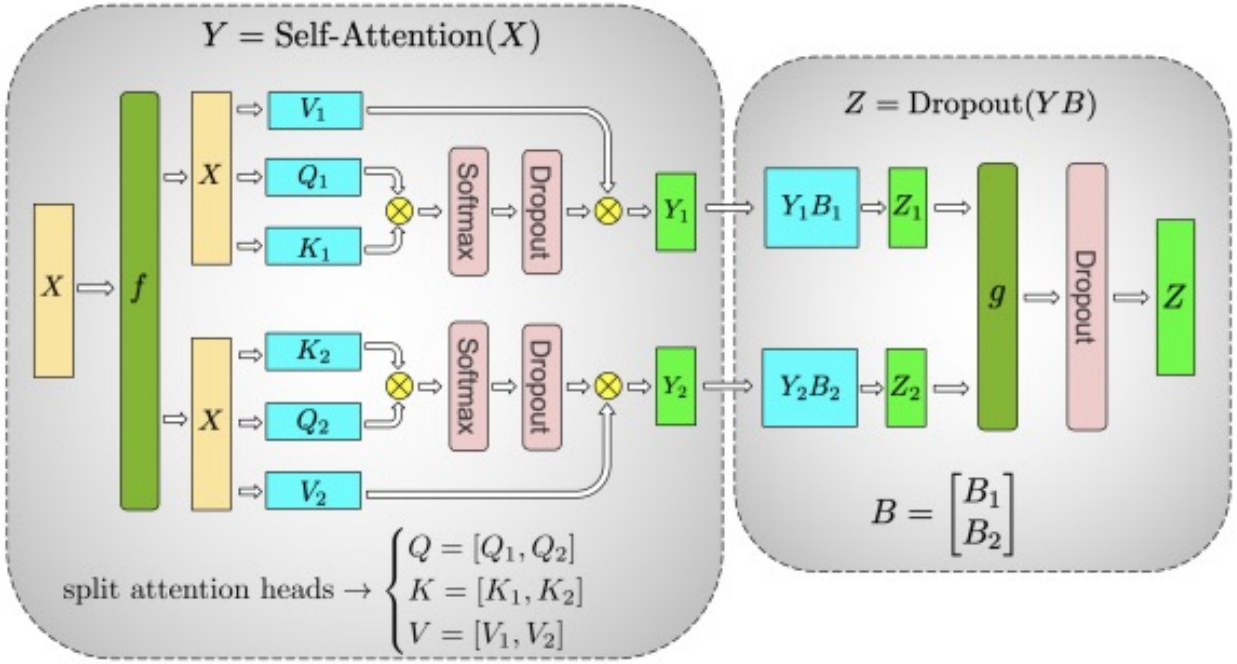


(a) MLP

- f is the identity operator in the forward pass and the **AllReduce** operator in the backward pass.
- g is the **AllReduce** operator in the forward pass and the identity operator in the backward pass.

Tensor Model Parallelism - Multi-Head Attention

RELAXED SYSTEM LAB

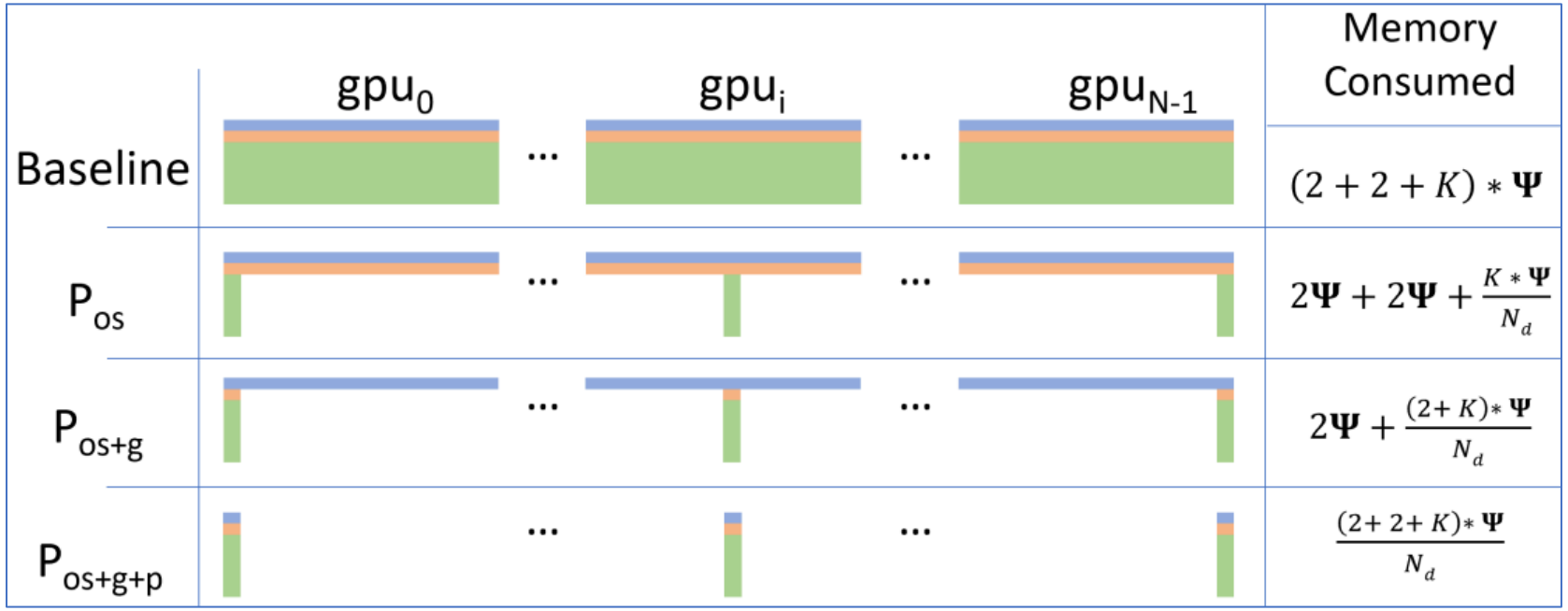


(b) Self-Attention

- f is the identity operator in the forward pass and the **AllReduce** operator in the backward pass.
- g is the **AllReduce** operator in the forward pass and the identity operator in the backward pass.

Zero Redundancy Optimizer (ZeRO)

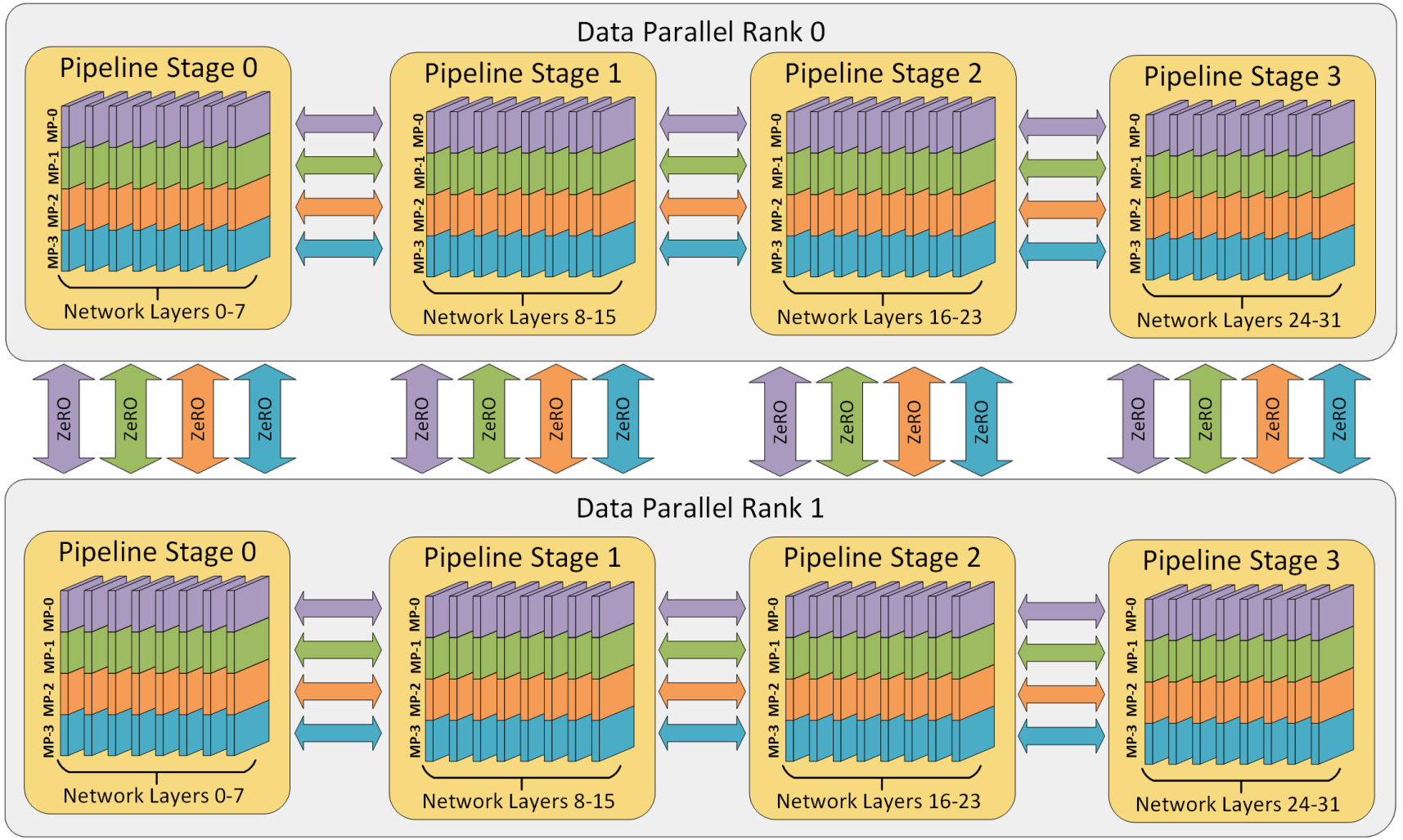
RELAXED SYSTEM LAB



- ψ is the total number of parameters;
- K denotes the memory multiplier of optimizer states;
- N_d denotes the parallel degree.

Data-, Pipeline-, Tensor Model-, Optimizer- Hybrid Parallelism

RELAXED SYSTEM LAB



https://www.deepspeed.ai/getting-started/

Relevant Presentation Topics

RELAXED SYSTEM LAB

- 9 - Data Parallel System Optimization.
- 10 - Data Parallel Algorithmic Optimization - Gradient Compression.
- 15 - Optimizer Parallel System Optimization.
- 16 - Long Sequence Parallel Training.
- 17 - Mixture of Expert Parallel Training.
- 19 - Auto Parallelism.
- 20 - Robust Training.

Parameter Efficient Fine-Tuning

RELAXED SYSTEM LAB

- *Parameter efficient fine-tuning (PEFT):* Rather than finetuning the entire model, we finetune only small amounts of weights.
  - Frozen layer/Subset fine-tuning: pick a subset of the parameters, fine-tune only those layers, and freeze the rest of the layers.
  - Adapters: add additional layers that have few parameters and tune only the parameters of those layers, keeping all others fixed.
  - Low-rank adaption (LoRA): learn a low-rank approximation of the weight matrices.

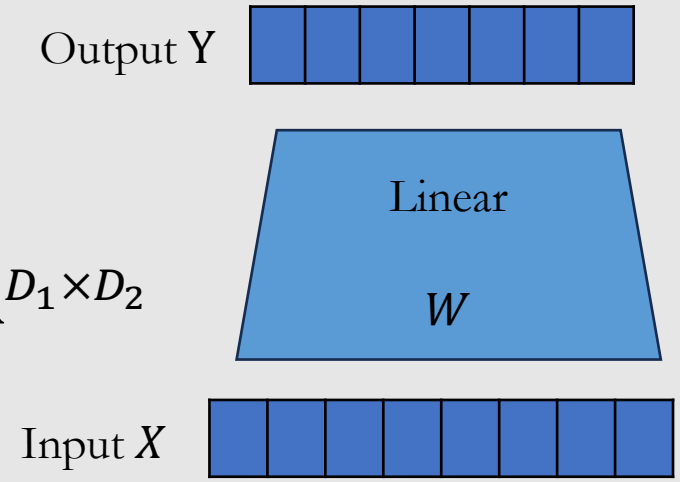
LoRA

RELAXED SYSTEM LAB

- Keep the original pre-trained parameters W fixed during fine-tuning;
- Learn an additive modification to those parameters ΔW;
- Define ΔW as a low-rank decomposition: $\Delta W = AB$.

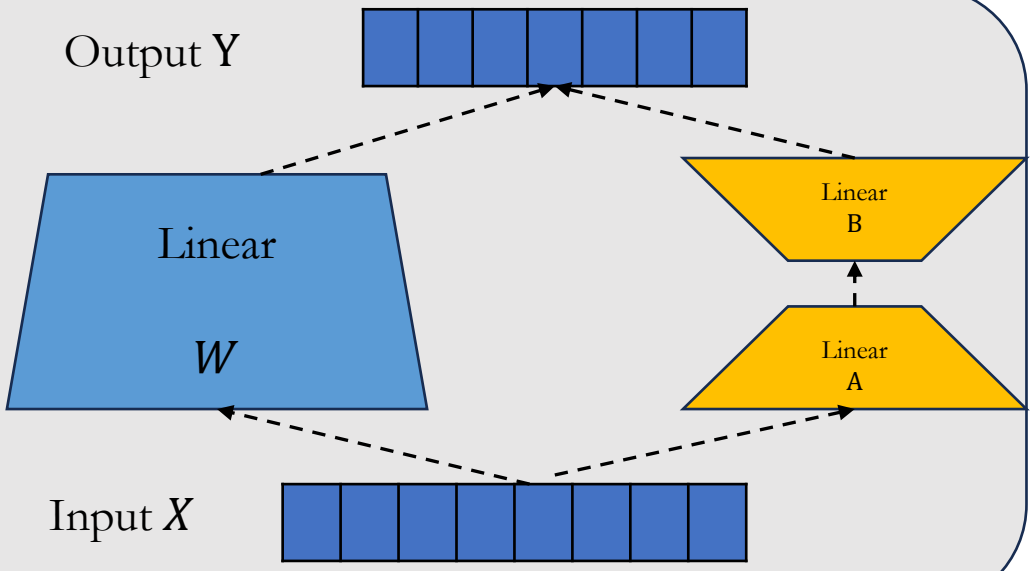
Standard Linear Layer

- $Y = XW$
- $X \in \mathbb{R}^{D_1}, Y \in \mathbb{R}^{D_2}, W \in \mathbb{R}^{D_1 \times D_2}$



LoRA Linear Layer

- $Y = XW + XAB = X(W + AB)$
- $X \in \mathbb{R}^{D_1}, Y \in \mathbb{R}^{D_2}, W \in \mathbb{R}^{D_1 \times D_2}$
- $A \in \mathbb{R}^{D_1 \times R}, B \in \mathbb{R}^{R \times D_2}$



Generative Inference

Autoregressive Generation

RELAXED SYSTEM LAB

- Autoregressive generation with two phrases computation:
  - **Prefill phrase**:
    - The model takes a prompt sequence as input and engages in the generation of a key-value cache (KV cache) for each Transformer layer.
    - Computation bounded.
  - **Decode phrase**:
    - For each decode step, the model updates the KV cache and reuses the KV to compute the output.
    - IO bounded.
- Parallel Generative Inference:
  - Pipeline parallelism;
  - Tensor model parallelism.

Prefill: TransformerBlocks($x \in R^{L\times D}$) $\rightarrow x' \in \mathbb{R}^{L\times D}$

RELAXED SYSTEM LAB

For each inference request:
- ~~$B = 1$;~~
- L is the input sequence length;
- D is the model dimension;
- Multi-head attention: $D = n_H \times H$
- H is the head dimension;
- n_h is the number of heads.

Computation	Input	Output
$Q = xW^Q$	$x \in \mathbb{R}^{L\times D}$, $W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$
$K = xW^K$	$x \in \mathbb{R}^{L\times D}$, $W^K \in \mathbb{R}^{D\times D}$	$K \in \mathbb{R}^{L\times D}$
$V = xW^V$	$x \in \mathbb{R}^{L\times D}$, $W^V \in \mathbb{R}^{D\times D}$	$V \in \mathbb{R}^{L\times D}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[K_1, K_2 \dots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{L\times D}$	$K_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[V_1, V_2 \dots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{L\times D}$	$V_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$\text{Score}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i, K_i \in \mathbb{R}^{L\times H}$	$\text{score}_i \in \mathbb{R}^{L\times L}$
$Z_i = \text{score}_i\ V_i, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{L\times L}, V_i \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$
$Z = \text{Merge}_{-1}([Z_1, Z_2 \dots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$	$Z \in \mathbb{R}^{L\times D}$
$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{L\times D}$, $W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{L\times D}$
$A = \text{Out}\ W^1$	$\text{Out} \in \mathbb{R}^{L\times D}$, $W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{L\times 4D}$
$A' = \text{relu}(A)$	$A \in \mathbb{R}^{L\times 4D}$	$A' \in \mathbb{R}^{L\times 4D}$
$x' = A'W^2$	$A' \in \mathbb{R}^{L\times 4D}$, $W^2 \in \mathbb{R}^{4D\times D}$	$x' \in \mathbb{R}^{L\times D}$

Generate the first token.

$p(x_{L+1}|x_{1:L}) = \text{softmax}(x_L W_{lm})$

Decode: TransformerBlocks($t \in R^{1\times D}$) $\rightarrow t' \in \mathbb{R}^{1\times D}$

RELAXED SYSTEM LAB

For each inference request:
- ~~$B = 1$;~~
- L is the current cached sequence length; it increases by 1 after each step.
- D is the model dimension;
- Multi-head attention: $D = n_H \times H$
- H is the head dimension;
- n_h is the number of heads.

Update the KV cache:

$K = \text{concat}(K_{\text{cache}}, K_d)$

$V = \text{concat}(V_{\text{cache}}, V_d)$

Generate the second token:

$p(x_{L+2}|x_{1:L+1}) = \text{softmax}(x_{L+1}W_{lm})$

Output of last transformer block's t'.

Computationt	Input	Output
$Q = Q_d = tW^Q$	$t \in \mathbb{R}^{1\times D}, W^Q \in \mathbb{R}^{D\times D}$	$Q, Q_d \in \mathbb{R}^{1\times D}$
$K_d = tW^K$	$t \in \mathbb{R}^{1\times D}, W^K \in \mathbb{R}^{D\times D}$	$K_d \in \mathbb{R}^{1\times D}$
$K = \text{concat}(K_{\text{cache}}, K_d)$	$K_{\text{cache}} \in \mathbb{R}^{L\times D}, K_d \in \mathbb{R}^{1\times D}$	$K \in \mathbb{R}^{(L+1)\times D}$
$V_d = tW^V$	$t \in \mathbb{R}^{1\times D}, W^V \in \mathbb{R}^{D\times D}$	$V_d \in \mathbb{R}^{1\times D}$
$V = \text{concat}(V_{\text{cache}}, V_d)$	$V_{\text{cache}} \in \mathbb{R}^{L\times D}, V_d \in \mathbb{R}^{1\times D}$	$V \in \mathbb{R}^{(L+1)\times D}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{1\times D}$	$Q_i \in \mathbb{R}^{1\times H}, i = 1, \dots n_h$
$[K_1, K_2 \dots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{(L+1)\times D}$	$K_i \in \mathbb{R}^{(L+1)\times H}, i = 1, \dots n_h$
$[V_1, V_2 \dots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{(L+1)\times D}$	$V_i \in \mathbb{R}^{(L+1)\times H}, i = 1, \dots n_h$
$\text{Score}_i = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i \in \mathbb{R}^{1\times H}, K_i \in \mathbb{R}^{(L+1)\times H}$	$\text{score}_i \in \mathbb{R}^{1\times(L+1)}$
$Z_i = \text{score}_i\, V_i, i = 1, \dots n_h$	$\text{score}_i \in \mathbb{R}^{1\times(L+1)}, V_i \in \mathbb{R}^{(L+1)\times H}$	$Z_i \in \mathbb{R}^{1\times H}$
$Z = \text{Merge}_{-1}([Z_1, Z_2 \dots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{1\times H}, i = 1, \dots n_h$	$Z \in \mathbb{R}^{1\times D}$
$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{1\times D}, W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{1\times D}$
$A = \text{Out}\, W^1$	$\text{Out} \in \mathbb{R}^{1\times D}, W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{1\times 4D}$
$A' = \text{relu}(A)$	$A \in \mathbb{R}^{1\times 4D}$	$A' \in \mathbb{R}^{1\times 4D}$
$t' = A'W^2$	$A' \in \mathbb{R}^{1\times 4D}, W^2 \in \mathbb{R}^{4D\times D}$	$t' \in \mathbb{R}^{1\times D}$

Customize Text Generation

RELAXED
SYSTEM LAB

- **Greedy search**: in every generation step, keep the token with the highest probability.
- **Beam search**: always keeps k candidates and picks the best of the candidates at the end.
- **Sampling**: instead of determinstic selecting the largest tokens, we use a random number generator to sample tokens following the distribution computed by the LM.
  - **Top-k sampling**: only the k (e.g., k = 6) most likely next words are filtered to be sampled.
  - **Top-p sampling**: chooses from the smallest possible set of words whose cumulative probability exceeds the probability p (e.g., p = 0.92).

Algorithm Optimization

RELAXED
SYSTEM LAB

- Algorithm optimization:
  - "Slightly" change the original computation at its bottleneck to make it run much faster;
- Decrease the I/O volume:
  - Model compression, i.e., quantization;
  - Knowledge distillation.
  - KV cache optimization.

Relevant Presentation Topics

RELAXED SYSTEM LAB

- 21 - LLM Inference Weight and Activation Compression.
- 22 - LLM Inference KV Cache Compression.

System Optimization

RELAXED SYSTEM LAB

- System optimization:
  - Improve the system efficiency without changing the original computation;
- General ideas
  - For each request, generate multiple tokens simultaneously:
    - Speculative decoding;
    - Parallel decoding.
  - For multiple requests, effectively batching them:
    - Continuous batching;
    - Disaggregated inference.

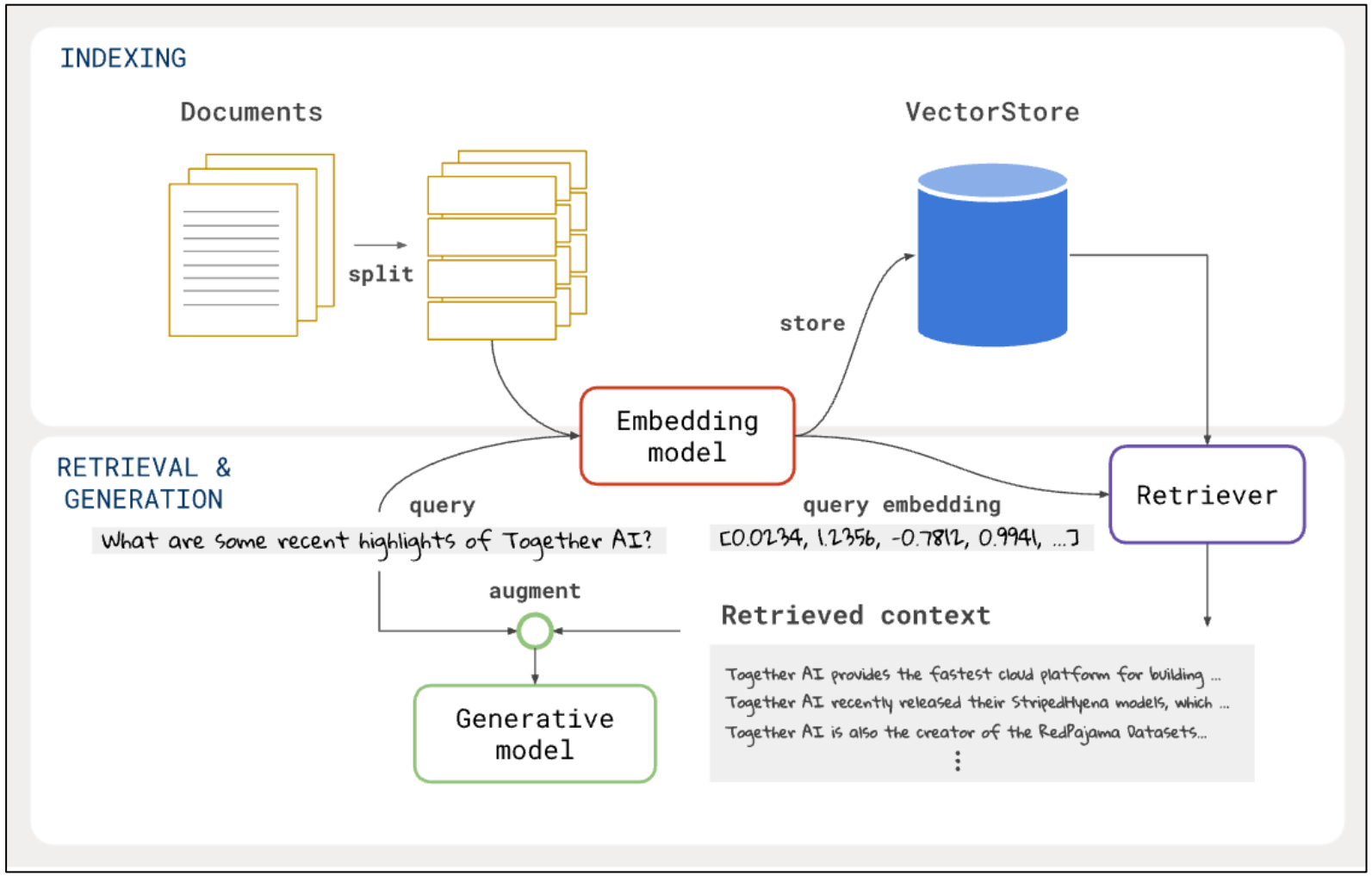
Relevant Presentation Topics

RELAXED
SYSTEM LAB

- 7 - Fairness in ML Service.
- 8 - Multi-Tenant LLM Service.
- 23 - LLM Speculative and Parallel Decoding.
- 24 - Offloading Inference System.
- 25 - Disaggregate Generative Inference.
- 26 - Parallel Inference for Diffusion Model.

RAG Retrieval and Generation

RELAXED SYSTEM LAB



Relevant Presentation Topics

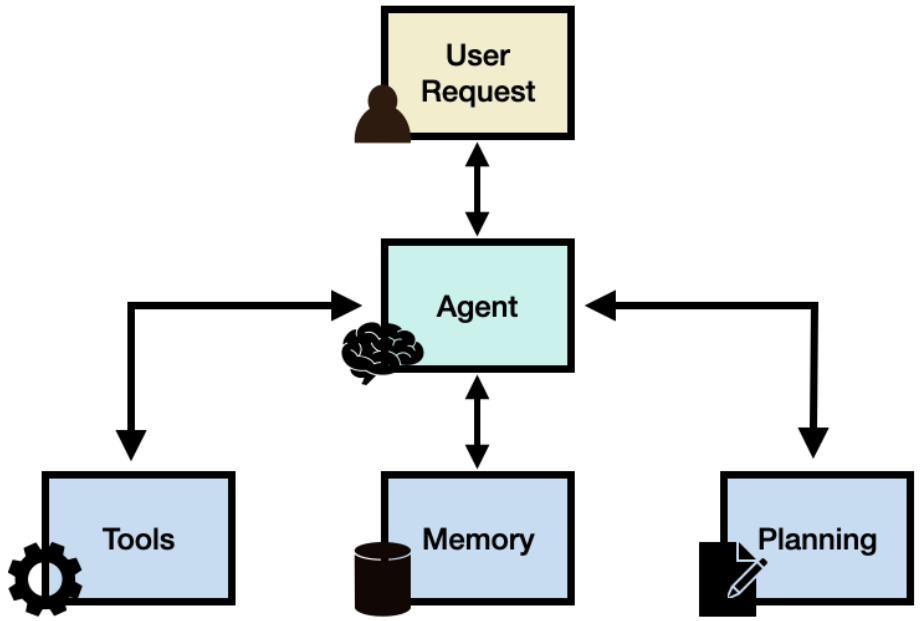
RELAXED SYSTEM LAB

- 27 - Retrieval Augmented Generation Embedding Models.
- 28 - Retrieval Augmented Generation Chunk Optimization.
- 29 - Retrieval Augmented Generation Adaptive Retrieval.
- 30 - Retrieval Augmented Generation ReRanking.
- 31 - Security in Retrieval Augmented Generation.
- 34 - Multimodal Retrieval Augmented Generation.

AI Agent & LLM Agent Framework

RELAXED SYSTEM LAB

- ***A rational agent:*** do the right thing.
- LLM agent framework: LLM applications that can execute complex tasks, which usually include:
  - **User Request**: a user question or request;
  - **Agent/Brain**: the agent core acting as coordinator;
  - **Planning**: assists the agent in planning future actions;
  - **Memory**: manages the agent's past behaviors;
  - **Tool**: interacts with external environments.



Relevant Presentation Topics

RELAXED
SYSTEM LAB

- 32 - Multi-LLM Interaction.
- 33 - Table-Augmented Generation.

Self-Proposed Topics:

RELAXED
SYSTEM LAB

- Multimodal Knowledge Graph.
- Hallucination of Large Language Model.
- Foundational Models as Agents.
- LLM for DL Library Fuzzing.
- ML System Simulation.

RELAXED
SYSTEM LAB

Logistics Again

Some Important Dates

RELAXED SYSTEM LAB

- ~~**09/05 in class**: Temporal list of presentation topics released by the lecturer.~~
- ~~**09/12 23:59**: DDL for proposal of new topics from your own interests.~~
- ~~**09/13 23:59**: Notification about whether the lecturer accepts the proposed topic.~~
- ~~**09/17 23:59**: Confirmation of the topic and presentation slot allocation by the lecturer.~~
- **Presentation slides upload**: 9:00 AM on your presentation day;
  - Send the PPT file by email to me: biyuan@ust.hk
- **Feedback for other groups**: 23:59 on that presentation day.
  - Submit through Google form, which I will send out by each lecture.
- **11/30 23:59**: Course Report (Last day of Semester)

Grading Policy

RELAXED SYSTEM LAB

- Course Report (70%):
  - Literature review (50%):
    - Cover the relevant techniques exhaustively. (10%)
    - Understand the relevant techniques correctly. (15%)
    - Organize the techniques by a good categorization. (15%)
    - The report is written in professional academic English. (10%)
    - Limits: 4 pages in NeurIPS template (excluding reference).
  - Research plan (20%):
    - The proposed research plan is executable. (10%)
    - The proposed research plan includes novelty and concrete design. (10%)
    - Limits: 4 pages in NeurIPS template (excluding reference).
- In-class Presentation (30%):
  - Clearly organize the material and present the problem definition, related work, and methodology appropriately. (20%)
  - Can answer the questions from the lecturers and other students appropriately. (5%)
  - Submit short feedback for all the other presentation sessions. (5%)
  - (Other student feedback determines 70% of the grades for this part.)

Announcement about the Presentation

RELAXED SYSTEM LAB

- Evaluation rubric:
  - The talk organizes the material with good categorization (0-5 points);
  - The talk is well-motivated (0-5 points);
    -- "I found this presentation interesting ."
  - The talk is self-explained (0-5 points);
    -- "I learned something interesting from this presentation."
  - The presenter answered the question appropriately (0-5 points).
- The presentation is in-person, you have to be ***physically here to give the presentation***.
  - Online auditing of the presentation is allowed.
- Do not miss your session, otherwise you get 0 credit for the presentation.

Some Suggestions

RELAXED
SYSTEM LAB

- Take a look at this great talk by Patrick Winston before preparing your own presentation:
  - https://www.youtube.com/watch?v=Unzc731iCUY&t=746s
  - You might not reach all those standards, but you should at least know what is right.
- Practice by yourself to make sure you are familiar with the material.
- Do not exceed the time limits:
  - Otherwise, I am sorry I will have to interrupt you and end your session.

How to Speak

Thank you, and I hope you enjoyed my lectures!

GitHub

https://github.com/Relaxed-System-Lab/COMP6211J_Course_HKUST

RELAXED
SYSTEM LAB

Thank you!