THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RELAXED SYSTEM LAB

Final Review

COMP6211J

Binhang Yuan

RELAXED
SYSTEM LAB

ML System Basics

ML System Basics

- Machine learning preliminary & PyTorch tensors:
  - Einstein notation in PyTorch.
- 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.

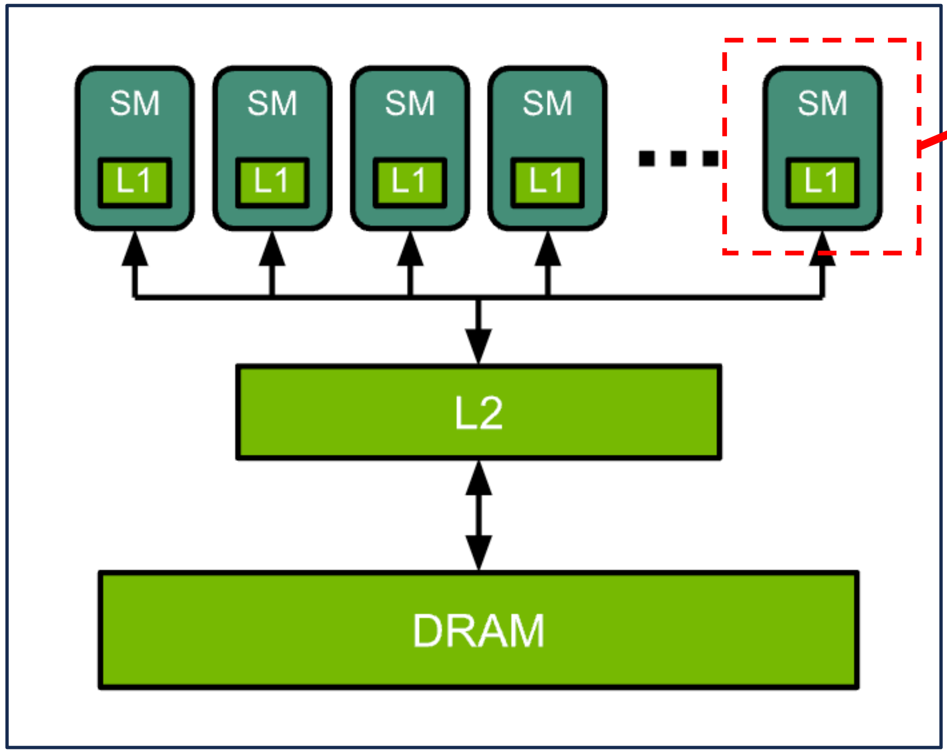
Summary of a Linear Layer Computation

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

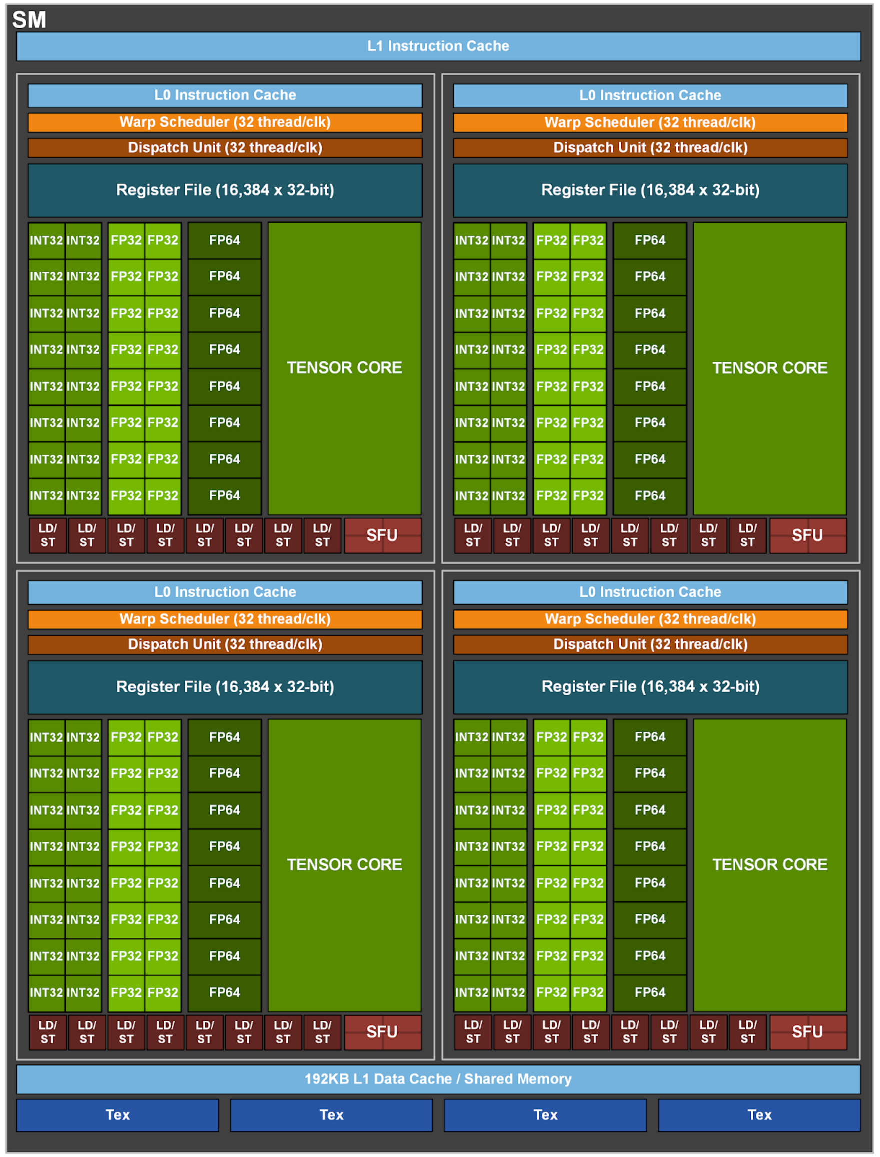
RELAXED
SYSTEM LAB

GPU Computation and Communication

Ampere GPU Architecture



108 SM in a A100 GPU



SM
L1 Instruction Cache
L0 Instruction Cache
Warp Scheduler (32 thread/clk)
Dispatch Unit (32 thread/clk)
Register File (16,384 x 32-bit)
INT32 INT32
FP32 FP32
FP64
TENSOR CORE
LD/ ST
SFU
192KB L1 Data Cache / Shared Memory
Tex

Arithmetic Intensity

- 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*.

Category 1. Architecture and Network for LLM

RELAXED SYSTEM LAB

Topic 1. AI Chip Design.

- Group 1-1
  - Member: Hongyi Wang and Xiangfeng Sun:
  - Paper: *Meta's Second Generation AI Chip: Model-Chip Co-Design and Productionization Experiences*
  - Slot: Session 1-1 (2025/10/21)
- Group 1-2
  - Member: Pengbo Li and Linkai Song
  - Paper: *The Sparsity-Aware LazyGPU Architecture*
  - Slot: Session 1-2 (2025/10/21)

Category 1. Architecture and Network for LLM

Topic 2. Heterogeneous Hardware for LLM Services.

- Group 2-1
  - Member: Shiyi Liu and Yiran Xia
  - Paper: *H2-LLM: Hardware-Dataflow Co-Exploration for Heterogeneous Hybrid-Bonding-based Low-Batch LLM Inference*
  - Slot: Session 1-3 (2025/10/21)
- Group 2-2
  - Member: Ruyi Song and Qi Liu
  - Paper: *LIA: A Single-GPU LLM Inference Acceleration with Cooperative AMX-Enabled CPU-GPU Computation and CXL Offloading*
  - Slot: Session 1-4 (2025/10/21)

NCCL Operators

- 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).

Category 1. Architecture and Network for LLM

RELAXED SYSTEM LAB

Topic 3. Communication Optimizations in LLM Serving.

- Group 3-1
  - Member: Zihao Wang and Zhaoxiang Bao
  - Paper: *MCCS: A Service-based Approach to Collective Communication for Multi-Tenant Cloud*
  - Slot: Session 2-1 (2025/10/23)
- Group 3-2
  - Member: Yichen Liu and Jichen Zhang
  - Paper: *Rdma over ethernet for distributed training at meta scale*
  - Slot: Session 2-2 (2025/10/23)

Language Model

Autoregressive Language Models

- 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*:

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

- 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).

Category 3. Algorithmic Advances for LLM

RELAXED SYSTEM LAB

Topic 9. Diffusion Language Model.

- Group 9-1
  - Member: Shiyuan Song and Junjie Hou
  - Paper: *Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution*
  - Slot: Session 4-4 (2025/10/30)
- Group 9-2
  - Member: Chaolei Tan and Ziqi Jiang
  - Paper: *Large language diffusion models*
  - Slot: Session 5-1 (2025/11/04)

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

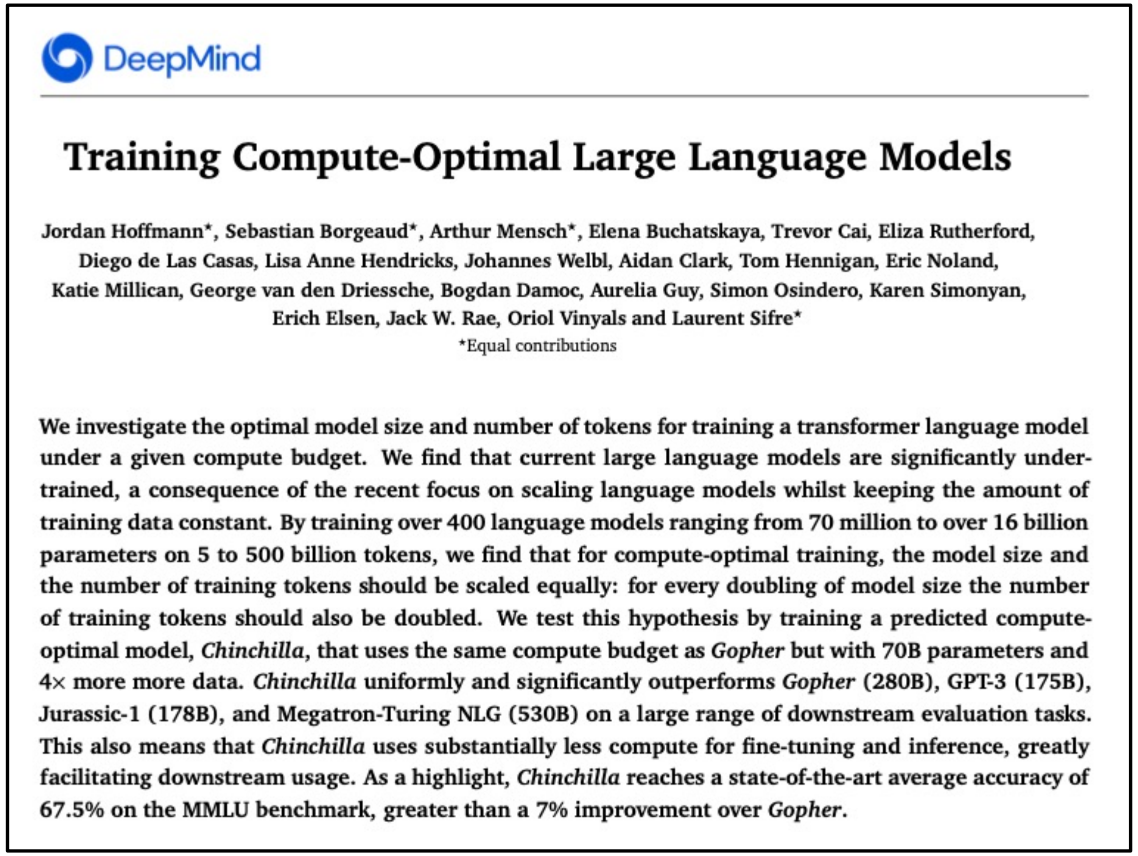
RELAXED SYSTEM LAB

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

Scaling Law

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


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 undertrained, 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

- 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.

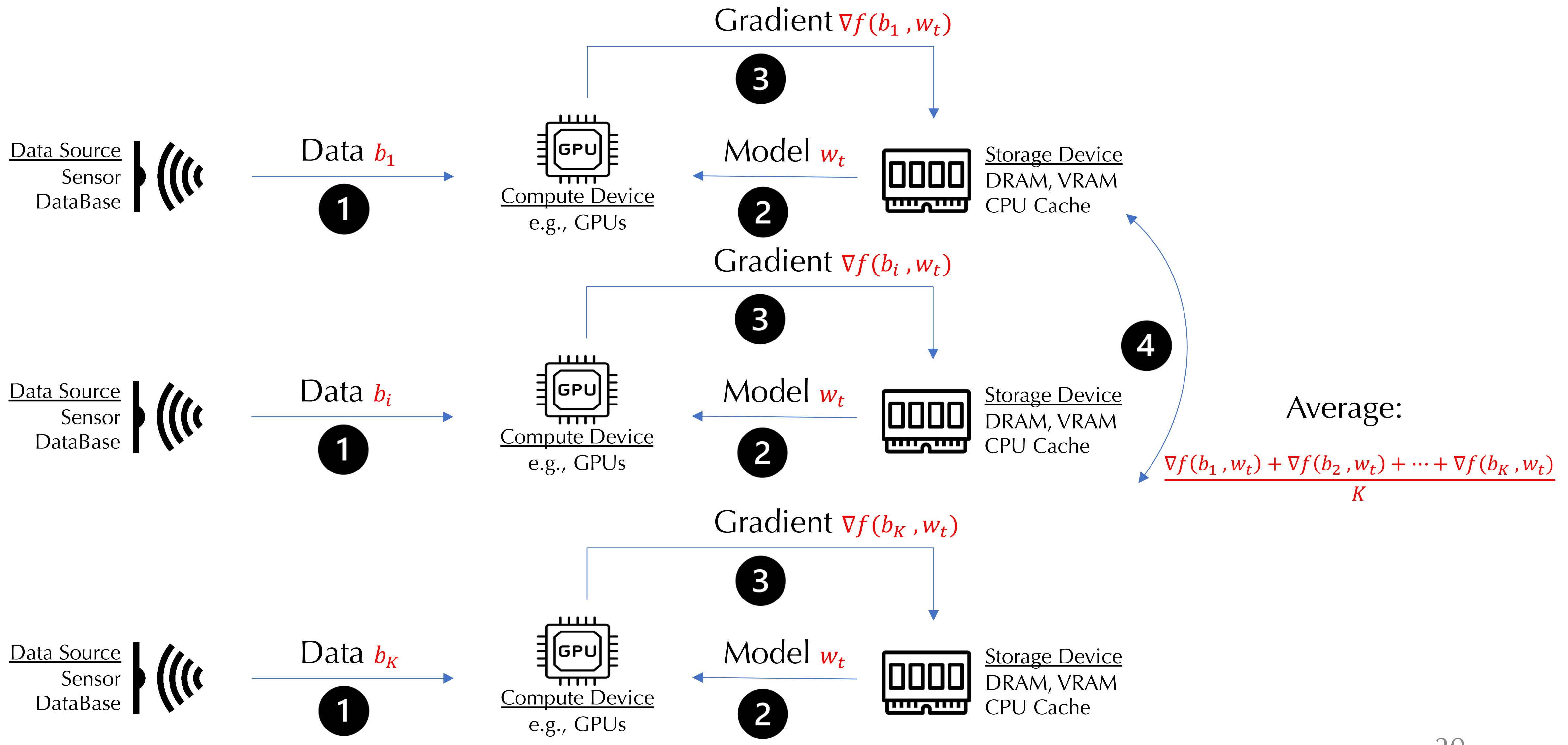
Parallel Training

Parallel Training

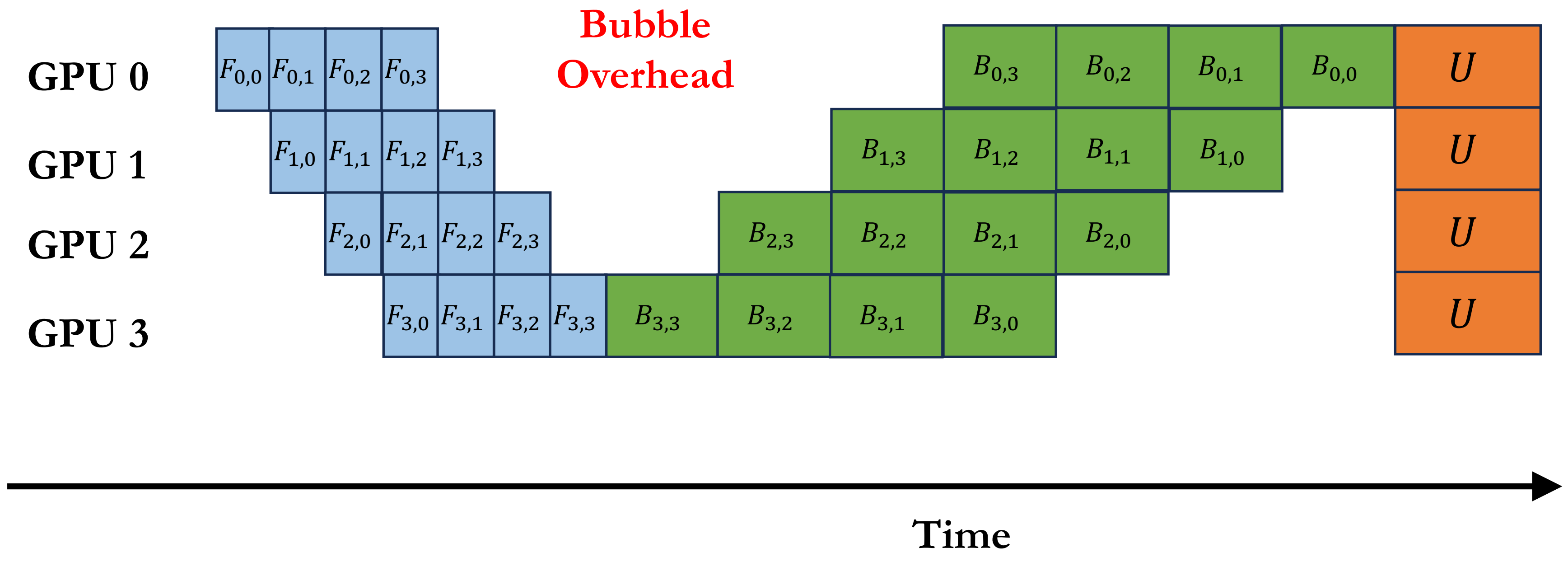
- Categories:
    - Data parallelism;
    - Pipeline parallelism;
    - Tensor model parallelism;
    - Optimizer parallelism.
    - MoE parallelism;
    - Parallelism for long sequence.
- What are the communication paradigms?
    - Communication targets;
    - (Collective) Communication operator(s);
    - Communication volume (i.e., corresponding number of bytes of the communication targets).
- What are their advantages and disadvantages?

Data Parallel SGD

RELAXED SYSTEM LAB



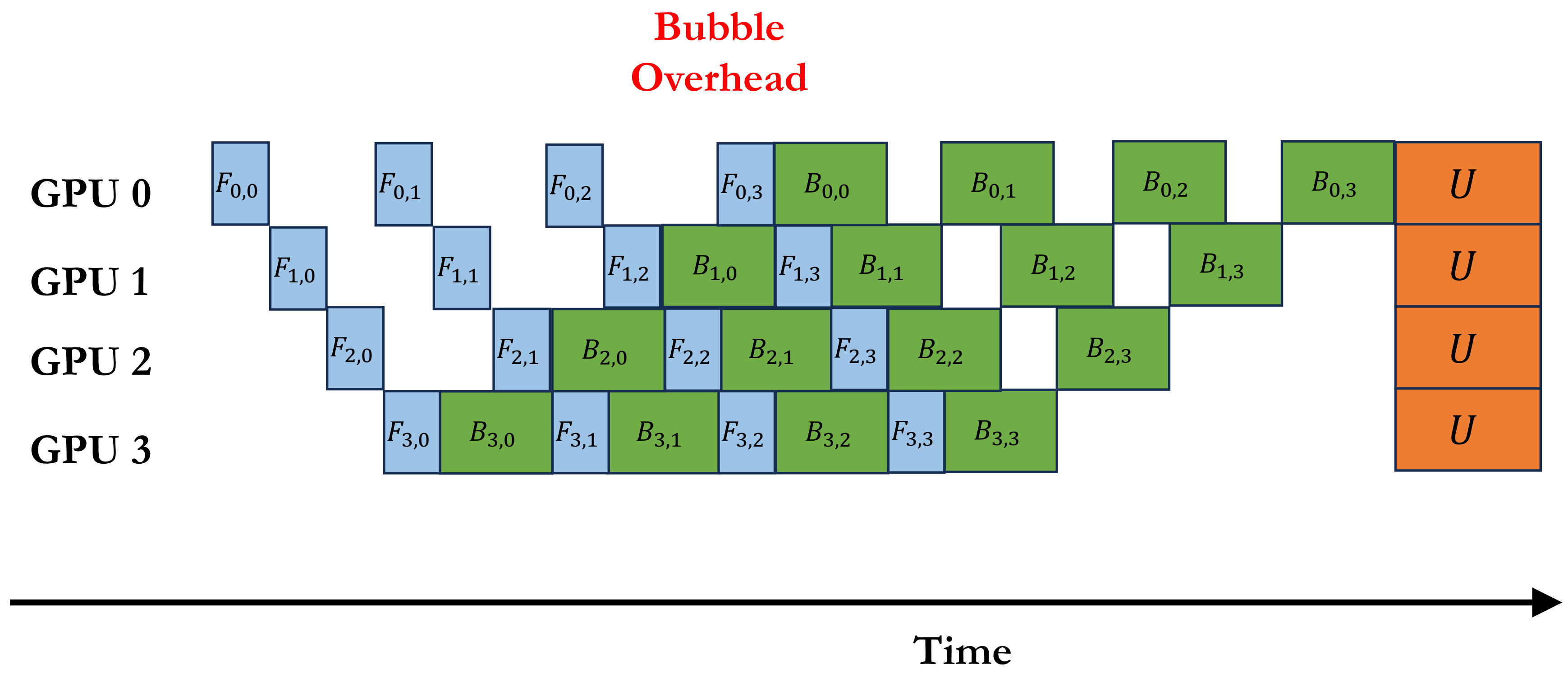
Pipeline Parallelism - GPipe


Bubble Overhead
GPU 0
GPU 1
GPU 2
GPU 3
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$
$F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$
$F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$
$F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$
$B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$
$B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$
$B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$
$B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$
U
Time

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

- 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? $\frac{K-1}{M+K-1}$

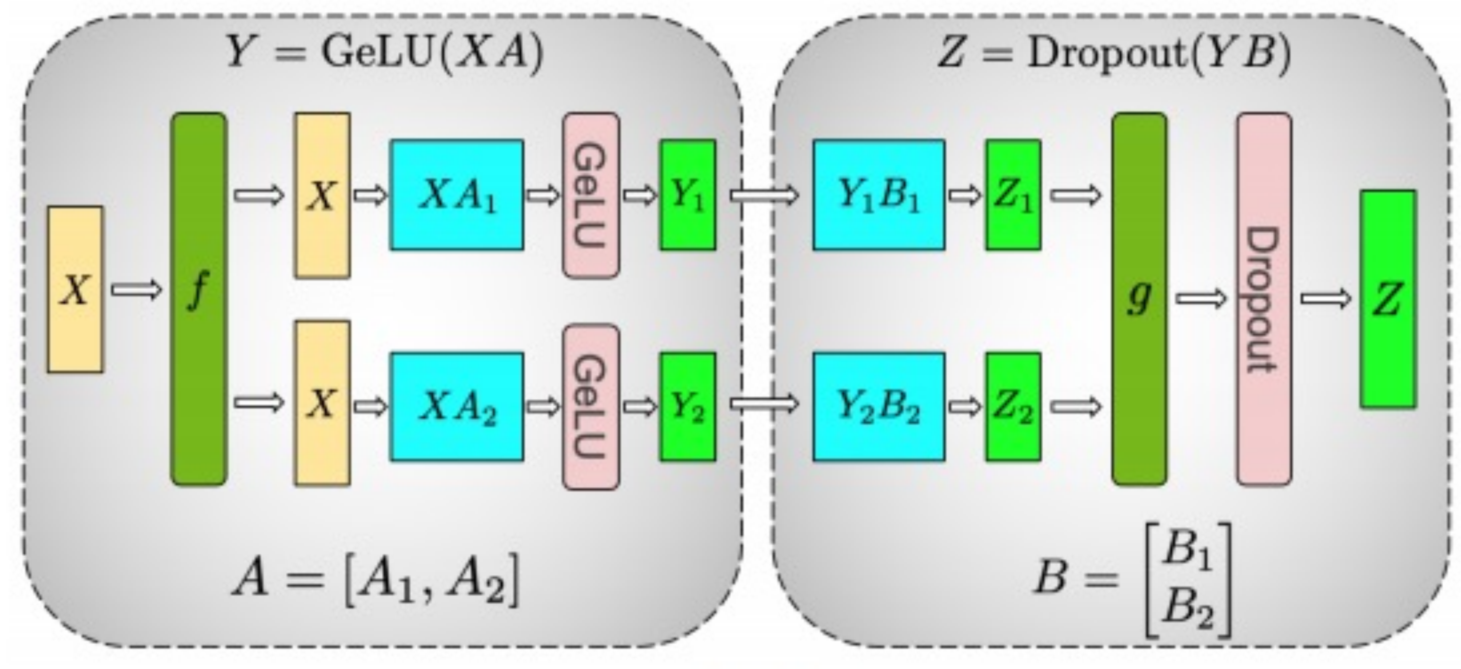
Pipeline Parallelism - 1F1B



- 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.

MLP in Tensor Model Parallelism

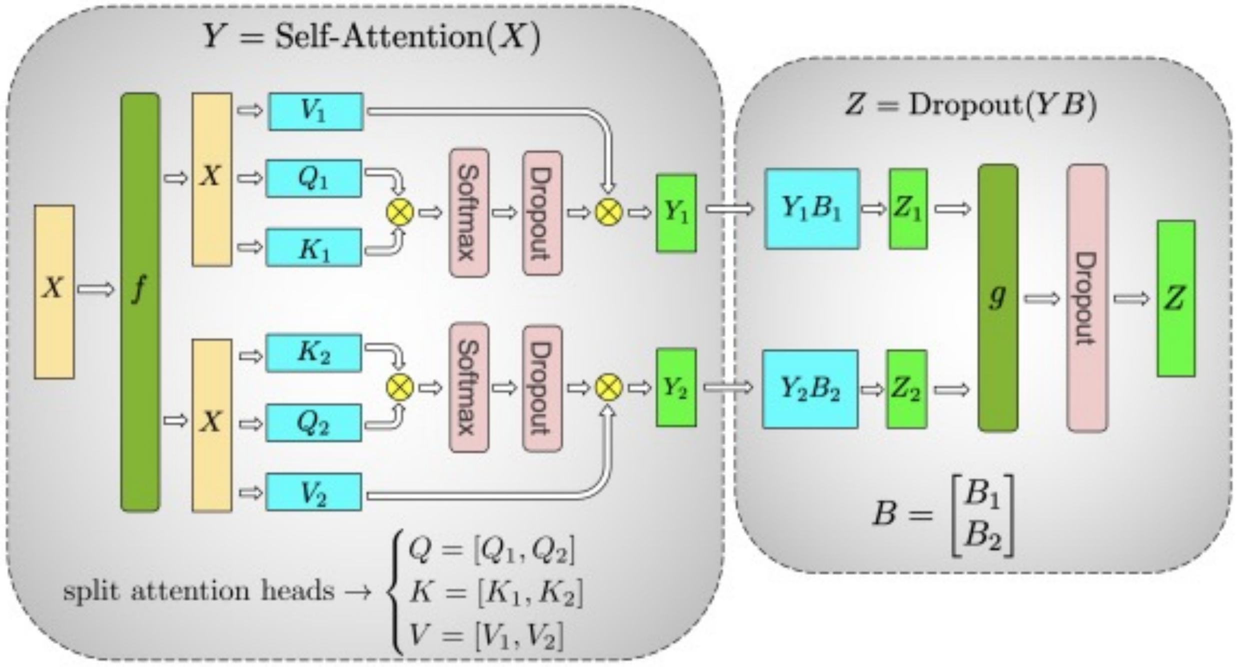


(a) MLP

- 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.

Multi-Head Attention in Tensor Model Parallelism

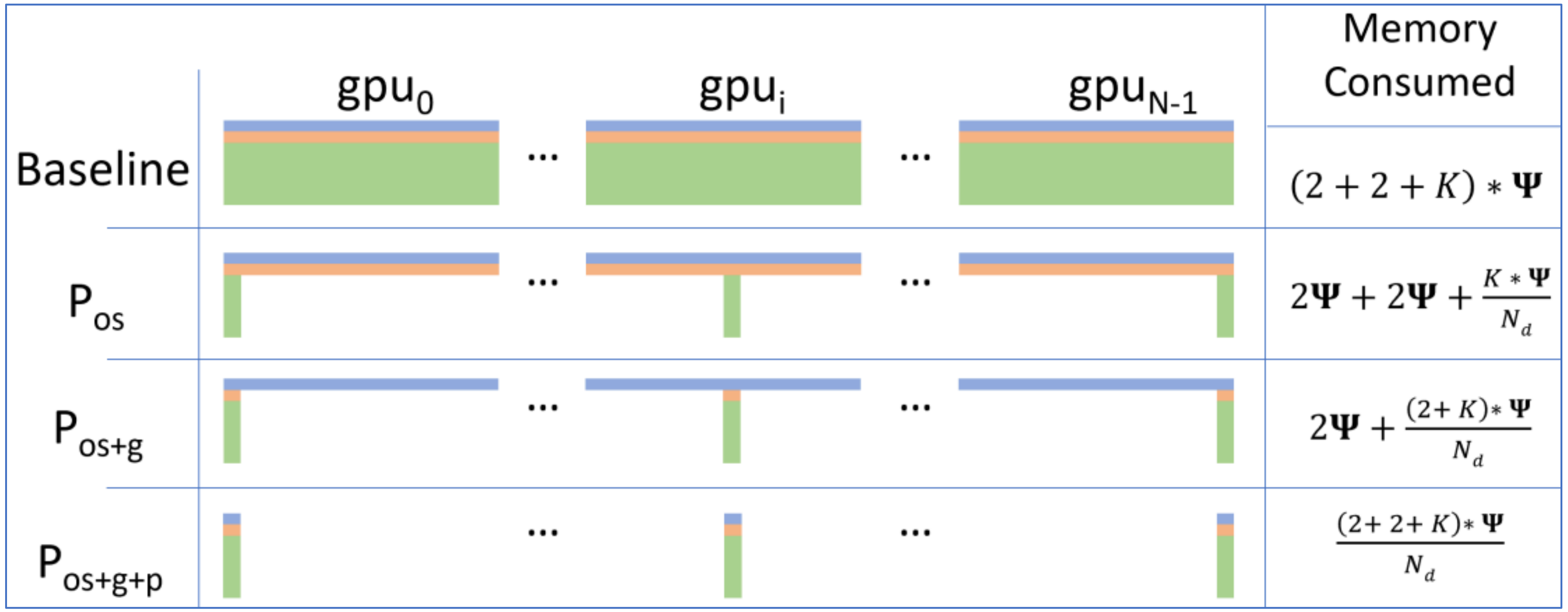
RELAXED SYSTEM LAB



(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.

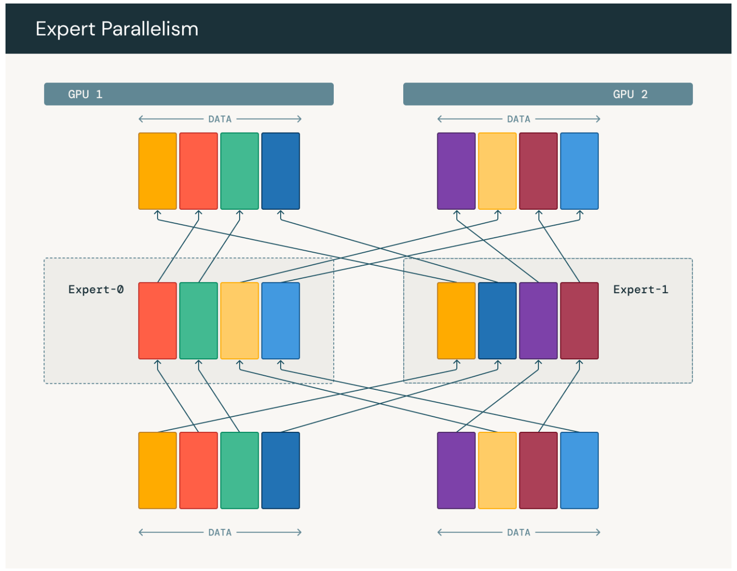
Zero Redundancy Optimizer (ZeRO)

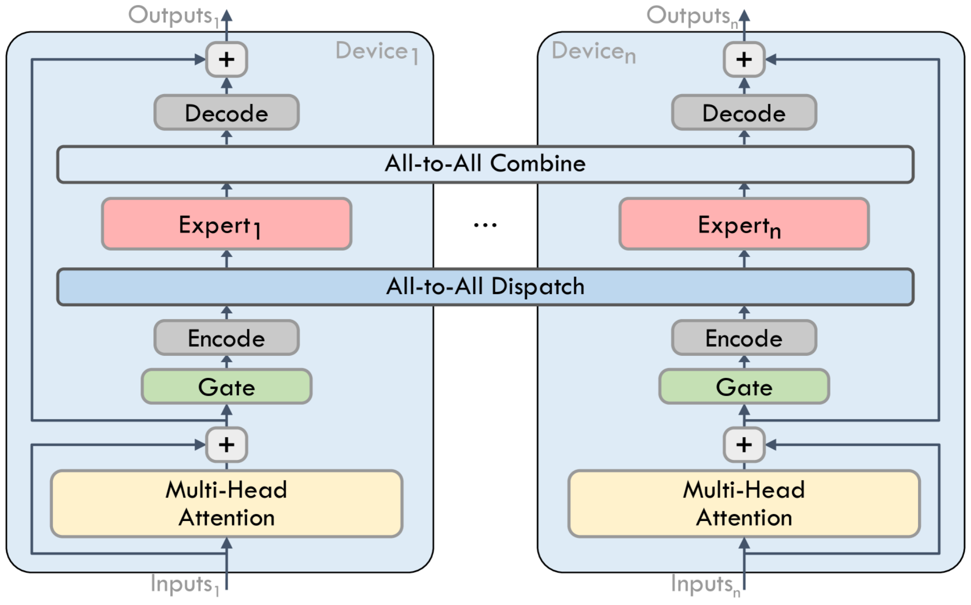

gpu$_0$
gpu$_i$
gpu$_{N-1}$
Memory Consumed
Baseline
$(2+2+K)*\Psi$
P$_{os}$
$2\Psi + 2\Psi + \frac{K*\Psi}{N_d}$
P$_{os+g}$
$2\Psi + \frac{(2+K)*\Psi}{N_d}$
P$_{os+g+p}$
$\frac{(2+2+K)*\Psi}{N_d}$

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

MoE Parallelism

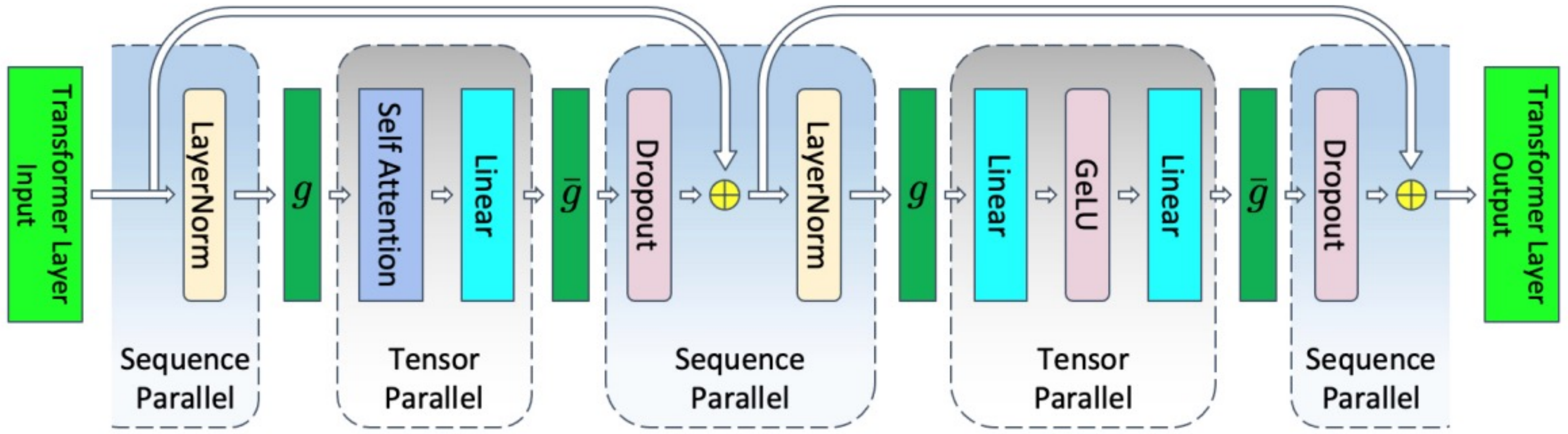
- In the forward pass:
  - Two **AlltoAll** operations to communicate the activations:
    - One **AlltoAll** operation dispatches the routed activations in the current data batch to the corresponding experts;
    - One **AlltoAll** operation combines the computed expert output in the data batch that needs to be processed by the device.
  - Notice that there could be a little additional overhead to first share the shape of the dispatch tensors.
- In the backward pass:
  - Two **AlltoAll** operations to communicate the corresponding gradients of the activations back to the original device.
  - **AllReduce** operations to synchronize the gradients of non-expert model weights.





RELAXED SYSTEM LAB

Megatron Sequence Parallelism

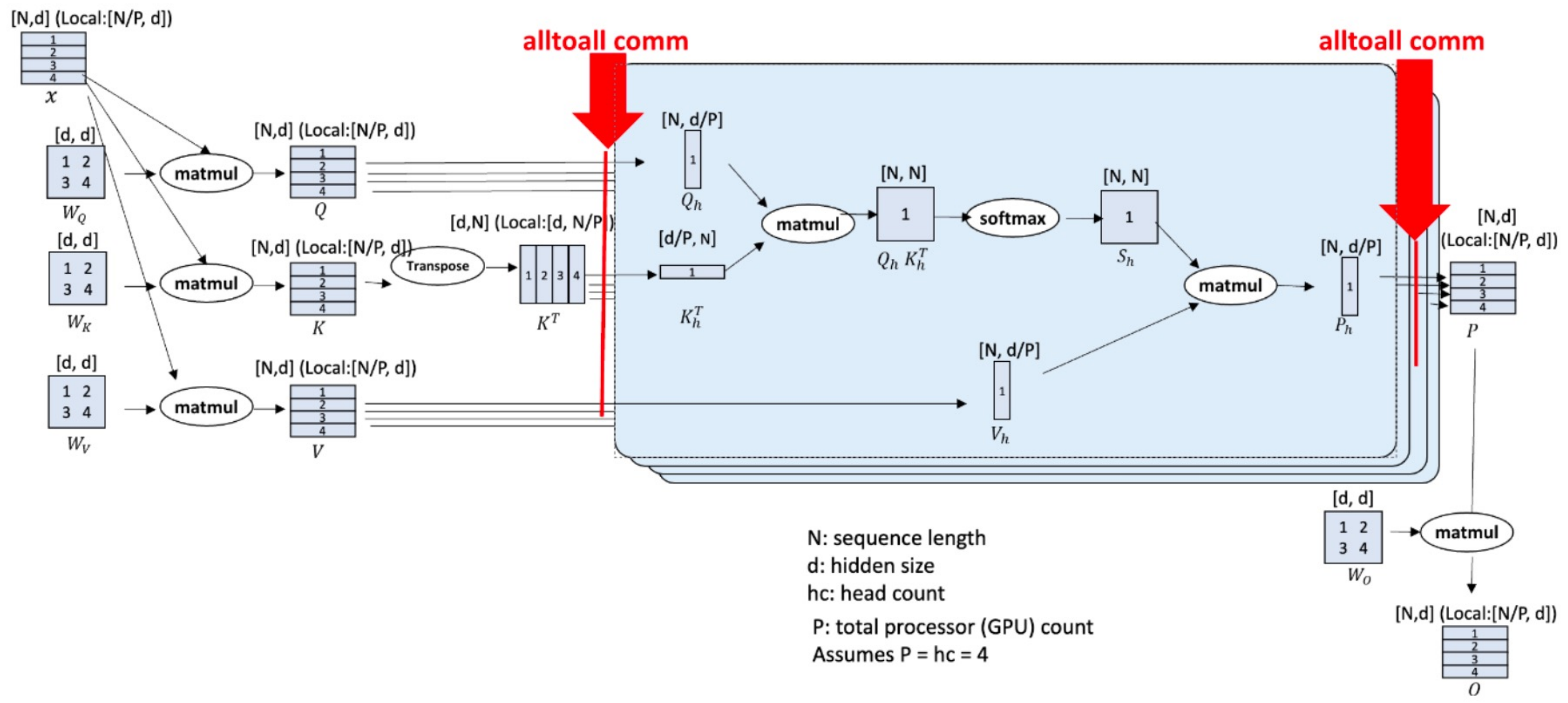


<u>In tensor model parallelism combined with sequence parallelis</u>:
g is **AllGather** operation in the forward pass and **ReduceScatter** in the backward pass.
$\bar{g}$ is **ReduceScatter** in the forward pass and **AllGather** operation in the backward pass.

- Sequence parallelism is an extension of the previous tensor model parallelism:
- Partitioning along the sequence dimension reduces the memory required for the activations. This extra level of parallelism introduces new collective communication paradigm.

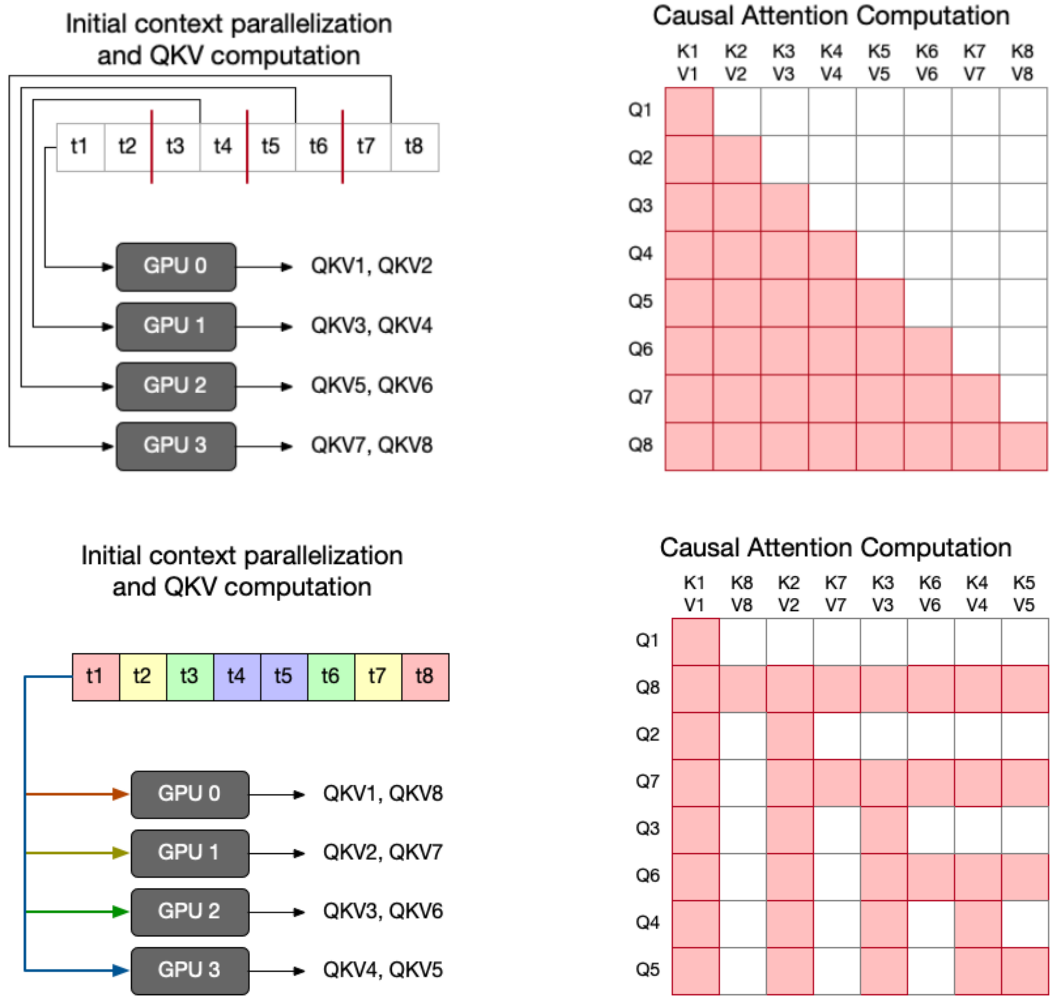
Deepspeed-Ulysses

- **Core idea:**
  - Partition the computation of attention heads among different GPUs;
  - Use one All-to-All to shuffle the input Query, Key, Value;
  - Use one All-to-All to collect the corresponding output.



Ring Attention/Megatron Context Parallelism

- Ring attention computation for the attention mechanism:
  - Split the data by sequence dimension;
  - Iteratively accumulate local attention results by iterating Q and K/V as a nested loop;
  - Each GPU holds a portion of queries, and sends its keys and values to the next GPU and receives them from the previous GPU (ring).
- A trivial issue to be fixed in Ring Attention: the load-balance in each iteration:
  - Megatron context parallelism implementation: an optimization not to compute unnecessary masked parts to balance the workflow.



Initial context parallelization and QKV computation
t1 t2 t3 t4 t5 t6 t7 t8
GPU 0 → QKV1, QKV2
GPU 1 → QKV3, QKV4
GPU 2 → QKV5, QKV6
GPU 3 → QKV7, QKV8
Causal Attention Computation
K1 V1, K2 V2, K3 V3, K4 V4, K5 V5, K6 V6, K7 V7, K8 V8
Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8
Initial context parallelization and QKV computation
t1 t2 t3 t4 t5 t6 t7 t8
GPU 0 → QKV1, QKV8
GPU 1 → QKV2, QKV7
GPU 2 → QKV3, QKV6
GPU 3 → QKV4, QKV5
Causal Attention Computation
K1 V1, K8 V8, K2 V2, K7 V7, K3 V3, K6 V6, K4 V4, K5 V5
Q1 Q8 Q2 Q7 Q3 Q6 Q4 Q5

https://insujang.github.io/2024-09-20/introducing-context-parallelism

Category 2. LLM Training Inference and RL Systems

Topic 4. Long Context LLM Training.

- Group 4-1
  - Member: Wong Hiu Tung and Ma Sum Yi
  - Paper: *DISTFLASHATTN: Distributed Memory-efficient Attention for Long-context LLMs Training*
  - Slot: Session 2-3 (2025/10/23)
- Group 4-2
  - Member: Dakai An and Tianyu Feng
  - Paper: *ByteScale: Communication-Efficient Scaling of LLM Training with a 2048K Context Length on 16384 GPUs*
  - Slot: Session 2-4 (2025/10/23)

Category 2. LLM Training Inference and RL Systems

Topic 5. Mixture of Expert Training.

- Group 5-1
    - Member: Zhenghong Huang and Wenxi Qiu
    - Paper: *Comet: Fine-grained computation-communication overlapping for mixture-of-experts*
    - Slot: Session 3-1 (2025/10/28)
- Group 5-2
    - Member: Wenkai Li and Mengming Li
    - Paper: *Hetermoe: Efficient training of mixture-of-experts models on heterogeneous gpus*
    - Slot: Session 3-2 (2025/10/28)

RELAXED
SYSTEM LAB

Generative Inference

Autoregressive Generation

- Autoregressive generation with two phrases computation:
  - **<u>Prefill phrase</u>**:
    - 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.
  - **<u>Decode phrase</u>**:
    - 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
$Q = XW^Q$	$X \in \mathbb{R}^{L\times D}$, $W^Q \in \mathbb{R}^{D\times D}$	$Q \in \mathbb{R}^{L\times D}$
$K = XW^K$	$X \in \mathbb{R}^{L\times D}$, $W^K \in \mathbb{R}^{D\times D}$	$K \in \mathbb{R}^{L\times D}$
$V = XW^V$	$X \in \mathbb{R}^{L\times D}$, $W^V \in \mathbb{R}^{D\times D}$	$V \in \mathbb{R}^{L\times D}$
$[Q_1, Q_2 \dots, Q_{n_h}] = \text{Partition}_{-1}(Q)$	$Q \in \mathbb{R}^{L\times D}$	$Q_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[K_1, K_2 \dots, K_{n_h}] = \text{Partition}_{-1}(K)$	$K \in \mathbb{R}^{L\times D}$	$K_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$[V_1, V_2 \dots, V_{n_h}] = \text{Partition}_{-1}(V)$	$V \in \mathbb{R}^{L\times D}$	$V_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$
$\text{Score}_\text{i} = \text{softmax}(\frac{Q_i K_i^T}{\sqrt{D}}), i = 1, \dots n_h$	$Q_i, K_i \in \mathbb{R}^{L\times H}$	$\text{score}_\text{i} \in \mathbb{R}^{L\times L}$
$Z_i = \text{score}_i\, V_i, i = 1, \dots n_h$	$\text{score}_\text{i} \in \mathbb{R}^{L\times L}, V_i \in \mathbb{R}^{L\times H}$	$Z_i \in \mathbb{R}^{L\times H}$
$Z = \text{Merge}_{-1}\,([Z_1, Z_2 \dots, Z_{n_h}])$	$Z_i \in \mathbb{R}^{L\times H}, i = 1, \dots n_h$	$Z \in \mathbb{R}^{L\times D}$
$\text{Out} = ZW^O$	$Z \in \mathbb{R}^{L\times D}$, $W^O \in \mathbb{R}^{D\times D}$	$\text{Out} \in \mathbb{R}^{L\times D}$
$A = \text{Out}\, W^1$	$\text{Out} \in \mathbb{R}^{L\times D}$, $W^1 \in \mathbb{R}^{D\times 4D}$	$A \in \mathbb{R}^{L\times 4D}$
$A' = \text{relu}(A)$	$A \in \mathbb{R}^{L\times 4D}$	$A' \in \mathbb{R}^{L\times 4D}$
$X' = A'W^2$	$A' \in \mathbb{R}^{L\times 4D}$, $W^2 \in \mathbb{R}^{4D\times D}$	$X' \in \mathbb{R}^{L\times D}$

Generate the first token.

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

-1 indicates the last TransformerBlock

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

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.

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

Update the KV cache:

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

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

Generate the second token:

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

-1 indicates the last TransformerBlock

Customize Text Generation

- **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).

Generative Inference Optimization

- 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.

Generative Inference Optimization

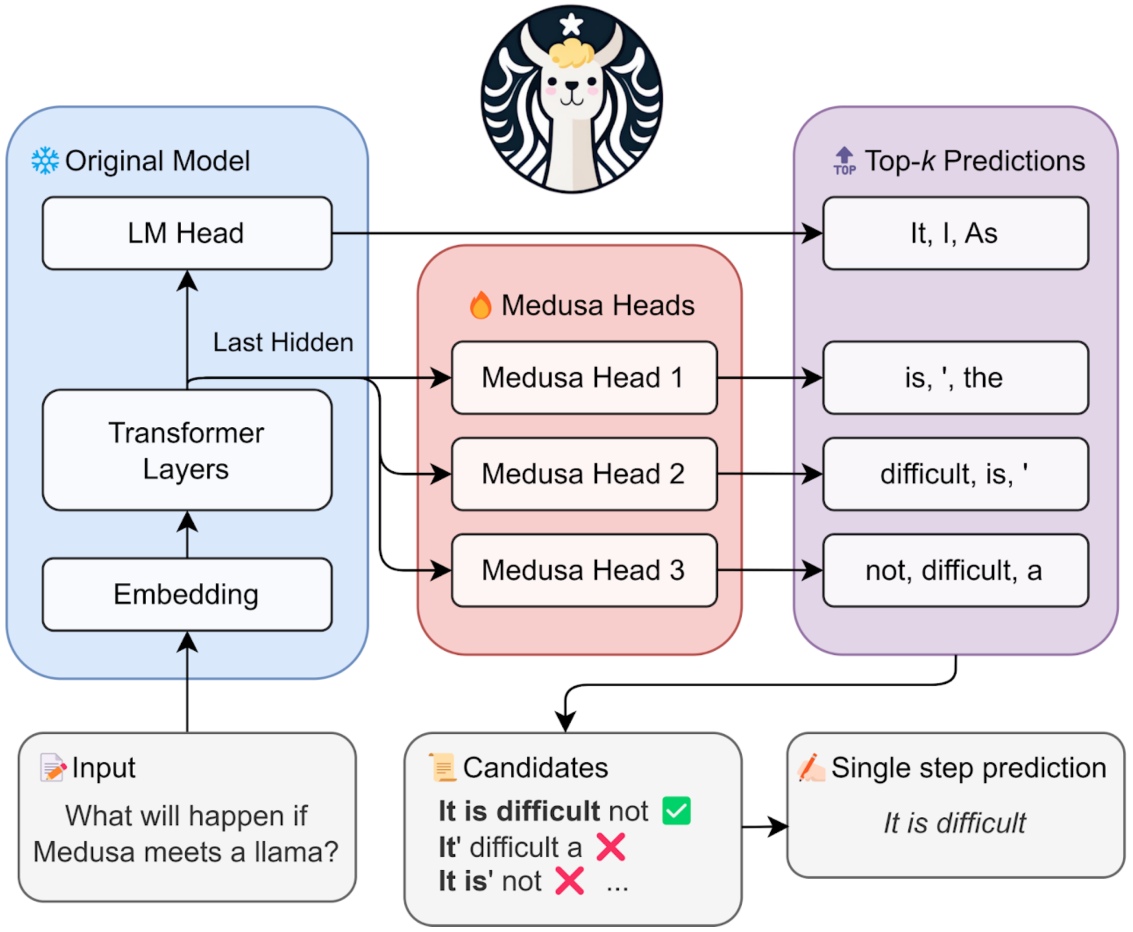
- System optimization:
  - Improve the system efficiency without changing the original computation;
- General ideas
  - For each request, generate multiple tokens simultaneously:
    - Speculative decoding;
    - Parallel decoding.
  - For multiple requests, effectively batching them:
    - Continuous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

Speculative Decoding

- **Observation**:
  - A small *assistant/speculative model* very often generates the same tokens as the large original LLM (sometime s referred to as the *main/target model*).
- Speculative decoding overview:
  - The assistant model auto-regressively generates a sequence of N candidate tokens;
  - The candidate tokens are passed to the original LLM to be verified. The original model takes the candidate tokens as input and performs ***a single forward pass***:
    - All candidate tokens up to the first mismatch are correct;
    - After the first mismatch, replace the first incorrect candidate token with the correct token from the main model (fox), and discard all predicted tokens that come after this mismatched token.
  - Repeat this process until the end condition is reached.

Parallel Decoding

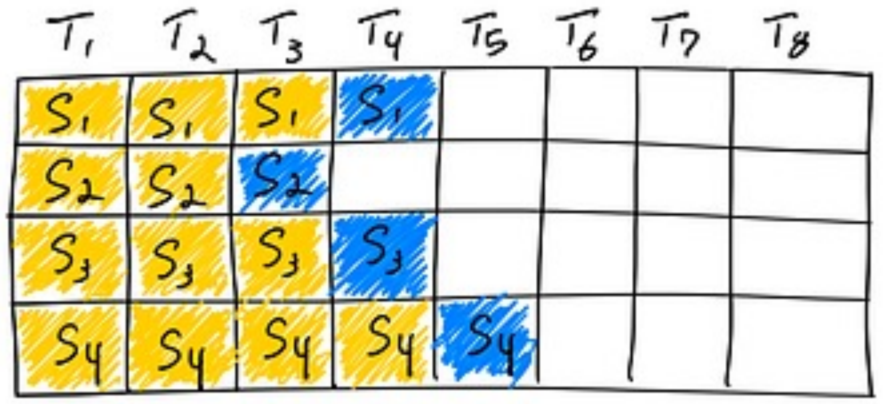
- Rather than pulling in an entirely new assistant model to predict subsequent tokens, Medusa simply extends the original LLM itself.
- Medusa heads are the additional decoding heads built on top of the last hidden states of the LLM, enabling the prediction of several subsequent tokens in parallel.
- Each Medusa head is a single layer of feed-forward network, augmented with a residual connection.
- Training these heads is straightforward: for a relatively small dataset, the original model remains static; only the Medusa heads are updated.

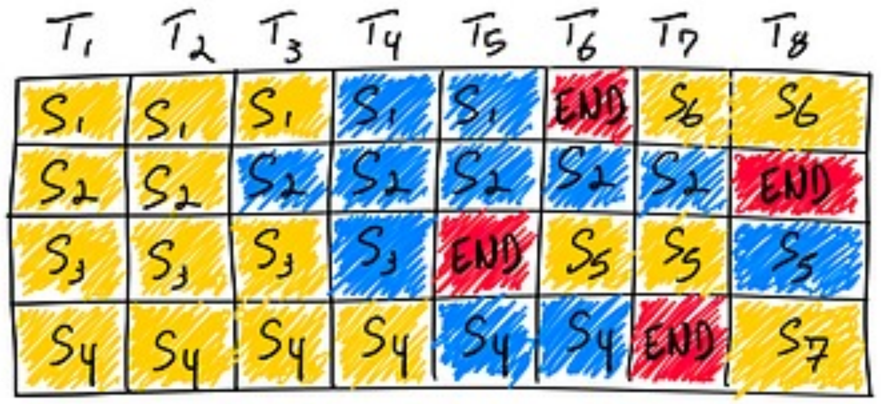


https://www.together.ai/blog/medusa

Continuous Batching

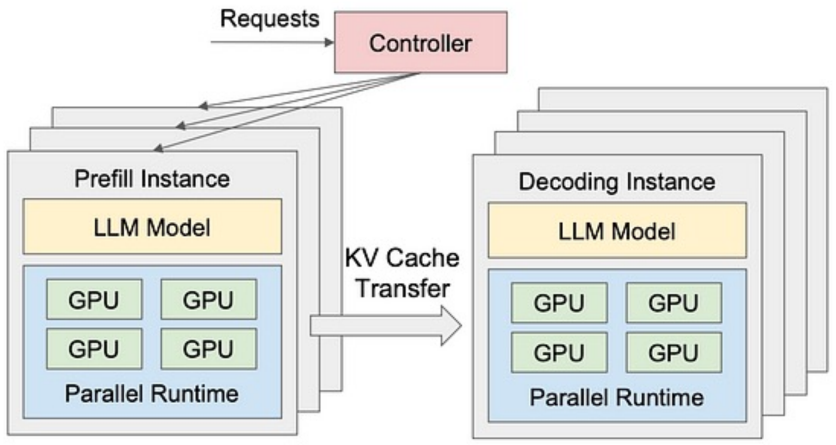
- Orca: https://www.usenix.org/conference/osdi22/presentation/yu
- Key idea:
  - Instead of waiting until every sequence in a batch has completed generation, Orca implements iteration-level scheduling;
  - Once a sequence in a batch has completed generation, a new sequence can be inserted in its place;
  - Higher GPU utilization than static batching.





Disaggregated Inference

- DistServe: https://www.usenix.org/system/files/osdi24-zhong-yinmin.pdf
- Key ideas:
  - Prefill computation on some GPUs;
  - Decoding computation on some other GPUs;
  - Prefill and decoding instances can have different parallel configurations;
  - Dynamic configuration of the prefill / decoding ratio;
  - Overhead: KV-cache communication.



Category 2. LLM Training Inference and RL Systems

RELAXED
SYSTEM LAB

Topic 6. Prefill-Decoding Disaggregated Inference.

- Group 6-1
  - Member: Bu Jin and Yu Liu
  - Paper: *Mooncake: Trading more storage for less computation—a KVCache-centric architecture for serving LLM chatbot*
  - Slot: Session 3-3 (2025/10/28)
- Group 6-2
  - Member: Heyang Sun and Xu Xu
  - Paper: *Taming the Chaos: Coordinated Autoscaling for Heterogeneous and Disaggregated LLM Inference*
  - Slot: Session 3-4 (2025/10/28)

Category 2. LLM Training Inference and RL Systems

Topic 7. Attention-Fully Connected Layer Disaggregated Inference.

- Group 7-1
  - Member: Songrun Xie
  - Paper: *MegaScale-Infer: Efficient Mixture-of-Experts Model Serving with Disaggregated Expert Parallelism*
  - Slot: Session 4-1 (2025/10/30)

Category 3. Algorithmic Advances for LLM

Topic 10. LLM Inference Acceleration Algorithms.

- Group 10-1
  - Member: Caieus Moreign and Chenyu Liu
  - Paper: *H2o: Heavy-hitter oracle for efficient generative inference of large language models*
  - Slot: Session 5-2 (2025/11/04)
- Group 10-2
  - Member: Haodong Wang and Qianli Liu
  - Paper: *LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding*
  - Slot: Session 5-3 (2025/11/04)
- Group 10-3
  - Member: Jiangnan Yu and Zhenxiao Cao
  - Paper: *Native sparse attention: Hardware-aligned and natively trainable sparse attention*
  - Slot: Additional Session (2025/11/27)

RELAXED
SYSTEM LAB

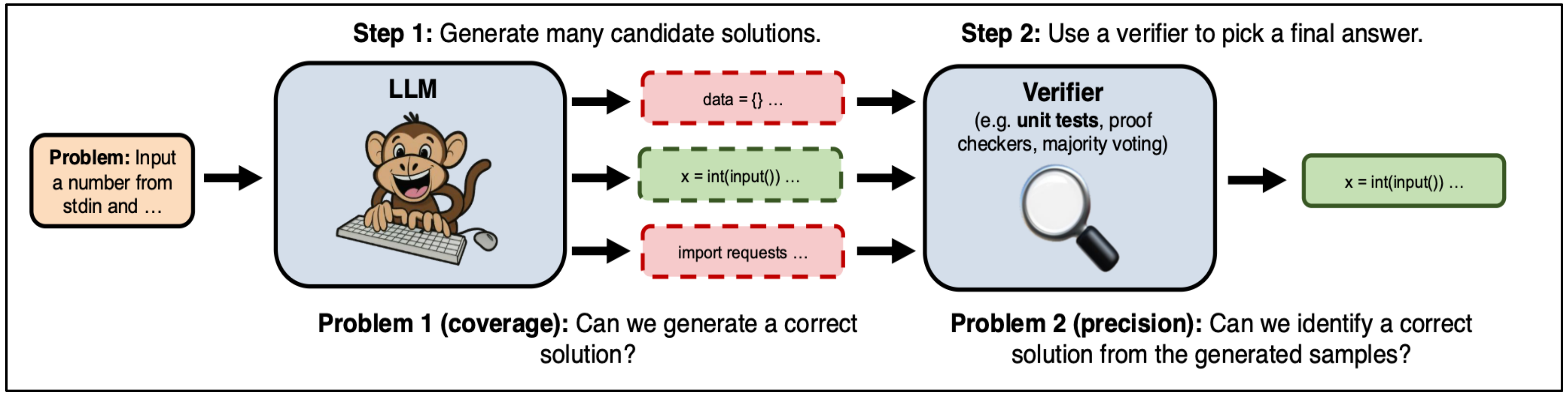
Improve the Generative Inference Quality

Prompt Engineering

- ***Prompt engineering*** is the practice of developing and optimizing prompts to efficiently use large language models (LLMs) for a variety of applications.
- Rules of Thumb from OpenAI.
- Advanced prompt engineering:
  - Baseline: zero-shot prompting;
  - Few-shot prompting;
  - Chain-of-thought prompting;
  - Self-consistency;
  - Tree of thoughts (ToT);
- Model safety issues:
  - Prompt injection;
  - Prompt leaking;
  - Jailbreaking.

Inference Time Scaling

- With repeated sampling scenarios, we consider:
  - ***Coverage***: As the number of samples increases, what fraction of problems can we solve using any sample that was generated?
  - ***Precision***: In settings where we must select a final answer from the collection of generated samples, can we identify the correct samples?


Step 1: Generate many candidate solutions.
Step 2: Use a verifier to pick a final answer.
Problem: Input a number from stdin and …
LLM
data = {} …
x = int(input()) …
import requests …
Verifier
(e.g. **unit tests**, proof checkers, majority voting)
x = int(input()) …
Problem 1 (coverage): Can we generate a correct solution?
Problem 2 (precision): Can we identify a correct solution from the generated samples?

- The log of coverage c as a function of the number of samples k:
  - $\log(c) = ak^{-b}$
  - $c = \exp(ak^{-b})$
- where $a, b \in \mathbb{R}$ are fitted model parameters.

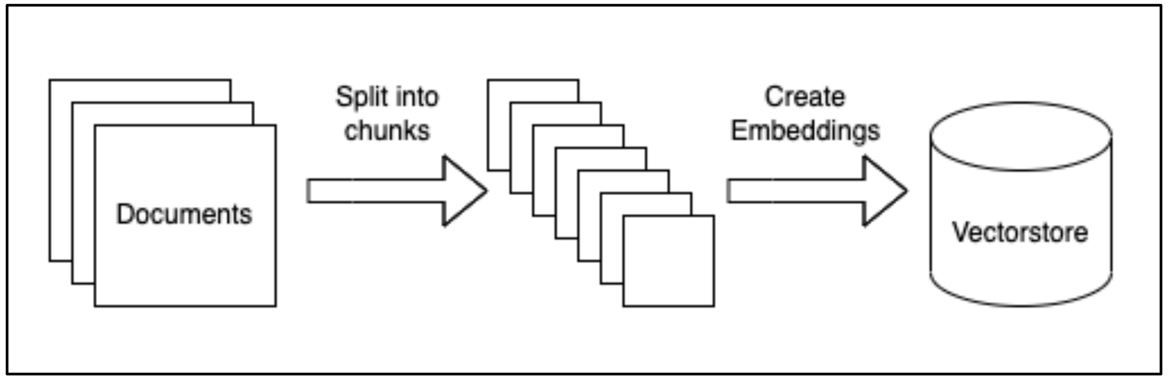
Inference Time Scaling

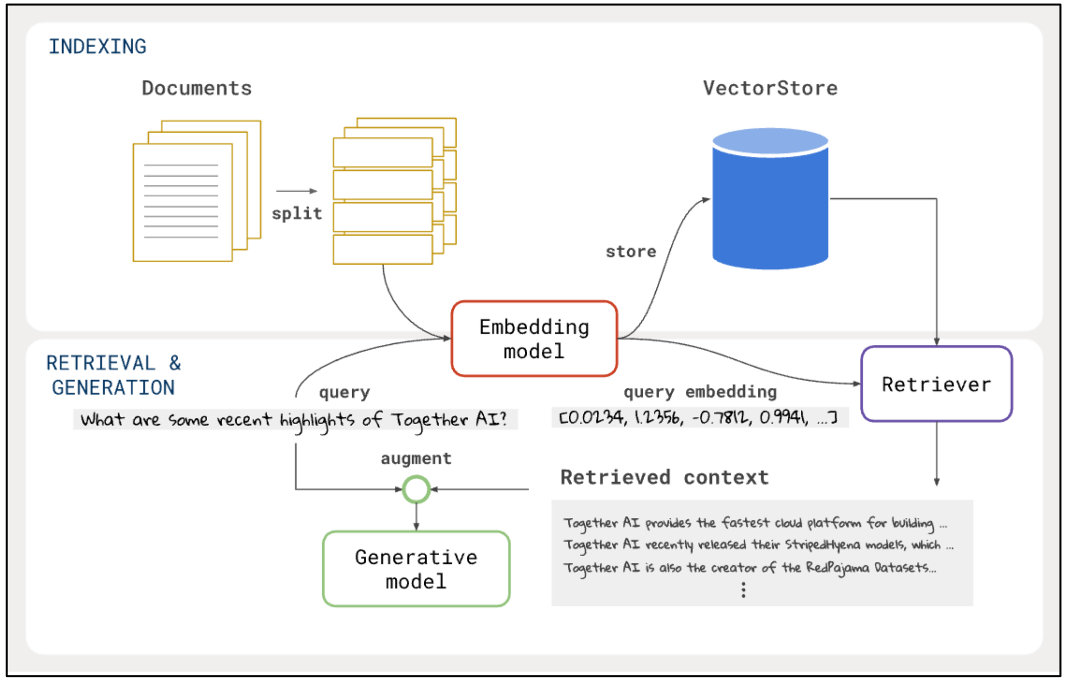
RELAXED SYSTEM LAB

- Inference scaling law indicates that we can scale the inference-time compute by sampling multiple branches in the solution space for performance boost.
- Consistency-based selection: a simple, effective, and general principle.
    - Self-consistency;
    - Universal self-consistency;
    - Train LLM ranker/verifier (ORM, PRM).
- When LLM self-evaluation works well: search in the partial solution space can help.

Retrieval-Augmented Generation

- **RAG Indexing:**
  - Load data sources to text.
  - Chunk text.
  - Embed text.
  - Load embeddings to vectorDB.
- **Retrieval and Generation:**
  - Generate the embedding of the input prompt.
  - Lookup relevant documents.
  - Augment the prompt.
  - Generate a response.




INDEXING
Documents
split
VectorStore
store
Embedding model
RETRIEVAL & GENERATION
query
What are some recent highlights of Together AI?
query embedding
[0.0234, 1.2356, -0.7812, 0.9941, ...]
Retriever
augment
Retrieved context
Together AI provides the fastest cloud platform for building ...
Together AI recently released their StripedHyena models, which ...
Together AI is also the creator of the RedPajama Datasets...
⋮
Generative model

Category 5. LLM Agent, Evaluation and Applications

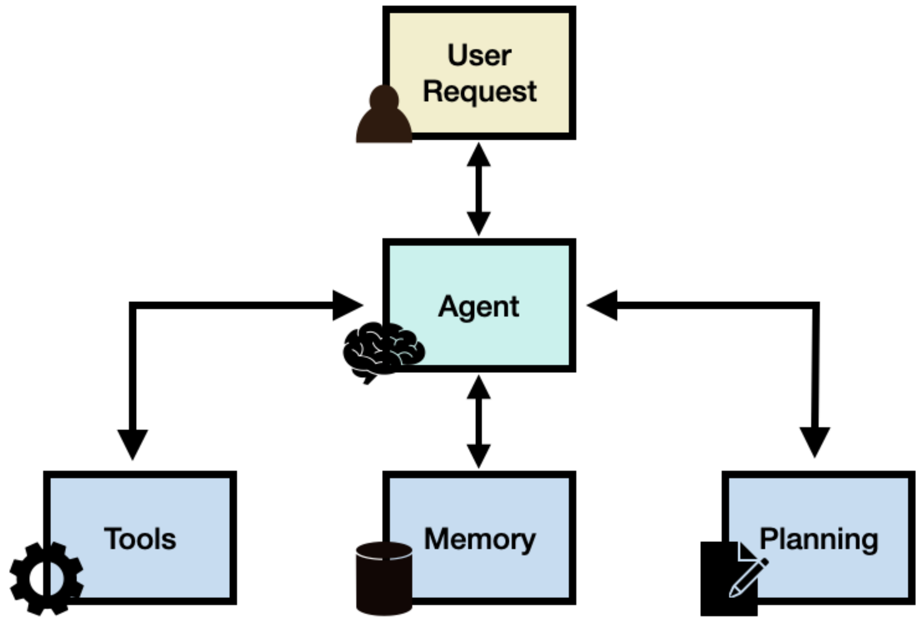
RELAXED SYSTEM LAB

Topic 18. Retrieval Augmented Generation.

- Group 18-1
  - Member: Bowen Liu and Siqi Wang
  - Paper: *Chameleon: A Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models*
  - Slot: Session 10-4 (2025/11/20)
- Group 18-2
  - Member: Chunyin Li and Junle Chen
  - Paper: *ReasonIR: Training Retrievers for Reasoning Tasks*
  - Slot: Session 11-1 (2025/11/25)
- Group 18-3
  - Member: Xiaoyu Han and Kwok Tsun On
  - Paper: *Search-r1: Training llms to reason and leverage search engines with reinforcement learning*
  - Slot: Session 11-2 (2025/11/25)

LLM Agent Framework

- LLM applications that can execute complex tasks:
  - Through the use of LLMs;
  - And other key modules like planning and memory.
- The architecture of an LLM Agent framework can be very flexible, we just provide one simple yet clear categorization, which includes:
  - **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 behaviours;
  - **Tool**: interacts with external environments.
- Examples: GUI Agent, Coding Agent.



Category 5. LLM Agent, Evaluation and Applications

Topic 15. LLM for Coding.

- Group 15-1
  - Member: Zhantong Xue and Le Xu
  - Paper: *Deepseek-coder-v2: Breaking the barrier of closed-source models in code intelligence*
  - Slot: Session 9-3 (2025/11/18)
- Group 15-2
  - Member: Danxuan Liang and Yu Kei Jian
  - Paper: *Qwen2. 5-xCoder: Multi-Agent Collaboration for Multilingual Code Instruction Tuning*
  - Slot: Session 9-4 (2025/11/18)

Category 5. LLM Agent, Evaluation and Applications

Topic 16. LLM for Math.

- Group 16-1
  - Member: Zhaochen Su and Junteng Liu
  - Paper: *Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition*
  - Slot: Session 10-1 (2025/11/20)

Category 5. LLM Agent, Evaluation and Applications

RELAXED SYSTEM LAB

Topic 17. GUI Agent.

- Group 17-1
    - Member: Yuxuan Cao and Junlong Li
    - Paper: *AgentTrek: Agent Trajectory Synthesis via Guiding Replay with Web Tutorials*
    - Slot: Session 10-2 (2025/11/20)
- Group 17-2
    - Member: Shijue Huang and Changxuan Fan
    - Paper: *Opencua: Open foundations for computer-use agents*
    - Slot: Session 10-3 (2025/11/20)

RELAXED
SYSTEM LAB

LLM Alignment after Pre-Training

Parameter Efficient Fine-Tuning

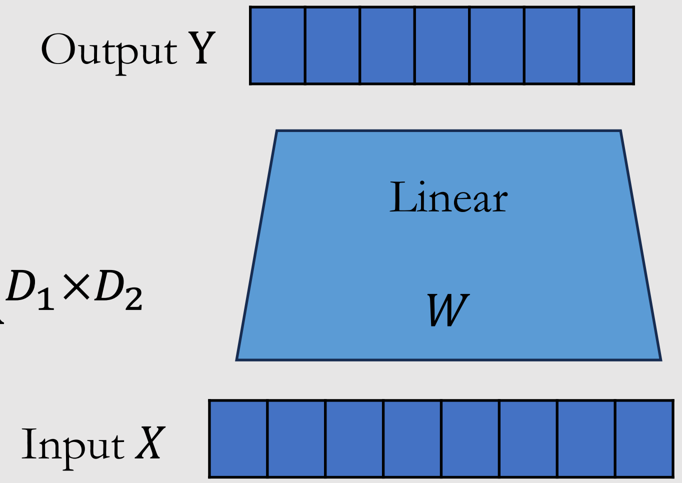
- *Parameter efficient fine-tuning (PEFT):* Rather than finetuning the entire model, we finetune only small amounts of weights.
    - Frozen layer/Subset fine-tuning: pick a subset of the parameters, fine-tune only those layers, and freeze the rest of the layers.
    - Adapters: add additional layers 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

- 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: $\Delta W = AB$.

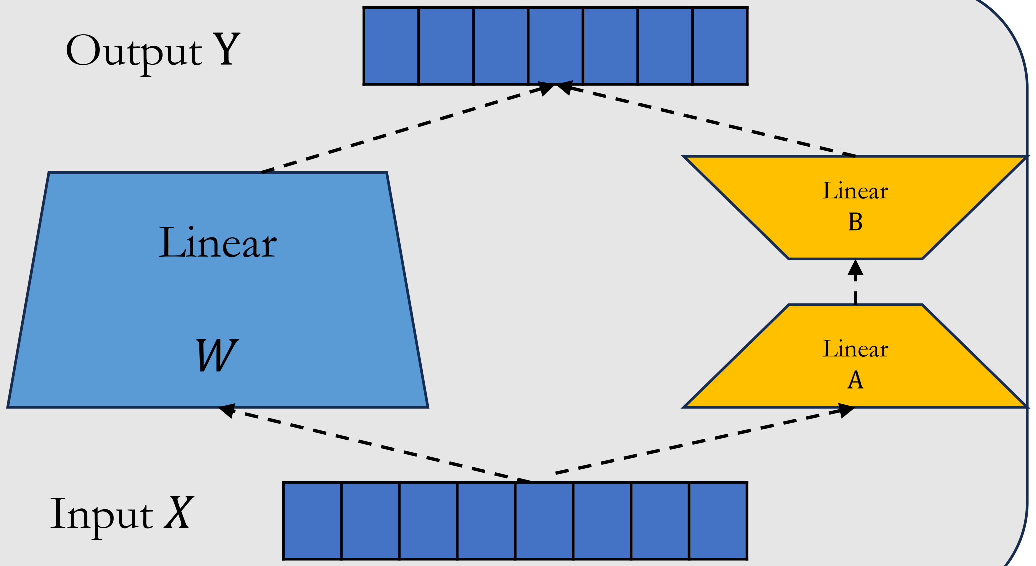
Standard Linear Layer

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



LoRA Linear Layer

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



Category 3. Algorithmic Advances for LLM

Topic 11. Efficient SFT Algorithms.

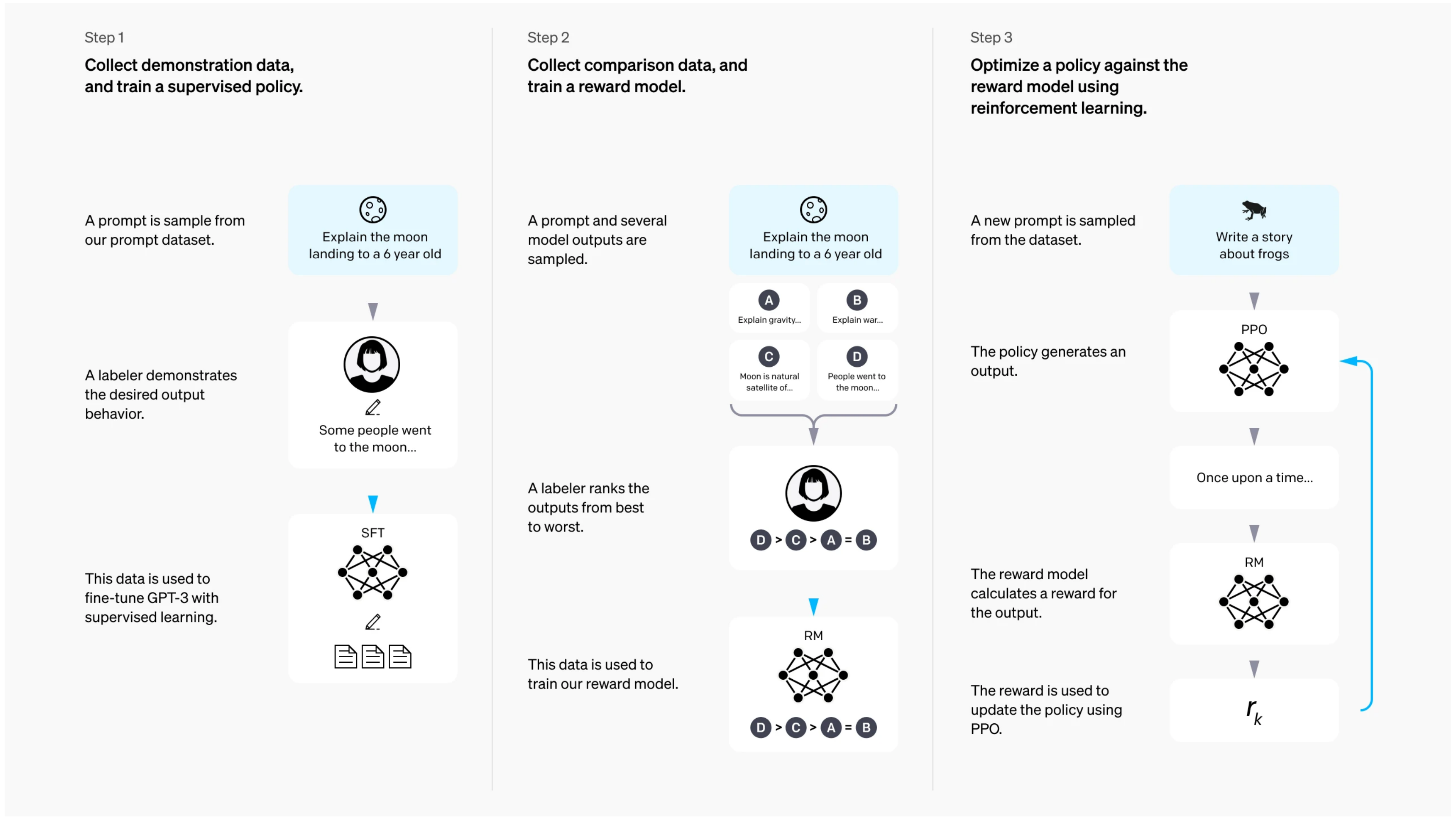
- Group 11-1
  - Member: Yejia Liu and Haoxian Liu
  - Paper: *GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection*
  - Slot: Session 7-2 (2025/11/11)
- Group 11-2
  - Member: Vinayak Khurana and Eman Ansar
  - Paper: *Flora: Low-Rank Adapters Are Secretly Gradient Compressors*
  - Slot: Session 6-2 (2025/11/06)

RL Alignment for LLM

- Under the context of LLM alignment by RL:
  - **State**: The state corresponds to the input prompt provided to the language model. In RL terms, this is the observation that informs the agent's next action.
  - **Action**: The action is the response generated by the LLM in reaction to the input prompt. Each token produced can be considered an individual action, making the entire response a sequence of actions. This sequential generation aligns with the token-by-token decision-making process in reinforcement learning.
  - **Policy**: The policy in this context is the LLM itself, which maps input prompts (states) to probability distributions over possible next tokens (actions).
  - **Reward**: The reward is a scalar value representing the quality of the generated response, which guides the policy updates, encouraging the model to produce outputs that align with human expectations.

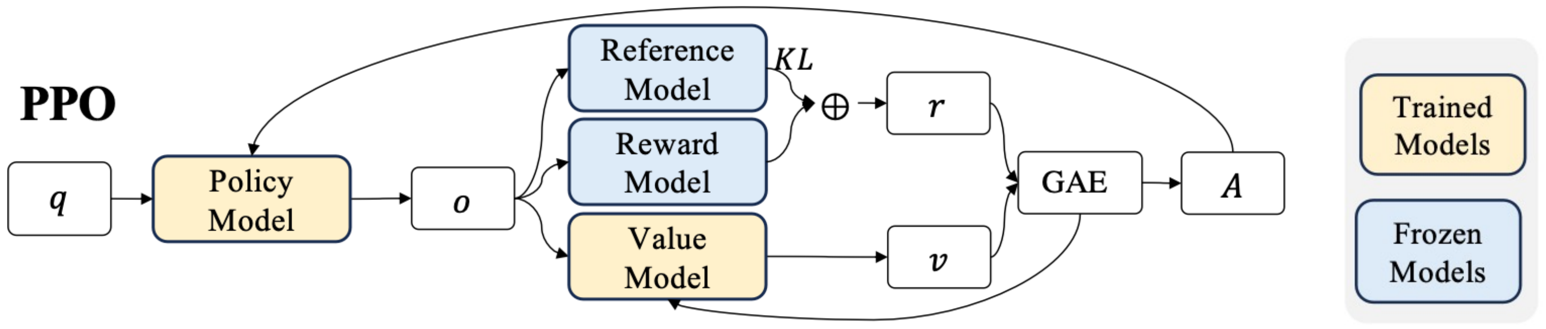
RLHF

RELAXED SYSTEM LAB



Step 1
Collect demonstration data, and train a supervised policy.
A prompt is sample from our prompt dataset.
Explain the moon landing to a 6 year old
A labeler demonstrates the desired output behavior.
Some people went to the moon...
This data is used to fine-tune GPT-3 with supervised learning.
SFT
Step 2
Collect comparison data, and train a reward model.
A prompt and several model outputs are sampled.
Explain the moon landing to a 6 year old
A Explain gravity...
B Explain war...
C Moon is natural satellite of...
D People went to the moon...
A labeler ranks the outputs from best to worst.
D > C > A = B
This data is used to train our reward model.
RM
D > C > A = B
Step 3
Optimize a policy against the reward model using reinforcement learning.
A new prompt is sampled from the dataset.
Write a story about frogs
The policy generates an output.
PPO
Once upon a time...
The reward model calculates a reward for the output.
RM
The reward is used to update the policy using PPO.
r_k

PPO



Category 2. LLM Training Inference and RL Systems

Topic 8. RL Systems.

- Group 8-1
  - Member: Yuguang Zhou and Jiazhi Mi
  - Paper: *Hybridflow: A flexible and efficient rlhf framework*
  - Slot: Session 4-2 (2025/10/30)
- Group 8-2
  - Member: Ding Pan and Jiayi Cheng
  - Paper: *AReaL: A Large-Scale Asynchronous Reinforcement Learning System for Language Reasoning*
  - Slot: Session 4-3 (2025/10/30)

Category 3. Algorithmic Advances for LLM

RELAXED SYSTEM LAB

Topic 12. RL Algorithms for LLM reasoning.

- Group 12-1
  - Member: Runze Zhang and Mingyang Zhao
  - Paper: *Proximal policy optimization algorithms*
  - Slot: Session 6-3 (2025/11/06)
- Group 12-2
  - Member: Haochen Shi and Baixuan Xu
  - Paper: *Deepseekmath: Pushing the limits of mathematical reasoning in open language models*
  - Slot: Session 6-4 (2025/11/06)
- Group 12-3
  - Member: Zibin Meng and Pengfei Wu
  - Paper: *Group sequence policy optimization*
  - Slot: Session 7-1 (2025/11/11)

RELAXED
SYSTEM LAB

LLM Evaluation

Quality Evaluation Format and Metric

- **Multiple-Choice Question (MCQ)**:
  - Evaluation metric: ***accuracy***, i.e. , the percentage of questions answered correctly.
- **Coding Problem**:
  - Evaluation metric: ***pass@k*** e.g., if at least one of k tries passes all tests.
- **Open-ended Question**:
  - Evaluation method 1- exact match: ***accuracy.***
  - Evaluation method 2 – partial match**:** ***precision, recall, F1 score.***
  - Evaluation method 3 - human evaluation: ***pairwise comparisons (Leaderboard ChatArena)***
  - Evaluation method 4 – LLM-as-a-Judge: ***pairwise comparison, single output scoring (w/wo reference-answer).***

Representative Benchmarks

- General purpose benchmark:
  - MMLU (MMLU-Pro, MMMLU), GPQA.
- Coding benchmark:
  - CodeContests, SWE-Bench.
- Math benchmark:
  - Math-500, AIME.

Category 5. LLM Agent, Evaluation and Applications

RELAXED
SYSTEM LAB

Topic 19. LLM Evaluation and Benchmarks.

- Group 19-1
  - Member: Ling Liang and Xinyu Geng
  - Paper: *The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models*
  - Slot: Session 11-3 (2025/11/25)
- Group 19-2
  - Member: Yuwei Wu and Longge Deng
  - Paper: *DeepScholar-Bench: A Live Benchmark and Automated Evaluation for Generative Research Synthesis*
  - Slot: Session 11-4 (2025/11/25)

Multi-Modal LM

Category 4. Multimodal Foundation Model

Topic 13. Multimodal Information Modeling and Reasoning.

- Group 13-1
  - Member: Pusen Gao and Yiyao Peng
  - Paper: *Qwen2. 5-vl technical report*
  - Slot: Session 6-1 (2025/11/06)
- Group 13-2
  - Member: Tse Wai Chung and Tuan An To
  - Paper: *Mmada: Multimodal large diffusion language models*
  - Slot: Session 7-3 (2025/11/11)
- Group 13-3
  - Member: Tianci Yin and Yu Foon Darin Chau
  - Paper: *RoboMonkey: Scaling Test-Time Sampling and Verification for Vision-Language-Action Models*
  - Slot: Session 7-4 (2025/11/11)
- Group 13-4
  - Member: Yubo Zhao and Dongjie Yang
  - Paper: *Qwen2.5-Omni Technical Report*
  - Slot: Session 8-1 (2025/11/13)

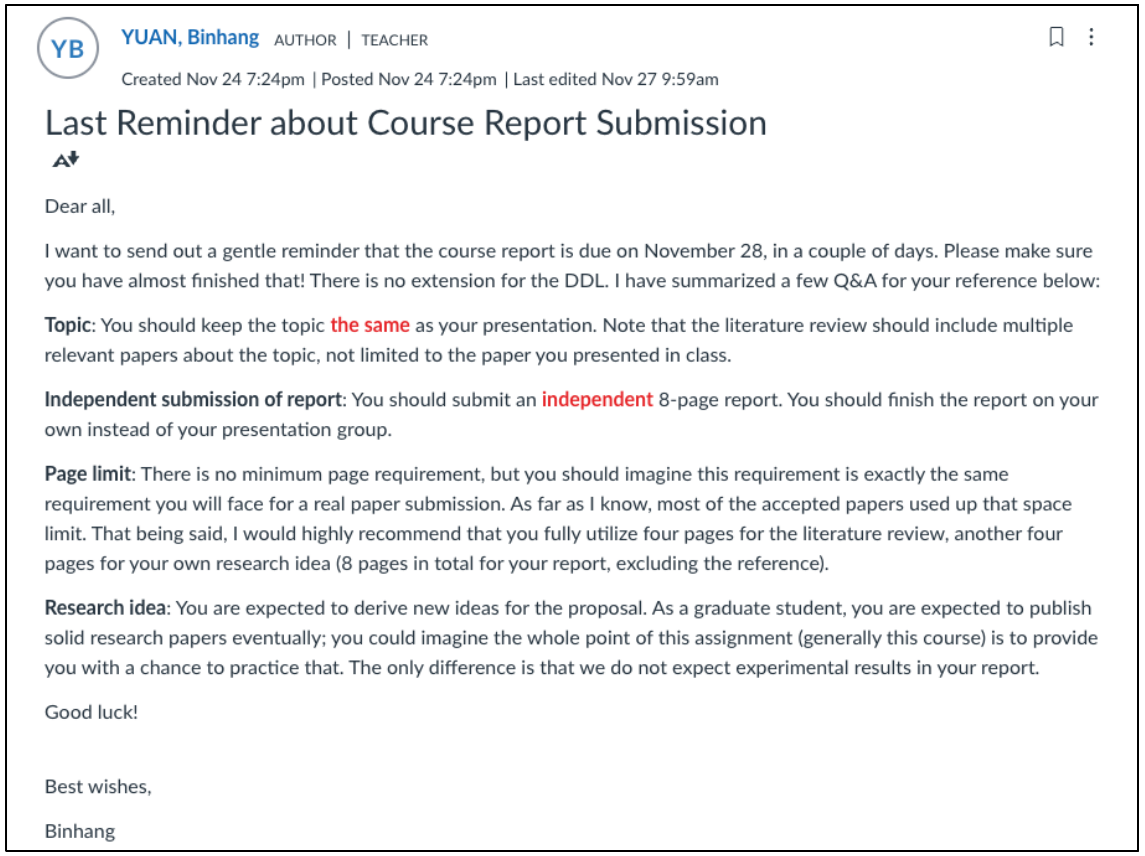
Category 4. Multimodal Foundation Model

Topic 14. Image- Video- Generation and Acceleration.

- Group 14-1
  - Member: Zhizhou Zhong and Zhenyuan Zhang
  - Paper: *CogVideoX: Text-to-Video Diffusion Models with An Expert Transformer*
  - Slot: Session 8-2 (2025/11/13)
- Group 14-2
  - Member: Gongye Liu and Zixuan Ye
  - Paper: *Seedance 1.0: Exploring the Boundaries of Video Generation Models*
  - Slot: Session 8-3 (2025/11/13)
- Group 14-3
  - Member: Meng Chu and Mingzhe Zheng
  - Paper: *Radial Attention: $ O (n$\backslash$log n) $ Sparse Attention with Energy Decay for Long Video Generation*
  - Slot: Session 8-4 (2025/11/13)
- Group 14-4
  - Member: Jianxin Huang and Chenran Huang
  - Paper: *Sparse VideoGen2: Accelerate Video Generation with Sparse Attention via Semantic-Aware Permutation*
  - Slot: Session 9-1 (2025/11/18)
- Group 14-5
  - Member: Hanlin Wang and Xuanhua He
  - Paper: *SpargeAttention: Accurate and Training-free Sparse Attention Accelerating Any Model Inference*
  - Slot: Session 9-2 (2025/11/18)

Last A Couple of Things

- Course report due: Tomorrow 23:59.
  - Do not miss that!
  - Do not worry, if you put a reasonable amount of effort into this, you will get a good grade.
  - If you want to further make a formal top-tier conference submission and need some help, my door would be open for you.
- Please submit the course feedback (SFQ).
  - It would be important for me to improve the course!



Created Nov 24 7:24pm | Posted Nov 24 7:24pm | Last edited Nov 27 9:59am

Last Reminder about Course Report Submission

Dear all,

I want to send out a gentle reminder that the course report is due on November 28, in a couple of days. Please make sure you have almost finished that! There is no extension for the DDL. I have summarized a few Q&A for your reference below:

Topic: You should keep the topic the same as your presentation. Note that the literature review should include multiple relevant papers about the topic, not limited to the paper you presented in class.

Independent submission of report: You should submit an independent 8-page report. You should finish the report on your own instead of your presentation group.

Page limit: There is no minimum page requirement, but you should imagine this requirement is exactly the same requirement you will face for a real paper submission. As far as I know, most of the accepted papers used up that space limit. That being said, I would highly recommend that you fully utilize four pages for the literature review, another four pages for your own research idea (8 pages in total for your report, excluding the reference).

Research idea: You are expected to derive new ideas for the proposal. As a graduate student, you are expected to publish solid research papers eventually; you could imagine the whole point of this assignment (generally this course) is to provide you with a chance to practice that. The only difference is that we do not expect experimental results in your report.

Good luck!

Best wishes,

Binhang

RELAXED
SYSTEM LAB

Thank you!