

📌 重要概念

1. 模型
 - 模型训练与推理
 - 模型部署
2. **SFT** (Supervised Finetune)
 - 模型训练与推理
 - 模型部署
 - 模型训练与推理 LoRA Adapter 模型训练与推理 SFT 模型
3. **RLHF** (Reinforcement Learning)
 - 模型训练与推理
 - 模型部署
4. 模型
 - 模型训练与推理

SFT (Supervised Finetune)


模型训练与推理

模型训练与推理 dirty work 模型训练与推理

模型训练与推理

- **Few-Shot Prompting** 模型训练与推理 1-5 模型训练与推理
- **Seed Prompt** 模型训练与推理 task_type 模型训练与推理
- 模型
 - 模型训练与推理
 - answer 模型训练与推理
 - task_type 模型训练与推理 sft 模型训练与推理
- 模型
 - 模型 prompt
 - 模型 task_type 模型训练与推理 seed prompt 模型训练与推理 seed 模型训练与推理 pretrain 模型训练与推理
 - 模型 seed 模型训练与推理 prompt
 - 模型 answer
 - 模型 GPT4/Claude3
 - 模型 Qwen_72B/deepseek_MoE

模型训练与推理

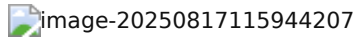
 image-20250817002125325

RL (Reinforcement Learning)

模型训练与推理

模型训练与推理

강의 목표



- **Agent** 에이전트
- **action** 행동
- **Environment** 환경 reward
- **reward** 보상
- **State** 상태

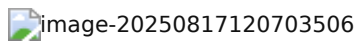
강의 목표 **MDP**

강의 목표

강의 목표

강의 목표 MDP

1. M^2 문제
2. 문제
3. M^2 문제



강의 목표

- S_t 상태

$$\begin{bmatrix} S_1 & S_1 & S_1 & S_2 & S_1 & S_2 & S_1 & S_2 & S_2 & S_3 \\ S_3 & S_1 & S_3 & S_2 & S_3 & S_3 \end{bmatrix}$$

- S_t 상태 S_t 상태 S_{t+1} 상태
- $p(S_{t+1}|S_t)$

$$p(S_{t+1}|S_t) = p(S_{t+1}|S_1, \dots, S_t)$$

- $P_{ss'}$
- $P_{ss'} = p(S_{t+1}=s'|S_t=s)$

강의 목표 **MRP**

강의 목표 $\langle S, P, R, \gamma \rangle$

- S 상태
- P 전이 확률
- R 보상 $R_S = E[R_{t+1}|S_t=s]$
 - 상태
 - s 상태 s' 상태
 - s 상태 s' 상태

- $\gamma \in [0,1]$

- Return G_t

-
-

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- $V(s)$

-
-

$$V(s) = E[G_t | S_t = s]$$

MRPs

$$V(s) = E[G_t | S_t = s] \quad G_t = R_{t+1} + \underbrace{\gamma R_{t+2} + \gamma^2 R_{t+3} + \dots}_{\gamma V(S_{t+1})} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$V(s) = E[R_{t+1} + \gamma V(S_{t+1}) | S_t = s]$$

MDP

$$\langle S, A, P, R, \gamma \rangle$$

- S
- A
- P $P_{ss'}^a = p[S_{t+1} = s' | S_t = s, A_t = a]$
- R $R_s^a = E[R_{t+1} | S_t = s, A_t = a]$
 -
 -
 -
- $\gamma \in [0,1]$

$$\pi(a|s) = P[A_t = a | S_t = s]$$

-
-
- MDP

$$M = \langle S, A, P, R, \gamma \rangle \quad \pi$$

-

$$P_{s,s'}^a = \sum_{a \in A} \pi(a|s) P_{s,s'}^a \quad R_s^a = \sum_{a \in A} \pi(a|s) R_s^a$$

-
-

□□□□s□□□□□\$**\pi**\$□□□□□

 $V_{\{\pi\}}(s) = E_{\{\pi\}}[G_t | S_t = s]$ $\square\square\square\square$ $G_t =$

$$\underbrace{R_{t+1}}_{\text{}} + \underbrace{\gamma R_{t+2} + \gamma^2 R_{t+3} + \dots}_{\text{}} = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad \text{}$$

$$V_{\pi}(s) = E_{\pi}[R_{t+1} + \gamma V_{\pi}(S_{t+1}) | S_t = s] \quad \text{}$$

5/5

□□□□s□□□□a□□□□□\$\\pi\$□□□□

 image-20250817175237288 \$\$ q_{\{\pi\}}(s,a) = E_{\{\pi\}}[G_t|S_t = s, A_t = a] \$\$ □□□□□□□□□□□□□□□□

$$q_{\pi}(s,a) = E_{\pi}[R_{t+1} + \gamma q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

□□□□□□□□□□□□□□□□

$$V_{\pi}(s) = \sum_{a \in A} \pi(a|s) q_{\pi}(s,a)$$

 image-20250817181356989

$$q_{\pi}(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_{\pi}(s')$$

 image-20250817181417147

□□□□□□

$$V_{\pi}(s) = \max_{\pi} V_{\pi}(s) \quad q_{\pi}(s,a) = \max_{\pi} q_{\pi}(s,a)$$

- `XXXXXXXXXX$pi_*\ge any\pi$`
- `XXXXXXXXXXXXXXXXXXXX`
- `XXXXXXXXXXXXXXXXXXXX`
- `XXXXXXXXXXXXXXXXXXXX`

$$\pi^* \geq \text{any } \pi$$
$$V_{\pi'}(s) \geq V_{\pi}(s) \quad \pi' > \pi$$

□□□□□□□□□□□□□□□□

1111111111111111

$$V_{\{\pi\}}(s) = \sum_{a \in A} \pi(a|s) q_{\{\pi\}}(s,a) \quad q_{\{\pi\}}(s,a) = R_s^{a+\gamma} \sum_{s' \in S} P_{ss'}^{aV_{\{\pi\}}(s')} \\ V_{\{\pi\}}(s) = \sum_{a \in A} \pi(a|s) (R_s^{a+\gamma} \sum_{s' \in S} P_{ss'}^{aV_{\{\pi\}}(s')})$$
$$V_{\{\pi\}}(s) = \sum_{\{a \in A\}} \pi(a|s) q_{\{\pi\}}(s, a)$$

□□□□\$\pi_*\$□□□□□□□□s□□□□□□\$q_*(s,a)\$

- $\pi_i^* \leq V_{\pi_i^*}(s) \leq V_{\pi_i^*}(s)$
- $q_{\pi_i^*}(s, a) \leq q_{\pi_i^*}(s, a)$
- $V_{\pi_i^*}(s) \leq q_{\pi_i^*}(s, a)$

$$q_{\pi}(a \mid s) = \begin{cases} 1 & \text{if } a = \arg \max_a q_{\pi}(s, a), \\ 0 & \text{otherwise.} \end{cases}$$

$$V_*(s) = \max_a q_*(s, a)$$

$$q_*(s, a) = R_s + \gamma \sum_{s'} P_{s, s'} a V_*(s')$$

$$V_*(s) = \max_a (R_s + \gamma \sum_{s'} P_{s, s'} a V_*(s'))$$
$$q_{\pi}(s, a) = R_s + \gamma \sum_{s' \in S} P_{ss'} a V_{\pi}(s')$$

$$V_{\pi}(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V_{\pi}(s')$$

$$q_{\pi}(s, a) = R_s + \gamma \sum_{s' \in S} P_{ss'} a \max_{a'} q_{\pi}(s', a')$$

LLaMA

GPT LLaMA Transformer Decoder-only

-
-
-

RMSNorm

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2 + \epsilon}} \cdot \gamma$$

- x
- d
- ϵ
- γ

SwiGLU

Swish

$$\text{Swish}(x) = x \cdot \sigma(x)$$

- σ Sigmoid

$$\text{GLU}(x) = \sigma(W_1 x + b_1) \odot (W_2 x + b_2)$$

- \odot
- W_1, W_2, b_1, b_2

$$\text{SwiGLU}(x) = \text{Swish}(\text{Linear}_1(x)) \odot \text{Linear}_2(x)$$

- Swish
- GLU

RoPE

GQA

LLaMA2 3

GPT

Deepseek

MLA

- GPU
- GPU

- DP Data Parallel
- DDP Distributed Data Parallel
- FSDP Fully Sharded Data Parallel

All-Reduce

- GPU
- GPU
 - GPU SUM MAX
 - GPU

All-Gather

- GPU GPU
- GPU
 - GPU
 - GPU

DP Data Parallel

Python GIL CPU

image-20250814165126760

1. CPU端を主としてGPU端
2. GPU端を主としてGPU0端
3. GPU0端を主としてGPU端
4. GPU端を主としてGPU端

GPU

GPU端を主としてGPU端

- GPU0端を主としてGPU端
- GPU端を主としてGPU端

GPU

- GPU端を主としてGPU端
- GPU0端を主としてGPU端

DDP (Distributed Data Parallel)

GPU

- GPU端を主としてGPU端
- GPU0端を主としてGPU端

GPU

- GPU0端を主としてGPU端
- GPU端を主としてGPU端
- GPU端を主としてGPU端
- GPU端を主としてGPU端
- GPU端を主としてGPU端
- GPU端を主としてGPU端

GPU

GPU端を主としてGPU端

GPU端を主としてGPU端

- Scatter-Reduce (GPU端を主としてGPU端)
- All-Gather (GPU端を主としてGPU端)

GPU端を主としてGPU端

Ring-AllReduce

- GPU端を主としてGPU端
- GPU端を主としてGPU端
- GPU端を主としてGPU端

GPU

Scatter-Reduce

- GPU端を主としてGPU端
- GPU端を主としてGPU端

All-gather

- GPU端を主としてGPU端
- GPU端を主としてGPU端

- ψ_i GPU j $(j-i-1) \bmod n$ $(j-i-2) \bmod n$

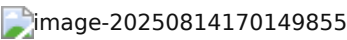


FSDP (Fully Sharded Data Parallel)

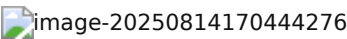
- GPU
- GPU
- GPU CPU

DeepSpeed ZeRO-1

3 GPU GPU



GPU ZeRO-1 GPU



-
- GPU0 GPU1 GPU2 GPU2 GPU
- GPU FP32 FP16
- FP16 GPU

ψ N

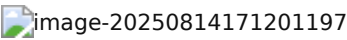
GPU

- $(N-1) \frac{\psi}{N} \approx \psi$
- $(N-1) \frac{\psi}{N} \approx \psi$

2ψ

DeepSpeed ZeRO-2

ZeRO-2 FP16 GPU



-
- GPU0 GPU1 GPU2 GPU
- GPU FP32 FP16
- FP16 GPU

$\frac{\psi}{N}$

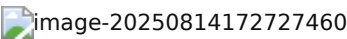
GPU

- $\frac{\psi}{N-1} \approx \psi$
- $\frac{\psi}{N-1} \approx 2\psi$

2ψ

DeepSpeed ZeRO-3

ZeRO-3 **FP16**



- GPU **GPU**
-
- GPU GPU
-

$\frac{\psi}{N}$

GPU

- $\frac{\psi}{N-1} \approx \psi$
- $\frac{\psi}{2(N-1)} \approx 2\psi$

3ψ

TP

GPU

$Y = XW$

- $X \in \mathbb{R}^{2 \times 2}, W \in \mathbb{R}^{2 \times 2}$

$$\begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix}$$

$$W = \begin{bmatrix} W_0 & W_1 \\ W_2 & W_3 \end{bmatrix}$$

$$Y_0 = XW_0 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix}$$

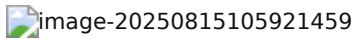
$$Y_1 = XW_1 = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ w_2 & w_3 & w_4 & w_5 \end{bmatrix}$$

$$Y = \begin{bmatrix} Y_0 & Y_1 \end{bmatrix}$$

□□□□□□□□

$$Y = XW$$

- $X \in \mathbb{R}^{2 \times 2} \square W \in \mathbb{R}^{2 \times 2}$

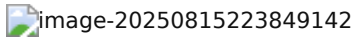
$$\begin{aligned} \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \end{pmatrix} &= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} w_1 & w_2 & w_3 & w_4 \end{pmatrix} \\ W &= W_0 W_1 \\ X &= \begin{pmatrix} X_1 & X_2 \end{pmatrix} \quad Y_1 = \begin{pmatrix} x_1 & x_3 \end{pmatrix} \begin{pmatrix} w_1 & w_2 \end{pmatrix} = \\ &= \begin{pmatrix} y_{11} & y_{12} & y_{31} & y_{32} \end{pmatrix} \quad Y_2 = \begin{pmatrix} x_2 & x_4 \end{pmatrix} \begin{pmatrix} w_3 & w_4 \end{pmatrix} = \begin{pmatrix} y_{23} & y_{24} & y_{43} & y_{44} \end{pmatrix} \\ Y &= Y_1 + Y_2 \end{aligned}$$


□□□□□**PP**□

□□□□□□□□□□□□□□□□□□□□**GPU**□

□□□□□□

- Stage Stage GPU
-



1111

- GPU2 is not available
- GPU2 is not available
- GPU1 is not available

5/5

- GPU GPU GPU GPU GPU GPU GPU GPU GPU GPU
- CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU

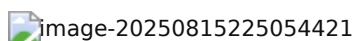
□□□□□□

F-then-B

- `tf.nn.conv2d`
- `tf.nn.conv2d` mini-batch `tf.nn.conv2d` mini-batch

1F1B

- 00000000000000000000000000000000
- 0000000000 stage4 0 F42 stage4 00 2 0 micro-batch 0000000000 F42 000000 F41 0000 B41 stage4 00 1 0 micro-batch 000000000000000000000000 F41 00000000 **F42** 0000 **F41** 00000000



- | | | | | | | | |
|--|--|--|--|--|--|--|--|
| | | | | | | | |
|--|--|--|--|--|--|--|--|

Gpipe

□□□□□□□□□□□□□□□□□□□□**F-then-B**□□



111

111

111

Pip

111

Gpi

- □

Pipe

- ☐



111

- □

111

01F

- **V**

- □

1. □

- **L**

- □

1. M

2. **Stage** Micro-batch W_1
3. W_1

1111

```
1#####device#####stage#####
```

000

- Device 1 0000500000000000100000000010000000
- Device 1 00005000000000004000000000102034000000



Weight Stashing

```

device=
minibatch=

```

111

- Device 1 00050000000000000000000000000000
- Device 1 00050000000000000000000000000000

```
2 device stage
```

111

- **Device 1** 0000500000000000100000000010000000
- **Device 2** 000050000000000020000000001020000000



Vertical Sync

pipeline device device

111

- Device 1
- Device 2

□ □ □ □ □ □

```

float16 float32 Master-Weight

```

0000

1. Master-Weight fp16
2. inputs FP16
3. fp32



Loss Scaling

1. fp16
2. fp32
3. fp16fp32
4. fp32

fp16FP16

image-20250813121114207

RNNfp16fp32

image-20250813115950009

image-20250813115500851

- 1.
- 2.

image-20250813115927153

MHAMQAGQA

image-20250816120438735

MHA

MQAMulti-Query Attention

- TransformerkeyvalueKVQ
- KV
- KV

- 计算复杂度为 $O(N^2)$
- 计算复杂度为 $O(N^2)$

GQA Grouped-Query Attention

计算复杂度

计算复杂度

- Query 分组为 N 组，Key/Value 不分组
- 计算复杂度为 $O(N \times N) = O(N^2)$ MHA

计算复杂度

- 计算复杂度为 $O(N \times N) = O(N^2)$ MHA

MLA Multi-Head Latent Attention

Deepseek-V3

计算复杂度

- KV Cache 计算复杂度为 $O(N \times N)$
- 计算复杂度为 $O(N \times N) = O(N^2)$ MHA

计算复杂度

- 计算复杂度为 $O(N \times N) = O(N^2)$ KV Cache
- 计算复杂度为 $O(N \times N) = O(N^2)$ MHA

计算复杂度

- Key/Value 计算复杂度为 $O(N \times N) = O(N^2)$ KV Cache
- 计算复杂度为 $O(N \times N) = O(N^2)$ MHA

image-20250816120905940

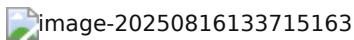
计算复杂度

Query Key 计算复杂度为 $O(N \times N) = O(N^2)$ $\begin{matrix} q_t^R & q_t^C \\ k_t^R & k_t^C \end{matrix}$ $\begin{matrix} q_t^R & q_t^C \\ k_t^R & k_t^C \end{matrix}$ RoPE (q_t^R, k_t^R)

1. 计算复杂度为 $O(N \times N)$
2. 计算复杂度为 $O(N \times N)$
 - $O(N \times N)$ KV Cache
 - $O(N \times N)$ Q
3. Key/Value 计算复杂度
 - Key Value $k_{t,i}^C, v_{t,i}^C$
 - $k_{t,i}^R$ 计算复杂度
4. Query 计算复杂度为 $O(N \times N)$ $q_{t,i}^R, q_{t,i}^C$
5. QKV 计算复杂度

image-20250816131927886

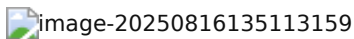
KV 计算复杂度



计算 $c_t^{KV} k_t^C v_t^C$ 的公式为 $c_t^{KV} = W^{DKV} h_t \cdot k_t^C = W^{UK} c_t^{KV} \cdot v_t^C = W^{UV} c_t^{KV}$ 的公式为 $c_t^{KV} = W^{UV} c_t^{KV}$

计算 c_t^{KV} 的公式为 $c_t^{KV} = W^{UV} c_t^{KV}$

计算 Q 的公式为 $Q = W^{DQ} h_t$



计算 $c_t^Q = W^{DQ} h_t \cdot q_t^C = W^{UQ} c_t^Q$ 的公式为 $c_t^Q = W^{UQ} c_t^Q$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 k_t^R 的公式为 $k_t^R = W^{DK} h_t$

- 计算 Q 的公式为 $Q = W^{DQ} h_t$
- 计算 c_t^{KV} 的公式为 $c_t^{KV} = W^{UV} c_t^{KV}$

计算 $K_{rot} = R_n(W_{QK} \cdot c_t^{KV})$

- 计算 W_{QK} 的公式为 $W_{QK} = W_{QK}$

计算 $S = (W^{UQ} c_t^Q)^T (W^{DK} h_t) = (W^{UQ} c_t^Q)^T (W^{DK} h_t)$

计算 Q 的公式为 $Q = W^{DQ} h_t$

- 计算 k_t^C 的公式为 $k_t^C = W^{DK} h_t$
- 计算 Q 的公式为 $Q = W^{DQ} h_t$
- 计算 K 的公式为 $K = [k_t^C; k_t^R]$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 Q 的公式为 $Q = W^{DQ} h_t$

计算 Q 的公式为 $Q = W^{DQ} h_t$


计算 Q 的公式为 $Q = W^{DQ} h_t$

📄

- GPU vs CPU 推理性能对比
- 推理/训练 GPU 显存需求对比


📄

- 推理性能对比 (假设 1024 token 输入, 4 token 输出)
 - 推理速度: 1024 token 输入, 1024 token 输出, 4 token 输出
- 显存需求对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出)
 - 显存需求: 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出
- 推理性能对比 (假设 500MB 显存, 1GB 显存, 300MB 显存, 2GB 显存)
 - 推理速度: 500MB 显存, 1GB 显存, 300MB 显存, 2GB 显存


 image-20250819121745375

📄

- 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出)

 image-20250819121817645

- 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出)
- 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出) Paged Attention 推理性能对比 KV Cache


 image-20250819121841203

- 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出)
- KV Cache 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出) Paged Attention 推理性能对比 KV Cache
 - 推理速度: 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出
- Copy-on-Write 推理性能对比 (假设 1000 token 输入, 5 token 输出, 10 token 输出, 995 token 输出) 3 token 5 token 推理性能对比

Flash Attention

推理性能对比 IO 推理性能对比 SRAM 推理性能对比 IO

推理性能对比 GPU 推理性能对比 SRAM 推理性能对比 HBM 推理性能对比 Flash attention 推理性能对比 Attention 推理性能对比 HBM

 image-20250820134630626

📄

- 推理性能对比 HBM 推理性能对比 SRAM
- 推理性能对比 GPU 推理性能对比 SRAM
- 推理性能对比 SRAM 推理性能对比 Pytorch 推理性能对比 SRAM

pytorch attention

- HBM QK SRAM
- $S = QK^T$
- S HBM
- HBM S HRAM
- $P = \text{softmax}(S)$
- P HBM
- HBM P SRAM
- $O = PV$
- O HBM
- O

- **Compute-Bound**
 -
 - GPU
- **Memory-Bound**
 - ReLU, Softmax, Sum, Dropout
 - GPU

Safe-Softmax

softmax $\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$ x_i

Safe Softmax $\text{softmax}(x_i) = \frac{e^{x_i - \max(x)}}{\sum_{j=1}^n e^{x_j - \max(x)}}$

- $\max(x)$
- $e^{\max(x)}$

- **Tiling Algorithm**
 - SRAM
 - SRAM Flash Attention SRAM
- **Recomputation**
 - GPU HBM
 - $S = QK^T$ Flash Attention HBM SRAM
 - $P = \text{softmax}(S)$ HBM
- **Kernal Fusion**
 - HBM
 - Kernal

SRAM

- QK V SRAM
- $S = QK^T$
- $P = \text{softmax}(S)$
 - softmax softmax

- o **Safe-Softmax**
- o **Safe-Softmax**
 - $x_1 = [1, 2]$, $x_2 = [3, 4]$
 - $m(x_1) = 2 = m(x)$
 - $f(x_1) = [e^{1-m(x_1)}, e^{2-m(x_1)}] = [e^{-1}, e^0]$
 - $l(x_1) = e^{-1} + e^0$
 - $m(x) = \max(m(x_1), m(x_2)) = 4$
 - $f(x_2) = [e^{3-m(x_2)}, e^{4-m(x_2)}] = [e^{-1}, e^0]$
 - $l(x_2) = e^{-1} + e^0$
 - $f(x) \leq l(x)$

$$\begin{aligned} f(x) &= [e^{m(x_1)-m(x)}f(x_1), e^{m(x_2)-m(x)}f(x_2)] \\ f(x) &= [e^{-2}(e^{-1}, e^0), e^0(e^{-1}, e^0)] \\ l(x) &= e^{m(x_1)-m(x)} * l(x_1) + e^{m(x_2)-m(x)} * l(x_2) \\ l(x) &= e^{-3} + e^{-2} + e^{-1} + e^0 \end{aligned}$$

- $EO = PV$
- $SRAM \ll EO \ll HBM$

GPU

[illegible]

GPU Warp 32

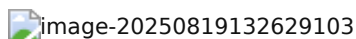
SM GPU

- **CUDA Cores** CUDA 코어 코어
- **Tensor Cores** 텐서 코어 코어
- **Warp Scheduler** Warp 스케줄러 Warp
- **Shared Memory** 공유 메모리
- **L1 Cache** L1 캐시
- **Load/Store Units** 로드/스토어 유닛 HBM 공유 메모리
- **Register File** 레지스터 파일
- **SFU** 스칼라 기능 유닛

GPU

- **SRAM** L1 Cache **SM** L2 Cache **SM** Shared Memory
- **HBM**

1111



- NVLink □**GPU**□□□□□□□□□□
- PCIe □**CPU-GPU**□□□□

GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM

GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM

1. GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM
2. GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM
 - GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM
3. GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM
4. GPU 的 HBM 是通過 PCIe 與 CPU 共享的 GPU HBM