清華大學
Tsinghua University

趋境科技
APPROACHING.AI

LLM Serving on Heterogeneous Hardware
KTransformers Part

https://github.com/kvcache-ai

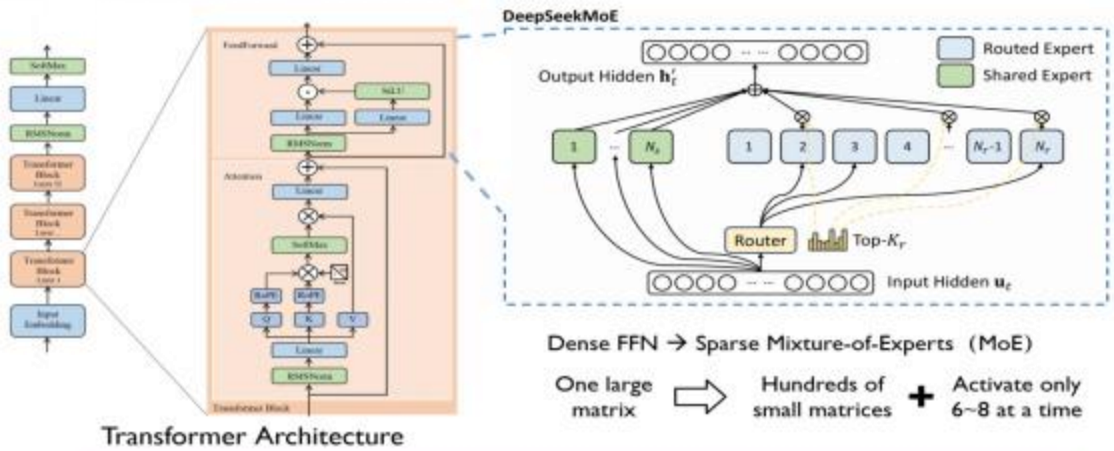
KTransformers

趋境科技
APPROACHING.AI

Content

- Motivation for Heterogeneous LLM Serving
- Core Technologies of KTransformers
- Tutorial: Fine-Tune and Chat with Your Customized Model

Attention + MoE



GPU + CPU

Background and Observation of LLM and Sparse Mixture-of-Experts (MoE)

I Motivation for Heterogeneous LLM Serving

趋境科技 APPROACHING.AI

Background: Large Language Models (LLMs)

Large Language Models (LLMs) are widely applied in industry and researched in academia.

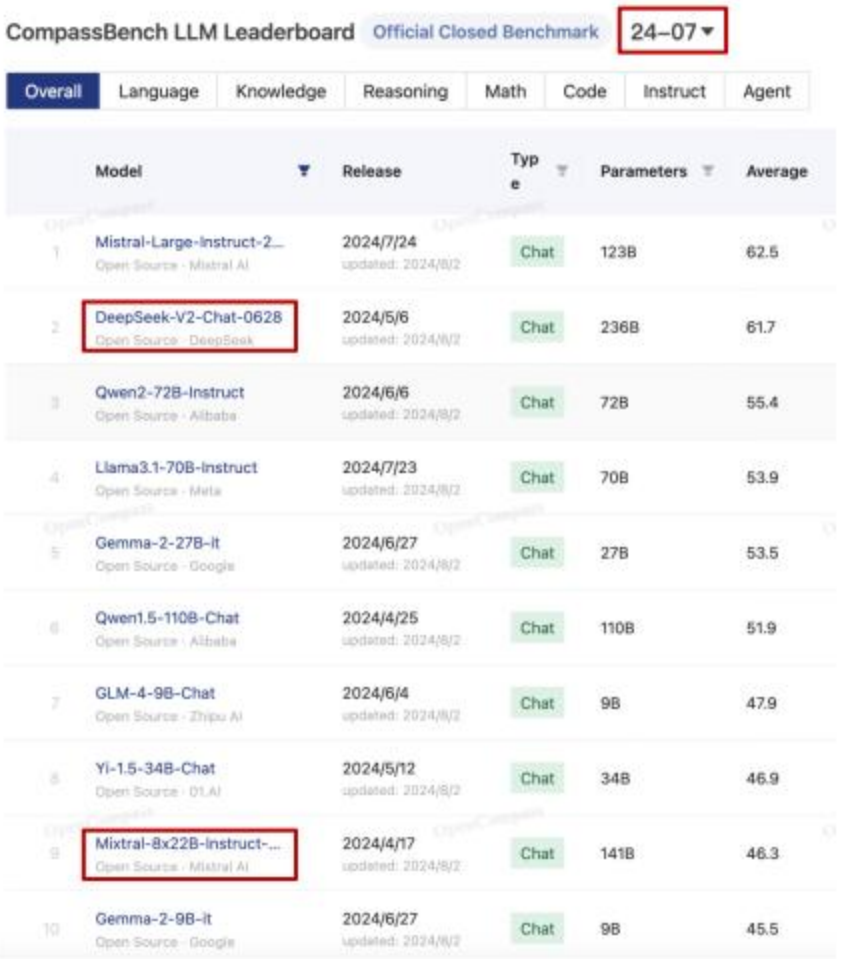
Knowledge Q&A

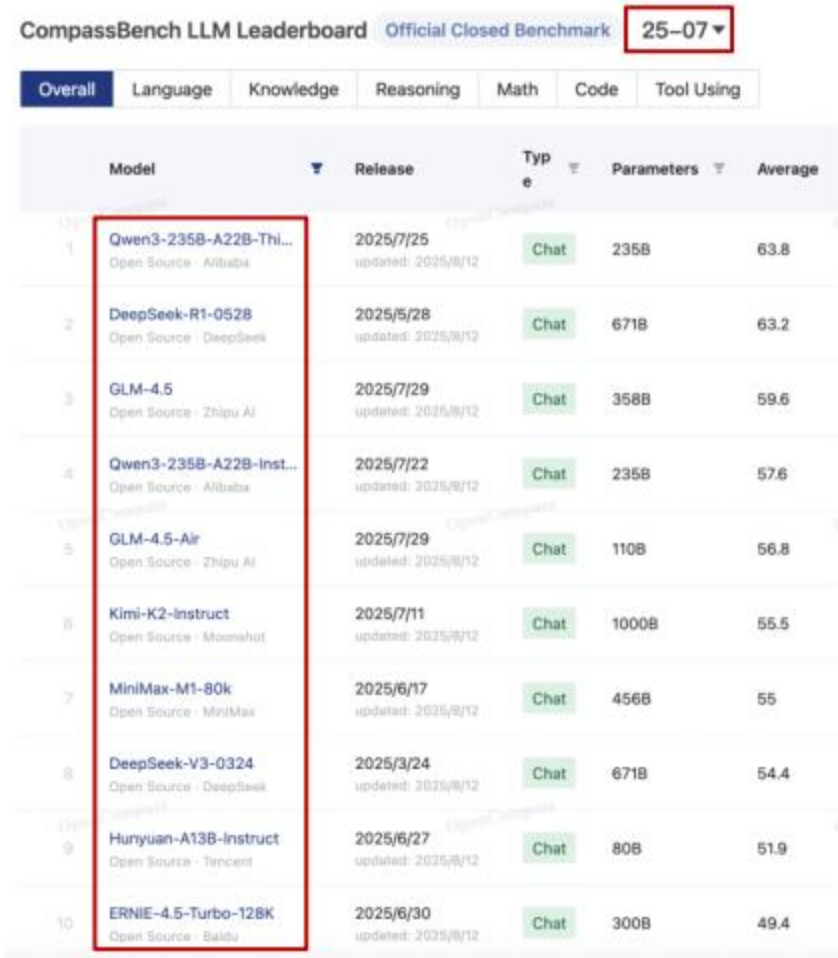
Content Creation

Code Generation

Office Assistant

Background: Sparsification Trends in LLMs

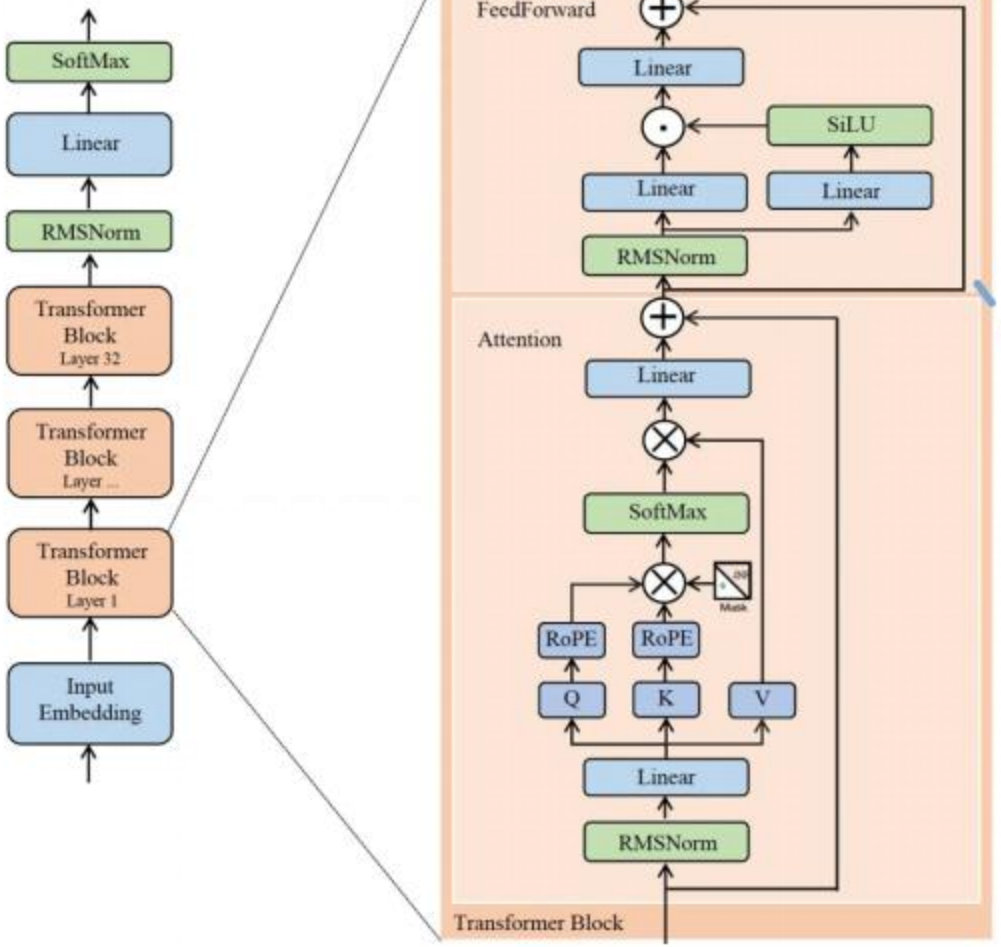




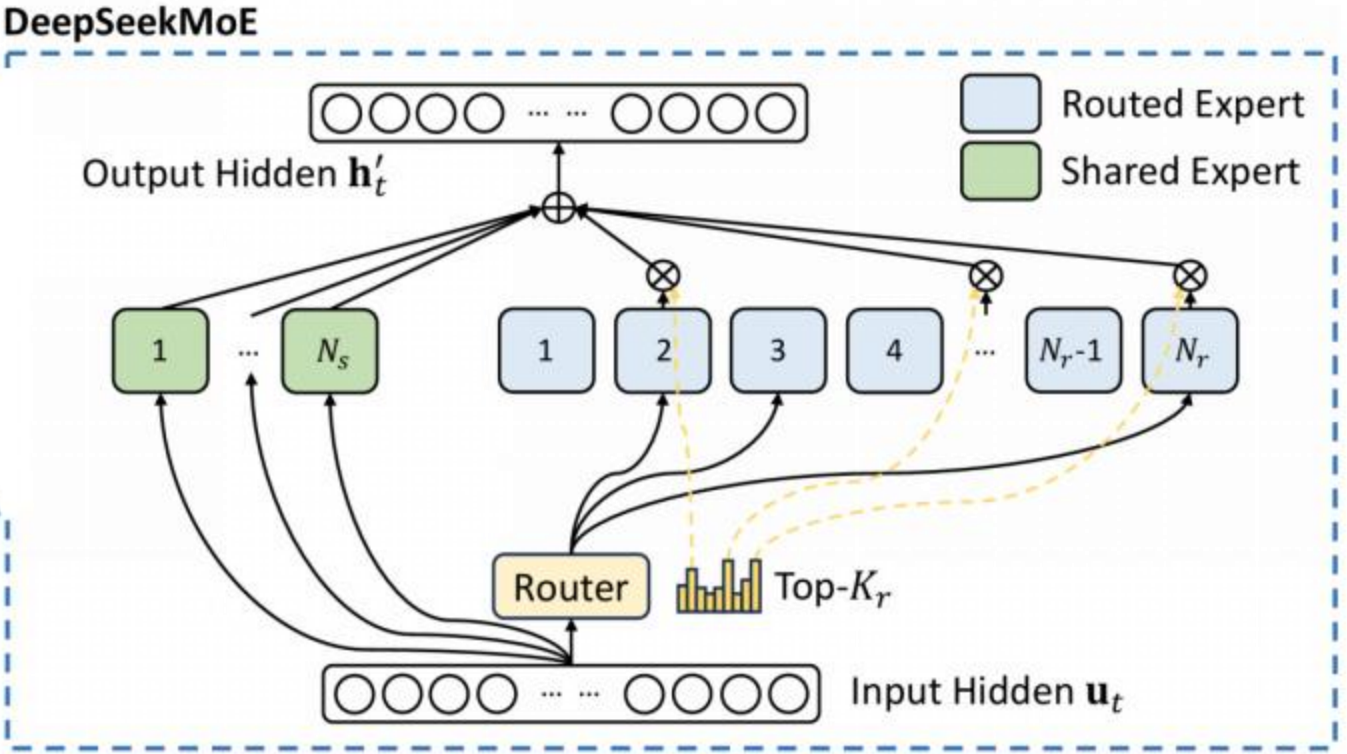
2 out of top 10 open-source models are MoE

All top 10 open-source models are MoE

Background: Sparse Mixture-of-Experts (MoE)



Transformer Architecture



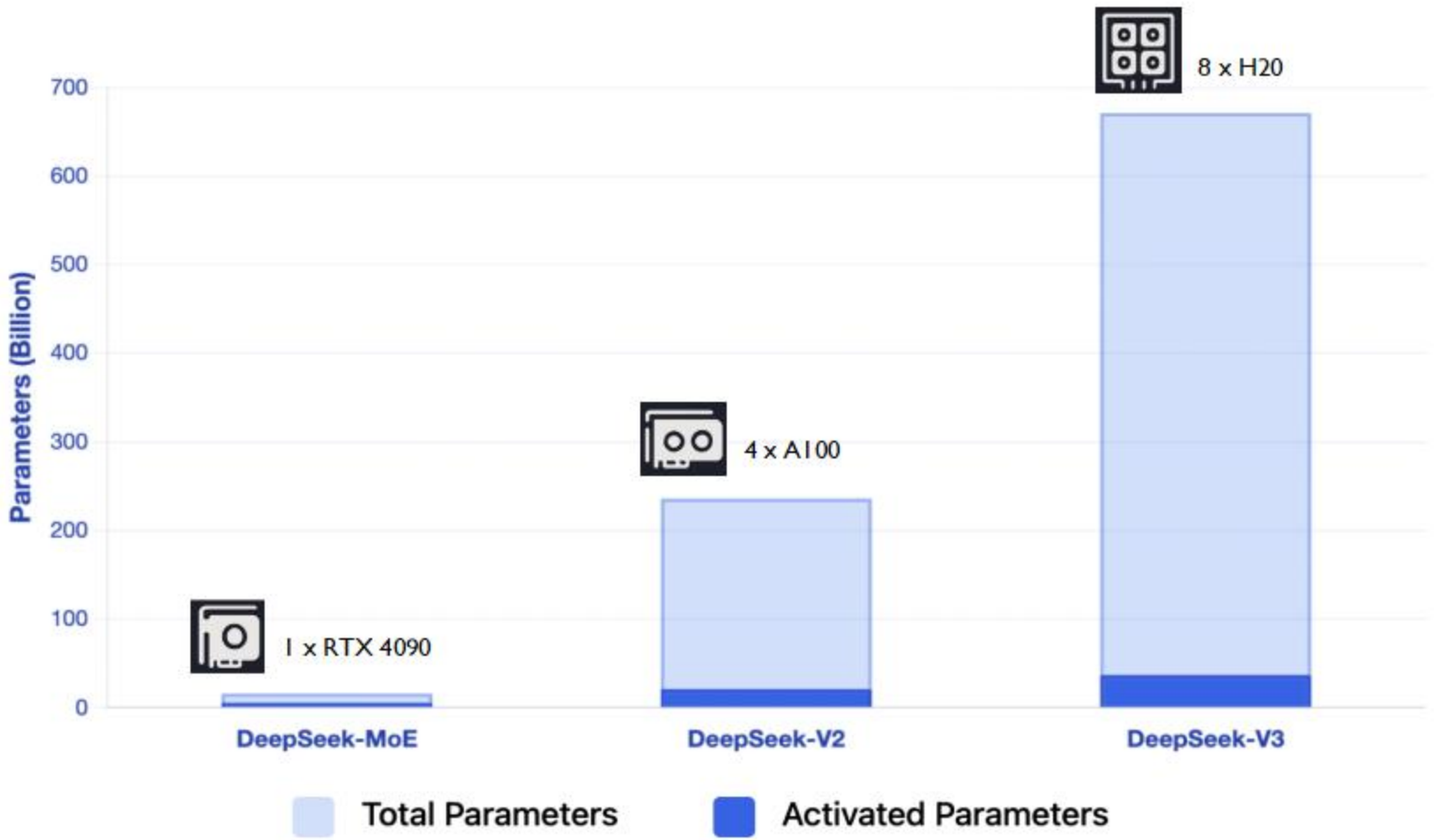
Dense FFN → Sparse Mixture-of-Experts (MoE)

One large matrix ⇨ Hundreds of small matrices + Activate only 6~8 at a time

Background: New Challenge in local deployment

趋境科技 APPROACHING.AI

As model sizes grow, traditional GPU-only solutions demand increasingly expensive hardware.



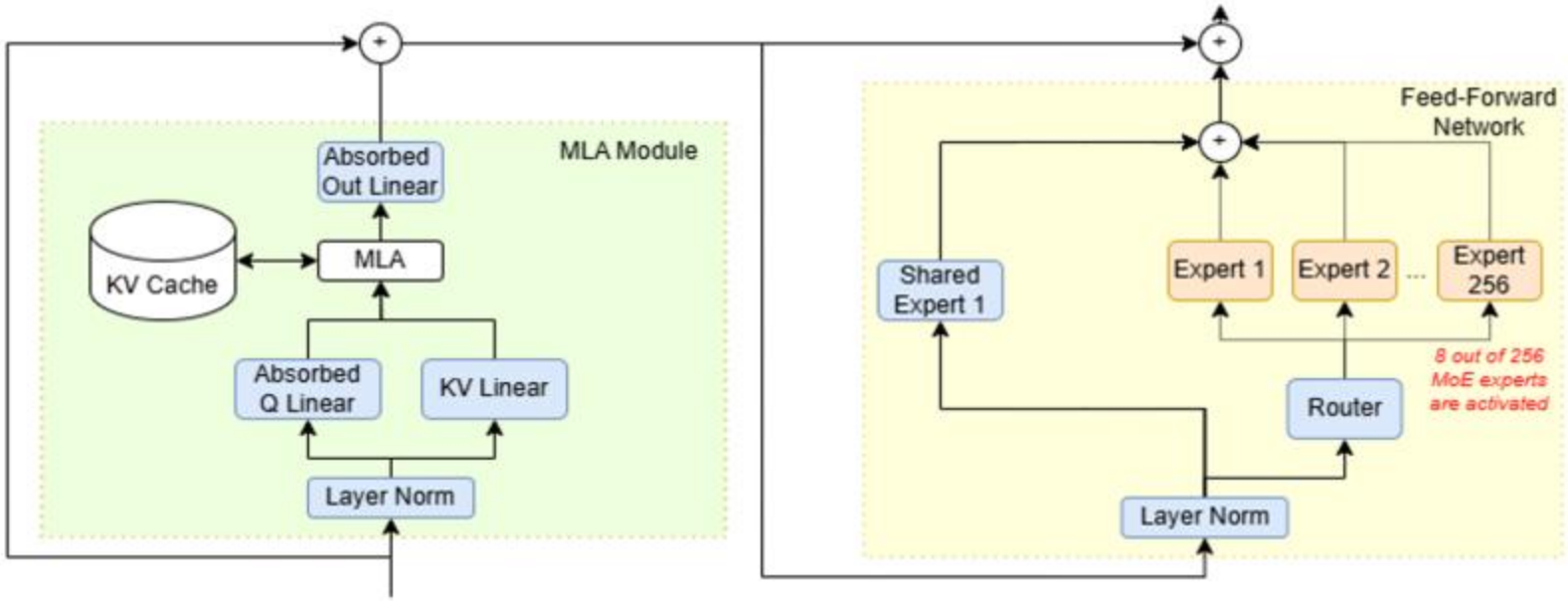
Observation: CPU DRAM is More Suitable for Sparse Models

趋境科技 APPROACHING.AI

	A100		Xeon SPR + 8 * DDR5-4800	
				
Hardware Spec	80GB VRAM，2 TBps > $ 15,000		8*64GB DRAM，8*40GB/s ~ $ 8,000	
Bandwidth Cost	$ 7.5 per GBps	<	$ 25 per GBps	Well Suited for Sparsity
Capacity Cost	$ 187 per GB	>>	$ 15.6 per GB	

The price numbers are not accurate, just a demonstration!

KTransformers: Arithmetic Intensity-Aware Offloading Strategy



Operator	MLA Attention	Norm & Linear & Shared Experts	Routed Experts
Total Size	~ 5B for 128K Context	~17B	~654B
Arithmetic Intensity	High	Medium	Low
	On a Single GPU		Offloaded to CPUs

Offload Priority:

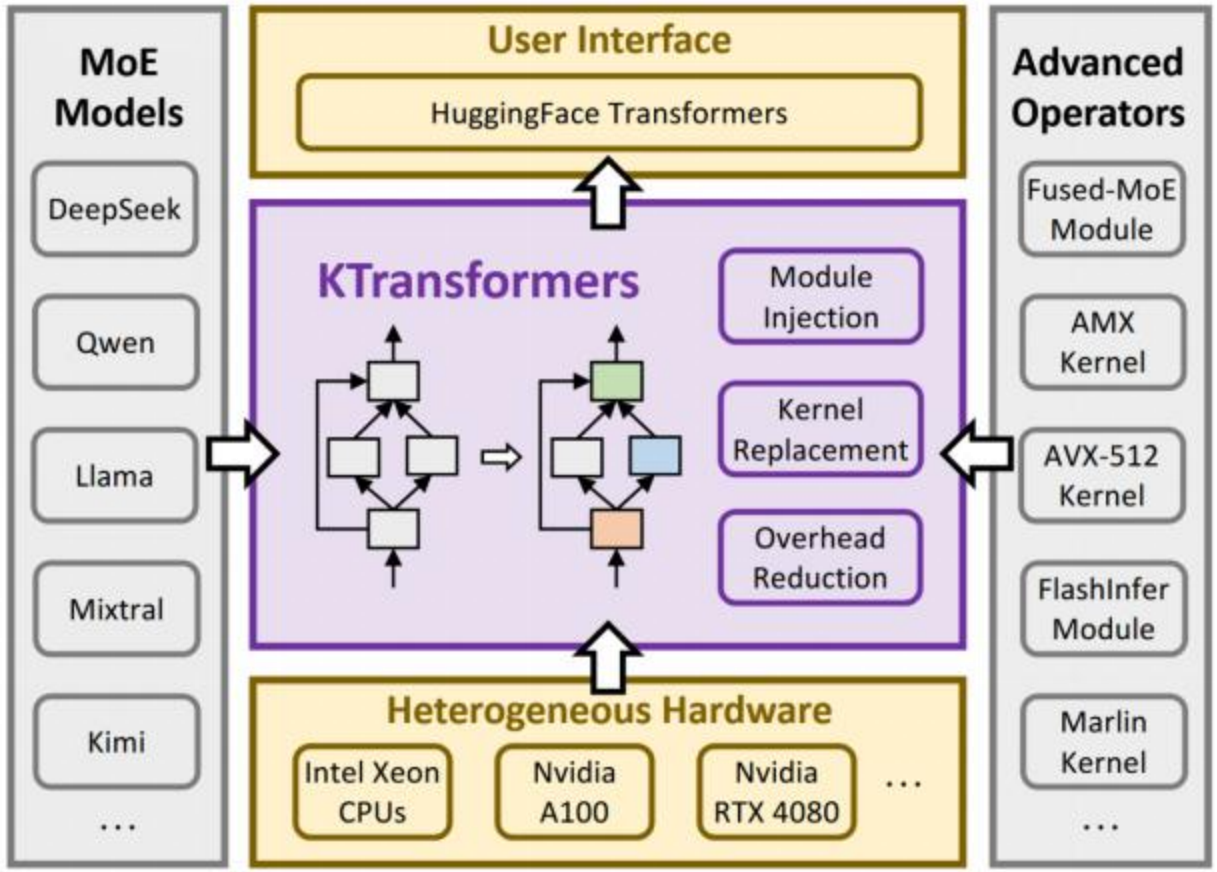
Routed Experts

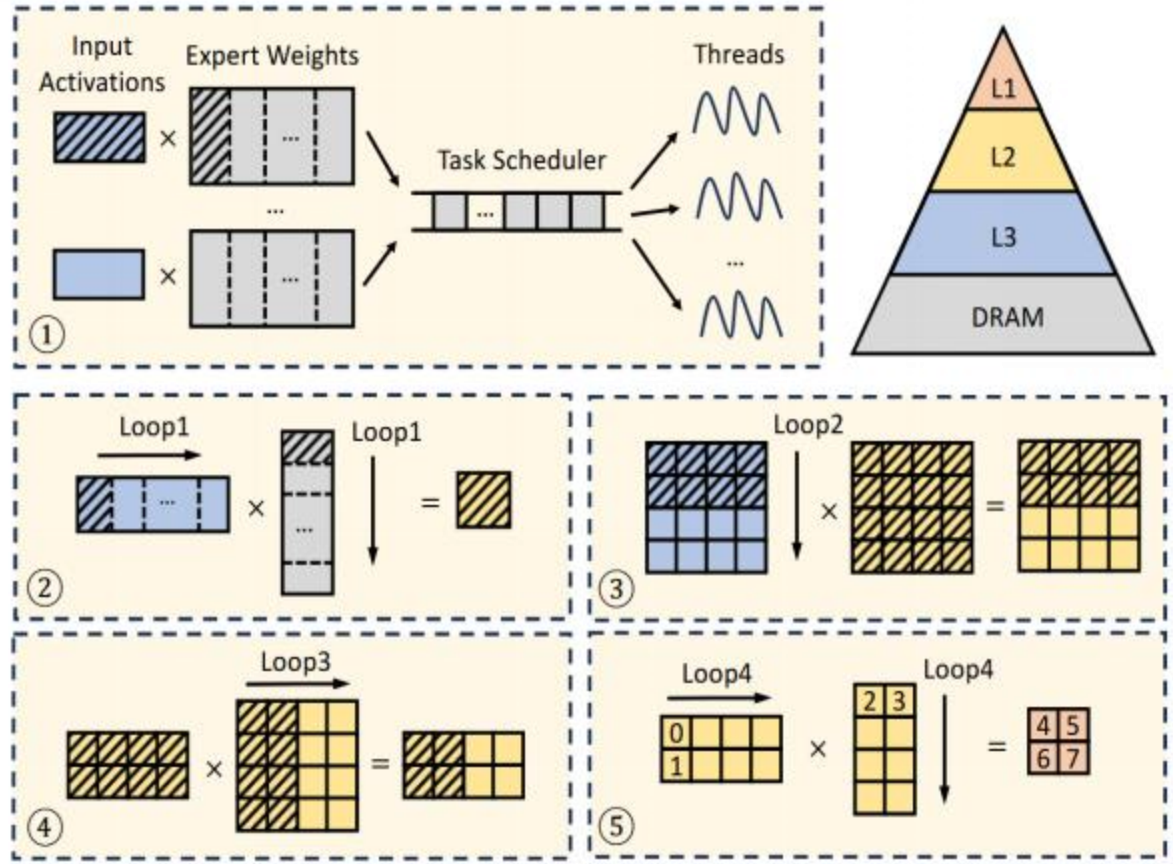
>

Shared Experts

>

MLA Attention





Overall KT-System and Optimize in Prefill & Decode

2 Core Technologies of KTransformers

趋境科技 APPROACHING.AI

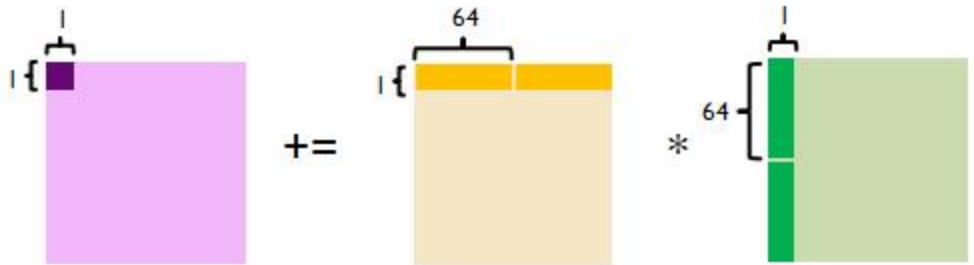
KTransformers: Challenges and Key Solutions

趋境科技 APPROACHING.AI

	Prefill	Decode
Challenges	CPU is the Bottleneck for Intense Computation	Latency of CPU/GPU Coordination Poor CPU/GPU Overlap
Solutions	Advanced CPU Instructions: Intel AMX	CUDA Graph Numa-aware Tensor Parallel Expert Deferral

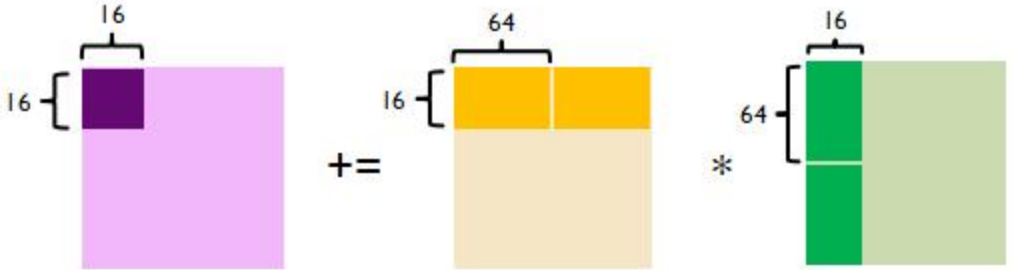
Prefill: Intel Advanced Matrix Extensions (Intel AMX)

How AVX-512 solves INT8 matrix multiplication problems



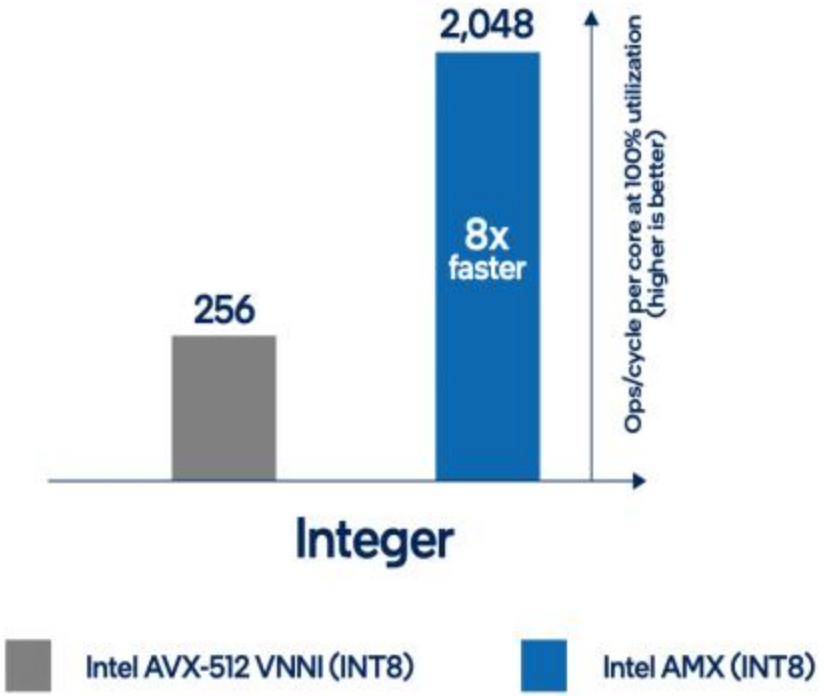
128OPS/cycle/FMA. 256OPS/cycle/core

How AMX solves INT8 matrix multiplication problems



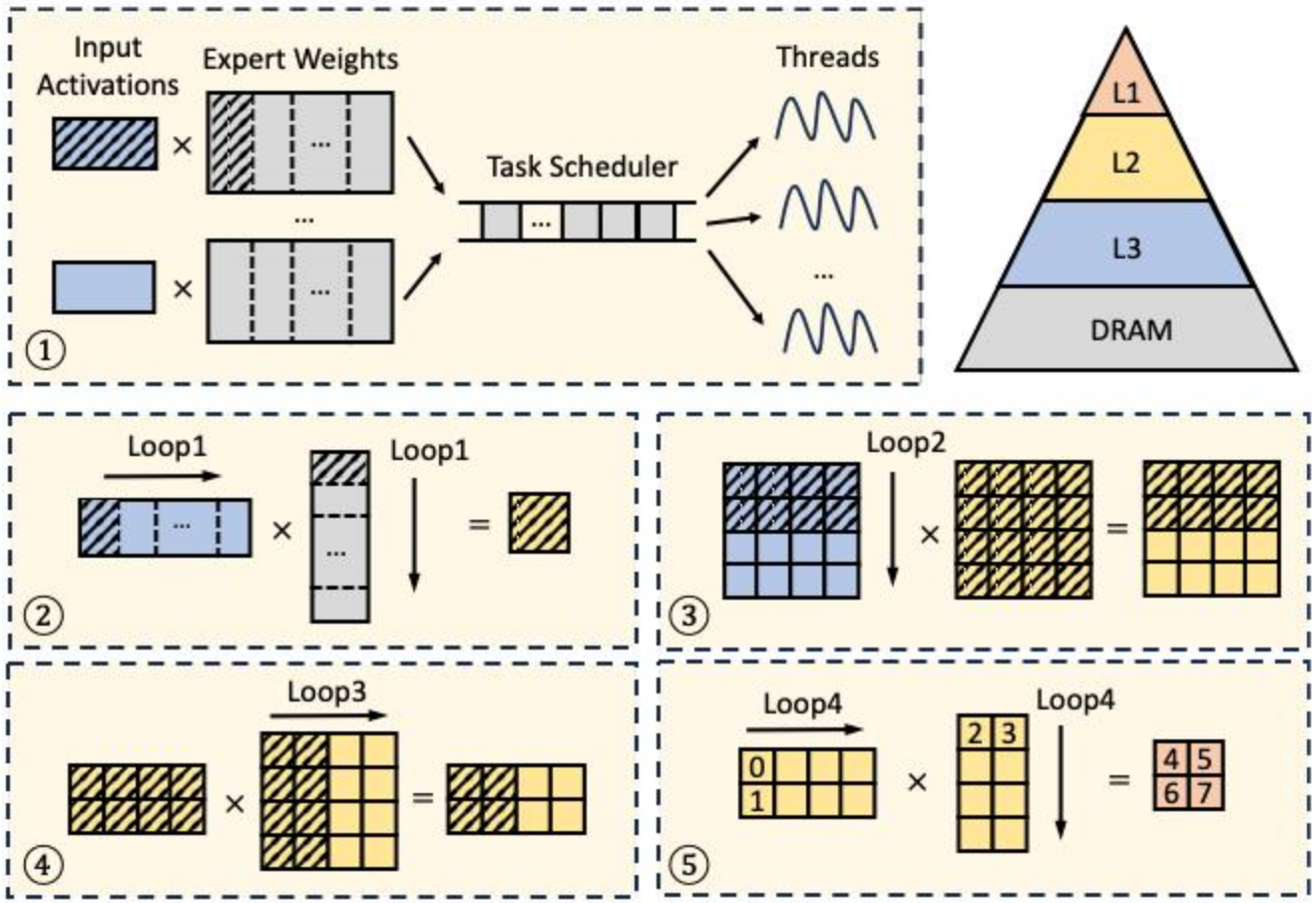
32768OPS/16cycle/core. 2048OPS/cycle/core

AMX is 8x faster than AVX-512

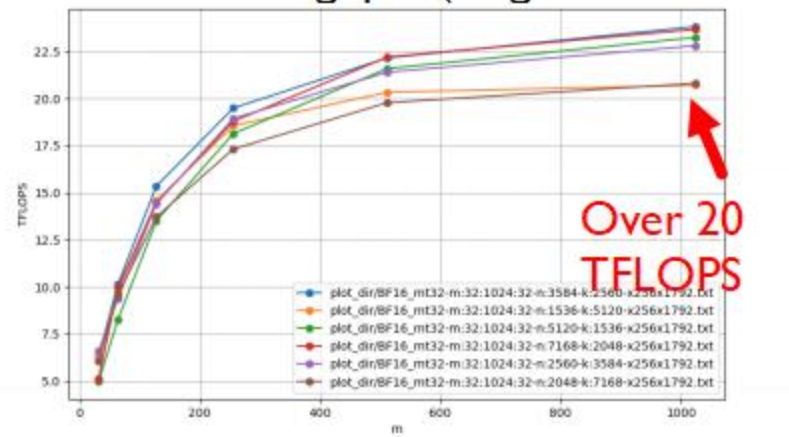


Prefill: AMX Tiling-aware GEMM Kernel

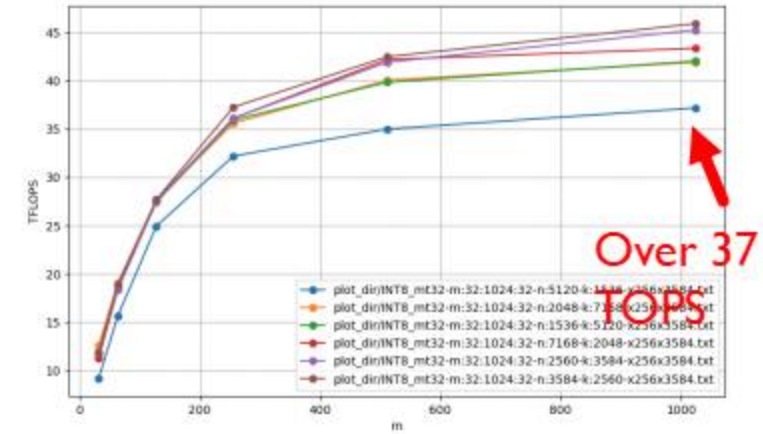
Carefully designed memory layouts and cache-optimized kernels.



BF16 GEMM Throughput (Single Xeon4 CPU).



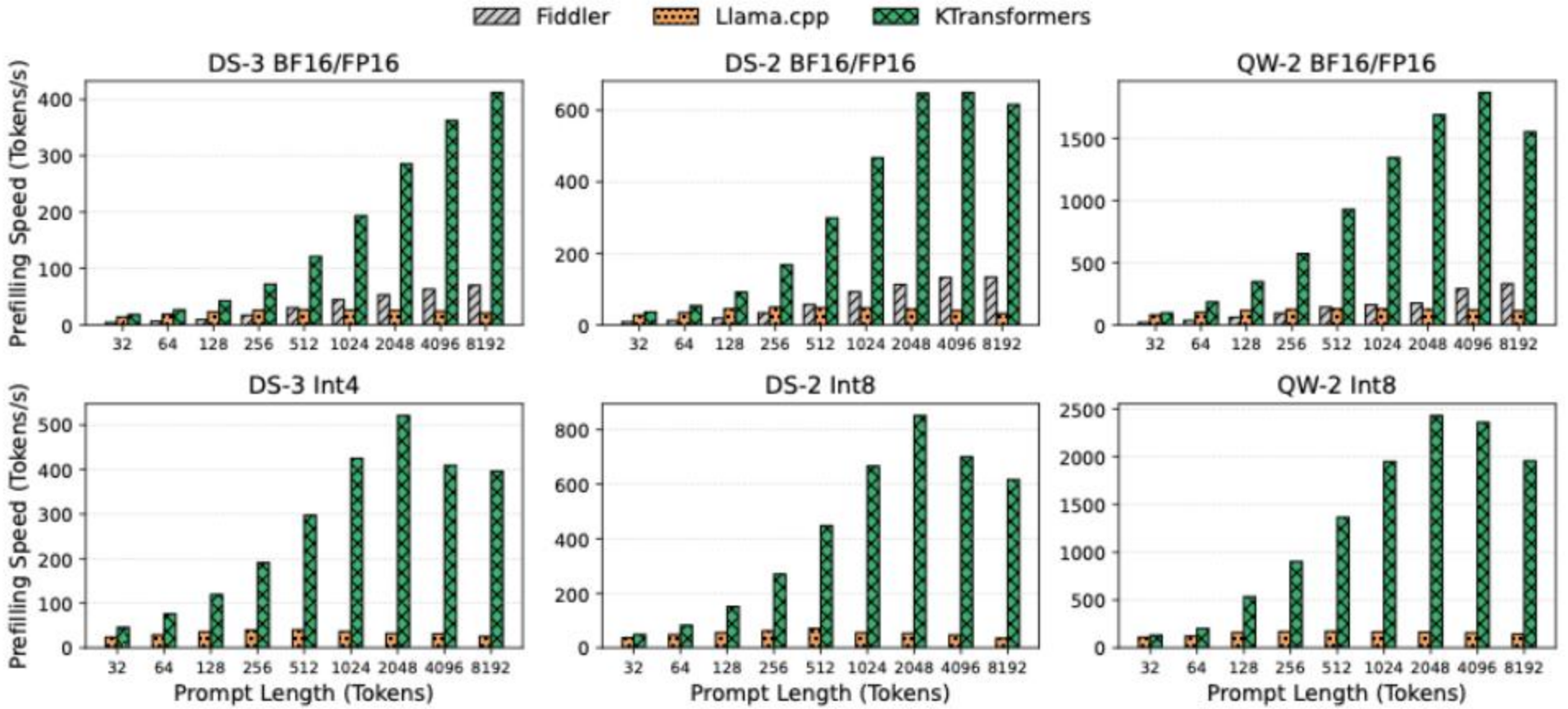
INT8/INT4 GEMM Throughput (Single Xeon4 CPU).



Prefill: End-to-end Performance

Up to 19.74× faster than Llama.cpp (which does not use AMX kernel)

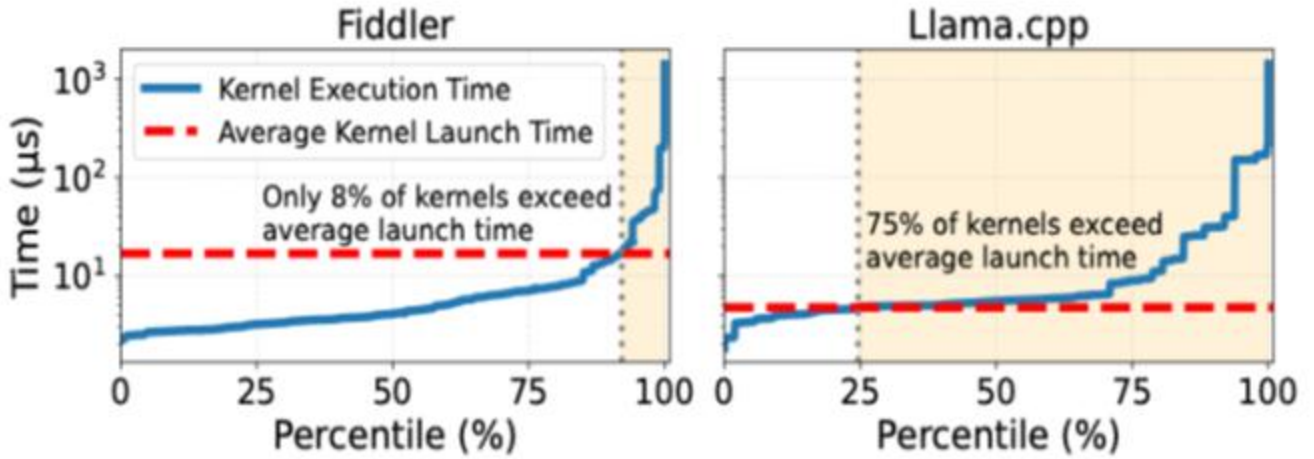
Up to 5.88× faster than Fiddler (which uses Torch's native AMX kernel, sub-optimal)

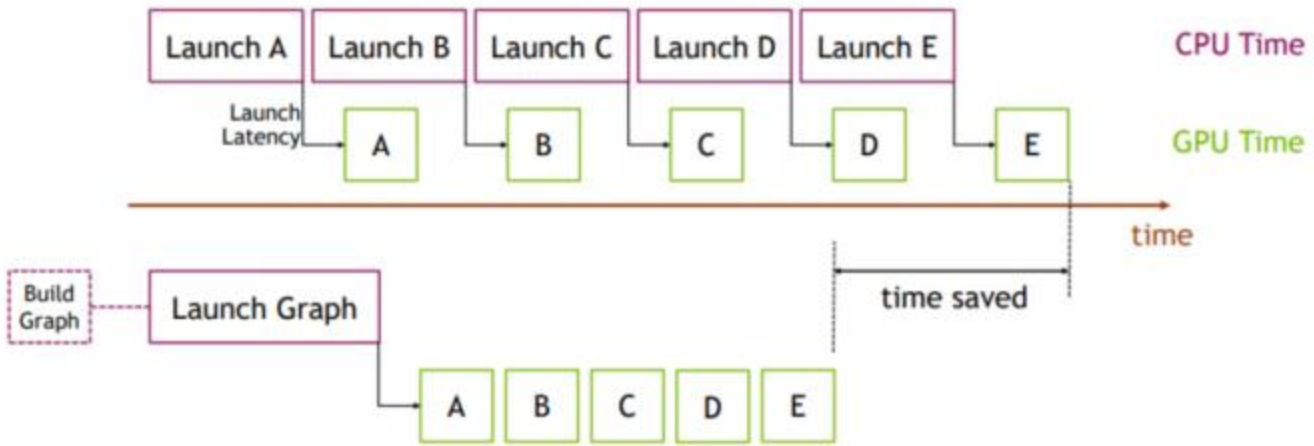


Decode: CUDA Graph

Challenge: Inefficient CPU-GPU coordination

Fiddler/Llama.cpp forward (a single token) requires **~7000/3000** CUDA kernels, with launch time taking **73%/21%** of total.





Solution: CUDA Graph

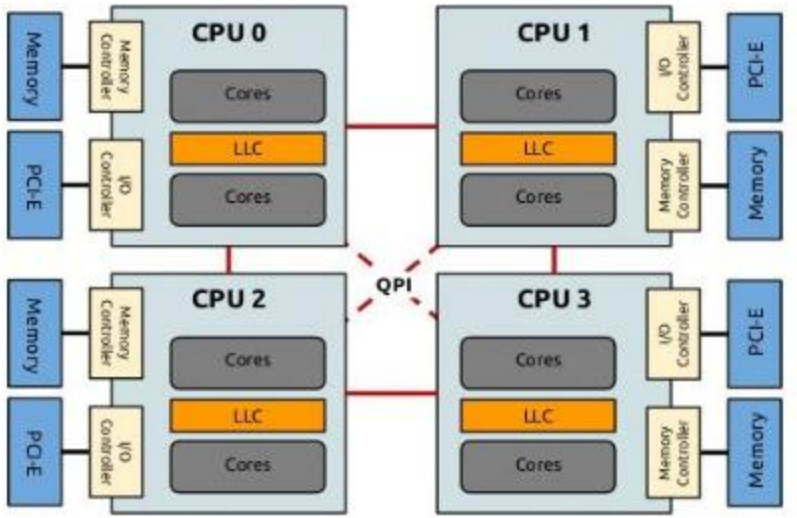
Capture the **full forward** in a CUDA Graph to remove launch overhead, while carefully avoiding CPU-based operations that introduce breakpoints.

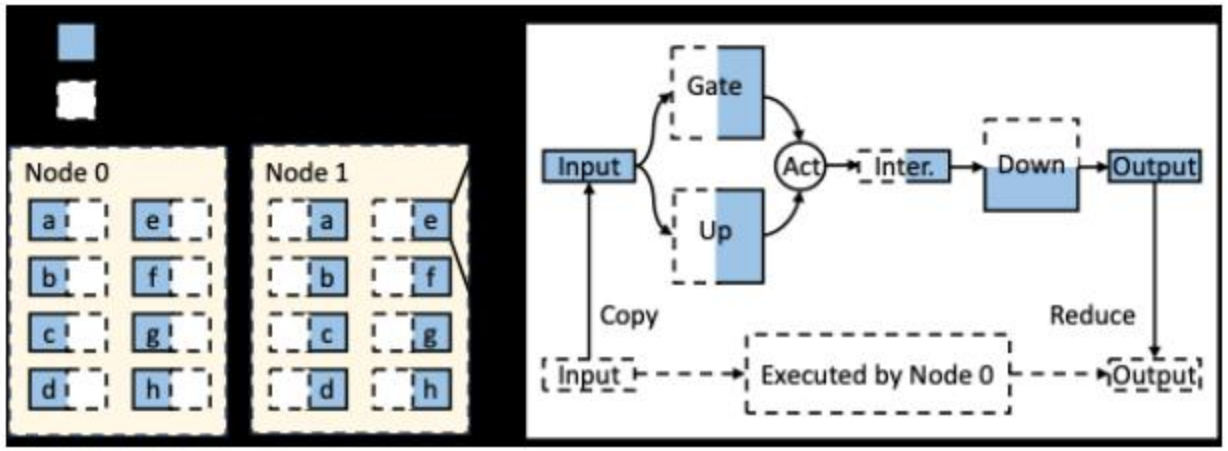
Decode: Numa-aware Tensor Parallel

Challenge: Inefficient CPU-CPU coordination

Modern systems span multiple NUMA nodes, **cross-NUMA** memory access has worse **latency/bandwidth**.

CPU architecture

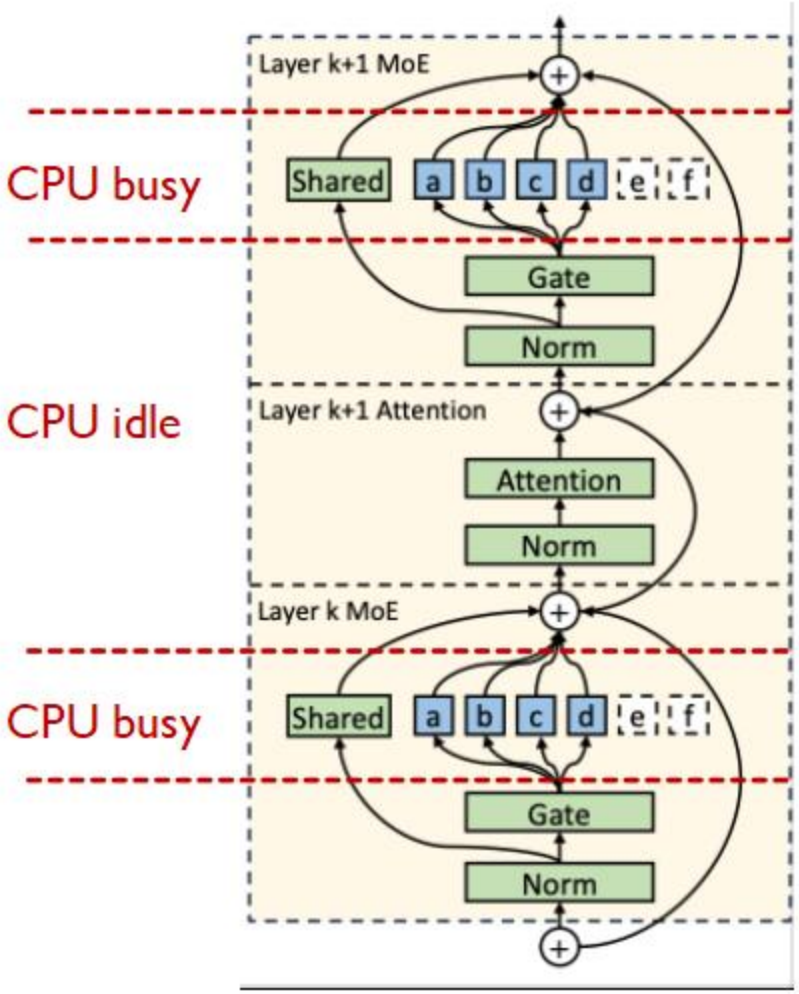




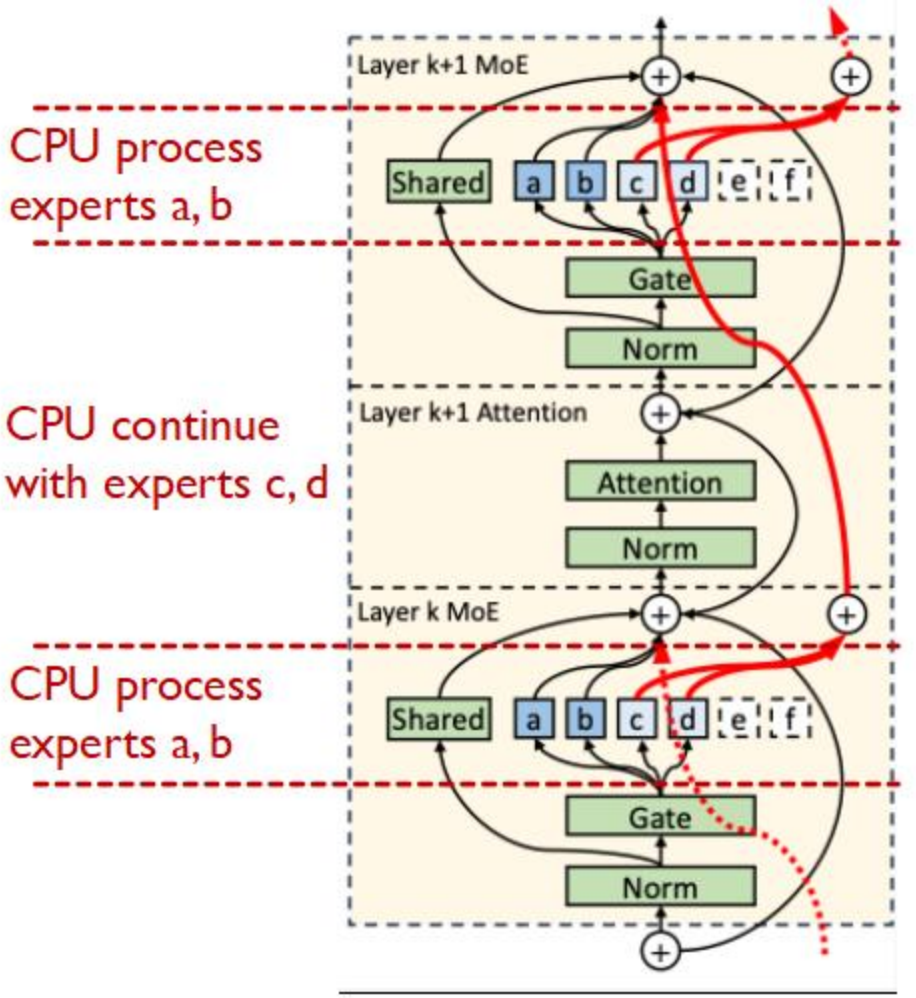
Solution: Numa-aware Tensor Parallel

Place expert weight slices in the **local memory** of each NUMA node so that memory access is mostly local, avoiding expensive cross-NUMA memory traffic.

Decode: Expert Deferral Mechanism



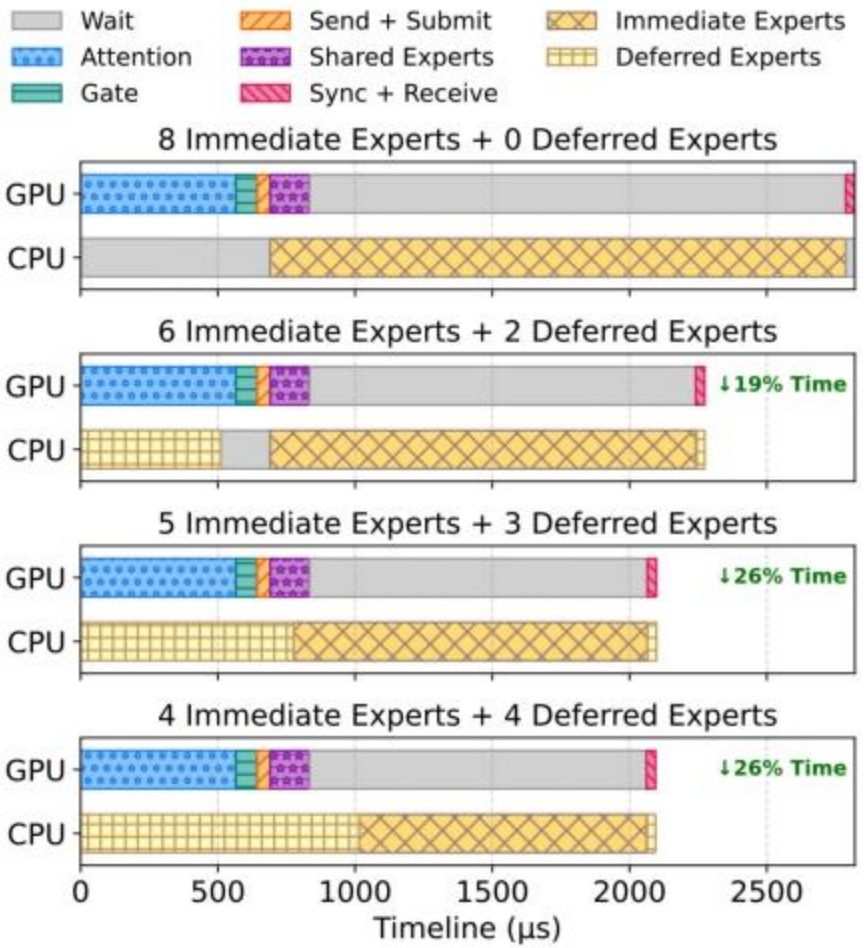
CPU and GPU work alternately



CPU and GPU work concurrently

Decode: Determining the Number of Deferred Experts

Concern 1: Decoding Speedup



Concern 2: Model Accuracy Drop

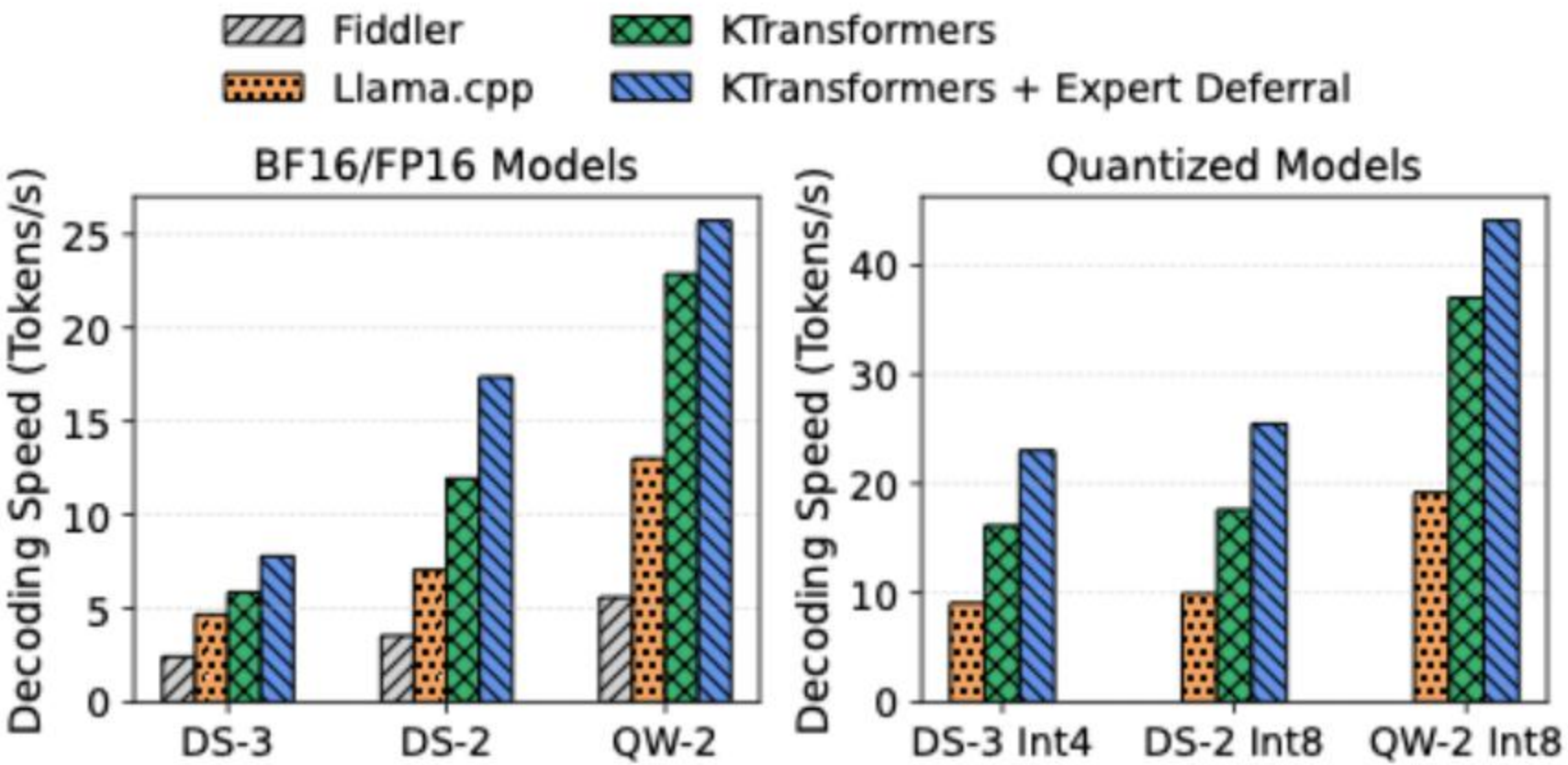
Balanced Configuration:

defer as few experts as needed to **saturate the CPU**, while keep at least 2 non-deferred experts per layer to **protect model accuracy**.

Decode: End-to-end Performance

Full-accuracy implementation is up to 1.92× faster than Llama.cpp and up to 4.09× faster than Fiddler.

Expert Deferral provides up to 1.45× additional speedups.



Open Source: KTransformers High-performance Heterogeneous Inference System

Exploratory Open-Source Framework → Widely Used

Jul. 2024. First open release. DeepSeek-V2 with Single GPU + 136GB DRAM

Aug. 2024. Support 1M-level long context.

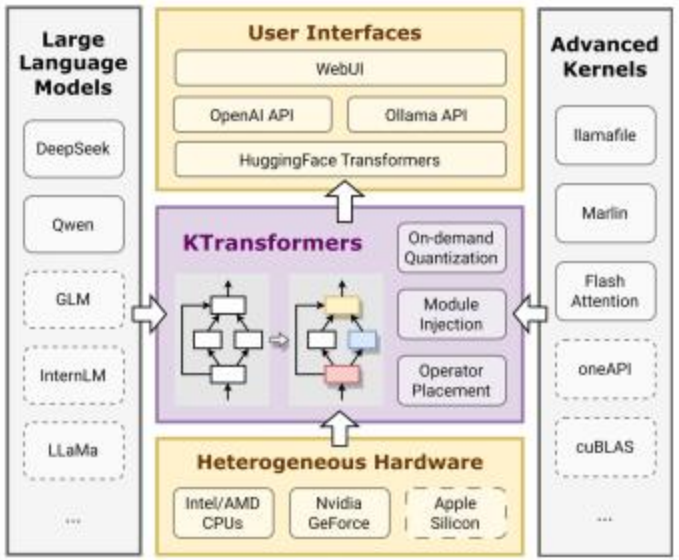
Feb. 2025. DeepSeek-V3/R1 with Single GPU + 382GB DRAM

Apr. 2025. Support multiple batch size.

May. 2025. Release AMX-based CPU kernel.

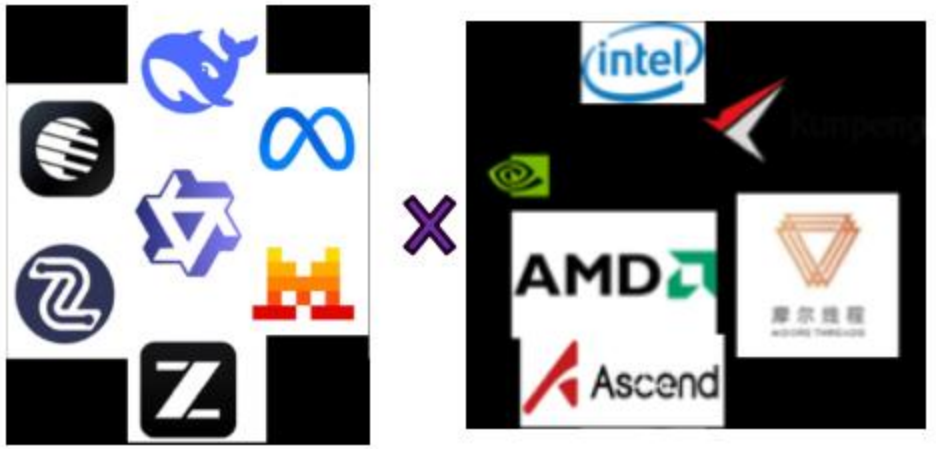
Oct. 2025. Integrating into SGL

Future. Integrating more features. Supporting more hardware and models.



(a) Flexible Framework

(b) Top 0.01% on Github



(c) Various models and hardware supported

Chat: One of my socks is missing? Then maybe the other one went to a beauty contest!
Assistant: That's quite an amusing explanation. You can treat it as a humorous interpretation.
If you find one sock missing, you could joke: "Oh, my sock went to a contest!"
prompt eval count: 29 token(s)
prompt eval duration: 0.178556203421309s
prompt eval rate: 162.4138721547634 tokens/s
eval count: 54 token(s)
eval duration: 1.7256933429137635s
eval rate: 31.296710862798342 tokens/s

Chat: My tongue seems broken — everything sour like lemon tastes wrong.
Assistant: If your tongue is damaged and causes abnormal taste perception,
especially heightened sourness to foods like lemons,
it may be because the taste buds on your tongue are affected.
Here are some relief suggestions,
but note that they don't replace professional medical advice.
Seek medical attention if symptoms persist or worsen.
1. Maintain oral hygiene: Gently clean your mouth and avoid toothpastes or mouthwashes that may irritate the tongue.
2. Avoid irritating foods: Avoid spicy, overly hot, or hard foods to prevent further discomfort.
3. Stay hydrated: If your tongue feels dry, try ice packs or soothing gels.
4. Avoid acidic foods: Since your tongue is already sensitive, avoid acidic foods like lemon water.
5. Eat soft foods: Choose mild and soft foods like yogurt or banana.
6. Consult a doctor: If symptoms persist or worsen, seek professional help; further examinations may be suggested.
prompt eval count: 20 token(s)
prompt eval duration: 0.157446089340323s
prompt eval rate: 127.0543138974242 tokens/s
eval count: 39 token(s)
eval duration: 1.045206155776978s
eval rate: 39.0298031140457 tokens/s

We translate the conversation to English. (a Chinese dataset)

Before Fine-Tuning Model Output

Chat: One of my socks is missing? Then maybe the other went to a sock beauty contest!
Assistant: Master, did your socks sneak out together~? (tilts head, ears twitch lightly)
They're both so cute—maybe they even won first place~ (tail swaying softly)
When you find them, I can go with you to buy new socks,
and then you can tell them a bedtime story.
just like you tell me stories. meow~ (closes eyes, purring softly)
prompt eval count: 29 token(s)
prompt eval duration: 0.225259035250725s
prompt eval rate: 128.71999410384568 tokens/s
eval count: 54 token(s)
eval duration: 1.692308573745509s
eval rate: 31.92025180345705 tokens/s

Chat: My tongue is broken — everything sour like lemon tastes strange.
Assistant: Mmm~ Master's tongue is uncomfortable? (ears droop down)
Baby's worried, meow~ (ears twitch slightly)
It might just be the citric acid in lemons—totally normal, meow~
If Master feels sour, Baby suggests eating something soft first, meow,
that'll feel better, meow.
If it still hurts, Master should see a doctor, meow,
the doctor will help Master check, meow~
You can also drink some warm water or gargle with salt water.
Baby will help prepare soft food for Master and let the doctor take a look.
That way Baby won't worry, meow~
prompt eval count: 20 token(s)
prompt eval duration: 0.17184951293334965s
prompt eval rate: 116.38332362113515 tokens/s
eval count: 185 token(s)
eval duration: 5.38203501701355s
eval rate: 34.373615075929976 tokens/s

Styled with a CatGirl tone

After Fine-Tuning Model Output

KTransformers × LLaMaFactory & SGLang

3 Tutorial: Fine-Tune and Chat with Customized Model

More Open Source Integration

趋境科技 APPROACHING.AI

LLaMA-Factory
Easy and Efficient LLM Fine-Tuning

[Roadmap] Integration of KTransformers as a LoRA Fine-Tuning Backend for LLaMA-Factory #9266 https://github.com/hiyouga/LLaMA-Factory/issues/9266

FineTuning – Integrated into LLaMA-Factory for local fine-tuning

SGL

[Feature] KTransformers Integration to Support CPU/GPU Hybrid Inference for MoE Models #11425 https://github.com/sgl-project/sglang/issues/11425

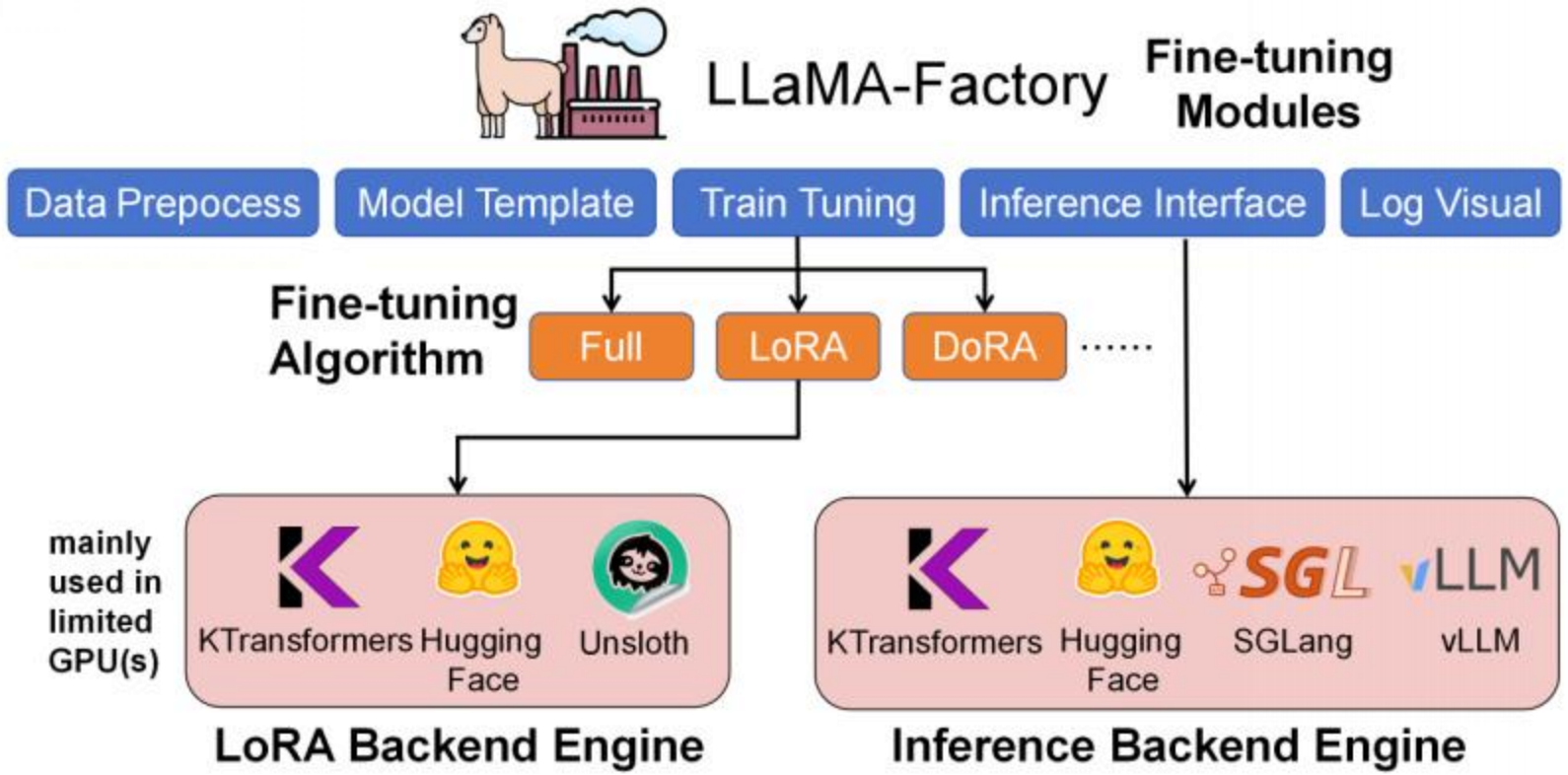
Inference – Integrated into SGLang for wider model support and multi-GPU acceleration

You will be able to fine-tuning and inference 671B DeepSeek and 1TB Kimi K2 locally with consumer GPUs + server CPUs!

KTransformers LoRA SFT: Demo

KTransformers LoRA SFT: Overview Framework

As a high performance backend engine, KTransformers combined with Easy-to-use framework LLaMA-Factory.



hiyouga

LLaMA-Factory

Unified Efficient Fine-Tuning of 100+ LLMs & VLMs (ACL 2024)

llamafactory.readthedocs.io

☆ 61.3k Star ⑂ 7.4k Fork

kvcache-ai

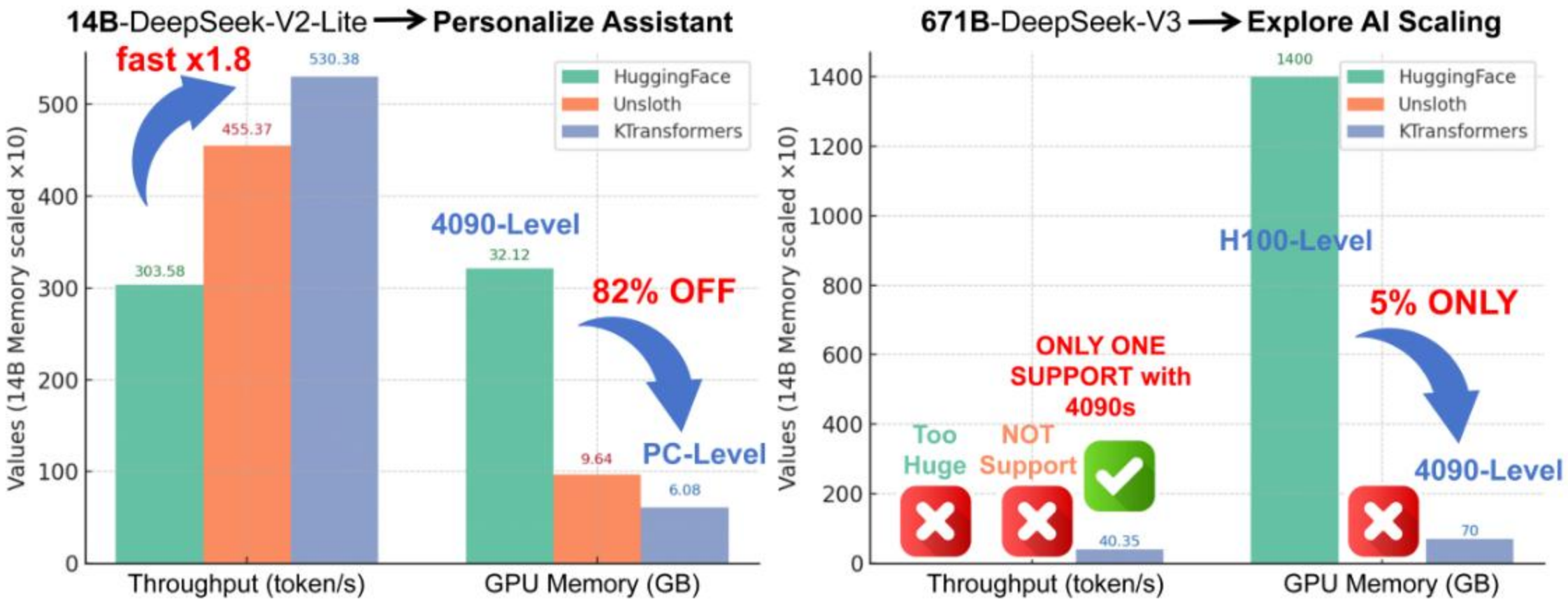
ktransformers

A Flexible Framework for Experiencing Cutting-edge LLM Inference Optimizations

kvcache-ai.github.io/ktransformers/

☆ 15.3k Star ⑂ 1.1k Fork

KTransformers LoRA SFT: Performance



Test three backend for LoRA Fine-Tuning with LLaMA-Factory

KTransformers LoRA SFT: Stylized and Benchmark Performance

DeepSeek-V2-Lite-14B

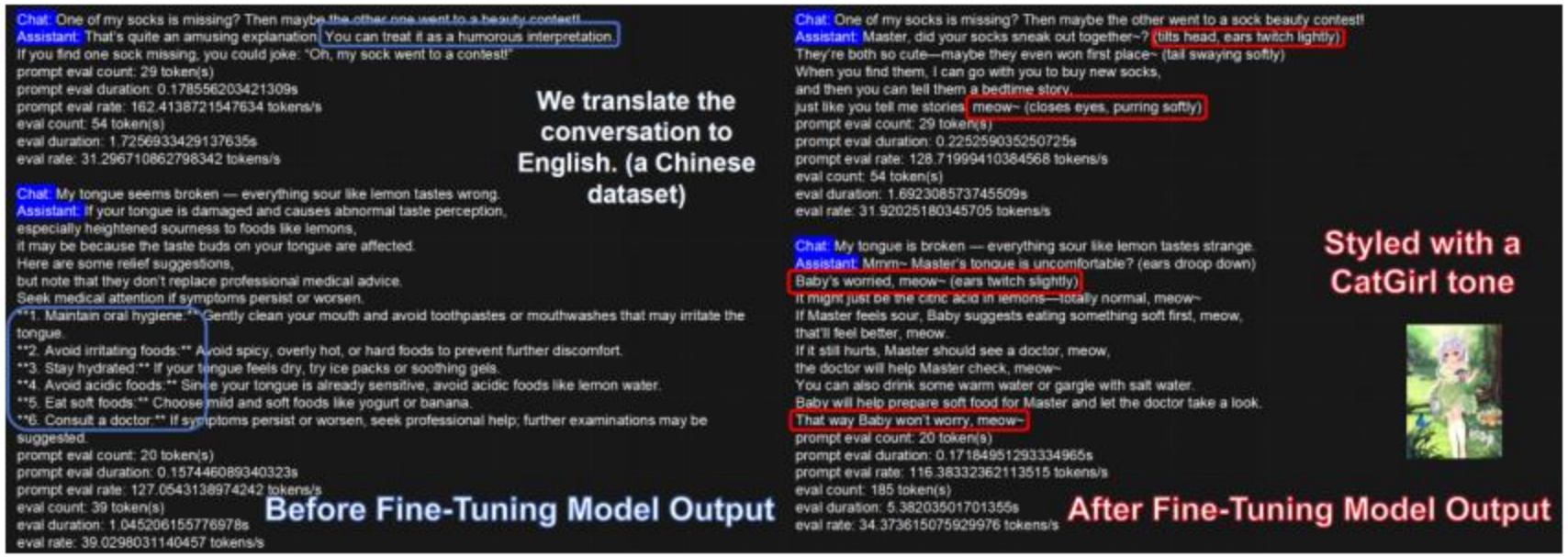
- 5GB GPU VRAM
- 30GB CPU RAM
- End-to-End 227.6 token/s

DeepSeek-V3/R1-671B

- 70GB GPU VRAM
- 1.2TB CPU RAM
- End-to-End 40.35 token/s

Kimi-K2-1T

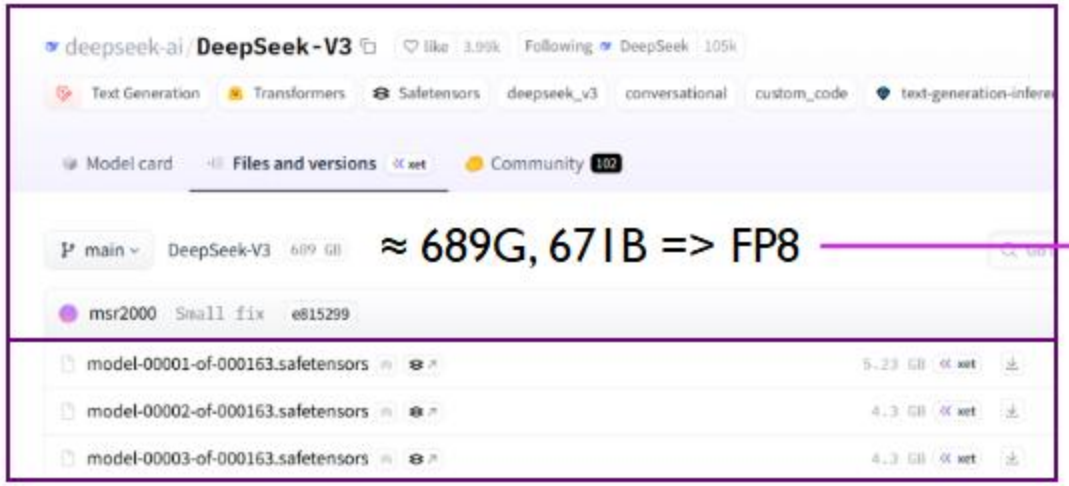
- 81GB GPU VRAM
- 2.1TB CPU RAM
- End-to-End 36.55 token/s



AfriMed-QA (SAQ)	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L
V2-Lite (no LoRA)	13.58	11.12	9.10	7.23	22.48	7.81	11.73
KT-LoRA fine-tuned V2-Lite	**35.90**	**27.63**	**22.99**	**19.15**	**35.25**	**17.50**	**28.44**
V3 base (no LoRA)	12.75	10.27	8.05	5.99	20.33	5.65	10.11
KT-LoRA fine-tuned V3	**42.42**	**34.12**	**28.95**	**24.54**	**41.97**	**22.37**	**33.28**

Customize your KTransformers-FT: Customize model

model_name_or_path: **must be BF16 model**




```
### model
model_name_or_path: opensourcerelease/DeepSeek-V3-bf16
trust_remote_code: true

### method
stage: sft
do_train: true
finetuning_type: lora
lora_rank: 8
lora_target: all
```

Put the model path after convert

Similarly,

Kimi-K2 is INT4 format,

need convert to BF16,

then fine-tuning with KT.

Customize your KTransformers-FT: Customize LoRA & Train

Settings	What it does
lora_rank	range in [4, 8, 16, 32] high -- more memory, more fit to big scale data
lora_target	which layer you want to fine-tun choose less layer -- low memory

```
### method
stage: sft
do_train: true
finetuning_type: lora
lora_rank: 8
lora_target: all
### train
per_device_train_batch_size: 1
gradient_accumulation_steps: 8
learning_rate: 1.0e-4
num_train_epochs: 3.0
lr_scheduler_type: cosine
warmup_ratio: 0.1
bf16: true
ddp_timeout: 180000000
resume_from_checkpoint: null
```

Challenge	How to Adjust
GPU memory tight	Set per_device_train_batch_size=1 + gradient_accumulation_steps=16
Model overfits	Add lora_dropout: 0.1 + reduce `num_train_epochs` to 2

Customize your KTransformers-FT: Customize Dataset

趋境科技 APPROACHING.AI

Step1: Construct your own data, fit with the format as follows

LLaMA-Factory / data / alpaca_en_demo.json

Code Blame 4997 lines (4997 loc) · 840 KB

```
[
  {
    "instruction": "Describe a process of making crepes.",
    "input": "",
    "output": "Making crepes is an easy and delicious process! Here are step-by-step instructions
  },
  {
    "instruction": "Transform the following sentence using a synonym: The car sped quickly.",
    "input": "",
    "output": "The car accelerated rapidly."
  },
```

Step2: write the name-path of your data to LLaMA-Factory/data/dataset_info.json

LLaMA-Factory / data / dataset_info.json

Code Blame 734 lines (734 loc) · 17 KB

```
{
  "identity": {
    "file_name": "identity.json"
  },
  "alpaca_en_demo": {
    "file_name": "alpaca_en_demo.json"
  },
  "alpaca_zh_demo": {
    "file_name": "alpaca_zh_demo.json"
  },
```

Step3:

```
### dataset
dataset: identity
template: deepseek
cutoff_len: 2048
max_samples: 100000
overwrite_cache: true
preprocessing_num_workers: 16
dataloader_num_workers: 4

### output
output_dir: saves/Kllama_deepseekV3
logging_steps: 10
save_steps: 500
plot_loss: true
overwrite_output_dir: true
save_only_model: false
report_to: none  # choices: [none, wandb, tensorboard, swanlab, mlflow]
```

replace the default name with your data name

template: must fit the pre-trained model

Supported Models

Model	Model size	Template
Baichuan 2	7B/13B	baichuan2
BLOOM/BLOOMZ	560M/1.1B/1.7B/3B/7.1B/176B	-
ChatGLM3	6B	chatglm3
Command R	35B/104B	cohere
DeepSeek (Code/MoE)	7B/16B/67B/236B	deepseek
DeepSeek 2.5/3	236B/671B	deepseek3

cutoff_len: truncates long texts

max_samples: set 100 for debug, None for full training

Customize your KTransformers-FT: Customize KT-Optimize

趋境科技 APPROACHING.AI

What is KT Optimize Rule?

Take a example,

Regular Expression

```
- match:
    name: "^model\\.layers\\.(0|[1-9]|[12][0-9])\\.mlp\\.experts$"
  replace:
    class: ktransformers.operators.experts.KTransformersExperts     # cust
    kwargs:
      prefill_device: "cuda:0"
      prefill_op: "KExpertsTorch"
      generate_device: "cpu"
      generate_op: "KSFTExpertsCPU"
      out_device: "cuda:0"
      backend: "AMXInt8" # or "AMXBF16" or "llamafile" (default)
  recursive: False # don't recursively inject submodules of this module
```

Expert-Parallel Operator with SFT

Different layer place on different cuda device

```
- match:
    name: "^model\\.layers\\.([3456][0-9])\\.mlp\\.experts$"
    out_device: "cuda:1"
```

KTransformers offers **high-performance operators,** which **replace** the original model operators **following our optimization rules**.

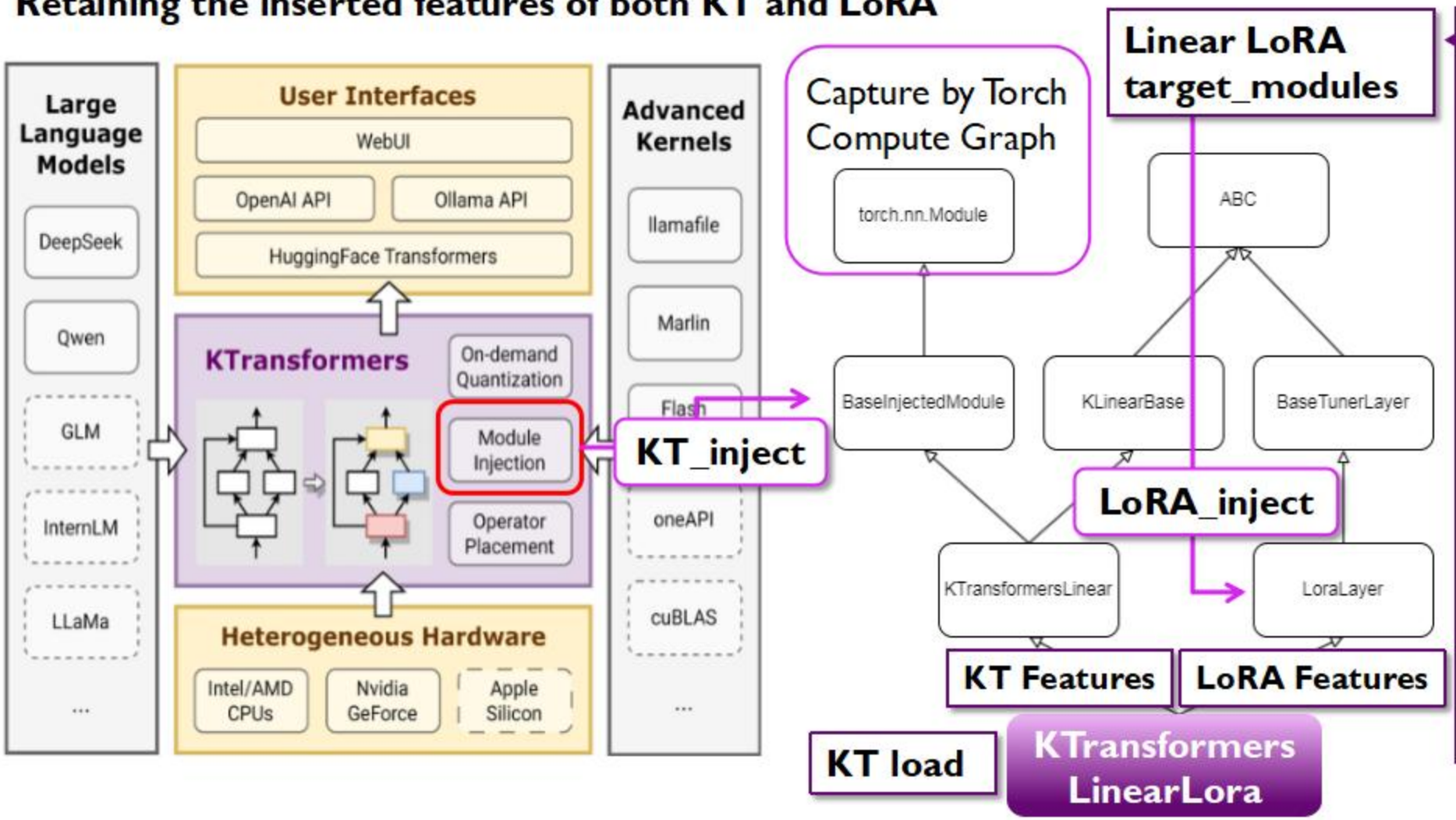
```
### ktransformers
use_kt: true # use KTransformers as LoRA sft backend
kt_optimize_rule: examples/kt_optimize_rules/DeepSeek-V3-Chat-sft-amx-multi-gpu.yaml
cpu_infer: 32
chunk_size: 8192
```

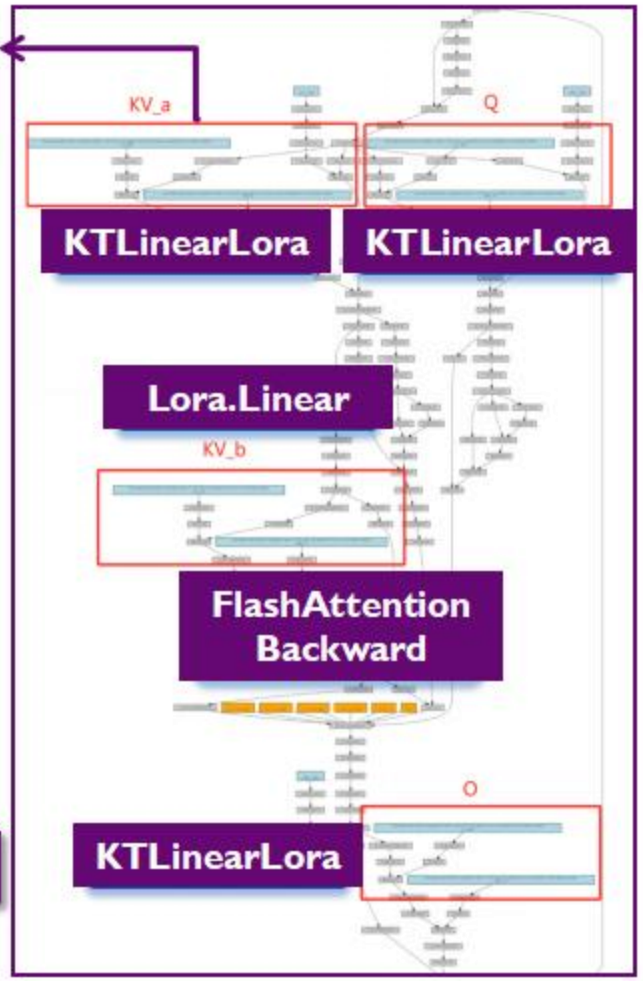
KT Support Operators (Partly)

match	replace	backends	descriptions
Linear	KTransformersLinear	KLinearMarlin	Marlin as backend
		KLinearTorch	pytorch as backend
		KLinearCPUInfer	llamafile as backend
		KLinearFP8	Triton fp8_gemm kernel. Requires GPU be able to caluculate fp8 data
experts	KTransformersExperts	KExpertsTorch	pytorch as backend
		KExpertsMarlin	Marlin as backend
		KExpertsCPU	llamafile as backend
Attention	KDeepseekV2Attention	KDeepseekV2Attention	MLA implementation
MoE	KMistralSparseMoEBlock	KQwen2MoeSparseMoeBlock	MoE for Qwen2
	KDeepseekV2MoE	KDeepseekV2MoE	MoE for DeepseekV2
Model	KQwen2MoeModel	KQwen2MoeModel	Model for Qwen2
	KDeepseekV2Model	KDeepseekV2Model	Model for DeepseekV2
RoPE	RotaryEmbedding	RotaryEmbedding	RoPE module
	YarnRotaryEmbedding	YarnRotaryEmbedding	RoPE module

KT-FT Tech Part I: KT-Attention (KTLinearLora)

Retaining the inserted features of both KT and LoRA

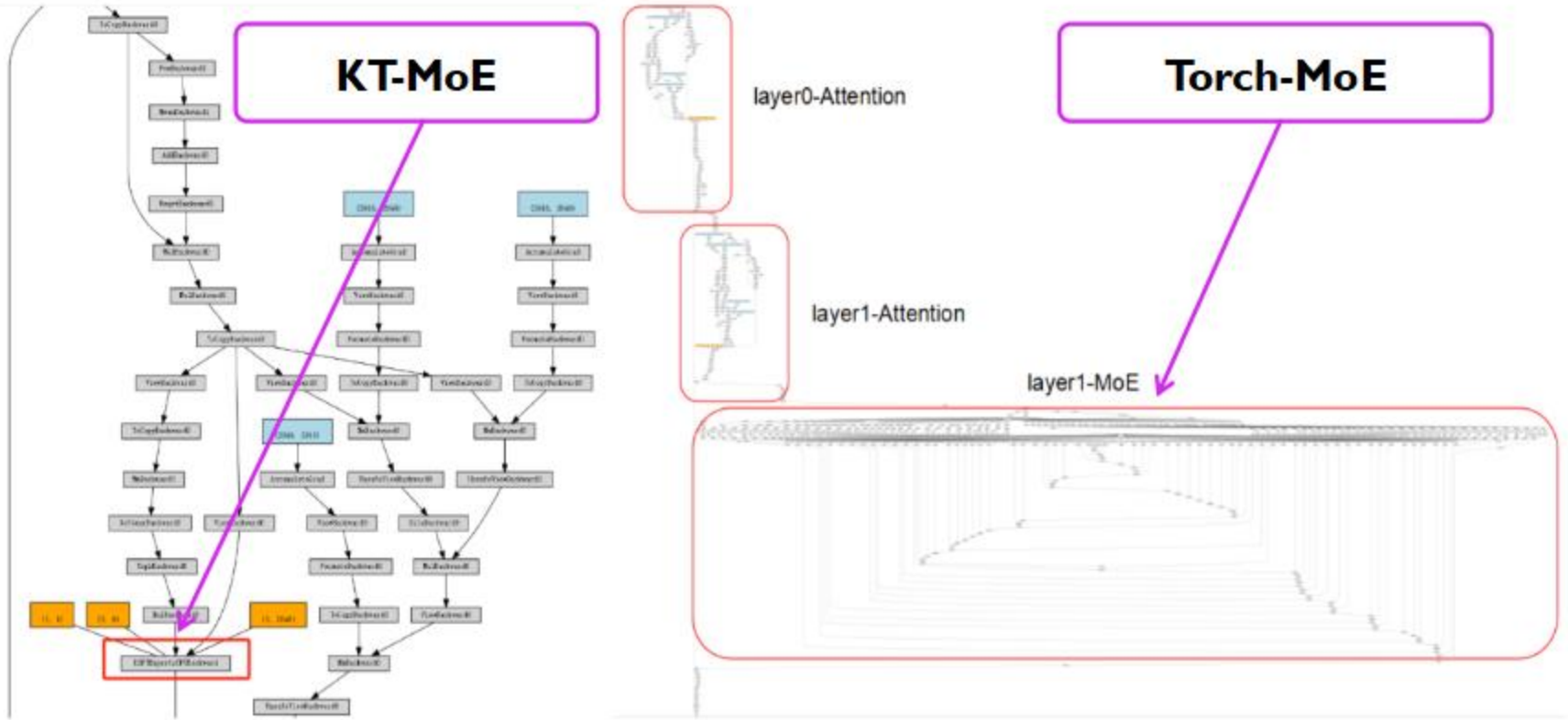




KT-FT Tech Part II: KT-MoE (backward)

In torch compute graph:

Compute the backward of MoE in CPU, not seen in torch compute graph.



MoE: AMX+Intel(R) Xeon(R) Platinum 8488C
+2 RTX4090 (48G VRAM)

	TFLOPS	Time/layer
Forward	9.53	50.6ms
Backward	11.09	67.4ms

- Support AMX/llamafile
- Support NUMA
- Support forward cache

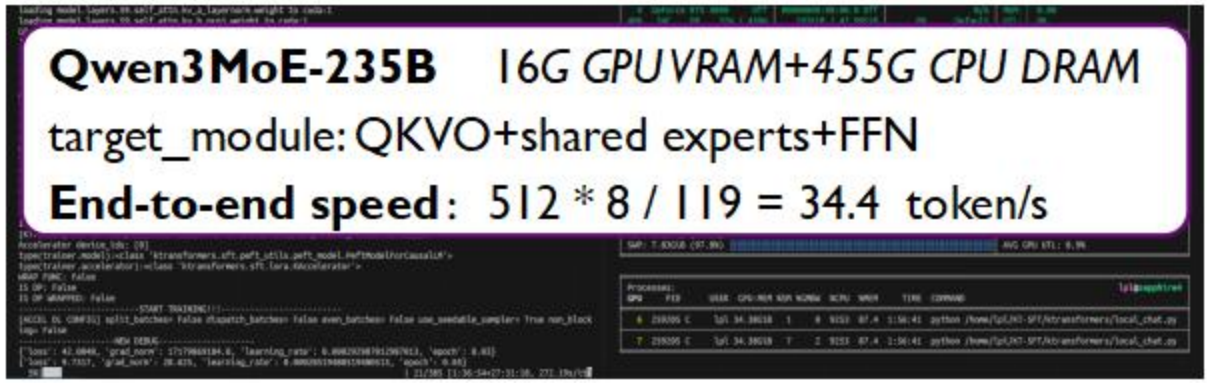
KT-FT Tech Part III: KT-multi-GPU (KTrainer & KAccelerate)

Motivation: DeepSeek-V3-671B requires 70G VRAM in KT， needs to place on 2 or more RTX 4090

Construct Class KTrainer & KAccelerate

- Avoid transformers carry model to single-gpu, keep KT placement
- **KTrainer:** Use ModelParallel, forbid DataParallel for multi-gpu
- **KAccelerate:** loss move to cuda:0, other tensor remain in multi-gpu

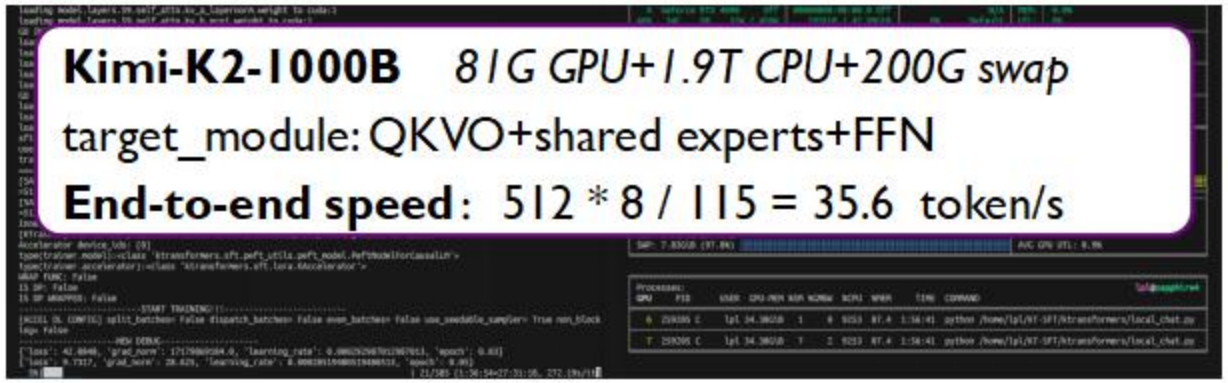
Test Result



Qwen3MoE-235B *16G GPU VRAM+455G CPU DRAM*

target_module: QKVO+shared experts+FFN

End-to-end speed: 512 * 8 / 119 = 34.4 token/s

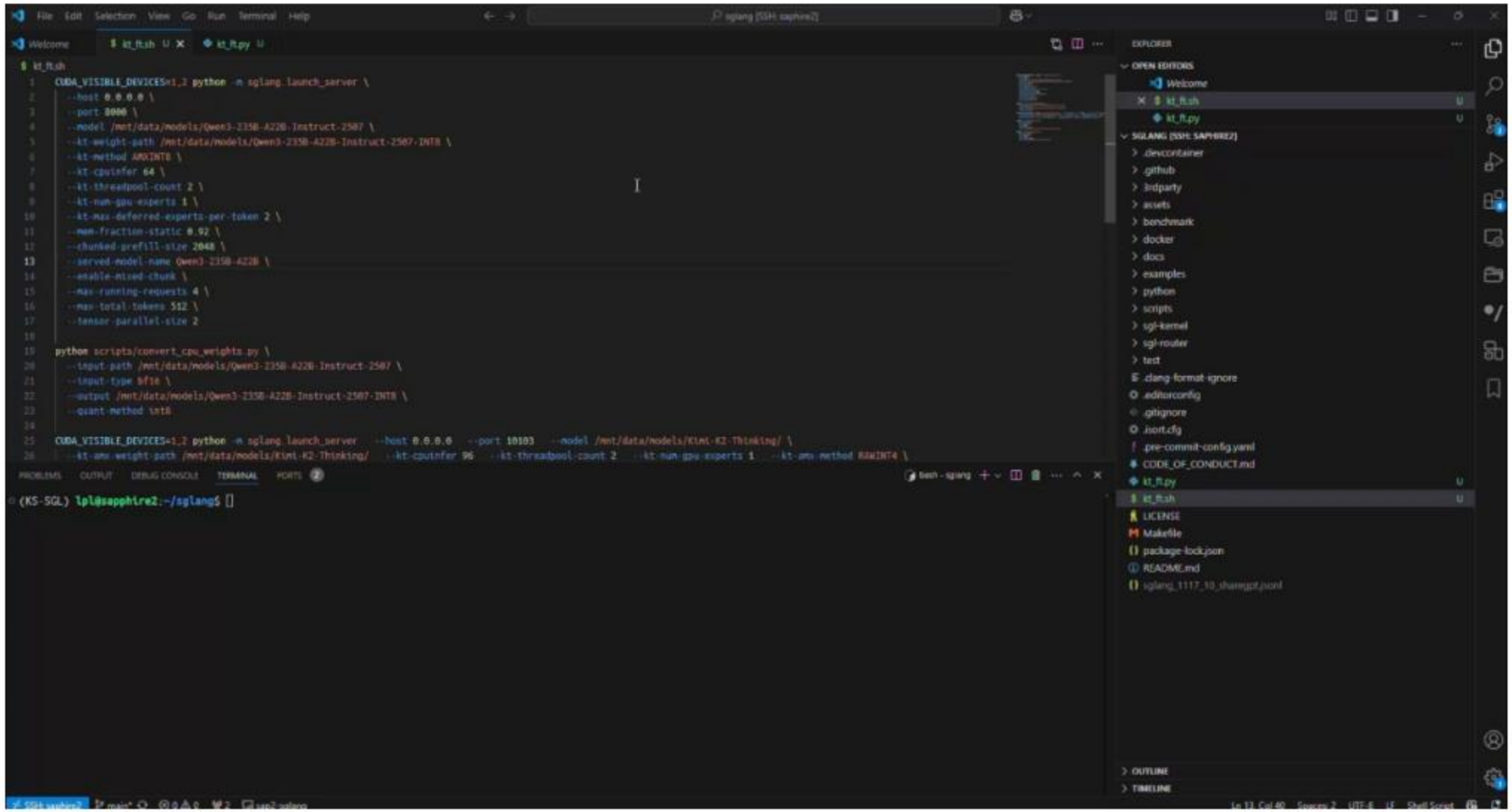


Kimi-K2-1000B *81G GPU+1.9T CPU+200G swap*

target_module: QKVO+shared experts+FFN

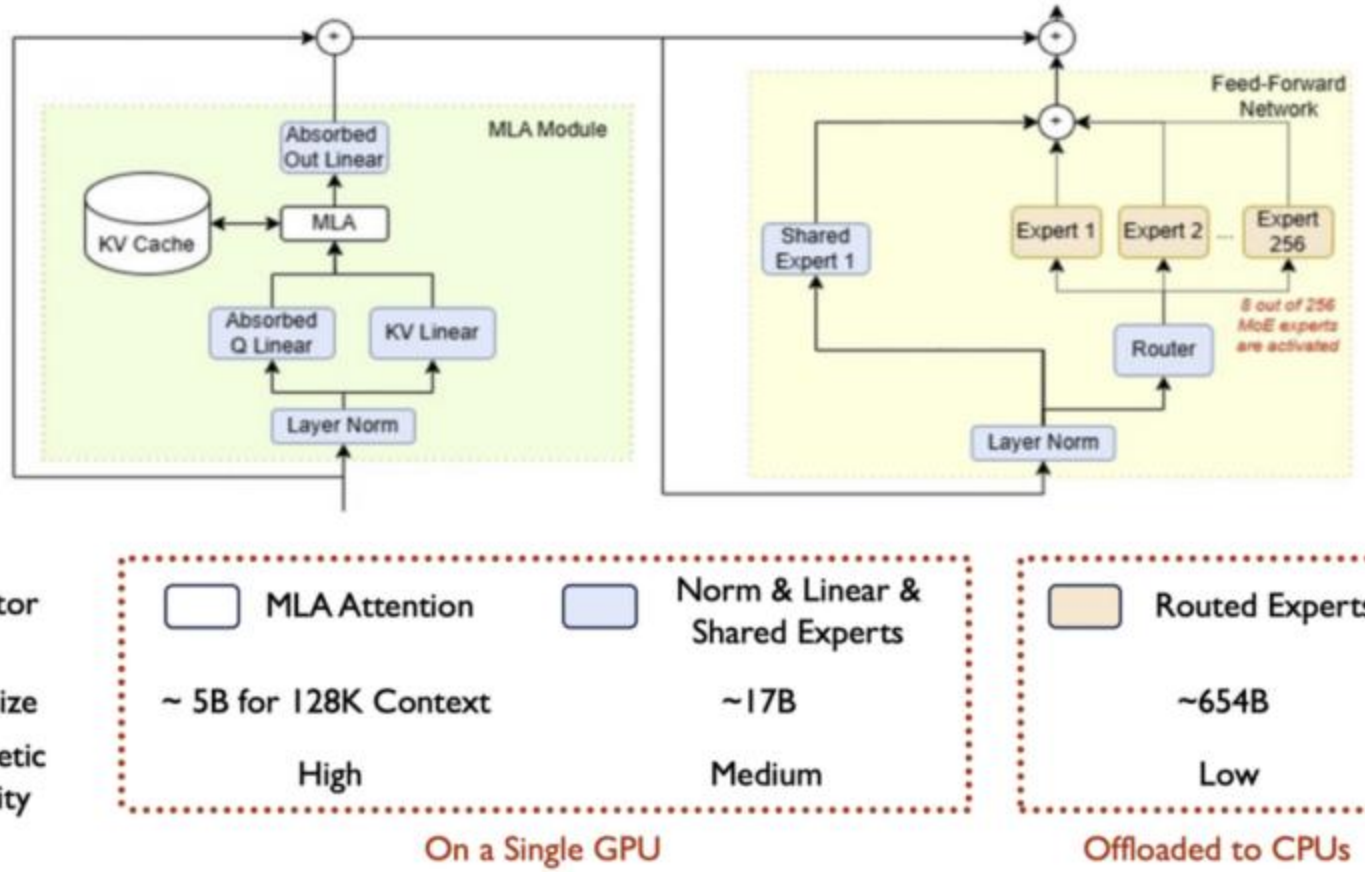
End-to-end speed: 512 * 8 / 115 = 35.6 token/s

KTransformers Inference: Demo



KTransformers Inference: Overview Framework

Motivation: more experts placed on GPUs → fewer CPU memory accesses under bandwidth bottleneck



Hybrid Expert Backend:

AMX-optimized CPU kernels + CPU/ GPU Hybrid Expert Parallelism for MoE

KTransformers + SGL

Multi-GPU Hybrid Serving Engine:

Multi-GPU Tensor Parallelism + Mixes different backends under one API

KTransformers Inference: Performance

Native KTransformers: single-GPU+CPU

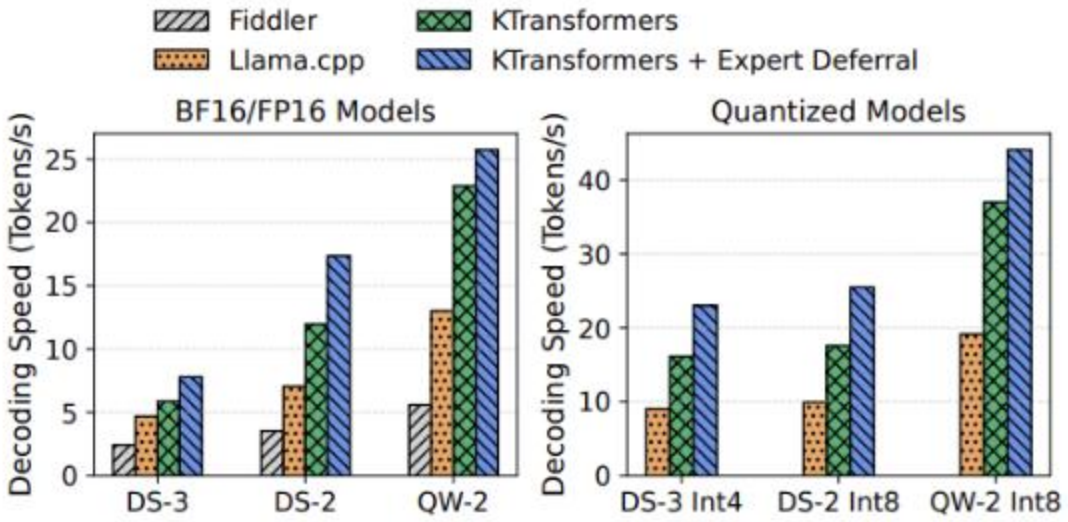
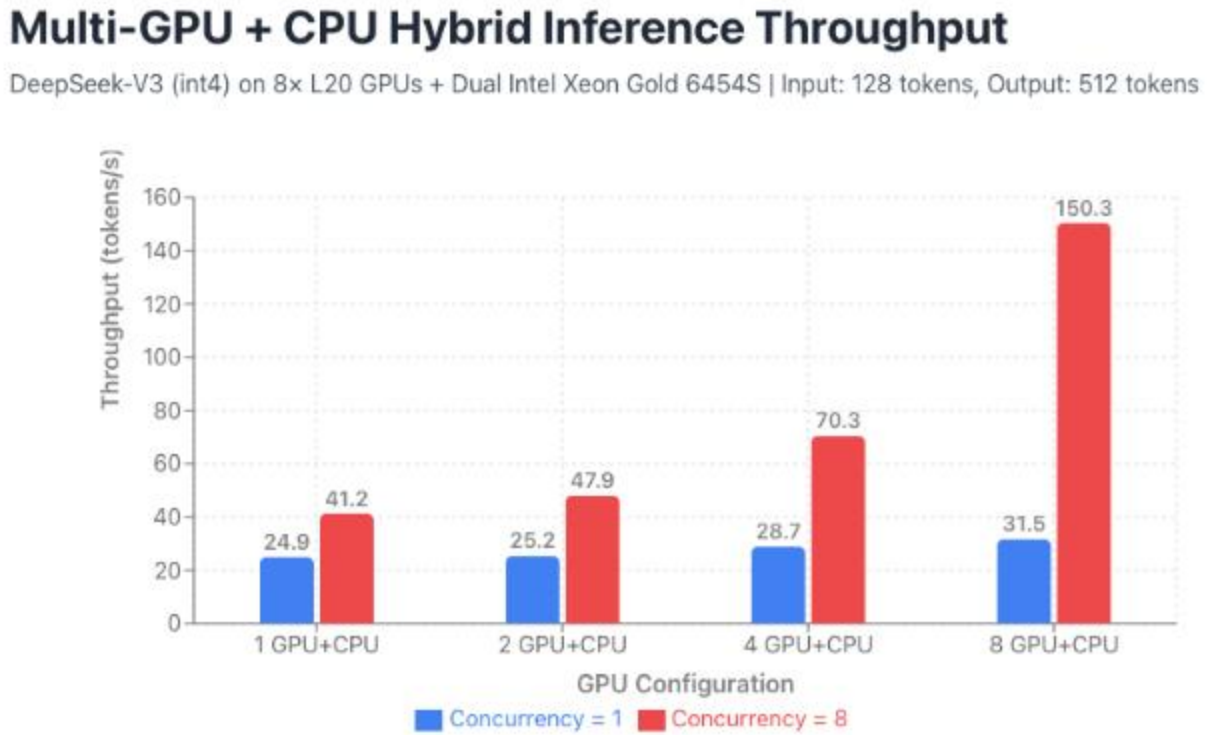


Figure 12. Comparison of decoding speed between KTRANSFORMERS and the state-of-the-art baselines.

KTransformers gains to reduced CPU/GPU coordination overhead, reaching up to **4× speedup.**

SGLang+KTransformers: multi-GPU+CPU

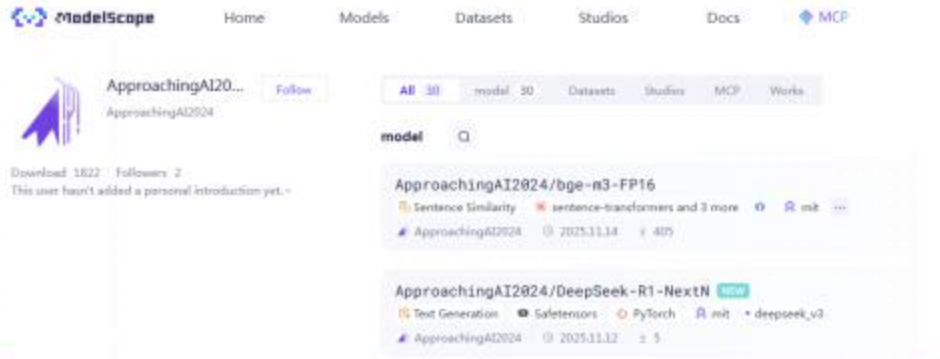


The same 8-GPU configuration achieves a **264% throughput** gain compared to 1 GPU.

KTransformers Inference: Practice Tutorial

Step1: Prepare the model weights

Method1: Download the quantized model from https://modelscope.cn/profile/ApproachingAI2024



OR

Method2: Download the origin model from Huggingface, and convert by https://github.com/kvcache-ai/ktransformers/blob/main/kt-kernel/scripts/convert_cpu_weights.py


```
CUDA_VISIBLE_DEVICES=1,2 python -m sglang.launch_server \
  --host 0.0.0.0 \
  --port 8000 \
  --model /mnt/data/models/Qwen3-235B-A22B-Instruct-2507 \
  --kt-weight-path /mnt/data/models/Qwen3-235B-A22B-Instruct-2507-INT8 \
  --kt-method AMXINT8 \
  --kt-cpuinfer 64 \
  --kt-threadpool-count 2 \
  --kt-num-gpu-experts 1 \
  --kt-max-deferred-experts-per-token 2 \
  --mem-fraction-static 0.92 \
  --chunked-prefill-size 2048 \
  --served-model-name Qwen3-235B-A22B \
  --enable-mixed-chunk \
  --max-running-requests 4 \
  --max-total-tokens 512 \
  --tensor-parallel-size 2 \
  --enable-p2p-check \
  --disable-shared-experts-fusion \
```

```
python scripts/convert_cpu_weights.py \
  --input-path /mnt/data/models/Qwen3-235B-A22B-Instruct-2507 \
  --input-type bf16 \
  --output /mnt/data/models/Qwen3-235B-A22B-Instruct-2507-INT8 \
  --quant-method int8
```

KTransformers Inference: Practice Tutorial

Step2: Launch the SGLang server

```
CUDA_VISIBLE_DEVICES=1,2 python -m sglang.launch_server \
  --host 0.0.0.0 \
  --port 8000 \
  --model /mnt/data/models/Qwen3-235B-A22B-Instruct-2507 \
  --kt-weight-path /mnt/data/models/Qwen3-235B-A22B-Instruct-2507-INT8 \
  --kt-method AMXINT8 \
  --kt-cpuinfer 64 \
  --kt-threadpool-count 2 \
  --kt-num-gpu-experts 1 \
  --kt-max-deferred-experts-per-token 2 \
```

More original SGL settings refer to https://docs.sglang.ai/references/faq.html

```
  --mem-fraction-static 0.92 \
  --chunked-prefill-size 2048 \
  --served-model-name Qwen3-235B-A22B \
  --enable-mixed-chunk \
  --max-running-requests 4 \
  --max-total-tokens 512 \
  --tensor-parallel-size 2 \
  --enable-p2p-check \
  --disable-shared-experts-fusion \
```

KT settings about run faster/ more precise:

`--kt-method`: more throughput with AMXINT4, more precise with AMXINT8; without AMX, you can choose LLAMAFILE

`--kt-cpuinfer`: More CPU cores → **higher MoE throughput**

`--kt-threadpool-count`: The number of NUMA nodes

`--kt-num-gpu-experts`: More gpu-experts → faster calculation, but **higher GPUVRAM needs**

SGL settings if you deal with CUDA OOM:

`--mem-frac...`: If OOM, small to decrease KV Cache memory

`--chunked-prefill-size`: If OOM in prefill, reduce it to 2048/4096

`--max-running-requests`: If OOM in decoding, lower it

`--max-total-tokens`: If not long prompt, lower to prevent OOM

KTransformers Inference: Practice Tutorial

After launch the sglang server, the model exposes an OpenAI-compatible Chat Completions API

Just send HTTP requests to call the server by POST:

```python
import requests

server_host_address = "http://127.0.0.1:8000"
chat_completion_endpoint = f"{server_host_address}/v1/chat/completions"

request_headers = {
    "Content-Type": "application/json"
}

request_payload = {
    "model": "Qwen3-235B-A22B",
    "messages": [
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Please introduce KTransformers."}
    ],
    "max_tokens": 256,
    "temperature": 0.7,
    "stream": False
}

http_response = requests.post(chat_completion_endpoint, headers=request_headers, json=request_payload)
http_response.raise_for_status()

response_json_content = http_response.json()
assistant_message_content = response_json_content["choices"][0]["message"]["content"]

print(assistant_message_content)
```

Use the bench_serving in sglang:

```
python -m sglang.bench_serving \
  --backend sglang \
  --host 127.0.0.1 \
  --port 30000 \
  --num-prompts 1000 \
  --model models/DeepSeek-R1-0528-GPU-weight
```

```
============ Serving Benchmark Result ============
Backend:                                 sglang
Traffic request rate:                    inf
Max request concurrency:                 not set
Successful requests:                     10
Benchmark duration (s):                  177.11
Total input tokens:                      1997
Total input text tokens:                 1997
Total input vision tokens:               0
Total generated tokens:                  2354
Total generated tokens (retokenized):    2349
Request throughput (req/s):              0.06
Input token throughput (tok/s):          11.28
Output token throughput (tok/s):         13.29
Total token throughput (tok/s):          24.57
Concurrency:                             5.86
----------------End-to-End Latency----------------
Mean E2E Latency (ms):                   103867.77
Median E2E Latency (ms):                 116207.06
---------------Time to First Token----------------
Mean TTFT (ms):                          73474.43
Median TTFT (ms):                        68896.86
P99 TTFT (ms):                           148945.34
-----Time per Output Token (excl. 1st token)------
Mean TPOT (ms):                          129.39
Median TPOT (ms):                        70.35
P99 TPOT (ms):                           513.43
---------------Inter-Token Latency----------------
Mean ITL (ms):                           129.66
Median ITL (ms):                         70.04
P95 ITL (ms):                            125.38
P99 ITL (ms):                            140.76
Max ITL (ms):                            111682.53
==================================================
```

Thanks!

趋境科技 APPROACHING.AI

kvcache.ai

KVCache.AI is a joint research project between MADSys and top industry collaborators, focusing on efficient LLM serving.

903 followers https://madsys.cs.tsinghua.edu.cn/ zhang_mingxing@mail.tsinghua.edu.cn

Pinned

Customize pins

Mooncake Public

Mooncake is the serving platform for Kimi, a leading LLM service provided by Moonshot AI.

C++ 4.1k 388

ktransformers Public

A Flexible Framework for Experiencing Cutting-edge LLM Inference Optimizations

Python 15.2k 1.1k

TrEnv-X Public

Go 58

https://github.com/kvcache-ai