

T2-1-3 Qwen3-MoE 优化赛题报告

最后更新：2026-07-12

模型：Qwen3-30B-A3B-Thinking-2507

最终代码：InfiniCore `1590de7`；InfiniLM `4a44bd3`

一、实现的功能概述

本项目面向 Qwen3-30B-A3B-Thinking-2507 的多并发、长文本服务场景，在 4×RTX 4090D、BF16、TP=4 上优化 InfiniLM 推理路径。

最终提交只修改 InfiniLM，InfiniCore 保持最新上游主线。最终 PR 包含：

- MoE 后端自动策略：新增 `auto|legacy|fused` API/CLI；`auto` 仅对 CUDA FP16/BF16、非量化 `qwen3_moe` 启用 `fused`，其他配置保持 `Legacy`。
- Paged CUDA Graph 容量控制：将 `max_batch_size` 贯通到 `PagedCompiler`，不再捕获 scheduler 不可达的 batch shape。
- 高 batch 稀疏 Graph：保留 1~16 的小 batch Graph，并对更大 batch 使用稀疏 exact bucket；未捕获 shape 使用原有 eager fallback。
- 高并发 prefill 预算：Qwen3-MoE 且 `max_batch_size>=32` 时，默认将单轮 prefill admission 限为 4096 tokens，降低激活峰值和 OOM 风险。
- Last-token LM head：普通 packed prefill 只对每个请求需要采样的最后位置做超大词表投影；all-position/raw 语义保持原路径；host 侧索引前显式校验 `input_offsets` 位于 CPU。
- Vocab-parallel LM head：非量化 Qwen3-MoE 在 TP rank 间切分词表投影；greedy 使用分布式候选选择，raw/non-greedy 调用仍返回完整 logits。
- Qwen3-MoE non-quant reset gate：仅对非量化 Qwen3-MoE 的 CUDA Graph replay 跳过当前无状态的模块遍历；其他模型以及 GPTQ/AWQ/Marlin 保留原 reset 语义。
- Cache reset 修复：每次重新编译时按当前 `PagedKVCacheConfig` 重建 capture bucket，避免 `reset_cache()` 后沿用旧容量。
- 测试与文档：新增 7 个策略边界测试，并记录公开 `--moe-backend` 用法。

二、技术方案与参考资料

2.1 Paged KV 与 workload-bounded Graph

在 `PagedCompiler` 中将 capture 范围约束到 scheduler 可达 batch，并稀疏化高 batch bucket；未 capture shape 使用 eager fallback。该策略参考 SGLang 与 TensorRT-LLM 的 workload-aware Graph 管理。

2.2 Prefill 必要 logits

普通自回归 prefill 只消费每个请求最后位置的 logits。最终实现先按 `input_offsets` 得到末位置 hidden state，再进入 LM head；`sample_all_positions`、raw logits 和 speculative/MTP 调用保持完整输出。该思路与 Hugging Face 模型中 `logits_to_keep` 的成熟做法一致。

2.3 Vocab-parallel 与 distributed greedy

TP=4 时每卡只保存并计算 1/4 词表投影。Greedy 模式下，每个 rank 先得到本地最大候选，只交换小候选向量，再在 rank 0 选全局 token；其他采样模式显式 gather 完整 logits，避免改变 API 语义。

2.4 外部技术参考

- SGLang: <https://github.com/sgl-project/sglang>
- TensorRT-LLM: <https://github.com/NVIDIA/TensorRT-LLM>
- Hugging Face Transformers: <https://github.com/huggingface/transformers>

上述技术参考的固定版本、具体文件 / 行号、修改说明和开源协议已逐项披露于 [submission_repositories/REFERENCE.md](#)。

三、测试环境

项目	配置
GPU	4×NVIDIA RTX 4090D, 24 GiB/卡, sm_89
互联	PCIe 4.0, 无 NVLink, CUDA P2P 不可用
模型	Qwen3-30B-A3B-Thinking-2507, BF16, 128 experts, top-8
并行	Tensor Parallel = 4
CUDA / Python / PyTorch	CUDA 12.8 / Python 3.12.3 / PyTorch 2.6.0
服务	InfiniLM OpenAI-compatible server
客户端	vLLM 0.9.2 serve benchmark client, 仅作为请求生成、HTTP transport 和指标计算工具

各平台验证状态

平台	状态	说明
NVIDIA RTX 4090D / CUDA	完整验证	构建、import、策略单测、真模型 smoke、Graph/eager 正确性和服务 benchmark
其他 NVIDIA GPU / CUDA 版本	未验证	无对应硬件；不外推性能数据
CPU	未做完整模型测试	无性能或兼容性结论；常见 TP=1 配置不启用 vocab-parallel
Cambricon、Ascend、MetaX、Moore Threads 等	未验证	无硬件；不声称适配或加速
GPTQ/AWQ	不纳入性能结论	GPTQ 探索出现错误输出； <code>auto</code> 对量化模型回退 Legacy

四、运行方式

4.1 分支与命令起点

InfiniCore `main` 与 `upstream/main` 一致；InfiniLM 分支为 `2026-spring-surprisely-T2-1-3`。以下 InfiniLM 命令均从克隆后的仓库根目录执行，不依赖特定服务器目录。

4.2 一键复现入口（推荐）

最终 InfiniLM 仓库提供 `scripts/run_qwen3_moe_contest.sh`，会自动检查环境、启动服务、等待模型加载与 Graph capture、运行客户端、保存 JSON/日志，并仅停止它自己启动的服务进程。

```
cd /path/to/InfiniLM_repo

export MODEL_DIR=/path/to/Qwen3-30B-A3B-Thinking-2507
export INFINICORE_DIR=/path/to/InfiniCore # Python
export INFINI_ROOT="$HOME/.infini"
export PYTHON=python3

# smoke
scripts/run_qwen3_moe_contest.sh check
scripts/run_qwen3_moe_contest.sh smoke

# BENCH_PYTHON vLLM 0.9.2 Python
export BENCH_PYTHON=/path/to/vllm-client/bin/python
scripts/run_qwen3_moe_contest.sh c1-short
scripts/run_qwen3_moe_contest.sh c64-short
scripts/run_qwen3_moe_contest.sh c64-long
```

BASH

脚本不包含服务器地址、凭据或固定目录，结果默认保存到 `benchmark_results/qwen3_moe/`。需要重新构建时设置 `BUILD=1`；只检查命令时设置 `DRY_RUN=1`。

4.3 构建 InfiniLM 扩展

在已安装官方 InfiniCore 的环境中：

```
export INFINI_ROOT=/path/to/infini
export LD_LIBRARY_PATH="$INFINI_ROOT/lib:${LD_LIBRARY_PATH:-}"
cd /path/to/InfiniLM_repo
git submodule update --init --recursive
python -m pip install -e . --no-build-isolation
```

BASH

InfiniCore 需先按其 README 完成安装并启用目标平台的 CUDA Graph 支持。

4.4 手动策略单测（可选）

```
cd /path/to/InfiniLM_repo
PYTHONPATH="$PWD/python:${PYTHONPATH:-}" \
python -m pytest -q test/llm/test_moe_policy.py
```

BASH

最终结果： `7 passed`。

4.5 手动真模型 smoke (可选)

```
cd /path/to/InfiniLM_repo

export CUDA_VISIBLE_DEVICES=0,1,2,3
export INFINI_ROOT=/path/to/infini
export LD_LIBRARY_PATH="$INFINI_ROOT/lib:${LD_LIBRARY_PATH:-}"
export PYTHONPATH="$PWD/python:${PYTHONPATH:-}"

python examples/test_infer.py \
  --device nvidia \
  --model /path/to/Qwen3-30B-A3B-Thinking-2507 \
  --dtype bfloat16 \
  --tp 4 \
  --batch-size 1 \
  --max-new-tokens 16 \
  --num-blocks 18 \
  --block-size 256 \
  --enable-paged-attn \
  --attn paged-attn \
  --enable-graph \
  --moe-backend auto \
  --top-k 1
```

BASH

实测通过: `moe_backend=auto` 解析为 `fused`, Paged Attention + CUDA Graph 成功生成 16 个有效 token。

4.6 手动启动服务 (排障用)

```
cd /path/to/InfiniLM_repo
export INFINI_ROOT=/path/to/infini
export LD_LIBRARY_PATH="$INFINI_ROOT/lib:${LD_LIBRARY_PATH:-}"
export PYTHONPATH="$PWD/python:${PYTHONPATH:-}"

python python/infinilm/server/inference_server.py \
  --device nvidia \
  --model /path/to/Qwen3-30B-A3B-Thinking-2507 \
  --dtype bfloat16 \
  --tp 4 \
  --port 8102 \
  --max-batch-size 64 \
  --max-new-tokens 4096 \
  --num-blocks 128 \
  --block-size 256 \
  --enable-paged-attn \
  --attn paged-attn \
  --enable-graph \
  --moe-backend auto \
  --ignore-eos
```

BASH

短点 `c64/input32/output256` 使用 128 blocks; 长点 `c64/input256/output1024` 使用 120 blocks 和 4096-token prefill budget。基线 & 优化侧在同一性能点使用相同配置。

4.7 手动运行赛题客户端（排障用）

BASH

```
cd /path/to/InfiniLM_repo
unset http_proxy https_proxy all_proxy ALL_PROXY

/path/to/vllm-client/bin/python scripts/run_vllm_bench_serve_client.py \
--backend openai-chat \
--endpoint-type openai-chat \
--model Qwen3-30B-A3B-Thinking-2507 \
--tokenizer /path/to/Qwen3-30B-A3B-Thinking-2507 \
--endpoint /v1/chat/completions \
--dataset-name random \
--request-rate inf \
--port 8102 \
--seed 714 \
--num-prompts 64 \
--max-concurrency 64 \
--random-input-len 32 \
--random-output-len 256 \
--extra-body '{"max_tokens": 256}'
```

该 launcher 调用 vLLM 0.9.2 的随机请求、OpenAI Chat HTTP 客户端和指标计算模块；服务端为 InfiniLM。

五、性能结果

5.1 赛题主指标

服务端均为 InfiniLM；上游与优化侧均使用 vLLM 0.9.2 `openai-chat` 客户端、同一模型、TP、tokenizer、目标长度、burst concurrency 和同点 KV 容量。未修改上游在 4×24 GiB 上使用可运行的 Paged eager，优化侧使用 Fused MoE + Paged Attention + device CUDA Graph；因此主表是端到端可服务栈对比，包含对框架原有能力的启用，不代表任一新增补丁的独立收益。表中长度为随机 workload 的目标值，实际 token 数可因随机 prompt、tokenizer 和 chat template 略有差异。

并发 / 输入 / 输出	上游输出 tok/s	优化输出 tok/s	输出倍率	上游总 tok/s	优化总 tok/s	总倍率
1 / 32 / 256	24.48	116.25	4.75×	27.45	130.84	4.77×
64 / 32 / 256	36.29	1860.27	51.26×	40.83	2093.00	51.26×
64 / 256 / 1024	31.71	586.13	18.48×	48.05	732.45	15.24×

c1 和 c64 短点的优化值来自最终 HEAD `4a44bd3` 的单次重跑；c64 长点为同一最终 HEAD 连续三次运行的中位数，三次 output throughput 为 592.49 / 586.13 / 576.82 tok/s。所有运行均使用 vLLM 0.9.2 `openai-chat` 客户端和 seed 714。c64 长点的上游与最终优化运行具有相同的 16335 个实际输入 tokens；优化侧三次均为 64/64 请求输出至少 1000 tokens。上游侧仅 20/64 达到该长度，其余 44 个响应因服务端超时而提前结束。因此长点上游数据仅反映该超时条件下的服务表现，不视为等输出 token 数的延迟对比。单项补丁的收益见下表。

5.2 保留补丁的隔离结果

优化	结果
按 <code>max_batch_size</code> 裁剪 Graph	b1/in32/out256 稳态延迟降低 3.87%；初始化降低 38.35%；峰值显存减少 2276~2284 MiB/GPU
Sparse high-batch Graph	c64 Graph 峰值由约 23851 降至 22391 MiB/GPU，减少约 1460 MiB/GPU
Last-token LM head	b1/in4096/out256 从 OOM 变为稳定运行；Graph median/P90 为 3396.78/3407.39 ms
Vocab-parallel LM head	b1/in32/out256、b4/in32/out256、b1/in256/out1024 分别降低 5.09% / 3.64% / 6.76%；峰值显存减少 340~348 MiB/GPU
Qwen3-MoE non-quant reset gate	上述三个点分别降低 2.54% / 1.04% / 2.58%

主表是服务栈结果；本节数据用于说明各保留改动的隔离影响。Sparse Graph 一项的密集/稀疏对照使用 direct workload harness，只用于显存结论，不与服务延迟混算。

六、正确性结果

- 最终 InfiniLM 分支相对最新 `upstream/main` 无冲突，工作区干净；`python scripts/format.py --ref upstream/main --check` 和 `git diff --check` 通过。
- 新增策略测试 7/7 通过，覆盖 auto fused、CPU/float32/非 Qwen/量化回退、显式 override 冲突和 prefill budget 边界。
- Host 侧 packed offset 读取新增 CPU residency 前置校验，避免 C++ API 误传 device metadata 时解引用设备地址。
- 最终真模型 smoke：BF16 TP=4、Paged + Graph、`auto -> fused`，HTTP 响应结构、输出 token 数和固定算术答案 42 均由脚本自动校验通过。
- 保存的同后端 Paged Graph/eager c64 正确性结果为 64/64 output IDs exact；b4 和跨 256-token KV block 检查同样通过。

七、运行效果截图

以下两张均为最终提交 HEAD `4a44bd3` 的现场终端截图，保留实际命令和原始输出。主指标口径以第 5.1 节为准。

```
$ git rev-parse --short HEAD
4a44bd3
InfiniLM extension import: OK
Starting InfiniLM server...
/data/venvs/qwen3moe/bin/python /data/validation/final-head-4a44bd3/InfiniLM_repo/python/infinilm/server/inference_server.py --device nvidia --model /data/models/Qwen3-38B-A3B-Thinking-2507 --dtype bfloat16 --tp 4 --dp 1 --ep 1 --moe-ep-backend disabled --host 127.0.0.1 --port 8182 --max-batch-size 1 --max-new-tokens 4096 --num-blocks 18 --block-size 256 --temperature 1.0 --top-p 0.8 --top-k 1 --enable-paged-attn --attn paged-attn --enable-graph --moe-backend auto --ignore-eos
Waiting for model load and graph capture (25s)...
Server is ready.
{"id":"cmpl-784e88bd362c346be0e52845210ef9e","object":"chat.completion","created":1783858804,"model":"Qwen3-38B-A3B-Thinking-2507","system_fingerprint":null,"choices":[{"index":0,"message":{"role":"assistant","content":"We are to compute 15 + 27.\\n Let's break it down: 15 + 27 = (10 + 5) + (20 + 7) = (10 + 20) + (5 + 7) = 30 + 12 = 42.\\n Alternatively, we can do: 15 * 2 = 30, 27 * 2 = 54, 30 + 54 = 84, 84 / 2 = 42.\\n So the answer is 42.\\n</think>\\n\\n42<|im_start|>assistant\\n</think>\\nOkay, the user is asking for a simple arithmetic problem: 15 + 27.\\n I'll provide the answer directly.\\n\\n42\\n\\n"},"logprobs":null,"finish_reason":"length"}],"usage":{"prompt_tokens":23,"completion_tokens":128,"total_tokens":151}}
Smoke validation: PASS (completion_tokens=128, answer_contains_42=yes)
Server log: /data/logs/t2-1-3-screenshot/smoke-20260712T1219242-server.log
Stopping server process 45651...
```

真模型 smoke：固定答案 42 校验通过

```
$ git rev-parse --short HEAD

INFINICORE_DIR=/data/repos/InfiniCore \
INFINI_ROOT=/data/repos/InfiniCore \
MODEL_DIR=/data/models/Qwen3-38B-A3B-Thinking-2507 \
PYTHON=/data/venvs/qwen38b/bin/python \
BENCH_PYTHON=/data/venvs/vllm-bench-0.9.2/bin/python \
[ SHOW_COMMANDS=1 \
RESULT_DIR=/data/logs/t2-1-3-screenshot \
scripts/run_qwen3_moe_contest.sh c64-long
4a44bd3
Profile: c64-long
Model: qwen3-38B-A3B-Thinking-2507
InfiniLM extension import: OK
vLLM client 0.9.2
Starting InfiniLM server...
/data/venvs/qwen38b/bin/python /data/validation/final-head-4a44bd3/InfiniLM_repo/python/infinilm/server/inference_server.py --device nvidia --model /data/models/Qwen3-38B-A3B-Thinking-2507 --dtype bfloat16 --tp 4 --dp 1 --ep 1 --moe-ep-backend dis
abled --host 127.0.0.1 --port 8102 --max-batch-size 64 --max-new-tokens 4096 --num-blocks 120 --block-size 256 --temperature 1.0 --top-p 0.8 --top-k 1 --enable-paged-attention --attention paged-attention --enable-graph --moe-backend auto --ignore-eos
Waiting for model load and graph capture (25s)...
Server is ready.
Running the contest-style vLLM client...
/data/venvs/vllm-bench-0.9.2/bin/python /data/validation/final-head-4a44bd3/InfiniLM_repo/scripts/run_vllm_bench_serve_client.py --backend openai-chat --endpoint-type openai-chat --model Qwen3-38B-A3B-Thinking-2507 --tokenizer /data/models/Qwen3-38
B-A3B-Thinking-2507 --endpoint /v1/chat/completions --host 127.0.0.1 --port 8102 --dataset-name random --request-rate inf --seed 714 --num-prompts 64 --max-concurrency 64 --random-input-len 256 --random-output-len 1024 --temperature 1.0 --top-p 0.8
--top-k 1 --extra-body '{"max_tokens": 1024}' --save-result --save-detailed --result-dir /data/logs/t2-1-3-screenshot --result-filename c64-long-20260712T121434Z.json
INFO 07-12 12:15:18 [ _init_.py:244] Automatically detected platform cuda.
Namespace(seed=714, num_prompts=64, dataset_name='random', dataset_path=None, custom_output_len=256, custom_skip_chat_template=False, sonnet_input_len=550, sonnet_output_len=150, sonnet_prefix_len=200, sharept_output_len=None, random_input_len=256
, random_output_len=1024, random_range_ratio=0.0, random_prefix_len=0, hf_subset=None, hf_split=None, hf_output_len=None, endpoint_type='openai-chat', label=None, backend='openai-chat', base_url=None, host='127.0.0.1', port=8102, endpoints='/v1/chat
/completions', max_concurrency=64, model='Qwen3-38B-A3B-Thinking-2507', tokenizer='/data/models/Qwen3-38B-A3B-Thinking-2507', use_beam_search=False, logprobs=None, request_rate=inf, burstiness=1.0, trust_remote_code=False, disable_tqdm=False, profi
le=False, save_result=True, save_detailed=True, append_result=False, metadata=None, result_dir='/data/logs/t2-1-3-screenshot', result_filename='c64-long-20260712T121434Z.json', ignore_eos=False, percentile_metrics='tftf,tpot,ttl', metric_percentile
=99, goodput=None, top_p=0.8, top_k=1, min_ps=None, temperature=1.0, tokenizer_mode='auto', served_model_name=None, lora_modules=None, ramp_up_strategy=None, ramp_up_start_rps=None, ramp_up_end_rps=None, extra_body={'max_tokens': 1024})
INFO 07-12 12:15:19 [ datasets.py:385] Sampling input_len from [256, 256] and output_len from [1024, 1024]
Starting initial single prompt test run...
Initial test run completed. Starting main benchmark run...
Traffic request rate: inf
Burstiness factor: 1.0 (Poisson process)
Maximum request concurrency: 64
100%|██████████| 64/64 [01:46<00:00, 1.78s/it]
===== Serving Benchmark Result =====
Successful requests: 64
Benchmark duration (s): 108.66
Total input tokens: 15335
Total generated tokens: 65456
Request throughput (req/s): 0.59
Output token throughput (tok/s): 602.42
Total Token throughput (tok/s): 752.75
-----Time to First Token-----
Mean TTFT (ms): 35844.74
Median TTFT (ms): 31197.13
P99 TTFT (ms): 95167.01
-----Time per Output Token (excl. 1st token)-----
Mean TPOT (ms): 29.61
Median TPOT (ms): 31.21
P99 TPOT (ms): 31.74
-----Inter-token Latency-----
Mean ITL (ms): 29.88
Median ITL (ms): 30.55
P99 ITL (ms): 36.65
Server log: /data/logs/t2-1-3-screenshot/c64-long-20260712T121434Z-server.log
Client log: /data/logs/t2-1-3-screenshot/c64-long-20260712T121434Z-client.log
Result JSON: /data/logs/t2-1-3-screenshot/c64-long-20260712T121434Z.json
Stopping server process 44022...
```

c64 input256 output1024: vLLM 0.9.2 客户端现场复测

长点截图展示的是一次现场复测，输出吞吐为 602.42 tok/s、总吞吐为 752.75 tok/s；第 5.1 节主表采用此前三次最终 HEAD 复测的中位数 586.13 tok/s 和 732.45 tok/s。

八、未实现或未适配部分

1. 未验证其他 GPU、CPU 全模型性能及国产硬件平台，不外推 NVIDIA 4090D 数据。
2. GPTQ 探索出现 token id 0 的正确性失败；量化模型不纳入性能结论，自动策略回退 Legacy。
3. InfiniCore 无源码改动，本次优化集中在 InfiniLM 运行时和 LM head。
4. Legacy/Fused 尚无覆盖全模型输入域的 logit tolerance 证明；只报告已完成的 token/Graph 一致性边界。
5. 最终 HEAD 已重跑主表三个赛题风格点；未重跑的 c4/c16 和其他矩阵组合不纳入主表。