DeepSeek-V3.2-Exp高吞吐优化实践：基于CANN的全链路优化

作者：刁莹煜

时间：2025.11.15

CANN

News > DeepSeek-V3.2-Exp Release 2025/09/29

Introducing DeepSeek-V3.2-Exp

🚀 Introducing DeepSeek-V3.2-Exp — our latest experimental model!

✨ Built on V3.1-Terminus, it debuts DeepSeek Sparse Attention (DSA) for faster, more efficient training & inference on long context.

👉 Now live on App, Web, and API

💰 API prices cut by 50%+!

> **昇腾0Day适配和参考实践**

昇腾在DeepSeek-V3.2-Exp一发布开源即实现了DeepSeek-V3.2-Exp BF16模型部署，并在CANN平台上完成对应的优化适配，整体部署策略沿用DeepSeek的大EP并行方案，针对稀疏DSA结构，叠加实现长序列亲和的CP并行策略，兼顾时延和吞吐，在128K长序列下能够保持TTFT低于2秒、TPOT低于30毫秒的推理生成速度。

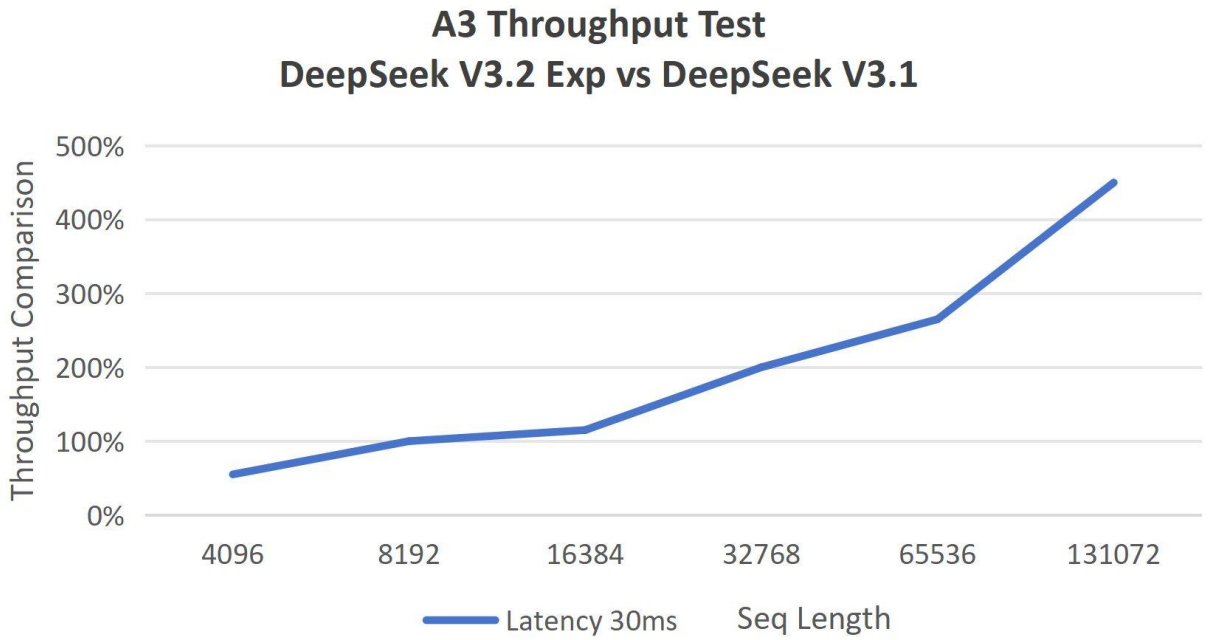
NPU DeepSeek-V3.2-Exp推理优化实践：
https://gitcode.com/cann/cann-recipes-infer/blob/master/docs/models/deepseek-v3.2-exp/deepseek_v3.2_exp_inference_guide.md

- DeepSeek 2025.9.29发布了V3.2-Exp模型，通过DSA稀疏Attention加速长序列推理，降低推理成本

- 本实践0day实现了NPU DeepSeek-V3.2-Exp BF16模型部署，2week完成了W8A8C8模型优化

CANN

整网性能Benchmark



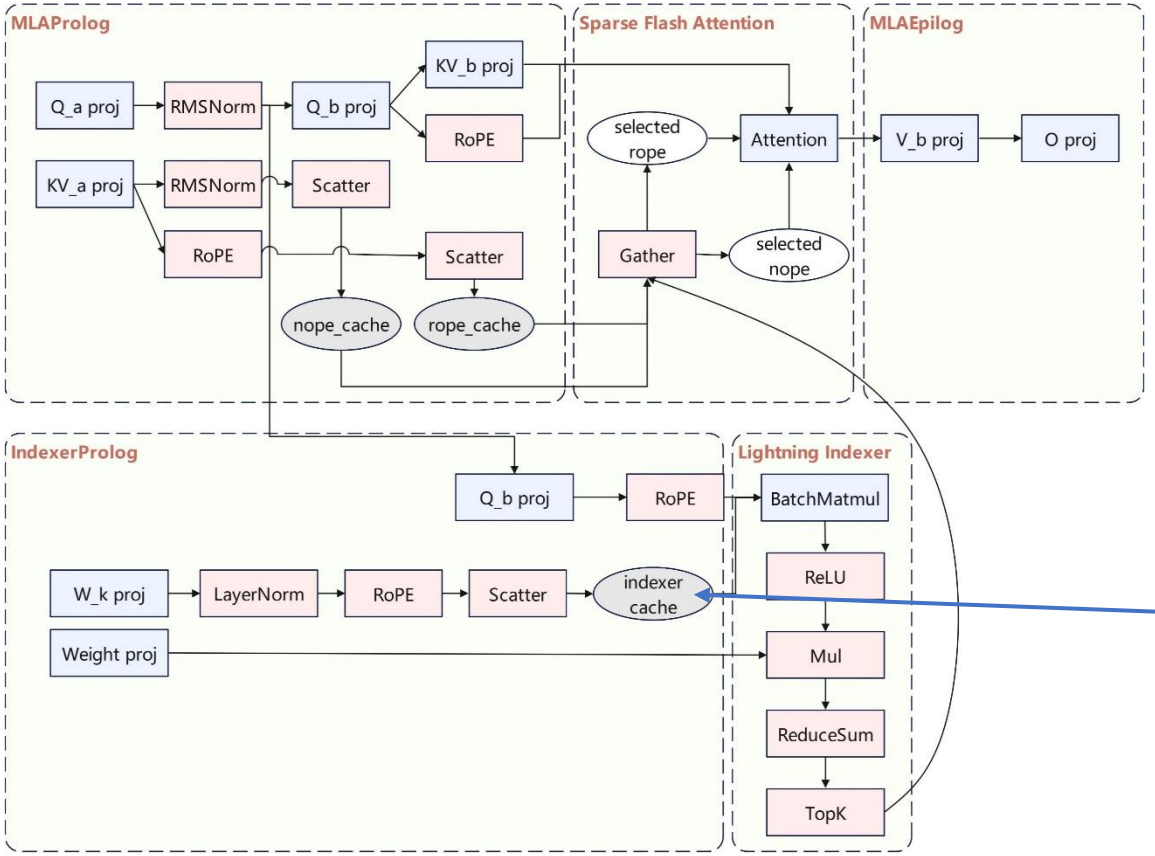
随着序列长度增加，DeepSeek V3.2 Exp 的性能优势逐步扩大，

当序列长度达到 128K 时，其吞吐达到 DeepSeek V3.1 的 450%。

目录

DeepSeek V3.2 Exp模型解析—结构

相较DeepSeek-V3.1，DeepSeek-V3.2-Exp主要新增了**Lightning Indexer**稀疏模块，旨在从长序列中稀疏选择TopK个Token用于计算注意力



https://gitcode.com/cann

- **Attention计算量**

相较DeepSeek-V3.1，MLA的计算复杂度从O(L^2)下降到O(Lk)；Indexer部分的计算复杂度仍为O(L^2)，但是相较原始MLA，其head_num、head_dim较小，且节约了P@V的矩阵乘法

- **权重内存**

相较DeepSeek-V3.1，主要新增了Indexer模块中的Q_b proj、W_k proj和Weight proj三个Linear层，参数量新增0.85B左右，较为轻量

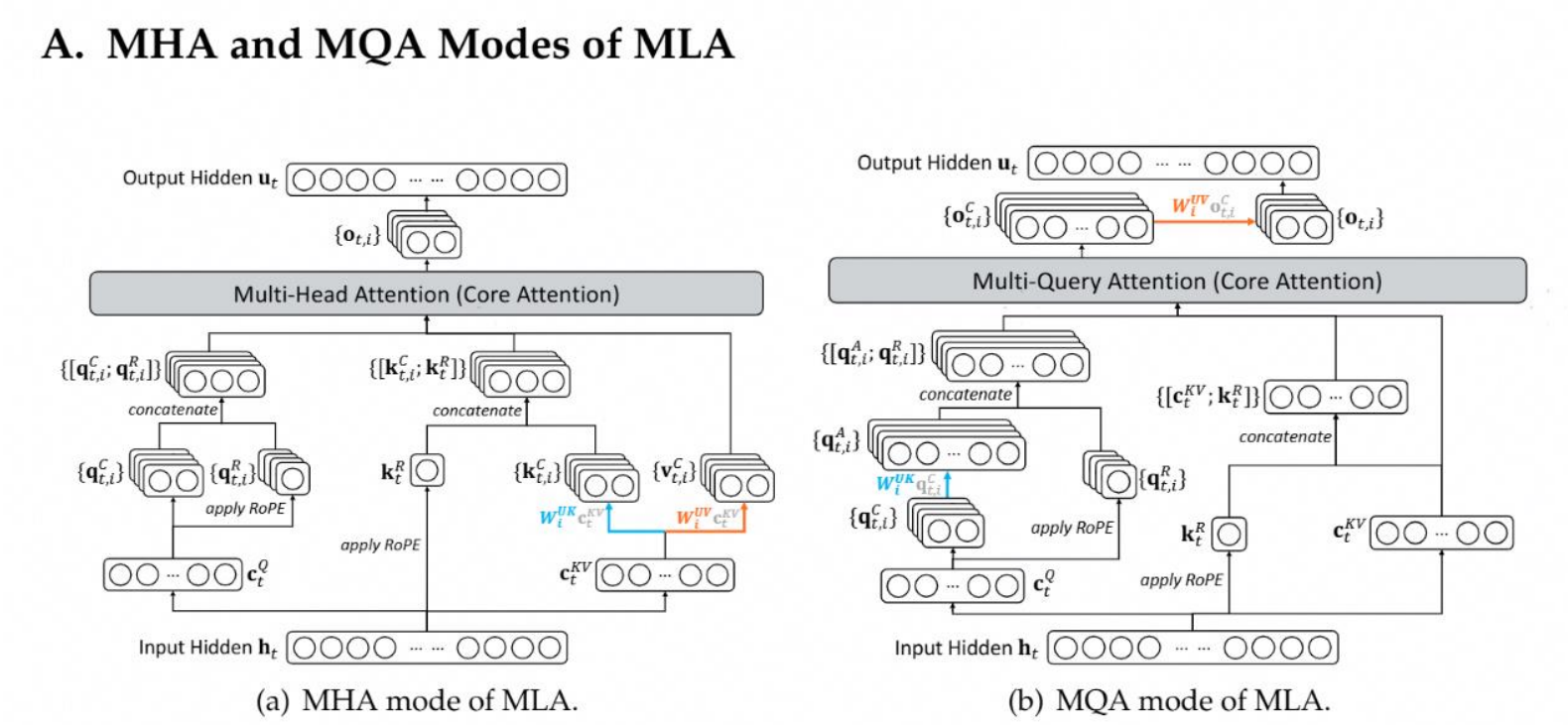
- **KVCache内存**

为了保障模型效果，DeepSeek-V3.2-Exp仍然缓存了完整MLA的Full KVCache。与此同时，为了提高推理效率，避免decode阶段重复计算Indexer Key，需要**新增缓存Indexer Key Cache**

以每个rank处理4batch 64K序列长度为例，BF16场景新增的Indexer Key Cache约为4GB左右，C8场景约2GB

CANN

DeepSeek V3.2 Exp模型解析—Prefill MLA计算流



- 在DeepSeek V3.1部署时，通常Prefill使用Naive (MHA)模式部署，Decode阶段使用Absorb (MQA)模式进行部署
- 在DeepSeek V3.2 Exp稀疏Attention模型部署时，本实践Prefill和Decode均选用Absorb (MQA)模式部署，计算流归一

CANN

DeepSeek V3.2 Exp模型解析—Prefill MLA计算流

方案一： MLA Naive(MHA) + Sparse Mask

方案一	Batch	M	K	N
BMM1(Q*K^T)	1*128	64K	192	64K
BMM2(P*V)	1*128	64K	64K	128

性能评估：方案一计算量和原始的Full Attention一致，但是无法拿到DSA的稀疏计算收益，长序列场景下性能不佳。

方案二： MLA Naive(MHA) + Sparse Attention

方案二	Batch	M	K	N
BMM1(Q*K^T)	1*64K*128	1	192	2048
BMM2(P*V)	1*64K*128	1	2048	128

性能评估：方案二的优点在于BMM的计算量较小，相对原始的Full Attention计算量减小了64K/2048=32倍，但是存在以下问题：

- BMM的M轴为1，矩阵乘法计算效率较低
- BMM的HBM访存量相较原始的Full Attention激增2048倍，将会面临访存瓶颈

方案三： MLA Absorb(MQA) + Sparse Attention

方案三	Batch	M	K	N
BMM1(Q*K^T)	1*64K	128	576	2048
BMM2(P*V)	1*64K	128	2048	512

性能评估：

- 方案三的计算量增加了3倍左右，但其BMM的M轴为128，对于矩阵乘法更为友好
- 方案三的HBM访存量相对方案二降低几十倍，访存耗时更低

结论：

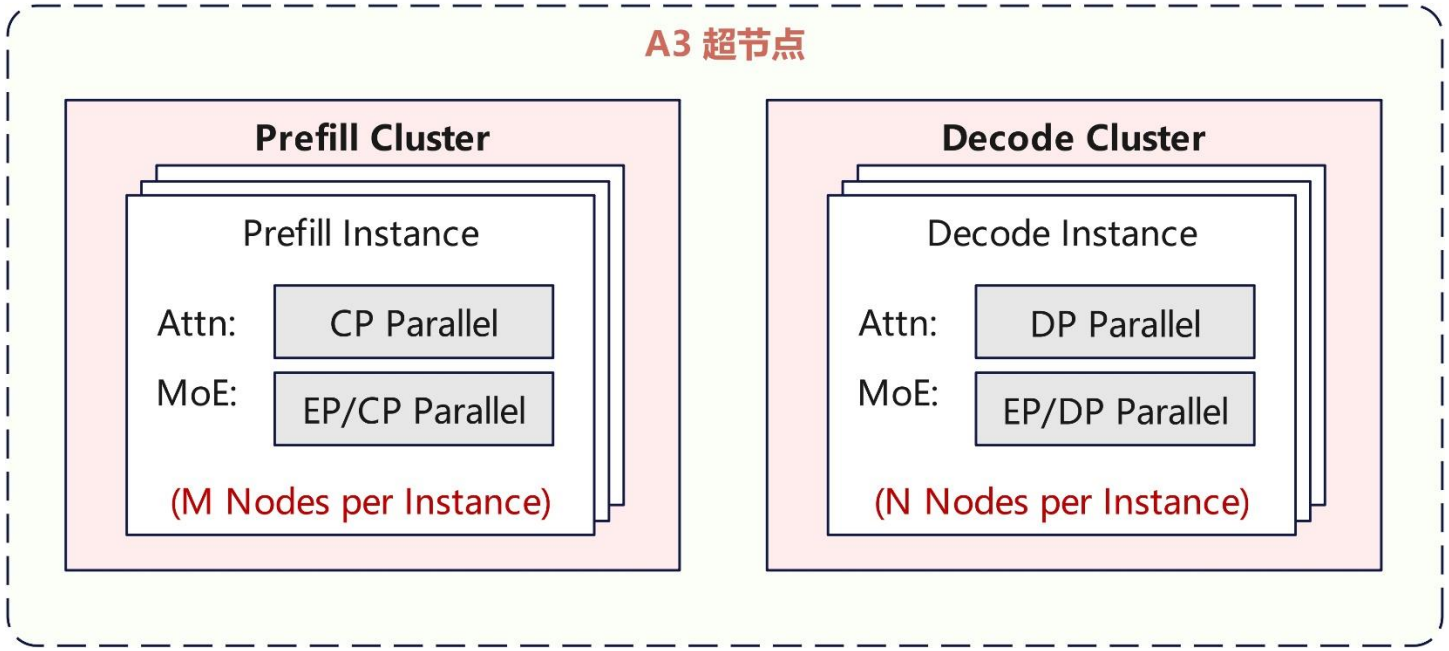
综合考虑计算和访存耗时，以及长序列应用场景，本实践选择基于方案三(MLA Absorb + Sparse Attention)来完成Prefill部署，从而Prefill和Decode的MLA计算流可以归一。

目录

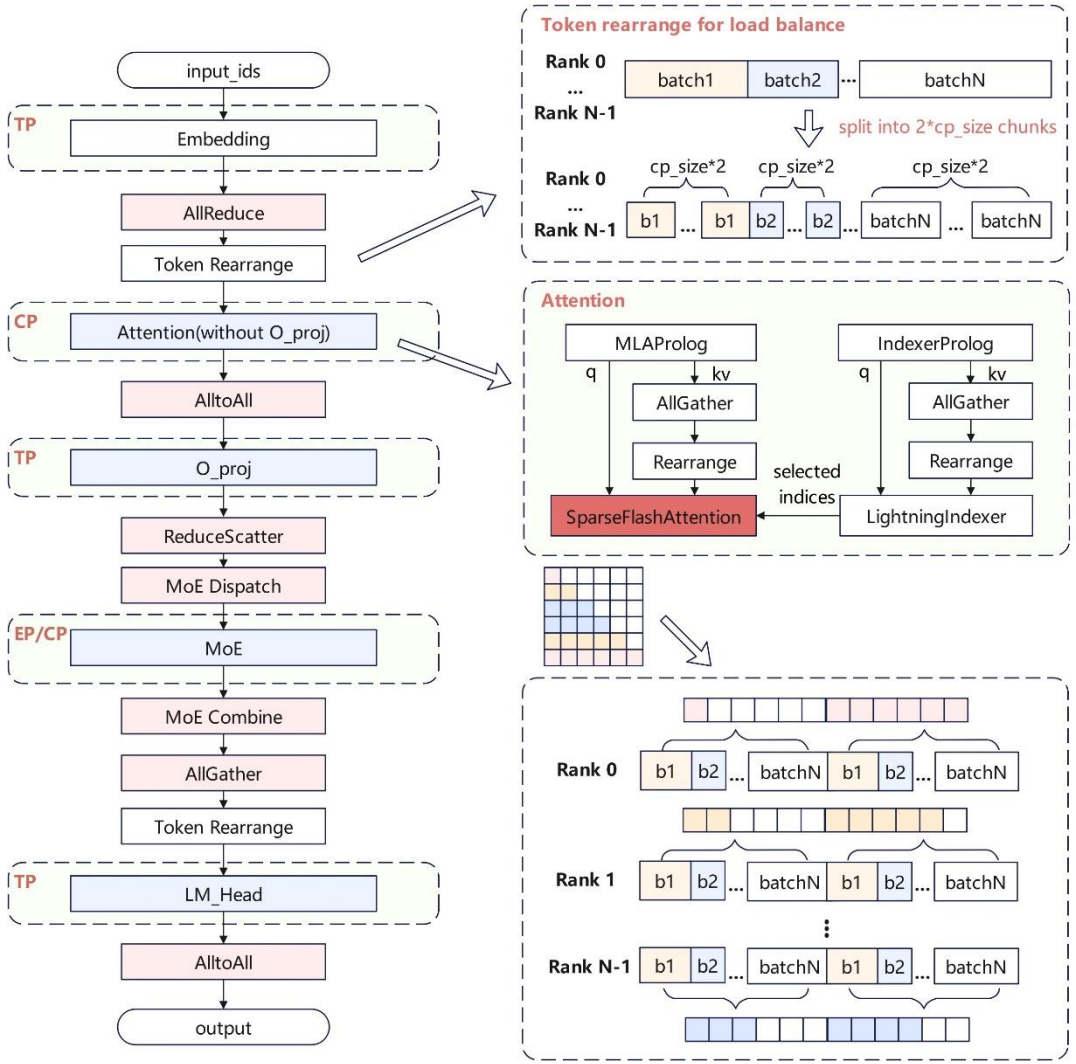
基于A3集群的整网优化方案解析—并行策略

A3集群推荐部署策略如下图所示，Prefill使用M个节点部署，Decode使用N个节点部署，每个节点包含8卡。

推荐根据资源数量、SLA等约束， BF16场景 M在4~8，N在4~16内动态调整；W8A8C8场景 M在2~8，N在2~16内动态调整。



基于A3集群的整网优化方案解析—Prefill并行策略



https://gitcode.com/cann

Q：为什么Attention模块不使用Tensor Parallel?

A：DSA的Indexer模块针对head轴做了Reduce Sum操作，TP切分会产生大量的通信开销，影响整网性能；并且Prefill MLA使用了Absorb形式，kv head num为1，不适合TP切分

Q：为什么Attention模块不使用Data Parallel?

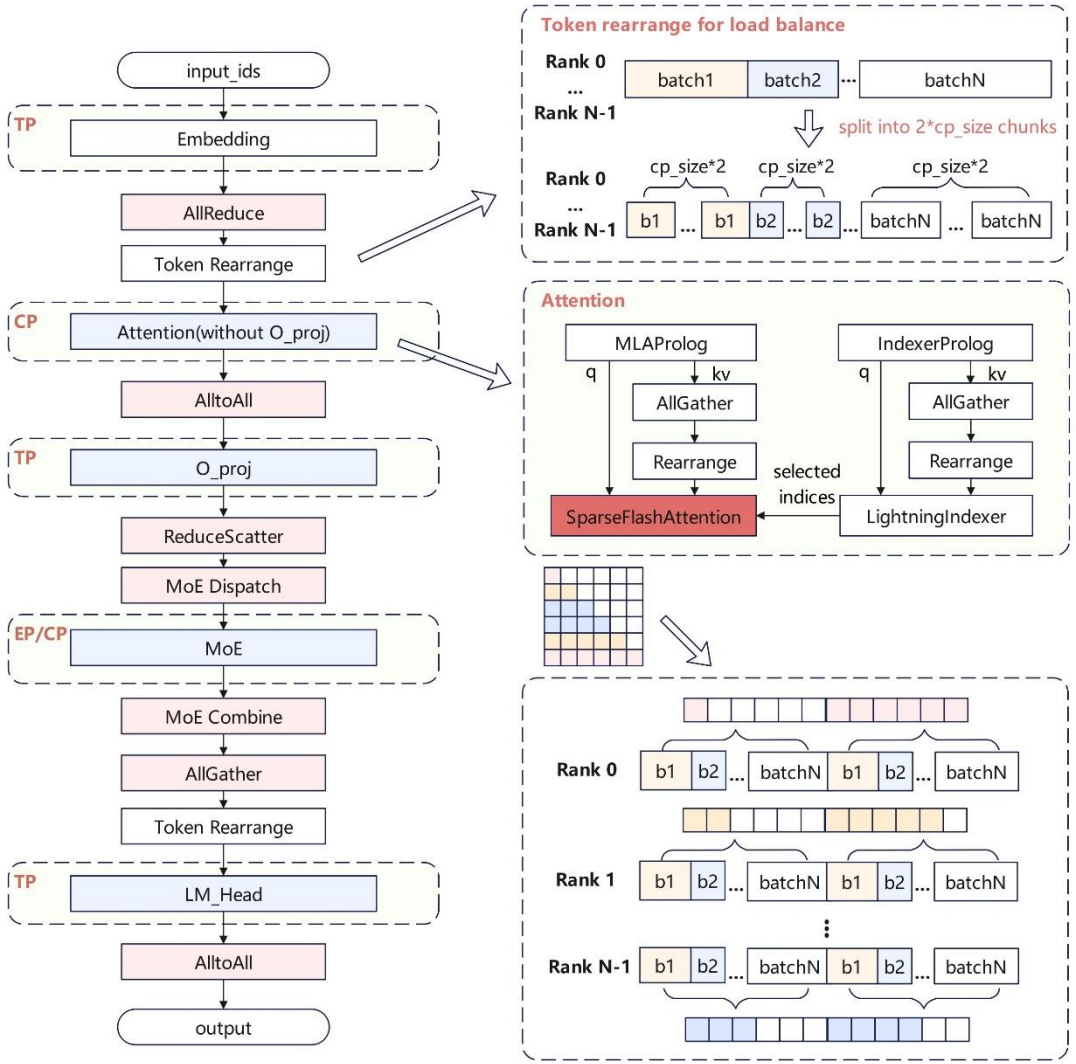
A：DP每个rank都需要推理不同的超长序列，计算量和内存较大，TTFT较高，用户体验较差

Q：为什么Attention模块选用Context Parallel?

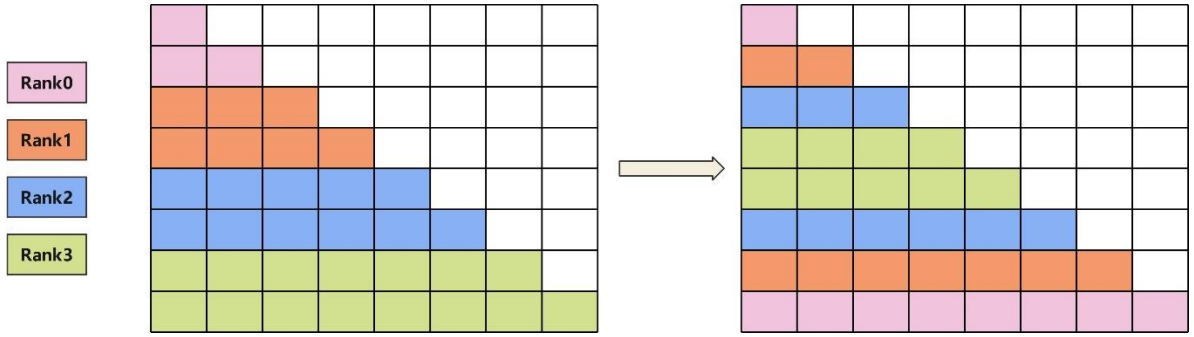
A：CP可以让多个rank均摊长序列请求的计算，单rank的计算量和activation内存都较小，TTFT较为可控，用户体验更好；但需要注意不同rank的负载均衡处理

CANN

基于A3集群的整网优化方案解析—Prefill并行策略

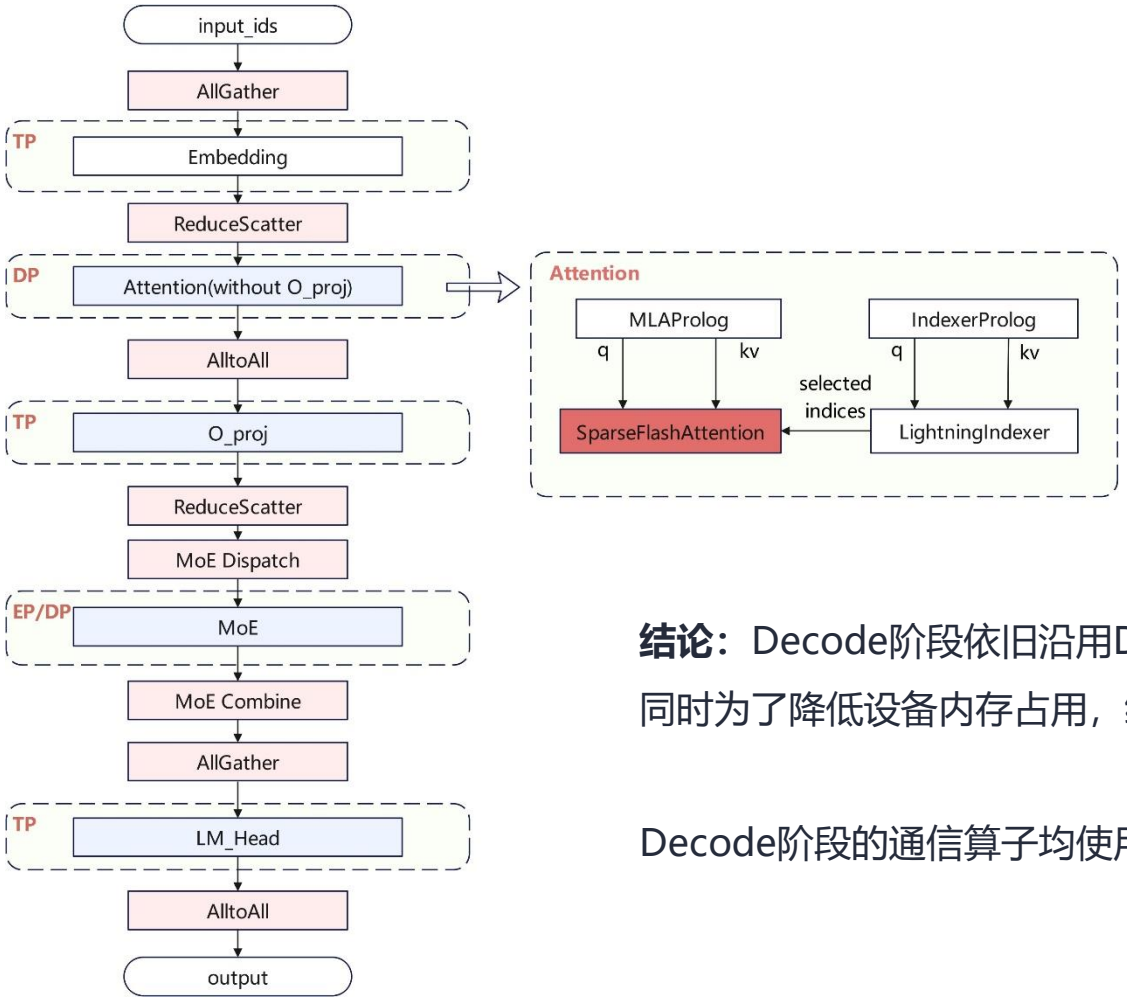


CP负载均衡



结 论： DeepSeek-V3.2-Exp 模 型 Prefill Attention 选 用 Context Parallel(CP)并行，MoE模块则沿用DeepSeek-V3.1的EP并行，兼顾吞吐与时延。

基于A3集群的整网优化方案解析—Decode并行策略



结论：Decode阶段依旧沿用DeepSeek-V3.1的部署策略，选用Attention DP + MoE EP部署

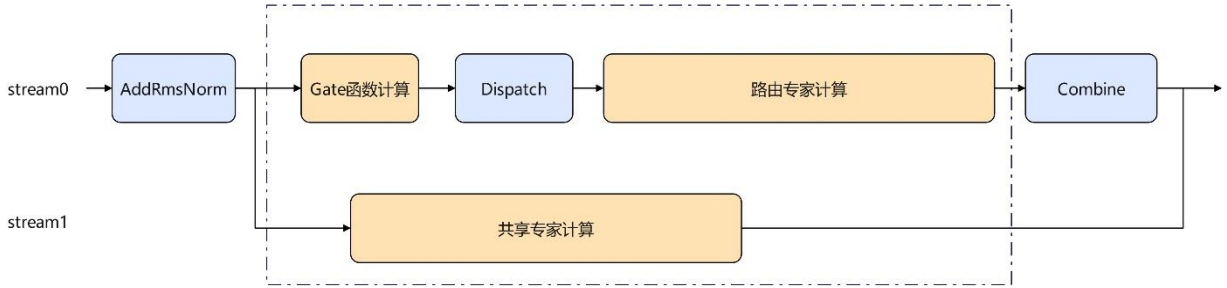
同时为了降低设备内存占用，缓解权重访存瓶颈， O_proj/LM_Head/Embedding使用局部TP切分

Decode阶段的通信算子均使用AIV通信

基于A3集群的整网优化方案解析—多流并行

CANN提供了多流并行的机制，可支持计算/通信并行等。由于A3芯片为CV分离架构，也可支持Cube/Vector并行。MLAProlog/IndexerProlog/MOE模块均可使用多流并行加速

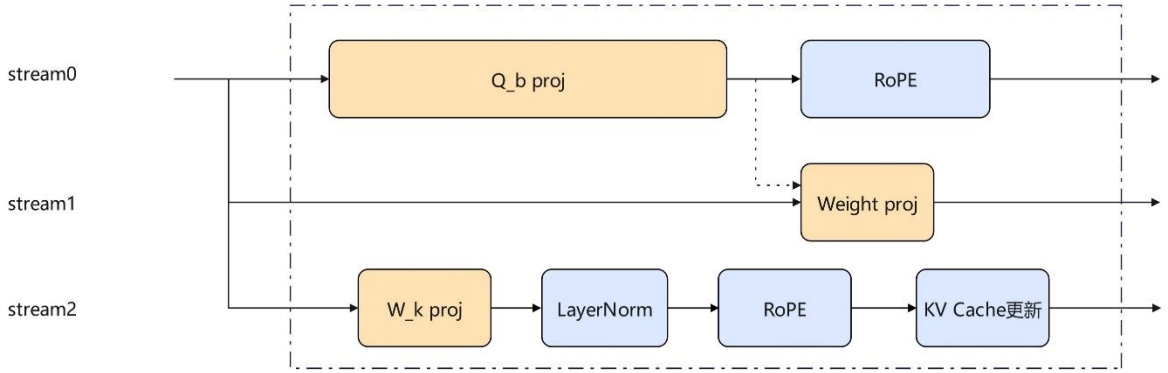
MOE多流并行



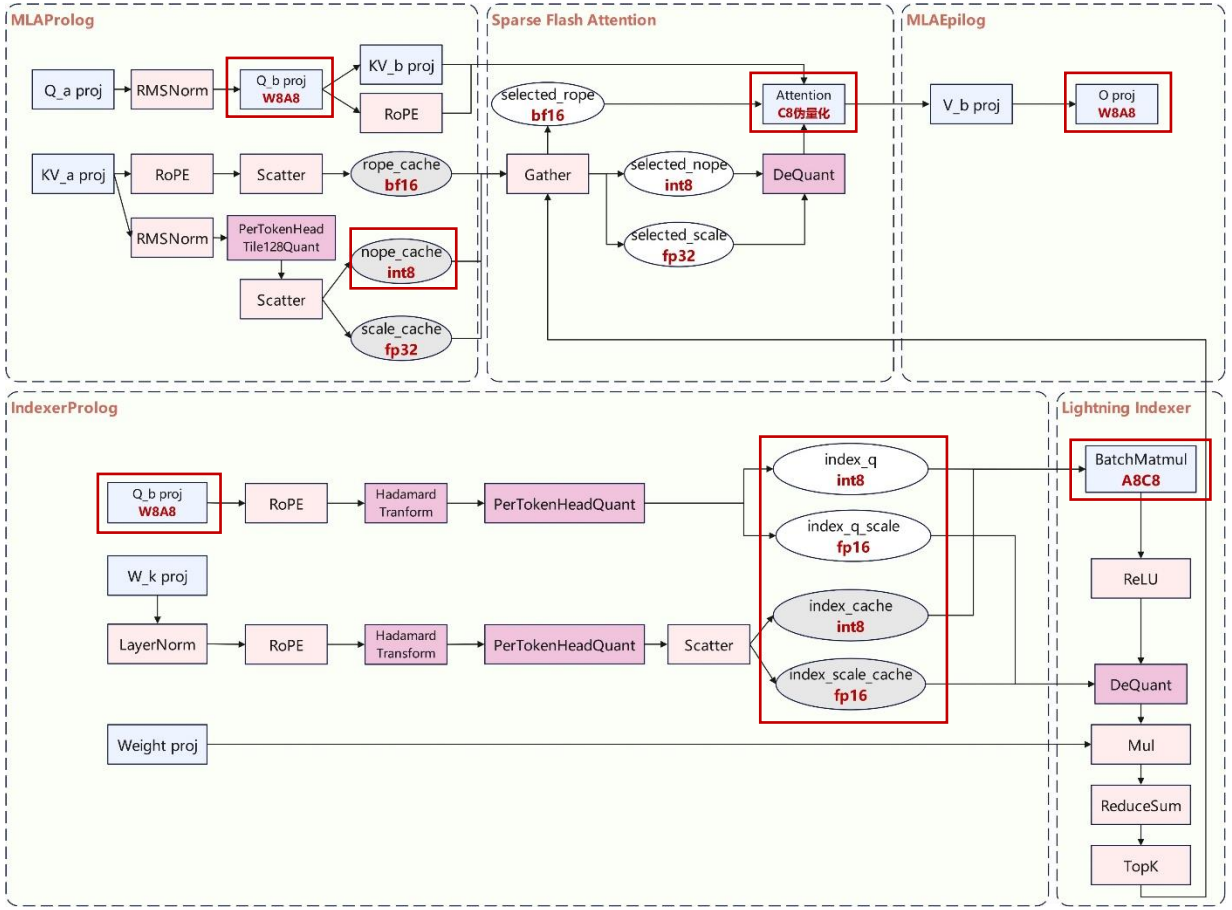
简单代码示例：

```
with torchair.scope.npu_stream_switch(stream_tag, stream_priority):
    # shared_expert use another stream
    hidden_states_share = self.shared_experts(hidden_states)
# router_experts use main stream
hidden_states_router = self.router_experts(hidden_states)
hidden_states = hidden_states_share + hidden_states_router
```

IndexerProlog多流并行



基于A3集群的整网优化方案解析—量化方案



- **MLAProlog**：除Q_b_proj使用W8A8量化，其他Linear均不量化；KVCache量化到Int8存储
- **Sparse Flash Attention**：KVCache Int8存储，BF16计算
- **IndexerProlog**：Q_b_proj使用W8A8量化，其他Linear均不量化；Indexer Q和Indexer Cache量化到Int8
- **Lightning Indexer:** BatchMatmtul使用Int8计算
- **MLAEpilog**：O_proj使用W8A8量化
- **MoE**：路由专家和共享专家使用W8A8量化

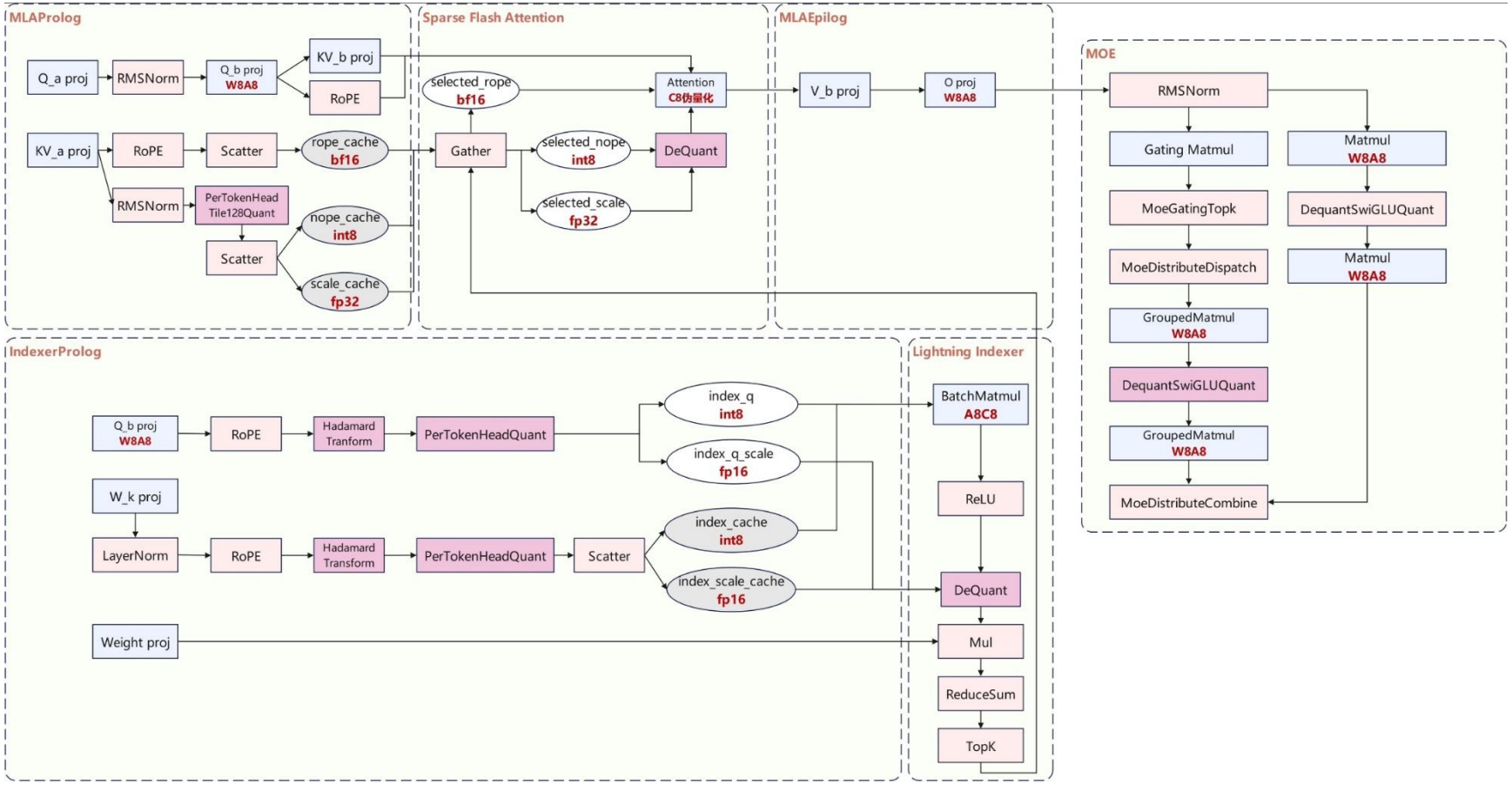
备注：

W8A8：W8指权重做Per-Channel静态INT8量化，A8指activation做动态Per-Token INT8量化

A8C8：A8表示Lightning Indexer中的Q做动态Per-Token-Head INT8量化，Indexer Cache做动态Per-Token-Head INT8量化

KVCache C8：表示KVCache 做动态Per-Token-Head-Tile-128 INT8量化

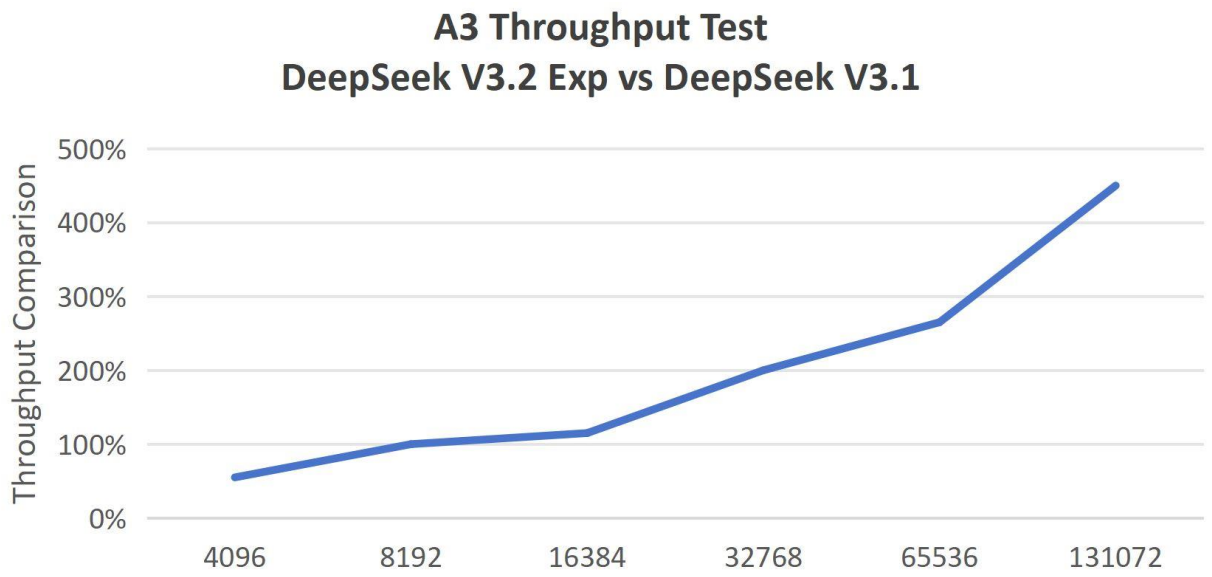
基于A3集群的整网优化方案解析—融合Kernel



Fusion kernels

- **Sparse Flash Attention**
- **Lightning Indexer**
- **MLAProlog**
- **IndexerProlog (PYPTO)**
- MoeDistributeDispatch
- MoeDistributeCombine
- GroupedMatmul
- DequantSwigluQuant

整网性能Benchmark



随着序列长度增加，DeepSeek V3.2 Exp 的性能优势逐步扩大，**当序列长度达到 128K 时，其吞吐达到 DeepSeek V3.1 的 450%**。

Global Batch Size	Seq Length	Chips	TPOT (ms)	Throughput (tokens/p/s)
256	65536	64	19.84	202
512	65536	64	22.78	351
128	131072	64	18.85	106
512	131072	64	25.8	310

长序列场景下，模型吞吐性能仍存在进一步优化空间。当前受限于 KVCache 内存容量，batch size 难以进一步提升，后续可通过 **KVCache Offload** 技术方案进一步提升吞吐。

https://gitcode.com/cann

CANN

优化点总结

- **部署策略**

  Prefill CP+EP；Decode DP+EP

  O-proj/Embedding/LM Head TP

- **低比特量化：**

  Weight W8A8；KV Cache伪量化C8；Indexer Cache A8C8

- **融合Kernel：**

  Sparse Flash Attention、Lightning Indexer、MLAProlog、IndexerProlog (PYPTO)

  GroupedMatmul、DequantSwigluQuant

- **通信：**

  MoeDistributeDispatch、MoeDistributeCombine、HCCL AIV通信

- **多流并行：**

  MOE共享/路由专家多流、MLAProlog、IndexerProlog多流等

- **多头MTP**

- …

目录

Future Plan

- **量化：**未来进一步开发低比特量化版本，探索KVCache量化压缩算法，软硬协同优化NPU计算效率，降低系统时延
- **KVCache Offload：**长序列场景会面临KVCache的内存瓶颈，可参考Shadow KV和Infinite LLM，达到类似的KVCache Offload效果，提升系统吞吐

欢迎体验和贡献

cann-recipes-infer

cann-recipes-train

cann-recipes-sig小组

cann-recipes交流群

cann-recipes-infer:
https://gitcode.com/cann/cann-recipes-infer

cann-recipes-train:
https://gitcode.com/cann/cann-recipes-train

欢迎广大开发者体验并参与贡献，如有疑问也可通过issue、SIG或者cann-recipes交流群联系我们！

Thank you.

社区愿景：打造开放易用、技术领先的AI算力新生态

社区使命：使能开发者基于CANN社区自主研究创新，构筑根深叶茂、跨产业协同共享共赢的CANN生态

Vision: Building an Open, Easy-to-Use, and Technology-leading AI Computing Ecosystem

Mission: Enable developers to independently research and innovate based on the CANN community and build a win-win CANN ecosystem with deep roots and cross-industry collaboration and sharing.

上CANN社区获取干货

关注CANN公众号获取资讯

https://gitcode.com/cann