

# DeepSeek-V4昇腾Day 0首发： 基于CANN的高性能推理优化实践

作者：许可、宦睿智

时间：2026年4月

# DeepSeek架构演进

## DeepSeek-V3

- 总参数 671B
- MLA架构
- 支持128K上下文
- MoE 混合专家模式  
稀疏激活

## DeepSeek-V3.2

- 总参数 685B
- 稀疏Attention架构
- 支持160K上下文
- MoE 混合专家模式  
稀疏激活

## DeepSeek-V4

- 总参数 1.6T/284B
- Hybrid Attention  
CSA + HCA  
滑窗 + 稀疏 + 压缩
- 支持1M长序列
- MoE 混合专家模式  
稀疏激活

# 昇腾Day 0支持DeepSeek-V4

## 950 PR/DT 系列首发支持

- 多种数据格式原生支持
- 基于UB高速互联
- 全链路推理优化



950DT系列10ms/20ms低时延推理

Atlas-A3系列30ms高吞吐

## Atlas-A3 大EP高吞吐实践

- 超节点高速互联
- PyPTO & TileLang生态易用
- AutoFuse加速性能

# 昇腾Highlights

## 核心算子原生适配

mHC、SparseAttentionSharedKV、Compressor、MoE融合算子原生适配，助力推理性能极致优化

## Torchtitan训练入图

Torchtitan-npu插件 + 算子Autofuse能力，“PyTorch Native”训练入图全新方案，训练开箱性能&易用性大幅提升

## PyPTO

融合算子的编程易用性提升，算子前端无需感知芯片的代际差异，后端通过pass IR和PTO-ISA指令进行区分，实现代际兼容

## SHMEM 通信编程范式

基于昇腾硬件，提供基于对称内存语义的通信编程API。通过 HIXL 集成至 Mooncake 系统，实现高性能 KV Offload，为超长文本推理提供低成本缓存扩展

<https://gitcode.com/cann>

## 低时延&高吞吐 极致优化

## CANN 全链路优化

## 社区开源支持 CANN-Recipe-Infer SGLang & vLLM

## 多种数据格式原生支持

支持原生FP8, MXFP4量化，进一步支持昇腾亲和的MXFP8量化，加速推理性能

## NpuGraphEx后端加速

针对 torch.compile 推出的高性能后端，基于 Fx 图与 AclGraph 架构，叠加多种独有特性，深度释放昇腾算力，实现了极致的图模式加速性能

## TileLang

开源社区TileLang同步支持融合算子，分别对应 Tilelang-Ascend的Expert和Developer开发模式，提供Ascend C基础指令和PTO AS两种对接层次，为各种编程前端语言和编译器提供多层开放接口

## AutoFuse自动融合

原生对接 PyTorch 2.x Inductor 技术栈，通过模式匹配自动识别并映射高性能子图，对接 AutoFuse JIT算子生成后端。源码增加一行 import，即可提升模型性能

# CANN

# 目录

Part 1 模型解析

Part 2 基于950PR/DT和A3集群的整网优化方案解析

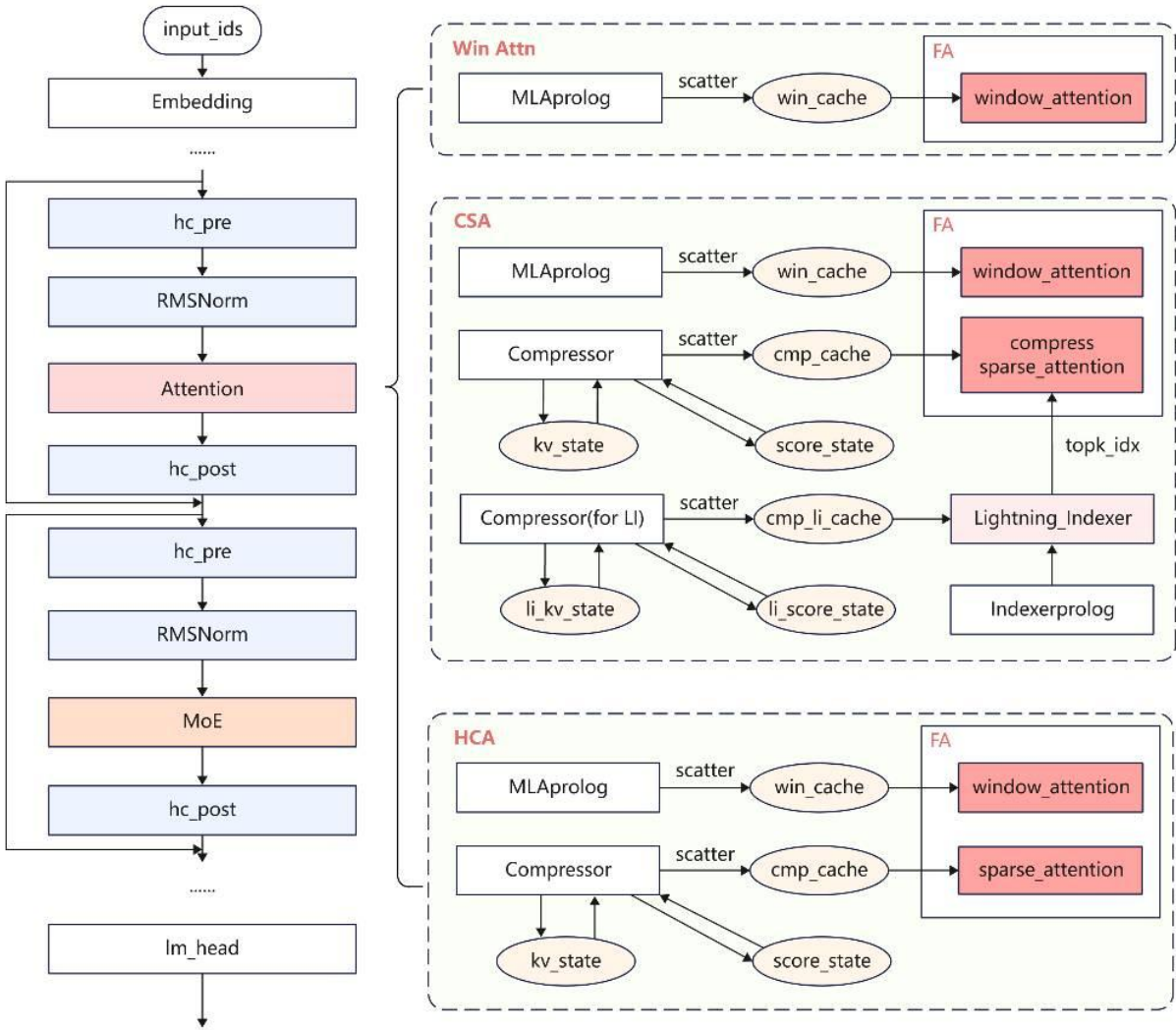
Part 3 Future Plan

# 模型解析 —— 整体结构

## 整体架构

- 支持1M长序列上下文;
- Hybrid Attention结构组合滑窗、稀疏、压缩算法;
- MoE架构稀疏激活;
- mHC 扩展传统残差链接;
- 原生支持MTP投机推理。

结构	DeepSeek-V4-Flash	DeepSeek-V4-Pro
参数量	284B	1.6T
激活参数	13B	49B
Hidden Size	4096	7168
层数	43	61
MoE	256路由, 1共享	384路由, 1共享
Gate Topk	6	6
Hash Routing	3	3
MTP	支持	支持



# 模型解析 —— mHC

## mHC 结构

### ➤ Manifold-Constrained Hyper Connection (mHC)

通过mHC，将residual从单流扩展到多流，拓宽了层间的信息传递效率，且更加稳定：

- 首层将 hidden states 的表征扩展为n倍， $n=4$ ；
- 在Attention/MoE前，**pre mapping**将扩展的维度融合回H维；
- 在Attention/MoE后，**post mapping**将hidden State的特征维度扩展n倍，激活内存增加：

$$h_i: (B, S, H) \rightarrow (B, S, nH)$$

- 通过res mapping将多流残差混合，和post mapping后的特征相加；
- 通过Sinkhorn-Knopp算法将残差矩阵限定在双随机矩阵的流形上，提升稳定性。

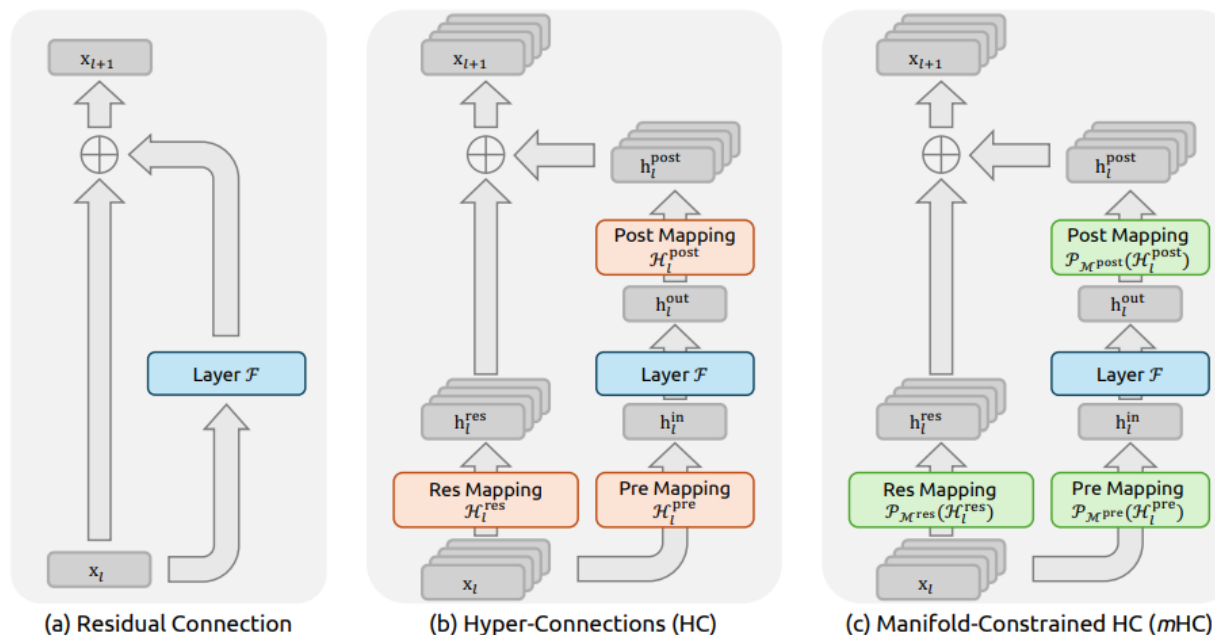


Figure source: [mHC: Manifold-Constrained Hyper-Connections \(Xie et al., 2026\)](#)



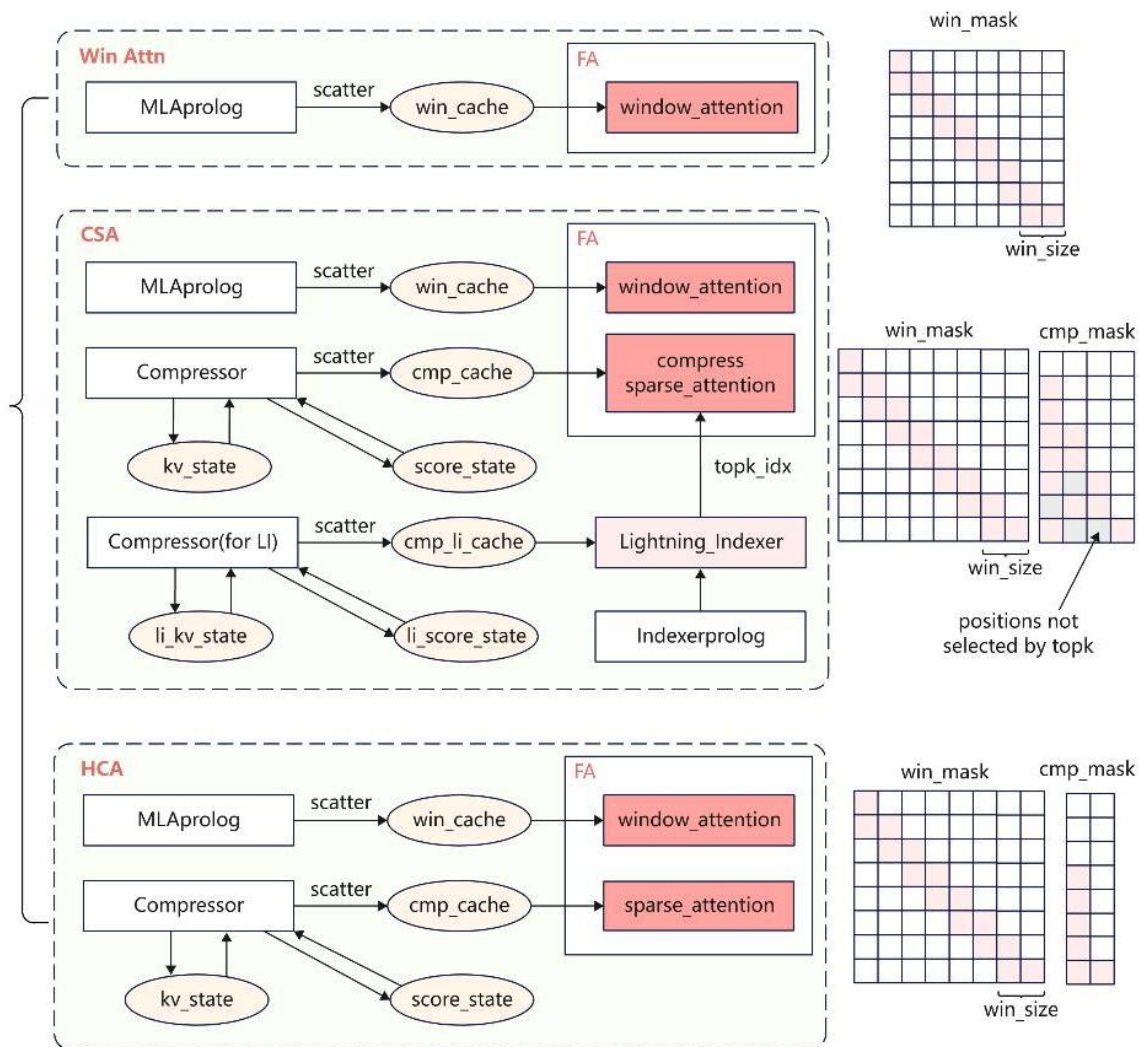
# 模型解析 —— Attention

## Attention 架构

### ➤ 多种Attention加速长序列计算

整体采用MQA，对比DeepSeek-V3.2系列，新增了滑窗注意力机制和KV Cache压缩算法，在整网中，交织使用不同的attention结构：

- **Window Attention 滑窗注意力机制：**
  - a. 采用window size 128；
- **Compressed Sparse Attention 滑窗+压缩4倍+稀疏：**
  - a. 对原始上下文采用 window size 128的滑窗注意力机制；
  - b. 对原始上下文压缩4倍，压缩时存在overlap，存储压缩后的cache；
  - c. 通过Lightning Indexer稀疏选取top 512 KV，用于DSA计算；
- **Heavily Compressed Attention 滑窗+压缩128倍：**
  - a. 对原始上下文采用 window size 128的滑窗注意力机制；
  - b. 对原始上下文压缩128倍，压缩时无overlap，不做稀疏；
- 在output projection时采用O\_a\_proj和O\_b\_proj；
- 使用attention sink。





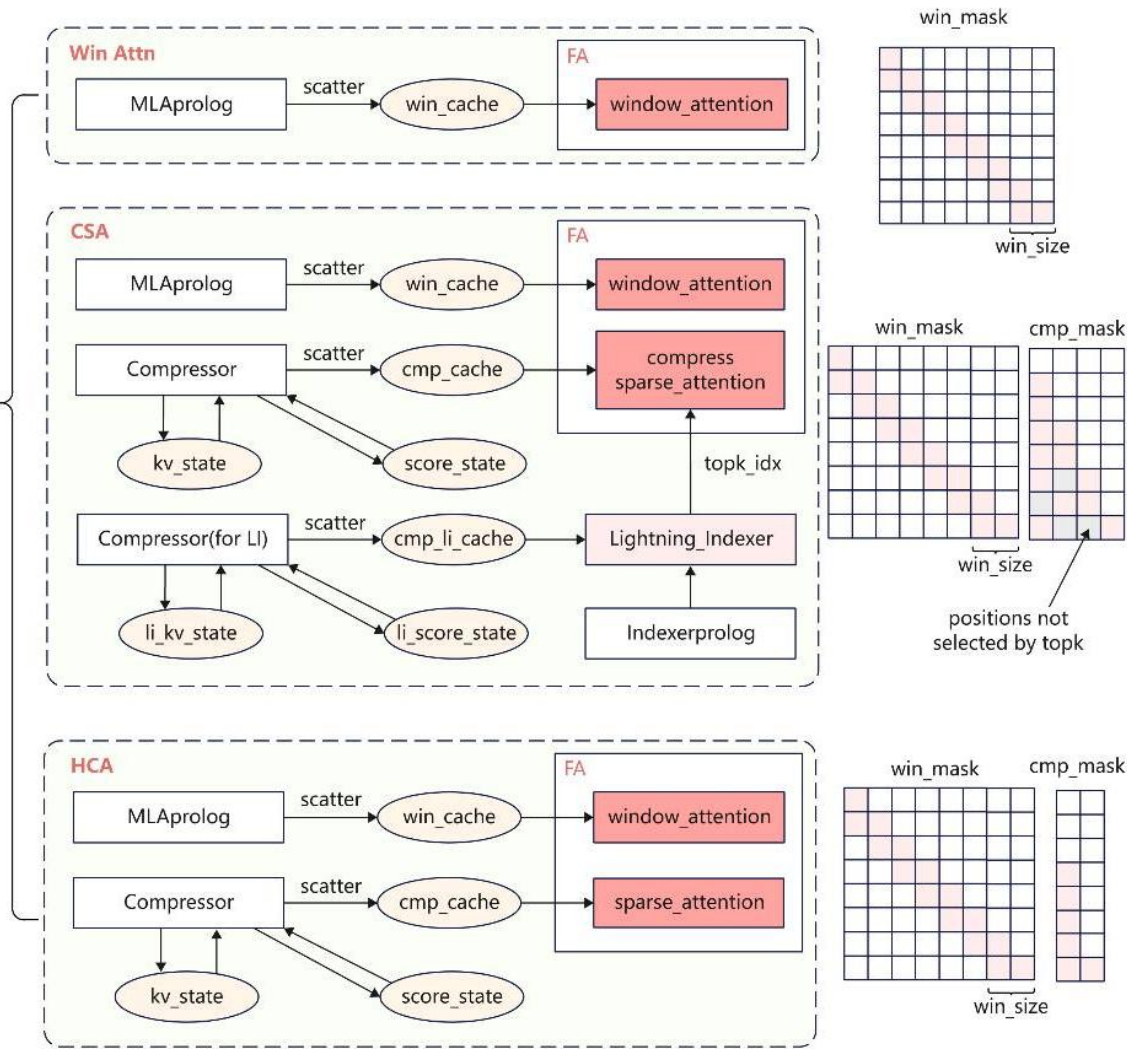
# 模型解析 —— Attention

## Attention 架构

### ➤ 多种Attention加速长序列计算

DeepSeek-V4-Flash 和 DeepSeek-V4-Pro 在Attention部分的结构差异如下:

Attention结构	DeepSeek-V4-Flash	DeepSeek-V4-Pro
层数	43	61
SWA	2层 (Layer 0, 1)	/
CSA	21层 (Layer 2, 4, ...)	30层 (Layer 2, 4, ...)
HCA	20层 (Layer 3, 5, ...)	31层 (Layer 0, 1, 3, 5...)
MTP	SWA	SWA
LI TopK	512	512
Num Attn Head	64	128
O_a group	8	16



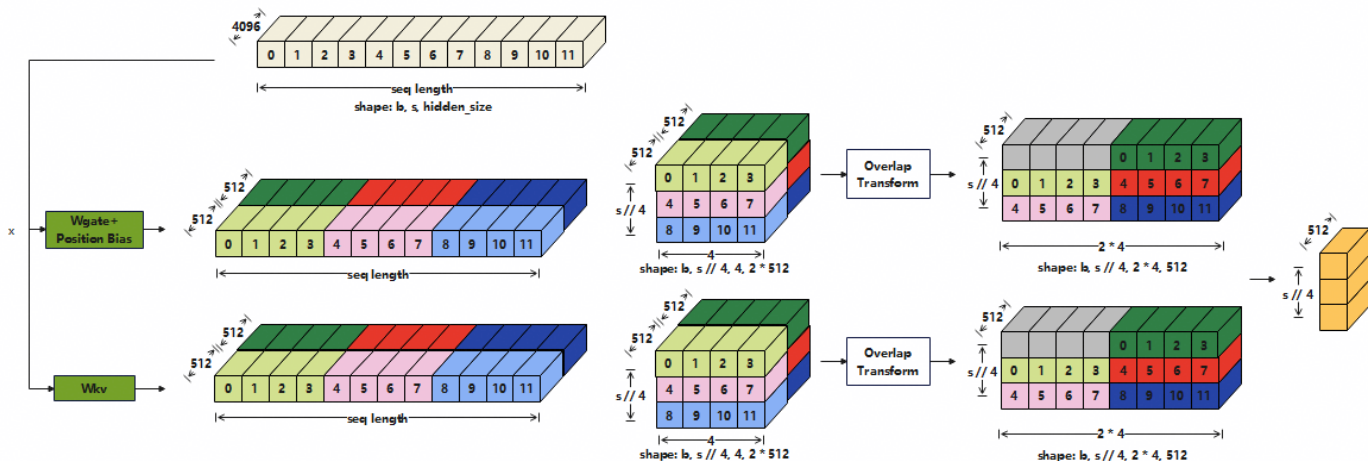
# 模型解析 —— Compressor

## Compressor 结构

### ➤ 不同压缩倍率交织，大幅降低KV Cache内存占用

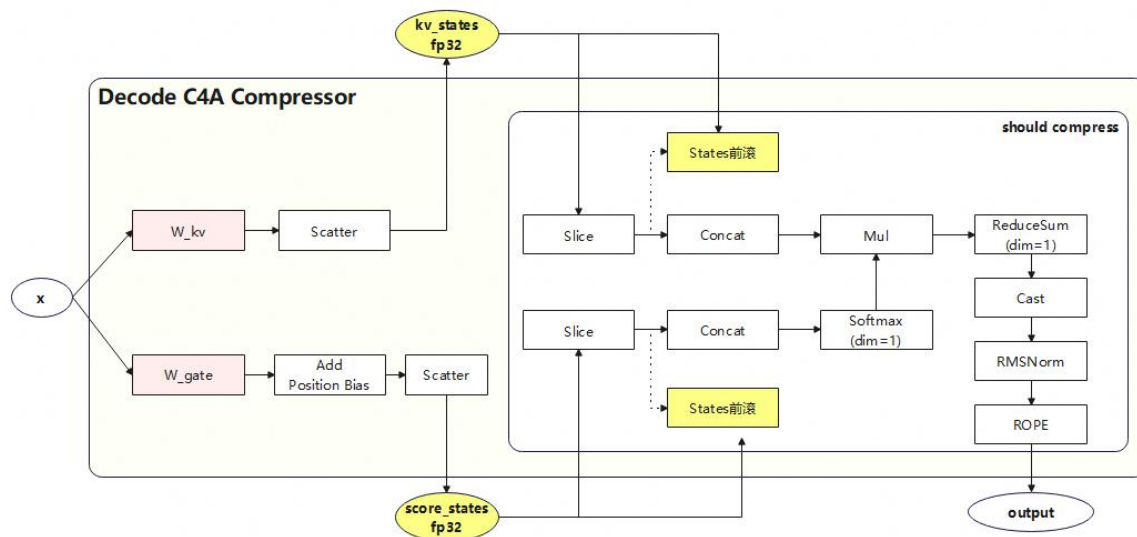
#### • CSA 压缩4倍，overlap token:

- 对原始上下文压缩4倍，压缩时存在overlap，每8个token压缩一次；
- 存储压缩后的 KV Cache；
- 不满足压缩长度的token信息暂存在 State Cache 和 Score Cache中；



#### • HCA 压缩128倍，无overlap:

- 对原始上下文压缩128倍，每128个token压缩一次；
- 存储压缩后的 KV Cache；
- 不满足压缩长度的token信息暂存在 State Cache 和 Score Cache中；



# 模型解析 —— 计算量与内存分析

## 模型计算量

### ➤ Attention计算量

Full Attention:

- 计算复杂度为 $O(L^2d)$ , L为Query序列长度;

MQA的整体计算量下降显著:

- SWA: MQA计算复杂度下降到 $O(L \cdot \text{win\_size} \cdot d)$ ;
- CSA: 稀疏 topk, attention  $O(Lkd)$ , k=512; Indexer对压缩4倍后的KV计算, 计算复杂度减少16倍  $O((\frac{L}{4})^2)$ ;
- HCA: 计算复杂度 $O(L \cdot \frac{L}{128} \cdot d)$ ;
- 增加了Compressor压缩KV Cache的计算开销。

### ➤ MoE计算量

单token激活6个路由专家和一个共享专家, 计算的专家参数量为:

$$(\text{top\_6} + 1) * \text{hidden\_size} * \text{intermediate\_size} * 3$$

## 内存分析

### ➤ 权重内存

权重内存	DeepSeek-V4-Flash 284B	DeepSeek-V4-Pro 1.6T
MoE W4	138 GB	721GB
Attention	6.8 GB	24GB
Compressor	0.57 GB	1.45GB
Embedding, etc.	1 GB	1.7GB

### ➤ Cache内存

- Window Cache: window size 128 长度的cache空间:

$$\text{batch\_size} * \text{win\_size} * \text{head\_dim} * \text{bytes} * \text{num\_layers}$$

- Compressed Cache 压缩后的KV Cache:

$$\text{batch\_size} * \frac{\text{kv\_length}}{\text{compress\_ratio}} * \text{head\_dim} * \text{bytes} * \text{num\_comp\_layers}$$

- Remainder Cache 不满足压缩长度的状态特征, 暂存在Cache内:

$$\text{batch\_size} * (\text{overlap} - 1) * \text{ratio} * \text{comp\_dim} * \text{bytes} * \text{num\_comp\_layers}$$

# 目录

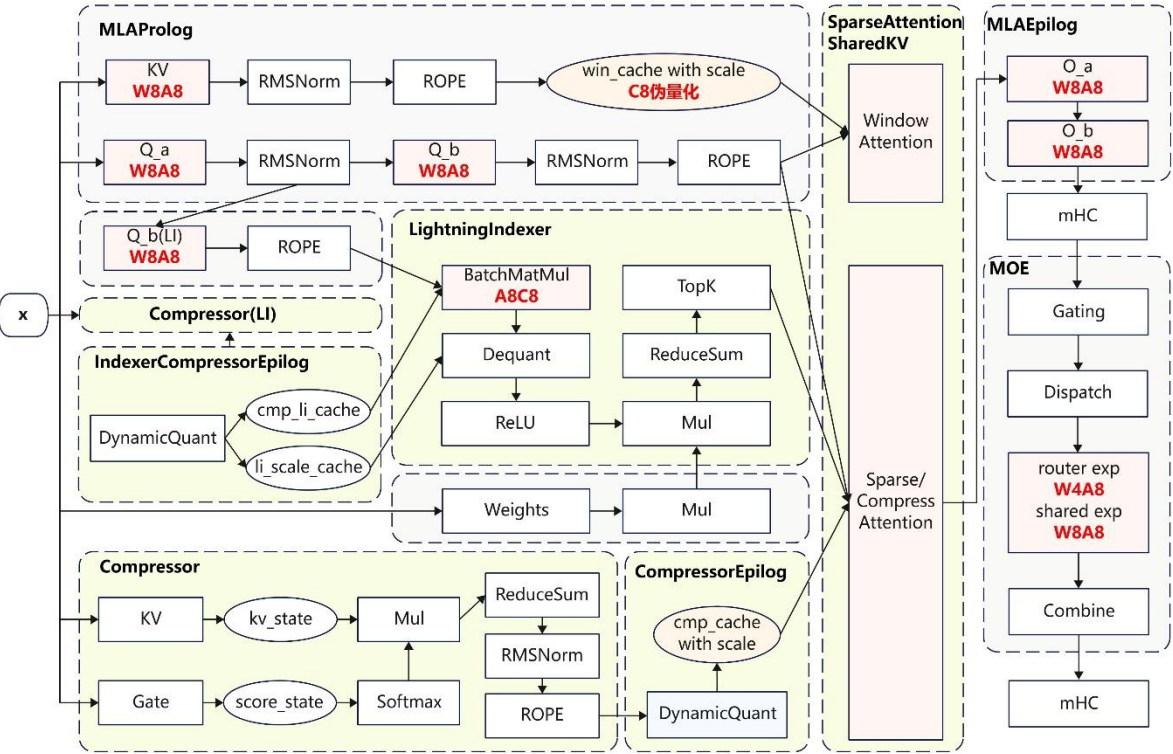
Part 1 模型解析

Part 2 基于950PR/DT和A3集群的整网优化方案解析

Part 3 Future Plan

# 整网优化方案解析 —— 量化策略

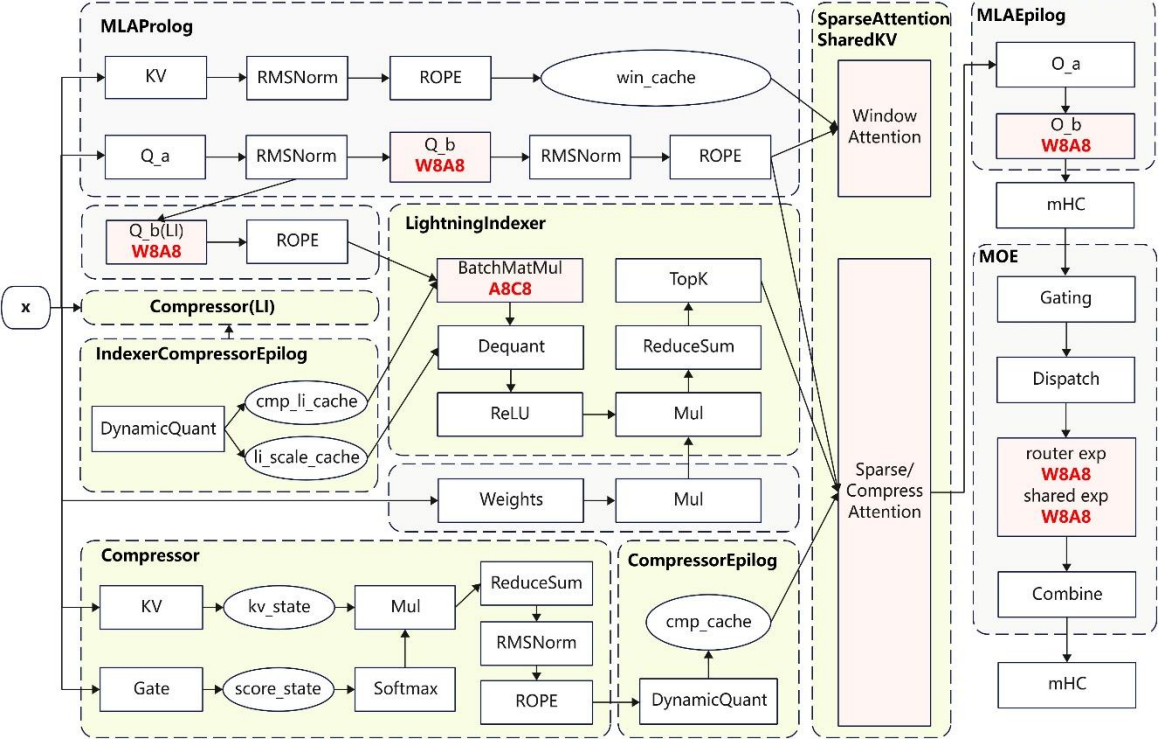
## 950PR/DT



### ➤ FP8/MXFP8 W8A8 + MoE W4A8推理

基于950PR/DT平台，支持了原生FP8量化、MoE MXFP4量化，同时开源支持硬件亲和的MXFP8量化，在算力密集的prefill场景可以取得比FP8更优的推理性能。

## Atlas-A3



### ➤ Int8 W8A8推理

对参数量较多的Linear进行量化，可以减少权重搬运的开销。Linear里对MLA的Q\_b\_proj, O\_b\_proj和Indexer的Q\_b\_proj进行量化。

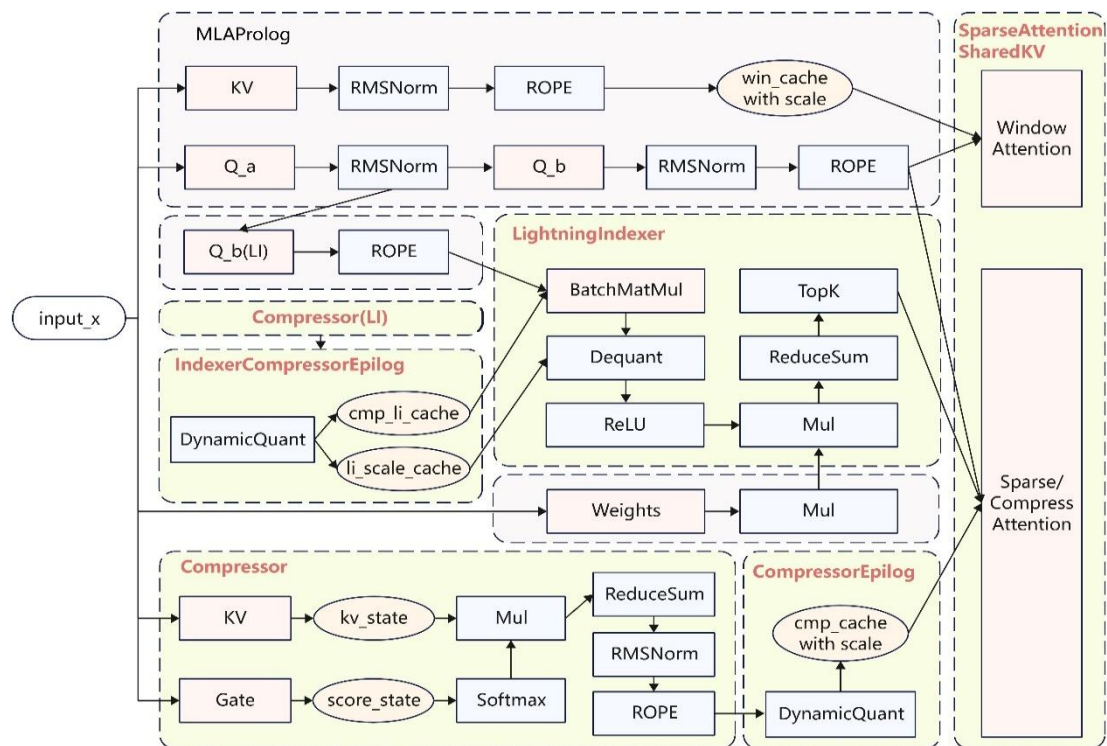


# 整网优化方案解析 —— 融合Kernel

## Attention

### ➤ 针对Attention模块，开源提供高性能融合算子：

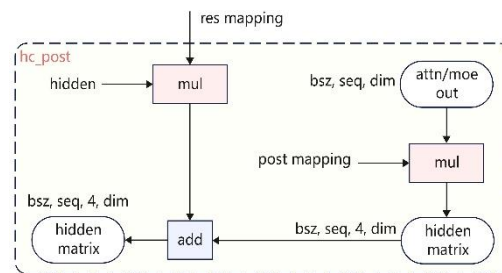
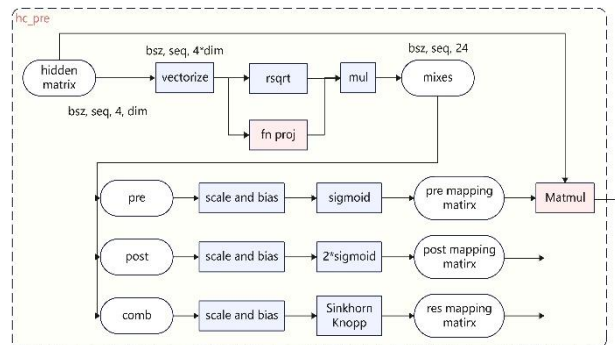
- **SparseAttnSharedKV (SAS)**：一个接口支持多种attention；
- **Compressor & Compressor Epilog**：支持不同倍率的KV Cache压缩与更新；
- **LightningIndexer (LI)**：高效筛选TopK稀疏KV；
- **Prolog**：开源PyPTO实现，加速prolog计算。



## mHC

### ➤ 多种方式共同开源，高效实现mHC

- 基于AscendC实现 HcPre & HcPost；
- 基于PyPTO实现 HcPre；
- 基于TileLang支持DS原生开源的Kernel。



## MoE

### ➤ 融合算子加速专家计算与通信

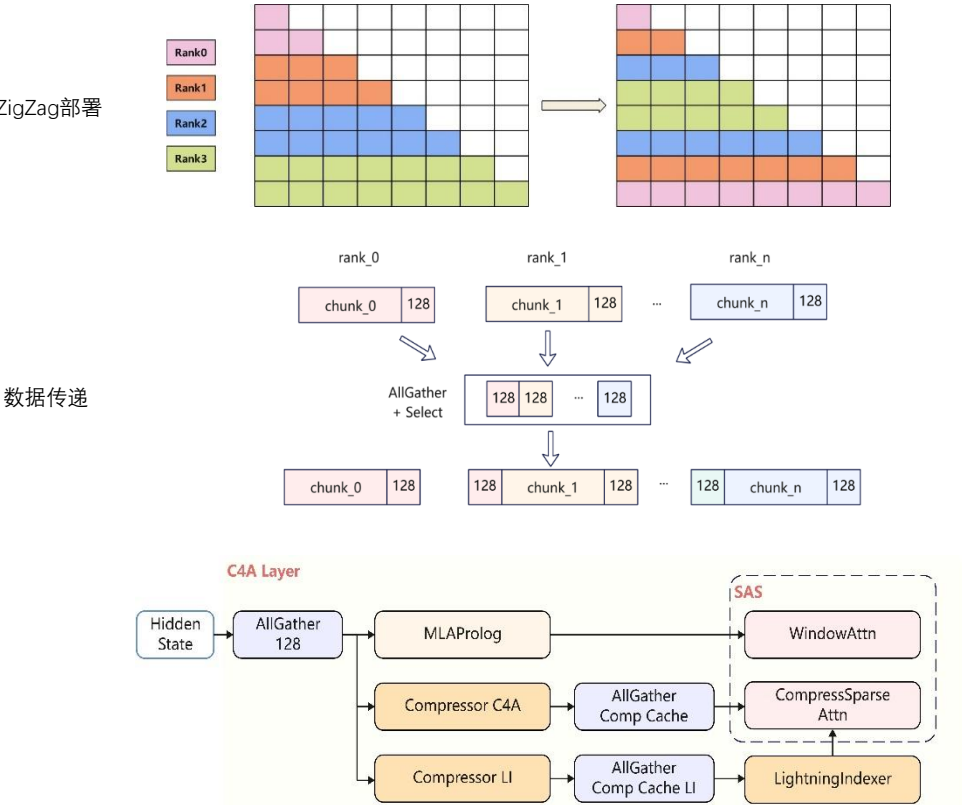
- **MoEGatingTopK**：支持本次模型的hash routing和sqrt\_softplus；
- **GroupedMatmul**：同时处理多专家计算，提升计算效率；
- **MoE Dispatch & Combine**：加速EP并行下的多卡token通信路由。



# 整网优化方案解析 —— 并行策略

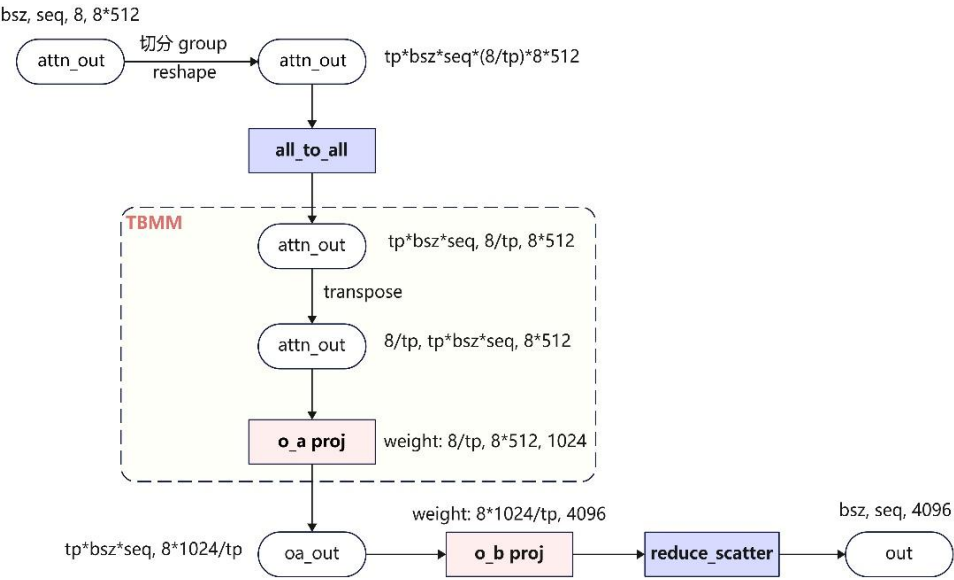
## Prefill Context Parallel & Expert Parallel

- 针对长序列场景，使用CP并行，优化TTFT：
  - 对Prefill输入沿SeqLength维度切分于不同卡上；
  - 通过zig-zag方式部署context chunk可以使卡间计算量相对均衡；
  - 针对window cache, CSA和HCA场景，通过AllGather或Send/Receive获取前序token。



## Decode Data Parallel & Expert Parallel

- Decode大EP部署降低时延
  - 选用Attention DP部署减少通信开销；
  - MoE EP部署降低单卡计算量；
  - O\_proj, LM\_Head, Embedding使用局部TP切分，降低device内存占用，缓解访存瓶颈。



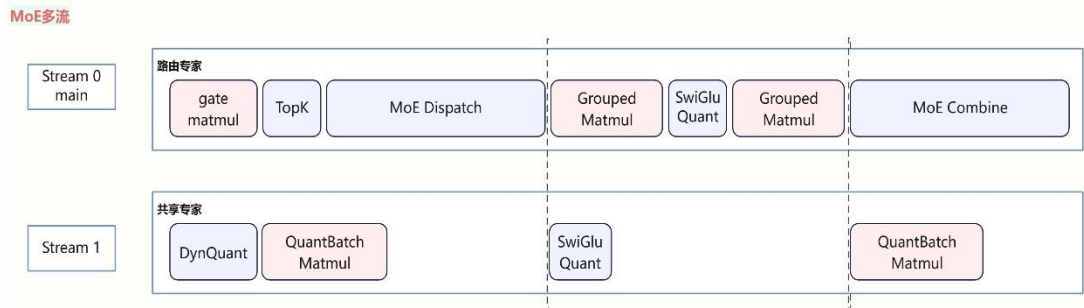
# 整网优化方案解析 —— 多流并行&CV控核

## 多流并行&CV控核

- 灵活并行最大化利用算力
  - 没有数据依赖的计算、通信算子，可以在多条计算流上并行执行；
  - 通过合理设置多流并行的位置，可以最大化利用计算资源；
  - 通过CV控核技术，减少计算资源的抢占。

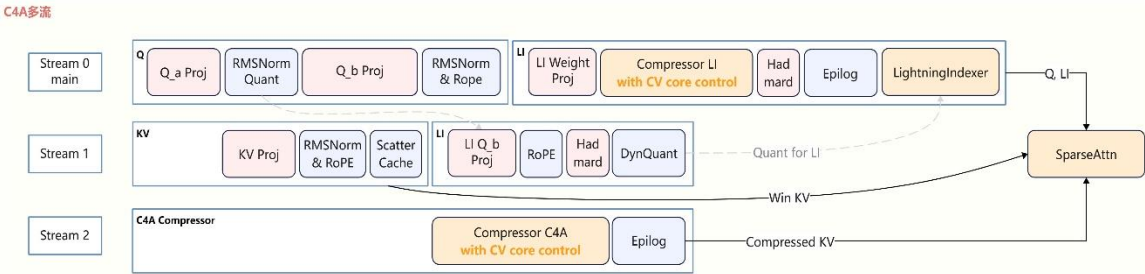
## MoE专家多流

- 共享专家&路由专家并行
  - 路由专家：计算与通信密集，grouped matmul等占用cube核；mc2 dispatch & combine 使用vector核；
  - 共享专家：耗时较短，算子较少；可以并行掩盖在路由专家下；
  - 效果：共享专家耗时完全掩盖，路由专家性能不劣化。



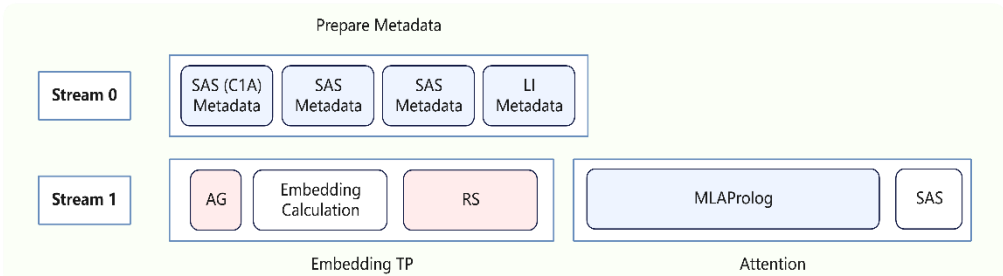
## Attention多流

- Prolog, LI, Compressor多路并行
  - Prolog、compressor 和 LI 之间存在较多并行掩盖的空间；
  - 通过合理控制Cube和Vector核数，可以减少计算资源的抢占，大幅提升性能；在C4A时，将C4A compressor 完全掩盖。



## AICPU Scheduler多流

- Attention算子的tiling计算流程可与计算掩盖
  - 将aicpu scheduler算子与模型计算并行。

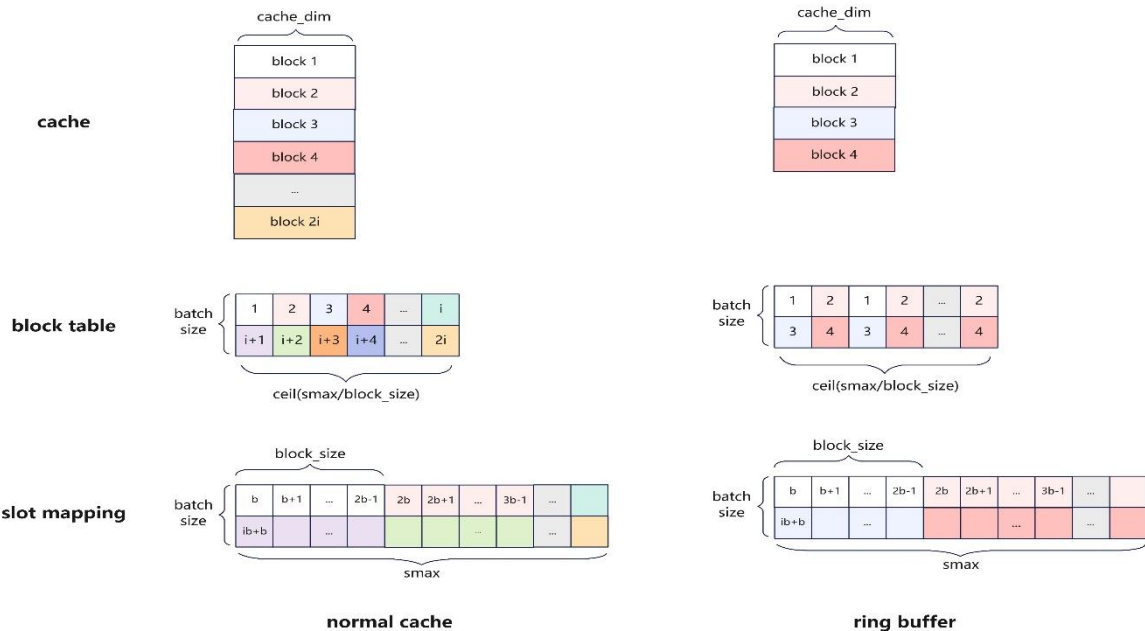


# 整网优化方案解析 —— MTP

## 背景分析

### ➤ Page Attention cache

本模型针对window kv及compressor模块里的kv/score state设计了ring buffer复用cache内存，实现最小化的cache空间占用；其余随上下文长度增长的KV Cache的内存，按照最大长度申请block。



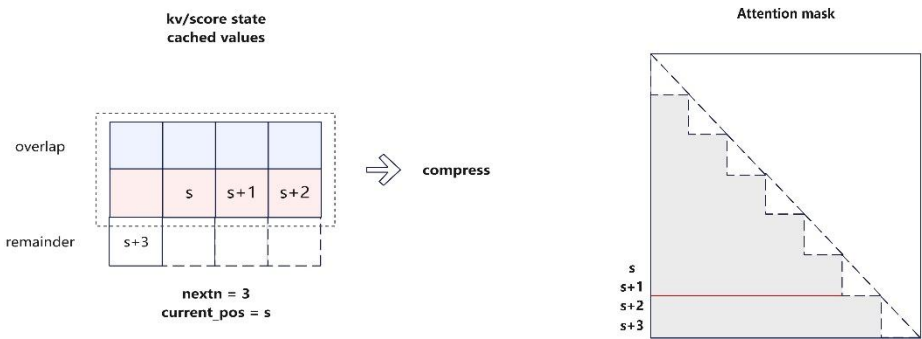
### ➤ MTP Spec model

- 模型结构：MTP模型仅使用window attention。
- Cache结构：在当前实现中，多个MTP头共享一份权重和一份KV Cache

## Main model适配

### ➤ MTP场景适配

- 投机未被接收的token在下一个step需要重新推理，因此compressor的kv/score state，以及win cache需要额外缓存，支持重新压缩。
- LI和SAS算子内置梯形掩码，确保包含待验证token的压缩cache不参与主模型token的计算。



### ➤ Cache申请

- 结合ring buffer设计，需要考虑MTP回滚，预留额外的Cache空间。

# 整网优化方案解析 —— npugraph\_ex

## 简介

npugraph\_ex是一个基于CANN的PyTorch FX图优化组件，通过torch.compile入口无缝对接NPU图模式（ACLGraph），帮助开发者在零额外成本下充分释放NPU性能。

- **快速接入**：CANN提供ACLGraph，有效的消除了算子调度时延，并已经集成在PyTorch中，可快速使能；
- **高性能**：在复用原有ACLGraph调度能力前提下，借助于PyTorch的FX图，进一步叠加NPU亲和的优化；
- **可集成**：面向大模型推理的服务化框架，可以做到快速集成，无缝对接到vLLM/SGLang等服务化框架中。

## 静态Kernel编译

对于纯静态shape网络或者shape变化较少的动态shape网络，如需提升网络执行性能，可通过算子预先静态编译达到目的，该方式简称为静态Kernel编译。

- 编译时已知所有Tensor的大小，存储空间利用率高。
- 编译时可以针对实际的shape大小做针对性优化。
- 编译时完成Scalar计算，避免运行时打断并行指令流。
- 已知确切数据规模，无需插入同步点，保持全并行执行。

## 模型编译缓存

成图编译涉及两段耗时，一段是Dynamo的编译耗时，另一段是基于Dynamo编译出的FX图进行再处理的耗时，再处理的行为会根据模式不同而有所变化。为降低成图编译的耗时，npugraph\_ex提供了模型编译缓存方案，通过cache\_compile接口将首次编译的结果落盘，从而加快torch.compile图模式的启动时间。



## 多流+控核

- **支持torch原生的多流接口**：对于一些可并行的场景，可以划分多个stream提升执行效率。
- **提供算子级控核能力**：合理分配计算资源，避免计算资源抢占。

npugraph\_ex 详情：

<https://gitcode.com/cann/community/tree/master/events/meetup/slides/sig-ge/20260210>

# 昇腾950DT性能Benchmark —— DeepSeek-V4-Flash 284B

- 单请求时延低于10ms，多并发吞吐4722 TPS@20.15ms。
- 950PR/DT支持原生FP8+MXFP4量化权重部署；
- 采用16P部署，叠加Attention Data Parallel（DP）与MoE Expert Parallel（EP）混合并行策略，在低时延和高吞吐场景均能发挥性能优势。

Global Batch Size	Chips	MTP	Data Type	Seq Length	TPOT (ms)	Throughput (Tokens/p/s)
16	16	3	Hybrid MXFP8-MXFP4	8192	6.71	148.96
256	16	3	Hybrid MXFP8-MXFP4	8192	9.84	1625.31
1536	16	1	Hybrid MXFP8-MXFP4	8192	20.33	4722.22
16	16	3	Hybrid MXFP8-MXFP4	131072	7.80	128.15
256	16	3	Hybrid FP8-MXFP4	8192	11.06	1447.00
1536	16	1	Hybrid FP8-MXFP4	8192	24.28	3953.49

# 昇腾950DT性能Benchmark —— DeepSeek-V4-Pro 1.6T

- 单请求时延20ms，多并发吞吐388TPS@21ms。
- 950PR/DT支持原生FP8+MXFP4量化权重部署；
- 采用16P部署，叠加Attention Data Parallel（DP）与MoE Expert Parallel（EP）混合并行策略，在低时延和高吞吐场景均能发挥性能优势。

Global Batch Size	Chips	MTP	DataType	Seq Length	TPOT (ms)	Throughput (Tokens/p/s)
16	16	3	Hybrid MXFP8-MXFP4	8192	17.64	56.70
64	16	3	Hybrid MXFP8-MXFP4	8192	19.03	210.16
128	16	3	Hybrid MXFP8-MXFP4	8192	20.61	388.23



# 昇腾Atlas-A3性能Benchmark —— DeepSeek-V4-Flash 284B

- 超节点高速互联，助力大EP高吞吐部署3733 TPS@30MS
  - 采用64卡大EP部署；
  - 支持INT8量化高效部署；
  - 采用W8A8C16量化策略。

Global Batch Size	Chips	MTP	Seq Length	TPOT (ms)	Throughput (Tokens/p/s)
7168	64	1	8192	30	3733

# 整网优化总结

## 部署策略

- Prefill CP+EP;
- Decode DP+EP;
- O-proj/Embedding/LM Head TP;

## 低比特量化

Weight W8A8 FP8/MXFP8; MoE W4A8 MXFP4 ; KV Cache伪量化C8; Indexer Cache A8C8;

## 融合Kernel

- Sparse Attention、Lightning Indexer、Prolog、IndexerProlog (PyPTO);
- GroupedMatmul、DequantSwigluQuant;

## 通信优化

MoeDistributeDispatch、MoeDistributeCombine、HCCL AIV通信;

## 多流并行与控核

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

## NpuGraph\_Ex

## 多头MTP

<https://gitcode.com/cann>



# 目录

Part 1 模型解析

Part 2 基于950PR/DT和A3集群的整网优化方案解析

Part 3 Future Plan

# 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;
- **SuperKernel**: 使能Super Kernel进一步降低算子启动开销与调度间隙, 提升计算执行效率;
- **融合Kernel性能优化**: MegaMoE 融合、Attention Flash Decode 优化;
- **训练**: 0-Week 开源基于TorchTitan的续训练支持;

# DeepSeek-V4昇腾Day 0首发——系列直播

基于CANN的高性能推理优化实践



基于950的DSA & mHC高效优化实践



基于torchitan + AutoFusion的高效训练实践



TileLang & PyPTO助力模型快速调优



4月27日~29日  
晚上19:00  
扫码预约直播

# 欢迎体验和贡献

cann-recipes技术报告



cann-recipes-train



cann-recipes-infer:

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

cann-recipes-train:

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

cann-recipes社区



cann-recipes交流群



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！

**CANN**



# 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