DeepSeek-V3.2-Exp Ascend C 融合算子优化实践

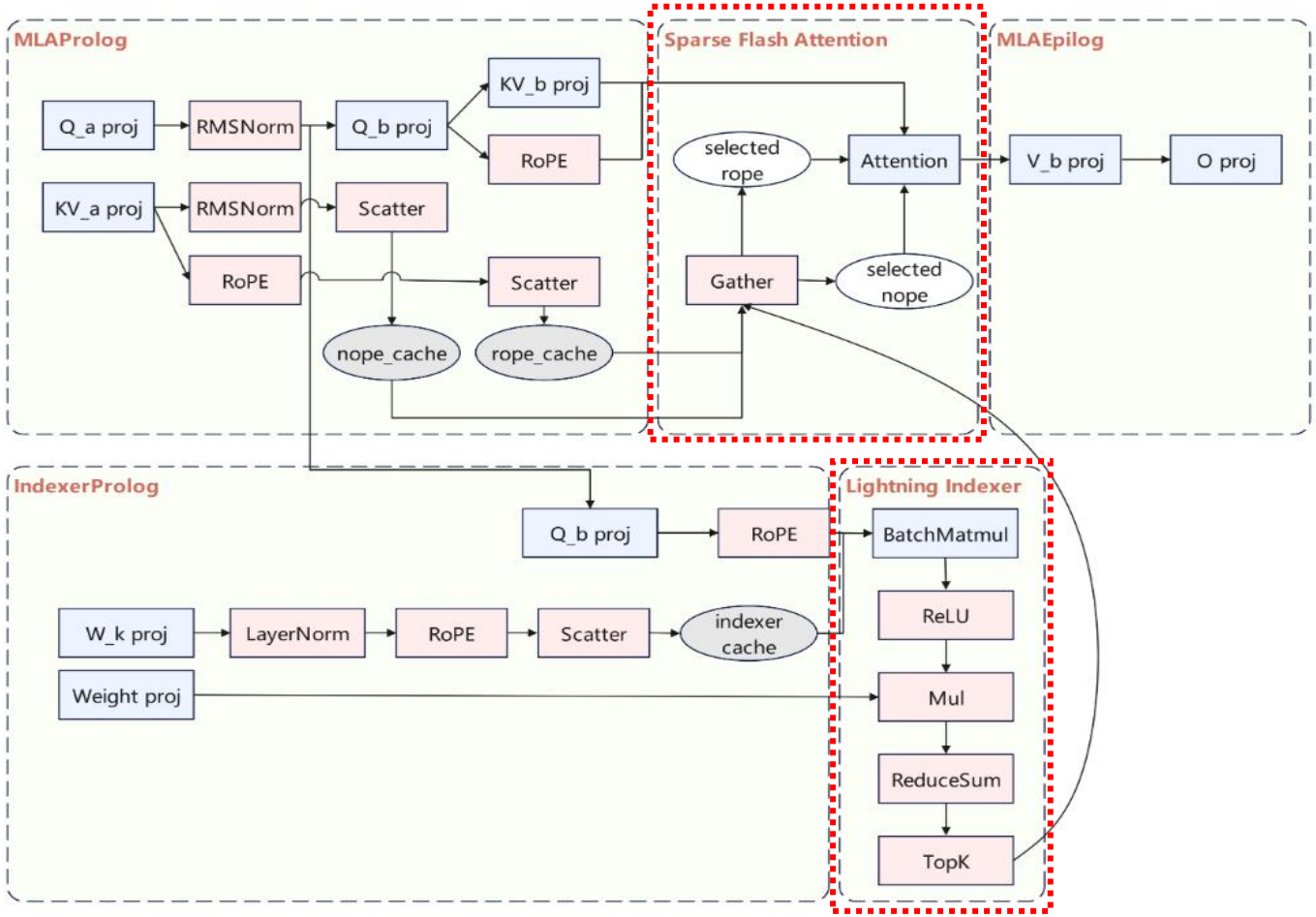
CANN

目录

DSA融合算子简介

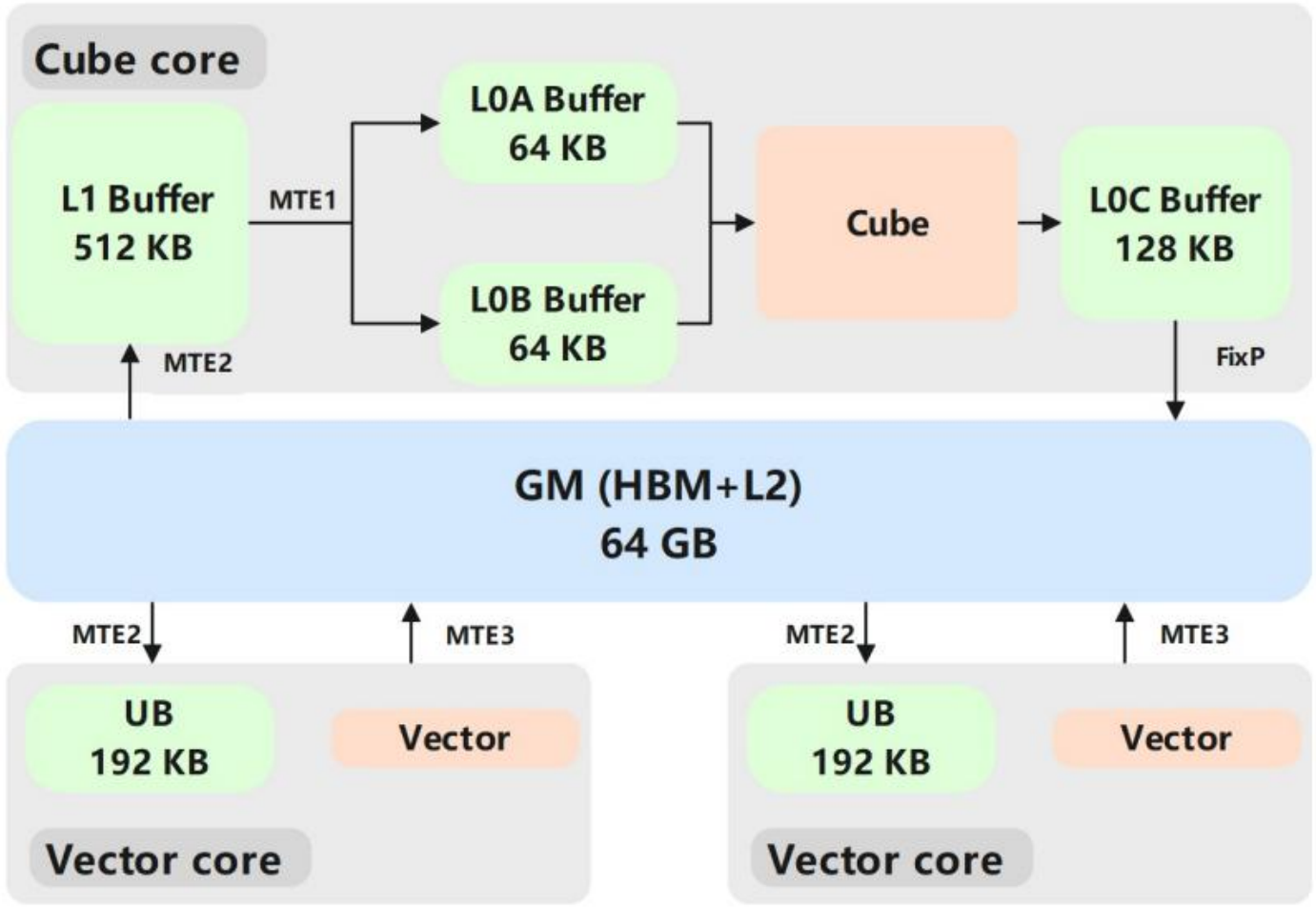
- **LightningIndexer**
  - > $S = Q_{idx} \ @ \ K_{idx}^{T}$
  - > $S' = ReLU(S)$
  - > $S_w = \left(W \ @ \ [1]_{1*s_k}\right) \ * S'$
  - > $Score = \ [1]_{1*g} \ @ \ S_w$
  - > $TopK(Score)$

- **SparseFlashAttention**
  - > $\tilde{K} = gather(K, \ topk\_indices)$
  - > $\tilde{V} = gather(V, \ topk\_indices)$
  - > $FlashAttention(Q, \tilde{K}, \tilde{V})$
    - C_1: $Q@K^T$
    - V_1: online softmax
    - C_2: $P@V$
    - V_2: rescaling O



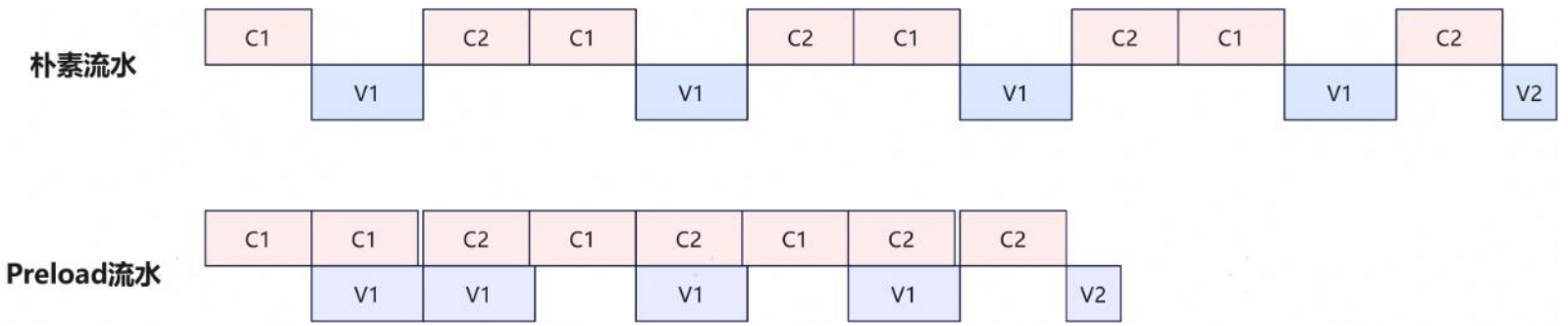
目录

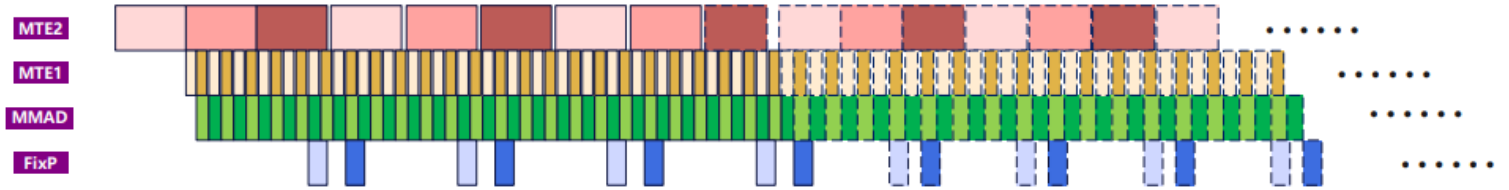
芯片架构简介


Cube core
L1 Buffer
512 KB
MTE1
L0A Buffer
64 KB
L0B Buffer
64 KB
Cube
L0C Buffer
128 KB
MTE2
FixP
GM (HBM+L2)
64 GB
MTE2
MTE3
MTE2
MTE3
UB
192 KB
Vector
Vector core
UB
192 KB
Vector
Vector core

Pipeline设计

- 两级流水
  - > **核间流水**
  - > **核内流水**
- 核间流水
  - > **preload流水策略**
  - > **核间基本块**
- 核内流水
  - > **DoubleBuffer**
  - > **核内基本块**

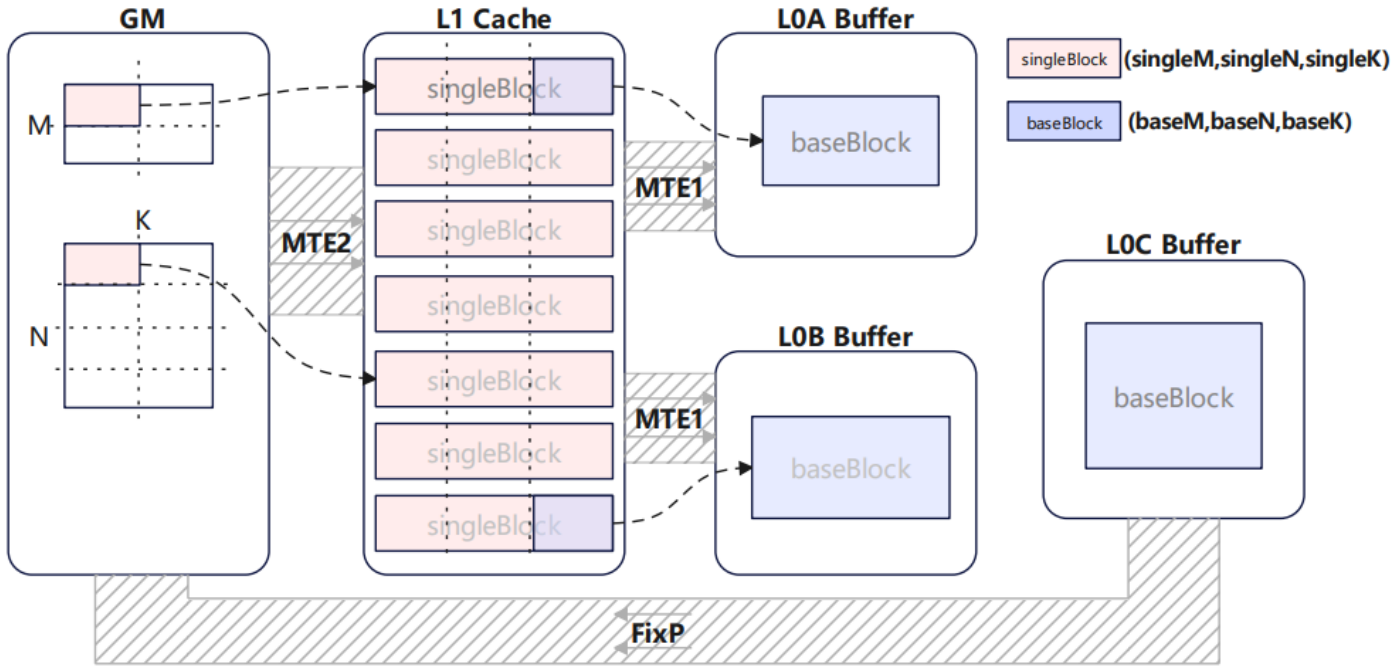




CANN

Tiling方案

- 两级tiling设计 *(以cube核为例)*
  - > gm->L1
  - > L1->L0A/B->L0C
- L1空间划分
  - > Q/P矩阵分时复用：256*576 = 288KB
  - > KV 3-buffer循环复用：3*128*288 = 216KB
- L0A/B/C空间划分
  - > L0A：2*32KB
  - > L0B：2*32KB
  - > L0C：2*64KB



MM计算与HBM带宽的关系

MM计算与带宽的关系

1. cube算力
   a. bf16数据类型下，每cycle可计算 4096 个数
2. HBM带宽
   a. 折算到每核每cycle约 32B

对于M、N、K的tiling块，在A矩阵常驻L1的前提下，若达到计算与访存的均衡，需满足以下条件：

$$\frac{M \cdot N \cdot K}{4096} = \frac{N \cdot K \cdot 2}{32}$$

由以上公式可解出：M=256；

故只需要M切256，即可实现MM计算与HBM访存的完美均衡；

L0 buffer与切分的关系

L0 buffer与MNK切分的关系

1. L0 buffer大小
   a. L0A: 64K
   b. L0B: 64K
   c. L0C: 128K

若考虑到流水排布，L0A/B/C均需要开DB，对应的可使用空间为32KB、32KB、64KB；则对于bf16进fp32出的计算场景，每块buffer的可以存放16384个数；

可得到以下关系：

$$M \cdot K = 16384$$
$$N \cdot K = 16384$$
$$M \cdot N = 16384$$

由以上公式可以解出：M = N = K = 128

MM计算与MTE1带宽的关系（一）

MM计算与L1->L0搬运的关系

1. MTE1的带宽
   a. L1->L0A: 256B/cycle
   b. L1->L0B: 128B/cycle

由以上章节可知，最优的MNK切分均为128；此时的计算耗时为 $\frac{128 \cdot 128 \cdot 128}{4096} = 512$ cycle，L1->L0的搬运耗时为 $\frac{128 \cdot 128}{128} + \frac{128 \cdot 128}{64} = 384$ cycle；可以做到计算bound；

但Attention计算有两个mm计算：

1. mm1: $q \cdot k^T$
   a. m = $s1 \cdot g$
   b. n = s2
   c. k = D = $D_n + D_r$
2. mm2: $p \cdot v$
   a. m = $s1 \cdot g$
   b. n = D = D_n
   c. k = s2

MM计算与MTE1带宽的关系（二）

其中D为网络超参，deepseek网络结构中固定为576，而该参数为mm1的k轴；

- 若mm1的k轴仍按照128切分，则会切分为： 576=128*4 + 64；此时mm1的最后一次计算的cycle数为：$\frac{128 \cdot 128 \cdot 64}{4096} = 256$，mm2的第一次L1->L0搬运cycle为：$\frac{128 \cdot 128}{128} + \frac{128 \cdot 128}{64} = 384$；此时计算无法掩盖访存；
- 若mm1的k轴仍按照96切分，则会切分为： 576=96*6；此时mm1的最后一次计算的cycle数为：$\frac{128 \cdot 128 \cdot 96}{4096} = 384$，mm2的第一次L1->L0搬运cycle为：$\frac{128 \cdot 128}{128} + \frac{128 \cdot 128}{64} = 384$；此时计算完美掩盖访存；

故最终的tiling切分为：

1. mm1
   a. m=128, n=128, k=96
2. mm2
   a. m=128, n=128, k=128

TopK实现

- 指令介绍
  - > **VBS32：** 32个token的排序
  - > **VMS4v2：** 4组有序向量归并

- 实现方案
  - > **分组排序**：*S = n * 32*

通过VBS32指令将每32个 token 进行稳定降序排列，直到将整个序列的 token 分组排序完毕；

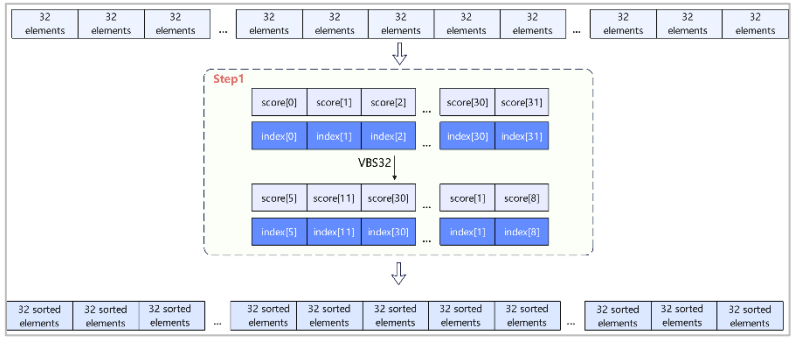
  - > **归并**：*4*32->128, 4*128->512, 4*512->2048*

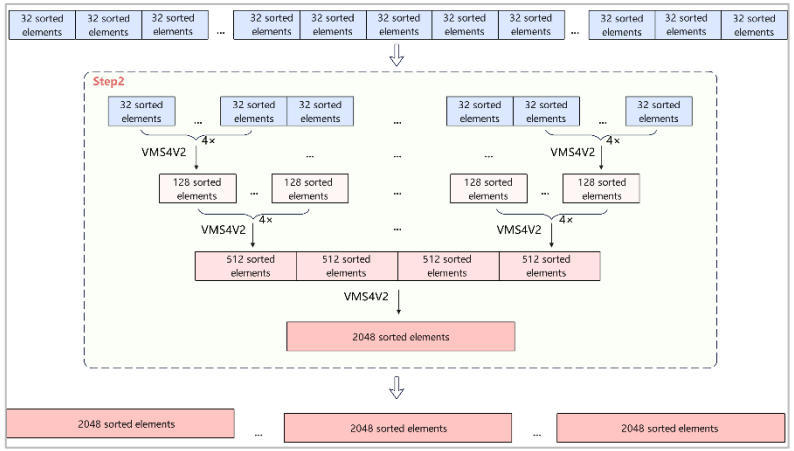
通过VMS4v2指令将至多4组（可为2/3组）长度为32、128、512的已排序向量进行归并，直到合并后的向量长度达到2048；

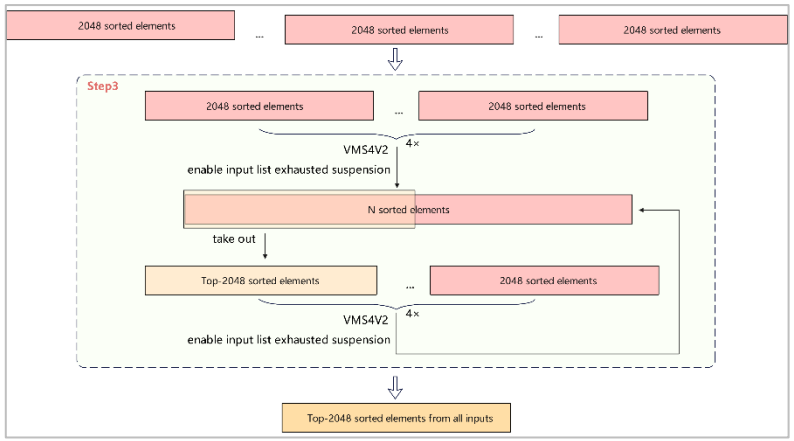
  - > **规约**：*4*2048 -> 2048+ -> 2048*

采用VMS4v2指令的耗尽模式将至多4组长度为2048已排序的向量进行归并，取出前2048个数与未参与合并排序的有序向量重复进行归并，直到比较完毕；

- 复杂度评估
  - > **分组排序+规约**：*4S*
  - > **规约**：*0.36S + 3.32k*







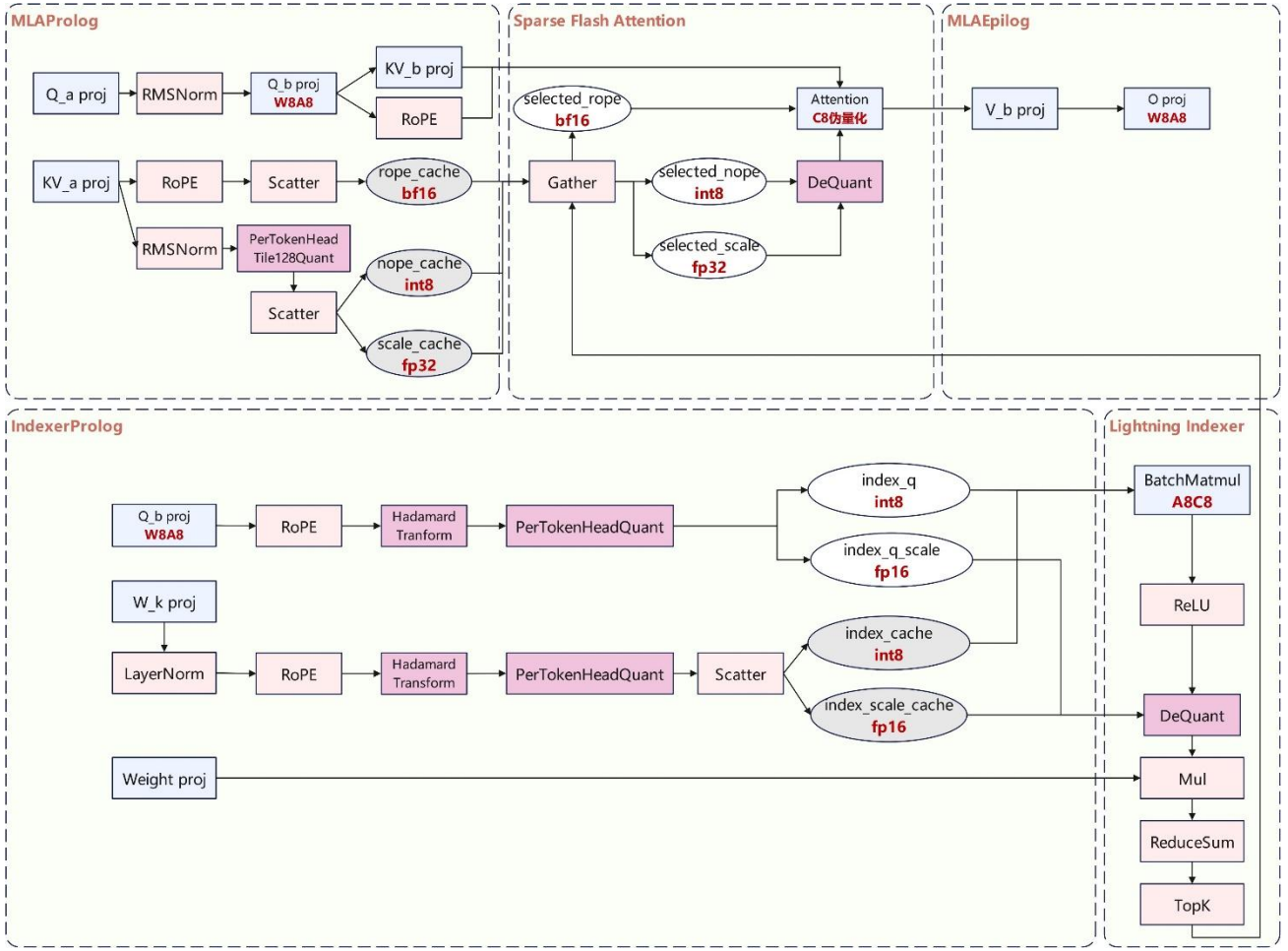
https://gitcode.com/cann

CANN

目录

支持量化

- **LightningIndexer – A8C8:**
  - **Qidx/Kidx**
    - **per-token-head**
    - **int8**
- **SparseFlashAttention - A16C8:**
  - **Q: BF16**
  - **KV cache:**
    - **NOPE: per-tile int8**
    - **ROPE: Bf16**



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