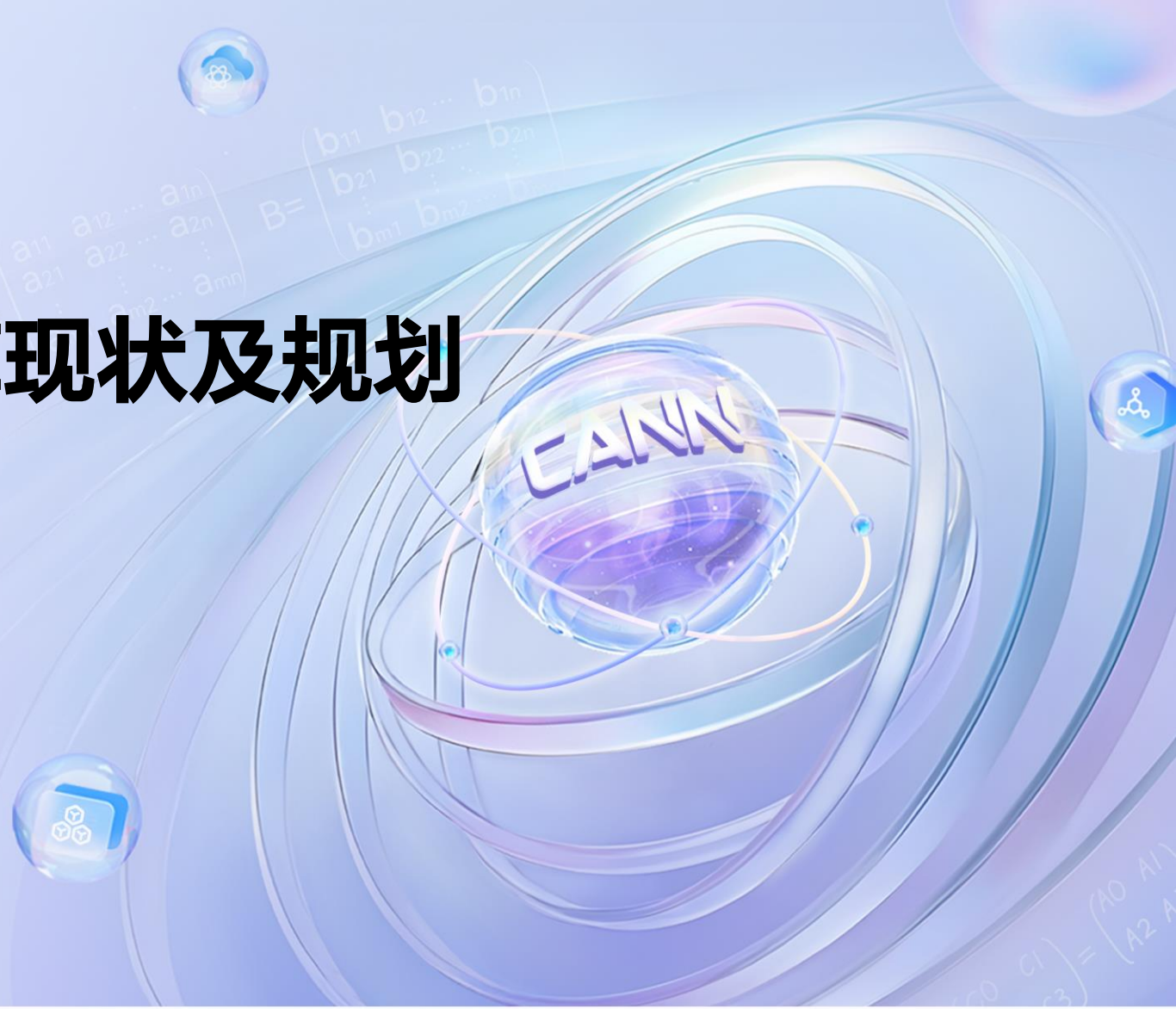
CATLASS算子模板库现状及规划

黄鑫 CATLASS仓Maintainer



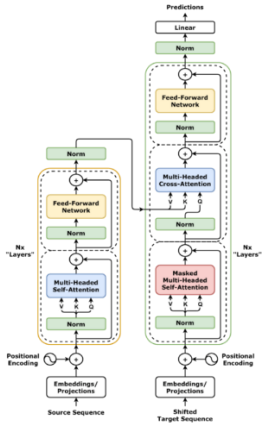
CANN

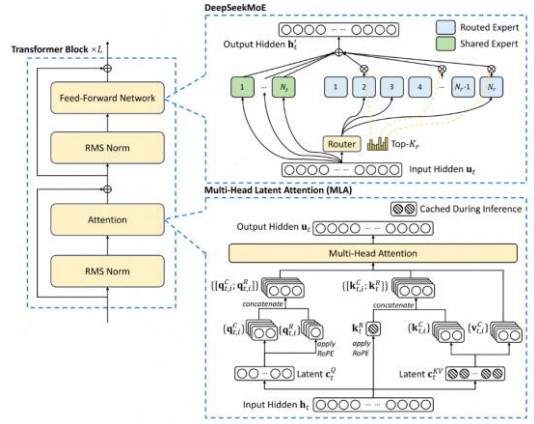
目录

问题背景：GEMM类算子开发难度较高，开发周期长

问题背景

Transformer架构特点：当前大模型基于Transformer架构设计，**GEMM相关计算(Matmul/FA/Conv及其融合)**占据其中主要运算部分，优化其性能对提升整体系统效率至关重要





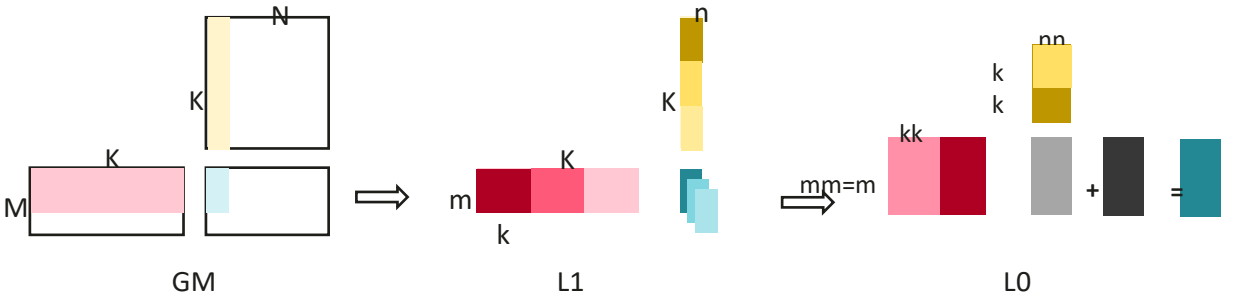
GEMM类算子开发存在难点

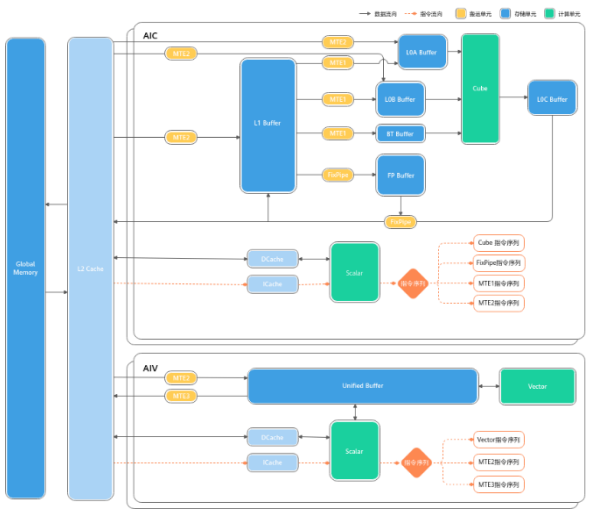
- 形状变种多
- 功能变种多
- 底层指令学习难度大

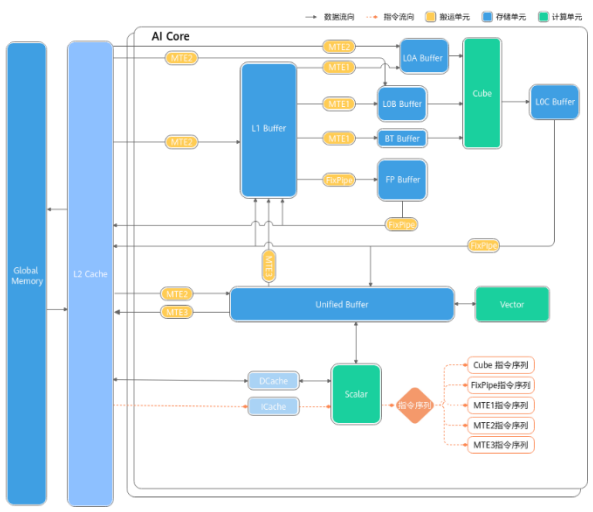
核心诉求

- 模块拆解可复用
- 提取共性，屏蔽差异

→ **高效开发**
极致性能定制算子







https://www.hiascend.com/document/detail/zh/CANNCommunityEdition/83RC1/opdevg/Ascendcopdevg/atlas_ascendc_10_0011.html

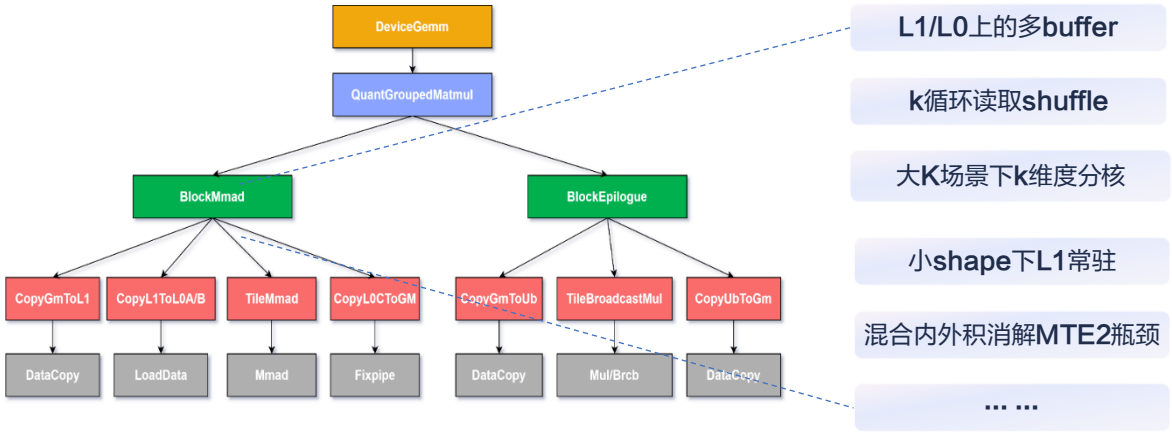
解决方案：推出CATLASS昇腾算子模板库，提升GEMM开发效率

CATLASS：CANN Templates for Linear Algebra Subroutines

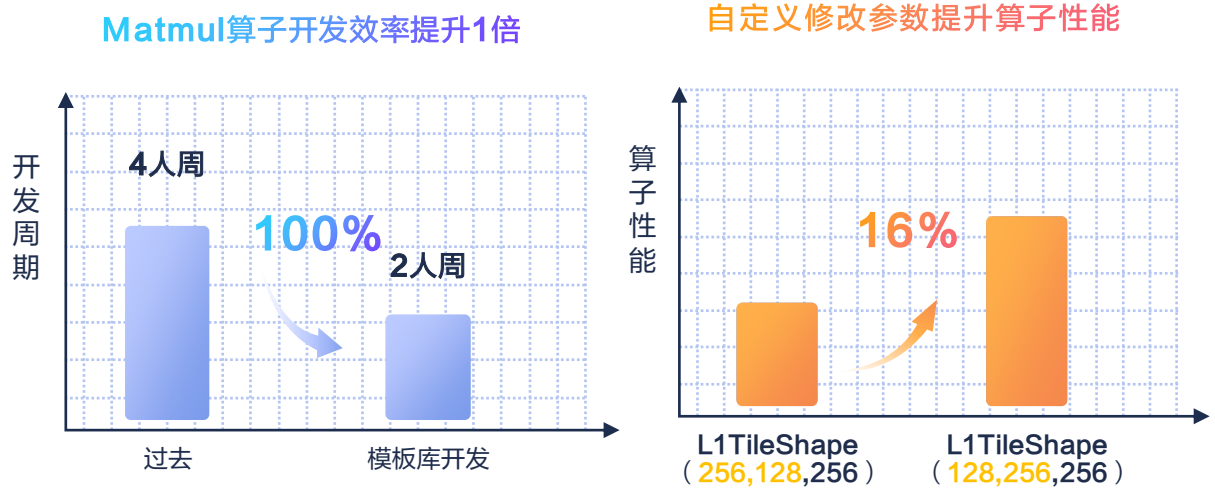
按照计算粒度，提供各层级源码，支持灵活组合

https://gitcode.com/cann

模块化设计，分层复用



支撑开发效率与优化效率提升



CATLASS模板库的实现：算子模板如何拼装

基础使用：组件快速拼装

```
using ArchTag = Arch::AtlasA2;
// Block level, define BlockMmad
constexpr bool enableUnitFlag = true;
using MmadDispatchPolicy =
      Gemm::MmadAtlasA2Pingpong<enableUnitFlag>;
```

①指定平台与物理层模块

```
using L1TileShape = GemmShape<128, 256, 256>;
using L0TileShape = GemmShape<128, 256, 64>;
using AType = Gemm::GemmType<half, LayoutA>;
using BType = Gemm::GemmType<half, LayoutB>;
using CType = Gemm::GemmType<half, LayoutC>;
using BlockMmad = Gemm::Block::BlockMmad<MmadDispatchPolicy, L1TileShape,
L0TileShape, AType, BType, CType>;
// Block level, define BlockEpilogue
```

②配置切分大小和数据类型

```
using EpilogueDispatchPolicy = Epilogue::EpilogueAtlasA2ElemWiseOneSource;
using XType = CType;
using DType = CType;
using ComputeType = CType;
constexpr uint32_t computeLength = 16384;
using TileElemWiseEpilogue = Epilogue::Tile::TileElemWiseAdd<ArchTag,
ComputeType, computeLength>;
using EpilogueTileCopy = Epilogue::Tile::TileCopy<ArchTag, CType, XType,
DType>;
using BlockEpilogue = Epilogue::Block::BlockEpilogue<EpilogueDispatchPolicy,
CType, XType, DType,
  TileElemWiseEpilogue, EpilogueTileCopy>;
std::vector<fp16_t> hostD(lenD);
// Define BlockScheduler
// Swizzle offset is 3 and direction is 0.
using BlockScheduler = typename Gemm::Block::GemmIdentityBlockSwizzle<3, 0>;
```

③配置后融合与遍历策略

```
// Kernel level
using MatmulKernel = Gemm::Kernel::MatmulEpilogue<BlockMmad, BlockEpilogue,
BlockScheduler>;
// Prepare params
typename MatmulKernel::Arguments arguments{
 options.problemShape, sizeof(half), deviceA, deviceB, deviceD};
using MatmulAdapter = Gemm::Device::DeviceGemm<MatmulKernel>;
MatmulAdapter matmul_op;
```

④组装完成算子配置

进阶使用：实现按需定制

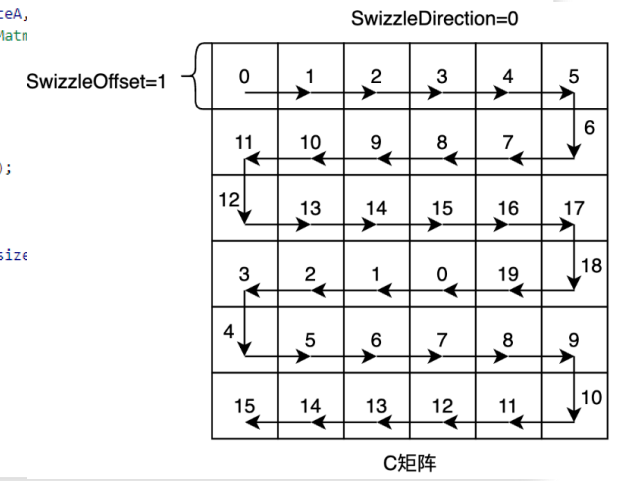
```
if (options.problemShape.k() > options.problemShape.n()) {
    constexpr uint32_t preloadStages = 1;
    constexpr uint32_t l1Stages = 2;
    constexpr uint32_t l0AStages = 2;
    constexpr uint32_t l0BStages = 4;
    constexpr uint32_t l0CStages = 1;
    constexpr bool enableUnitFlag = true;
    constexpr bool enableShuffleK = true;

    using ArchTag = arch::AtlasA2;
    using DispatchPolicy = matmul::MmadAtlasA2PreloadAsync<
        preloadStages,
        l1Stages, l0AStages, l0BStages, l0CStages,
        enableUnitFlag, enableShuffleK
    >;
    using L1TileShape = MatmulShape<256, 128, 256>;
    using L0TileShape = MatmulShape<256, 128, 64>;

    using AType = matmul::MatmulType<half, LayoutA>;
    using BType = matmul::MatmulType<half, LayoutB>;
    using CType = matmul::MatmulType<half, LayoutC>;

    using BlockMmad = matmul::block::BlockMmad<DispatchPolicy, L1TileShape, L0TileShape, AType, BType, CType>;
    using BlockEpilogue = void;
    using BlockScheduler = typename matmul::block::MatmulIdentityBlockSwizzle<3, 0>;
    // kernel level
    using MatmulKernel = matmul::kernel::GroupedMatmulM<BlockMmad, BlockEpilogue, BlockScheduler, int64_t>;

    MatmulKernel::Arguments arguments{
        options.problemShape, problemCount, deviceGroupList, deviceA,
    using MatmulAdapter = matmul::device::MatmulUniversalAdapter<Matm
    // call a kernel
    MatmulAdapter matmul_op;
    //judge arguments can run
    matmul_op.can_implement(arguments);
    // get workspace
    size_t sizeWorkspace = matmul_op.get_workspace_size(arguments);
    uint8_t *deviceWorkspace{nullptr};
    if(sizeWorkspace > 0){
        ACL_CHECK(
        aclrtMalloc(reinterpret_cast<void **>(&deviceWorkspace), size
        );
    }
    // initalize kernel argument
    matmul_op.initialize(arguments, deviceWorkspace);
    matmul_op(stream, aicCoreNum);
```

Swizzle模板类的定义

用户可以调用的接口,可修改的自定义内部实现

①按需编写遍历次序逻辑

②对接预留接口完成组装



目录

CATLASS发展历程：25年5月发布，已逐步覆盖GEMM类主要使用场景模板样例

里程碑

2025-0530
- 首次代码正式开源，Gitee托管，支持A2/A3
- 20个样例，覆盖matmul主要使用场景

2025-0930
- 基于Gitcode完成仓库搭建，完成全量代码迁移
- 覆盖30个样例，支持Conv模板
- 提供msTuner调优工具

2025-1030
- CANN社区CATLASS-SIG运营
- 已覆盖Matmul/FA/Conv主要使用场景
- 社区贡献样例5+
- 支持Matmul泛化工程样例

2025-1130

支持后融合组合机制
- 覆盖35个样例
- 新增5+说明文档

2025-1230

扩展支持950平台
- 覆盖40个样例
- 支持Atlas推理系列

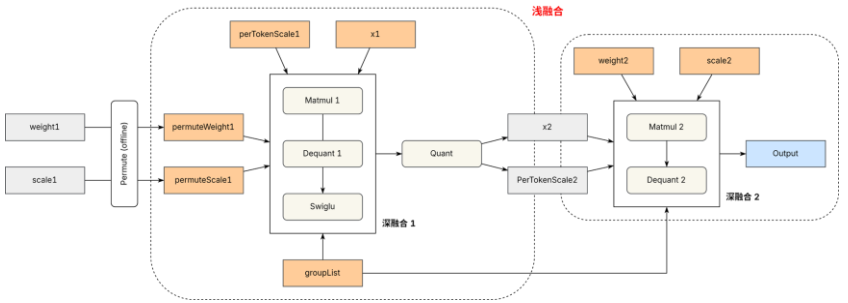
2025年5月KADC大会正式发布

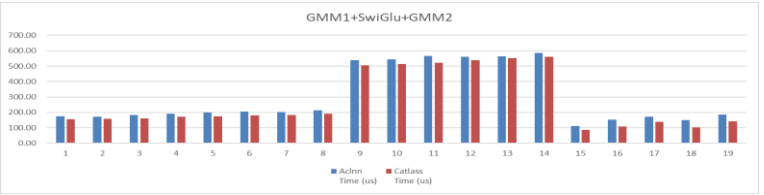
https://gitcode.com/cann/catlass

生态伙伴贡献模板样例5+
联合生态联合共建CANN生态

- 合作贡献者
  - 华南理工大学 陆璐教授团队
  - 科大讯飞 研究院工程组

产业落地多家企业
支撑定制算子极致性能提升





CATLASS代码仓内容

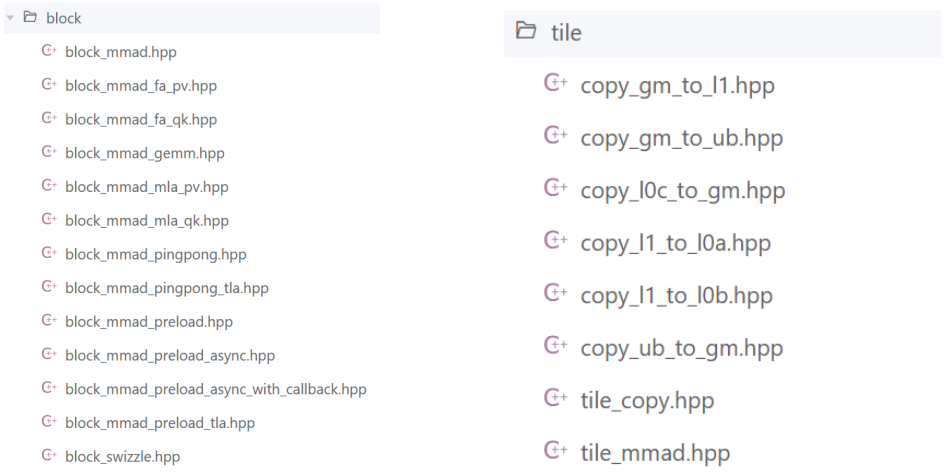
模板样例

模板算子样例30+，覆盖主要应用场景

分类	数量	样例名称举例
Matmul	17	basic_matmul，quantmatmul，splitkmatmul
Mamtul后融合	4	matmul+add，matmul+gelu
Grouped Matmul	6	grouped_matmul_slice_m
Gemm/Gemv	4	gemm，gemv
FA	2	mla, flash_attention_infer
Conv	2	conv_bias, Conv2d

模板组件

上库预置模板组件30+



调优工具

MsTuner工具针对CATLASS工具提供快速调优能力

接入示例

提供Python Extension模型接入示例

python扩展

为方便开发者使用CATLASS算子，代码仓基于pybind11和torch提供了使用python调用CATLASS算子的示例.

代码结构

```
python_extension
├── CMakeLists.txt                    # CMake配置文件
├── README.md                         # 说明文档
├── pyproject.toml                    # 项目配置文件
├── setup.py                          # 安装脚本
├── src
│   ├── bindings
│   │   ├── pybind_bindings.cpp       # pybind11绑定文件
│   │   └── torch_bindings.cpp        # torch绑定文件
│   ├── include
│   │   └── wrapper
│   │       └── catlass_kernel_wrapper.h    # wrapper头文件
│   └── wrapper
│       └── catlass_kernel_wrapper.cpp      # catlass算子wrapper文件
├── tests
│   └── test_python_extension.py      # 测试脚本
└── torch_catlass
    └── __init__.py                   # 初始化入口，用于打包
```

```python
import torch_catlass
import torch
import torch_npu
from torch_npu.testing.testcase import TestCase, run_tests

class CatlassTest(TestCase):
    def test_basic_matmul(self):
        a = torch.ones((2, 3)).to(torch.float16).npu()
        b = torch.ones((3, 4)).to(torch.float16).npu()
        result = torch_catlass.basic_matmul(a, b, "float16")
        golden = torch.mm(a, b)
        self.assertRtolEqual(result, golden)
    def test_basic_matmul_torch_lib(self):
        a = torch.ones((2, 3)).to(torch.float16).npu()
        b = torch.ones((3, 4)).to(torch.float16).npu()
        torch.ops.load_library("../../output/python_extension/libcatlas
        result = torch.ops.CatlassTorch.basic_matmul(a, b, "float16")
        golden = torch.mm(a, b)
        self.assertRtolEqual(result, golden)

if __name__ == "__main__":
    run_tests()
```

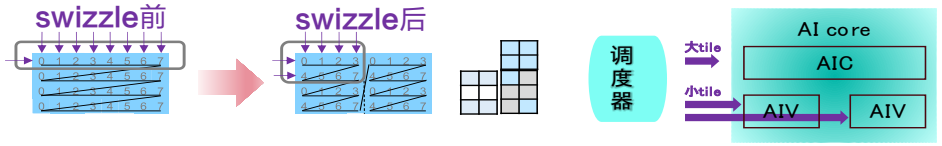
目录

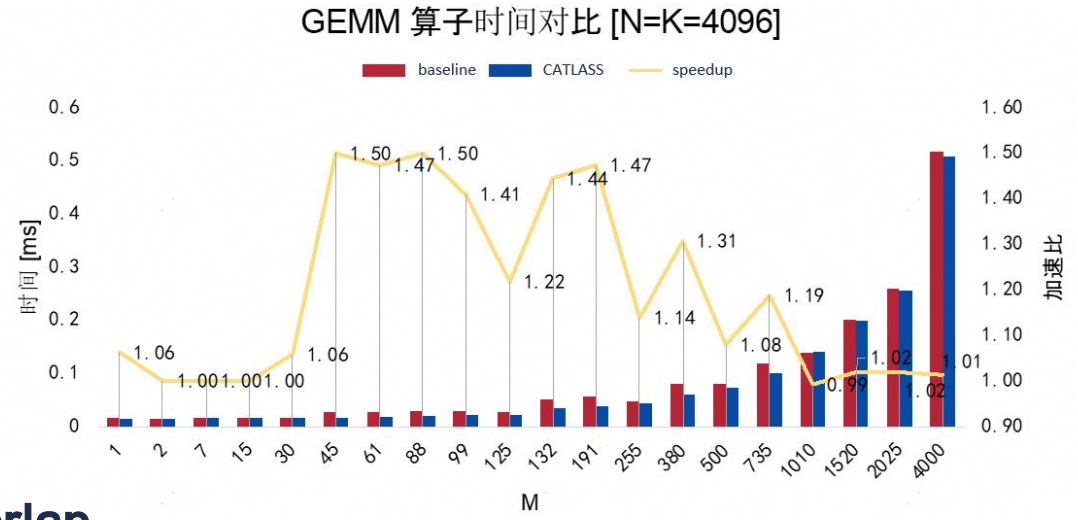
CATLASS算子模板库赋能科研应用创新：无问芯穹算子优化团队

GEMM/GroupGEMM

GEMM/GroupGEMM优化

- 精细的L2->L1->L0数据搬运和复用策略
- 异构任务映射充分利用AIC计算核心和AIV计算核心
- 负载均衡和自适应策略：根据任务的规模自适应调整

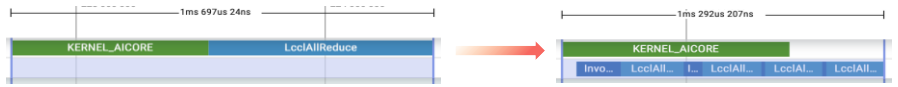




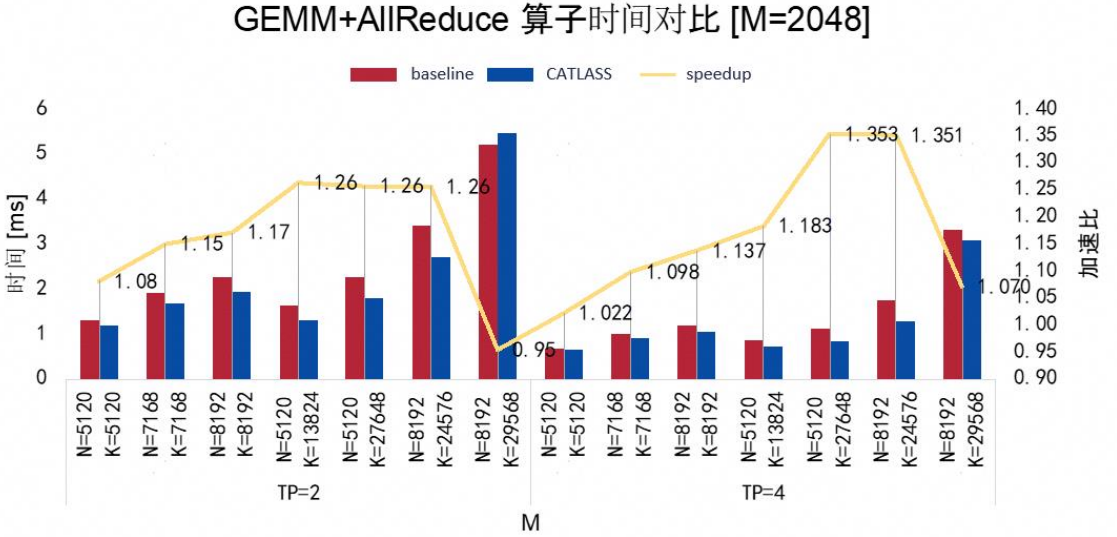
FlashOverlap

基于CATLASS库完成通算实现

- 基于CATLASS+HCCL接口完成通算实现
- FlashOverlap在昇腾上，GEMM+AllReduce性能相比基线提升高达35%

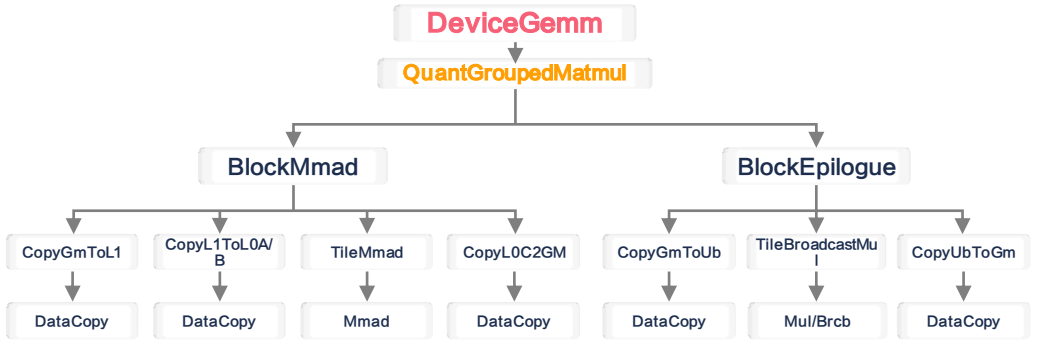


acINN+HCCL Timeline　　FlashOverlap Timeline



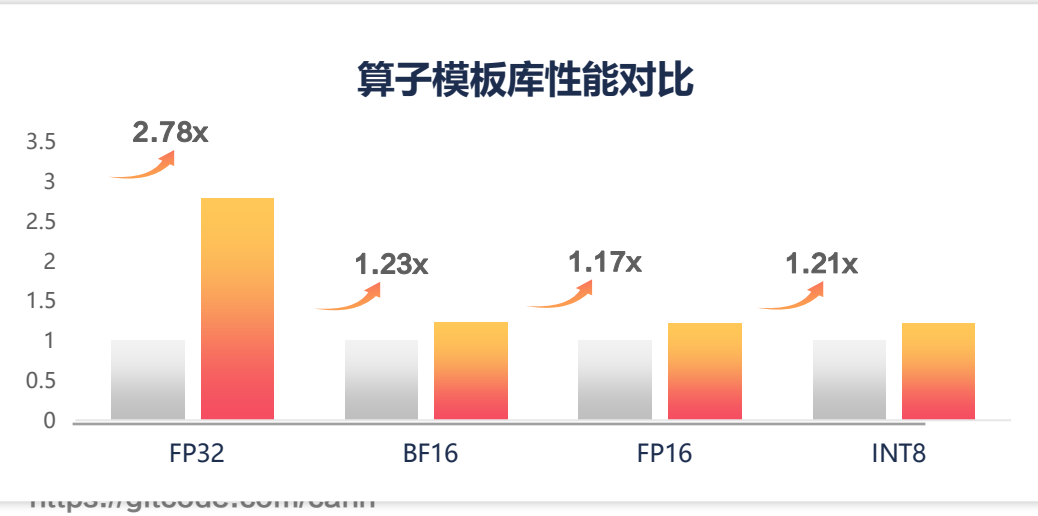
CATLASS算子模板库赋能科研应用创新：华南理工大学陆璐教授团队

Gemm算子基础组件分层，敏捷开发

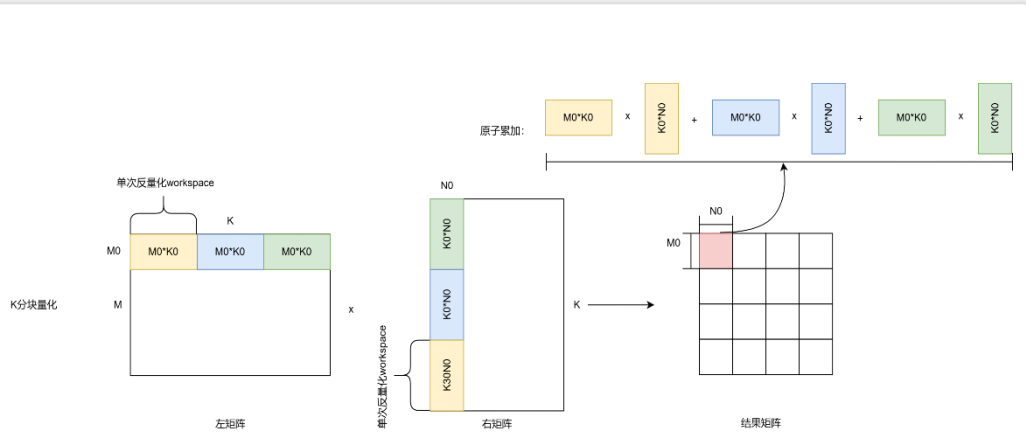


设计分层架构，灵活支持性能优化

驱动开发效率提升，赋能算子性能优化



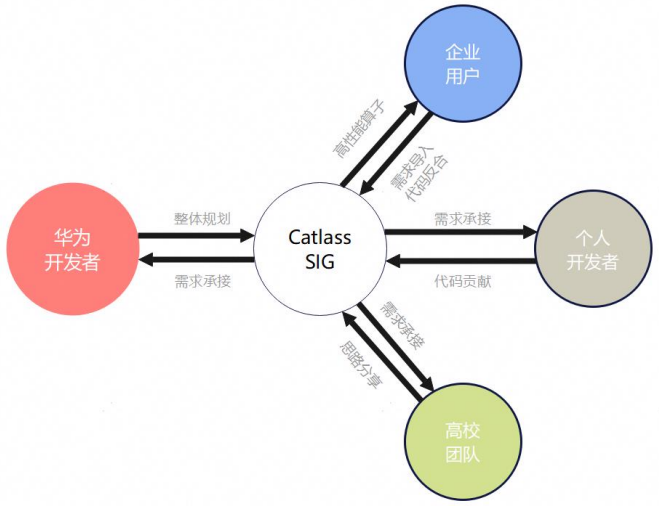
基于模板库创新突破，根据内存状态动态切分



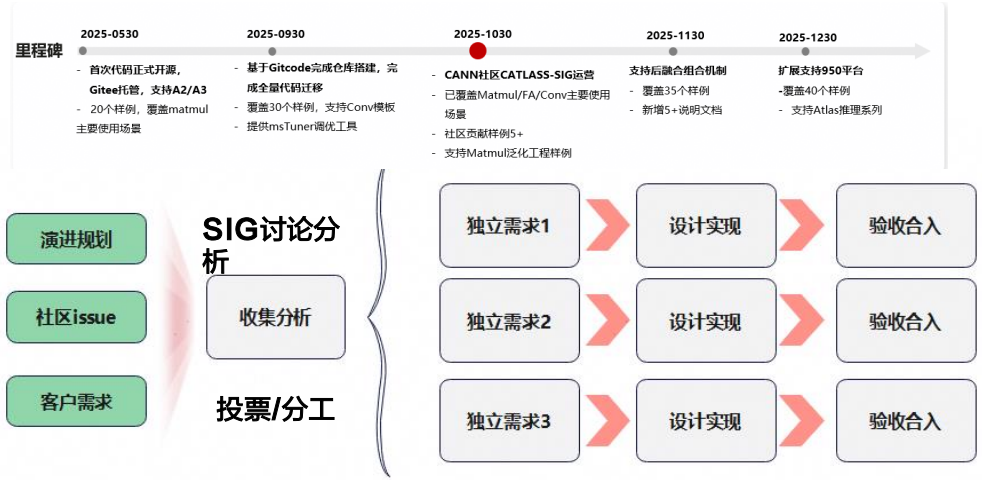
目录

CATLASS SIG已启动运作：快速迭代，社区贡献繁荣生态

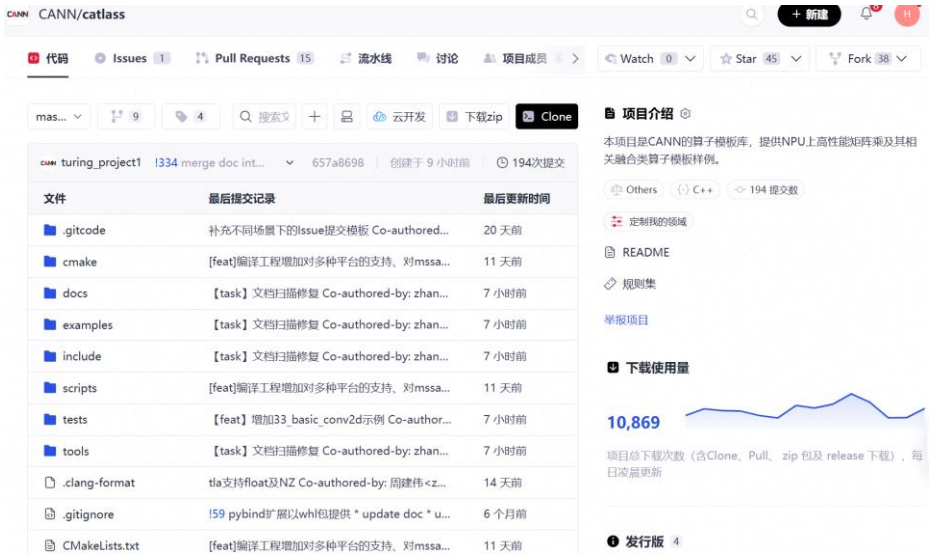
基于SIG组运作，与业界学界共同讨论需求规划



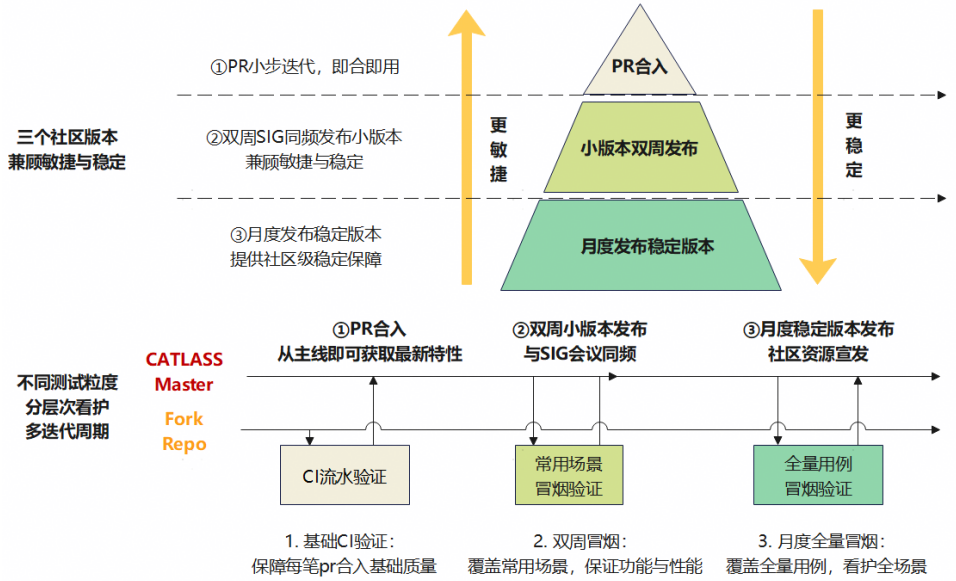
提前公开发布路标节点，吸纳外部开发者参与贡献



Gitcode社区开源，直接获取最新特性



小快灵敏捷迭代，分层次满足用户多样需求



欢迎开发者使用和贡献

扫码关注代码仓

扫码订阅SIG会议

代码仓地址：https://gitcode.com/cann/catlass

文档地址：https://gitcode.com/cann/catlass/tree/master/docs