Graph-Engine开源

--暨Graph-Engine开源介绍以及优秀实践

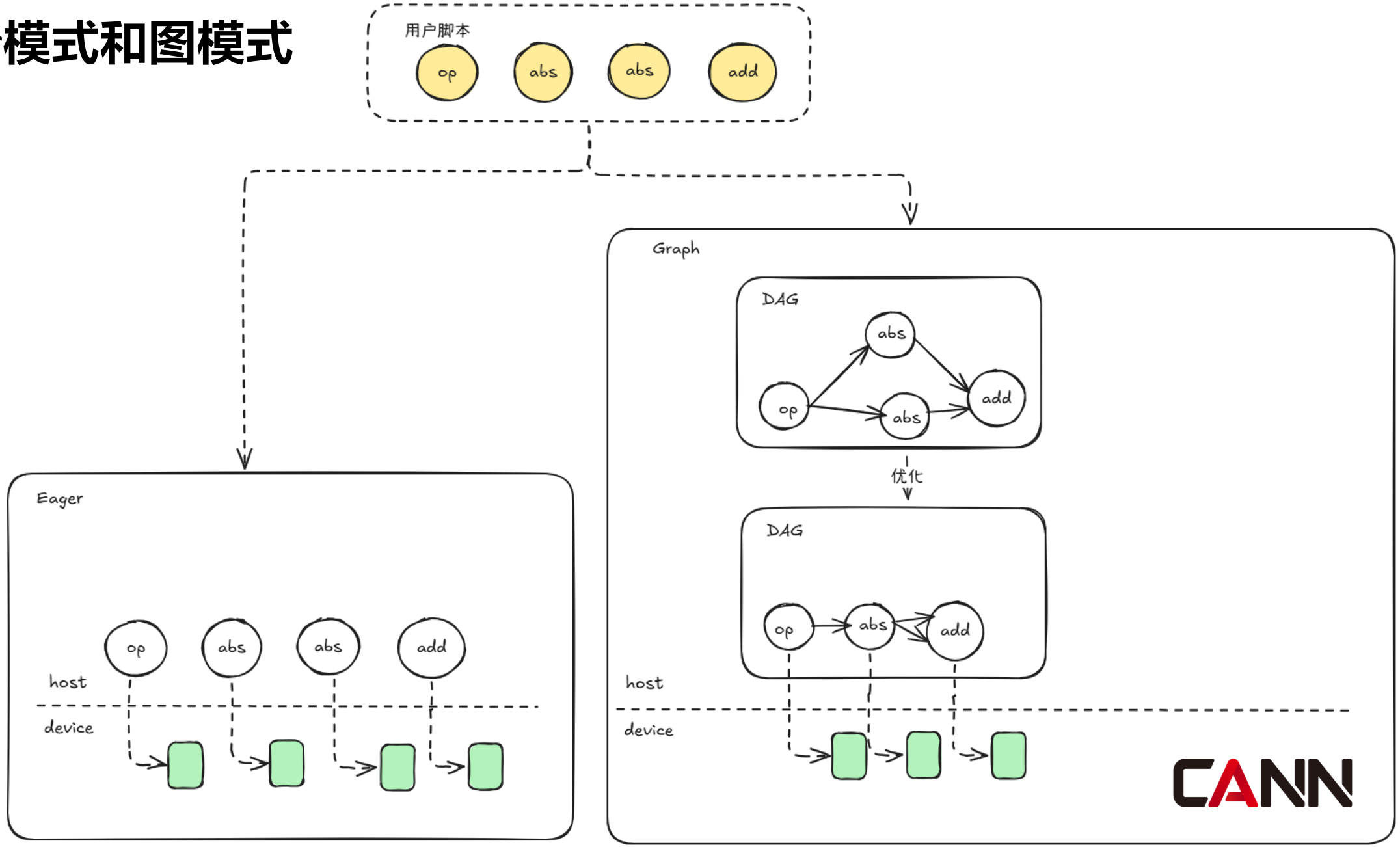
作者：赵鑫鑫、黄桂军

时间：2025.12.26

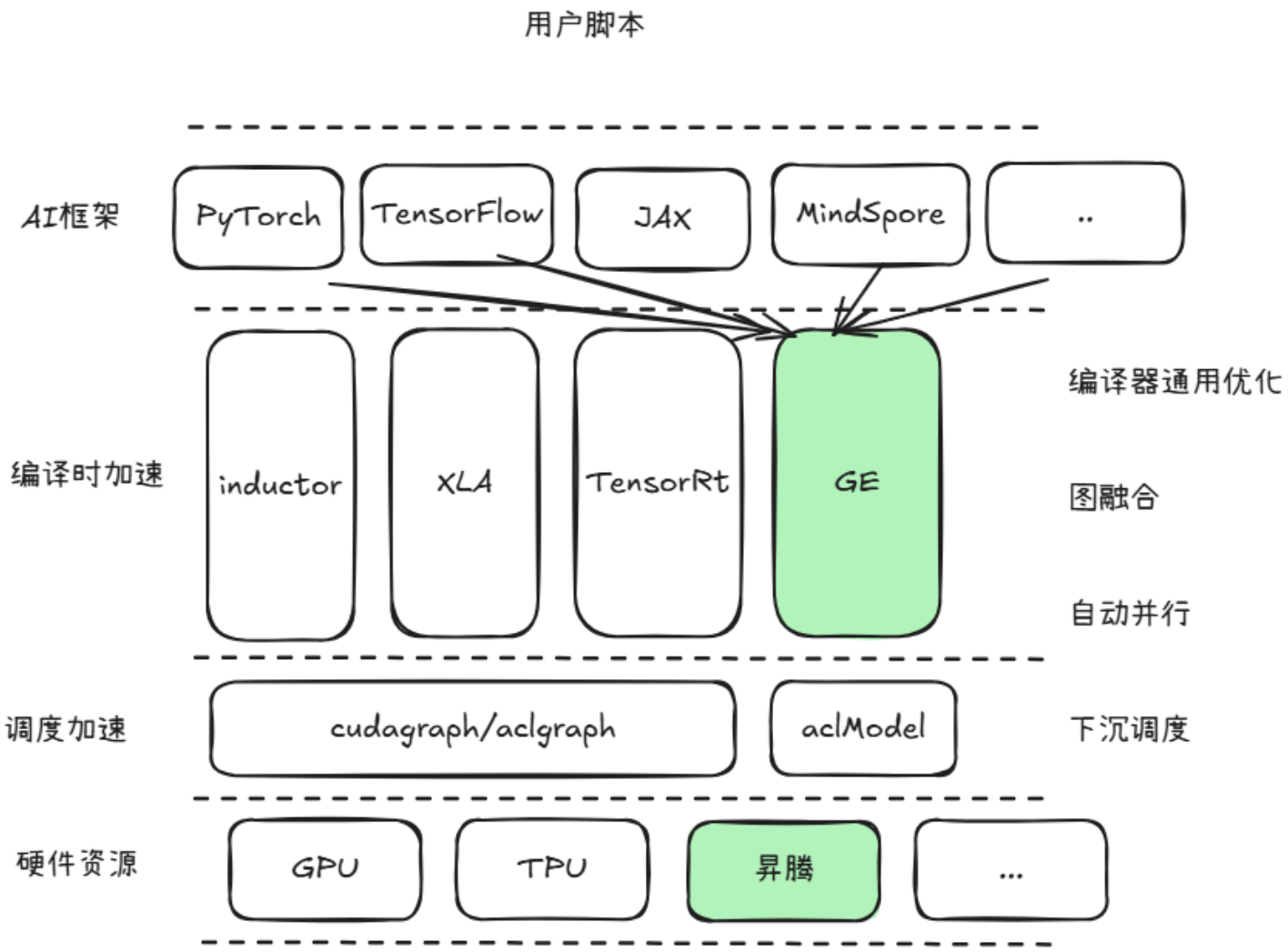
https://gitcode.com/cann/ge

CANN

Eager模式和图模式



1 Graph-Engine(GE)是什么?



- 生于CANN社区，长在昇腾上
- 图模式的加速利器
  （1）编译优化
  （2）调度优化

CANN

优秀实践：昇腾图模式性能标杆

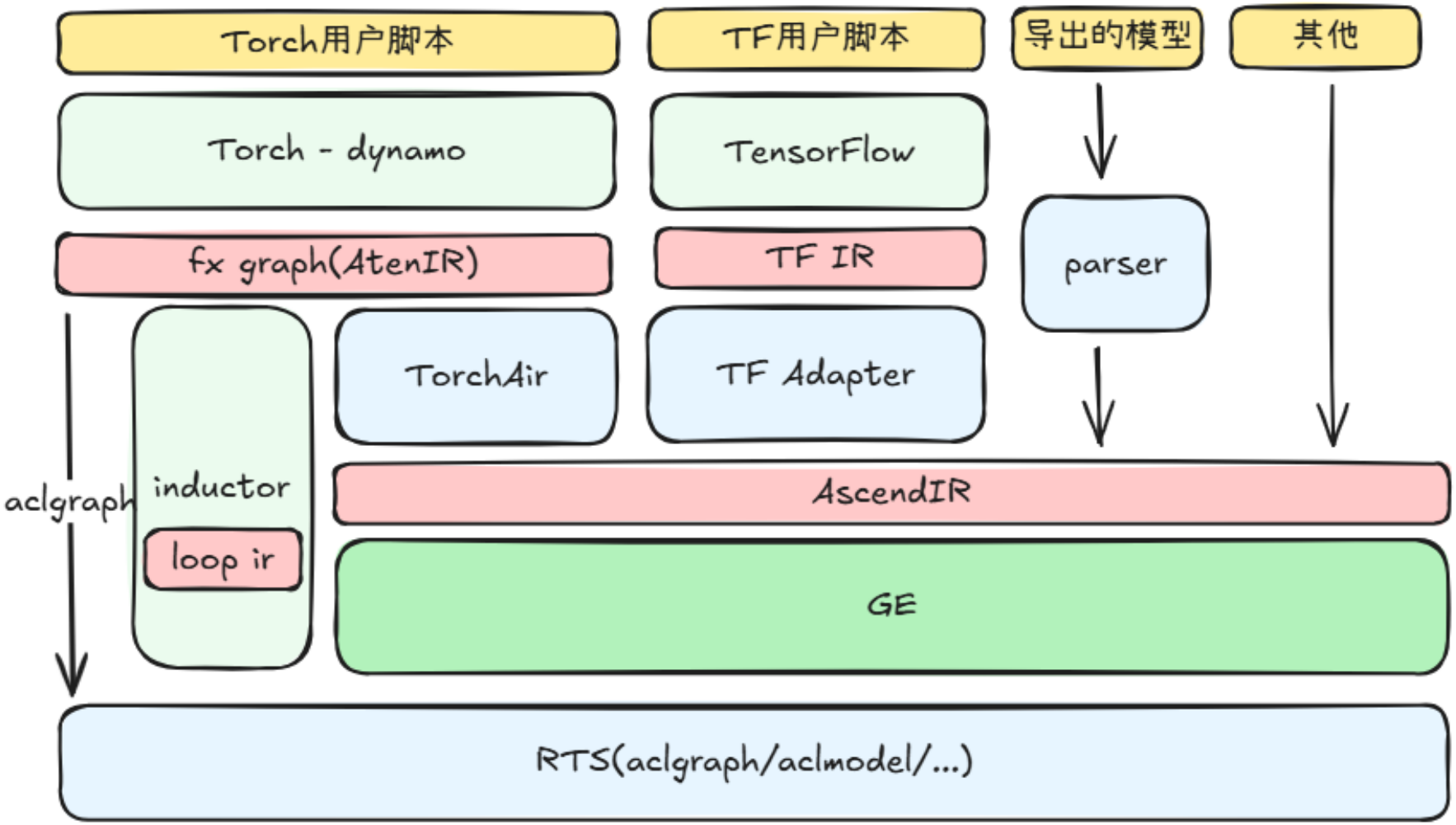
基于Atlas 800T A3

高吞吐场景：
1、DeepSeek，某互联网客户现场复现了2k + 2k 1920TPS@50ms每卡

低时延场景：
1、LongCat网络推理时延10ms
2、DeepSeek，某互联网客户的3.5k + 1k的600QPM（双机每分钟处理请求数）
3、DeepSeek，某互联网客户长序列的16k + 1k的70QPM（双机每分钟处理请求数）

CANN

2 Graph-Engine在哪里?



CANN

3 Graph-Engine为什么开源?

GE的初级用户

GE的深度用户

开源开放

We are together!

- 特性源码可见
- Sample丰富多样，快速接入
- 核心优化变成小组件，集成更方便

You are not alone!

- 有疑问交流群发消息，快速答复
- 有Bug提Issue，极速修复
- 有想法提SIG议题，广开思路

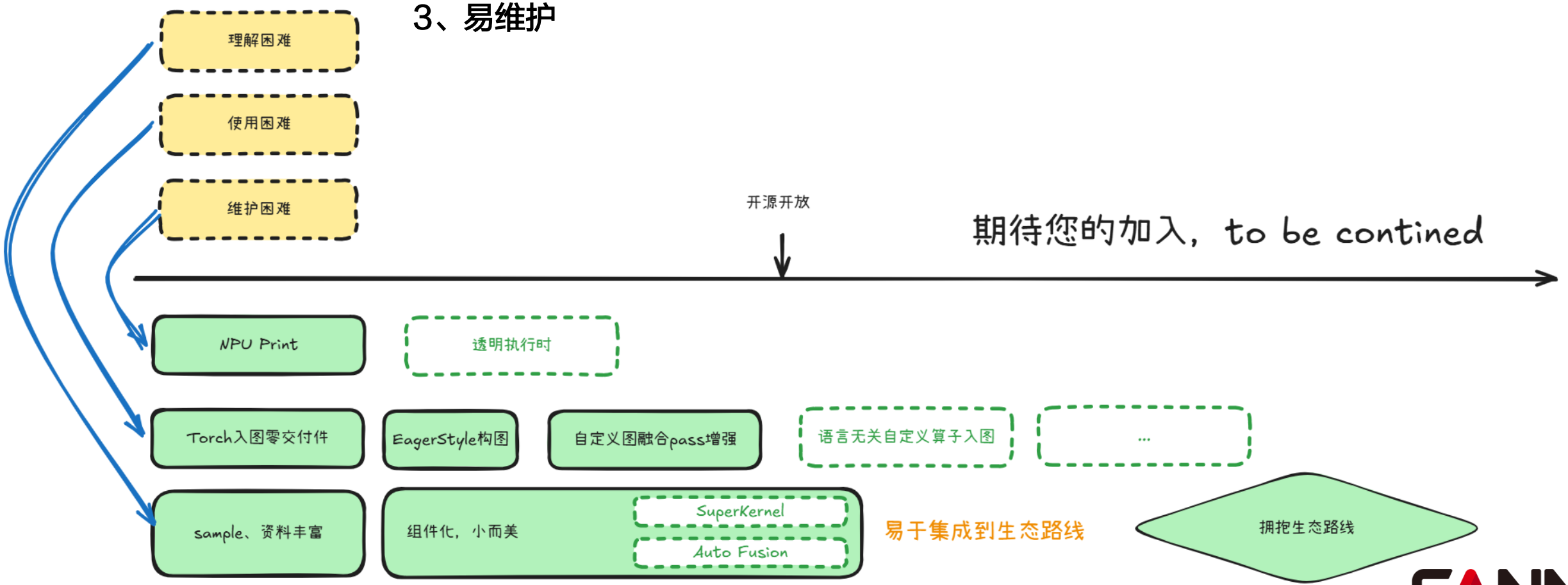
CANN

Graph-Engine开源总览

业界图模式常见问题：

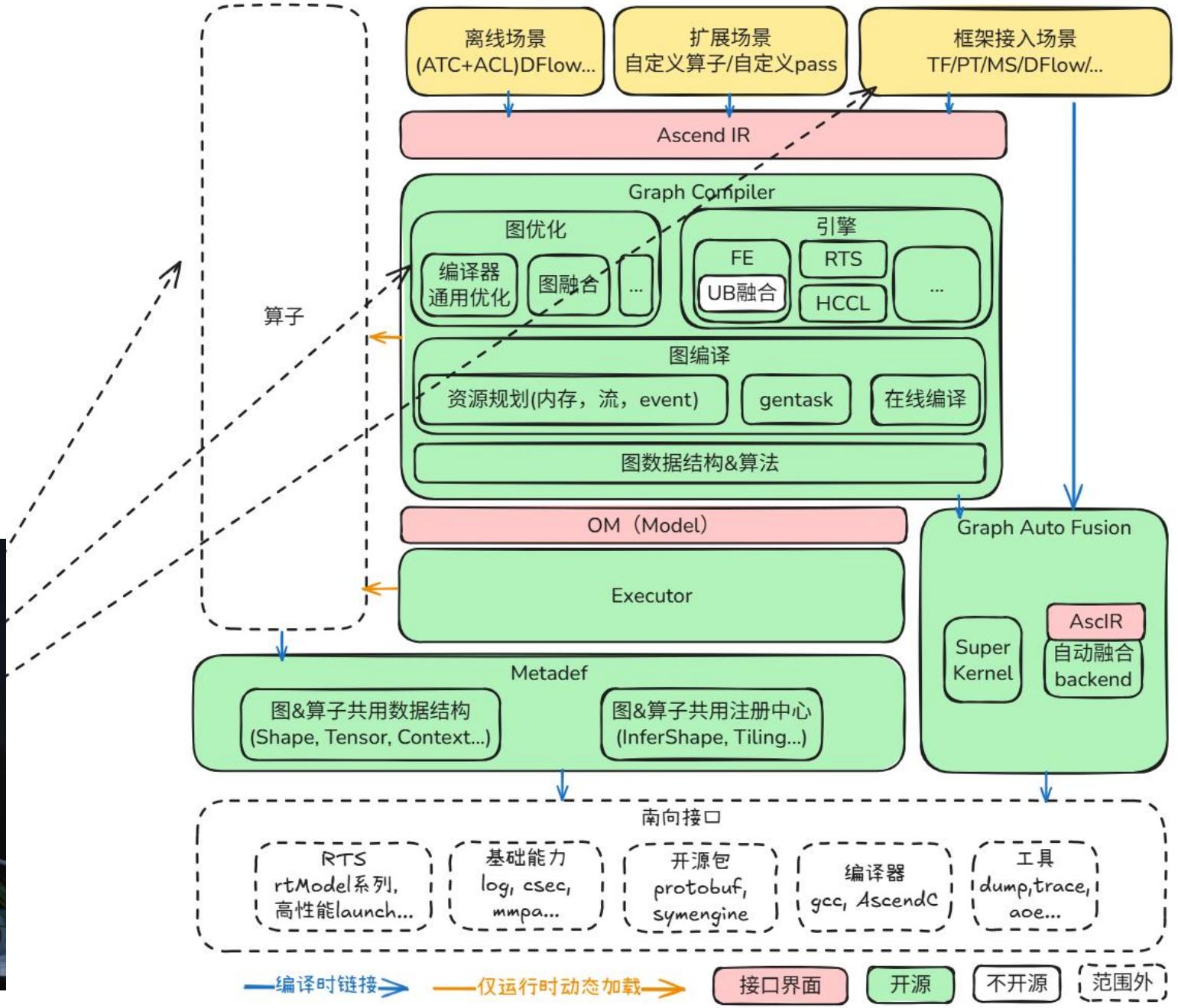
开源开放设计原则：
1、可替换
2、易接入
3、易维护

期待&展望：
1、携手共建
2、生态繁荣



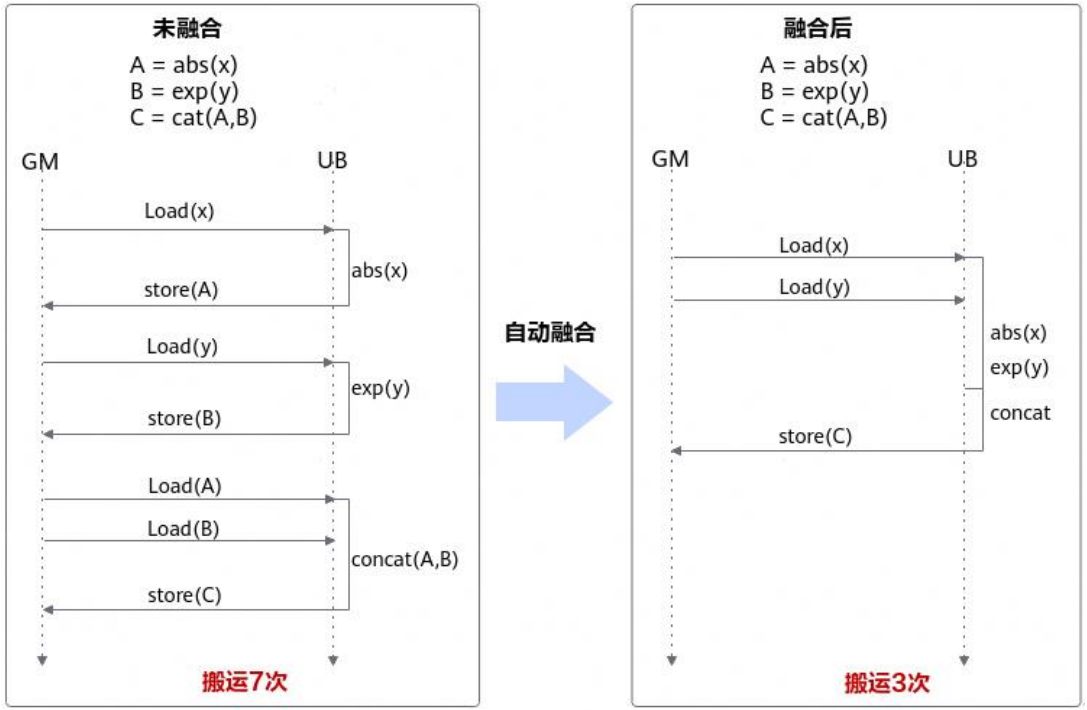
CANN

4 解剖Graph-Engine



4.1 核心技术-编译优化-图化简降算子数量

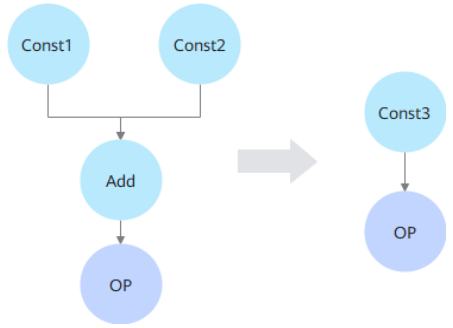
硬件相关–自动融合优化

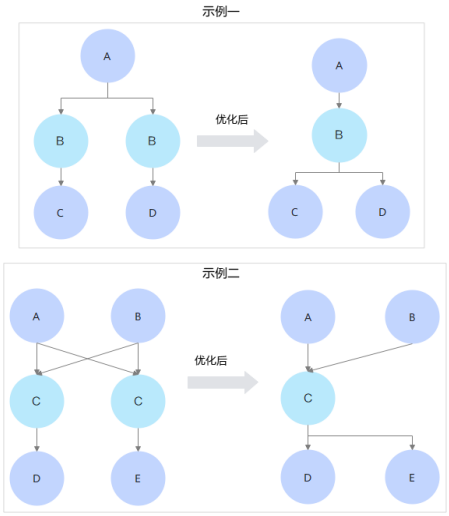


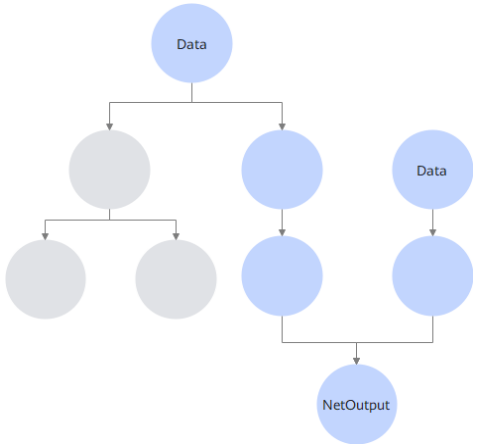
MMOE：算子数下降40%，算子耗时下降16.8%

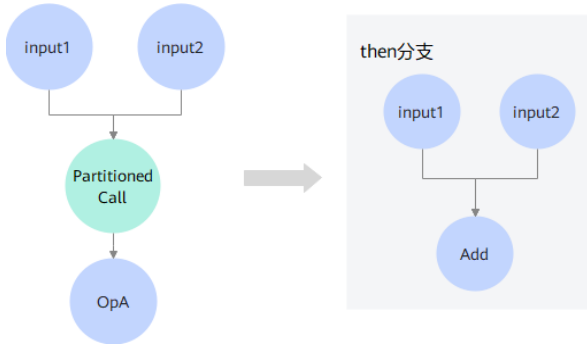
GR网络：算子数下降50.2%，算子耗时下降44.6%

硬件无关优化



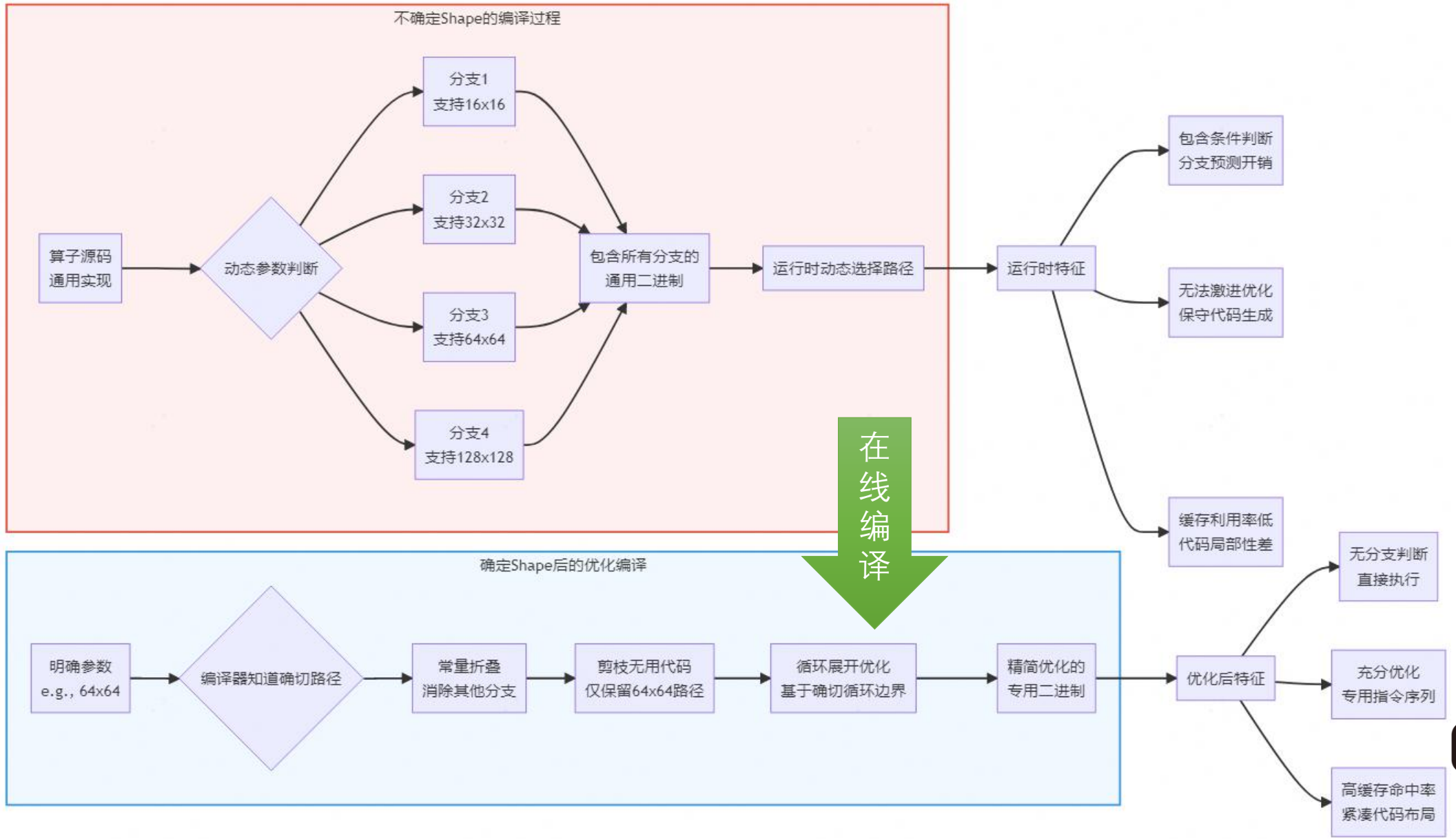






CANN

4.2 核心技术-编译优化-在线编译提kernel性能



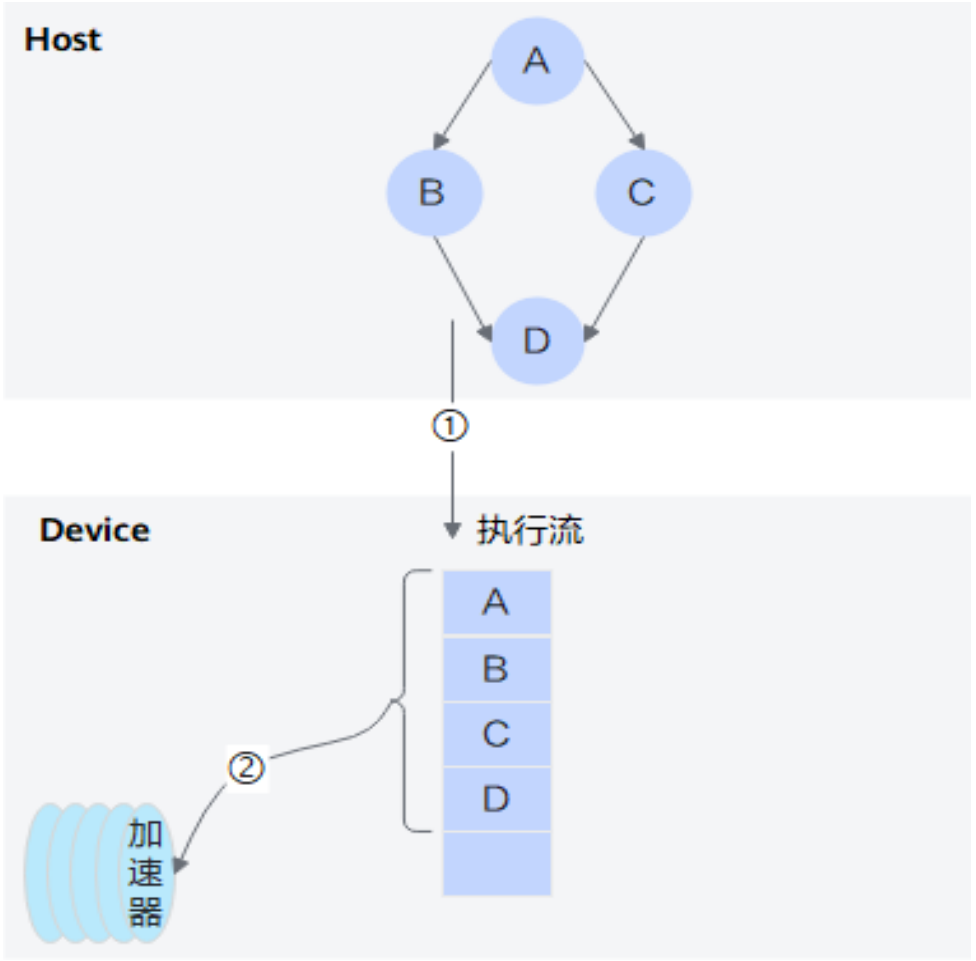
优秀实践：在线编译提升Kernel性能

GE提供默认的 Kernel 静态化编译能力，静态化编译指对于静态 Shape 的算子，除了在编译时提前计算 Tiling 结果外，还会用计算得到的 Tiling 结果，作为常量输入，与 Kernel 代码一起重新编译，利用编译器的常量折叠能力，降低 Kernel 执行时的 Scalar 计算开销。

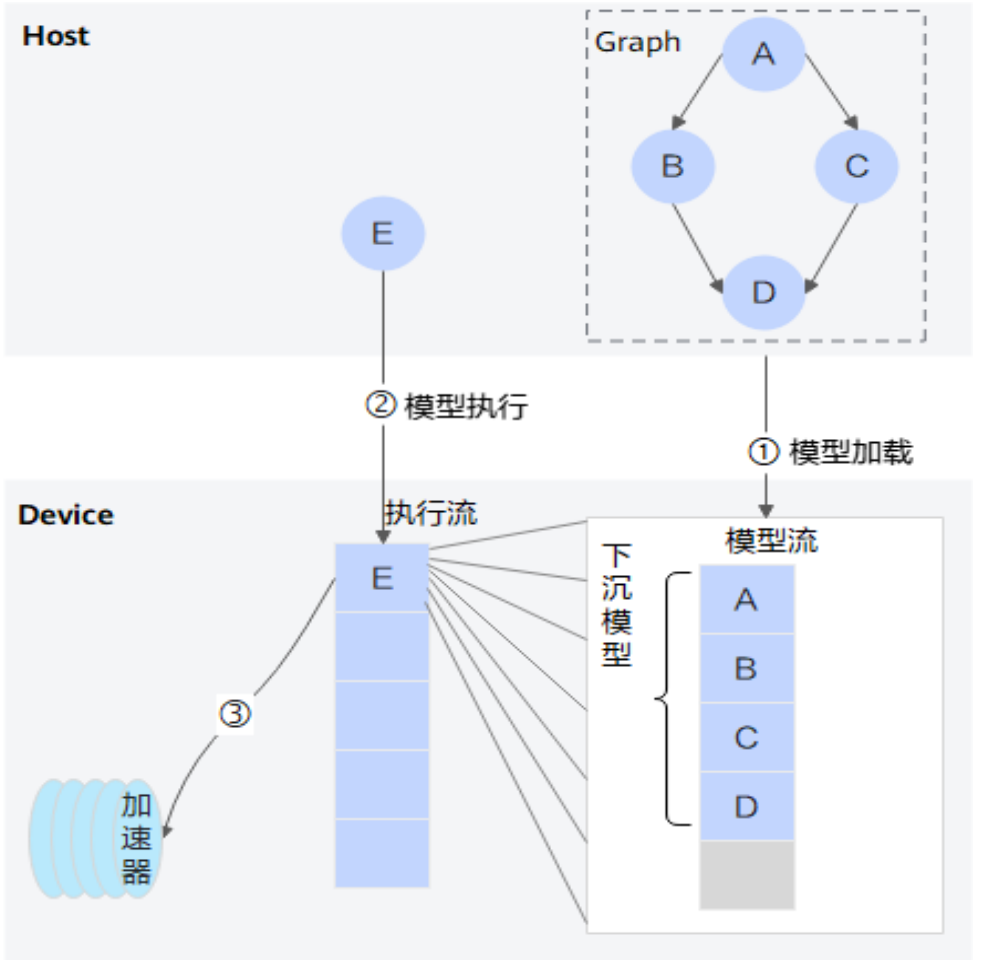
Eager执行				图模式			
OP Type	Core Type	Total Time(us)		OP Type	Core Type	Total Time(us)	
MoeDistributeDispatchV2	MIX_AIV	302367.4		GroupedMatmul	MIX_AIC	55631.173	
GroupedMatmul	MIX_AIC	52827.26		FusedInferAttentionScore	MIX_AIC	49463.77	
FusedInferAttentionScore	MIX_AIC	48955.52		MatMul	AI_CORE	35372.863	
MatMulV2	AI_CORE	35949.92		MlaPrologV2	MIX_AIC	22130.505	
MoeDistributeCombineV2	AI_VECTOR_CORE	20889.58		MoeDistributeCombineV2	AI_VECTOR_CORE	19201.821	↓8%
MlaPrologV2	MIX_AIC	19091.9		MoeDistributeDispatchV2	MIX_AIV	16030.082	
Transpose	AI_VECTOR_CORE	11361.9		TransposeBatchMatMul	AI_CORE	5017.102	
SwiGlu	AI_VECTOR_CORE	10131.54		SwiGlu	AI_VECTOR_CORE	8951.079	↓11%
BatchMatMulV2	AI_CORE	5373.62		MatMulV3	MIX_AIC	3469.532	
MatMulV3	MIX_AIC	5036.42		Add	AI_VECTOR_CORE	2327.446	
AddRmsNorm	AI_VECTOR_CORE	2193.36		InplaceAddRmsNorm	AI_VECTOR_CORE	1670.993	
MoeGatingTopK	AI_VECTOR_CORE	2108		MoeGatingTopK	AI_VECTOR_CORE	1833.755	↓13%
Add	AI_VECTOR_CORE	2025.92		AddRmsNormCast	AI_VECTOR_CORE	1081.959	
Cast	AI_VECTOR_CORE	1986.6		StridedSliceD	AI_VECTOR_CORE	171.983	
Range	AI_VECTOR_CORE	1815.7		ConcatV2D	AI_VECTOR_CORE	77.582	
ConcatD	AI_VECTOR_CORE	329.06		Data	AI_VECTOR_CORE	47.001	
RmsNorm	AI_VECTOR_CORE	31.22		RmsNorm	AI_VECTOR_CORE	28.861	↓7%
GatherV2	AI_VECTOR_CORE	17.9		GatherV2	AI_VECTOR_CORE	15.3	↓14.5%

4.3 核心技术-调度优化-下沉调度降host开销

Host调度

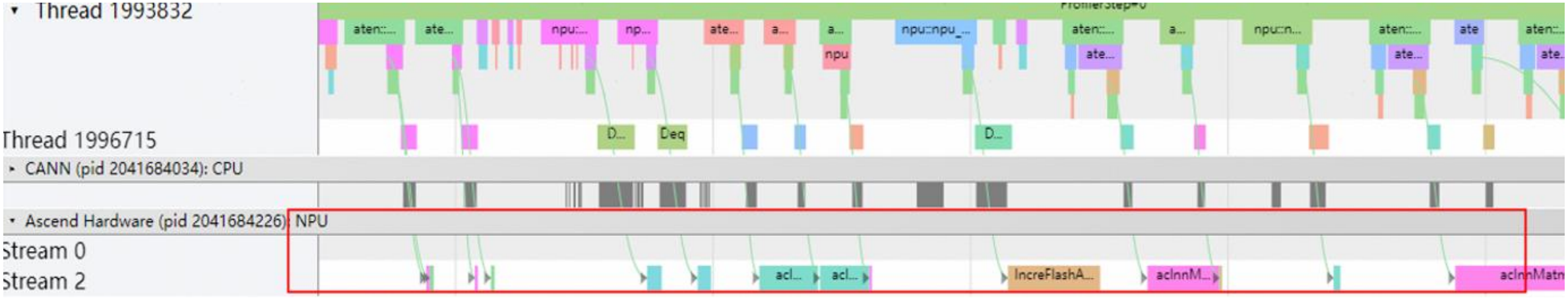


下沉调度

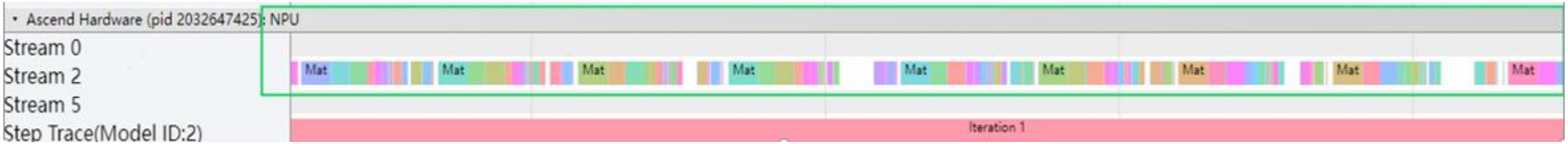


优秀实践：下沉调度消除Host下发瓶颈

Eager执行（Host调度）　LLaMA-7B Decoding profiling



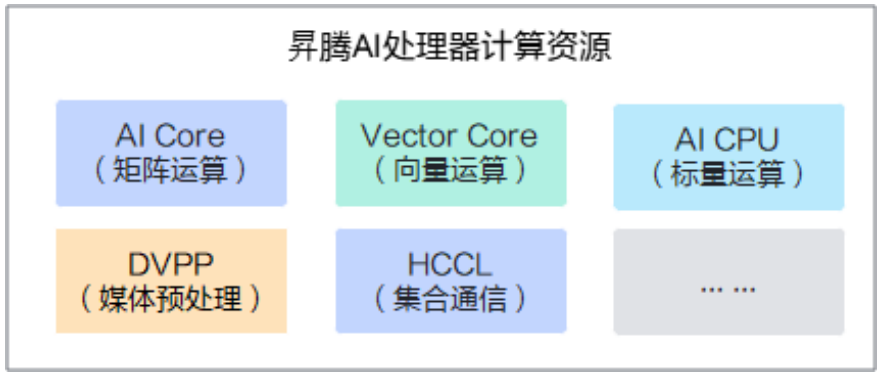
图执行-下沉调度　端到端有 18ms 的收益，吞吐量整体提升 37%

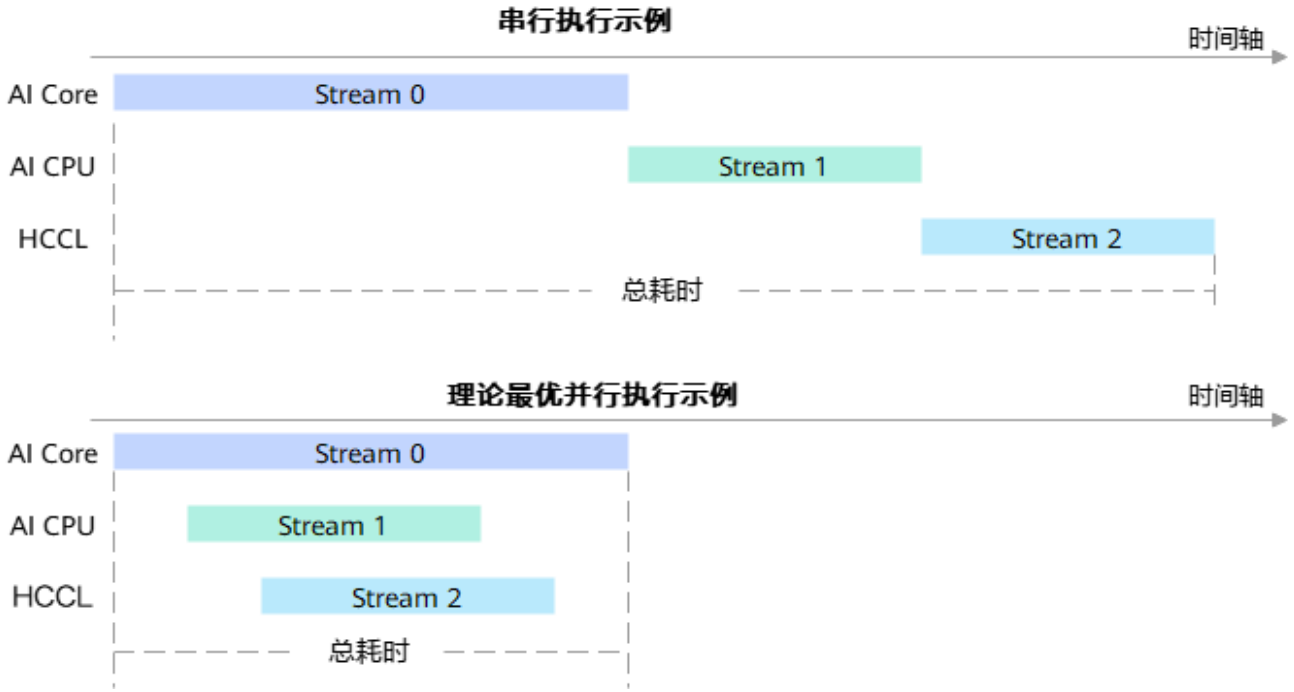


CANN

4.4 核心技术-调度优化-多流并发device满血工作

多样化的计算任务以task的形式下发到各硬件资源执行，GE（Graph Engine）图引擎采用多流并行算法，在满足计算图任务内部依赖关系的前提下，支持高效并发执行计算任务，从而大大提高硬件资源利用率和AI计算效率。



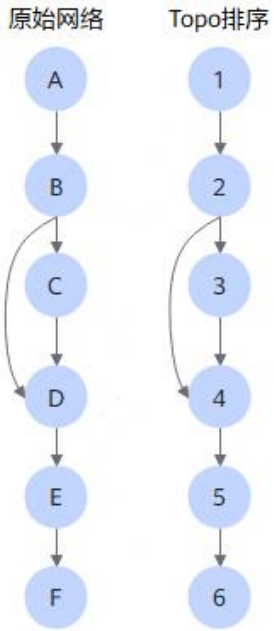


LLaMA-65B全量图，相比单流提升 30%

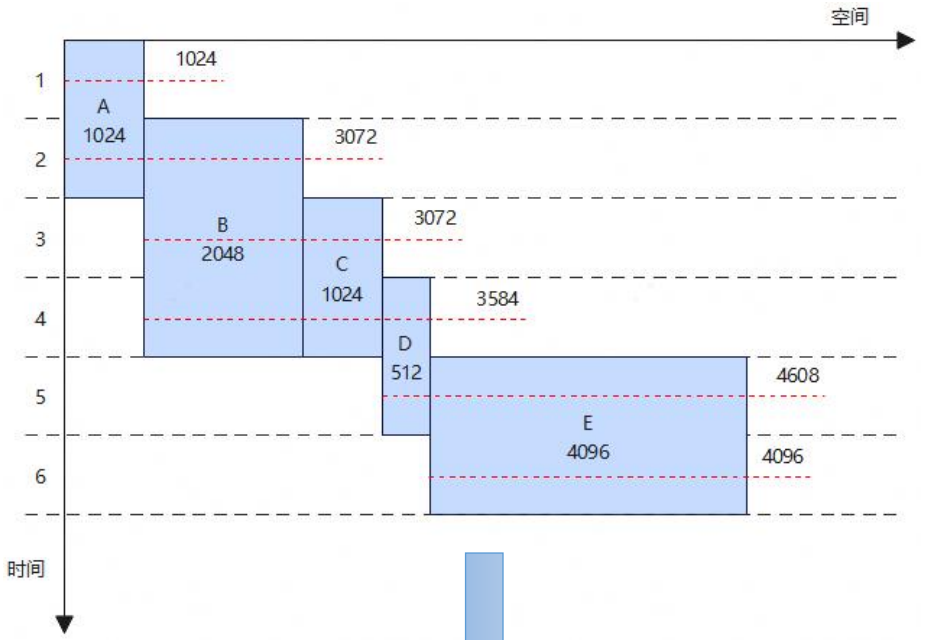
盘古71B，相比单流提升 15%

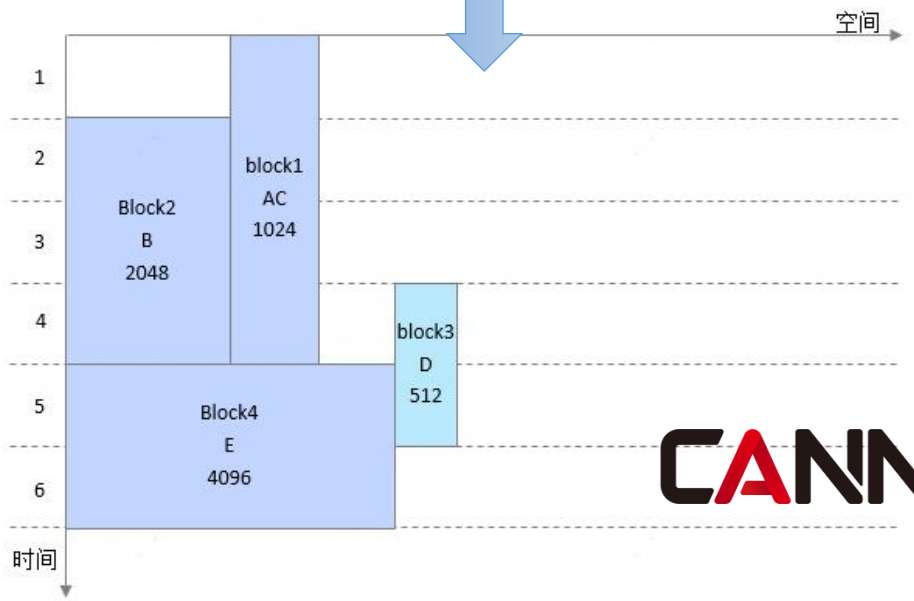
CANN

4.5 核心技术-内存复用-降内存跑更大的模型



节点	内存大小	生命周期
A	1024	[1,2]
B	2048	[2,4]
C	1024	[3,4]
D	512	[4,5]
E	4096	[5,6]
F	0	





Widedeep:
Eager 模式5.6G，GE图模式 4.6G，内存占用↓ 17.8%

CANN

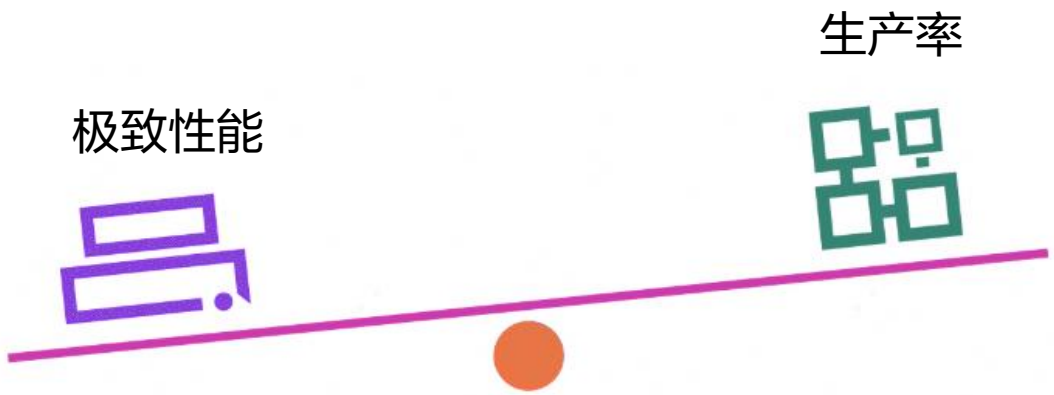
5 Graph-Engine**近期体验优化**

CANN

5 设计思想转变

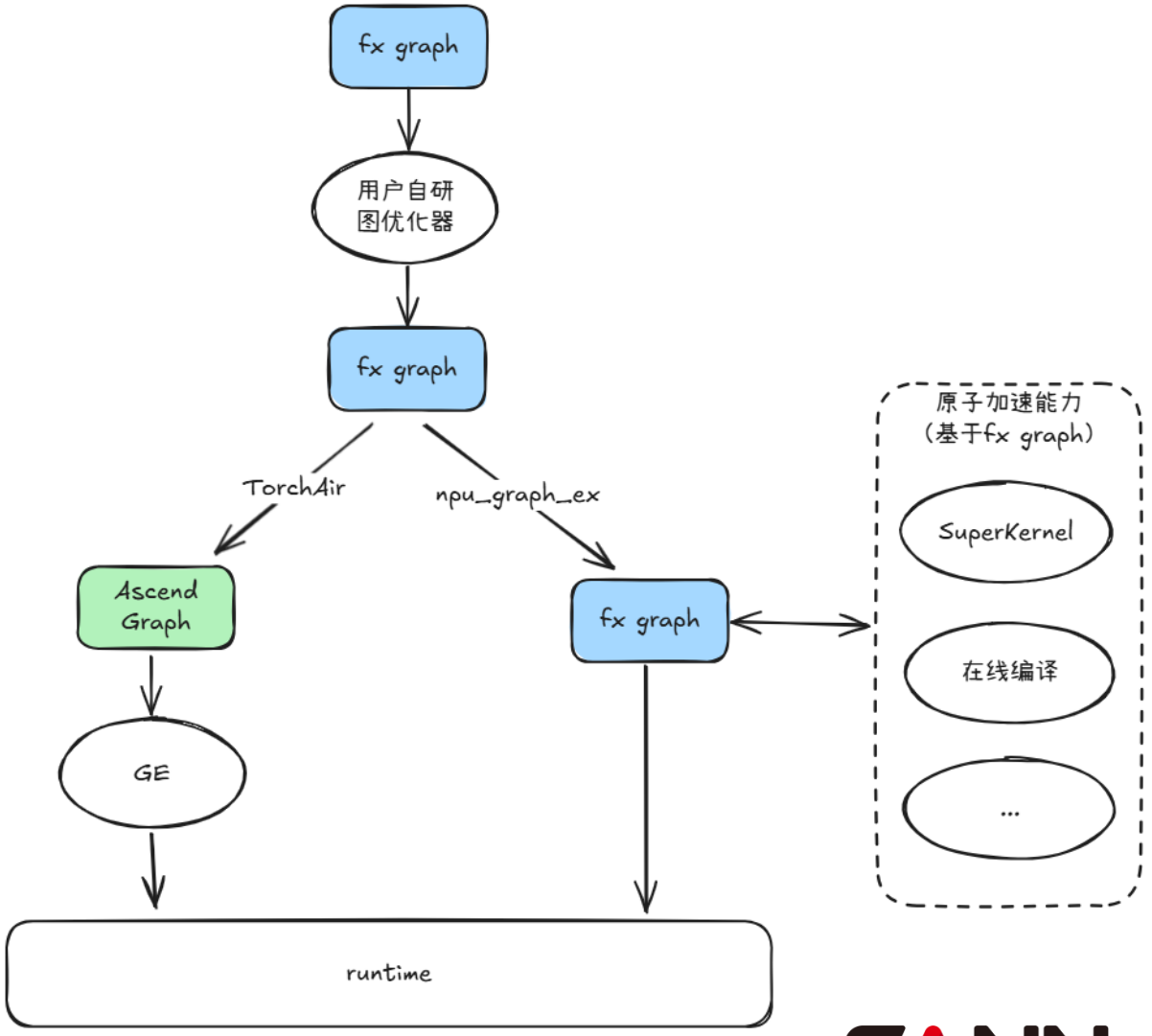
5.1 可替换

生产率和极致性能的跷跷板：



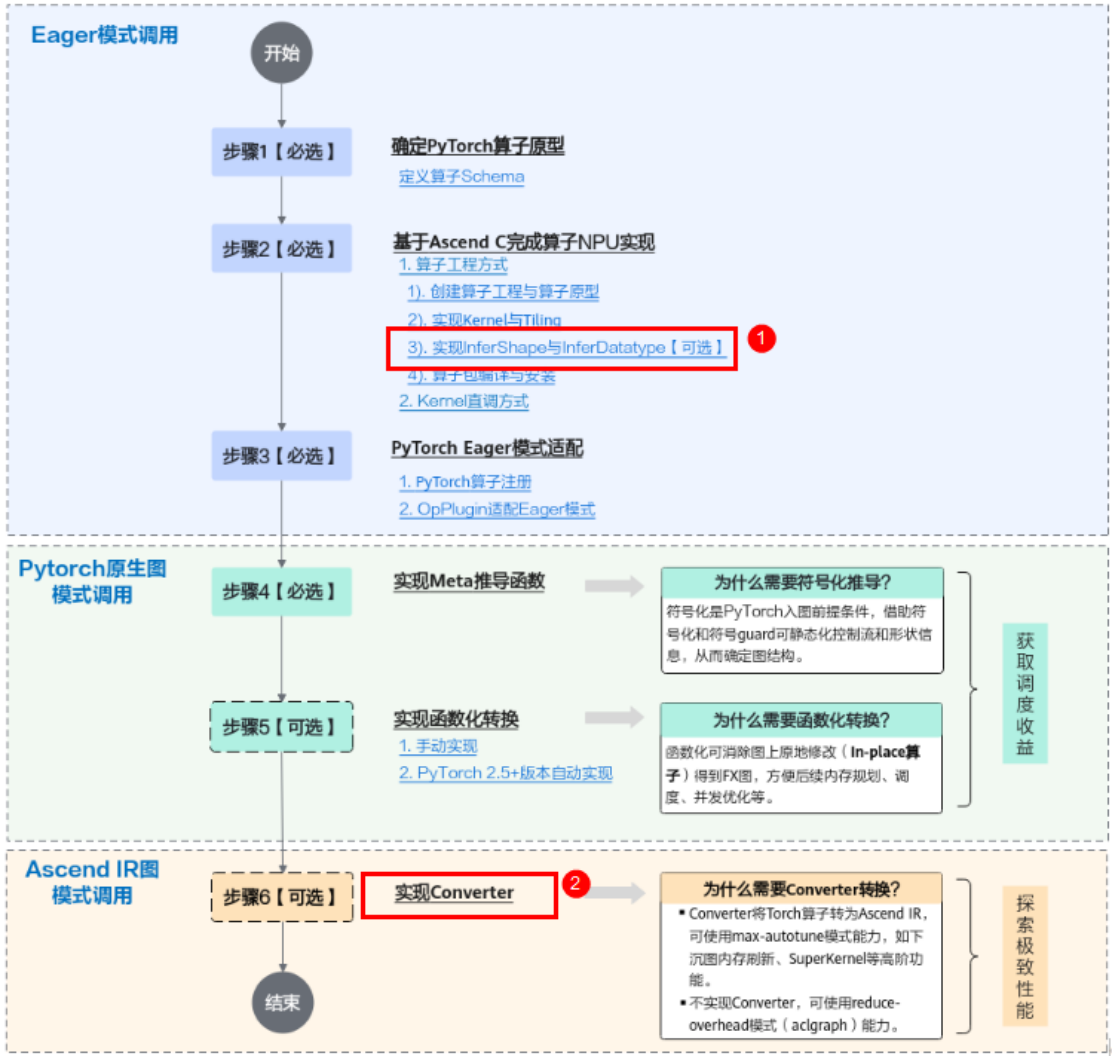
追求极致的性能往往需要复杂的技术，从而增加开发难度和时间成本；
注重开发效率可能会牺牲部分运行性能。

因此我们提供了双路径。



CANN

5.2 算子接入-PyTorch场景自定义算子入图0新增交付件



- npu_graph_ex模式
  > 与原生入图交付件一致，0新增交付件。

- TorchAir-GE模式
  > InferShape/InferDtype复用Torch推导函数自动生成。
  > AtenIR到AscendIR的转换，根据IR定义自动转换。

CANN

5.3 模型快速构建-通过 EagerStyle 构图 API 提升构图易用性

As Is：

```
void MakeSubGraphByIrAndDump() {
  // 实例化图对象
  ge::Graph graph("ir sample");
  // 实例化输入节点并设置属性
  auto data = ge::op::Data("input_0").set_attr_index(0);
  // 实例化常量节点，并设置常量的tensor属性作为携带的数据
  auto const0 = ge::op::Const("Const1");
  std::vector<int64_t> data_of_const0 = {0};
  ge::TensorDesc desc_of_const0(ge::Shape({1}), ge::FORMAT_ND, ge::DT_INT64);
  ge::Tensor tensor_of_const0(desc_of_const0, reinterpret_cast<uint8_t *>
(data_of_const0.data()), sizeof(int64_t) * data_of_const0.size());
  const0.update_output_desc_y(desc_of_const0);
  const0.set_attr_value(tensor_of_const0);
  // 实例化sub节点，设置sub的两个输入
  auto sub = ge::op::Sub("Sub1").set_input_x1(data).set_input_x2(const0);
  std::vector<ge::Operator> inputs{data};
  std::vector<ge::Operator> outputs{sub};
  // 设置图的输入和输出，设置输入接口内部完成构图
  graph.SetInputs(inputs).SetOutputs(outputs);
}
```

现阶段构图接口痛点：
1.使用繁琐，基于原型将ir实例化，然后对应原型的定义设置输入输出，属性等
2.犯错不易察觉，需要真正开始编译图的时候可能才能发现构图的错误
3.C++接口，无ABI兼容性保证
4.仅考虑向后兼容，未考虑向前兼容

To Be（ES构图）：

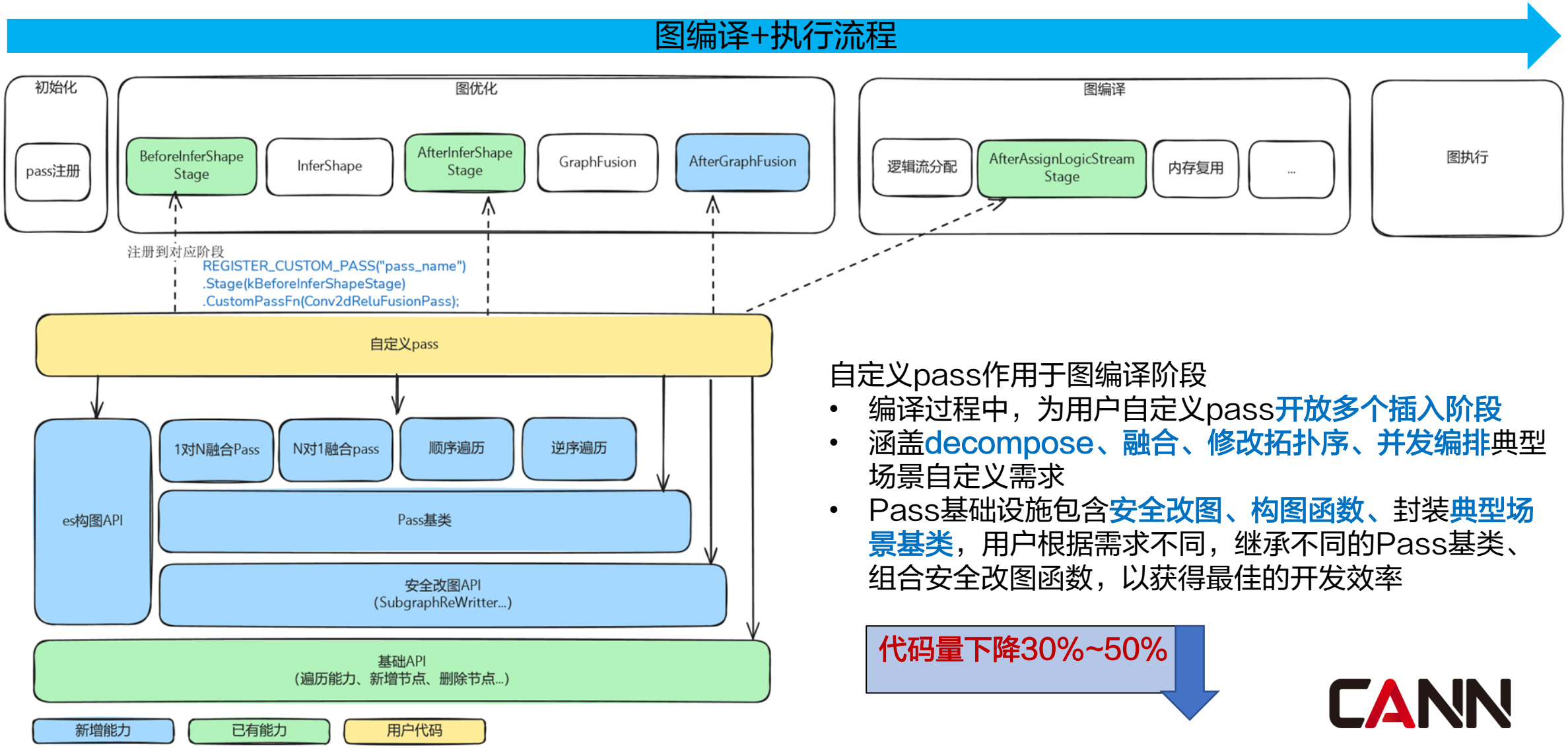
```
using namespace ge::es;
std::unique_ptr<ge::Graph> MakeSubGraphByEs() {
  // 创建图构建器（EsGraphBuilder）
  auto builder = std::make_unique<EsGraphBuilder>("EsbGraph Sample");
  // 创建输入节点
  auto data0 = builder->CreateInput(0);
  // 通过操作符重载创建sub节点，指定sub的另外一个输入是const节点
  data0 = data0 - builder->CreateVector({0});
  // 设置图输出
  (void) builder->SetOutput(data0, 0);
  // 完成构图，获取构造好的`ge::Graph`对象，`builder`中的资源随析构而销毁
  auto graph = builder->Build();
  return graph;
}
```

ES构图API提升易用性：
1. 通过C++语法自动检查错误，不易犯错
2. 通过函数表达连边，降低构图复杂度
3. 常用操作封装函数，例如创建scalar操作，进一步降低构图复杂度

代码量下降 30%

CANN

5.4 自定义优化快速实现-融合pass增强



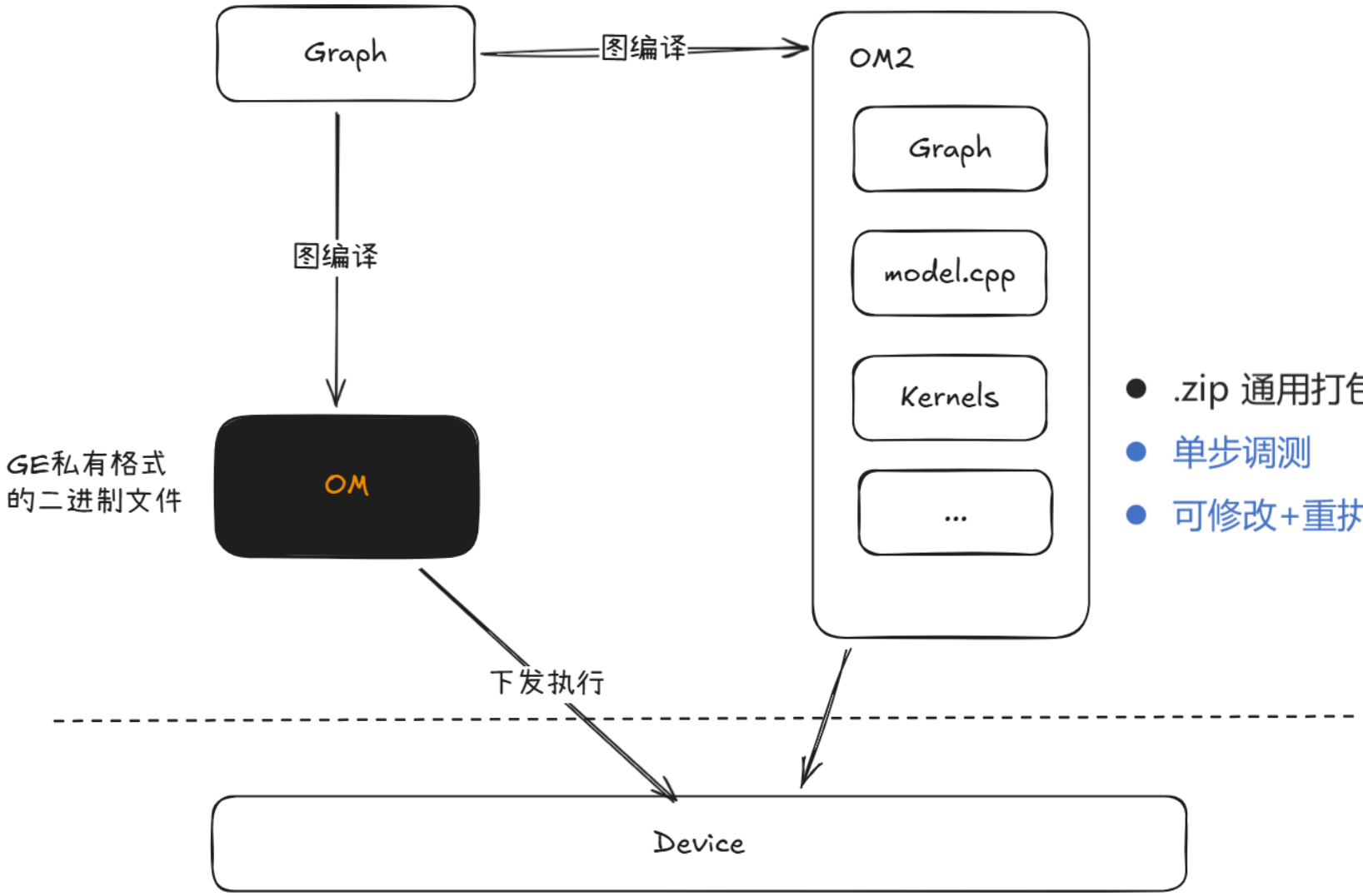
自定义pass作用于图编译阶段

- 编译过程中，为用户自定义pass**开放多个插入阶段**
- 涵盖**decompose、融合、修改拓扑序、并发编排**典型场景自定义需求
- Pass基础设施包含**安全改图、构图函数**、封装**典型场景基类**，用户根据需求不同，继承不同的Pass基类、组合安全改图函数，以获得最佳的开发效率

代码量下降30%~50%

CANN

5.5 可维护-易于调测的透明执行时



- .zip 通用打包格式，内容透明可见
- 单步调测
- 可修改+重执行

集成方式

Pytorch脚本开启图模式，使能GE

```
config = torchair.CompilerConfig()

npu_backend = torchair.get_npu_backend(compiler_config=config)

# 使用TorchAir的后端调用compile接口编译模型
opt_model = torch.compile(model, backend=npu_backend)
```

TensorFlow脚本，使能GE

```
config = tf.ConfigProto(allow_soft_placement=True)

# 添加名字为 “NpuOptimizer”的NPU优化器，网络编译时，NPU只会遍历 “NpuOptimizer”下的session配置。
custom_op = config.graph_options.rewrite_options.custom_optimizers.add()
custom_op.name = "NpuOptimizer“

with tf.Session(config=config) as sess:
```

自动融合开启方式

```
export AUTOFUSE_FLAGS="--enable_autofuse=true"
```

Pytorch使能图模式指导

Tensorflow 模型迁移指导

自动融合开启指导

加入我们

该二维码7天内(1月2日前)有效，重新进入将更新

CANN开源社区 GE组交流群

代码仓地址：https://gitcode.com/cann/ge

GE SIG**组地址：**https://gitcode.com/cann/community/tree/master/CANN/sigs/ge

欢迎Star及参与贡献，共建CANN生态！

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

CANN