基于CANN的
AI 具身智能巡检解决方案及算子开发实例

杭州天宽科技有限公司　吴伟豪

2025年12月20日

Content

目录

背景介绍

01

公司介绍

杭州天宽科技有限公司成立于2007年，总部位于浙江省杭州市，专精特新**“小巨人”**企业。公司以**AI产业解决方案**为核心，是昇腾**生态运营伙伴**、**算子开发伙伴**（基础软硬件平台）、**算力使能服务伙伴**，拥有省级企业研究院、CMMI5、高新技术企业等多项资质。

01

2020年，和华为一起建设杭州人工智能计算中心（一期）

02

2021-2024年，杭州人工智能计算中心**连续扩容**至“三期”

03

2024年，推出基于昇腾原生的**行业大模型适配服务**解决方案推出**一站式AICC运营服务**解决方案，助力全国昇腾AICC商业化运营

04

2025年，联合CANN打造智能巡检第三方算子库，并联合云深处一起构建**具身智能巡检解决方案**。

05

未来，将和CANN开源社区一起，联合伙伴和开发者，**助力打造智能巡检产业高地**

痛点分析

在巡检点位配置层面，依赖人工现场逐点预设的粗放模式，工作量大部署困难；在数据采集层面，智巡终端的 安装与仍凭经验设计，空间覆盖率无法统计，摄像头、机器人、无人机之间协同困难；在数据及算法层面，缺陷识别算法鲁棒性差，跨场景应用准确率离散度达15%-35%；小样本负样本库数据量受限，对非典型缺陷检 出效果不佳，智能水平提升不足，制约全场景感知能力与决策精度。

效率瓶颈

- 巡检点位配置依赖人工：单站数千巡检点需厂商人工手动配置，平均60人天/站；
- 摄像头规划性不足：摄像机盲区率＞15%，重复巡视率达22%；
- 空天地协同巡检困难：无人机/机器人/摄像头联动率<40%，数据采集完整度低。

能力短板

- 算法鲁棒性不足：缺陷识别模型跨场景迁移能力不足，易受光照、遮挡等干扰因素影响；
- 复杂场景漏检率高：绝缘子破损漏检率38%|导线断股29%｜金具锈蚀41%。
- 低频缺陷样本缺口严重：低频缺陷样本量极低（约200例/类），无法满足模型训练需求（5000例/类）；

方案架构

02

天宽&CANN&云深处能源行业“地空协同”AI智巡检解决方案





应用平台	**数据采集与感知** 设备状态 环境状态 图像视频 设备接入	**智能分析与决策** AI 分析 健康评估 故障诊断 数字孪生	**智能执行与控制** 巡检任务 ISDP 缺陷闭环 视频会商	**系统管理与运维** 用户权限 数据存储 监控维护 数据管理	**扩展应用与生态** **数据开放 三方接入**	
巡检模型	**YOLOv11-x**	**SuperPoint**	**SuperGlue**	**SAM-H**	**Gemma3-12B**	**PaddleOCR**
应用使能	**MindSpeed** 训练框架		**MindIE** 推理框架		**MindSDK** AI应用软件开发套件	...
AI框架	PyTorch		[M]S 昇思 MindSpore		飞桨	
CANN **异构计算架构**	**算子库** **智能巡检场景算子库** 基础算子库	**通信库** 高效实现分布式通信 高性能集合通信算法	**图引擎** 图编译、图执行，使能处理器计算加速，提升模型性能	**领域加速库** 覆盖不同开发场景加速套件 提供ATB、SIP等套件	**工具** 支持算子调试，性能调优，提供可视化能力	
	编程语言 Ascend C | ...					
	毕昇编译器 异构编译优化 | PTO Instruction | 支持Triton等三方编程语言					
	运行时 控制流 | 内存管理 | 任务调度					
	驱动 板级驱动 | 加速器驱动 | 设备管理					
昇腾硬件	Ascend	昇腾系列处理器				

天宽具身智能巡检场景算子库：tk_cannops

序号	算子仓名	开源类型	算子名称	开箱性能/token	提升后	整体提升
1	tk_cannops	公开开源	AddRelu	500 μs	200 μs	60%
2	tk_cannops	公开开源	Knn	230ms	875 μs	200倍
3	tk_cannops	公开开源	Nms2d	120 μs	54 μs	45%
4	tk_cannops	公开开源	UniqueV2	278ms	165 μs	1800倍
5	tk_cannops	公开开源	IndexSelect	15ms	14ms	1ms
6	tk_cannops	公开开源	IndexCopy	15ms	14ms	1ms
7	tk_cannops	公开开源	GetRows	15ms	14ms	1ms
8	tk_cannops	公开开源	SetRows	16ms	15ms	1ms
9	tk_cannops	公开开源	npu_unique	54s	0.019s	5000倍

算子开发

03

天宽具身智能巡检场景算子：Unique

如何找到性能瓶颈?

1.uniad模型迁移后

800T A3 上单卡单步耗时 48s 十六卡耗时55s

业内主流卡上 单卡 单步耗时 1.0s 八卡1.2s

2. 使用Ascend PyTorch Profiler 打印profiling文件

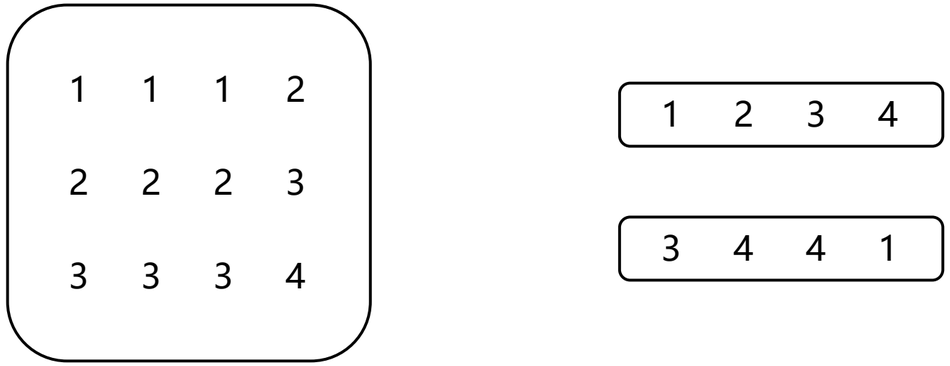
3.使用mindstudio insight 查看profiiing文件

```
318     model.train()
320     end = time.time()
321     experimental_config = torch_npu.profiler._ExperimentalConfig(
322         profiler_level=torch_npu.profiler.ProfilerLevel.Level1,
323         data_simplification=False)
324     with torch_npu.profiler.profile(
325         activities=[
326             torch_npu.profiler.ProfilerActivity.CPU,
327             torch_npu.profiler.ProfilerActivity.NPU
328             ],
329         schedule=torch_npu.profiler.schedule(wait=0, warmup=0, active=1, repeat=1, skip_first=1),
330         on_trace_ready=torch_npu.profiler.tensorboard_trace_handler("./profiling_data"),
331         experimental_config=experimental_config) as prof:
332         for i, (images, target) in enumerate(train_loader):
...
355             # measure elapsed time
356             batch_time.update(time.time() - end)
357             end = time.time()
358
359             prof.step()
```

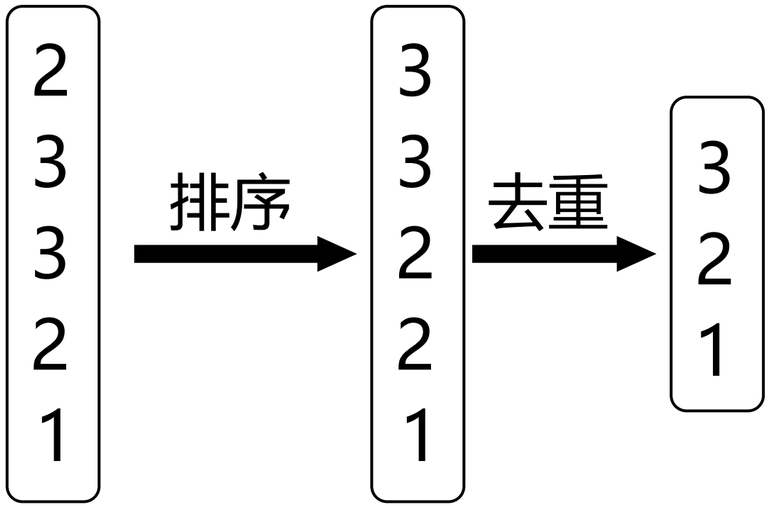
天宽具身智能巡检场景算子：Unique

Unique 的作用是对一个张量去重, 这经常被用在特征索引、查找表构建、图计算等场景

unique 所有元素去重，返回去重后的元素和元素数量统计



实现原理:

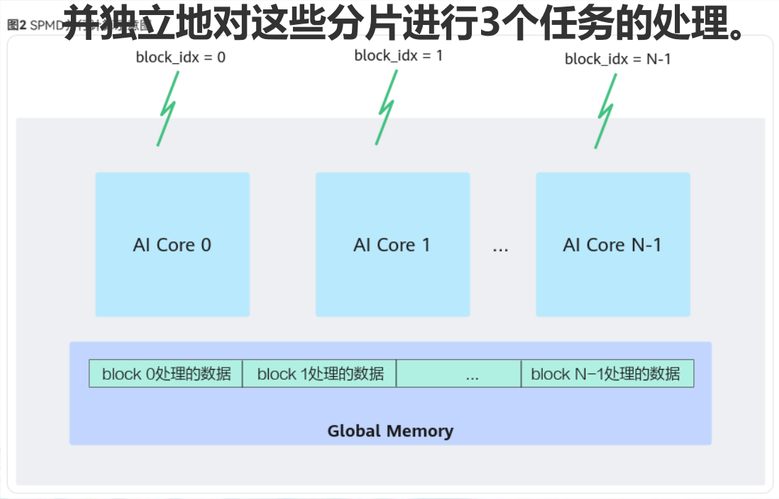


天宽具身智能巡检场景算子：Unique

如何实现Unique

- SPMD（Single-Program Multiple-Data）编程:

  SPMD模式下，系统会启动一组进程，并行处理待处理的数据：首先待处理数据会被切分成多个数据分片，切分后的数据分片随后被分发给不同进程处理，每个进程接收到一个或多个数据分片，并独立地对这些分片进行3个任务的处理。



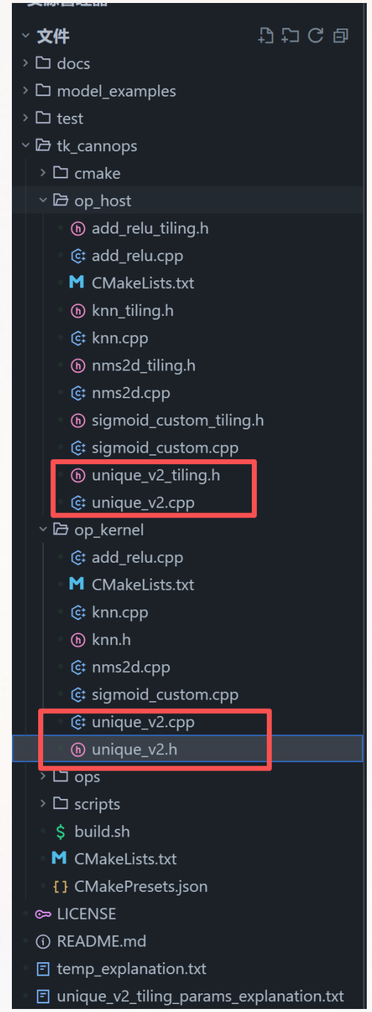
算子开发架构

- **cpu (Host)侧算子的实现:**

  主要是负责数据的切分，比方我们aicore中的local memory 只有192KB,那么我好几M大小的数据必定是不能一次性传完的，所以就是由搬运一部分输入数据进行计算然后搬出，再搬运下一部分输入数据进行计算，这个过程也就是 tiling

- **npu(device)侧 算子的实现：**

  在Kernel函数内部通过解析Host侧传入的Tiling结构体获取Tiling信息，根据Tiling信息控制数据搬入搬出Local Memory的流程；通过调用计算、数据搬运、内存管理、任务同步API，实现算子逻辑。



天宽具身智能巡检场景算子：Unique

算子运行使用的AscendC 指令

排序指令:

- Sort32：

  功能说明:

  一次迭代可以完成32个数的排序，数据需要按如下描述结构进行保存：

  score和index分别存储在src0和src1中，按score进行排序（score大的排前面），排序好的score与其对应的index一起以（score, index） 的结构存储在dst中。不论score为half还是float类型，dst中的（score, index）结构总是占据8Bytes空间。

- MrgSort：

  功能说明:

  将已经排好序的最多4条队列，合并排列成1条队列，结果按照score域由大到小排序。

  每个队列包含32个(score,index)的8Bytes结构

去重指令:

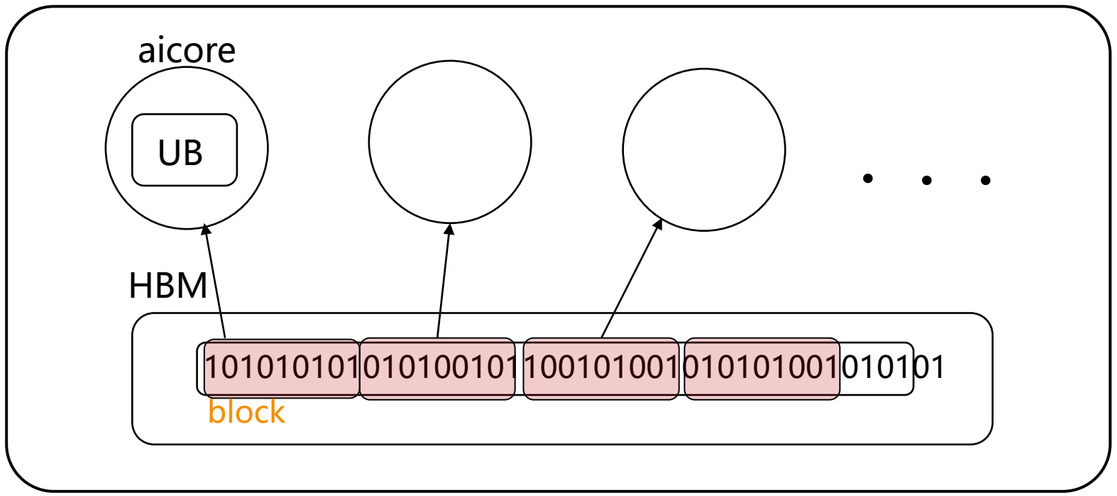
- Compare:

  功能说明:

  逐元素比较两个tensor大小，如果比较后的结果为真，则输出结果的对应比特位为1，否则为0。

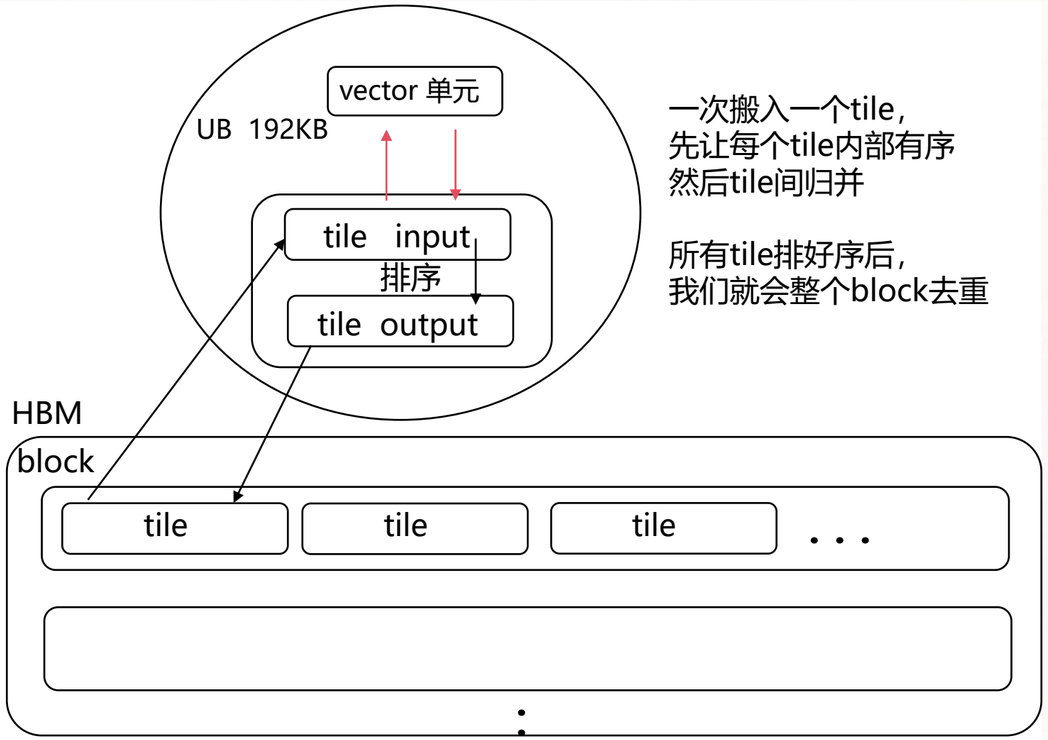
天宽具身智能巡检场景算子：Unique

block间排序:



每个aicore计算自己的block中的数据，先排序再去重，
block_size=total/core_num
各自排完序后，核间归并，40个核心，只需归并3轮

block内tile排序



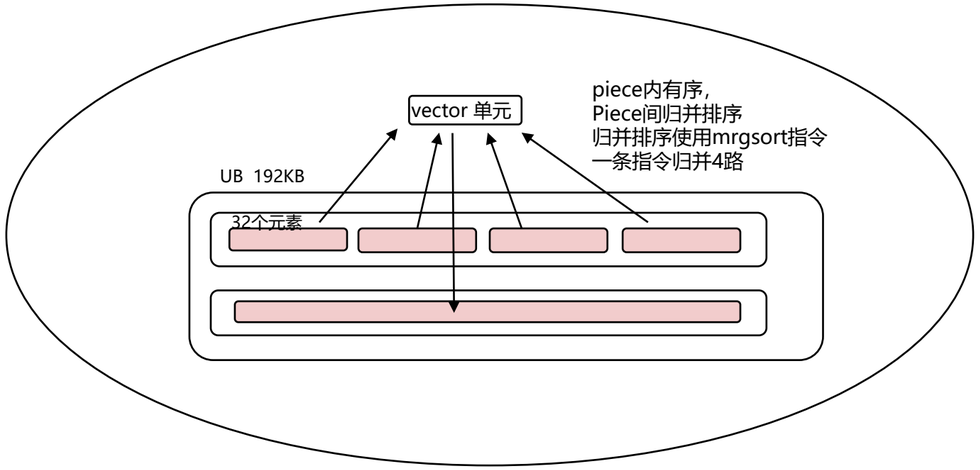
一次搬入一个tile，
先让每个tile内部有序
然后tile间归并

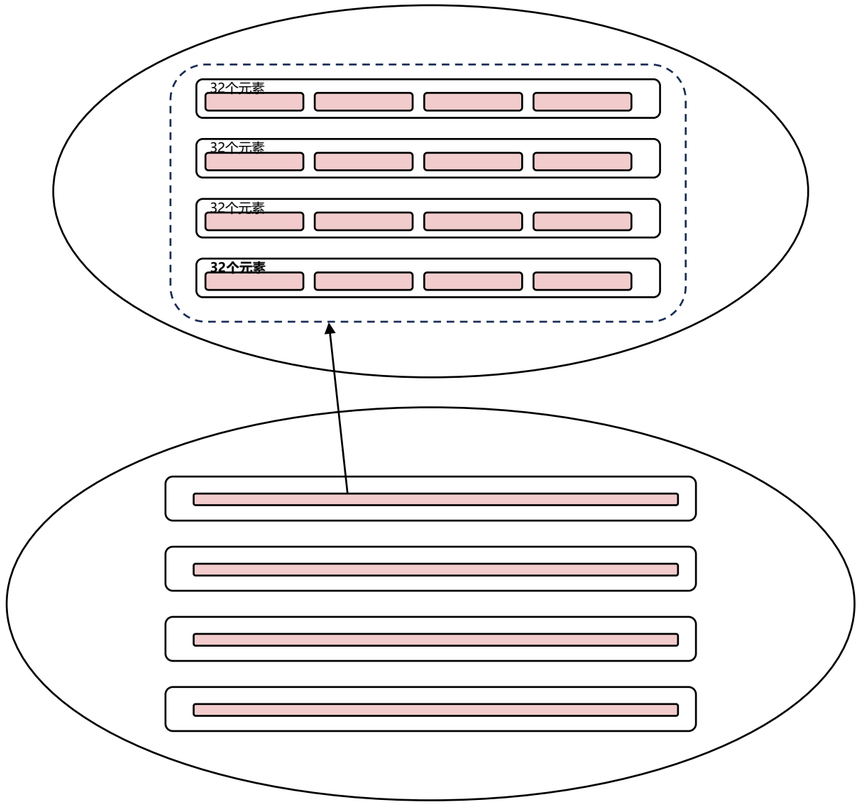
所有tile排好序后，
我们就会整个block去重

天宽具身智能巡检场景算子：Unique

tile内归并排序:

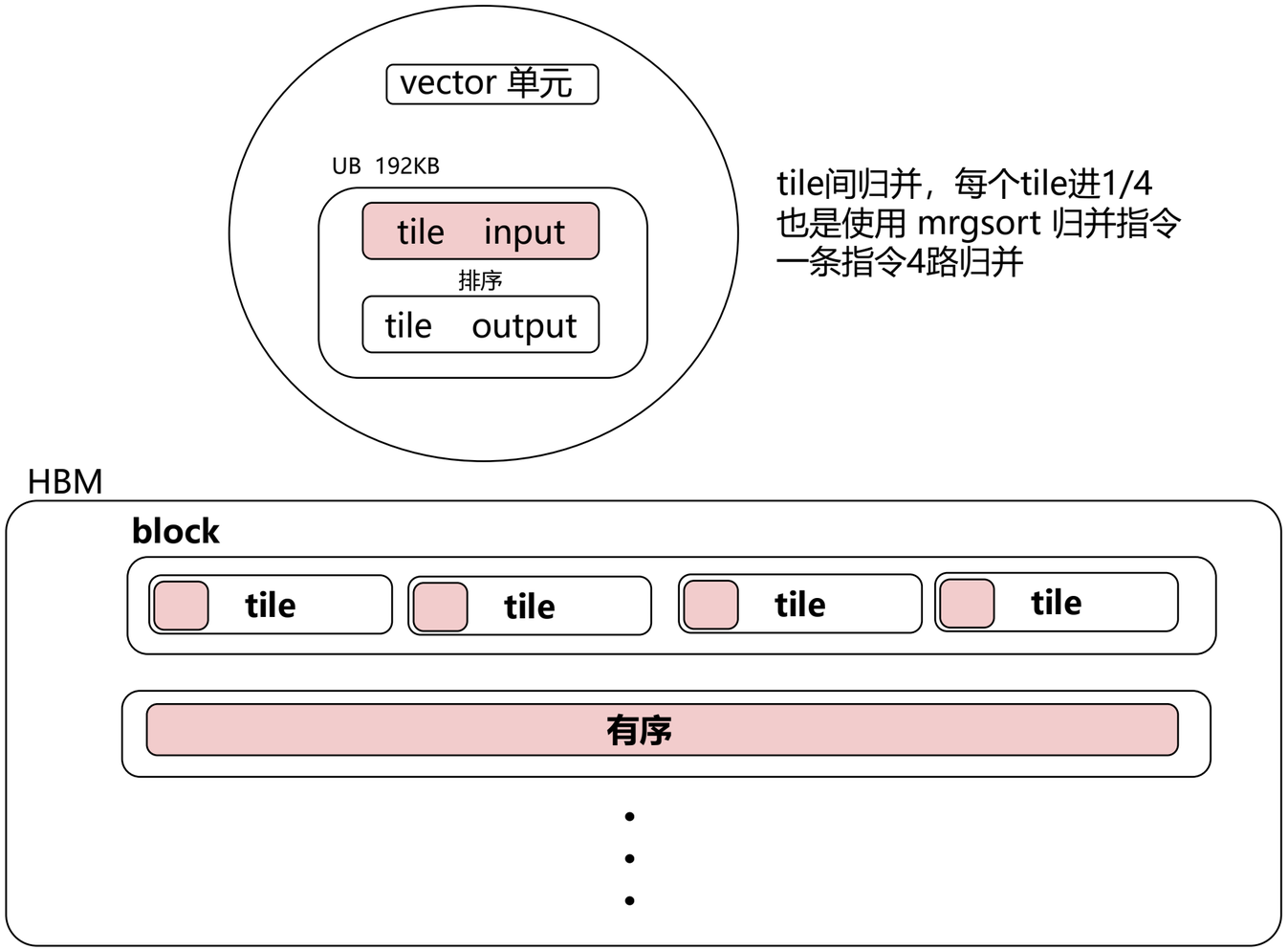
通过sort32指令对32个排序进行排序,再使用mrgsort指令进行排序





天宽具身智能巡检场景算子：Unique

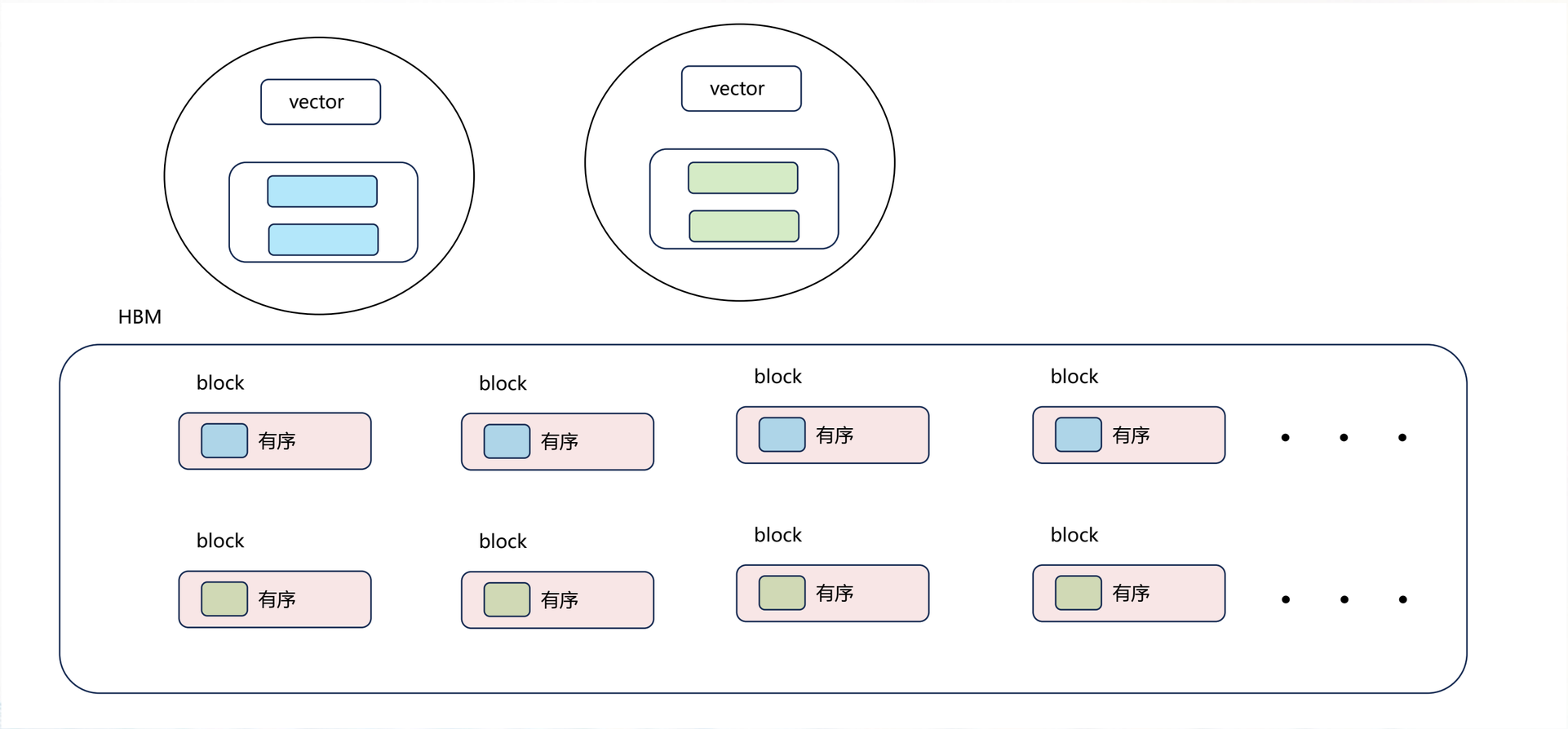
tile间归并排序:每次读取每条tile的1/4，通过mrgsort进行归并排序



天宽具身智能巡检场景算子：Unique

block排序:

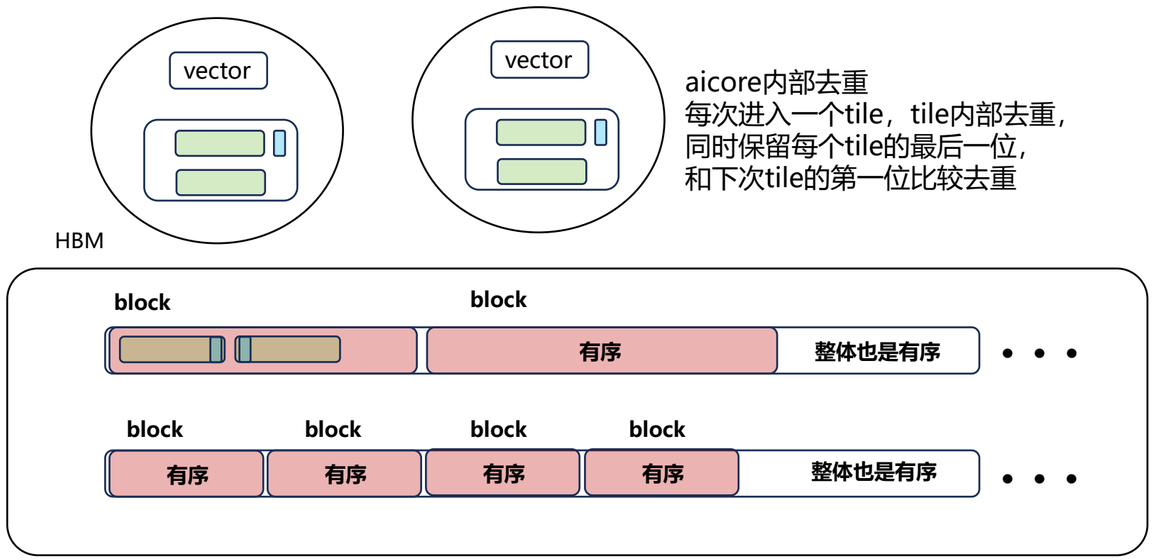
原理和block内tile归并 一样，我们每次都取每个block中的1/4的tile，使用mrgsort进行归并排序。

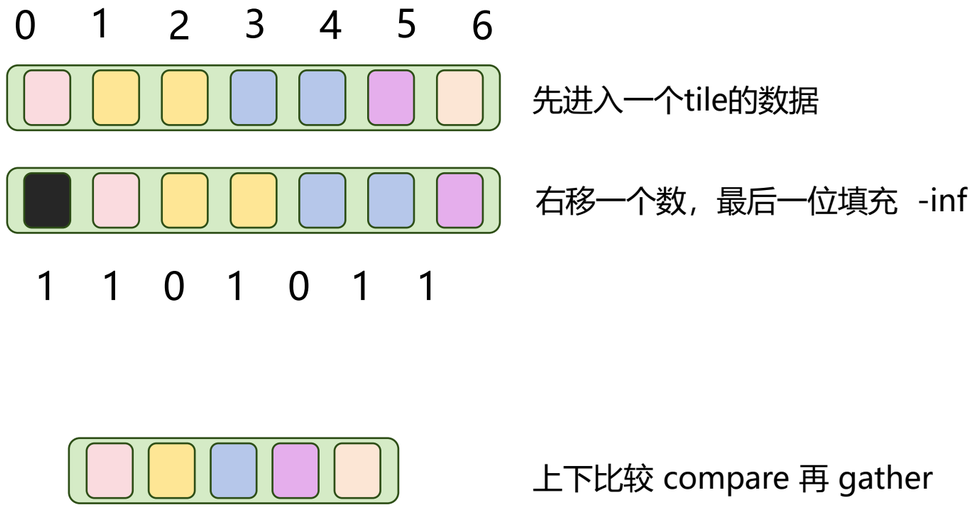


天宽具身智能巡检场景算子：Unique

unique去重:tile去重

对每条tile，进行右移操作，最后一位填充 -inf，对偏移的tile和未偏移的tile使用compare指令，对结果去重，同时保留每个tile的最后一位，和下次tile的第一位比较去重，使用错位比较的掩码进行gather指令操作

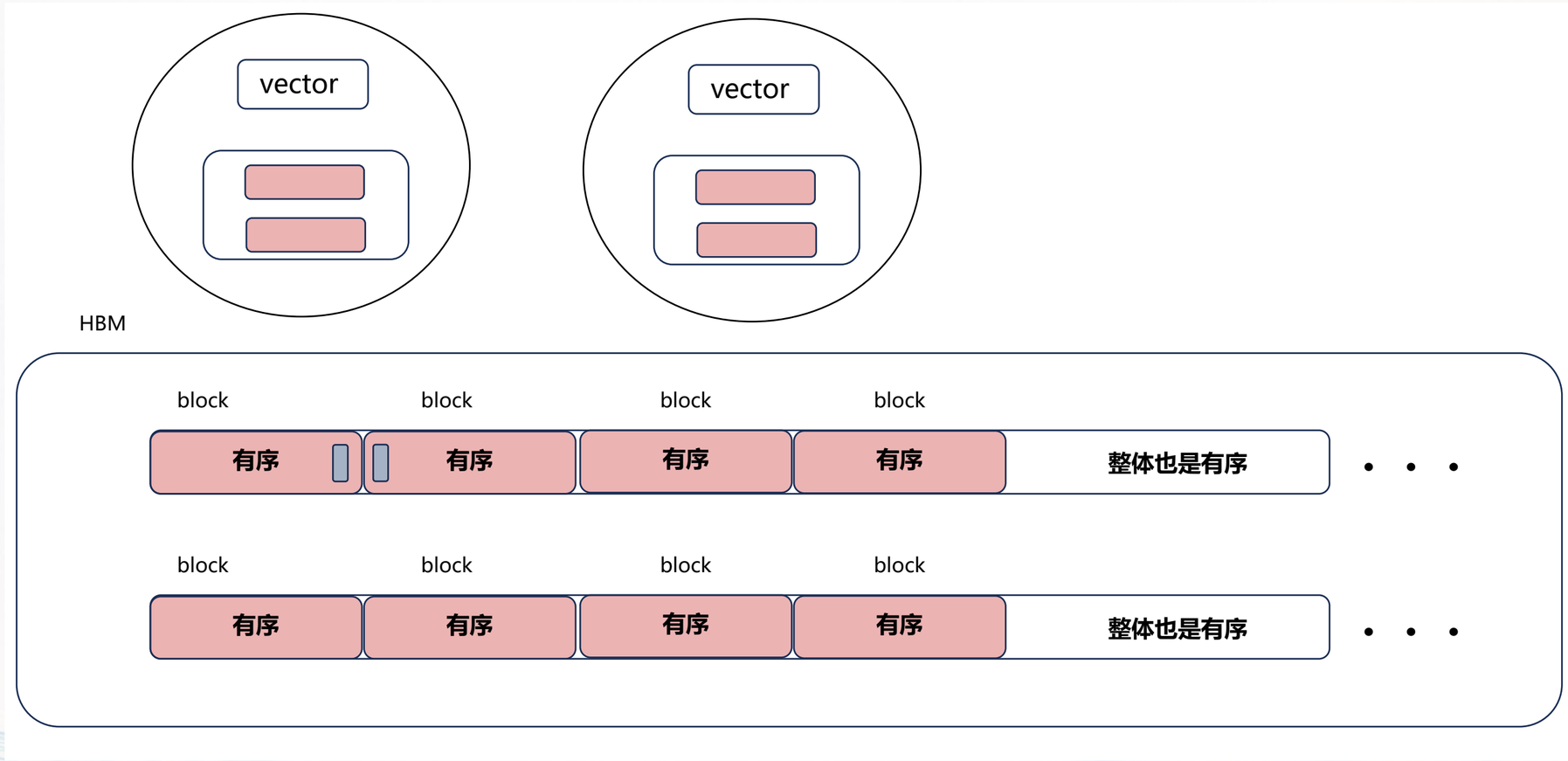




天宽具身智能巡检场景算子：Unique

bolock间去重

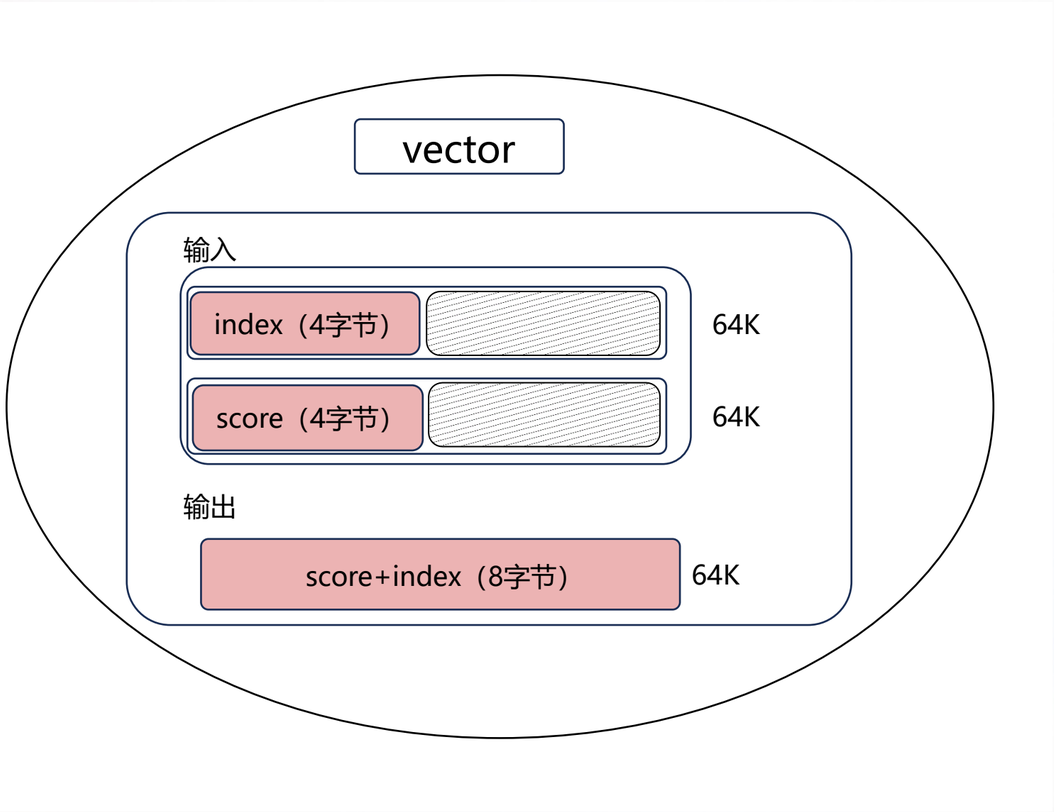
每个AICORE负责把自己的block去重，然后再扫一遍 block 边界数值去重



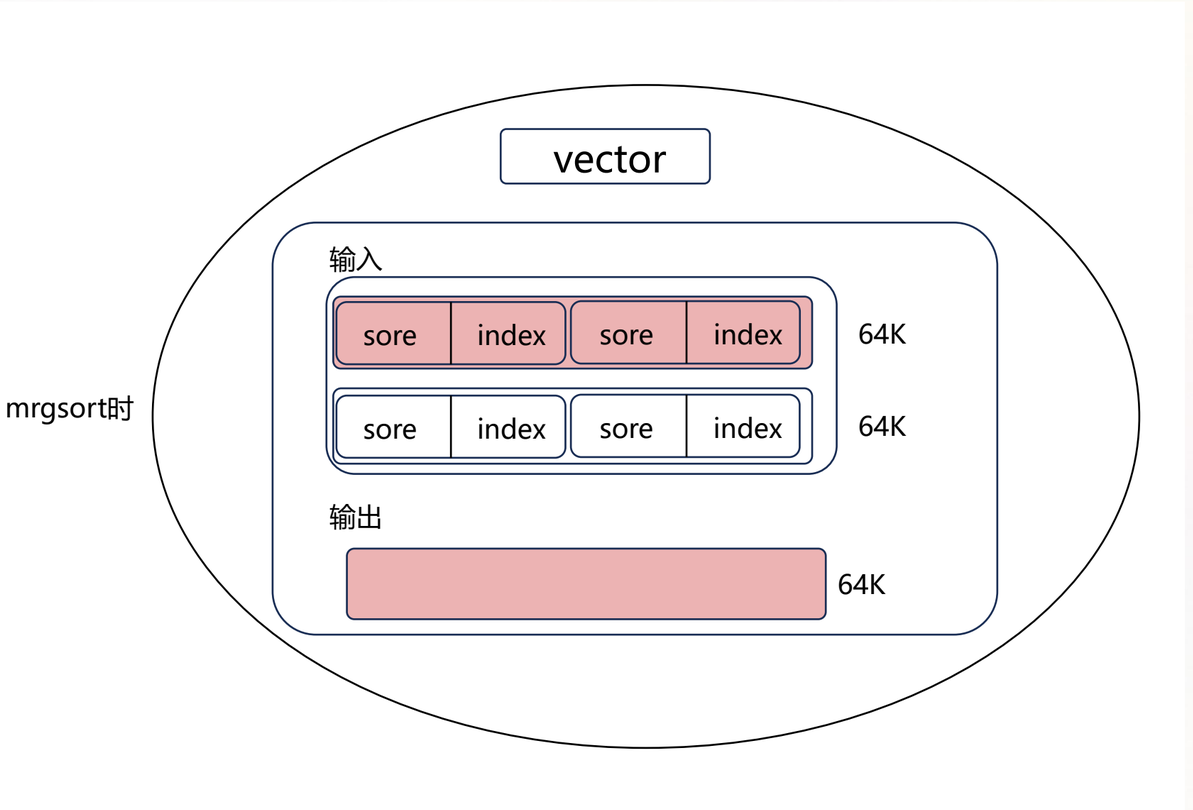
天宽具身智能巡检场景算子：Unique

tiling function的设计:

sort32



mrgsort



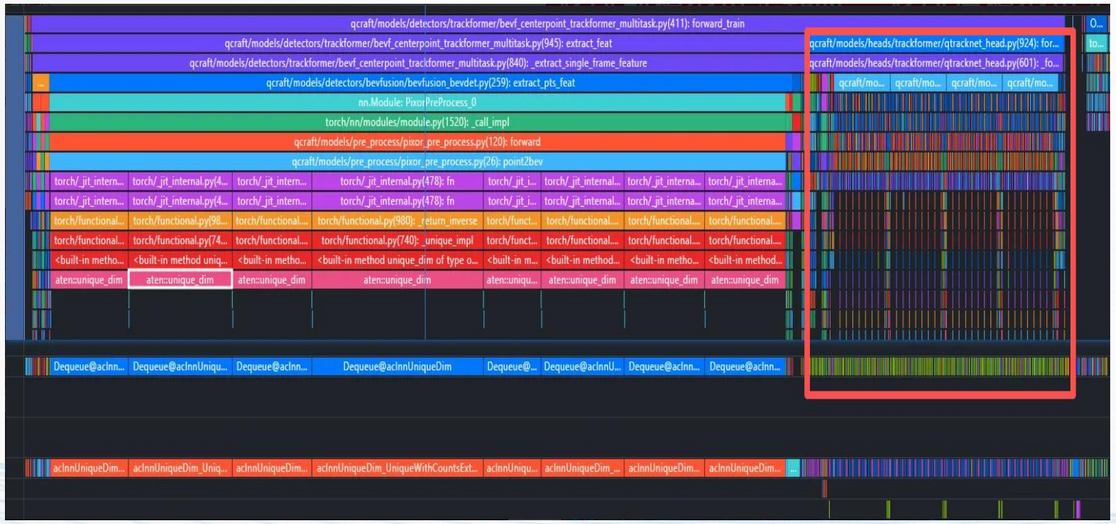
天宽具身智能巡检场景算子：Unique

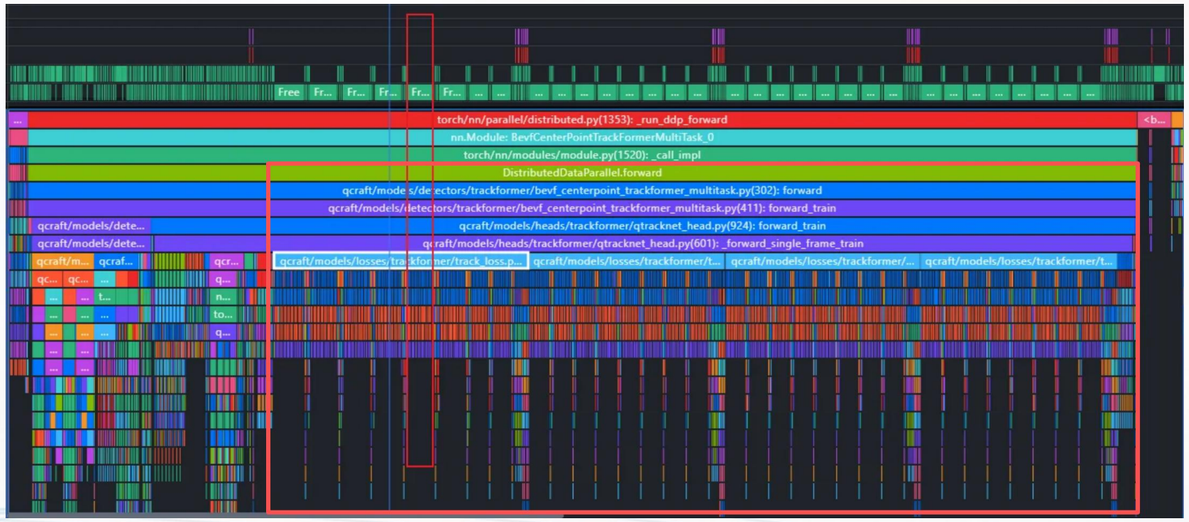
效果提升展示:

```
#测试脚本
import torch_npu
import time
import torch
device = torch.device("npu:0")
input = torch.randint(0,100,[62459,3],dtype=torch.int32).to(device)

torch.npu.synchronize()
ts = time.time()
rs = torch.unique(input,return_inverse=True,return_counts=True,dim=0)
torch.npu.synchronize()
td = time.time()

print(td-ts)
```

修改前单算子耗时	修改后单算子耗时
54s	0.02s

修改前模型单步耗时	修改后模型单步耗时
48s	3.8s





Thanks!

访问CANN开源社区

关注昇腾CANN公众号