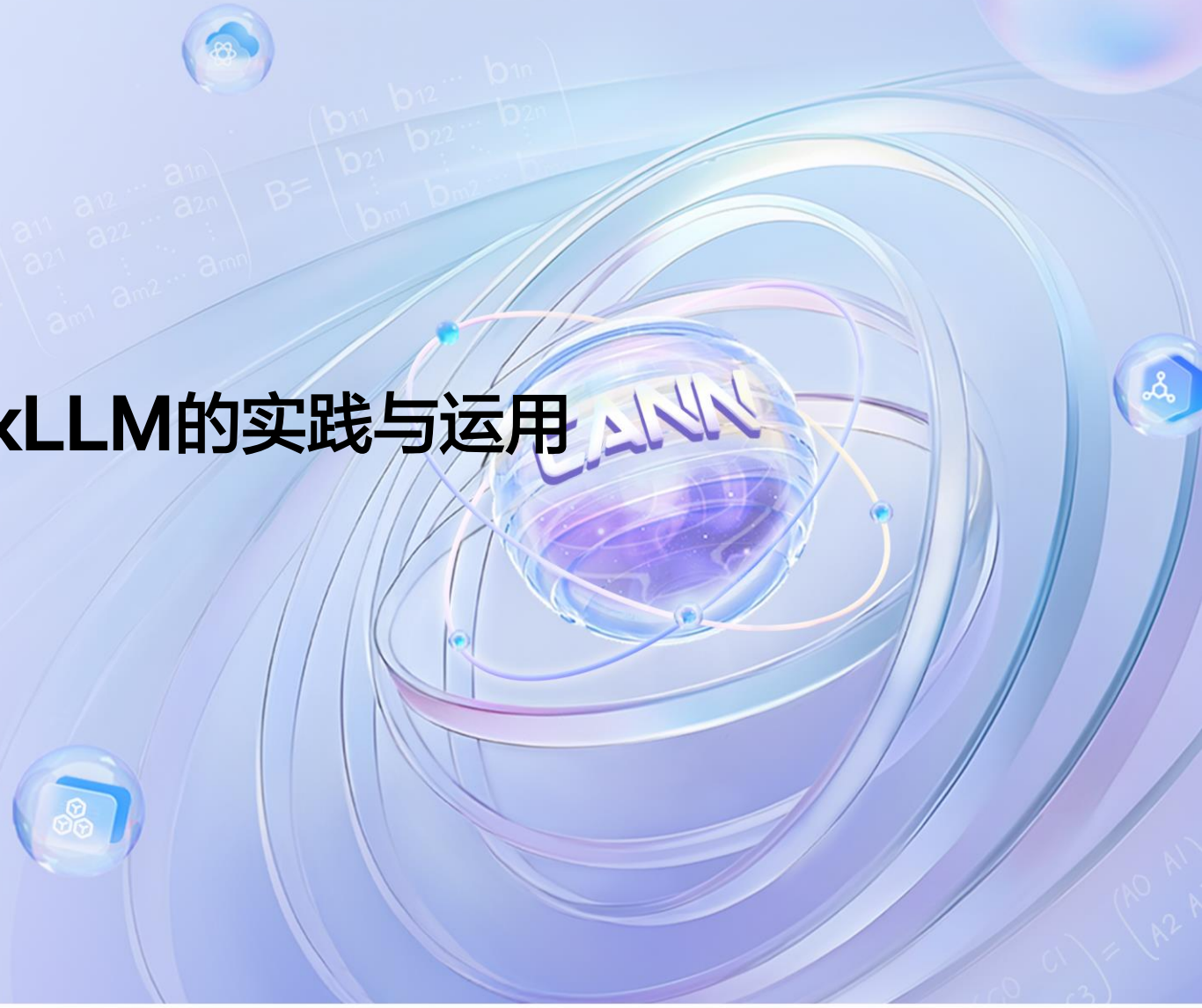
CANN开发者说

-CANN在大模型推理引擎xLLM的实践与运用

李门信 xLLM推理引擎社区

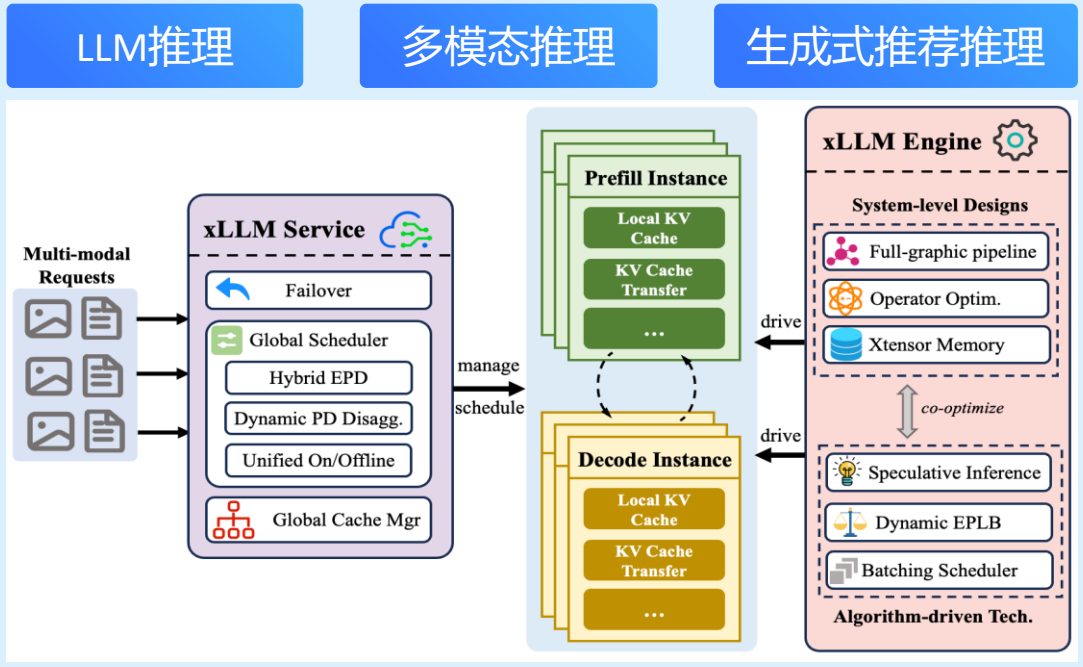


xLLM | CANN

目录

xLLM | CANN

xLLM-面向国产芯片的大模型推理引擎

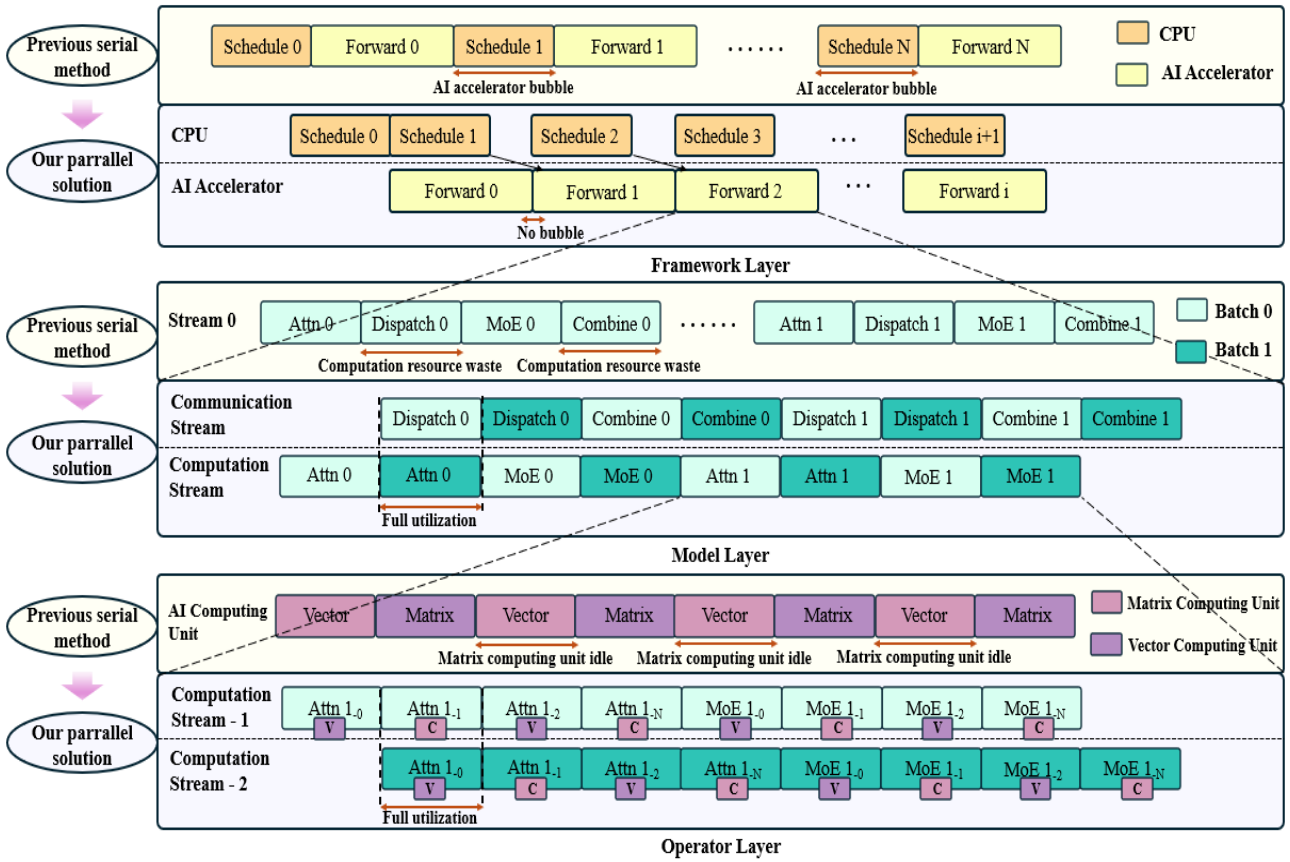


Highlight Feature

- 深度解耦的分布式系统设计，推理从单机走向集群方式
  - 大规模专家并行结合动态负载均衡，引领MoE高性能
  - Hybrid EPD分离式架构，多实例协同推理
  - 全局多级KV Cache Pool，重构推理内存池
- 全局智能调度，严格SLO保证的同时最大化资源利用率
  - 在离线混部任务的统一资源池化与弹性调度
  - 动态自适应的PD调度策略，实例角色弹性调整
  - 请求快迁移、实例快恢复的智能容错机制
- Runtime运行时，追求极致的性能优化
  - 全图化/多层流水线执行编排
  - 动态shape的图执行优化
  - xTensor显存管理优化
  - 高效融合算子库，异构硬件单元、通信与计算并行

xLLM | CANN

推理引擎：算子–模型–框架



多层流水线并纯异步推理挖掘硬件极限

- 框架
  - CPU调度和XPU Forward串行执行，带来XPU bubble问题
  - 通过异步调度，掩盖CPU调度开销，实现XPU连续计算
- 模型
  - 交替执行计算、通信，导致计算/通信资源利用不充分
  - 通过micro batch双流并行，实现通信与计算并行执行，掩盖通信开销
- 算子
  - 计算和数据搬运串行，无法打满算力
  - 矩阵计算单元、向量计算单元与访存流水并行，算子层面实现硬件单元的最大化利用

xLLM | CANN

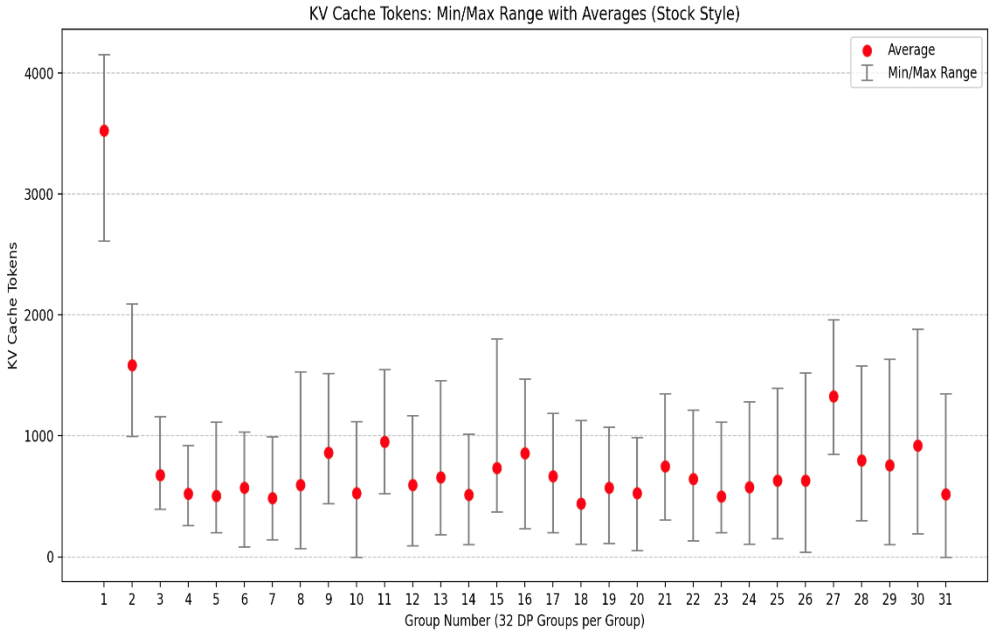
目录

xLLM | CANN

MLA优化背景

背景

1. 使用xLLM运行DeepSeek R1，发现一个DP组内最长的sequence和最短的sequence 差1K个token
2. MLA算子实现中，其切分策略是一条请求分配到一个cube单元，未考虑请求的长短情况，造成资源空闲

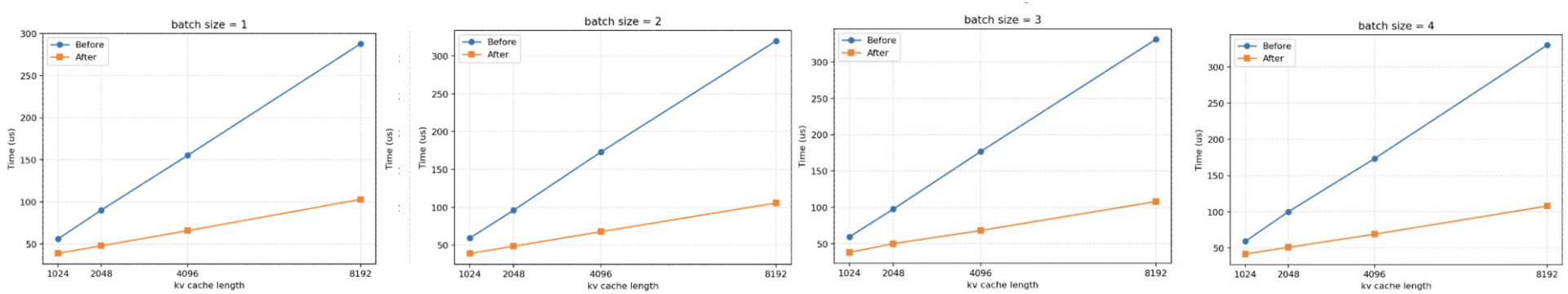


xLLM | CANN

MLA优化实践

实现

- 调度（请求shuffle）：替换原kernel Round-Robin的分配策略，将请求顺序重排
- 算子优化：改变尾块切分逻辑，实现任务均匀分配，减少资源空闲时间



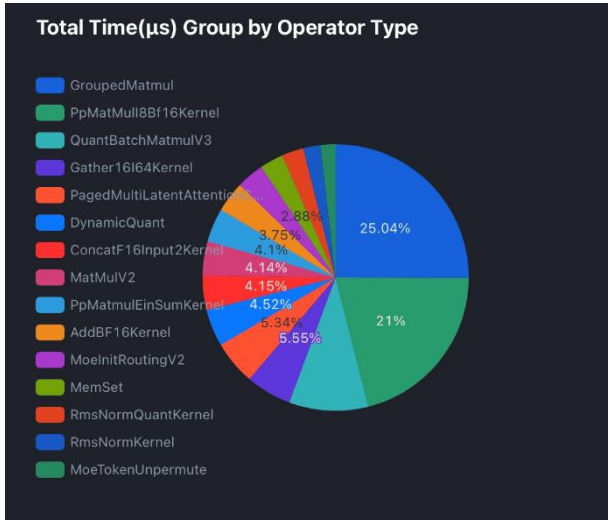
效果

- 算子耗时较之前MLA算子最多下降70%，接入xLLM中，实现TPOT下降5%

GroupGemm优化背景

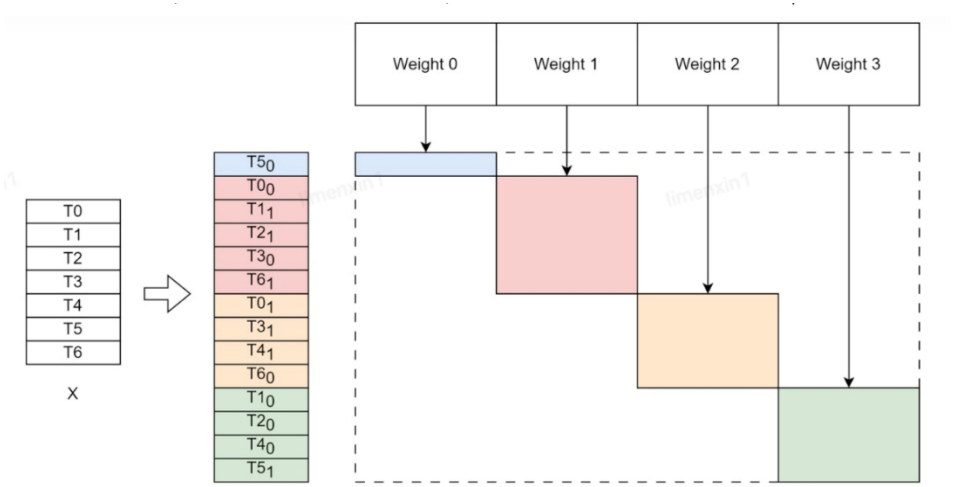
背景

- 通过性能分析工具发现DeepSeeK-R1中，GroupGemm 耗时占比最大，25.04%
- GroupGemm算子瓶颈为memory bound



动机

- GroupGemm算子为确保各专家计算的连续性，会将每个token向量复制若干份暂存于L2缓存，增大MTE2的传输负担

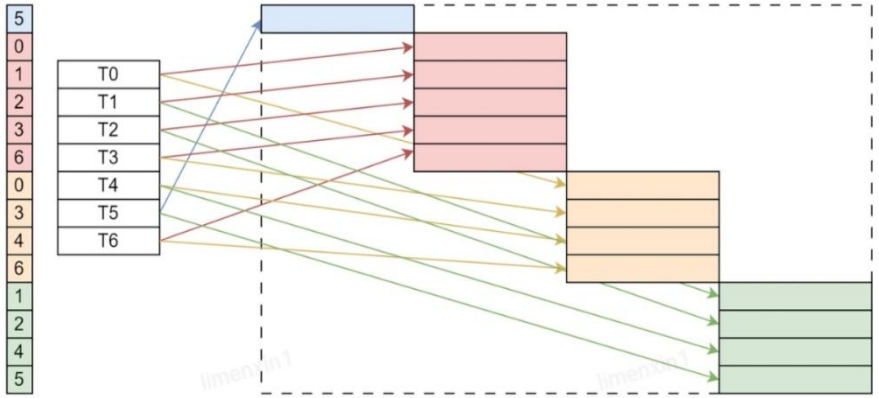


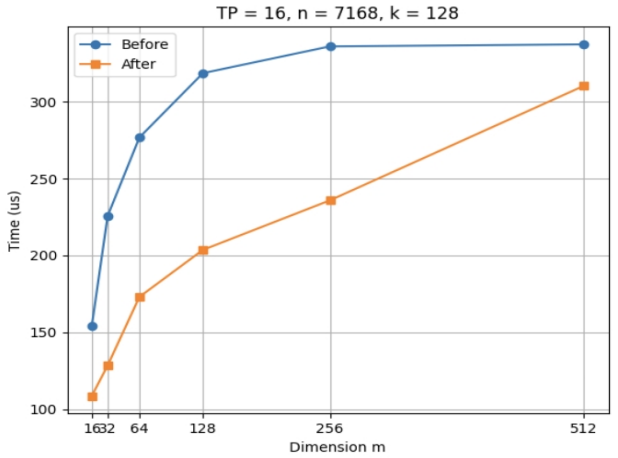
xLLM | CANN

GroupGemm优化实践

实现

- 算法优化：维护专家分配索引表，通过token行号索引直接映射到对应专家计算单元，并将token分配调度与矩阵乘法融合为单一kernel
- 实现优化： 在算子内部实现多级流水，同时开启doubler shot，使得多级流水之间空泡更少





效果

- 优化后的算子性能降低10%至50%的时延，接入xLLM框架后，实现TPOT下降5%

xLLM | CANN

xLLM已开源！

xLLM 项目地址：https://github.com/jd-opensource/xllm
https://github.com/jd-opensource/xllm-service
https://gitcode.com/xLLM-AI/xllm_ops

xLLM 技术报告：https://arxiv.org/pdf/2510.14686

xLLM | CANN