CCF 2025 HIP编程竞赛设计报告

user026队

赛题简要介绍

本次CCF 2025 HIP编程竞赛包含三个GPU并行计算挑战，旨在测试参赛者在AMD GPU平台上的高性能计算能力。三个赛题分别为：

1. 前缀和（Prefix Sum）

- **目标**：计算数组的前缀和，即对于输入数组A，输出数组S满足 $S[i] = \sum_{j=0}^{i} A[j]$
- **约束**：数组长度可达10亿

2. Softmax函数

- **目标**：实现数值稳定的softmax函数，计算 $y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$
- **约束**：数组长度可达1亿，必须使用数值稳定形式避免溢出

3. 全对最短路径（APSP）

- **目标**：计算有向加权图中所有顶点对之间的最短路径距离
- **约束**：顶点数最多4万，边权重非负

所有赛题均要求使用HIP（AMD GPU编程框架）实现，不允许使用外部库，仅支持单GPU计算。

实现方案

1. 前缀和算法实现

采用三阶段分块算法：

1. **局部扫描阶段**：每个块内部并行计算前缀和
2. **块和扫描阶段**：对块和数组进行前缀和计算
3. **偏移添加阶段**：将块偏移量添加到局部结果中

2. Softmax算法实现

采用单kernel融合设计，避免多次kernel启动开销：

1. **全局最大值计算**：使用grid-stride循环和block归约
2. **指数和计算**：基于全局最大值计算稳定的指数和
3. **归一化输出**：计算最终的softmax值

3. APSP算法实现

采用三阶段分块Floyd-Warshall算法：

1. **阶段1**：处理主对角线块，计算块内最短路径

2. **阶段2**：更新主对角线行和列，使用新计算的主对角线距离
3. **阶段3**：更新剩余块，使用更新后的行和列距离

优化与改进

前缀和（Prefix Sum）

1. **内存访问：** 共享内存分块 + warp/wave shuffle，减少全局访存与块内同步；合并访问；hipHostMalloc 页锁定内存配合异步拷贝，加速 H↔D 传输。
2. **计算与并行：** 三阶段（块内扫描 → 块和扫描 → 偏移回写）仅对必要偏移写回，降低写放大；采用 grid-stride 循环与多元素/线程提升指令级并行；按寄存器/共享内存占用调优 blockDim 与 items/thread；小规模输入直接走 CPU 以避免 kernel 启动开销。
3. **同步策略：** 块内使用 __syncthreads()；跨阶段用多 kernel 分解替代全局栅栏，降低复杂同步风险。
4. **效果：** 十个自测case中均通过，并且测试总时间达到最快4.72s。

Softmax

1. **数值稳定：** 最大值平移 x←x−max(x)，避免 exp 溢出/下溢；必要时引入补偿求和（可选）在严格误差阈值下开启。
2. **计算与并行：** 两遍归约（最大值、sum(exp)）或融合为少量 kernel；grid-stride + 块内归约；合并访问与（可选）向量化加载提升有效带宽与 SM 利用率。
3. **同步策略：** 优先以“多 kernel 阶段边界”完成跨块同步；仅在验证充分时采用原子 + __threadfence() 实现轻量全局栅栏；块内使用 __syncthreads()。
4. **效果：** 10个测试样例均通过，误差达到要求水平，并且测试总时间达到约5.3s水平。

APSP（分块 Floyd–Warshall）

1. **内存访问：** 16×16 分块，三阶段（主对角块 → 同行/同列 → 其余块）在共享内存中三次复用数据，显著降低全局带宽压力；行主序分块与对齐，减少 bank 冲突。
2. **计算与并行：** 块内循环展开与寄存器缓存热数据；在共享内存占用与占用率间调优 blockDim；必要时按显存容量进行批次化处理。
3. **I/O 优化：** 大规模下采用缓冲区快读/批量写，显著降低 I/O 占比；相比其余优化，I/O优化起到了更为明显的作用。
4. **效果：** 十个自测样例均通过，消耗时间最优达到6.71s。

遇到的困难与解决方法及 LLM 的应用

比赛中我们主要遇到大规模数据带来的显存与带宽压力、跨块同步的正确性，以及 Softmax 的数值稳定问题。针对 APSP，我们采用分块 Floyd–Warshall 与批次化 I/O 降低峰值显存并缓解带宽瓶颈；前缀和与 Softmax 拆分为多 kernel 阶段，用阶段边界替代全局栅栏，并结合合并访存、共享内存 tile、grid-stride 循环与必要的向量化提升吞吐；

LLM 方面，我们用其对比方案与加速原型（提供 HIP 归约/扫描骨架）、辅助定位 bank 冲突/越界等问题。

总结

本次竞赛的三题我们均完成单 GPU 实现，样例通过并获得稳定加速。其中在优化方案的获取和实现过程中，LLM起到了较为重要的作用。值得一提的是，在完成GPU实现后的优化调度中，I/O优化起到了举足轻重的地位。