南开大学 Nankai University

南　　开　　大　　学

计　算　机　学　院

并行计算技术课程文档

Mindspore Lite 中 ArgMixMax 算子并行化开题报告

邵铭恺 2120240747

专业：计算机技术

指导教师：王刚

2025 年 5 月 17 日

目录

一、 绪论 **1**

（一） 简介 1

（二） 研究背景及意义 1

二、 问题描述 **1**

三、 相关工作 **3**

（一） TopK=1 算法 3

（二） TopK>1 算法 3

四、 研究计划 **4**

（一） 研究内容 4

（二） 研究计划 4

（三） 可行性分析 4

一、 绪论

(一) 简介

本文档是南开大学 2025 春《并行计算技术》课程期末研究报告的开题报告，作者为卻铭恺，学号为 2120240747，选题为 Mindspore Lite 中 ArgMinMax 算子的并行化设计。文章将对研究背景及意义进行介绍，定义研究问题，并对本问题的前人研究工作撰写简要的文献综述、梳理研究历史和现状并分析其优缺点，以此来引出我们的工作内容与贡献。此外，本文还将制定研究计划，进行可行性分析，并描述内容分工。

(二) 研究背景及意义

MindSpore 是华为开发的新一代全场景 AI 框架，面向高效、易用、安全而设计，期望支持云、边、端的全场景灵活部署，开发全新的 AI 编程范式并降低 AI 编程门槛。MindSpore 与其他同类型编程框架相比，如 PyTorch 和 TensorFlow，通过自动并行化、软硬件协同优化等方法，支持自动化算子融合、动态图静态图，并且支持自动微分、切分、分发等，来进一步改善 AI 模型的开发、应用复杂度。算子作为 MindSpore 中的基础计算单元，具备极其重要的作用，其直接对应具体的数值计算步骤，包括简单的数值加减乘除指数对数、归约操作、激活函数和损失函数以及如矩阵乘法和快速傅里叶变换的复杂类型计算等内容，它们执行的效率直接关系到这个框架的使用效率。

年初时我们实验室接到了 MindSpore 算子开发项目，主要的项目目标是对两种芯片开发多种数据类型下的 c 语言类型和线性汇编类型的高性能算子，需要对大型数值计算算子进行并行化以及流水线方法来提高它们的性能。本次实验我将聚焦与该项目中我负责的一些算子，对它们研究并行化方法来完成实验，并期望能达到性能指标要求的效果。

二、 问题描述

本次实验将基于此工作中我负责的 ArgMinMax 算子，研究它的并行化策略方法。ArgMinMax 的具体操作较为简单，含义是取出某一行元素的前 k 大/小元素的索引值，其本质是一个归约操作。MindSpore 中对于 ArgMinMax 算子的情况划分较为细致，包含较多参数，包括对某个轴（axis）操作、是否取出索引对应值、大小比较函数等等，并且 MindSpore Lite 中限制张量大小为前 4 维，它的主要计算流程如算法1所示。

该算子包含两种情况，两种情况分别对应两个算法。一是在 topk 大小为 1 时，使用 top1 算法计算，其计算过程为按照轴维度参数得出索引元素，然后遍历取最大，是一个时间复杂度为 $O(n)$ 的算法。另一个算法则是下面的 switch 语句对应的算法，对于多个维度单独进行操作，算法流程为按照轴和偏移量取出对应数组后，将数组排序后直接取前 k 大的元素。由此可见，第一种算法的主要性能瓶颈在数组遍历，而第二种算法的性能瓶颈则在排序阶段。期末研究报告中，我将对这两种算法的并行化策略进行研究，通过优化关键路径的方式来提升算法性能，在本报告中将对前人对这些算法的研究进行简要综述，并且给出详细的工作计划。

Algorithm 1 MindSpore ArgMinMax 算子执行流程

Input: 输入数组 *input*，形状 *in_shape*，参数结构体 *param*（含维度 *dims_size*，轴 *axis*，是否取最大值 *get_max*，topk 值 *topk*）

Output: 输出索引数组 *output*，输出值数组 *output_value*

计算前轴维度 *pre_axis_count*，轴维度 *axis_count*，后轴维度 *after_axis_count*
if *param.topk* == 1 **then**
    if *param.get_max* **then**
        ArgMaxTopK1(*input*, *output*, *output_value*, 轴维度参数)
    else
        ArgMinTopK1(*input*, *output*, *output_value*, 轴维度参数)
    end if
    return
end if
初始化比较函数 *compare_function*
if *param.get_max* **then**
    compare_function ← ArgCompareDesc32
else
    compare_function ← ArgCompareAsc32
end if
switch (*param.axis*)
case 0**:**
    ArgMinMaxDim0(*input*, *output*, *output_value*, *compare_function*)
case 1**:**
    ArgMinMaxDim1(*input*, *output*, *output_value*, *compare_function*)
case 2**:**
    ArgMinMaxDim2(*input*, *output*, *output_value*, *compare_function*)
case 3**:**
    ArgMinMaxDim3(*input*, *output*, *output_value*, *compare_function*)
end switch
return

三、 相关工作

(一) TopK=1 算法

TopK=1 时，算法较为简单，即在一个一维数组中找出这一维的最大值，并将其返回。一个简单而朴素的算法就这么得出了，设置一个初始变量为对应数据类型的最小值，然后遍历一次一维数组，每遍历到一个比它大的元素，就将这个初始变量设置为这个比它大的值，到最后就能够得出执行结果。这种朴素方法的时间复杂度为 $O(n)$。

如果现在我们不满于这个算法的性能，我们可以使用向量指令，使用 SIMD 的方法对这个算法进行处理。Intel 和 ARM 公司提出的 AVX [1] 和 NEON [2] 支持我们通过掩码寄存器来实现极值查找的向量化，可以将算法的性能提升数倍。

仅取 TopK=1 的情况，我们也可以使用多线程来对其进行性能优化，可以将数据按照块划分或者循环划分的方式，将其划分为多段，分给单个线程执行一样的算法，最后让多个线程得出的局部最大值进行比较即可得到一个全局最大值。多种编程语言中，有多种并行化库可以做到这些点，如 c++ 的 std::thread，pthread，Java 和 Python 各自的 thread 类，以及 openmp 自动并行化等。

由上文描述，我们可以知道该算子的本质是一个归约操作，NVIDIA 公司在 NVIDIA 编程指南的一部分中介绍了树形归约 [3] 的方法，如图**??**它内部将每相邻的两个元素两两进行比较，然后利用 GPU 的强大算力将其迅速折叠缩小为一个数，这种方式非常适合具备众多核数的 GPU 来进行并行化。此外，2008 年 Dean 等人提出了 MapReduce 的概念 [4]，可以在 Map 阶段将其分为多块后，再在 Reduce 阶段进行结果的聚合和归约操作。现在多个大型 AI 计算框架中都具备各自进行优化后的的 ArgMinMax 算子，我们可以在实验中选取一种最适合的方法（或者是聚合方法）来完成我们的算法研究。

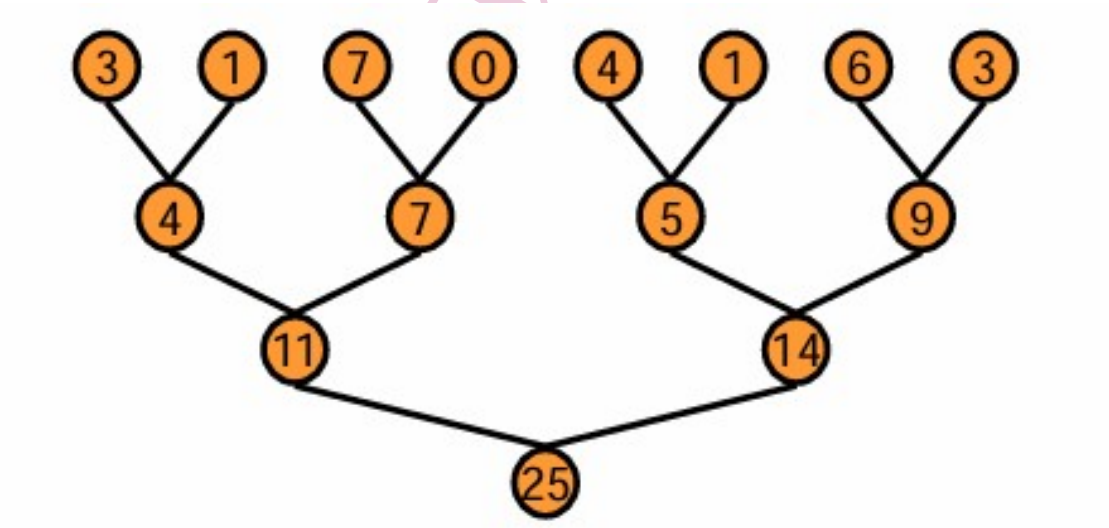

图 1: 树形归约示意图

(二) TopK>1 算法

TopK>1 是，算法的主要计算时间瓶颈在排序的一步上，由此我们可以从排序算法并行化研究入手。MindSpore 中为我们提供的实现中，内部调用了 C 语言标准库的 qsort 函数，使用串行方式的快速排序来完成算法，我们在这里可以考虑将其修改为并行版本，如并行快速排序，或者更换为更适合并行化的排序算法。排序算法种类繁多，并且有着不同的优缺点，许多研究者都

对排序算法的并行化有一些研究，在算子开发项目中我们可以根据实际并行情况选取最合适的算法来进行应用。

关于 SIMD，对于快速排序而言，SIMD 可能难以适用，这里考虑其他的排序算法。我们可以考虑双调排序 [5] [6] 的方法来完成工作，此外也可以考虑归并排序和基数排序的优化。在数据量较大时，双调排序也可以充分利用 GPU 的多个计算单元，使用 CUDA 进行优化。

关于多线程的方法，可以选用快速排序和归并排序的方法，通过共享内存进行线程间通信来进行并行化（多核同理）。需要注意，实际设计排序算法时，可能会出现输入数据过大导致核内存储空间无法存储全部输入数据的情况，此时为充分利用缓存性能，或许需要采用与 MySQL 类似的外部排序算法来进行设计和实现。

四、 研究计划

本节中将对我们的实际研究工作进行介绍，制定研究计划，并进行可行性分析。

(一) 研究内容

上述提到的研究方法中，均在不同场景中被实践为可行。本次报告中，我将通过复现以上研究方法的方式来提升 MindSpore 中算子的性能，通过与 MindSpore 原本实现性能对比的方式证明所复现方法的有效性，并根据时间精力来进行融合算法的设计与实现，寻找性能平衡点，来完成一个综合性能较强的 ArgMinMax 算子实现。此外，在 MindSpore 项目中，我们需要考虑每个算子的数据的存放位置，三个位置包括核内 L2 Cache，核间共享内存和核外 DDR 上，其读写性能依次递减，因此需要考虑使用 DMA 加速的方式来进一步提升性能，并且排序算法在一部分数据存在外存时，也有单独的优化方法。在本报告中，如后期还有精力，会考虑将外存排序也纳入该研究。

(二) 研究计划

结合 MindSpore 项目进度，制定研究计划，大体如下：

1. TopK=1 算法多线程方法实现及测试：5 月 24 日之前；
2. TopK=1 算法其他方法实现及测试：5 月 31 日之前；
3. TopK=1 算法融合方法实现及测试：6 月 7 日之前；
4. TopK>1 算法多线程方法实现及测试：5 月 24 日之前；
5. TopK>1 算法其他方法实现及测试：5 月 31 日之前；
6. TopK>1 算法融合方法实现及测试：6 月 7 日之前；
7. 考虑外存的算法设计、实现及测试：6 月 21 日之前；
8. 其他的性能调优及报告撰写：6 月 30 日之前。

(三) 可行性分析

对于 TopK=1 算法，每个算法的并行化思路和实现方式都较为简单，具备良好的可行性。为和 MindSpore 项目开发进度进行对接，实际会主要专注于每个算法的性能对比以及融合算法的开发，来实现一个综合性能较高的算法。

对于 TopK>1 算法，排序算法种类较多，实际会根据时间精力，选取一些较为适合并行化的算法进行编写与调优，保证正确性的同时，尽可能提升算法的性能。并且在实际的算子开发中，需要对外存排序情况也进行对应处理，该任务具有一定的挑战性，会根据实际情况具体选择对应实现。

参考文献

[1] Intel® Intrinsics Guide. https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html. Accessed: 2025.

[2] Neon –Arm®. https://www.arm.com/technologies/neon. Accessed: 2025.

[3] Optimizing Parallel Reduction in CUDA. https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf. Accessed: 2025.

[4] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[5] Gianfranco Bilardi and Alexandru Nicolau. Adaptive bitonic sorting: An optimal parallel algorithm for shared-memory machines. *SIAM Journal on Computing*, 18(2):216–228, 1989.

[6] Mihai F Ionescu and Klaus E Schauser. Optimizing parallel bitonic sort. In *Proceedings 11th International Parallel Processing Symposium*, pages 303–309. IEEE, 1997.