中国科学技术大学
University of Science and Technology of China

并行计算三篇论文评论
Stencil 模板计算的 GPU 加速范例

谭邵杰 *SA21011013*

2022 年 3 月 23 日

目录

1 论文一

1.1 论文题目与来源

CCF A 类会议，计算机体系结构顶会 PPoPP 20 的一篇文章，

题目是 High-Level Hardware Feature Extraction for GPU Performance Prediction of Stencils[1], 来自苏格兰的爱丁堡大学 (The University of Edinburgh) 和德国的明斯特大学 (University of Münster)

1.2 概述总结与评论

(664 个字符 550 个中文字符)

High-level 函数式编程抽象已经在 HPC(高性能计算) 领域开始显示出的良好的前景。Lift、Futhark 或 Delite 等方法已经表明，High-level 抽象和性能两者兼得是可能的，甚至对于像 Stencil 模板计算这样的 HPC 工作负载也是如此。

此外，在编译器本身中，这些高级函数抽象还可以用来表示程序及其优化变体。

然而，这种高级方法严重依赖于编译器来优化程序，这在针对 gpu 时是非常困难的。编译器要么使用手工制作的启发式来指导优化，要么使用迭代编译来搜索优化空间。

第一种方法的编译时间很快，但是它不能跨不同的设备进行性能移植，并且需要大量的人力来构建启发式方法。

另一方面，迭代编译具有自动搜索优化空间的能力，并适应不同的设备。然而，这个过程通常非常耗时，因为需要评估数千种变体。

为了加快优化空间的探索，提出了基于统计技术的性能模型。然而，它们依赖于底层硬件特性，以性能计数器或底层静态代码特性的形式存在。

使用 Lift 框架，文章演示了如何从高级函数表示中直接提取低级的、特定于 gpu 的特征。Lift IR(中间表示) 实际上是一个非常合适的选择，因为所有的优化选择都暴露在 IR 水平。这篇文章展示了如何提取底层特征，比如每个 WARP 的唯一缓存行数，这对构建精确的 GPU 性能模型至关重要。

使用这种方法，我们能够在许多模板应用程序中，在 AMD GPU 上平均加速 2000 倍，在 Nvidia 上平均加速 450 倍。(这里不是指代码运行时间，是指编译器搜索出最佳优化策略的时间。

评论: Stencil GPU 编译器自动优化的又一前进的一步。

2 论文二

2.1 论文题目与来源

CCF A 类会议，计算机体系结构顶会 PPoPP 18 的一篇文章，

题目是 Register Optimizations for Stencils on GPUs[2], 来自美国俄亥俄州立大学 (The Ohio State University)

2.2 概述总结与评论

(331 个字符 306 个中文字符)

计算密集型 GPU 架构允许应用程序开发人员使用高阶 3DStencil 模板计算常见的一种优化策略, 通过循环展开等方式公开充分的数据重用，并期望实现寄存器级重用。

然而，产生的代码常常受到寄存器压力的高度限制。虽然目前最先进的寄存器分配器对大多数应用程序都很满意，但它们无法有效地管理这种复杂的高阶模板的寄存器压力，从而导致出现大量寄存器溢出的次优代码。

文章开发了一个语句重排序框架，该框架将模板计算建模为具有共享叶子的树的 DAG，并采用了一个最优调度算法来最小化表达式树的寄存器使用。通过对应用代码中提取的一系列模板的实验结果，验证了该方法的有效性。

评论: 提出了可以未来应用于编译器优化代码的一种自动化分析并且优化 Stencil 代码寄存器使用的方法，复杂但十分重要。

3 论文三

3.1 论文题目与来源

CCF B 类会议 Distributed, Parallel, and Cluster Computing 2020 的一篇文章，

题目是 Accelerating High-Order Stencils on GPUs[3], 来自美国莱斯大学 (Rice University)

3.2 概述总结与评论

(494 个字符 401 个中文字符符)

文章指出了在低阶 Stencil 模板计算的 GPU 加速理论已经得到了很好的研究，但是并不是所有的技术都适用于高阶 Stencil 模板，例如用于地震成像的 Stencil 计算。此外，应对对于

复杂的边界条件，通常需要不同的计算逻辑，这使得在 GPU 上高效利用线程级并行性变得复杂。

文章通过研究在具有有意义边界条件的大区域上的基于高阶模板的 GPU 地震成像计算。并通过手动构建包含 25 个点的 CUDA 代码。

从下面几个方面评估了 Stencil 代码性能：区域分解，采用 GPU 共享内存，采用寄存器位移数据分布形状，内存层次结构使用情况、数据预取模式和其他性能属性。

并使用几种成熟的新兴工具 (如高性能分析工具 HPCToolkit, NVIDIA 公司 Nsight) 进行了实证评估和讨论我们的定量研究结果。

最终实现的性能是使用 OpenACC 开发的 C 语言专有代码的两倍，并且通过实验证明具有出色的性能可移植性。

评论: 文章归纳了大部分的 Stencil GPU 优化的通用技巧，并且通过大量的实验数据的比较来寻找最佳的边界划分方法。但遗憾的是没有开源代码。而且所有工作都是手动的，未来应该与编译器结合来提高开发效率和可移植性。

参考文献

[1] Remmelg T, Hagedorn B, Li L, et al. High-level hardware feature extraction for GPU performance prediction of stencils[C]//Proceedings of the 13th Annual Workshop on General Purpose Processing Using Graphics Processing Unit. 2020: 21-30.

[2] Rawat P S, Rastello F, Sukumaran-Rajam A, et al. Register optimizations for stencils on GPUs[C]//Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. 2018: 168-182.

[3] Sai R, Mellor-Crummey J, Meng X, et al. Accelerating high-order stencils on GPUs[C]//2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). IEEE, 2020: 86-108.