PLANT：基于多面体模型的张量编译器

毕业设计开题报告

MashPlant 李晨昊

清华大学计算机科学与技术系

2021 年 1 月 5 日

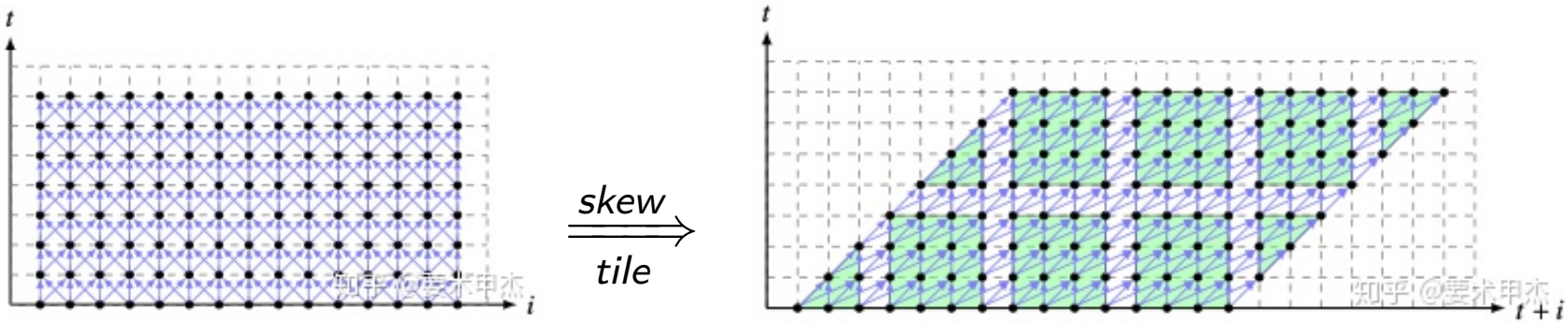
背景

- 张量（多维数组）计算在机器学习，图像处理，科学计算领域占据核心地位
- 随着体系结构的复杂性，高层计算框架和算子的多样性的增加，生成高效代码变得越来越困难：
  - 代码和数据布局转换
  - 硬件计算资源利用
  - 内存层次结构管理
  - 数据同步和通信
- 代码优化的核心是循环变换

多面体编译

- 将迭代范围抽象成多面体，即一组仿射不等式定义的集合
- 将内存访问，循环变换等抽象成仿射变换

```
for (int t = 0; t < T; ++t)
  for (int i = 1; i < N - 1; ++i)
    A[t + 1][i] = (A[t][i - 1] - 2 * A[t][i] + A[t][i + 1]) / 4;
```



[11]

相关工作

1. 人工调度的非多面体编译器：Halide [2013]，TVM [2018]
   - 需要手工编写调度指令，要求对体系结构的理解；可通过 AutoTVM [2018]，Ansor [2020] 等实现一定的自动化
   - 程序表达能力有限，基于区间的程序表示难以精确表达非矩形的迭代空间，非完美嵌套的循环，循环依赖的数据流图等
2. 自动调度的多面体编译器：PLuTo [2008]，PPCG [2013]
3. 人工调度的多面体编译器：CHiLL [2009]，AlphaZ [2013]，TIRAMISU [2019]

相关工作

1. 人工调度的非多面体编译器：Halide [2013]，TVM [2018]
2. 自动调度的多面体编译器：PLuTo [2008]，PPCG [2013]
   - 自动调度缺少精确的代价模型，生成的代码速度较低，用户也难以控制调度
   - 涉及复杂的整数线性规划，编译速度较低
   - 调度指令覆盖面有限，缺少硬件相关的调度指令
3. 人工调度的多面体编译器：CHiLL [2009]，AlphaZ [2013]，TIRAMISU [2019]

相关工作

1. 人工调度的非多面体编译器：Halide [2013]，TVM [2018]
2. 自动调度的多面体编译器：PLuTo [2008]，PPCG [2013]
3. 人工调度的多面体编译器：CHiLL [2009]，AlphaZ [2013]，TIRAMISU [2019]
   - 需要手工编写调度指令，且缺少自动调度相关的工作
   - 调度指令覆盖面有限

简介

- PLANT: PoLyhedral bAsed teNsor opTimizer
- 主体是人工调度的多面体编译器，向用户提供调度指令，同时借鉴非多面体编译器中的自动化方法
- 结合三类工作的优点：多面体模型的强大表达能力，高效的编译速度和优化效果，一定程度的自动化

技术方案

- 使用 Rust 构建系统
- 以 DSL 为输入，避免复杂的程序分析
- 计划实现 CPU(C/LLVM IR) 和 GPU(CUDA) 两个后端
- 补充现有的多面体编译器缺少的调度指令：管理内存层次结构，利用专用硬件原语

IR 设计

- 借鉴 TIRAMISU [2019] 中分层描述 IR 的思想，分离算法描述，循环转换，内存映射

```
let f = Func::new("matmul");
// algorithm description
let (ref i, ref j, ref k) = (f.iter(0), f.iter(1), f.iter(2));
let a = f.buf("a", I32, In, &[n, s]);
let b = f.buf("b", I32, In, &[s, m]);
let c_init = f.comp("C_init", &[(0, n), (0, m)], 0i32);
let c = f.comp("C", &[(0, n), (0, m), (0, s)], 0i32);
// C[i, j, k] = A[i, k] * B[k, j] + C[i, j, k - 1]
c.set_expr(a.at(&[i, k]) * b.at(&[k, j]) + c.at(&[i, j, &(k - 1)]));
// loop transformation
c_init.tile(0, 1, 32, 32);
c.tile(0, 1, 32, 32);
c.after(c_init, 4);
c.tag_dim(0, Parallel);
// memory mapping
let buf_c = f.buf("c", I32, Out, &[n, m]);
c_init.store(buf_c);
c.store_at(buf_c, &[i, j]);
f.codegen(&[a, b, buf_c], "matmul.c")
```

自动调度

- 实现基于模板的自动调度器，在用户定义的搜索空间上优化程序
- 复杂算子的搜索空间大小可达十亿级别，借鉴 AutoTVM [2018]，从程序表示中抽取特征，使用机器学习算法搜索最优参数组合

```
cfg.define_knob("tile_i", &[2, 4, 8, 16, 32]);
cfg.define_knob("tile_j", &[2, 4, 8, 16, 32]);
...
let (tile_i, tile_j) = (*cfg["tile_i"], *cfg["tile_j"]);
c_init.tile(0, 1, tile_i, tile_j);
c.tile(0, 1, tile_i, tile_j);
...
```

- 已经完成：实现 ISL [2010] 的 Rust Binding
- 一月：构建系统，实现关键调度指令和 CPU 代码生成
- 二月：实现大部分调度指令和 GPU 代码生成，测试运行简单 kernel
- 三月：实现自动调度器，测试运行一些有代表性的网络模型
- 四、五月：性能测试及调优，论文撰写

[1] R. Baghdadi, J. Ray, M. B. Romdhane, E. Del Sozzo, A. Akkas, Y. Zhang, P. Suriana, S. Kamil, and S. Amarasinghe. Tiramisu: A polyhedral compiler for expressing fast and portable code. In *Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization*, CGO 2019, page 193–205. IEEE Press, 2019. ISBN 9781728114361.

[2] U. Bondhugula, A. Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. In *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '08, page 101–113, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595938602. doi: 10.1145/1375581.1375595. URL https://doi.org/10.1145/1375581.1375595.

[3] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, M. Cowan, H. Shen, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy. Tvm: An automated end-to-end optimizing compiler for deep learning. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, OSDI'18, page 579–594, USA, 2018. USENIX Association. ISBN 9781931971478.

[4] T. Chen, L. Zheng, E. Yan, Z. Jiang, T. Moreau, L. Ceze, C. Guestrin, and A. Krishnamurthy. Learning to optimize tensor programs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 3393–3404, Red Hook, NY, USA, 2018. Curran Associates Inc.

[5] M. Hall, J. Chame, C. Chen, J. Shin, G. Rudy, and M. Khan. Loop transformation recipes for code generation and auto-tuning. volume 5898, pages 50–64, 10 2009. ISBN 978-3-642-13373-2. doi: 10.1007/978-3-642-13374-9_4.

[6] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe. Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *SIGPLAN Not.*, 48(6):519–530, June 2013. ISSN 0362-1340. doi: 10.1145/2499370.2462176. URL https://doi.org/10.1145/2499370.2462176.

[7] S. Verdoolaege. isl: An integer set library for the polyhedral model. In K. Fukuda, J. v. d. Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software – ICMS 2010*, pages 299–302, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15582-6.

[8] S. Verdoolaege, J. Carlos Juega, A. Cohen, J. Ignacio Gómez, C. Tenllado, and F. Catthoor. Polyhedral parallel code generation for cuda. *ACM Trans. Archit. Code Optim.*, 9(4), Jan. 2013. ISSN 1544-3566. doi: 10.1145/2400682.2400713. URL https://doi.org/10.1145/2400682.2400713.

[9] T. Yuki, G. Gupta, K. Daegon, T. Pathan, and S. Rajopadhye. Alphaz: A system for design space exploration in the polyhedral model. pages 17–31, 01 2013. doi: 10.1007/978-3-642-37658-0_2.

[10] L. Zheng, C. Jia, M. Sun, Z. Wu, C. H. Yu, A. Haj-Ali, Y. Wang, J. Yang, D. Zhuo, K. Sen, et al. Ansor: Generating high-performance tensor programs for deep learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 863–879, 2020.

[11] 要术甲杰. Polyhedral 编译调度算法 (1)—Pluto 算法. https://zhuanlan.zhihu.com/p/199683290.

Thanks!