PLANT：基于多面体模型的张量编译器

中期报告

MashPlant 李晨昊

清华大学计算机科学与技术系

2021 年 3 月 8 日

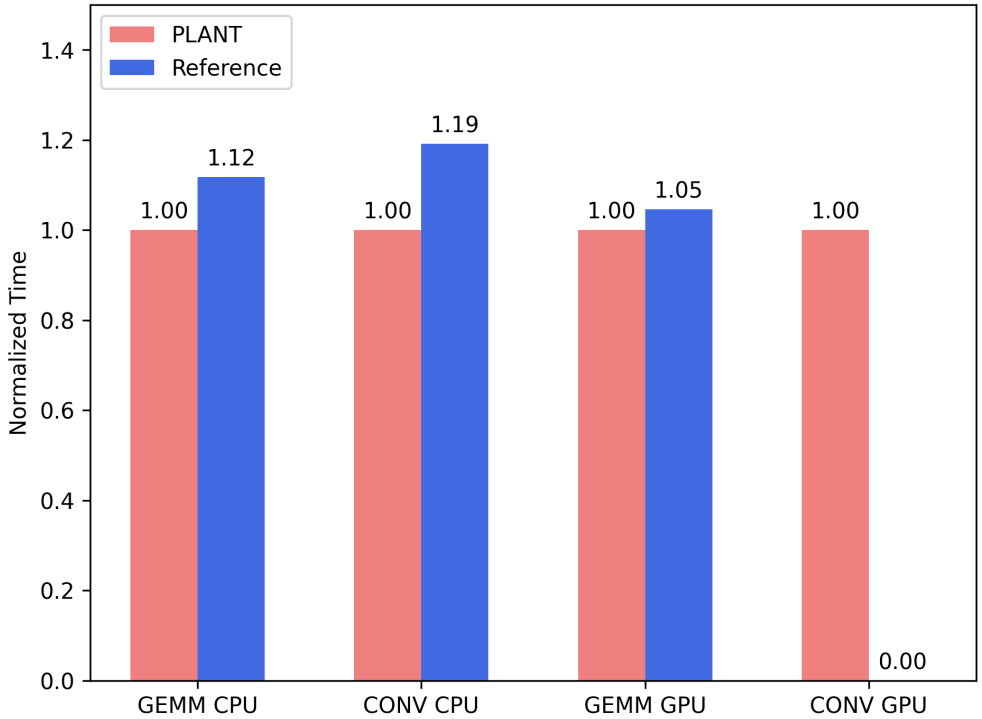
简介

- PLANT: PoLyhedral bAsed teNsor opTimizer
- 人工调度的多面体编译器，多面体模型仅用来实现循环调度，向用户提供调度指令
- 借鉴非多面体编译器中的自动化方法

进展

- 基本完成调度指令实现和自动调度器
- 实现了 C 和 CUDA 的代码生成
- 实现了 CPU，GPU 上的 GEMM，CONV
- 实现了 CPU 上的 ResNet[2016] V1

进展



算子[1]：对比 MKL[2],CUBLAS[3]

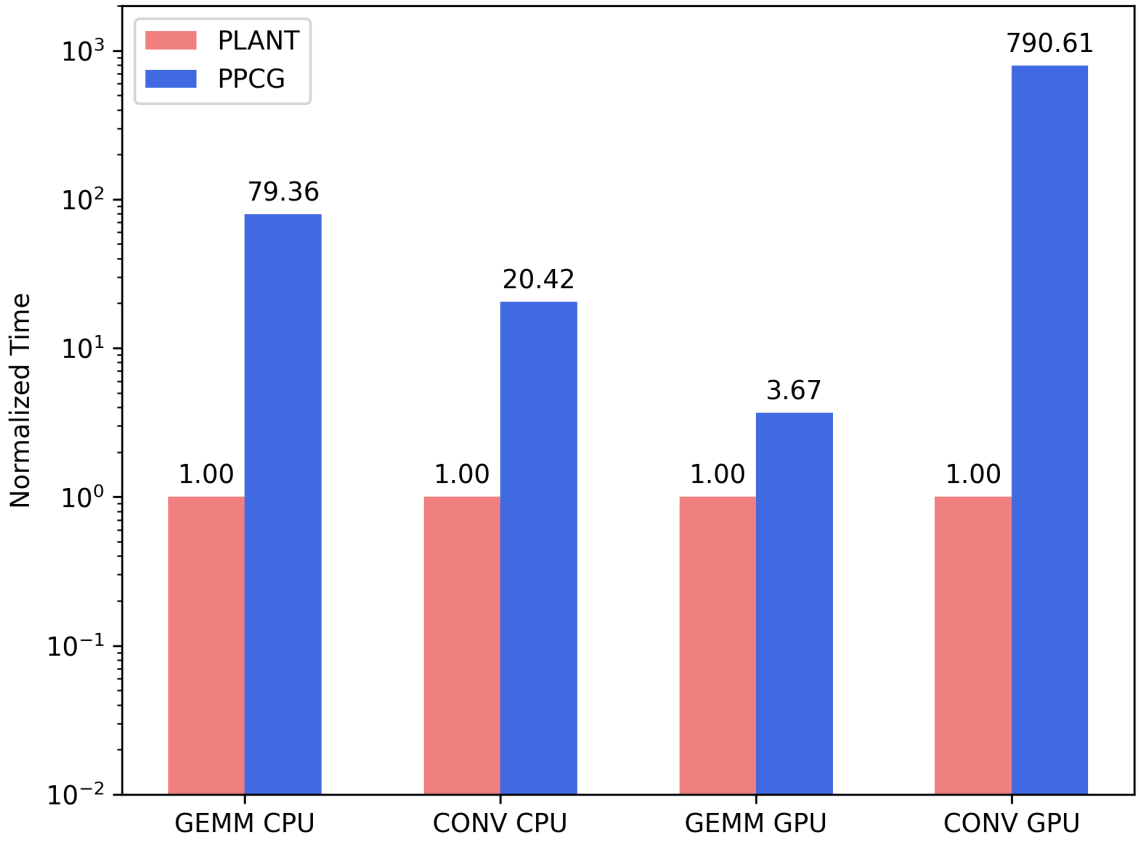
---

[1] GEMM: $M \times K \times N = 2048 \times 2048 \times 2048$, CONV: $NCHW * OIHW = (256 \times 256 \times 14 \times 14) * (512 \times 256 \times 3 \times 3)$

[2] Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz

[3] NVIDIA Tesla P100, CUDA 10.2

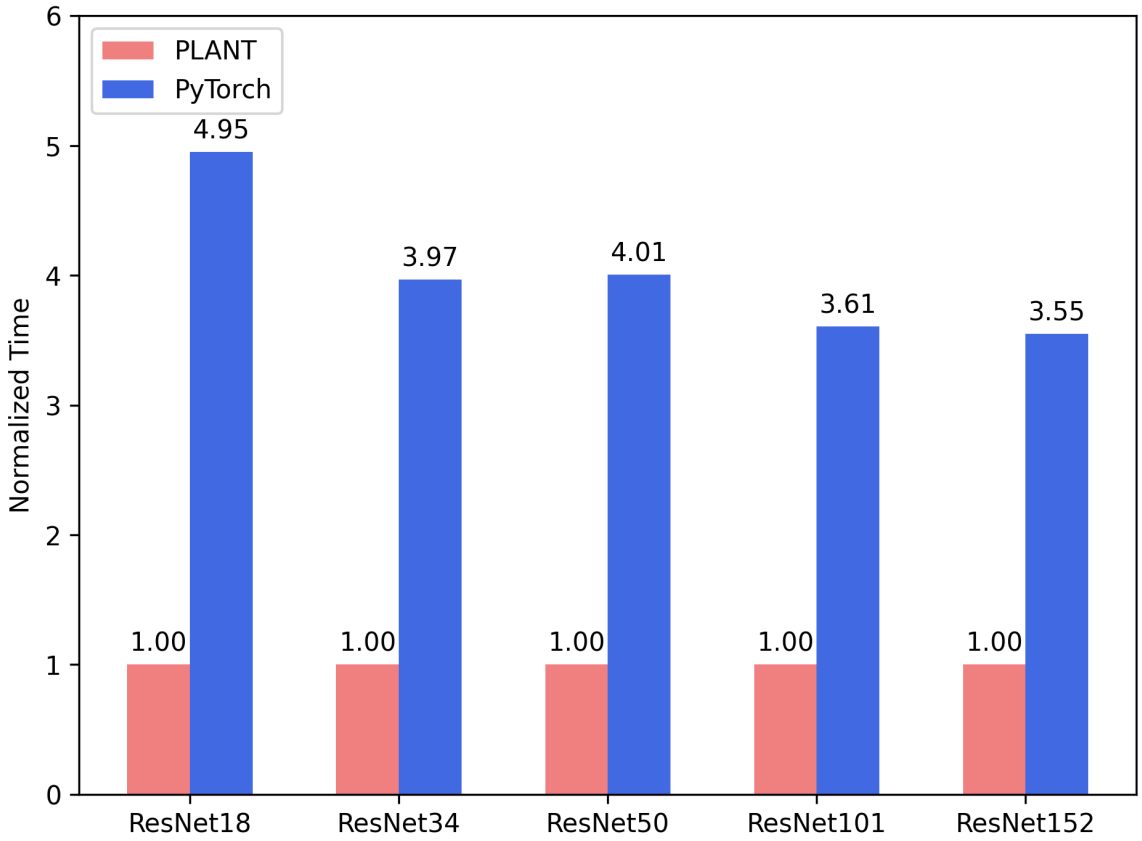
进展



算子：对比 PPCG [2013] [1]

[1] commit 8a74e46

进展



ResNet：对比 PyTorch[1]

[1] torch 1.7.1，torchvision 0.9.0, batch = 1

循环调度指令

指令	描述
split	将循环分裂为嵌套的两层
fuse	合并两个相邻的嵌套循环
reorder	调整循环的嵌套顺序
tile	利用 split 和 reorder 实现循环分块
skew	循环倾斜
shift	循环索引偏移
before/after	控制循环体执行位置
separate	将循环拆分为两个并列的循环
tag	将循环绑定到硬件资源 (并行/向量化/GPU)
apply_sch	底层原语，执行任意仿射变换

内存调度指令

指令	描述
store	控制计算结果的保存位置
cache_identity	自动完成恒等映射的 cache
cache	用户手动控制 cache 位置
set_loc	控制 Buf 存储位置
alloc_at	控制 Buf 申请和释放的位置
auto_transfer	控制 GPU Buf 在函数开头结尾传输数据

IR 设计

- 借鉴 TIRAMISU [2019] 中分层描述 IR 的思想，分离算法描述，循环调度指令，内存调度指令

```
let f = Func::new("matmul");
// algorithm description
let a = f.buf("a", I32, In, x![n, s]);
let b = f.buf("b", I32, In, x![s, m]);
let c_init = f.comp("C_init", x![n, m], x!(0));
let c = f.comp("C", x![n, m, s], x!(0));
c.set_expr(x!(a(i0, i2) * b(i2, i1) + c(i0, i1, i2 - 1)));
// loop transformation
c_init.tile(0, 1, 32, 32);
c.tile(0, 1, 32, 32);
c.after(c_init, 4);
c.tag(0, Parallel);
// memory mapping
let buf_c = f.buf("c", I32, Out, x![n, m]);
c_init.store(buf_c);
c.store_at(buf_c, x![i0, i1]);
f.codegen(&[a, b, buf_c]);
```

IR 设计

- 使用 Rust 过程宏提供表达式 DSL，实现对条件运算符，循环等的“重载”

```
let a_pad = c!(for n in 0..batch {
  for c in 0..channel {
    for y in 0..size + 2 * pad {
      for x in 0..size + 2 * pad {
        if y >= pad && y - pad < size && x >= pad && x - pad < size
          { a(n, c, y - pad, x - pad) } else { 0f32 }
      }
    }
  }
});
```

```
Apad = te.compute(
  (batch, channel, size + 2 * pad, size + 2 * pad),
  lambda n, c, y, x: tvm.tir.if_then_else(
    tvm.tir.all(y >= pad, y - pad < size, x >= pad, x - pad < size),
    A[n, c, y - pad, x - pad], tvm.tir.const(0.0, "float32")),
  name="Apad")
```

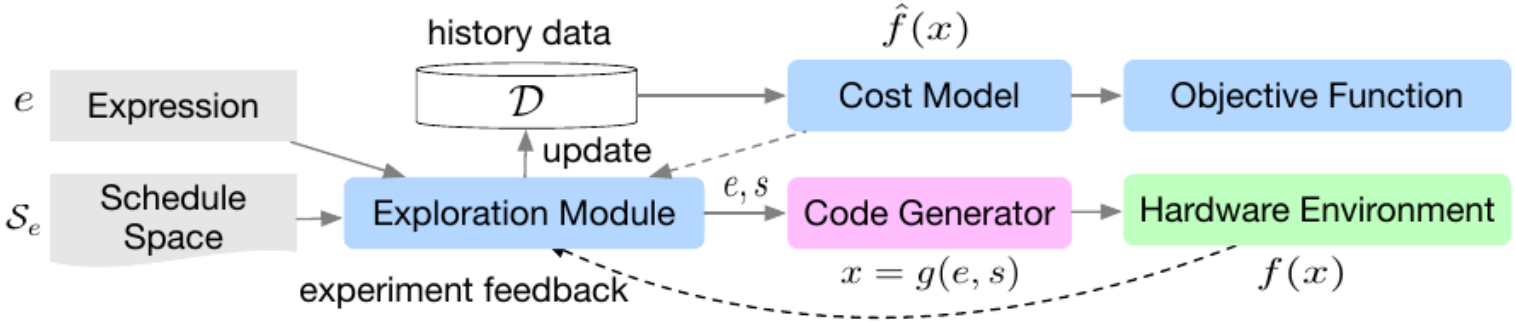
自动调度

- 实现了基于模板的 Auto-Tuning，在用户定义的搜索空间上调优参数
- 实现的调优选项：
  - Knob：用户提供一组可选值
  - Split：自动计算循环分裂的可能参数，可选因子/2 的幂次
  - Tag：调优循环 tag
  - Reorder：调优 Reorder 顺序

```
space.define_split("sp", SplitPolicy::new(oc)
  .set_n_output(4));
...
let sp = cfg.get("sp");
b.split(0, sp[0]).split(0, sp[1]).split(0, sp[2]);
...
```

自动调度

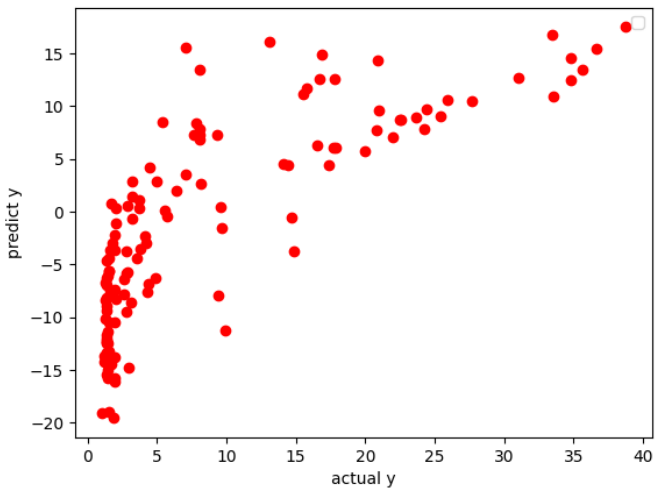
- 复杂算子的搜索空间无法穷尽搜索，使用模拟退火算法选择性地遍历
- 对枚举到的每个程序抽取特征向量，使用 XGBoost [2016] 模型计算估值函数，程序运行结果用于更新模型



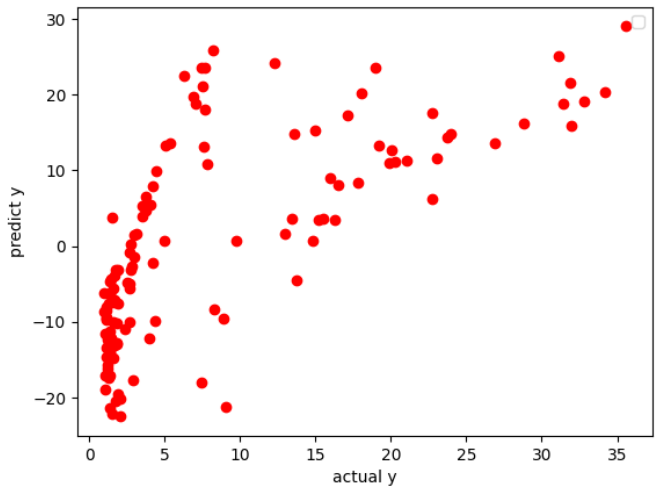
AutoTVM [2018] 架构

自动调度

- 特征抽取方法：
  - Knob：直接用参数值作为向量，其中 Tag 选项采用 One-Hot Encoding
  - Iter：对每层循环，抽取浮点运算次数，循环变量参与的访存的 stride，count，reuse 等信息，展开成向量
- 简单程序上差距不大，且 Knob 速度更快



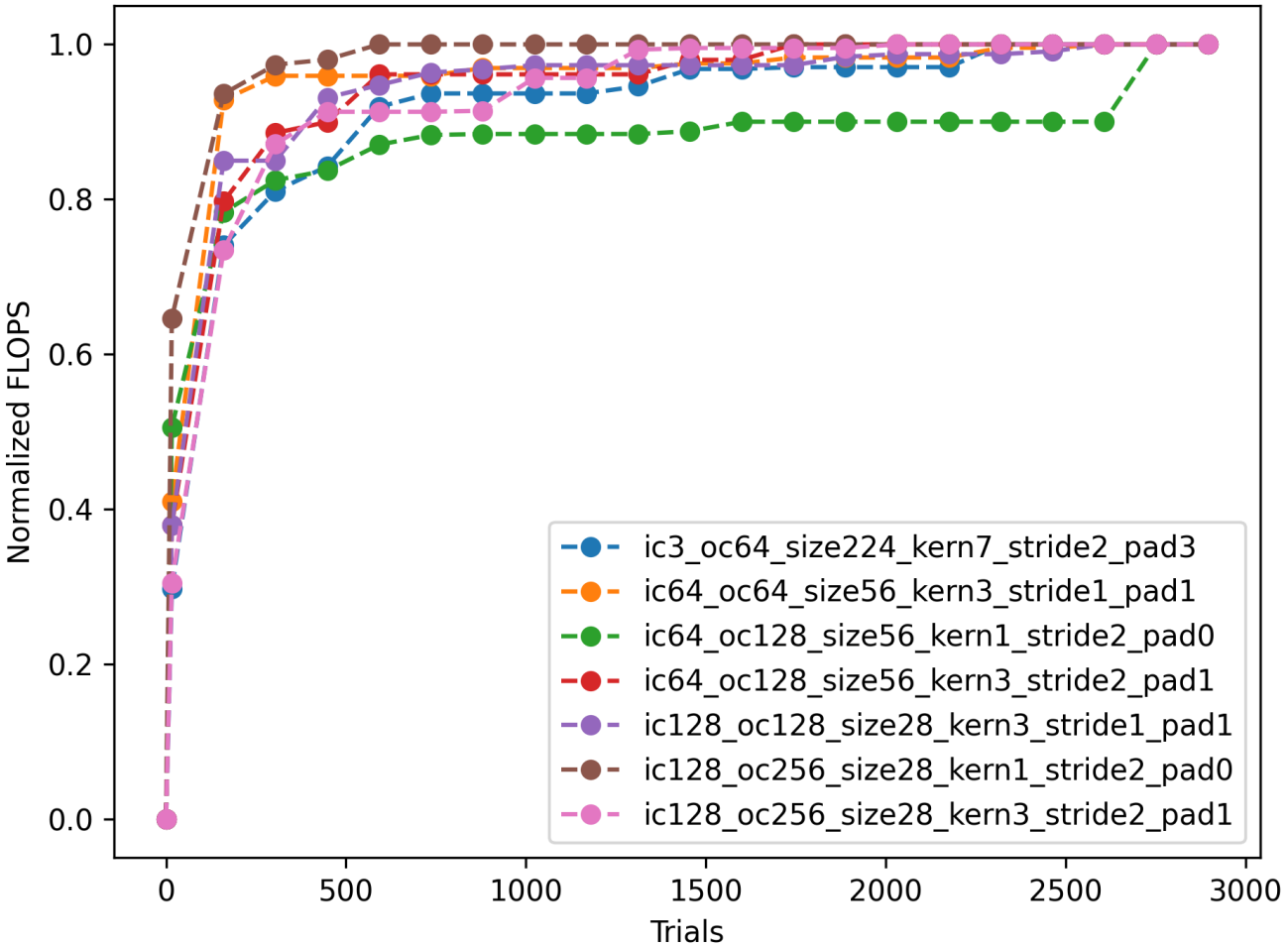
Knob



Iter

自动调度

- 自动调度 ResNet 中部分 CONV 的性能曲线



进度总结

- 构建系统，实现关键调度指令和 CPU 代码生成 ✓
- 实现大部分调度指令和 GPU 代码生成 ✓
- 实现自动调度器 ✓
- 测试运行几个重要的 kernel ✓
- 测试运行有代表性的网络模型 ✓
- 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] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

[3] 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.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] 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.

Thanks!