Tune Performance Domain Specific Language-Halide

wegatron

December 9, 2021

Table of contents

性能挑战

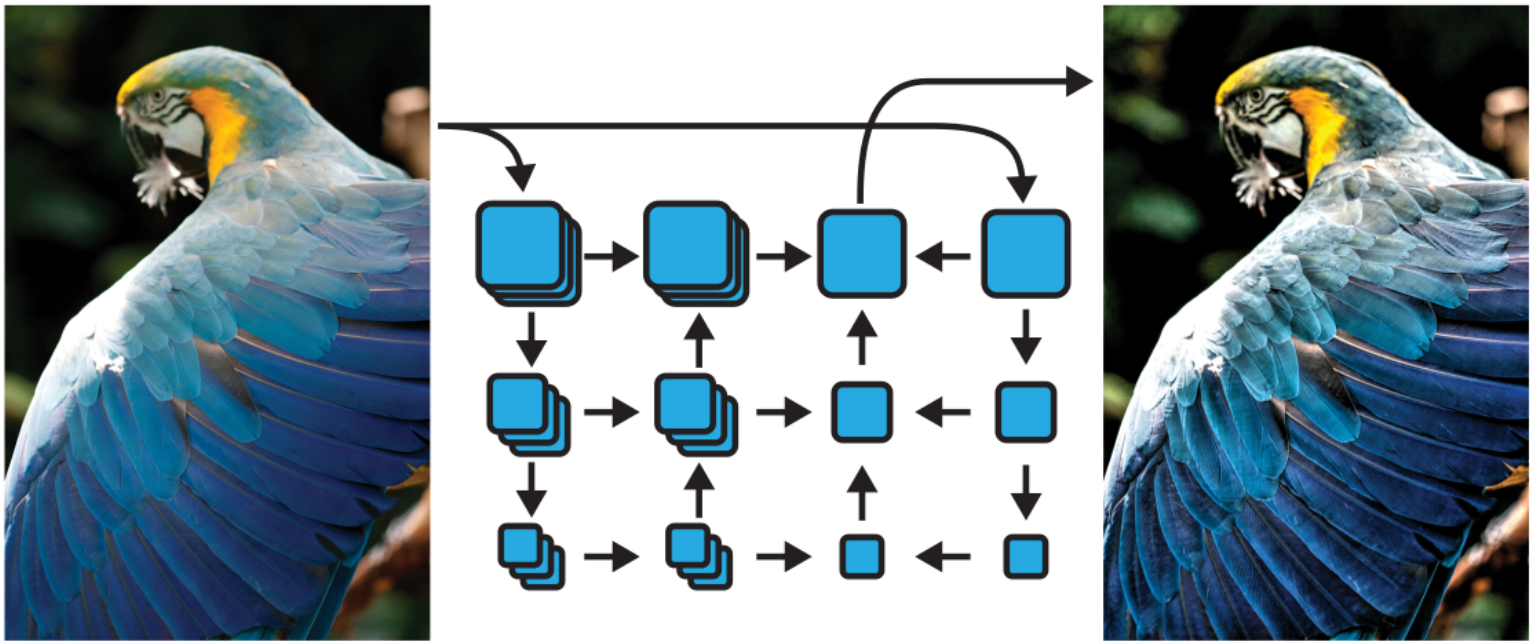

Figure: 在实际应用中，我们往往需要绞尽脑汁对算法做各种优化，以满足在移动设备上的流畅运行.

实例分析-初始代码

(a) Clean C++ : 9.94 ms per megapixel

```
void blur(const Image &in, Image &blurred) {
 Image tmp(in.width(), in.height());

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;

 for (int y = 0; y < in.height(); y++)
  for (int x = 0; x < in.width(); x++)
   blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;
}
```

Figure: box filter simple c++ code

实例分析-优化后的代码

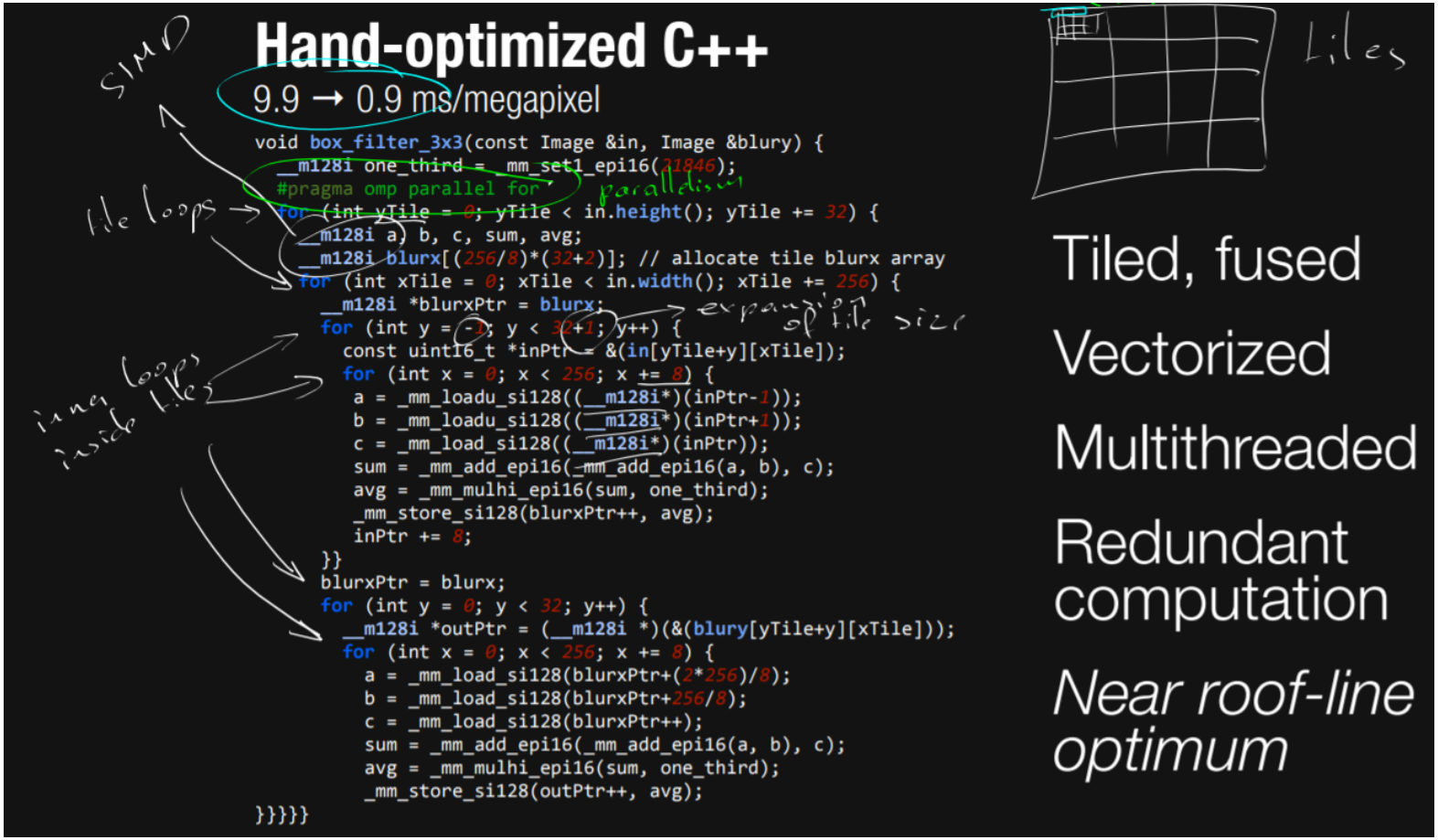

Figure: 使用矢量计算, tiling, 并行优化之后的快 10 倍的代码

影响因素分析

以出行做类比:

- 算法的设计/参数——计算量 (O(n), O(n^2), ...).
  出行的目的地/路线 (抄小道)
- 代码实现. 重复计算/无效计算, 并行, 矢量计算, 内存操作.
  出行具体时间的安排 (堵车, 等车...)
- 硬件能力.
  出行交通工具 (步行, 骑车, 汽车, 飞机)

优化方向

我们分两方向进行思考:

- Algorithm: 如何设计高效的算法 (算法定义了计算量)
- Schedule: 如何将算法编码为高效的机器二进制码

The Schedule

Schedule 包含的内容:

- 遍历的顺序
- 引用的计算是重算还是复用之前结果
- 如何将任务映射到多线程并行, SIMD 指令, GPU

Schedule Example

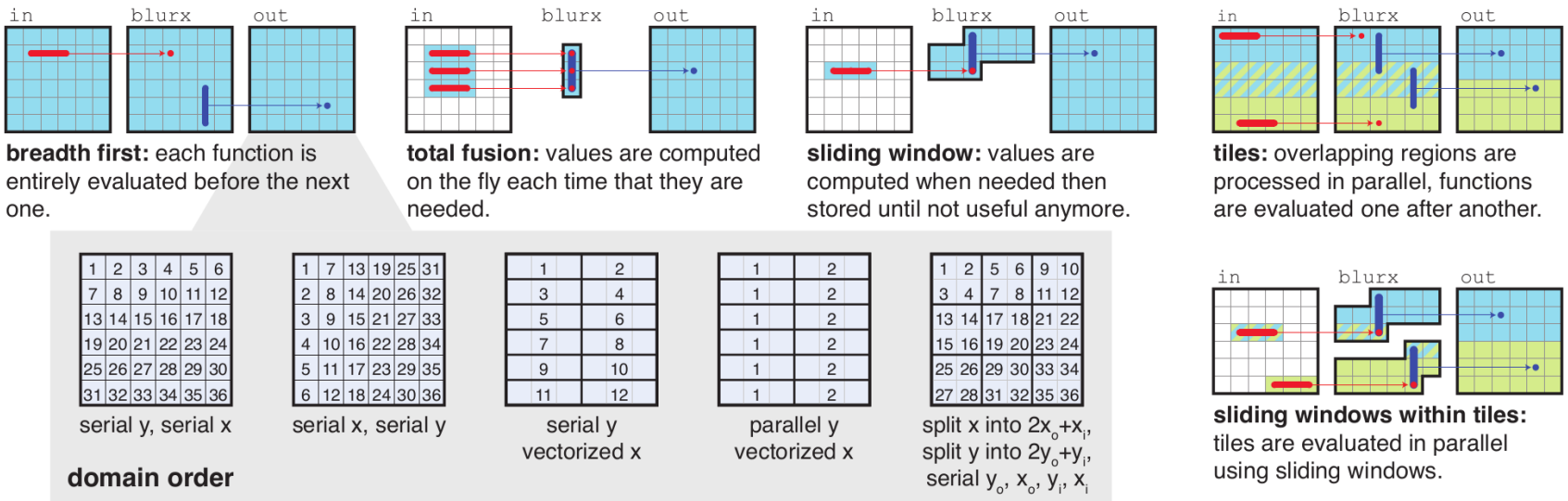

Figure: box filter 的几种可能的 Schedule 方案

直接优化代码

- 对程序员有很高的要求.
- 费力耗时
- 且代码与平台相关性高不易移植.

Using Halide

- Schedule 和 Algorithm 分离
- 支持多种平台代码生成 (cpu, metal, opengl, cuda ...)

Halide-Example

(c) Halide : 0.90 ms per megapixel

```
Func halide_blur(Func in) {
 Func tmp, blurred;
 Var x, y, xi, yi;

 // The algorithm
 tmp(x, y) = (in(x-1, y) + in(x, y) + in(x+1, y))/3;
 blurred(x, y) = (tmp(x, y-1) + tmp(x, y) + tmp(x, y+1))/3;

 // The schedule
 blurred.tile(x, y, xi, yi, 256, 32)
        .vectorize(xi, 8).parallel(y);
 tmp.chunk(x).vectorize(x, 8);

 return blurred;
}
```

Figure: halide box filter 代码实现

自动调优

Schedule 独立, 方便了性能调优, 但任然需要耗时费力的尝试.
有没有办法, 自动得到一个好的方案?

解空间的建模

Schedule 变化可以总结为两个方面:

- The Domain Order, 定义了每个函数空间的遍历方式
  - 顺序/并行
  - 循环展开以及矢量化
  - column major/row major
  - 拆分为多个小片, 按片遍历
- The Call Schedule, 定义了函数存储和计算的粒度如:
  - breadth-first, 先计算 blur-x 存储起来, 然后再计算 blur
  - fused, 每次重新计算 blur-x
  - sliding-window

解空间的搜索

变化的空间非常大, 使用遗传算法, 并通过一些先验来搜索. 一些策略:

- 从 breadth-first 开始进行搜索
- 片的长度的选择为 2 的整数倍
- 同一个函数采用相同的策略
- ...

Compiler Pipeline

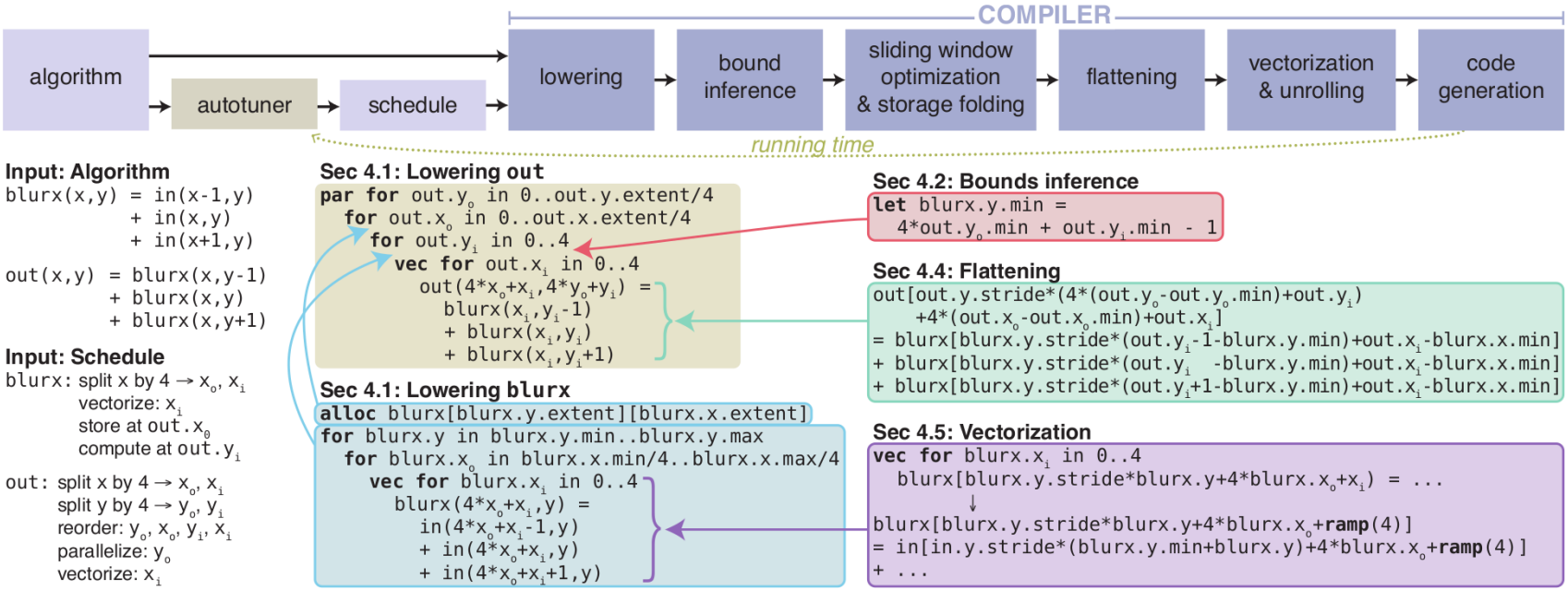

Figure: 编译流程

Reference

- Decoupling Algorithms from Schedules for Easy Optimization of Image Processing Pipelines
- Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines