计算机系统实验

第二次实验

CONTENTS

目录

Lab2 Perflab 性能实验

实验背景与目标

- Why Performance Optimization?
- 实际案例：
  - 图像处理（如Photoshop滤镜、医学影像）。
  - 科学计算（如气候模拟、分子动力学）。
- 性能提升的直接影响：用户体验、能耗、成本。

性能优化的核心挑战

- 硬件特性：CPU流水线、多级缓存、分支预测。
- 软件目标：减少指令数、提高并行性、优化访存模式。

实验目标

- 理解程序性能优化的核心思想，掌握数据局部性、循环展开等优化技术。
- 掌握优化图像处理方法，提升内存访问和计算效率，降低CPE。

实验环境介绍

- **实验环境：Linux 64-bit，C/汇编语言**
  - **T2507实验室，**推荐使用**WSL**
  - **T2617实验室，**推荐使用**Ubuntu**虚拟机，详见：Ubuntu虚拟机使用指南
  - 备选：远程实验平台，详见：远程实验平台使用指南
  - **可以使用自己搭建的 Linux 环境，**工具依赖：**gcc make gdb** 等
- **Windows和Linux之间互传文件工具：Mobaxterm、 VScode**

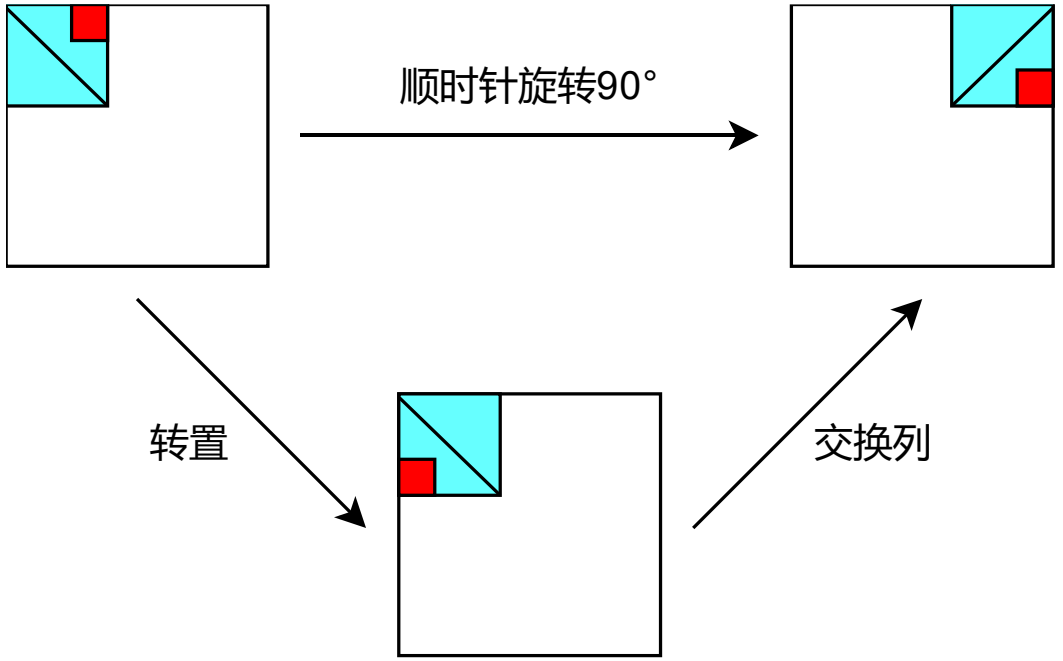
实验数据与文件

- **学生实验数据包：**perflab.tar
- **解压命令：**tar xvf perflab.tar （上传到Linux环境后解压）
- 数据包中主要包含下列文件：
  - **kernels.c**：图像处理核心代码，已提供初始实现 **naive_rotate**（图像旋转）和 **naive_smooth**（掩膜平滑），需通过优化重新实现 **rotate** 和 **smooth** 函数以提升性能。
  - **driver.c**：测试驱动，负责测试和评估图像处理函数性能。
  - **fcyc.c/clock.c**：高精度计时与性能测量工具

实验任务

任务1：图像旋转（Rotate）

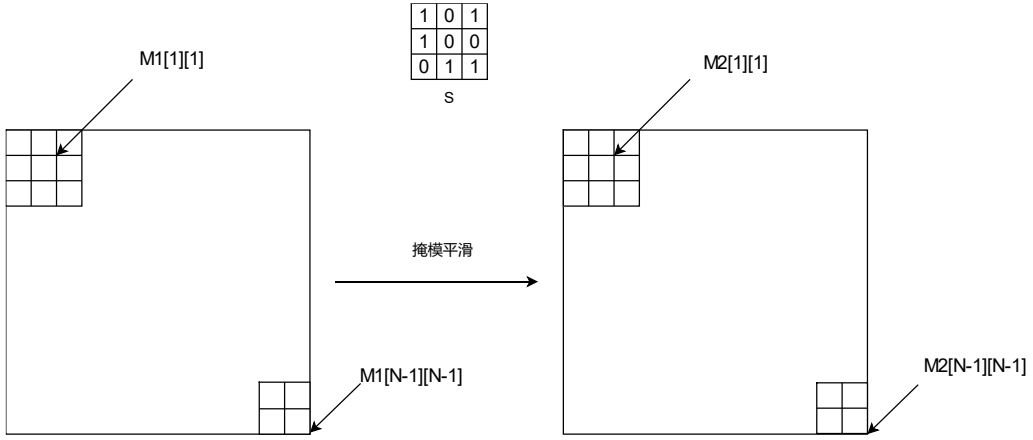
- **输入**：N×N像素的图像矩阵。
- **输出**：顺时针旋转90度的N×N图像。
- **关键挑战**：
  - 内存访问模式优化（避免跨步访问导致的Cache Miss）。
  - 循环结构的效率提升。



实验任务

任务2：图像平滑（Smooth）

- **输入**：N×N像素的图像矩阵。
- **卷积核**：**由学号后四位生成3×3的矩阵（每个同学不一样）**
- **输出**：每个像素替换为周围3×3区域的**均值**。
- **关键挑战**：
  - 减少重复计算（边界处理、中间值复用）。
  - 优化计算密集型循环。



例如，图像中的 M2 [1][1] 和 M2 [N - 1][N - 1] 的计算方式如下：

$$M_2[1][1] = \sum_{i=0}^{2}\sum_{j=0}^{2} \frac{M_1[i][j] * S[i][j]}{\sum_{i=0}^{2}\sum_{j=0}^{2} S[i][j]}$$

$$M_2[N-1][N-1] = \sum_{i=N-2}^{N-1}\sum_{j=N-2}^{N-1} \frac{M_1[i][j] * S[i-(N-2)][j-(N-2)]}{\sum_{i=0}^{1}\sum_{j=0}^{1} S[i][j]}$$

性能评价标准：CPE和加速比

- **CPE（每元素周期数）**
  - 衡量处理每个像素的平均开销，越低越好。
  - 示例：处理 64×64 图像（4096 像素）耗时 1000 cycles →

    CPE = 1000/4096 ≈ 0.244。
- **加速比（Speedup）**

$$\text{加速比}=\frac{\text{原始代码}_{\text{CPE}}}{\text{优化代码}_{\text{CPE}}}$$

个越高越好！

  - **>1**：优化有效　　**=1**：无优化　　**<1**：性能倒退
  - ⚠ **正确性一票否决**：若优化后的函数输出与基准不一致，**加速比记为0！**
  - 最终得分：不同图像尺寸下加速比的平均值。
    - Rotate 图像尺寸：64×64、128×128、256×256、512×512。
    - Smooth 图像尺寸：32×32、64×64、128×128、256×256。

性能优化技术——内存访问优化

- **问题**：旋转函数中的跨步访问（Stride）导致Cache Miss。
- **优化方法**：
  - 分块（Blocking）技术：将大矩阵拆分为小块，提高空间局部性。
  - 代码对比：

```
// 优化前（逐列访问）
for (i=0; i< N; i++)
  for (j=0; j<N; j++)
    dst[j][N-1-i] = src[i][j];

// 优化后（分块访问）
for (i=0; i< N; i+=BLOCK)
  for (j=0; j<N; j+=BLOCK)
    // 处理BLOCK×BLOCK子矩阵
```

- **效果**
  - 减少Cache Miss率。

性能优化技术——循环展开

- **问题**：循环控制开销（条件判断、计数器更新）。
- **优化方法**：
  - 手动展开内层循环（如展开4次），减少分支指令分块。
  - 代码示例：

```
// 优化前
for (i=0; i<N; i++) { sum += a[i]; }

//优化后（展开4次）
for (i=0; i<N; i+=4) {
  sum += a[i] + a[i+1] + a[i+2] + a[i+3];
}
```

- **效果**
  - 减少分支预测错误

  提高指令级并行（IPC）率

实验步骤

- **Step1：填写姓名和学号（kernels.c）**
- **Step2：编译与运行**
  - make clean && make

  # 清除中间文件和目标文件，再重新生成
  - ./driver　　# 运行
- **Step3：逐步优化与验证**
  - 修改kernels.c
  - 应用一种优化技术 → 重新编译 → 运行测试。

```
/*
 * Please fill in the following student struct
 */
student_t student = {
    "李明",              /* 姓名 */

    "190110218"     /* 学号 */
};
```

```
姓名：李明
学号：190110218
由学号生成的卷积核如下：
0 1 0
1 1 1
0 0 1

Rotate:
Dim             64      128     256     512     Mean
naive CPEs      3.2     4.6     7.6     12.7
your CPEs       3.2     4.6     7.6     12.7
Speedup         1.0     1.0     1.0     1.0     1.0

Smooth:
Dim             32      64      128     256     Mean
naive CPEs      69.6    70.7    70.1    70.8
your CPEs       70.2    70.4    70.1    70.5
Speedup         1.0     1.0     1.0     1.0     1.0

Summary of Your Best Scores:
  Rotate: 1.0
  Smooth: 1.0
```

参考资料

- **推荐阅读：**
  - 《Computer Systems: A Programmer’s Perspective》第5-6章。
- **在线资源：**
  - CS:APP官网实验材料：http://csapp.cs.cmu.edu/3e/labs.html
  - **课程主页及指导书**（校内网）：https://comp3052-2.p.cs-lab.top

作业提交

提交内容：实验报告（有模板）+ kernels.c（非压缩格式）

截止时间：

实验课后两周内提交至HITsz Grader 作业提交平台，具体截止日期参考平台发布。

- 登录网址： http://grader.tery.top:8000/#/login
- 推荐浏览器： Chrome
- 初始用户名、密码均为学号，登录后请修改

注意

上传后可自行下载以确认是否正确提交

同学们
请开始实验吧！