

Perflab 实验日志

优化代码 1

```
char rotate_descr[] = "rotate: Current working version";
void rotate(int dim, pixel *src, pixel *dst)
{
    int i,j,temp;
    for(j=0;j<dim;j++)
    {
        temp=dim-1-j;
        for(i=0;i<dim;i++)
        {
            dst[RIDX(temp,i,dim)]=src[RIDX(i,j,dim)];
        }
    }
}
```

详细注释：如上优化代码所示，进行了两部优化

A:将循环 i 放在外边，循环 j 放在里边

B:提前计算需要重复计算的 $\text{dim}-1-j$ ，在循环里直接使用结果

优化结果：

```
zhangjiwei@zhangjiwei-virtual-machine:~/perflab-handout$ ./driver
Teamname: first
Member 1: zhang Jiwei
Email 1: 2350700380@qq.com

Rotate: Version = naive_rotate: Naive baseline implementation:
Dim      64      128      256      512      1024      Mean
Your CPEs  2.2      3.6      4.6      8.5      10.6
Baseline CPEs 14.7     40.1     46.4     65.9     94.5
Speedup    6.7      11.1     10.0      7.8      8.9      8.7

Rotate: Version = rotate: Current working version:
Dim      64      128      256      512      1024      Mean
Your CPEs  1.7      1.9      2.5      4.4      7.2
Baseline CPEs 14.7     40.1     46.4     65.9     94.5
Speedup    8.5      20.9     18.4     14.9     13.2     14.5
```

优化思路及实现过程：

A:将循环 i 放在外边，循环 j 放在里边

减少了每次指针的移动弄位置，当 j 为循环外围时，每次循环都要将指针移动一个横排那么长，当 i 为循环外围时，每次只需寻址位置+1 即可。

B:减少重复计算

在原函数中，中间有一步 $\text{dim}-i-j$ 有重复运算，因此计划将其提前计算，在循环中直接使用即可。

优化代码 2

详细注释：建立临时变量储存需要重复计算的数据

每次循环 32 个数据 来求 cache 的命中

优化思路：将原循环分成 32×32 小块，提高空间的局部性，减少程序中的跳转语句。

实现过程：将其划分为 32×32 像素的小块，每次转移 32 个数据

```

char rotate_descr[] = "rotate: Current working version";
void rotate(int dim, pixel *src, pixel *dst)
{
    int i,j,i1,j1;
    for (i1 = 0; i1< dim; i1+=32)
        for (j1 = 0; j1< dim; j1+=32)
            for(i=i1;i<i1+32;i++)
                for(j=j1;j<j1+32;j++)
                    dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];
}

```

优化结果：

Rotate: Version = naive_rotate: Naive baseline implementation:						
Dim	64	128	256	512	1024	Mean
Your CPEs	2.2	3.3	4.8	17.8	10.3	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	6.6	12.0	9.6	3.7	9.1	7.6
Rotate: Version = rotate: Current working version:						
Dim	64	128	256	512	1024	Mean
Your CPEs	1.7	1.8	1.8	2.2	3.7	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	8.9	22.3	26.1	29.9	25.6	20.9

优化代码 3

详细注释：设置转换地点的初始值

每次变换 32 个数据，以求 cache 命中

将 32 行作为一个划分界限，每次将这 32 行的第一列在一个循环内一起转换

转换开始后 操作 32 次

每次转换完成之后将转换源向下移动一格

将接受转换的地点左移一位

转换源转换目标初始化 转换源向下移动 32 行

转化目标点和转换源相对应

在代码二的基础上将循环转换，进一步提升效率

优化结果：

Rotate: Version = naive_rotate: Naive baseline implementation:						
Dim	64	128	256	512	1024	Mean
Your CPEs	2.3	3.1	5.6	10.9	11.5	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	6.4	12.9	8.4	6.0	8.2	8.1
Rotate: Version = rotate: Current working version 0:						
Dim	64	128	256	512	1024	Mean
Your CPEs	2.2	2.3	2.9	4.6	6.2	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	6.6	17.7	16.2	14.2	15.2	13.2
Rotate: Version = rotate: Current working version 1.1:						
Dim	64	128	256	512	1024	Mean
Your CPEs	2.2	2.1	2.2	2.4	4.3	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	6.6	18.8	21.0	27.2	22.2	17.4
Rotate: Version = rotate: Current working version 2.0:						
Dim	64	128	256	512	1024	Mean
Your CPEs	2.2	2.1	2.1	2.1	4.0	
Baseline CPEs	14.7	40.1	46.4	65.9	94.5	
Speedup	6.8	19.0	22.5	30.9	23.4	18.4

优化思路：将循环次数减少 32 倍。减少关键路径的长度，有效提高程序运行速度。

实现过程：将其划分每 32 行为一个单位，每次在 32 行中只移动 1 小列的 32 个数据。

[illegible]