Perflab2 实验日志

一，rotate（旋转）

优化代码 3

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


            *dst=*src; src++;
            src-=(dim<<5)-dim;
            dst-=31+dim;
            }
        dst+=dim*dim;
        dst+=32;
        src+=(dim<<5)-dim;
        }
}
```

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。

二，smooth（平滑）

优化代码

```
char smooth_descr[] = "smooth: Current working version";
void smooth(int dim, pixel *src, pixel *dst)
{
    int i,j,ii,jj,max_1,max_2,min_1,min_2;
    pixel_sum sum;
    pixel current_pixel;
        for(i=0;i<dim;i++)
           for(j=0;j<dim;j++)
           {
                sum.red = sum.green = sum.blue = 0, sum.num = 0;
                max_1 = max(i-1, 0); max_2 = max(j-1,0);
                min_1 = min(i+1 ,dim-1);min_2 = min(j+1, dim-1);
               for(ii=max_1; ii<=min_1; ii++)
                   for(jj=max_2;jj<min_2;jj++)
                       accumulate_sum(&sum, src[RIDX(ii, jj, dim)]);
                assign_sum_to_pixel(&current_pixel, sum);
                dst[RIDX(i, j, dim)] = current_pixel;
           }
}
```

详细注释：

分别将初始化函数和 avg 函数直接实现

并将最大值最小值提前计算

优化结果：

```
Smooth: Version = naive_smooth: Naive baseline implementation:
Dim             32      64      128     256     512     Mean
Your CPEs       38.9    38.8    38.9    39.2    39.5
Baseline CPEs   695.0   698.0   702.0   717.0   722.0
Speedup         17.9    18.0    18.0    18.3    18.3    18.1
```

优化思路：

减少调用 提前计算

实现过程：

将初始化函数与 avg 函数不单独调用，直接实现在主函数里边

将 max 与 min 在 for 循环之前先计算出来，以后直接调用