

体系结构试验报告

元胞自动机优化

原有算法

构造下一代细胞矩阵的时候，对每一个细胞 $\text{newMap}[i][j]$ ，在原有 $\text{Map}[i][j]$ 中统计其邻居数目，根据邻居数目决定 $\text{newMap}[i][j]$ 等于0（死）或1（活）。

优化思路

研究发现，可以通过对 map 的矩阵变换，构造一张邻居表 $\text{neighbor}[i][j]$ ，表中每个元素代表 i 行列元素的邻居个数。

再遍历整个 map 得到 newMap 。

对应的矩阵变换为：

①对 Map_0 中的每一行，将它上一行与下一行加到本身一行（第一行的上一行定义为全0，最后一行的下一行定义为全0）得到 Map_1

②对 Map_1 的每一列，将它左一列与右一列加到本身一列（第一列的左一列定义为全0，最右一列的右一列定义为全0）

③ $\text{neighbor} = \text{Map}_1 - \text{Map}_0$ 即可得到邻居表 neighbor

针对①，我们可以用右图优化，也就是将二维矩阵看做一维长向量。

每次载入当前行的128位（middle），上一行的128位（top），下一行的128位（bottom），然后

```
//load
if(pmap<firstRow){
    middle1= _mm_load_ps(pmap);
    bottom1=_mm_load_ps(pmap+MAXCOL);
    middle1=_mm_add_ps(bottom1, middle1);
}else if(pmap>=lastRow){
    top1=_mm_load_ps(pmap-MAXCOL);
    middle1=_mm_load_ps(pmap);
    middle1=_mm_add_ps(top1, middle1);
}else{
    top1=_mm_load_ps(pmap-MAXCOL);
    middle1=_mm_load_ps(pmap);
    bottom1=_mm_load_ps(pmap+MAXCOL);
    middle1=_mm_add_ps(top1, middle1);
    middle1=_mm_add_ps(middle1, bottom1);
}
```

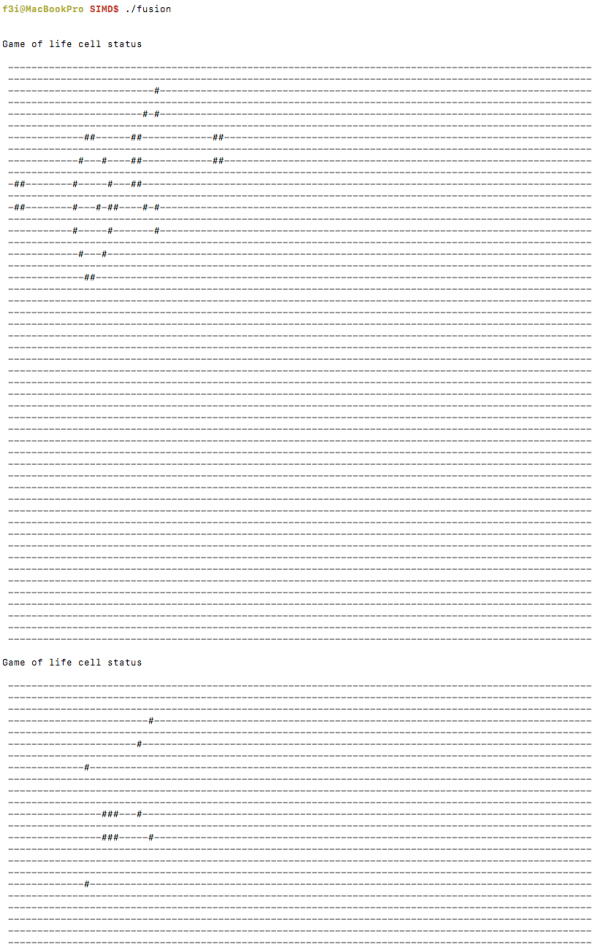
同时求和。

最后利用strip mining处理余下不可向量化数据的值。如左图。

```
//handle remain
for(int i=0; i<cntRem; i++)
{
    pSimd[i]=pmap[i]+pmap[i-MAXCOL];
}
```

对于②操作，虽然书上提到SIMD可以存在步幅，但我并未在intrinsic中找到对应的指令，故未优化。

通过右图的运行结果，我们发现优化算法能够保证与优化前完全一致的结果。



实验环境

Mac os x gcc 4.2.1

CPU 2.7 GHz Intel Core i5

支持指令SSE

编译指令 使用sublime text3 command+B默认编译指令

实验结果

为了方便展示，我编写一个counter.h用来统计cpu所耗周期，如右图。

统计了串行，SIMD下的进化所需cpu cycle数，并计算加速比。

输入为实验提供的文件input_50x100

```
int main()
{
    init();
    cout<<"SIMD enabled"<<endl;
    startTiming();
    evolutionSIMD();
    stopWithPrintTiming();

    init();
    cout<<"serilized "<<endl;
    startTiming();
    evolution();
    stopWithPrintTiming();

    return 0;
}
```

```
[f3i@MacBookPro SIMD$ ./fusion
SIMD enabled
-----Elapsed Timing(Cycles) : 704973
-----
serilized
-----Elapsed Timing(Cycles) : 1987374
-----
[f3i@MacBookPro SIMD$ ./fusion
SIMD enabled
-----Elapsed Timing(Cycles) : 699570
-----
serilized
-----Elapsed Timing(Cycles) : 1984182
-----
[f3i@MacBookPro SIMD$ ./fusion
SIMD enabled
-----Elapsed Timing(Cycles) : 705474
-----
serilized
-----Elapsed Timing(Cycles) : 2139807
-----
[f3i@MacBookPro SIMD$ ./fusion
SIMD enabled
-----Elapsed Timing(Cycles) : 691614
-----
serilized
-----Elapsed Timing(Cycles) : 1981602
-----
```

统计了进化代数为50，100，150，200，250，300下的加速比。如右图：每次运行结果中，上面一次为SIMD的耗费周期，下面一次为串行遍历的耗费周期。

最终得到右图结果，随着迭代次数增加，加速比略有上升。

我认为这是因为迭代次数越多，越多的L1, L2 cache被替换成程序执行所需的几个map，使得矩阵引用的命中率更高，得以获得更高加速比。

