高性能report2

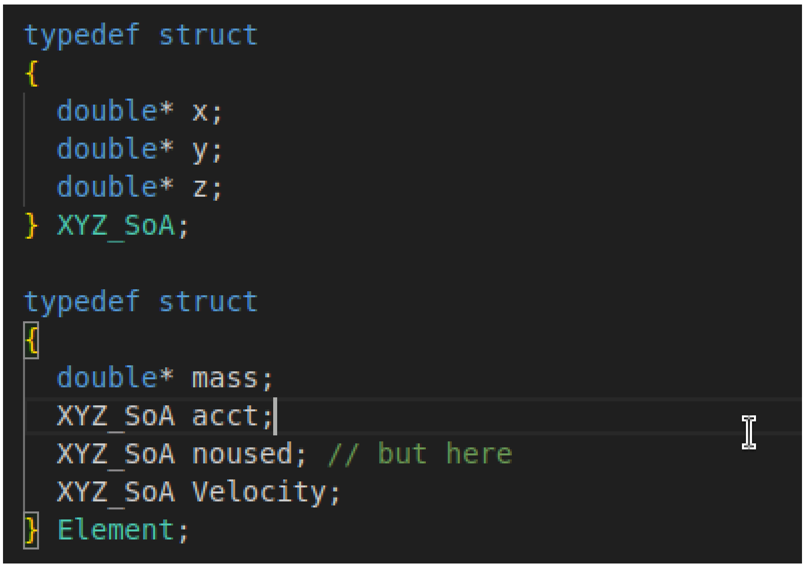
高性能计算编程实验2

1.实验环境
CPU：Dell Inc. OptiPlex Micro Plus 7020 Intel® Core™ i9-14900 × 32 64g
Os：ubuntu 22.04.05 LTS

Compiler: g++ (Ubuntu 11.4.0)

2.修改代码为SoA分配方式

主要修改原代码使用的数据结构XYZ和Element. 原代码中声明单一变量组成的结构体，然后通过结构体数组的形式存储数据。修改后，在声明时就确定结构体的成员为数组。



```
typedef struct
{
  double* x;
  double* y;
  double* z;
} XYZ_SoA;

typedef struct
{
  double* mass;
  XYZ_SoA acct;
  XYZ_SoA noused; // but here
  XYZ_SoA Velocity;
} Element;
```

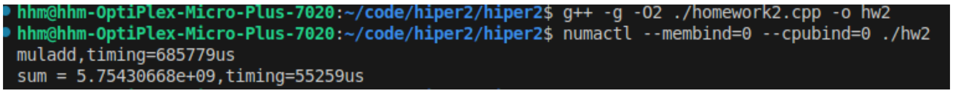
编译：

g++ -g -O2 ./homework2.cpp -o hw2

运行：

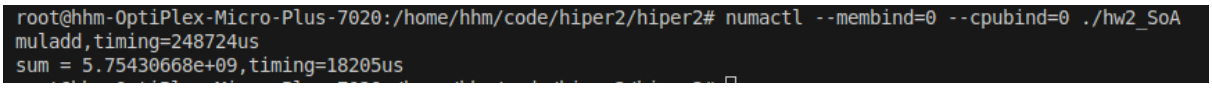
numactl --membind=0 --cpubind=0 ./hw2

运行情况：



```
hhm@hhm-OptiPlex-Micro-Plus-7020:~/code/hiper2/hiper2$ g++ -g -O2 ./homework2.cpp -o hw2
hhm@hhm-OptiPlex-Micro-Plus-7020:~/code/hiper2/hiper2$ numactl --membind=0 --cpubind=0 ./hw2
muladd,timing=685779us
sum = 5.75430668e+09,timing=55259us
```

优化后运行情况：

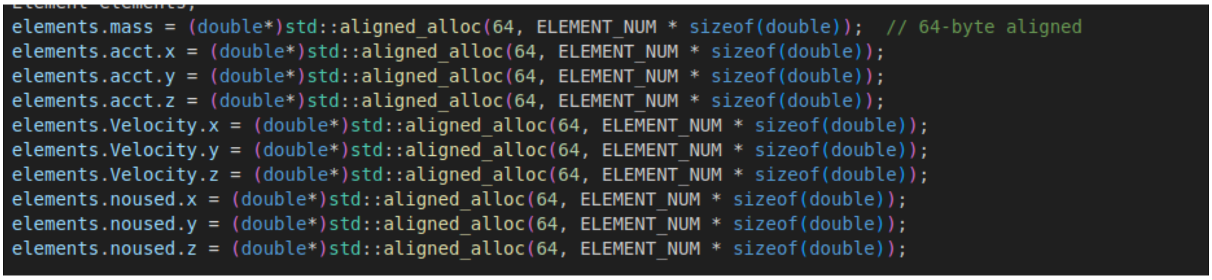


```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# numactl --membind=0 --cpubind=0 ./hw2_SoA
muladd,timing=248724us
sum = 5.75430668e+09,timing=18205us
```

3.SoA优化

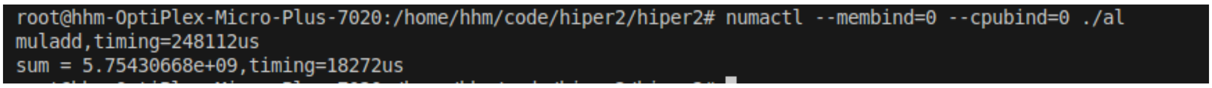
a.内存对齐

使用 `std::aligned_alloc` 来确保每个数组（如 `elements.mass`、 `elements.acct.x` 等）是按 64 字节对齐的。



```
elements.mass = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));  // 64-byte aligned
elements.acct.x = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.acct.y = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.acct.z = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.Velocity.x = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.Velocity.y = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.Velocity.z = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.noused.x = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.noused.y = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
elements.noused.z = (double*)std::aligned_alloc(64, ELEMENT_NUM * sizeof(double));
```

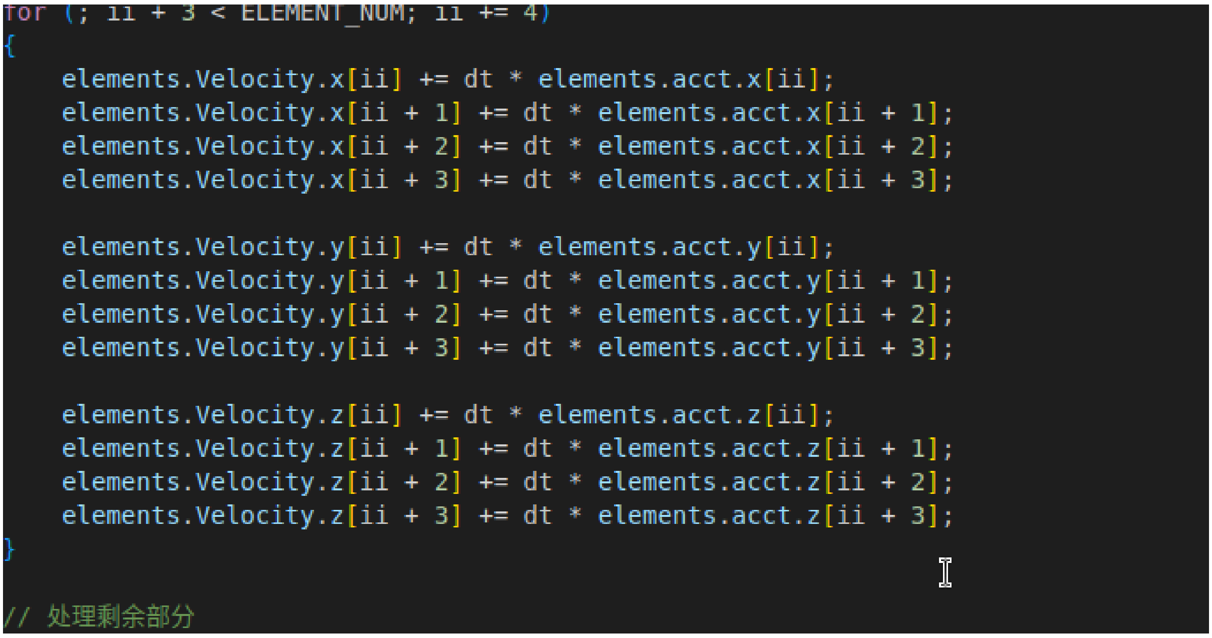
优化后运行结果：



```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# numactl --membind=0 --cpubind=0 ./al
muladd,timing=248112us
sum = 5.75430668e+09,timing=18272us
```

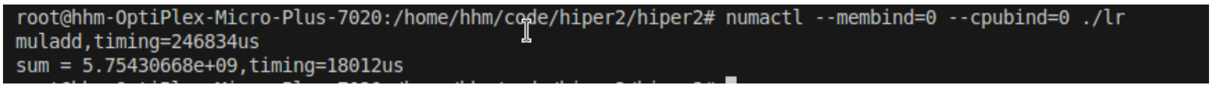
b.循环展开

通过将每次更新拆分为4次操作，尝试减少循环控制的开销



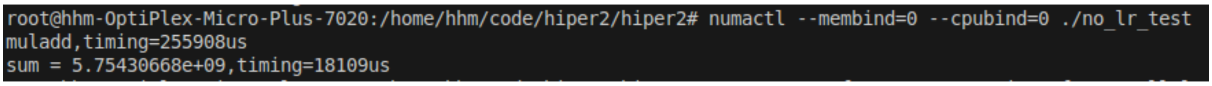
```
for (; ii + 3 < ELEMENT_NUM; ii += 4)
{
    elements.Velocity.x[ii] += dt * elements.acct.x[ii];
    elements.Velocity.x[ii + 1] += dt * elements.acct.x[ii + 1];
    elements.Velocity.x[ii + 2] += dt * elements.acct.x[ii + 2];
    elements.Velocity.x[ii + 3] += dt * elements.acct.x[ii + 3];

    elements.Velocity.y[ii] += dt * elements.acct.y[ii];
    elements.Velocity.y[ii + 1] += dt * elements.acct.y[ii + 1];
    elements.Velocity.y[ii + 2] += dt * elements.acct.y[ii + 2];
    elements.Velocity.y[ii + 3] += dt * elements.acct.y[ii + 3];

    elements.Velocity.z[ii] += dt * elements.acct.z[ii];
    elements.Velocity.z[ii + 1] += dt * elements.acct.z[ii + 1];
    elements.Velocity.z[ii + 2] += dt * elements.acct.z[ii + 2];
    elements.Velocity.z[ii + 3] += dt * elements.acct.z[ii + 3];
}

// 处理剩余部分
```

运行结果：



```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# numactl --membind=0 --cpubind=0 ./lr
muladd,timing=246834us
sum = 5.75430668e+09,timing=18012us
```

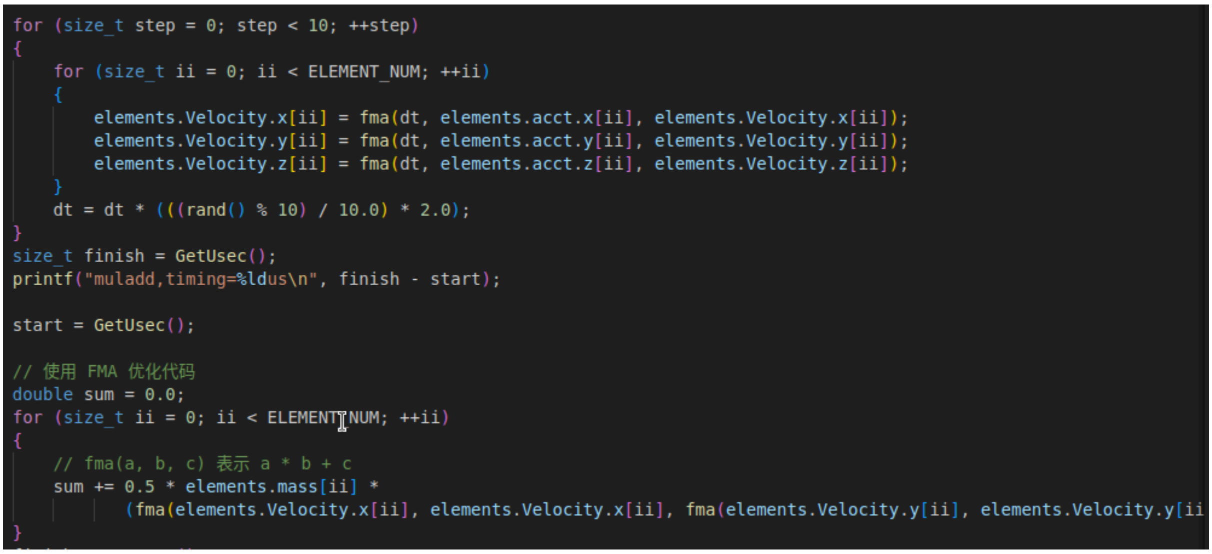
完全不使用循环展开：



```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# numactl --membind=0 --cpubind=0 ./no_lr_test
muladd,timing=255908us
sum = 5.75430668e+09,timing=18109us
```

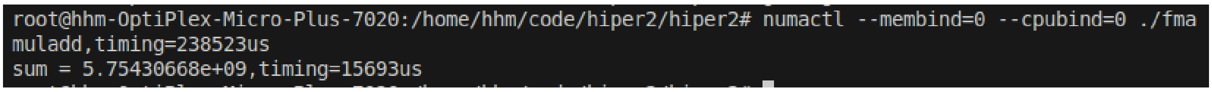
c.fma

通过fma将乘法和加法合并成一条指令



```
for (size_t step = 0; step < 10; ++step)
{
    for (size_t ii = 0; ii < ELEMENT_NUM; ++ii)
    {
        elements.Velocity.x[ii] = fma(dt, elements.acct.x[ii], elements.Velocity.x[ii]);
        elements.Velocity.y[ii] = fma(dt, elements.acct.y[ii], elements.Velocity.y[ii]);
        elements.Velocity.z[ii] = fma(dt, elements.acct.z[ii], elements.Velocity.z[ii]);
    }
    dt = dt * (((rand() % 10) / 10.0) * 2.0);
}
size_t finish = GetUsec();
printf("muladd,timing=%ldus\n", finish - start);

start = GetUsec();

// 使用 FMA 优化代码
double sum = 0.0;
for (size_t ii = 0; ii < ELEMENT_NUM; ++ii)
{
    // fma(a, b, c) 表示 a * b + c
    sum += 0.5 * elements.mass[ii] *
           (fma(elements.Velocity.x[ii], elements.Velocity.x[ii], fma(elements.Velocity.y[ii], elements.Velocity.y[ii
}
```

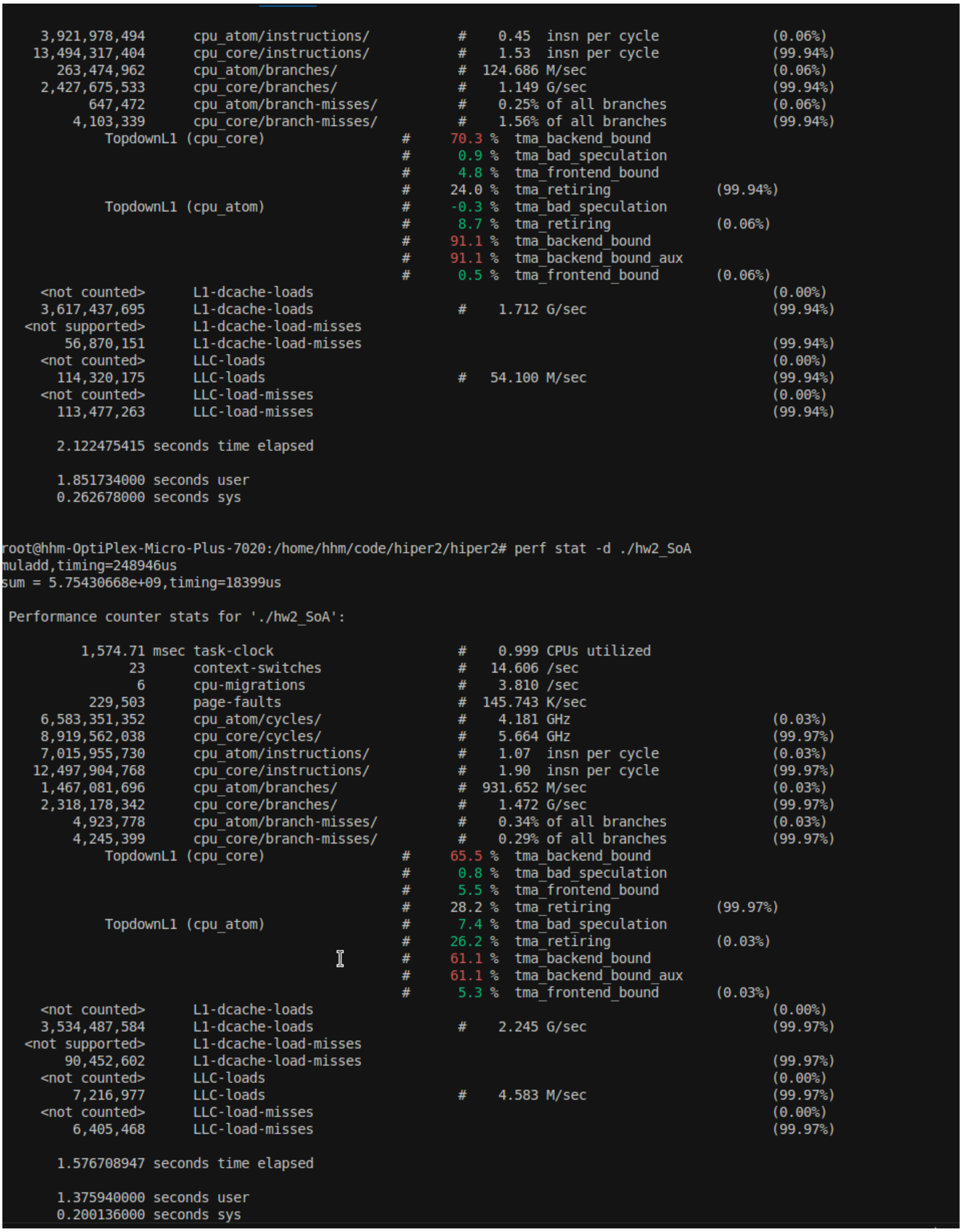
运行结果：



```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# numactl --membind=0 --cpubind=0 ./fma
muladd,timing=238523us
sum = 5.75430668e+09,timing=15693us
```

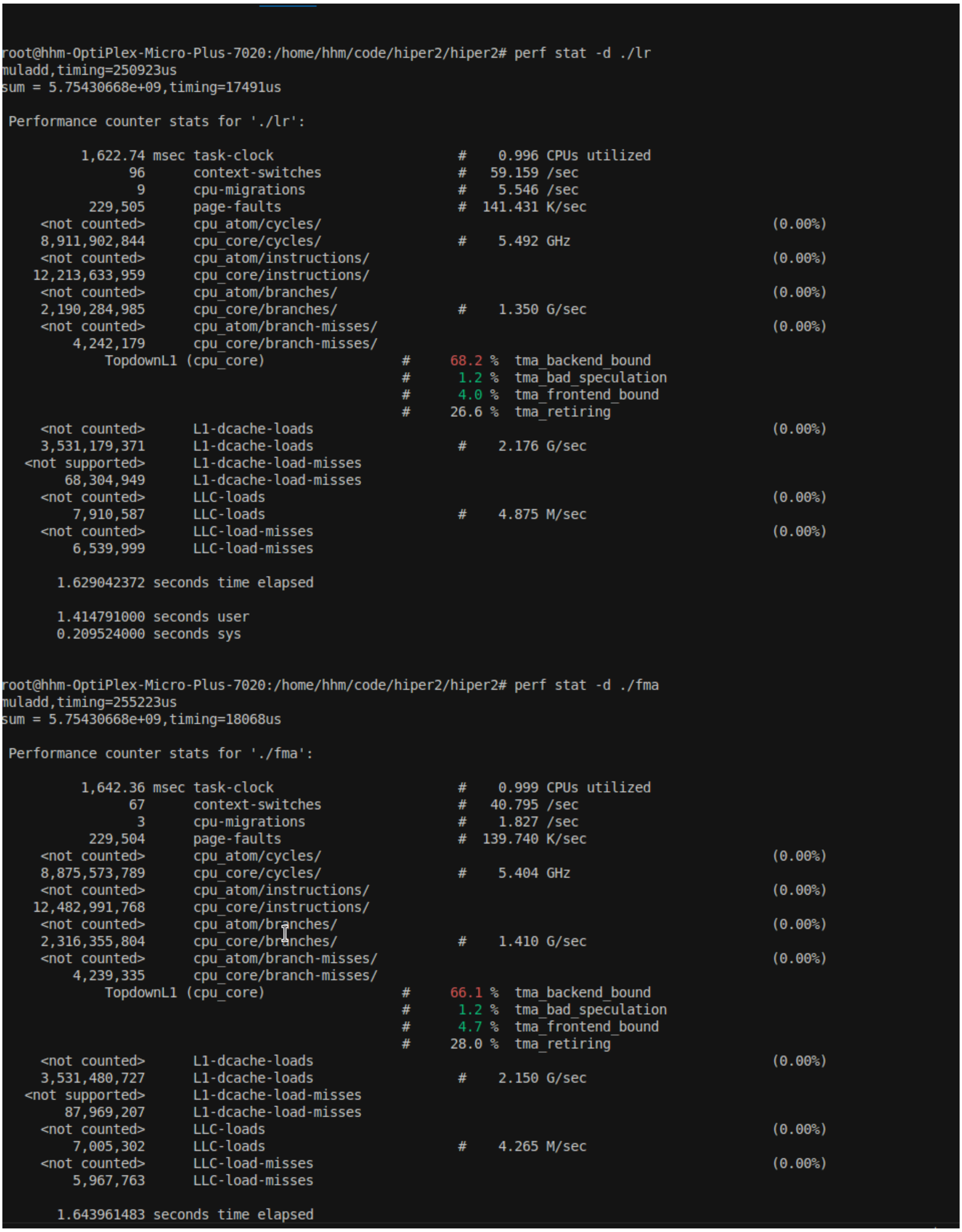
4.结论

从运行时间来看，AoS改为SoA对代码的运行时间有较大改善，这是由于SoA分配方式减少了大量的寻址工作导致的。使用perf stat对两个程序进行分析后发现SoA程序的IPC高于原程序，说明其指令流水线更加高效。此外，SoA的LLC访问减少，说明减少了高延迟的主存访问。



```
   3,921,978,494      cpu_atom/instructions/           #    0.45  insn per cycle              (0.06%)
  13,494,317,404      cpu_core/instructions/           #    1.53  insn per cycle              (99.94%)
     263,474,962      cpu_atom/branches/               #  124.686 M/sec                       (0.06%)
   2,427,675,533      cpu_core/branches/               #    1.149 G/sec                       (99.94%)
         647,472      cpu_atom/branch-misses/          #    0.25% of all branches             (0.06%)
       4,103,339      cpu_core/branch-misses/          #    1.56% of all branches             (99.94%)
             TopdownL1 (cpu_core)                #     70.3 %  tma_backend_bound
                                                 #      0.9 %  tma_bad_speculation
                                                 #      4.8 %  tma_frontend_bound
                                                 #     24.0 %  tma_retiring             (99.94%)
             TopdownL1 (cpu_atom)                #     -0.3 %  tma_bad_speculation
                                                 #      8.7 %  tma_retiring             (0.06%)
                                                 #     91.1 %  tma_backend_bound
                                                 #     91.1 %  tma_backend_bound_aux
                                                 #      0.5 %  tma_frontend_bound       (0.06%)
   <not counted>      L1-dcache-loads                                                     (0.00%)
   3,617,437,695      L1-dcache-loads                  #    1.712 G/sec                       (99.94%)
 <not supported>      L1-dcache-load-misses
      56,870,151      L1-dcache-load-misses                                               (99.94%)
   <not counted>      LLC-loads                                                           (0.00%)
     114,320,175      LLC-loads                        #   54.100 M/sec                       (99.94%)
   <not counted>      LLC-load-misses                                                     (0.00%)
     113,477,263      LLC-load-misses                                                     (99.94%)

     2.122475415 seconds time elapsed

     1.851734000 seconds user
     0.262678000 seconds sys


root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# perf stat -d ./hw2_SoA
muladd,timing=248946us
sum = 5.75430668e+09,timing=18399us

 Performance counter stats for './hw2_SoA':

         1,574.71 msec task-clock                       #    0.999 CPUs utilized
               23      context-switches                 #   14.606 /sec
                6      cpu-migrations                   #    3.810 /sec
          229,503      page-faults                      #  145.743 K/sec
   6,583,351,352      cpu_atom/cycles/                 #    4.181 GHz                         (0.03%)
   8,919,562,038      cpu_core/cycles/                 #    5.664 GHz                         (99.97%)
   7,015,955,730      cpu_atom/instructions/           #    1.07  insn per cycle              (0.03%)
  12,497,904,768      cpu_core/instructions/           #    1.90  insn per cycle              (99.97%)
   1,467,081,696      cpu_atom/branches/               #  931.652 M/sec                       (0.03%)
   2,318,178,342      cpu_core/branches/               #    1.472 G/sec                       (99.97%)
       4,923,778      cpu_atom/branch-misses/          #    0.34% of all branches             (0.03%)
       4,245,399      cpu_core/branch-misses/          #    0.29% of all branches             (99.97%)
             TopdownL1 (cpu_core)                #     65.5 %  tma_backend_bound
                                                 #      0.8 %  tma_bad_speculation
                                                 #      5.5 %  tma_frontend_bound
                                                 #     28.2 %  tma_retiring             (99.97%)
             TopdownL1 (cpu_atom)                #      7.4 %  tma_bad_speculation
                                                 #     26.2 %  tma_retiring             (0.03%)
                                                 #     61.1 %  tma_backend_bound
                                                 #     61.1 %  tma_backend_bound_aux
                                                 #      5.3 %  tma_frontend_bound       (0.03%)
   <not counted>      L1-dcache-loads                                                     (0.00%)
   3,534,487,584      L1-dcache-loads                  #    2.245 G/sec                       (99.97%)
 <not supported>      L1-dcache-load-misses
      90,452,602      L1-dcache-load-misses                                               (99.97%)
   <not counted>      LLC-loads                                                           (0.00%)
       7,216,977      LLC-loads                        #    4.583 M/sec                       (99.97%)
   <not counted>      LLC-load-misses                                                     (0.00%)
       6,405,468      LLC-load-misses                                                     (99.97%)

     1.576708947 seconds time elapsed

     1.375940000 seconds user
     0.200136000 seconds sys
```

然而，后面采用的几种优化方法似乎都没有达到预期的效果。通过添加选项禁止向量化和循环展开，确认并不是-O2编译级别中包含向量化和循环展开，而是以上三种发放对于原代码都不是特别有效。其中一种可能是，手动的循环展开只有4次，因此观察不到太大的效果。另外，SoA分配方式所申请的内存是连续的内存块，已经保证了内存空间的连续分布，因此可能不需要内存对齐。



```
root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# perf stat -d ./lr
muladd,timing=250923us
sum = 5.75430668e+09,timing=17491us

 Performance counter stats for './lr':

          1,622.74 msec task-clock                       #    0.996 CPUs utilized
                96      context-switches                 #   59.159 /sec
                 9      cpu-migrations                   #    5.546 /sec
           229,505      page-faults                      #  141.431 K/sec
     <not counted>      cpu_atom/cycles/                                                        (0.00%)
     8,911,902,844      cpu_core/cycles/                 #    5.492 GHz
     <not counted>      cpu_atom/instructions/                                                  (0.00%)
    12,213,633,959      cpu_core/instructions/
     <not counted>      cpu_atom/branches/                                                      (0.00%)
     2,190,284,985      cpu_core/branches/               #    1.350 G/sec
     <not counted>      cpu_atom/branch-misses/                                                 (0.00%)
         4,242,179      cpu_core/branch-misses/
             TopdownL1 (cpu_core)                 #     68.2 %  tma_backend_bound
                                                  #      1.2 %  tma_bad_speculation
                                                  #      4.0 %  tma_frontend_bound
                                                  #     26.6 %  tma_retiring
     <not counted>      L1-dcache-loads                                                         (0.00%)
     3,531,179,371      L1-dcache-loads                  #    2.176 G/sec
   <not supported>      L1-dcache-load-misses
        68,304,949      L1-dcache-load-misses
     <not counted>      LLC-loads                                                               (0.00%)
         7,910,587      LLC-loads                        #    4.875 M/sec
     <not counted>      LLC-load-misses                                                         (0.00%)
         6,539,999      LLC-load-misses

       1.629042372 seconds time elapsed

       1.414791000 seconds user
       0.209524000 seconds sys

root@hhm-OptiPlex-Micro-Plus-7020:/home/hhm/code/hiper2/hiper2# perf stat -d ./fma
muladd,timing=255223us
sum = 5.75430668e+09,timing=18068us

 Performance counter stats for './fma':

          1,642.36 msec task-clock                       #    0.999 CPUs utilized
                67      context-switches                 #   40.795 /sec
                 3      cpu-migrations                   #    1.827 /sec
           229,504      page-faults                      #  139.740 K/sec
     <not counted>      cpu_atom/cycles/                                                        (0.00%)
     8,875,573,789      cpu_core/cycles/                 #    5.404 GHz
     <not counted>      cpu_atom/instructions/                                                  (0.00%)
    12,482,991,768      cpu_core/instructions/
     <not counted>      cpu_atom/branches/                                                      (0.00%)
     2,316,355,804      cpu_core/branches/               #    1.410 G/sec
     <not counted>      cpu_atom/branch-misses/                                                 (0.00%)
         4,239,335      cpu_core/branch-misses/
             TopdownL1 (cpu_core)                 #     66.1 %  tma_backend_bound
                                                  #      1.2 %  tma_bad_speculation
                                                  #      4.7 %  tma_frontend_bound
                                                  #     28.0 %  tma_retiring
     <not counted>      L1-dcache-loads                                                         (0.00%)
     3,531,480,727      L1-dcache-loads                  #    2.150 G/sec
   <not supported>      L1-dcache-load-misses
        87,969,207      L1-dcache-load-misses
     <not counted>      LLC-loads                                                               (0.00%)
         7,005,302      LLC-loads                        #    4.265 M/sec
     <not counted>      LLC-load-misses                                                         (0.00%)
         5,967,763      LLC-load-misses

       1.643961483 seconds time elapsed
```