Задание 2
Отчёт
по CUDA ADI3d

Ши Хуэй shihuicollapsor@gmail.com

1. Постановка задачи

1. Программа должна автоматически определять доступный объём памяти на GPU и выбирать максимально возможный размер сетки (L), который поместится в эту память.

2. Нужно реализовать возможность запуска на CPU и GPU, а также режим сравнения, чтобы проверить, что результаты расчётов одинаковы.

3. Редукцию (вычисление максимального значения ошибки eps) необходимо распараллелить. В программе уже используется атомарная операция на GPU, но она может быть оптимизирована.

4. Создайте Git-репозиторий с вашим кодом. В нём должен быть:

•Makefile, который позволит собрать и запустить программу на любом сервере с GPU.

•Возможность выбирать, на каком устройстве (CPU или GPU) будет выполняться программа.

•Режим проверки совпадения результатов между CPU и GPU.

5. Проверьте производительность программы:

•Сравните время выполнения программы на GPU и CPU.

•Постройте таблицу или график, показывающий ускорение программы на GPU по сравнению с последовательной версией на CPU (с максимальными опциями оптимизации).

2. Формат командной строки

nvcc adi3d_cuda.cu -o cuda2

3. Спецификация системы

- Operating system : Linux 6.8.0-45-generic
- Vendor string and code : GenuineIntel (1, 0x1)
- Model string and code : Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz (165, 0xa5)
- CPU revision : 2.000000
- CPUID : Family/Model/Stepping 6/165/2, 0x06/0xa5/0x02
- CPU Max MHz : 5000
- CPU Min MHz : 800
- Total cores : 12

- SMT threads per core : 2
- Cores per socket : 6
- Sockets : 1
- Cores per NUMA region : 12
- NUMA regions : 1
- Running in a VM : no
- Number Hardware Counters : 10
- Max Multiplex Counters : 384

4. Описание алгоритмов

Для этой программы был написан файл Makefile.

Введите следующую команду в терминале:

nvcc adi3d_cuda.cu -o cuda

После этого мы войдем:

make run_cpu

Программа будет запущена на CPU, и результаты показаны справа в таблице ниже.

make run_gpu

Программа будет запущена на GPU, и результаты показаны слева в таблице ниже.

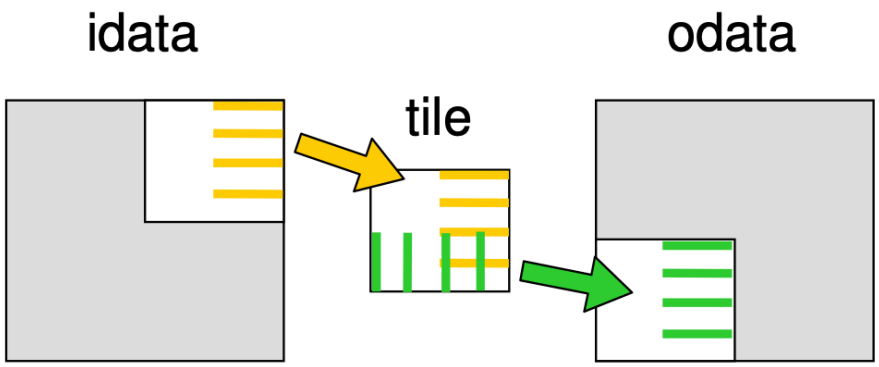
```
//(i,j,k)->(k,j,i)
__global__ void transposeMatrix(const double* in, double* out, int L)
{
    int j = blockIdx.y;
    __shared__ double storey[DIMS + 1][ DIMS + 1];

    int i = blockIdx.x * DIMS + threadIdx.y;
    int k = blockIdx.z * DIMS + threadIdx.x;

    if (k >= 0 && k < L) {
        for (int dim = 0; dim < DIMS; dim += ROWS)
        {
            int row = dim + i;
            if (row >= L)
                break;
            storey[threadIdx.y + dim][threadIdx.x] = in[INDEX(i + dim,j,k)];
        }
    }
    __syncthreads();

    int kk = blockIdx.x * DIMS + threadIdx.x;
    int ii = blockIdx.z * DIMS + threadIdx.y;
    if (kk >= 0 && kk < L) {
        for (int dim = 0; dim < DIMS; dim += ROWS)
        {
            int row = dim + ii;
            if (row >= L)
                break;
            out[INDEX(ii + dim, j, kk)] = storey[threadIdx.x][threadIdx.y + dim];
        }
    }
    //__syncthreads();
}
```

4.1 Транспонирование матрицы

Справочные ссылки：https://dmacssite.github.io/materials/MatrixTranspose.pdf ,

【在CUDA中优化矩阵转置 - CSDN App】https://blog.csdn.net/qq_62704693/article/details/141358466?sharetype=blogdetail&shareId=141358466&sharerefer=APP&sharesource=qq_51316405&sharefrom=link

В CUDA глобальный доступ к памяти организован с помощью warp (32 потока).Разделив матрицу на небольшие фрагменты (тайлы) и убедившись, что каждый поток соответствует элементу в тайле, можно добиться объединенного доступа к глобальной памяти.В моем коде DIMS определяет длину стороны каждой плитки, а ROWS определяет, сколько строк данных загружает каждый поток за раз.Таким образом, матрица разделена на несколько фрагментов размером DIMS*DIMS, каждый из которых представляет собой диапазон, за обработку которого отвечает блок потоков CUDA.



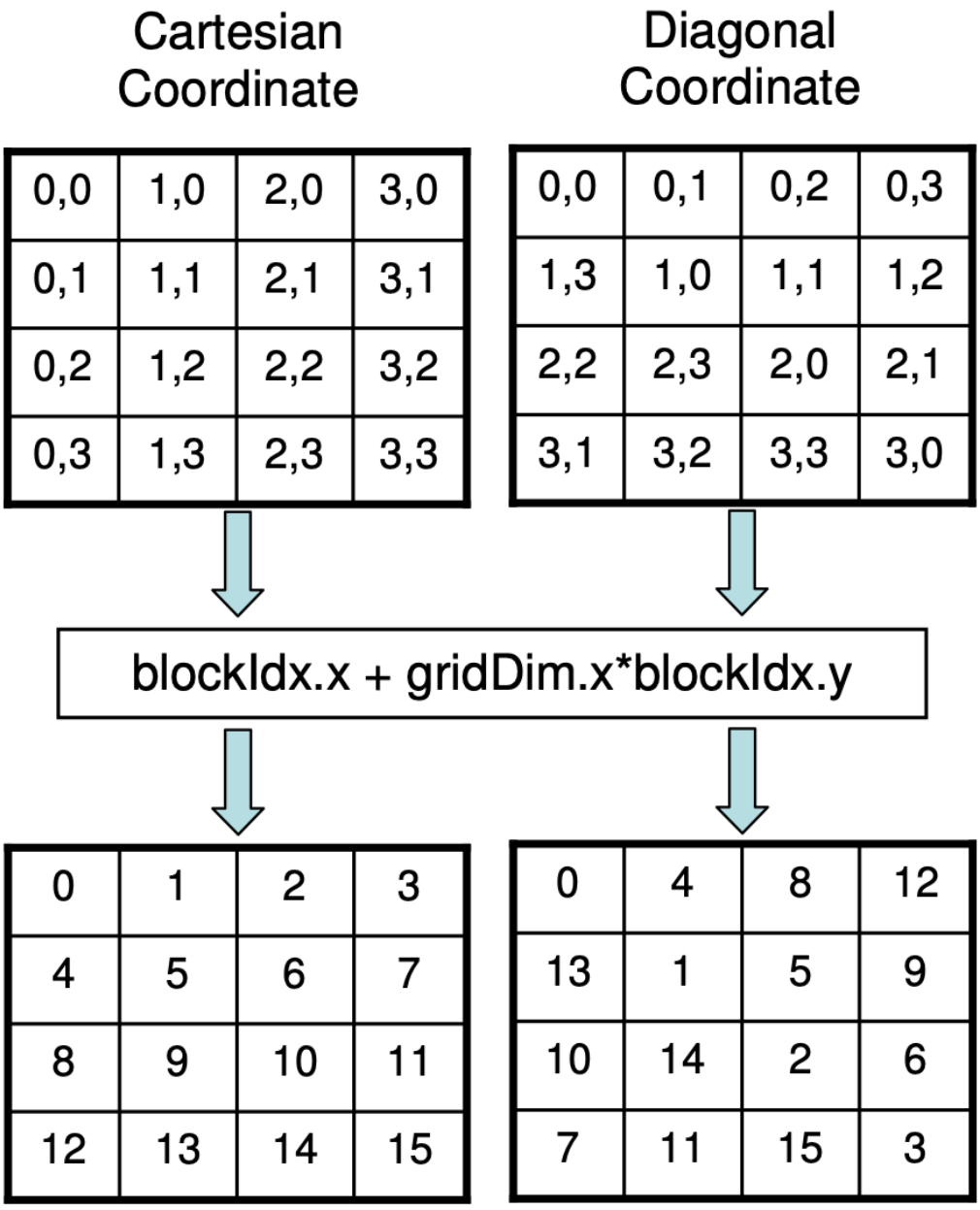
После этого каждый поток завершает обмен строками матричных блоков в общей памяти. Здесь я использовал заполнение (DIMS + 1), чтобы разрешить конфликт банка общей памяти с общей памятью.Транспонированная матрица записывается обратно в глобальную память из общей памяти.

4.1.1Переставляйте блоки по диагонали

Выполните перестановку блоков по диагонали с помощью следующего кода：

int i = blockIdx.x * DIMS + threadIdx.y;

int k = blockIdx.z * DIMS + threadIdx.x;



4.1.2 Зачем переносить

Если обновление выполняется непосредственно в направлении оси k, поскольку блок потоков (warp) в CUDA обрабатывает данные вместе каждые 32 потока, если поток обращается к данным по прерывистой оси k (то есть, доступ по строкам), эти адреса могут быть распределены в разных местах в памяти, что приводит к несогласованному доступу к памяти (non-coalesced memory access).Такой вид несогласованного доступа приведет к многочисленным транзакциям с памятью (transactions), что значительно снизит производительность.Однако данные на каждой оси i хранятся непрерывно, и потоки могут обращаться к адресам непрерывных данных по порядку.После настройки это может гарантировать, что каждый warp обращается к выровненному адресу памяти, избегая проблемы несогласованности.

4.2 Найдите глобальное максимальное значение

Затем найдите максимальную погрешность (eps) между двумя матрицами a_old и a_new.Каждый поток вычисляет абсолютную разницу diff соответствующего ему элемента position и сохраняет результат в sdata общей памяти.Впоследствии в блоке используется взаимодействие потоков для определения максимального значения ошибки текущего блока путем двустороннего сокращения.После завершения сокращения поток с идентификатором потока, равным 0, использует атомарную операцию atomicMaxDouble для сравнения и обновления максимальной ошибки текущего блока с глобальной переменной eps_d, чтобы вычислить глобальную максимальную ошибку одновременно и безопасно в нескольких блоках.

```
__global__ void kernel_compute_eps(const double *a_old, const double *a_new, double *eps_d, int L)
{
    extern __shared__ double sdata[];

    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    int k = blockIdx.z * blockDim.z + threadIdx.z;

    double diff = 0.0;
    if (i < L && j < L && k < L)
    {

        diff = fabs(a_new[INDEX(i,j,k)] - a_old[INDEX(i,j,k)]);
        //printf("diff_a[%d,%d,%d]=%.6lf \n",i,j,k,diff);
    }


    int tid = threadIdx.x + threadIdx.y * blockDim.x + threadIdx.z * blockDim.x * blockDim.y;
    sdata[tid] = diff;
    __syncthreads();

    for (unsigned int s = blockDim.x * blockDim.y * blockDim.z / 2; s > 0; s >>= 1)
    {
        if (tid < s)
        {
            sdata[tid] = Max(sdata[tid], sdata[tid + s]);
        }
        __syncthreads();
    }

    if (tid == 0)
    {
        atomicMaxDouble(eps_d, sdata[0]);
    }
}
```

4.3 модель ГПУ

4.3.1 Модель выполнения на графическом процессоре

- Нити и блоки нитей（Threads and Thread Blocks）

dim3 threads(BLOCKSIZE, BLOCKSIZE, 1);

dim3 blocks((L + BLOCKSIZE - 1) / BLOCKSIZE, (L + BLOCKSIZE - 1) / BLOCKSIZE, 1);

- Warp

4.3.2 Архитектура графической памяти

- Выравнивание памяти（Coalesced Memory Access）
- Общая память（Shared Memory）

5.Заключение

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ nvcc adi3d_cuda.cu -o cuda

collapsor@collapsor-G5-5500:~/Desktop/CUDA$ ./cuda

Free memory: 1106247680 bytes

Total memory: 6020661248 bytes

Dynamic grid size set to: 332 x 332 x 332

Running on GPU...	Running on CPU...
GPU IT = 1 EPS = 1.4939577E+01	CPU IT = 1 EPS = 1.4939577E+01
GPU IT = 2 EPS = 7.4546828E+00	CPU IT = 2 EPS = 7.4546828E+00
GPU IT = 3 EPS = 3.7197885E+00	CPU IT = 3 EPS = 3.7197885E+00
GPU IT = 4 EPS = 2.7841767E+00	CPU IT = 4 EPS = 2.7841767E+00
GPU IT = 5 EPS = 2.0838841E+00	CPU IT = 5 EPS = 2.0838841E+00
GPU IT = 6 EPS = 1.6174943E+00	CPU IT = 6 EPS = 1.6174943E+00
GPU IT = 7 EPS = 1.3835914E+00	CPU IT = 7 EPS = 1.3835914E+00
GPU IT = 8 EPS = 1.1865898E+00	CPU IT = 8 EPS = 1.1865898E+00
GPU IT = 9 EPS = 1.0262684E+00	CPU IT = 9 EPS = 1.0262684E+00
GPU IT = 10 EPS = 8.9621378E-01	CPU IT = 10 EPS = 8.9621378E-01
GPU IT = 11 EPS = 8.1386743E-01	CPU IT = 11 EPS = 8.1386743E-01
GPU IT = 12 EPS = 7.4003912E-01	CPU IT = 12 EPS = 7.4003912E-01
GPU IT = 13 EPS = 6.7499491E-01	CPU IT = 13 EPS = 6.7499491E-01
GPU IT = 14 EPS = 6.1804058E-01	CPU IT = 14 EPS = 6.1804058E-01
GPU IT = 15 EPS = 5.6770197E-01	CPU IT = 15 EPS = 5.6770197E-01
GPU IT = 16 EPS = 5.3173036E-01	CPU IT = 16 EPS = 5.3173036E-01
GPU IT = 17 EPS = 4.9832553E-01	CPU IT = 17 EPS = 4.9832553E-01
GPU IT = 18 EPS = 4.6790273E-01	CPU IT = 18 EPS = 4.6790273E-01
GPU IT = 19 EPS = 4.3984770E-01	CPU IT = 19 EPS = 4.3984770E-01
GPU IT = 20 EPS = 4.1435740E-01	CPU IT = 20 EPS = 4.1435740E-01

Running on GPU...	Running on CPU...
GPU IT = 21 EPS = 3.9085728E-01	CPU IT = 21 EPS = 3.9085728E-01
GPU IT = 22 EPS = 3.7277002E-01	CPU IT = 22 EPS = 3.7277002E-01
GPU IT = 23 EPS = 3.5568000E-01	CPU IT = 23 EPS = 3.5568000E-01
GPU IT = 24 EPS = 3.3966110E-01	CPU IT = 24 EPS = 3.3966110E-01
GPU IT = 25 EPS = 3.2465039E-01	CPU IT = 25 EPS = 3.2465039E-01
GPU IT = 26 EPS = 3.1051412E-01	CPU IT = 26 EPS = 3.1051412E-01
GPU IT = 27 EPS = 2.9735018E-01	CPU IT = 27 EPS = 2.9735018E-01
GPU IT = 28 EPS = 2.8494276E-01	CPU IT = 28 EPS = 2.8494276E-01
GPU IT = 29 EPS = 2.7487311E-01	CPU IT = 29 EPS = 2.7487311E-01
GPU IT = 30 EPS = 2.6529327E-01	CPU IT = 30 EPS = 2.6529327E-01
GPU IT = 31 EPS = 2.5612042E-01	CPU IT = 31 EPS = 2.5612042E-01
GPU IT = 32 EPS = 2.4742678E-01	CPU IT = 32 EPS = 2.4742678E-01
GPU IT = 33 EPS = 2.3914235E-01	CPU IT = 33 EPS = 2.3914235E-01
GPU IT = 34 EPS = 2.3122085E-01	CPU IT = 34 EPS = 2.3122085E-01
GPU IT = 35 EPS = 2.2372279E-01	CPU IT = 35 EPS = 2.2372279E-01
GPU IT = 36 EPS = 2.1656870E-01	CPU IT = 36 EPS = 2.1656870E-01
GPU IT = 37 EPS = 2.1053367E-01	CPU IT = 37 EPS = 2.1053367E-01
GPU IT = 38 EPS = 2.0475639E-01	CPU IT = 38 EPS = 2.0475639E-01
GPU IT = 39 EPS = 1.9919003E-01	CPU IT = 39 EPS = 1.9919003E-01
GPU IT = 40 EPS = 1.9381369E-01	CPU IT = 40 EPS = 1.9381369E-01
GPU IT = 41 EPS = 1.8865693E-01	CPU IT = 41 EPS = 1.8865693E-01
GPU IT = 42 EPS = 1.8370365E-01	CPU IT = 42 EPS = 1.8370365E-01
GPU IT = 43 EPS = 1.7892230E-01	CPU IT = 43 EPS = 1.7892230E-01
GPU IT = 44 EPS = 1.7431737E-01	CPU IT = 44 EPS = 1.7431737E-01
GPU IT = 45 EPS = 1.6990997E-01	CPU IT = 45 EPS = 1.6990997E-01
GPU IT = 46 EPS = 1.6610415E-01	CPU IT = 46 EPS = 1.6610415E-01
GPU IT = 47 EPS = 1.6240145E-01	CPU IT = 47 EPS = 1.6240145E-01
GPU IT = 48 EPS = 1.5882880E-01	CPU IT = 48 EPS = 1.5882880E-01
GPU IT = 49 EPS = 1.5536742E-01	CPU IT = 49 EPS = 1.5536742E-01
GPU IT = 50 EPS = 1.5200257E-01	CPU IT = 50 EPS = 1.5200257E-01

Running on GPU...	Running on CPU...
GPU IT = 51 EPS = 1.4873235E-01	CPU IT = 51 EPS = 1.4873235E-01
GPU IT = 52 EPS = 1.4558697E-01	CPU IT = 52 EPS = 1.4558697E-01
GPU IT = 53 EPS = 1.4253020E-01	CPU IT = 53 EPS = 1.4253020E-01
GPU IT = 54 EPS = 1.3955924E-01	CPU IT = 54 EPS = 1.3955924E-01
GPU IT = 55 EPS = 1.3667643E-01	CPU IT = 55 EPS = 1.3667643E-01
GPU IT = 56 EPS = 1.3416280E-01	CPU IT = 56 EPS = 1.3416280E-01
GPU IT = 57 EPS = 1.3170609E-01	CPU IT = 57 EPS = 1.3170609E-01
GPU IT = 58 EPS = 1.2930527E-01	CPU IT = 58 EPS = 1.2930527E-01
GPU IT = 59 EPS = 1.2696333E-01	CPU IT = 59 EPS = 1.2696333E-01
GPU IT = 60 EPS = 1.2469535E-01	CPU IT = 60 EPS = 1.2469535E-01
GPU IT = 61 EPS = 1.2247988E-01	CPU IT = 61 EPS = 1.2247988E-01
GPU IT = 62 EPS = 1.2031570E-01	CPU IT = 62 EPS = 1.2031570E-01
GPU IT = 63 EPS = 1.1820155E-01	CPU IT = 63 EPS = 1.1820155E-01
GPU IT = 64 EPS = 1.1615662E-01	CPU IT = 64 EPS = 1.1615662E-01
GPU IT = 65 EPS = 1.1415943E-01	CPU IT = 65 EPS = 1.1415943E-01
GPU IT = 66 EPS = 1.1220850E-01	CPU IT = 66 EPS = 1.1220850E-01
GPU IT = 67 EPS = 1.1046697E-01	CPU IT = 67 EPS = 1.1046697E-01
GPU IT = 68 EPS = 1.0876979E-01	CPU IT = 68 EPS = 1.0876979E-01
GPU IT = 69 EPS = 1.0711144E-01	CPU IT = 69 EPS = 1.0711144E-01
GPU IT = 70 EPS = 1.0548507E-01	CPU IT = 70 EPS = 1.0548507E-01
GPU IT = 71 EPS = 1.0389012E-01	CPU IT = 71 EPS = 1.0389012E-01
GPU IT = 72 EPS = 1.0232786E-01	CPU IT = 72 EPS = 1.0232786E-01
GPU IT = 73 EPS = 1.0080908E-01	CPU IT = 73 EPS = 1.0080908E-01
GPU IT = 74 EPS = 9.9319922E-02	CPU IT = 74 EPS = 9.9319922E-02
GPU IT = 75 EPS = 9.7859759E-02	CPU IT = 75 EPS = 9.7859759E-02
GPU IT = 76 EPS = 9.6427978E-02	CPU IT = 76 EPS = 9.6427978E-02
GPU IT = 77 EPS = 9.5029531E-02	CPU IT = 77 EPS = 9.5029531E-02
GPU IT = 78 EPS = 9.3665652E-02	CPU IT = 78 EPS = 9.3665652E-02
GPU IT = 79 EPS = 9.2434413E-02	CPU IT = 79 EPS = 9.2434413E-02
GPU IT = 80 EPS = 9.1223224E-02	CPU IT = 80 EPS = 9.1223224E-02

Running on GPU...	Running on CPU...
GPU IT = 81 EPS = 9.0031809E-02	CPU IT = 81 EPS = 9.0031809E-02
GPU IT = 82 EPS = 8.8866148E-02	CPU IT = 82 EPS = 8.8866148E-02
GPU IT = 83 EPS = 8.7724653E-02	CPU IT = 83 EPS = 8.7724653E-02
GPU IT = 84 EPS = 8.6602028E-02	CPU IT = 84 EPS = 8.6602028E-02
GPU IT = 85 EPS = 8.5497964E-02	CPU IT = 85 EPS = 8.5497964E-02
GPU IT = 86 EPS = 8.4412147E-02	CPU IT = 86 EPS = 8.4412147E-02
GPU IT = 87 EPS = 8.3349209E-02	CPU IT = 87 EPS = 8.3349209E-02
GPU IT = 88 EPS = 8.2308684E-02	CPU IT = 88 EPS = 8.2308684E-02
GPU IT = 89 EPS = 8.1285425E-02	CPU IT = 89 EPS = 8.1285425E-02
GPU IT = 90 EPS = 8.0279113E-02	CPU IT = 90 EPS = 8.0279113E-02
GPU IT = 91 EPS = 7.9289431E-02	CPU IT = 91 EPS = 7.9289431E-02
GPU IT = 92 EPS = 7.8389276E-02	CPU IT = 92 EPS = 7.8389276E-02
GPU IT = 93 EPS = 7.7507920E-02	CPU IT = 93 EPS = 7.7507920E-02
GPU IT = 94 EPS = 7.6639034E-02	CPU IT = 94 EPS = 7.6639034E-02
GPU IT = 95 EPS = 7.5782459E-02	CPU IT = 95 EPS = 7.5782459E-02
GPU IT = 96 EPS = 7.4938032E-02	CPU IT = 96 EPS = 7.4938032E-02
GPU IT = 97 EPS = 7.4105590E-02	CPU IT = 97 EPS = 7.4105590E-02
GPU IT = 98 EPS = 7.3291357E-02	CPU IT = 98 EPS = 7.3291357E-02
GPU IT = 99 EPS = 7.2489794E-02	CPU IT = 99 EPS = 7.2489794E-02
GPU IT = 100 EPS = 7.1699681E-02	CPU IT = 100 EPS = 7.1699681E-02
ADI Benchmark Completed.	ADI Benchmark Completed.
Size = 332 x 332 x 332	Size = 332 x 332 x 332
Iterations = 100	Iterations = 100
Time in seconds = 3.029143	Time in seconds = 66.321560
Operation type = double precision	Operation type = double precision
END OF ADI Benchmark	END OF ADI Benchmark

6.Запуск с помощью openmpi

Сбросьте размер сетки (размер=332). Выполнить команду:

```
gcc -O3 -fopenmp adi3d.c -o adi
./adi
```

IT = 1 EPS = 1.4939577E+01
IT = 2 EPS = 7.4546828E+00
IT = 3 EPS = 3.7197885E+00
IT = 4 EPS = 2.7841767E+00
IT = 5 EPS = 2.0838841E+00
IT = 6 EPS = 1.6174943E+00
IT = 7 EPS = 1.3835914E+00
IT = 8 EPS = 1.1865898E+00
IT = 9 EPS = 1.0262684E+00
IT = 10 EPS = 8.9621378E-01
IT = 11 EPS = 8.1386743E-01
IT = 12 EPS = 7.4003912E-01
IT = 13 EPS = 6.7499491E-01
IT = 14 EPS = 6.1804058E-01
IT = 15 EPS = 5.6770197E-01
IT = 16 EPS = 5.3173036E-01
IT = 17 EPS = 4.9832553E-01
IT = 18 EPS = 4.6790273E-01
IT = 19 EPS = 4.3984770E-01
IT = 20 EPS = 4.1435740E-01
IT = 21 EPS = 3.9085728E-01
IT = 22 EPS = 3.7277002E-01
IT = 23 EPS = 3.5568000E-01
IT = 24 EPS = 3.3966110E-01
IT = 25 EPS = 3.2465039E-01
IT = 26 EPS = 3.1051412E-01
IT = 27 EPS = 2.9735018E-01
IT = 28 EPS = 2.8494276E-01
IT = 29 EPS = 2.7487311E-01
IT = 30 EPS = 2.6529327E-01
IT = 31 EPS = 2.5612042E-01
IT = 32 EPS = 2.4742678E-01
IT = 33 EPS = 2.3914235E-01
IT = 34 EPS = 2.3122085E-01
IT = 35 EPS = 2.2372279E-01
IT = 36 EPS = 2.1656870E-01

IT = 37 EPS = 2.1053367E-01
IT = 38 EPS = 2.0475639E-01
IT = 39 EPS = 1.9919003E-01
IT = 40 EPS = 1.9381369E-01
IT = 41 EPS = 1.8865693E-01
IT = 42 EPS = 1.8370365E-01
IT = 43 EPS = 1.7892230E-01
IT = 44 EPS = 1.7431737E-01
IT = 45 EPS = 1.6990997E-01
IT = 46 EPS = 1.6610415E-01
IT = 47 EPS = 1.6240145E-01
IT = 48 EPS = 1.5882880E-01
IT = 49 EPS = 1.5536742E-01
IT = 50 EPS = 1.5200257E-01
IT = 51 EPS = 1.4873235E-01
IT = 52 EPS = 1.4558697E-01
IT = 53 EPS = 1.4253020E-01
IT = 54 EPS = 1.3955924E-01
IT = 55 EPS = 1.3667643E-01
IT = 56 EPS = 1.3416280E-01
IT = 57 EPS = 1.3170609E-01
IT = 58 EPS = 1.2930527E-01
IT = 59 EPS = 1.2696333E-01
IT = 60 EPS = 1.2469535E-01
IT = 61 EPS = 1.2247988E-01
IT = 62 EPS = 1.2031570E-01
IT = 63 EPS = 1.1820155E-01
IT = 64 EPS = 1.1615662E-01
IT = 65 EPS = 1.1415943E-01
IT = 66 EPS = 1.1220850E-01
IT = 67 EPS = 1.1046697E-01
IT = 68 EPS = 1.0876979E-01
IT = 69 EPS = 1.0711144E-01
IT = 70 EPS = 1.0548507E-01
IT = 71 EPS = 1.0389012E-01
IT = 72 EPS = 1.0232786E-01

IT = 73 EPS = 1.0080908E-01
IT = 74 EPS = 9.9319922E-02
IT = 75 EPS = 9.7859759E-02
IT = 76 EPS = 9.6427978E-02
IT = 77 EPS = 9.5029531E-02
IT = 78 EPS = 9.3665652E-02
IT = 79 EPS = 9.2434413E-02
IT = 80 EPS = 9.1223224E-02
IT = 81 EPS = 9.0031809E-02
IT = 82 EPS = 8.8866148E-02
IT = 83 EPS = 8.7724653E-02
IT = 84 EPS = 8.6602028E-02
IT = 85 EPS = 8.5497964E-02
IT = 86 EPS = 8.4412147E-02
IT = 87 EPS = 8.3349209E-02
IT = 88 EPS = 8.2308684E-02
IT = 89 EPS = 8.1285425E-02
IT = 90 EPS = 8.0279113E-02
IT = 91 EPS = 7.9289431E-02
IT = 92 EPS = 7.8389276E-02
IT = 93 EPS = 7.7507920E-02
IT = 94 EPS = 7.6639034E-02
IT = 95 EPS = 7.5782459E-02
IT = 96 EPS = 7.4938032E-02
IT = 97 EPS = 7.4105590E-02
IT = 98 EPS = 7.3291357E-02
IT = 99 EPS = 7.2489794E-02
IT = 100 EPS = 7.1699681E-02
ADI Benchmark Completed.
Size = 332 x 332 x 332
Iterations = 100
Time in seconds = 7.73
Operation type = double precision
Verification = UNSUCCESSFUL
END OF ADI Benchmark