

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Институт №8 «Информационные технологии и прикладная математика»  
Кафедра 806 «Вычислительная математика и программирование»**

**Лабораторная работа №3  
по курсу «Параллельная обработка данных»**

**Технология MPI и технология OpenMP**

Выполнил: А.В. Ефимов

Группа: 8О-407Б

Преподаватели: К.Г. Крашенинников,  
А.Ю. Морозов

Москва, 2021

## Условие

Совместное использование технологии MPI и технологии OpenMP. Реализация метода Якоби. Решение задачи Дирихле для уравнения Лапласа в трехмерной области с граничными условиями первого рода.

Вариант: Распараллеливание в общем виде с разделением работы между нитями вручную (“в стиле CUDA”).

## Программное и аппаратное обеспечение

```
##### CUDA Info

/opt/cuda/extras/demo_suite/deviceQuery Starting...

  CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 1 CUDA Capable device(s)

Device 0: "NVIDIA GeForce MX150"
  CUDA Driver Version / Runtime Version      11.4 / 11.4
  CUDA Capability Major/Minor version number: 6.1
  Total amount of global memory:             2003 MBytes (2099904512
bytes)
  ( 3) Multiprocessors, (128) CUDA Cores/MP: 384 CUDA Cores
  GPU Max Clock rate:                        1532 MHz (1.53 GHz)
  Memory Clock rate:                         3004 Mhz
  Memory Bus Width:                          64-bit
  L2 Cache Size:                             524288 bytes
  Maximum Texture Dimension Size (x,y,z)     1D=(131072), 2D=(131072,
65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers 1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(32768, 32768), 2048
layers
  Total amount of constant memory:            65536 bytes
  Total amount of shared memory per block:    49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                 32
  Maximum number of threads per multiprocessor: 2048
  Maximum number of threads per block:        1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     2147483647 bytes
  Texture alignment:                         512 bytes
  Concurrent copy and kernel execution:      Yes with 2 copy engine(s)
  Run time limit on kernels:                  Yes
  Integrated GPU sharing Host Memory:         No
  Support host page-locked memory mapping:    Yes
  Alignment requirement for Surfaces:         Yes
  Device has ECC support:                     Disabled
  Device supports Unified Addressing (UVA):    Yes
  Device supports Compute Preemption:         Yes
  Supports Cooperative Kernel Launch:         Yes
  Supports MultiDevice Co-op Kernel Launch:   Yes
  Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device
simultaneously) >
```

```
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 11.4, CUDA Runtime
Version = 11.4, NumDevs = 1, Device0 = NVIDIA GeForce MX150
Result = PASS
```

#### ##### CPU Info

```
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Address sizes:               39 bits physical, 48 bits virtual
Byte Order:                  Little Endian
CPU(s):                      8
On-line CPU(s) list:         0-7
Vendor ID:                   GenuineIntel
Model name:                  Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU family:                   6
Model:                       142
Thread(s) per core:          2
Core(s) per socket:          4
Socket(s):                   1
Stepping:                    10
CPU max MHz:                  3400.0000
CPU min MHz:                  400.0000
BogoMIPS:                     3601.00
Flags:                        fpu vme de pse tsc msr pae mce cx8 apic
sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht
tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2
x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm
abm 3dnowprefetch cpuid_fault epb invpcid_single pti ibrs ibpb stibp
tpu_shadow_vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2
smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt
xsavcvec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window
hwp_epp
Virtualization:              VT-x
L1d cache:                   128 KiB (4 instances)
L1i cache:                   128 KiB (4 instances)
L2 cache:                    1 MiB (4 instances)
L3 cache:                    6 MiB (1 instance)
NUMA node(s):                1
NUMA node0 CPU(s):           0-7
Vulnerability Itlb multihit:  KVM: Mitigation: VMX disabled
Vulnerability L1tf:           Mitigation; PTE Inversion; VMX conditional
cache flushes, SMT vulnerable
Vulnerability Mds:             Vulnerable: Clear CPU buffers attempted,
no microcode; SMT vulnerable
Vulnerability Meltdown:        Mitigation; PTI
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1:      Mitigation; usercopy/swapgs barriers and
__user pointer sanitization
Vulnerability Spectre v2:      Mitigation; Full generic retpoline, IBPB
conditional, IBRS_FW, STIBP conditional, RSB filling
Vulnerability Srbds:           Vulnerable: No microcode
Vulnerability Tsx async abort: Not affected
```

#### ##### RAM Info

	total	used	free	shared	buff/cache
available					
Mem:	7.6Gi	1.7Gi	4.6Gi	633Mi	1.4Gi
5.1Gi					
Swap:	0B	0B	0B		

## Метод решения

Лабораторная является надстройкой над седьмой. В ней дополнительно реализуются:

1. Передача границ напрямую из сетки (без промежуточного буфера) с использованием задаваемых через типы шаблонов;
2. Распараллеливание основного цикла вычисления сетки через OpenMP на подобие ядра CUDA.

Первое выполняется простым заданием для каждой границы через *subarray* одного шаблона для отправления и одного – для получения, и вставлением полученного шаблона в тип данных функций для получения и отправления в соответствующие границы.

Второе достигается подсчетом соответствующих 3д индексов сетки для каждого 1д значения (OpenMP не имеет сеточной абстракции как CUDA и возвращает только номер потока и количество потоков вообще), которые получены либо в начале, либо после сдвига.

## Описание программы

Пример шаблонов для двух сторон по X:

```
int sizes[3] = {pad_block_size[x_dir], pad_block_size[y_dir],
pad_block_size[z_dir]};
int subsizes[3], coords[3];

// X dir

MPI_Datatype yz_send_left, yz_send_right;
MPI_Datatype yz_recv_left, yz_recv_right;

subsizes[0] = 1;
subsizes[1] = block_size[y_dir];
subsizes[2] = block_size[z_dir];

coords[0] = 1;
coords[1] = 1;
coords[2] = 1;
MPI_Type_create_subarray(3, sizes, subsizes, coords, MPI_ORDER_FORTRAN,
MPI_DOUBLE, &yz_send_left);
MPI_Type_commit(&yz_send_left);

coords[0] = block_size[x_dir];
coords[1] = 1;
coords[2] = 1;
MPI_Type_create_subarray(3, sizes, subsizes, coords, MPI_ORDER_FORTRAN,
MPI_DOUBLE, &yz_send_right);
MPI_Type_commit(&yz_send_right);

coords[0] = 0;
coords[1] = 1;
coords[2] = 1;
```

```

MPI_Type_create_subarray(3, sizes, subsizes, coords, MPI_ORDER_FORTRAN,
MPI_DOUBLE, &yz_recv_left);
MPI_Type_commit(&yz_recv_left);

coords[0] = block_size[x_dir] + 1;
coords[1] = 1;
coords[2] = 1;
MPI_Type_create_subarray(3, sizes, subsizes, coords, MPI_ORDER_FORTRAN,
MPI_DOUBLE, &yz_recv_right);
MPI_Type_commit(&yz_recv_right); }

```

OpenMP формат основного цикла. Дополнительно дописывается директива для учета операции *max* редукции.

```

#pragma omp parallel reduction(max:max_diff)
{
    int thread_i = omp_get_thread_num();
    int offset = omp_get_num_threads();

    int grid_size = block_size[x_dir] * block_size[y_dir] *
block_size[z_dir];

    int x, y, z;

    for (size_t i = thread_i; i < grid_size; i += offset) {
        x = i % block_size[x_dir];
        y = (i / block_size[x_dir]) % block_size[y_dir];
        z = i / (block_size[x_dir] * block_size[y_dir]);

        double val = (
            (old_grid[calc_1d_o(x+1, y, z)] +
old_grid[calc_1d_o(x-1, y, z)]) * inv_hx2 +
            (old_grid[calc_1d_o(x, y+1, z)] +
old_grid[calc_1d_o(x, y-1, z)]) * inv_hy2 +
            (old_grid[calc_1d_o(x, y, z+1)] +
old_grid[calc_1d_o(x, y, z-1)]) * inv_hz2
            ) / (2 * (inv_hx2 + inv_hy2 + inv_hz2));
        new_grid[calc_1d_o(x, y, z)] = val;
        max_diff = std::max( max_diff, std::abs(val -
old_grid[calc_1d_o(x, y, z)]) );
    }
}

```

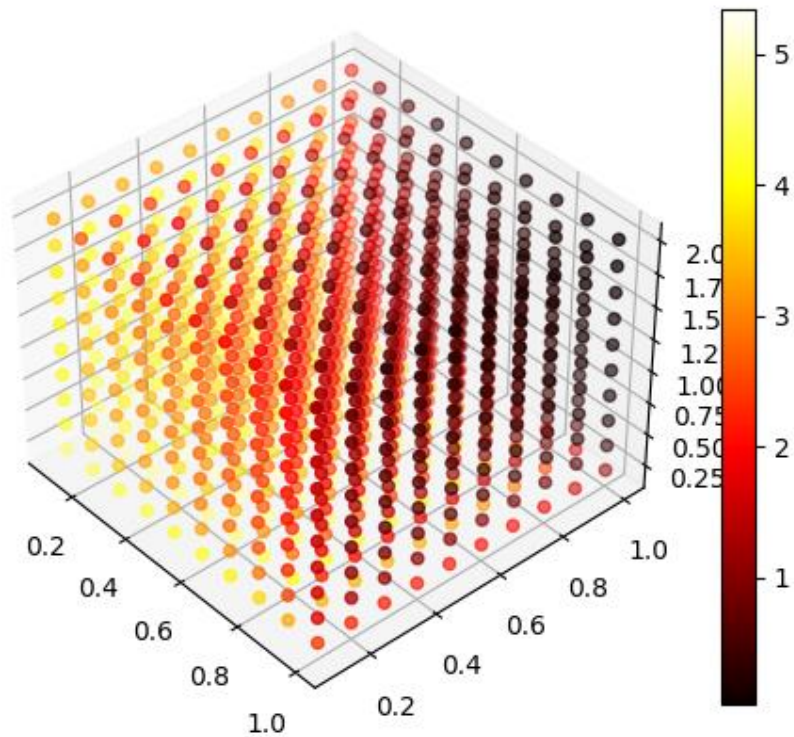
## Результаты

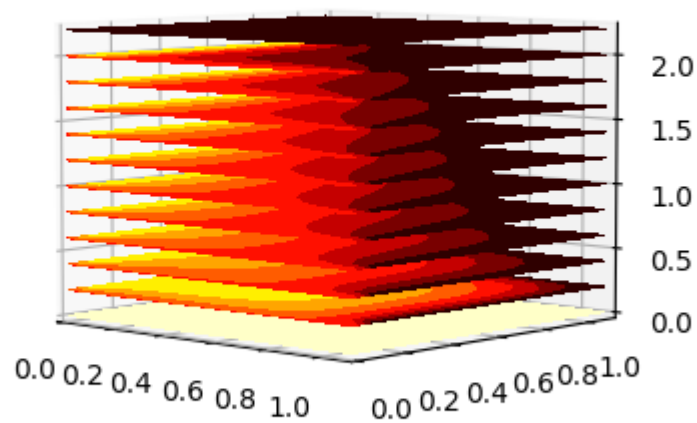
Формат бенчмарка такой же, как в предыдущей лабораторной.

Количество процессов по Z	CPU	GPU
1	37.7729	54.4979
2	20.8039	27.9975
3	33.2914	<b>Time Limit</b>
4	33.2914	<b>Time Limit</b>

При трех и четырех параллельных процессах был съеден весь процессор и достигнуты температуры выше нормы. За 5 минут вычислений не достигнуто.

Так как выходные данные не поменялись (а если поменялись, то незначительно для графика), используются те же графики:





## Выводы

- Так как здесь не происходит копирование на GPU, время, уходящее на это копирование равно нулю, причем копирование происходит напрямую из сетки.
- Параллелить алгоритмы хорошо, но нужно быть уверенным, что железо, на котором это все запускается, может выдержать взаимодействующие процессы, каждый из которых имеет свою горстку потоков.