Отчет по лабораторной работе №3. Мандарханов Данил Михайлович. Группа 22207. Вариант float-2

Задание

Постановка задачи

1. В векторизованной программе из практического задания 2 реорганизовать вычисления таким образом, чтобы за один проход по сетке выполнялось сразу несколько итераций метода. Сделать сначала реализацию для двух итераций за проход. Если есть ускорение, сделать реализацию для трёх итераций за проход. И так далее, пока ускорение не перестанет иметь место.
2. Проанализировать производительность наиболее быстрой версии программы аналогично анализу в задании 2 (включая roofline-модель). Сравнить результаты анализа с результатами в задании 2.

Параметры программы

$N_x = N_y = 8000, N_t = 128$

Не оптимизированная по памяти программа (программа из lab_2)

```
float computeDelta(float* phi, float* phi_new) {
    float d;
    float stepDelta = -1.0f;
    for (int i = 0; i < N_y; i++) {
        for (int j = 0; j < N_x; j++) {
```

```
            d = fabs(phi[i * N_x + j] - phi_new[i * N_x + j]);
            if (d > stepDelta) stepDelta = d;
        }
    }
    return stepDelta;
}

void computeLine(float* phi_new, float* phi, float* rho, int line_index) {
    int index;
    for (int j = 1; j < VECTORS_NUMBER_IN_LINE * VECTOR_SIZE_IN_FLOATS + 1; j += VECTOR_SIZE_IN_FLOATS) {
        index = line_index + j;

        __m128 v_phi_left     = _mm_loadu_ps(&phi[index - 1]);
        __m128 v_phi_right    = _mm_loadu_ps(&phi[index + 1]);
        __m128 v_phi_bottom   = _mm_loadu_ps(&phi[index - N_x]);
        __m128 v_phi_top      = _mm_loadu_ps(&phi[index + N_x]);
        __m128 v_phi_bot_left = _mm_loadu_ps(&phi[index - N_x - 1]);
        __m128 v_phi_bot_right= _mm_loadu_ps(&phi[index - N_x + 1]);
        __m128 v_phi_top_left = _mm_loadu_ps(&phi[index + N_x - 1]);
        __m128 v_phi_top_right= _mm_loadu_ps(&phi[index + N_x + 1]);

        __m128 v_rho_center   = _mm_loadu_ps(&rho[index]);
        __m128 v_rho_bottom   = _mm_loadu_ps(&rho[index - N_x]);
        __m128 v_rho_top      = _mm_loadu_ps(&rho[index + N_x]);
        __m128 v_rho_left     = _mm_loadu_ps(&rho[index - 1]);
        __m128 v_rho_right    = _mm_loadu_ps(&rho[index + 1]);

        __m128 first_line    = _mm_add_ps(_mm_mul_ps(firstKoef_m128, _mm_add_ps(v_phi_left, v_phi_right)),
                                          _mm_mul_ps(secondKoef_m128, _mm_add_ps(v_phi_bottom, v_phi_top)));
        __m128 second_line   = _mm_mul_ps(thirdKoef_m128,
                                          _mm_add_ps(_mm_add_ps(v_phi_bot_left, v_phi_top_left),
                                                     _mm_add_ps(v_phi_bot_right, v_phi_top_right)));
        __m128 third_line    = _mm_add_ps(_mm_mul_ps(_mm_set1_ps(0.25f),
                                                     _mm_add_ps(_mm_add_ps(v_rho_bottom, v_rho_top),
```

```
                                                                _mm_add_ps(v_rho_left, v_rho_right))),
                                            _mm_mul_ps(_mm_set1_ps(2.0f), v_rho_center));
        __m128 result        = _mm_mul_ps(mainKoef_m128,
                                            _mm_add_ps(first_line, _mm_add_ps(second_line, third_line)));

        _mm_storeu_ps(&phi_new[index], result);
    }

    for (int j = VECTORS_NUMBER_IN_LINE * VECTOR_SIZE_IN_FLOATS + 1; j < N_x - 1; j++) {
        index = line_index + j;

        phi_new[index] = mainKoef * (firstKoef * (phi[index - 1] + phi[index + 1]) +
                                     secondKoef * (phi[index - N_x] + phi[index + N_x]) +
                                     thirdKoef * (phi[index - N_x - 1] + phi[index - N_x + 1] + phi[index + N_x - 1] +
phi[index + N_x + 1]) +
                                     2.0f * rho[index] +
                                     0.25f * (rho[index - N_x] + rho[index + N_x] + rho[index - 1] + rho[index + 1]));
    }
}

void runJacobyMethod (float *rho) {
    float *phi;
    phi = (float*)malloc(2 * N_x * N_y * sizeof(float));
    float *phi_new = phi + N_x * N_y;
    for (int i = 0; i < N_y * 2; i++) {
        for (int j = 0; j < N_x; j++) {
            phi[i*N_y + j] = 0.0f;
        }
    }

    float stepDelta;
    float globalDelta = 1.0;
    int iterNumber = 0;

```

```
        long long t1, t2;
        double tDiff;
        struct timespec curTime;
        clock_gettime(CLOCK_BOOTTIME, &curTime);
        t1 = curTime.tv_sec * 1000000000 + curTime.tv_nsec;

    while (iterNumber < N_t) {
        for (int i = 1; i < N_y - 1; i++) {
            computeLine(phi_new, phi, rho, i * N_x);
        }

        stepDelta = computeDelta(phi, phi_new);
        if ((stepDelta - globalDelta) < 0.0000001) {
            globalDelta = stepDelta;
            swapFloatPointers(&phi, &phi_new);
            iterNumber++;
        }
        else {
            printf("Delta is growwing!\nJacoby method stopped\n");
            break;
        }
    }

        clock_gettime(CLOCK_BOOTTIME, &curTime);
        t2 = curTime.tv_sec * 1000000000 + curTime.tv_nsec;
        tDiff = (double) (t2 - t1) / 1000000000.0;
        printf("Time = %g s\n", tDiff);

    // fillFile(phi, "phi_unalign.dat");

    if(iterNumber % 2 == 1) swapFloatPointers(&phi, &phi_new);
    free(phi);
}
```

Оптимизированная по памяти программа (16 "итераций" за шаг)

```
    while (iterNumber < (N_t / LINE_NUMBER)) {

        for(int i = 1; i < LINE_NUMBER; i++) {
            for(int k = 1; k <= i; k++) {
                (k % 2 == 1) ?
                    computeLine(phi_new, phi, rho, (i - k + 1) * N_x)
                :
                    computeLine(phi, phi_new, rho, (i - k + 1) * N_x);
            }
        }

        for (int i = LINE_NUMBER; i < N_y - 1; i++) {
            for (int k = 1; k <= LINE_NUMBER; k++) {
                (k % 2 == 1) ?
                    computeLine(phi_new, phi, rho, (i - k + 1) * N_x)
                :
                    computeLine(phi, phi_new, rho, (i - k + 1) * N_x);
            }
        }

        int i = N_y - 2;
        for (int k = LINE_NUMBER - 1; k >= 1; k--) {
            (k % 2 == 1) ?
                computeLine(phi, phi_new, rho, (i + k - (LINE_NUMBER - 1)) * N_x)
            :
                computeLine(phi_new, phi, rho, (i + k - (LINE_NUMBER - 1)) * N_x);
        }

        stepDelta = computeDelta(phi, phi_new);
```

```
30          if ((stepDelta - globalDelta) < 0.0000001) {
31              globalDelta = stepDelta;
32              // swapFloatPointers(&phi, &phi_new);
33              iterNumber++;
34          }
35          else {
36              printf("Delta is growwing!\nJacoby method stopped\n");
37              break;
38          }
39      }
```

Результаты

```
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./1line.out
Time = 32.1672 s
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./2line.out
Time = 25.8492 s
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./4line.out
Time = 22.8982 s
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./8line.out
Time = 20.8512 s
```

```
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./16line.out
Time = 19.986 s
```

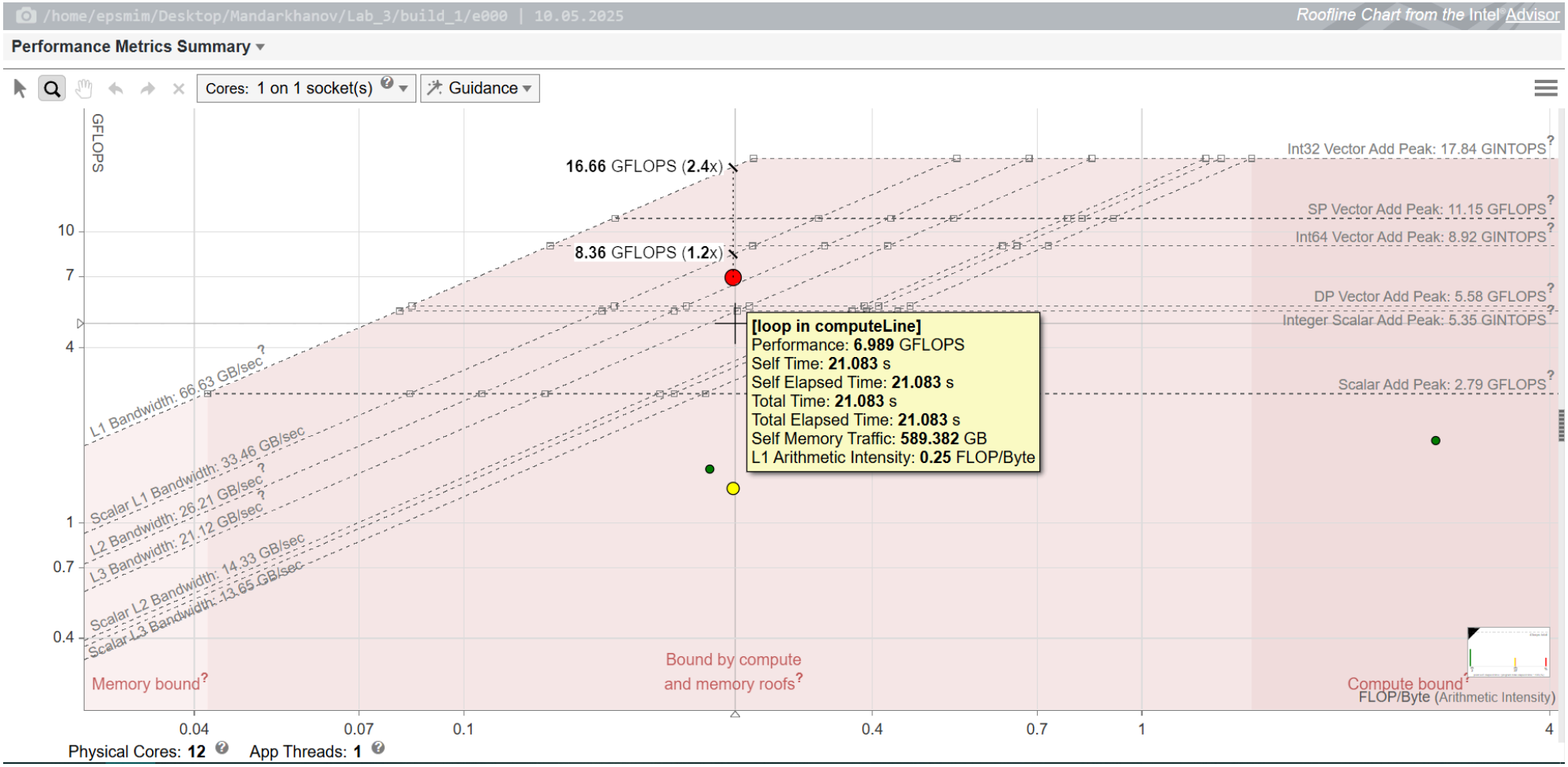
```
epsmim@comrade:~/Desktop/Mandarkhanov/Lab_3$ ./20line.out
Time = 20.6437 s
```

Как видно из запусков, начиная с 16-ти итераций за шаг скорость работы программы уменьшается мало

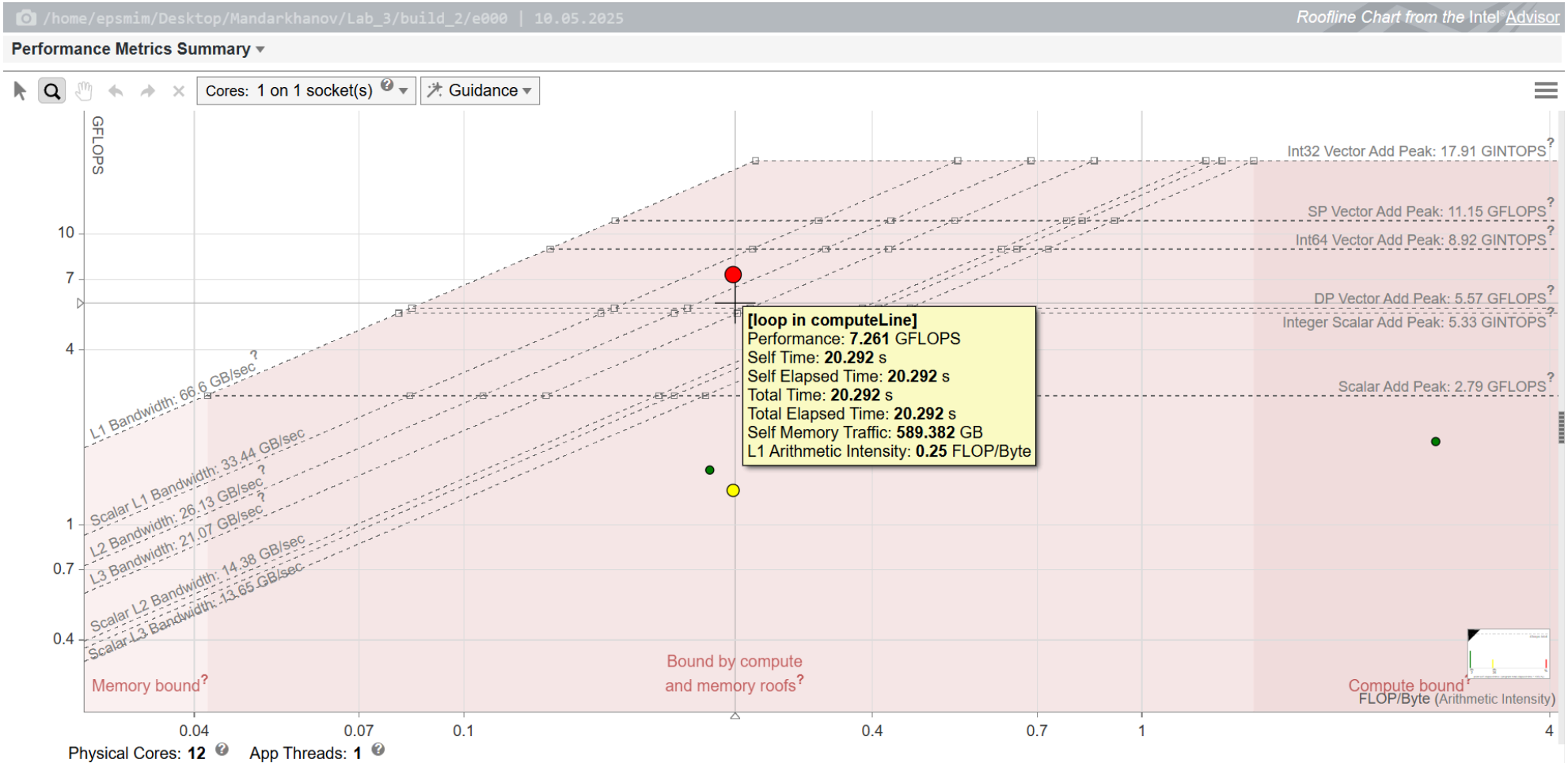
Далее увеличивая число итераций за шаг время работы программы увеличивается

Roofline-модель

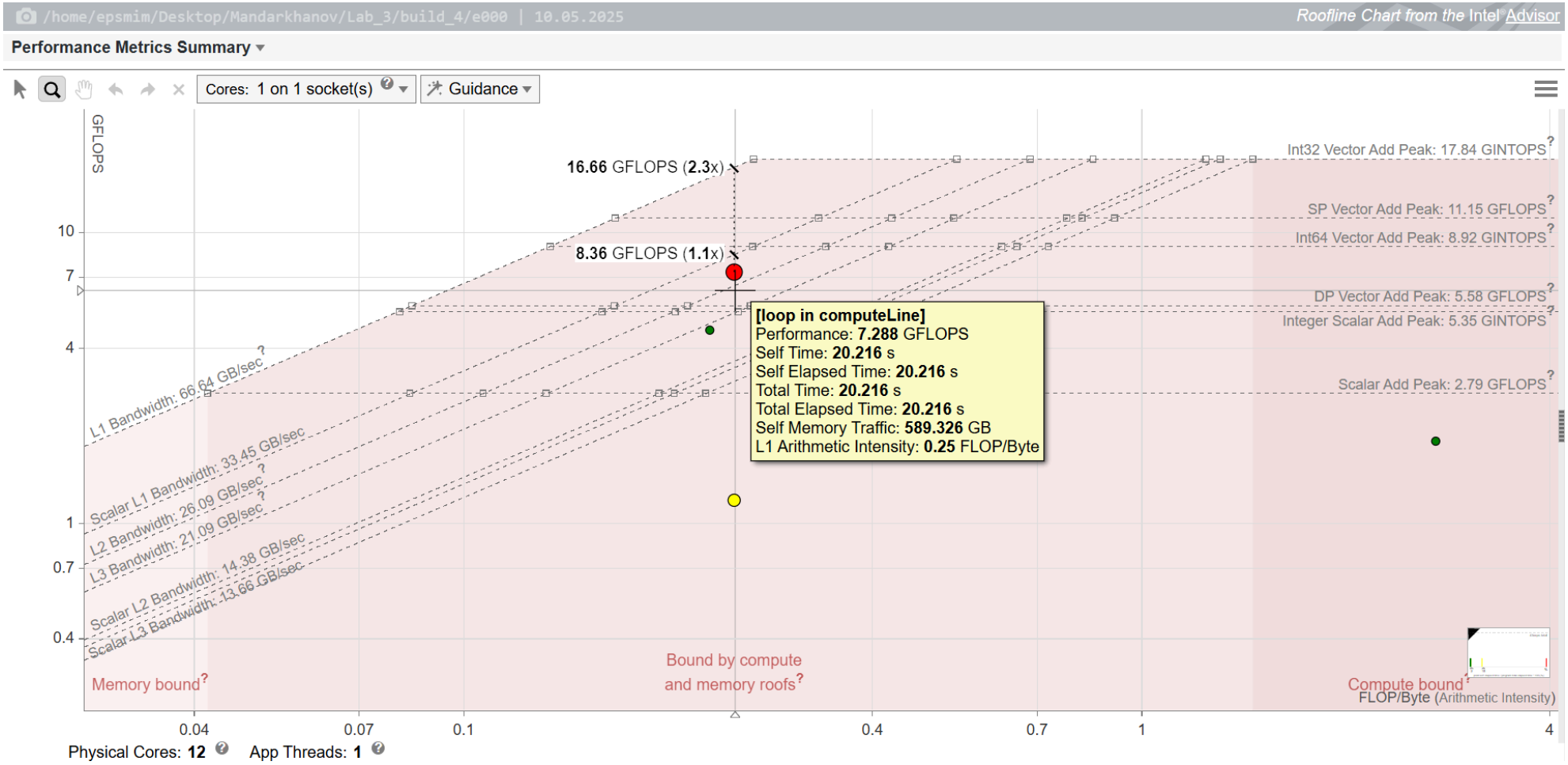
1line


/home/epsmim/Desktop/Mandarkhanov/Lab_3/build_1/e000 | 10.05.2025
Roofline Chart from the Intel Advisor
Performance Metrics Summary
Cores: 1 on 1 socket(s)
Guidance
GFLOPS
16.66 GFLOPS (2.4x)
8.36 GFLOPS (1.2x)
Int32 Vector Add Peak: 17.84 GINTOPS
SP Vector Add Peak: 11.15 GFLOPS
Int64 Vector Add Peak: 8.92 GINTOPS
DP Vector Add Peak: 5.58 GFLOPS
Integer Scalar Add Peak: 5.35 GINTOPS
Scalar Add Peak: 2.79 GFLOPS
L1 Bandwidth: 66.63 GB/sec
Scalar L1 Bandwidth: 33.46 GB/sec
L2 Bandwidth: 26.21 GB/sec
L3 Bandwidth: 21.12 GB/sec
Scalar L2 Bandwidth: 14.33 GB/sec
Scalar L3 Bandwidth: 13.65 GB/sec
[loop in computeLine]
Performance: 6.989 GFLOPS
Self Time: 21.083 s
Self Elapsed Time: 21.083 s
Total Time: 21.083 s
Total Elapsed Time: 21.083 s
Self Memory Traffic: 589.382 GB
L1 Arithmetic Intensity: 0.25 FLOP/Byte
Memory bound
Bound by compute and memory roofs
Compute bound
FLOP/Byte (Arithmetic Intensity)
10
7
4
1
0.7
0.4
0.04
0.07
0.1
0.4
0.7
1
4
Physical Cores: 12
App Threads: 1

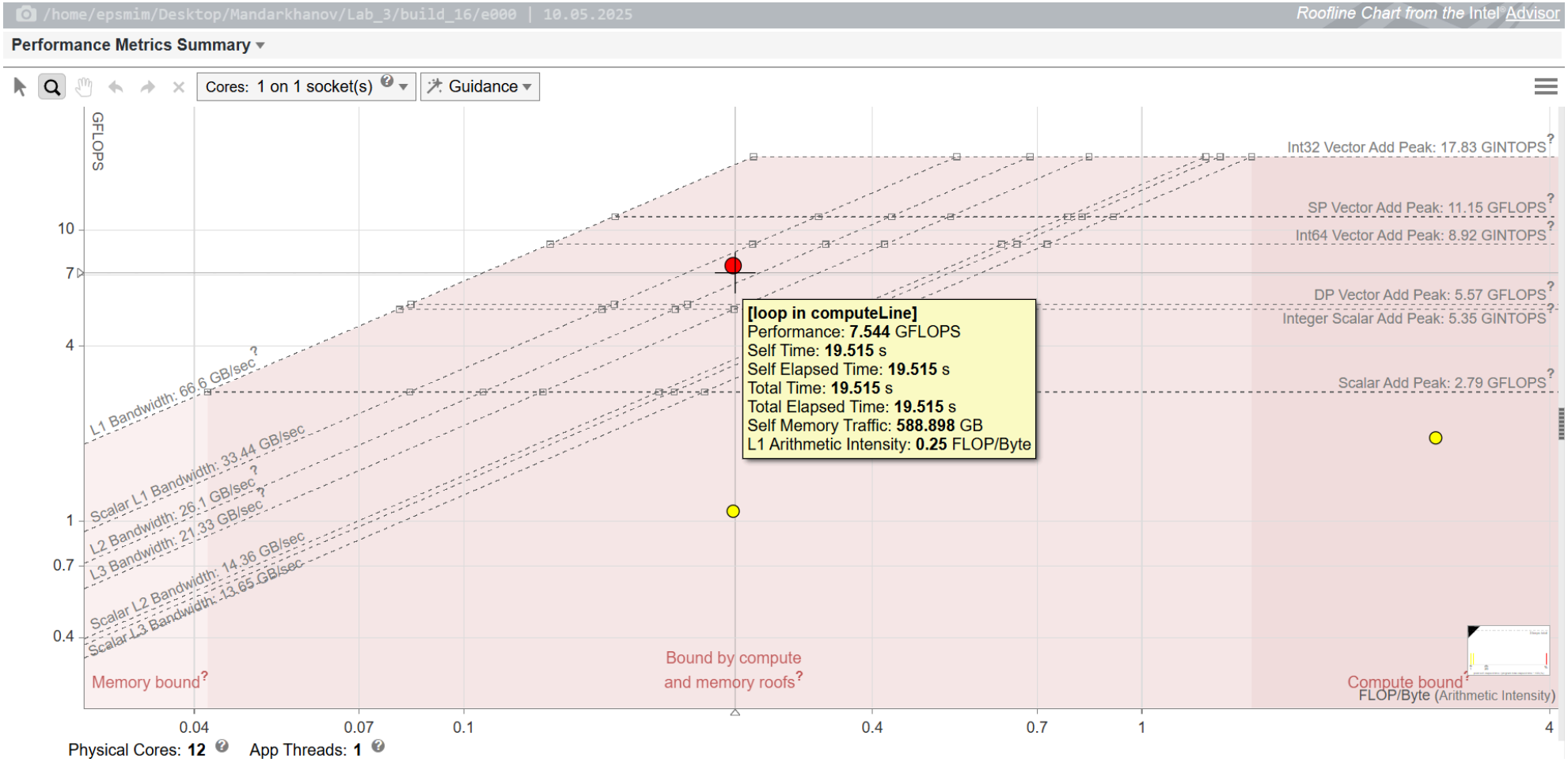
2line


/home/epsmim/Desktop/Mandarkhanov/Lab_3/build_2/e000 | 10.05.2025
Roofline Chart from the Intel Advisor
Performance Metrics Summary
Cores: 1 on 1 socket(s)
Guidance
GFLOPS
10
7
4
1
0.7
0.4
Int32 Vector Add Peak: 17.91 GINTOPS
SP Vector Add Peak: 11.15 GFLOPS
Int64 Vector Add Peak: 8.92 GINTOPS
DP Vector Add Peak: 5.57 GFLOPS
Integer Scalar Add Peak: 5.33 GINTOPS
Scalar Add Peak: 2.79 GFLOPS
L1 Bandwidth: 66.6 GB/sec
Scalar L1 Bandwidth: 33.44 GB/sec
L2 Bandwidth: 26.13 GB/sec
L3 Bandwidth: 21.07 GB/sec
Scalar L2 Bandwidth: 14.38 GB/sec
Scalar L3 Bandwidth: 13.65 GB/sec
[loop in computeLine]
Performance: 7.261 GFLOPS
Self Time: 20.292 s
Self Elapsed Time: 20.292 s
Total Time: 20.292 s
Total Elapsed Time: 20.292 s
Self Memory Traffic: 589.382 GB
L1 Arithmetic Intensity: 0.25 FLOP/Byte
Memory bound
Bound by compute and memory roofs
Compute bound
0.04
0.07
0.1
0.4
0.7
1
4
FLOP/Byte (Arithmetic Intensity)
Physical Cores: 12
App Threads: 1

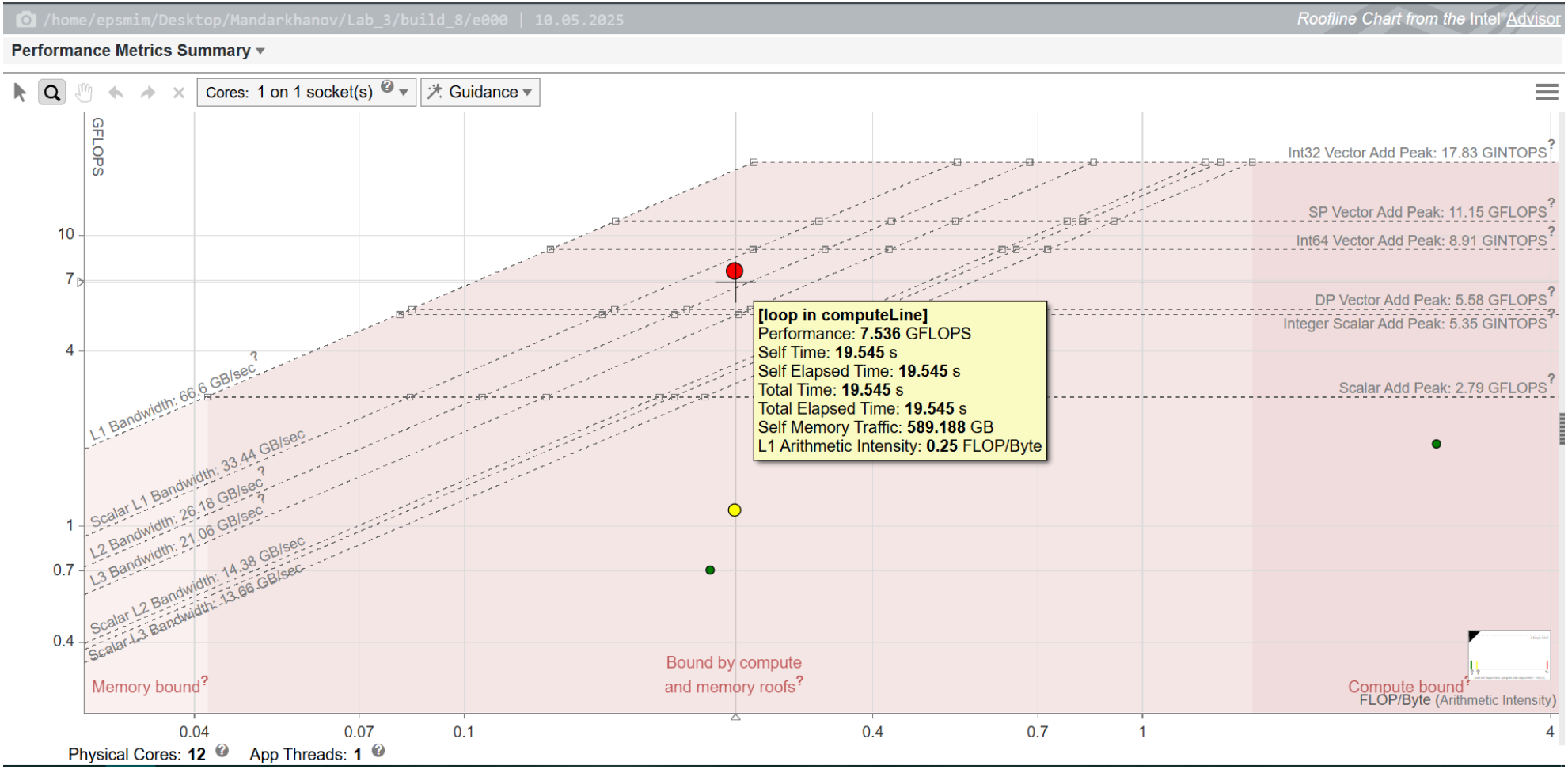
4line


/home/epsmim/Desktop/Mandarkhanov/Lab_3/build_4/e000 | 10.05.2025
Roofline Chart from the Intel Advisor
Performance Metrics Summary
Cores: 1 on 1 socket(s)
Guidance
GFLOPS
16.66 GFLOPS (2.3x)
8.36 GFLOPS (1.1x)
[loop in computeLine]
Performance: 7.288 GFLOPS
Self Time: 20.216 s
Self Elapsed Time: 20.216 s
Total Time: 20.216 s
Total Elapsed Time: 20.216 s
Self Memory Traffic: 589.326 GB
L1 Arithmetic Intensity: 0.25 FLOP/Byte
Int32 Vector Add Peak: 17.84 GINTOPS
SP Vector Add Peak: 11.15 GFLOPS
Int64 Vector Add Peak: 8.92 GINTOPS
DP Vector Add Peak: 5.58 GFLOPS
Integer Scalar Add Peak: 5.35 GINTOPS
Scalar Add Peak: 2.79 GFLOPS
L1 Bandwidth: 66.64 GB/sec
Scalar L1 Bandwidth: 33.45 GB/sec
L2 Bandwidth: 26.09 GB/sec
L3 Bandwidth: 21.09 GB/sec
Scalar L2 Bandwidth: 14.38 GB/sec
Scalar L3 Bandwidth: 13.66 GB/sec
Memory bound
Bound by compute and memory roofs
Compute bound
FLOP/Byte (Arithmetic Intensity)
10
7
4
1
0.7
0.4
0.04
0.07
0.1
0.4
0.7
1
4
Physical Cores: 12
App Threads: 1

8line


/home/epsmim/Desktop/Mandarkhanov/Lab_3/build_16/e000 | 10.05.2025
Roofline Chart from the Intel® Advisor
Performance Metrics Summary
Cores: 1 on 1 socket(s)
Guidance
GFLOPS
Int32 Vector Add Peak: 17.83 GINTOPS
SP Vector Add Peak: 11.15 GFLOPS
Int64 Vector Add Peak: 8.92 GINTOPS
DP Vector Add Peak: 5.57 GFLOPS
Integer Scalar Add Peak: 5.35 GINTOPS
Scalar Add Peak: 2.79 GFLOPS
L1 Bandwidth: 66.6 GB/sec
Scalar L1 Bandwidth: 33.44 GB/sec
L2 Bandwidth: 26.1 GB/sec
L3 Bandwidth: 21.33 GB/sec
Scalar L2 Bandwidth: 14.36 GB/sec
Scalar L3 Bandwidth: 13.65 GB/sec
[loop in computeLine]
Performance: 7.544 GFLOPS
Self Time: 19.515 s
Self Elapsed Time: 19.515 s
Total Time: 19.515 s
Total Elapsed Time: 19.515 s
Self Memory Traffic: 588.898 GB
L1 Arithmetic Intensity: 0.25 FLOP/Byte
Memory bound
Bound by compute and memory roofs
Compute bound
FLOP/Byte (Arithmetic Intensity)
10
7
4
1
0.7
0.4
0.04
0.07
0.1
0.4
0.7
1
4
Physical Cores: 12
App Threads: 1

16line



Вывод

В результате проведенных оптимизаций можно сделать вывод, что программа стала работать быстрее и с увеличением числа подсчета итераций за шаг все ближе приближается к Scalar L1 уровню кэша