Эффективное программирование современных микропроцессоров и мультипроцессоров

Практическое задание 1

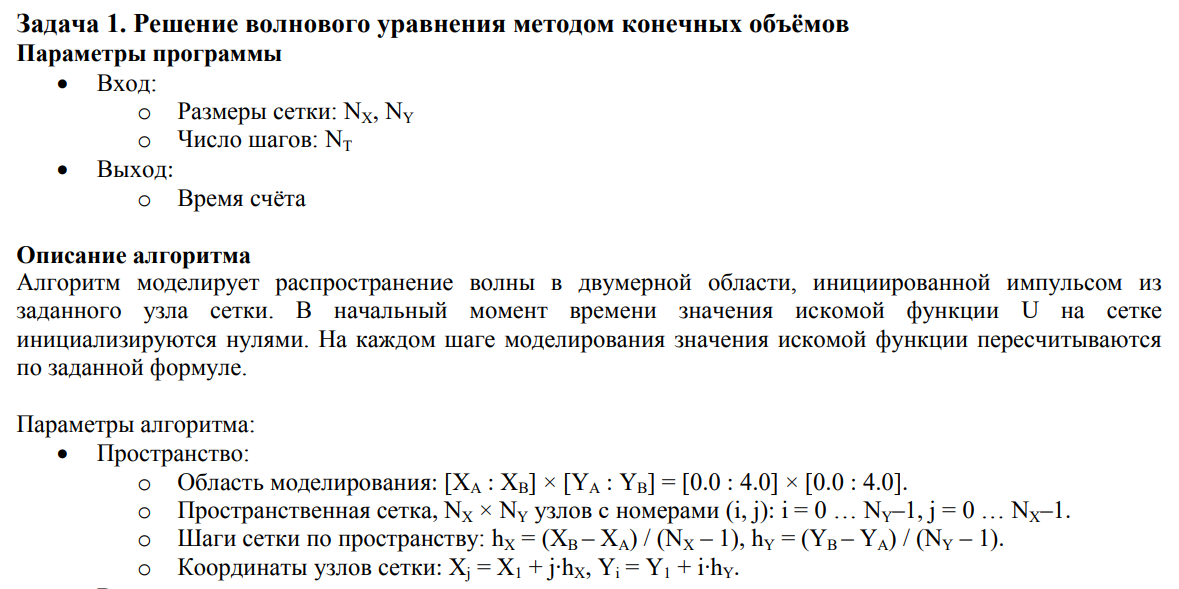
Выполнила:

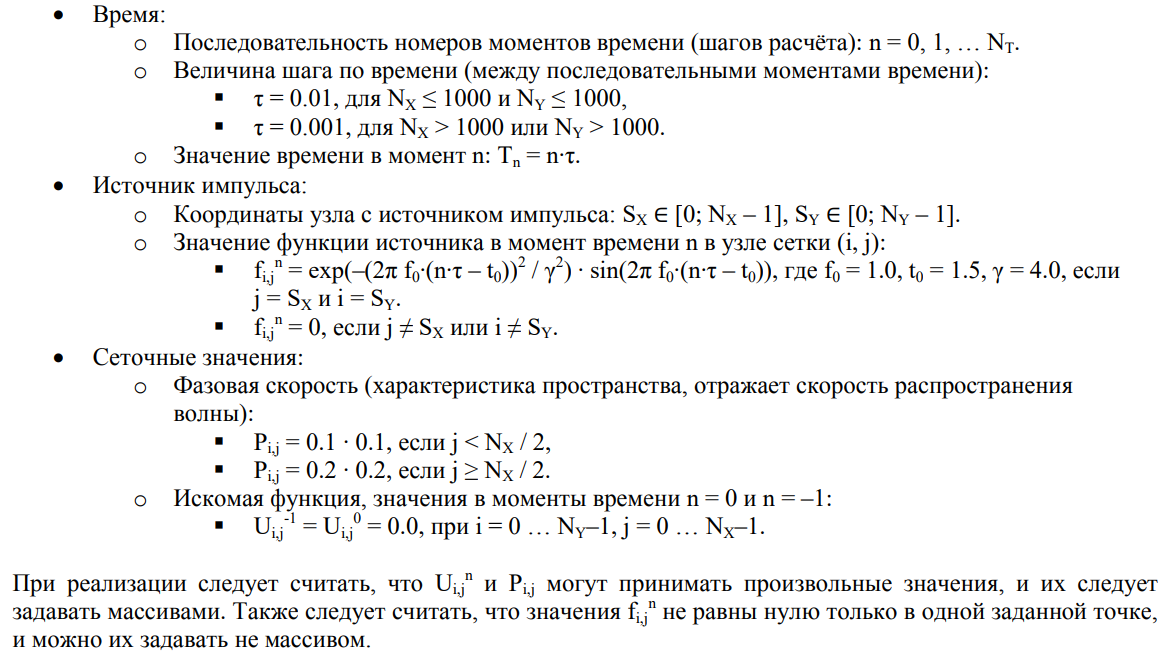
…

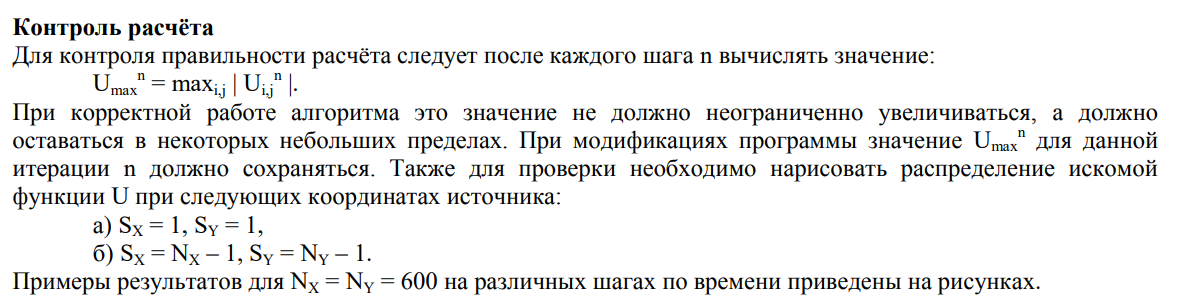
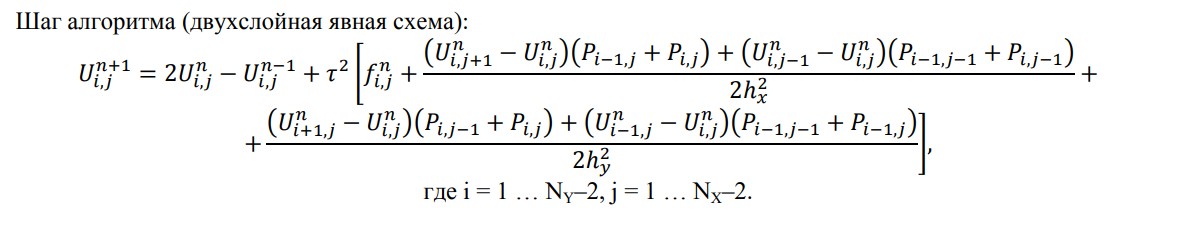
Группа …

Преподаватель:

…







**Результаты тестирования различных версий программы**

Nx = Ny = 8000, Nt = 100;

1. Изначальный вариант программы

|  |  |  |  |
| --- | --- | --- | --- |
| Без оптимизации | -О1 | -О3 | -Ofast |
| 204.2 | 47.2 | 37.5 | 22.9 |

1. Исключение повторяющихся вычислений

|  |  |  |  |
| --- | --- | --- | --- |
| Без оптимизации | -О1 | -О3 | -Ofast |
| 203.0 | 42.6 | 34.6 | 22.3 |

1. Замена всех операций деления на умножение

|  |  |  |  |
| --- | --- | --- | --- |
| Без оптимизации | -О1 | -О3 | -Ofast |
| 195.2 | 42.0 | 33.4 | 21.9 |

1. Заблаговременное обращение к массиву

|  |  |  |  |
| --- | --- | --- | --- |
| Без оптимизации | -О1 | -О3 | -Ofast |
| 139.7 | 38.7 | 32.9 | 22.4 |

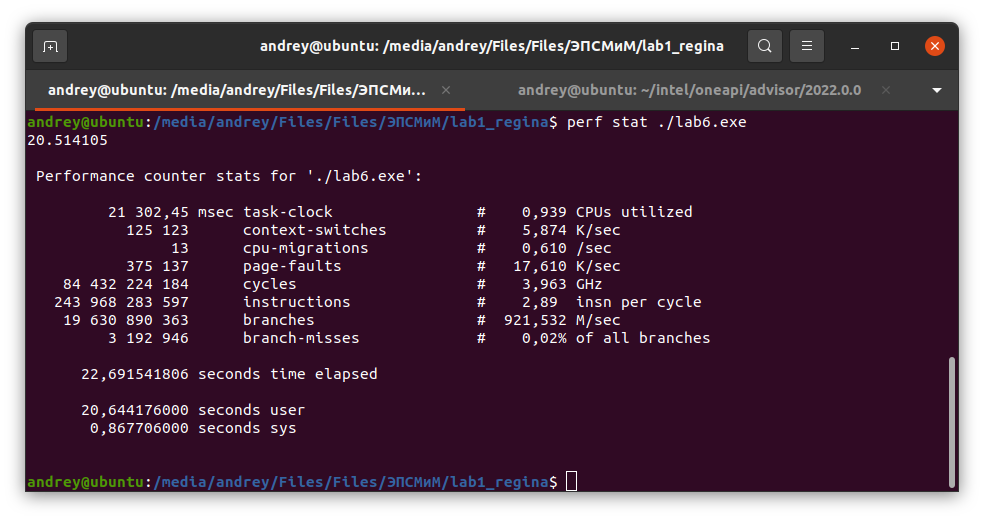
1. Заблаговременное вычисление индекса элемента массива

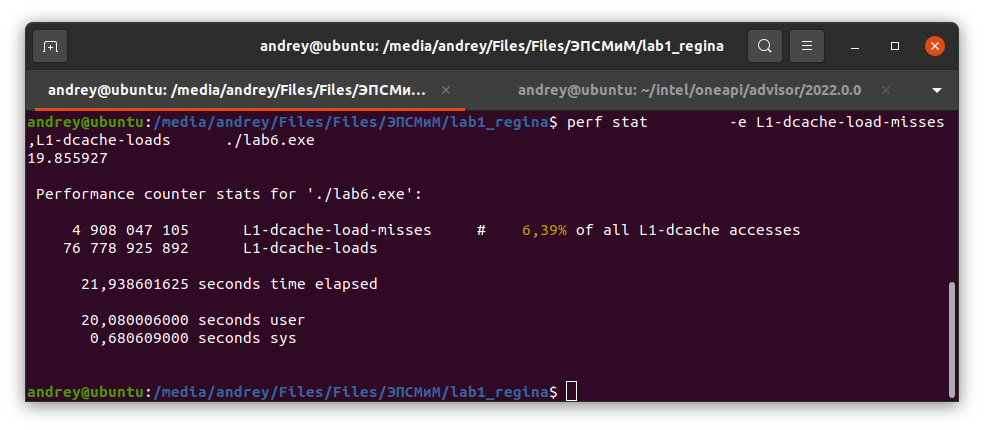
|  |  |  |  |
| --- | --- | --- | --- |
| Без оптимизации | -О1 | -О3 | -Ofast |
| 102.8 | 31.5 | 24.5 | 23.7 |

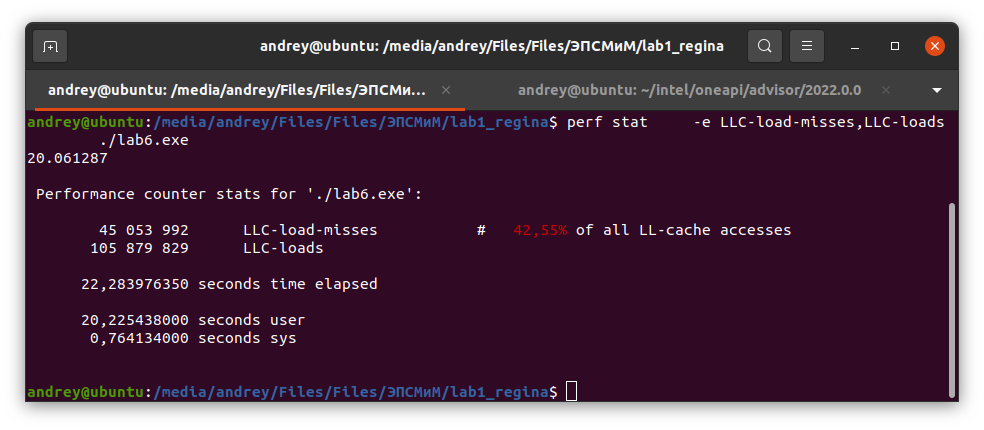
**Вывод**

Из приведённых выше результатов видно, что базовые способы оптимизации (исключение повторяющихся вычислений, замена операций деления на умножение, заблаговременное исчисление индексов) без оптимизаций компилятора помогли ускорить работу программы в 2 раза, а с –О3 в 1.5. Анализируя результаты профилирования, можно сделать вывод, что большая часть времени работы программы занята вычислениями, тогда как «память ждёт, пока к ней обратятся».

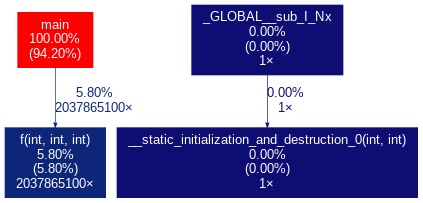
**Профилирование perf**



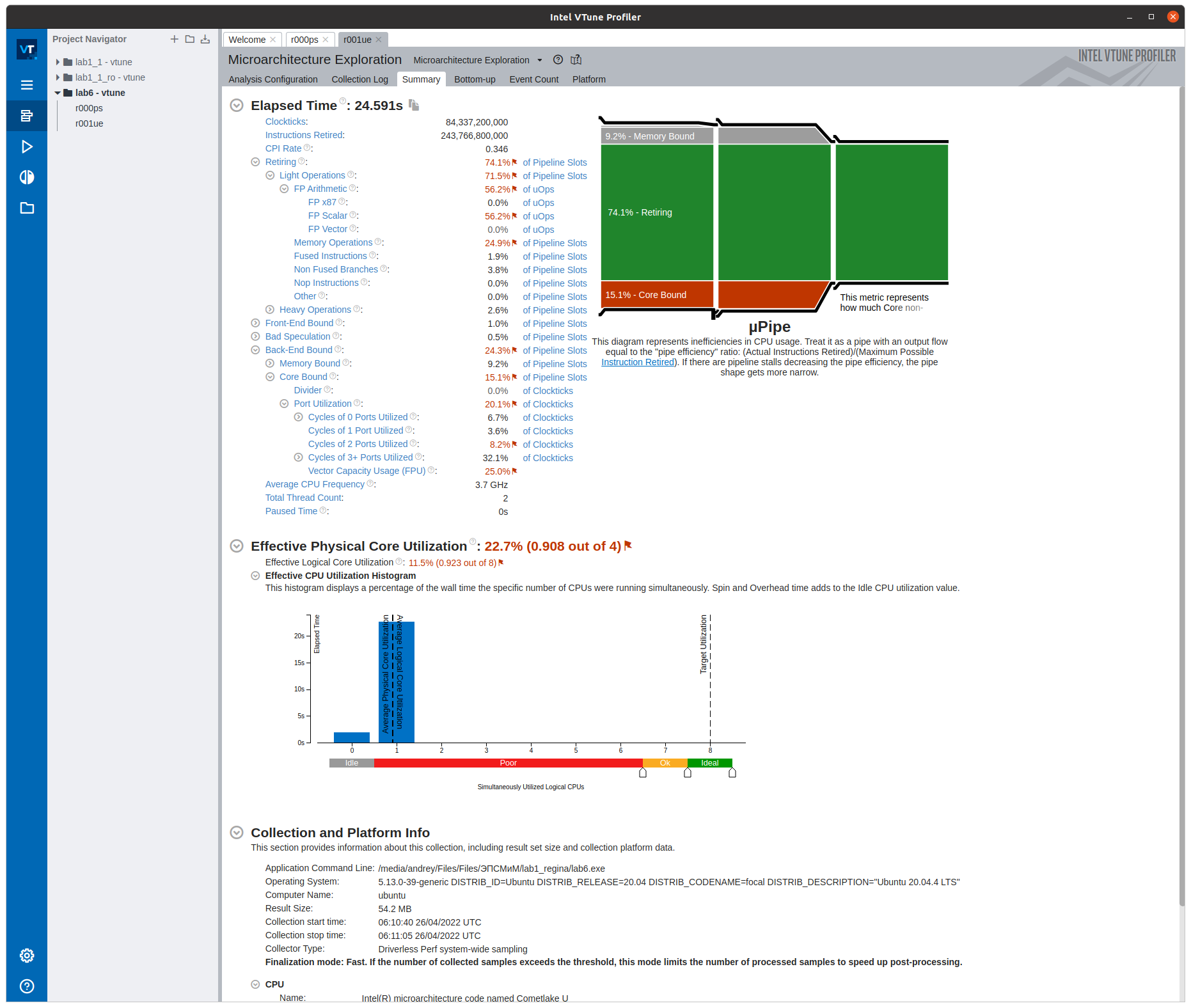




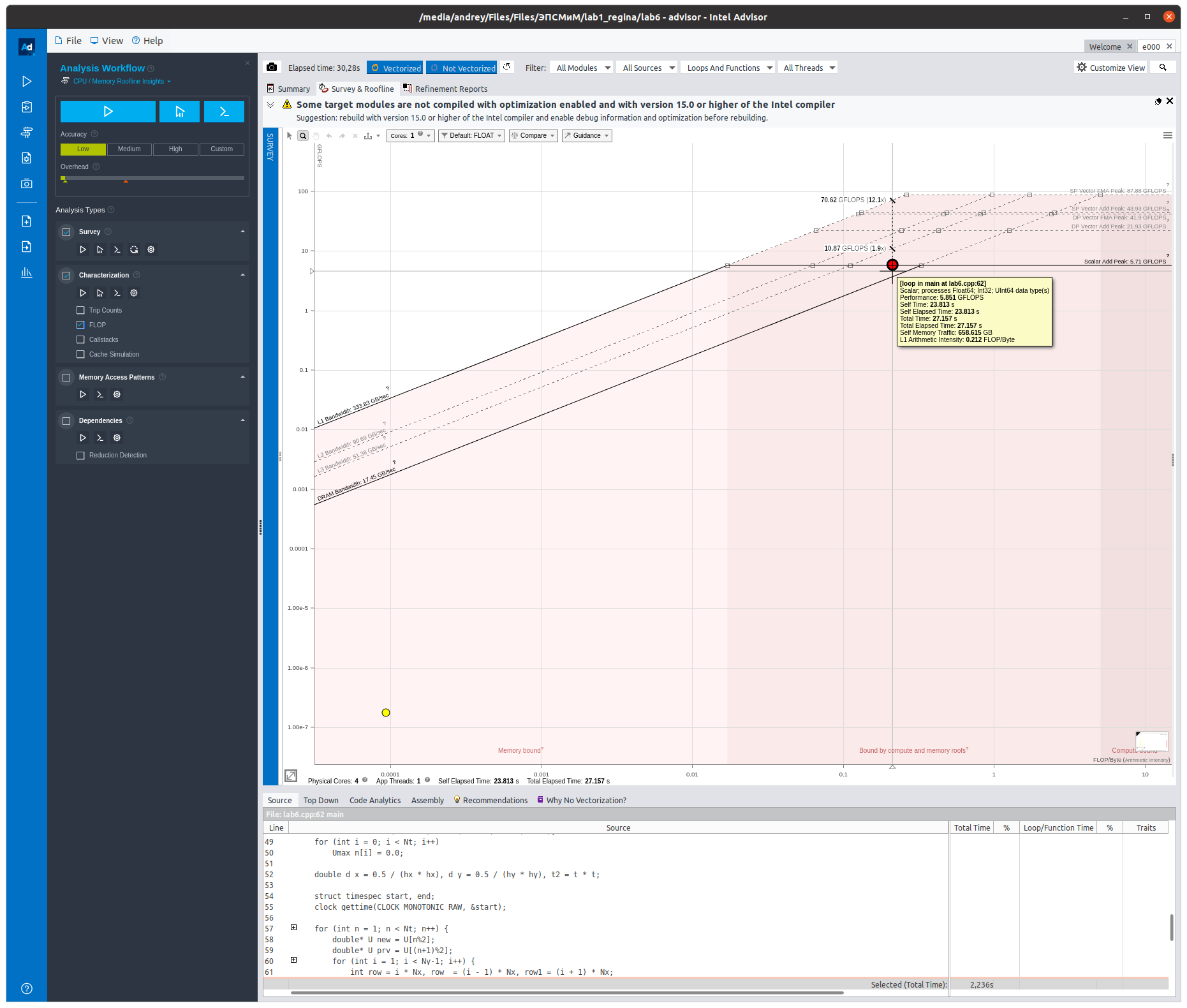
**Граф вызовов**

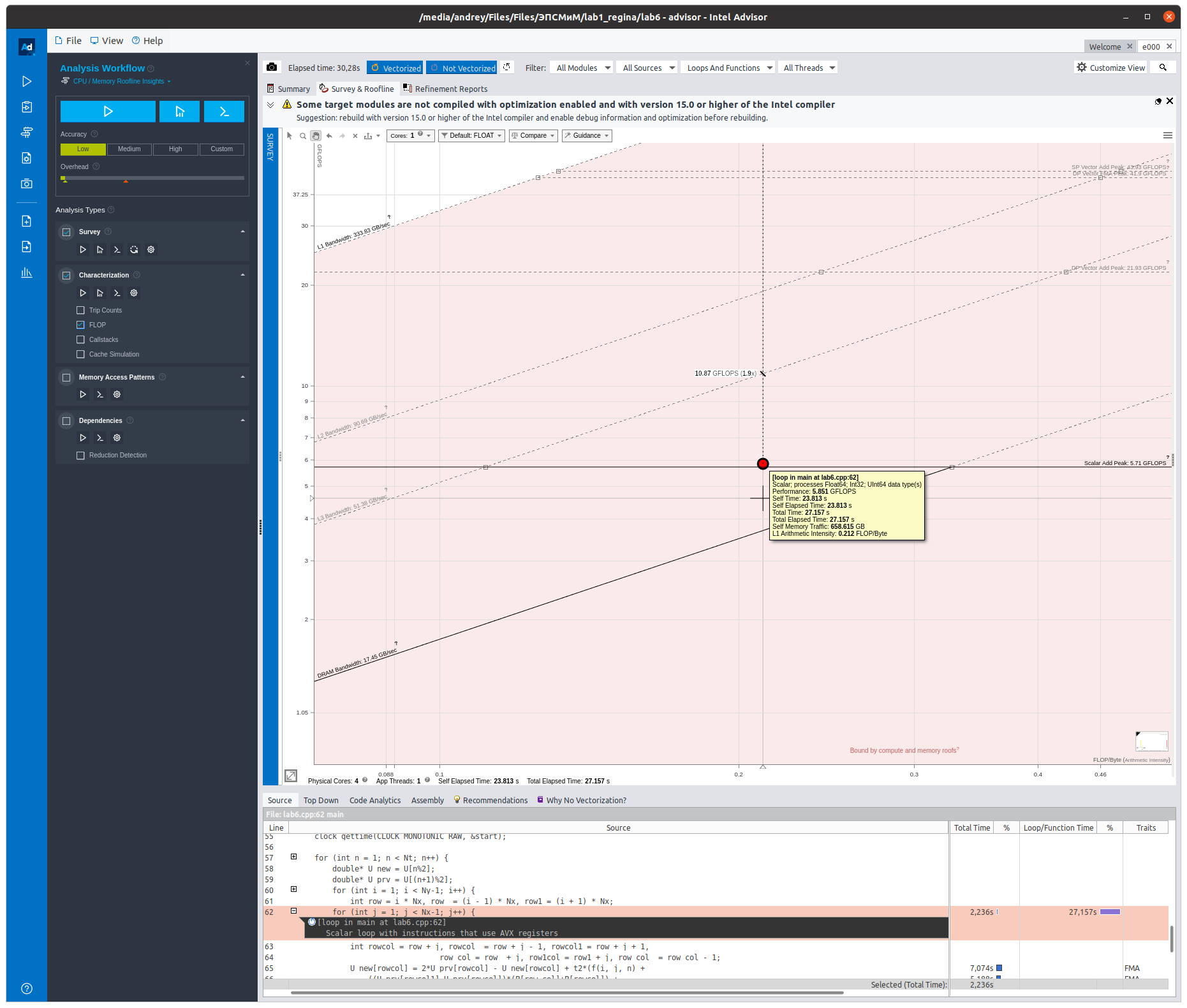


**Профилирование Vtune**



**Roofline-модель с точкой, соответствующей программе**





**Начальная версия программы**

#include <iostream>

#include <math.h>

#include <cstdlib>

#include <ctime>

#include <stdio.h>

using namespace std;

int Nx = 8000, Ny = 8000, Nt = 100;

int Sx = 1, Sy = 1;

double t;

double f (int i, int j, int n) {

if (j == Sx && i == Sy) {

double f0 = 1.0, t0 = 1.5, c = 4.0;

return exp(-1 \* ((2\*3.14\*f0\*(n\*t - t0))\*(2\*3.14\*f0\*(n\*t - t0))) / (c\*c)) \* sin(2\*3.14\*f0\*(n\*t - t0));

}

else

return 0.0;

}

int main(int argc, char\*\* argv) {

double Xa = 0.0, Ya = 0.0, Xb = 4.0, Yb = 4.0;

double hx = (Xb - Xa) / (Nx - 1), hy = (Yb - Ya) / (Ny - 1);

if (Nx <= 1000 && Ny <= 1000)

t = 0.01;

else

t = 0.001;

double\* P = (double\*)malloc(sizeof(double) \* Nx \* Ny);

for (int i = 0; i < Ny; i++)

for (int j = 0; j < Nx; j++) {

if (j < Nx/2)

P[i\*Nx + j] = 0.1\*0.1;

else

P[i\*Nx + j] = 0.2\*0.2;

}

double\*\* U = (double\*\*)malloc(2 \* sizeof(double\*));

U[0] = (double\*)malloc(Nx \* Ny \* sizeof(double));

U[1] = (double\*)malloc(Nx \* Ny \* sizeof(double));

for (int i = 0; i < Ny; i++)

for (int j = 0; j < Nx; j++) {

U[0][i\*Nx + j] = 0.0;

U[1][i\*Nx + j] = 0.0;

}

double\* Umax\_n = (double\*)malloc(sizeof(double) \* Nt);

for (int i = 0; i < Nt; i++)

Umax\_n[i] = 0.0;

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

for (int n = 1; n < Nt; n++) {

for (int i = 1; i < Ny-1; i++) {

for (int j = 1; j < Nx-1; j++) {

U[n%2][i\*Nx + j] = 2\*U[(n+1)%2][i\*Nx + j] - U[n%2][i\*Nx + j] + t\*t\*(f(i, j, n) +

((U[(n+1)%2][i\*Nx + j+1]-U[(n+1)%2][i\*Nx + j])\*(P[(i-1)\*Nx + j]+P[i\*Nx + j]) +

(U[(n+1)%2][i\*Nx + j-1]-U[(n+1)%2][i\*Nx + j])\*(P[(i-1)\*Nx + j-1]+P[i\*Nx + j-1]) ) / (2\*hx\*hx) +

((U[(n+1)%2][(i+1)\*Nx + j]-U[(n+1)%2][i\*Nx + j])\*(P[i\*Nx + j-1]+P[i\*Nx + j]) +

(U[(n+1)%2][(i-1)\*Nx+j]-U[(n+1)%2][i\*Nx+j])\*(P[(i-1)\*Nx + j-1]+P[(i-1)\*Nx + j]) ) / (2\*hy\*hy));

if (fabs(U[n%2][i\*Nx + j]) > Umax\_n[n])

Umax\_n[n] = fabs(U[n%2][i\*Nx + j]);

}

}

}

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

double time = end.tv\_sec-start.tv\_sec + 0.000000001\*(end.tv\_nsec-start.tv\_nsec);

printf("%lf\n", time);

FILE\* file = fopen("file.dat", "wb");

fwrite(U[1], sizeof(double), Nx \* Ny, file);

return 0;

}

**Оптимизированная версия программы**

#include <iostream>

#include <math.h>

#include <cstdlib>

#include <ctime>

#include <stdio.h>

using namespace std;

int Nx = 8000, Ny = 8000, Nt = 100;

int Sx = 1, Sy = 1;

double t;

double f (int i, int j, int n) {

if (j == Sx && i == Sy) {

double f0 = 1.0, t0 = 1.5, c = 4.0; // 1/c^2 = 0.0625

double arg = 2\*3.14\*f0\*(n\*t - t0);

return exp(-0.0625 \* (arg\*arg)) \* sin(arg);

}

else

return 0.0;

}

int main(int argc, char\*\* argv) {

double Xa = 0.0, Ya = 0.0, Xb = 4.0, Yb = 4.0;

double hx = (Xb - Xa) / (Nx - 1), hy = (Yb - Ya) / (Ny - 1);

if (Nx <= 1000 && Ny <= 1000)

t = 0.01;

else

t = 0.001;

double\* P = (double\*)malloc(sizeof(double) \* Nx \* Ny);

for (int i = 0; i < Ny; i++)

for (int j = 0; j < Nx; j++) {

if (j < Nx/2)

P[i\*Nx + j] = 0.1\*0.1;

else

P[i\*Nx + j] = 0.2\*0.2;

}

double\*\* U = (double\*\*)malloc(2 \* sizeof(double\*));

U[0] = (double\*)malloc(Nx \* Ny \* sizeof(double));

U[1] = (double\*)malloc(Nx \* Ny \* sizeof(double));

for (int i = 0; i < Ny; i++)

for (int j = 0; j < Nx; j++) {

U[0][i\*Nx + j] = 0.0;

U[1][i\*Nx + j] = 0.0;

}

double\* Umax\_n = (double\*)malloc(sizeof(double) \* Nt);

for (int i = 0; i < Nt; i++)

Umax\_n[i] = 0.0;

double d\_x = 0.5 / (hx \* hx), d\_y = 0.5 / (hy \* hy), t2 = t \* t;

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

for (int n = 1; n < Nt; n++) {

double\* U\_new = U[n%2];

double\* U\_prv = U[(n+1)%2];

for (int i = 1; i < Ny-1; i++) {

int row = i \* Nx, row\_ = (i - 1) \* Nx, row1 = (i + 1) \* Nx;

for (int j = 1; j < Nx-1; j++) {

int rowcol = row + j, rowcol\_ = row + j - 1, rowcol1 = row + j + 1,

row\_col = row\_ + j, row1col = row1 + j, row\_col\_ = row\_col - 1;

U\_new[rowcol] = 2\*U\_prv[rowcol] - U\_new[rowcol] + t2\*(f(i, j, n) +

((U\_prv[rowcol1]-U\_prv[rowcol])\*(P[row\_col]+P[rowcol]) +

(U\_prv[rowcol\_]-U\_prv[rowcol])\*(P[row\_col\_]+P[rowcol\_])) \* d\_x +

((U\_prv[row1col]-U\_prv[rowcol])\*(P[rowcol\_]+P[rowcol]) +

(U\_prv[row\_col]-U\_prv[rowcol])\*(P[row\_col\_]+P[row\_col])) \* d\_y);

if (fabs(U\_new[rowcol]) > Umax\_n[n])

Umax\_n[n] = fabs(U\_new[rowcol]);

}

}

}

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

double time = end.tv\_sec-start.tv\_sec + 0.000000001\*(end.tv\_nsec-start.tv\_nsec);

printf("%lf\n", time);

FILE\* file = fopen("file.dat", "wb");

fwrite(U[1], sizeof(double), Nx \* Ny, file);

return 0;

}