Universidade do Minho
Escola de Engenharia

TRABALHO PRÁTICO DE COMPUTAÇÃO PARALELA

Mestrado em Engenharia Informática

3DFLUID

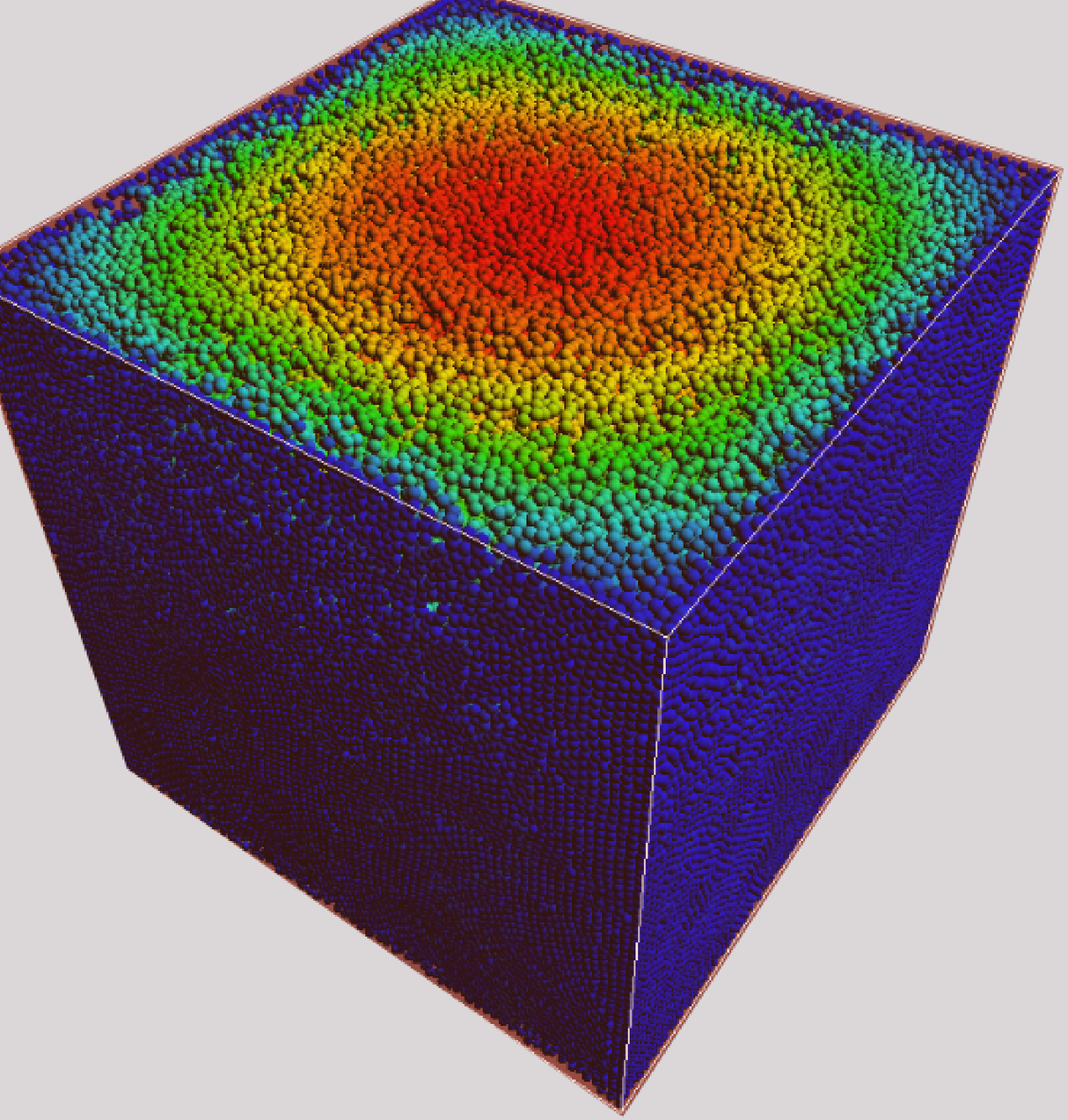
Grupo

Rúben Silva pg57900

Pedro Oliveira pg55093

Henrique Faria a91637

2025-01-08



3dfluid

- Sequencial
- OpenMP
- Cuda

Fase 1 - Sequencial

instruction level parallelism

Hierarquia de Memória

Instruction Level Parallelism

-Ofast -funroll-all-loops -march=native -flto

- -Ofast
- -funroll-all-loops
- -march=native
- -flto
- -mavx
- ftree-vectorize

Hierarquia de Memória

```
void lin_solve(int M, int N, int O, int b, float *x, float *x0, float a,
               float c)
{
  float new_c = 1 / c;
  int tile_size = 6;
  for (int l = 0; l < LINEARSOLVERTIMES; l++)
  {
    for (int jj = 1; jj <= N; jj += tile_size)
    {
      int block_Limit_J = (jj + tile_size < N + 1) ? jj + tile_size : N + 1;
      for (int kk = 1; kk <= O; kk += tile_size)
      {
        int block_Limit_K = (kk + tile_size < O + 1) ? kk + tile_size : O + 1;
        for (int ii = 1; ii <= M; ii += tile_size)
        {
          int block_Limit_I = (ii + tile_size < M + 1) ? ii + tile_size : M + 1;
          for (int j = jj; j < block_Limit_J; j++)
          {
            for (int k = kk; k < block_Limit_K; k++)
            {
              for (int i = ii; i < block_Limit_I; i++)
              {
                x[IX(i, j, k)] = (x0[IX(i, j, k)] +
                                  a * (x[IX(i - 1, j, k)] + x[IX(i + 1, j, k)] +
                                       x[IX(i, j - 1, k)] + x[IX(i, j + 1, k)] +
                                       x[IX(i, j, k - 1)] + x[IX(i, j, k + 1)])) *
                                  new_c;
              }
            }
          }
        }
      }
    }
    set_bnd(M, N, O, b, x);
  }
}
```

Resultados

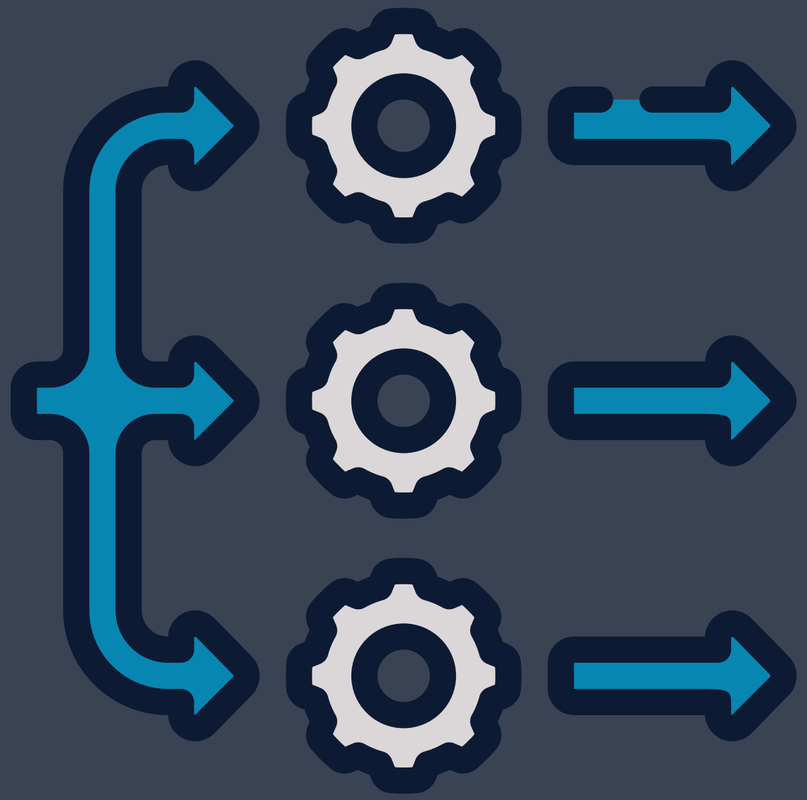
Sem hierarquia

Métricas	No Flags	-O2 -FRLoop -FtrVect -mavx	-O3 -FRLoop -FtrVect -mavx	-Ofast -FRLoop -MNative -flto
Instructions	166613827657	16291008681	15645802313	16671358346
Cycles	94417810364	43400774402	34525319160	22778985010
L1dcLM	2420130899	2846690855	2594925840	2530349767
Cache-Misses	1026	1273	6025	1276
Cache-Reference	1336375896	2420720487	1287953269	1205093828
CPI	0,567	2,664	2,207	1,366
Run-Time (ms)	31.2821	13.5709	10.7519	7.072
Result	81981.3	81981.3	81981.3	81981.5

Com hierarquia

Métricas	No Flags	-O2 -FRLoop -FtrVect -mavx	-O3 -FRLoop -FtrVect -mavx	-Ofast -FRLoop -MNative -flto
Instructions	168117489688	19328951330	17286351975	18894376660
Cycles	63205633692	21383833040	20042667776	12453730302
L1dcLM	243535495	244202794	243172253	245089457
Cache-Misses	737	859	3041	724
Cache-Reference	97980725	100185792	98714473	98948982
CPI	0.376	1,106	1.159	0,659
Run-Time (ms)	19.6830	6.5753	6.09	3.62
Result	81981.2	81981.2	81981.2	81981.5

Fase 2 - OpenMP

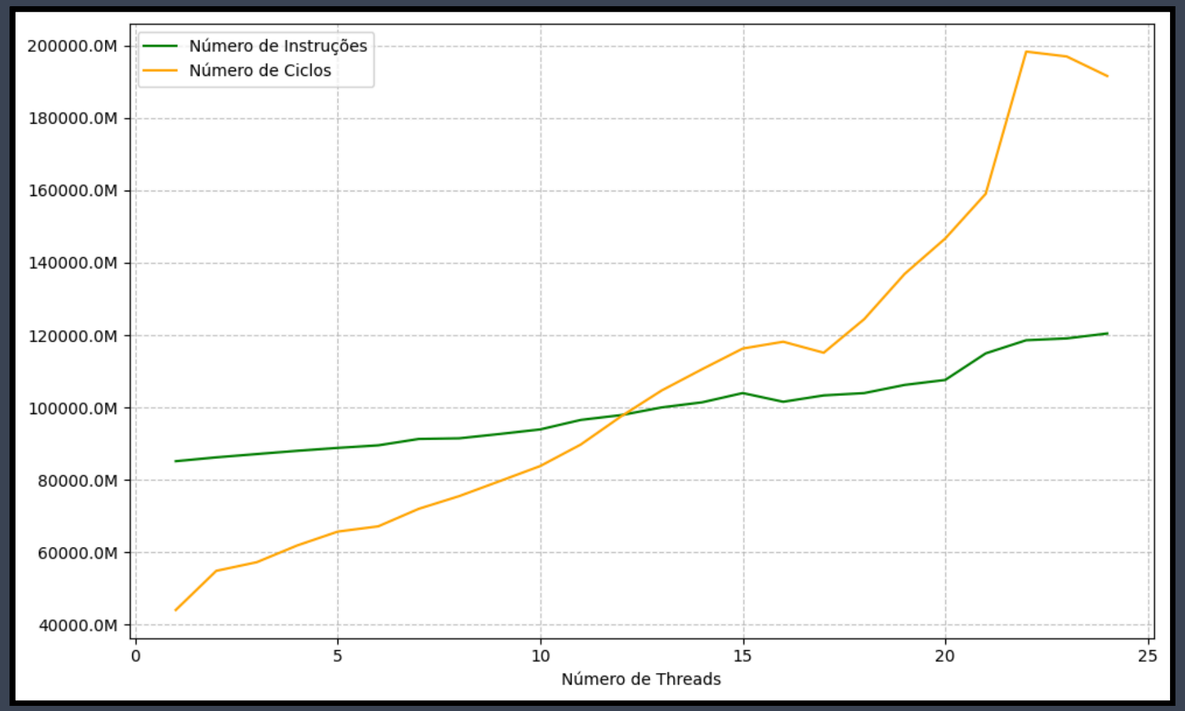


OpenMP

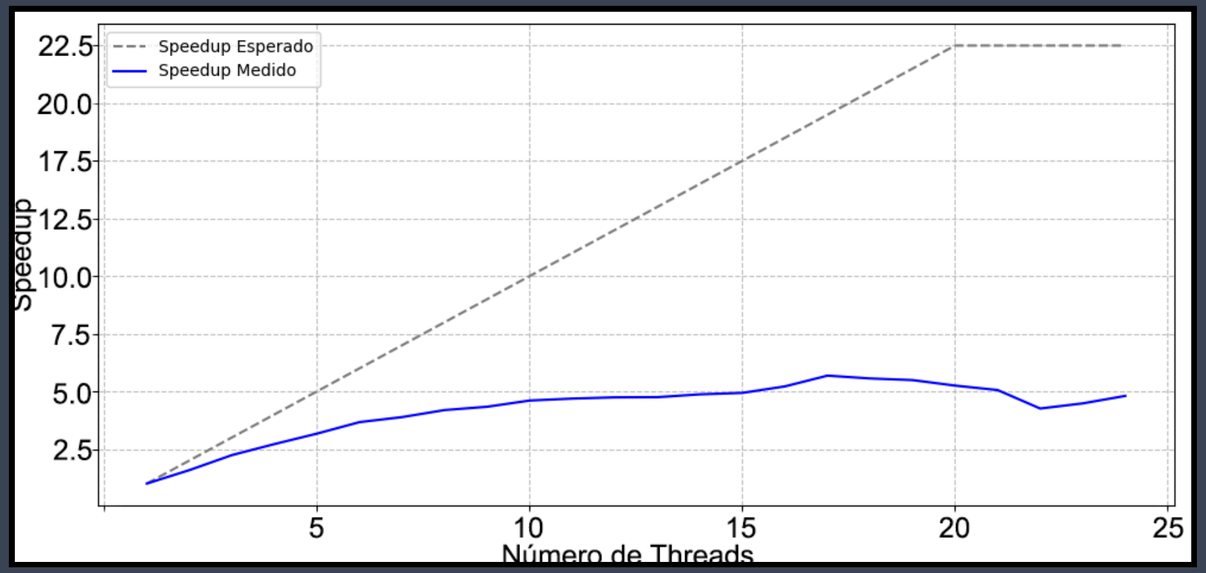
```
void lin_solve(int M, int N, int O, int b, float *x, float *x0, float a, float c)
  float tol = 1e-7, max_c, old_x, change;
  int l = 0;
  float new_c = 1 / c;
  do
  {
    max_c = 0.0f;
#pragma omp parallel reduction(max : max_c)
    {
#pragma omp for schedule(static) private(old_x, change)
      for (int k = 1; k <= O; k++)
      {
        for (int j = 1; j <= N; j++)
        {
          for (int i = 1 + (k + j) % 2; i <= M; i += 2)
          {
            old_x = x[IX(i, j, k)];
            x[IX(i, j, k)] = (x0[IX(i, j, k)] +
                              a * (x[IX(i - 1, j, k)] + x[IX(i + 1, j, k)] +
                                   x[IX(i, j - 1, k)] + x[IX(i, j + 1, k)] +
                                   x[IX(i, j, k - 1)] + x[IX(i, j, k + 1)])) *
                             new_c;
            change = fabs(x[IX(i, j, k)] - old_x);
            if (change > max_c)
              max_c = change;
          }
        }
      }
#pragma omp for schedule(static) private(old_x, change)
      for (int k = 1; k <= O; k++)
      {
        for (int j = 1; j <= N; j++)
        {
          for (int i = 1 + (k + j + 1) % 2; i <= M; i += 2)
          {
            old_x = x[IX(i, j, k)];
            x[IX(i, j, k)] = (x0[IX(i, j, k)] +
                              a * (x[IX(i - 1, j, k)] + x[IX(i + 1, j, k)] +
                                   x[IX(i, j - 1, k)] + x[IX(i, j + 1, k)] +
                                   x[IX(i, j, k - 1)] + x[IX(i, j, k + 1)])) *
                             new_c;
            change = fabs(x[IX(i, j, k)] - old_x);
            if (change > max_c)
              max_c = change;
          }
        }
      }
    }
    set_bnd(M, N, O, b, x);
  } while (max_c > tol && ++l < 20);
}
```

Resultados

Ciclos/Instruções



Speedup

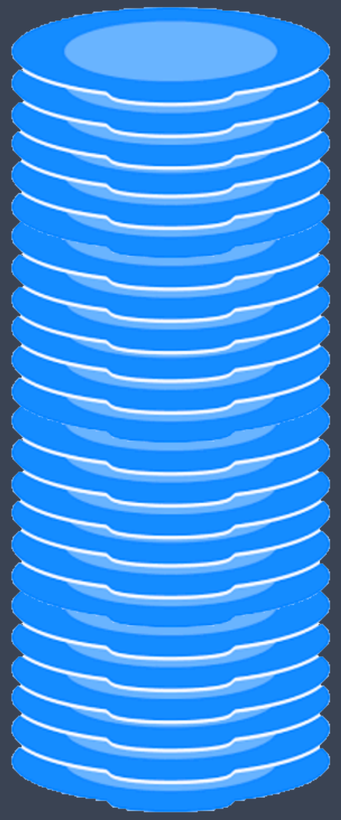


Fase 3 - CUDA

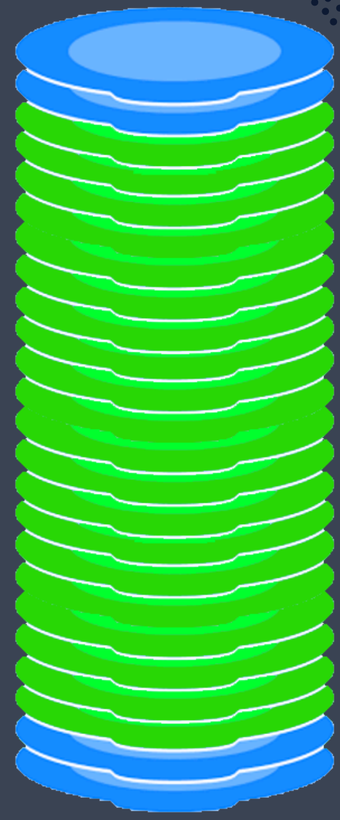
CUDA

CUDA

CPU



CPU



CUDA

CUDA

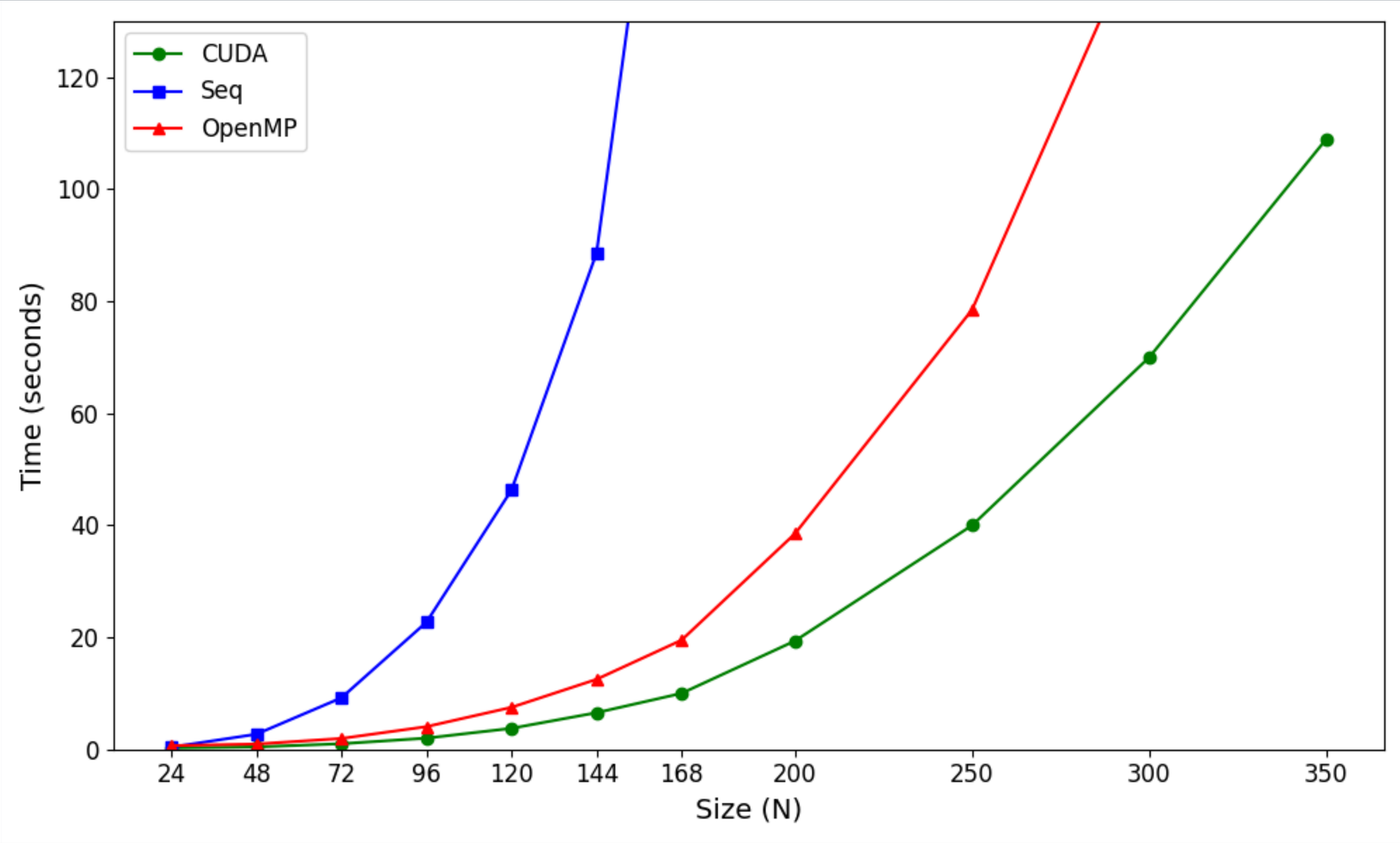
```cpp
__global__ void lin_solve_kernel(int M, int N, int O, float *x, float *x0,
                                 float a, float c, int phase, float *max_change)
{
    int tx = threadIdx.x;
    int ty = threadIdx.y;
    int tz = threadIdx.z;
    int j = blockIdx.y * (blockDim.y - 2) + ty;
    int k = blockIdx.z * (blockDim.z - 2) + tz;
    int i = 2 * (blockIdx.x * (blockDim.x - 2) + tx) + ((j + k) % 2 == phase ? 0 : 1);
    if (tx > 0 && tx < blockDim.x -1 &&
        ty > 0 && ty < blockDim.y -1 &&
        tz > 0 && tz < blockDim.z -1 &&
        i <= M && j <= N && k <= O)
    {
        float old_x = x[IX(i, j, k)];
        float new_x = (x0[IX(i, j, k)] +
                       a * (x[IX(i-1, j, k)] + x[IX(i+1, j, k)] +
                            x[IX(i, j-1, k)] + x[IX(i, j+1, k)] +
                            x[IX(i, j, k-1)] + x[IX(i, j, k+1)])) / c;

        x[IX(i, j, k)] = new_x;
        float change = fabsf(new_x - old_x);
        if (change > *max_change)
        {
            atomicMax((int *)max_change, __float_as_int(change));
        }
    }
}
void lin_solve(int M, int N, int O, int b, float *x, float *x0, float a, float c)
{
    dim3 block(BLOCK - 1, BLOCK - 1, BLOCK - 1);
    dim3 grid(((M/2) + block.x - 1) / block.x,
              (N + block.y - 1) / block.y,
              (O + block.z - 1) / block.z);
    float *max_change_dev;
    cudaMalloc(&max_change_dev, sizeof(float));
    float max_change_host;
    int iter = 0;
    float tol = 1e-7f;
    do
    {
        float init_max = 0.0f;
        cudaMemcpy(max_change_dev, &init_max, sizeof(float), cudaMemcpyHostToDevice);
        lin_solve_kernel<<<grid, block>>>(M, N, O, x, x0, a, c, 0, max_change_dev);
        lin_solve_kernel<<<grid, block>>>(M, N, O, x, x0, a, c, 1, max_change_dev);
        set_bnd_kernel<<<(MAX(MAX(M, N), O) + 255) / 256, 256>>>(M, N, O, b, x);
        cudaMemcpy(&max_change_host, max_change_dev, sizeof(float), cudaMemcpyDeviceToHost);
        iter++;
    } while (max_change_host > tol && iter < LINEARSOLVERTIMES);
    cudaFree(max_change_dev);
}
```

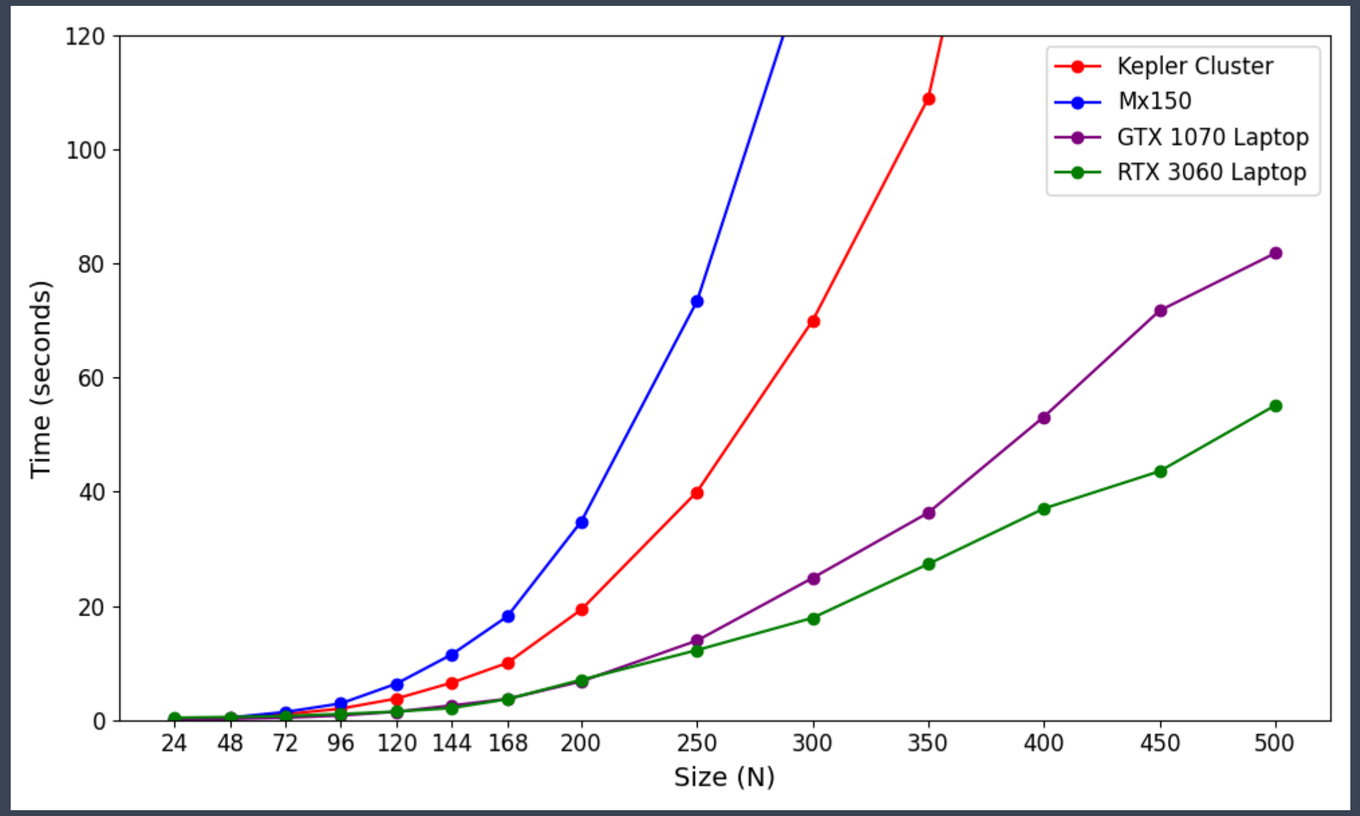
nvprof

```
==15418== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:   75.70%  7.49883s     14254  526.09us  520.36us  553.19us  lin_solve_kernel(int
                    8.62%  854.15ms       400  2.1354ms  2.1139ms  2.1879ms  advectKernel(int, in
                    5.67%  562.00ms       200  2.8100ms  2.7956ms  2.8280ms  project_kernel_2(int
                    3.26%  323.18ms       400  807.94us  786.25us  833.58us  addSourceKernel(int,
                    3.06%  302.92ms       200  1.5146ms  1.5044ms  1.5210ms  project_kernel_1(int
                    2.52%  249.19ms      8527  29.223us  14.785us  108.35us  set_bnd_kernel(int,
                    0.60%  59.252ms      7135  8.3040us     863ns  6.7061ms  [CUDA memcpy HtoD]
                    0.57%  56.846ms      7135  7.9670us  1.2480us  6.0435ms  [CUDA memcpy DtoH]
                    0.00%  73.569us        20  3.6780us  3.3280us  4.8000us  apply_events_kernel(
nt, int)
      API calls:   94.77%  9.94035s     14270  696.59us  5.8540us  11.654ms  cudaMemcpy
                    2.80%  293.86ms       608  483.33us  110.56us  216.72ms  cudaMalloc
                    1.74%  182.56ms     24001  7.6060us  5.5690us  627.62us  cudaLaunchKernel
                    0.68%  71.188ms       608  117.09us  83.200us  2.0474ms  cudaFree
                    0.01%  722.06us         1  722.06us  722.06us  722.06us  cuDeviceTotalMem
                    0.00%  385.77us       101  3.8190us     360ns  152.87us  cuDeviceGetAttribute
                    0.00%  32.913us         1  32.913us  32.913us  32.913us  cuDeviceGetName
                    0.00%  20.003us         1  20.003us  20.003us  20.003us  cuDeviceGetPCIBusId
                    0.00%  2.9130us         3     971ns     426ns  1.6750us  cuDeviceGetCount
                    0.00%  1.7560us         2     878ns     436ns  1.3200us  cuDeviceGet
                    0.00%  1.0910us         1  1.0910us  1.0910us  1.0910us  cuDeviceGetUuid
Simulation took CPU 10 seconds.
Total density after 100 timesteps: 160832
```

Resultado Final



Testes

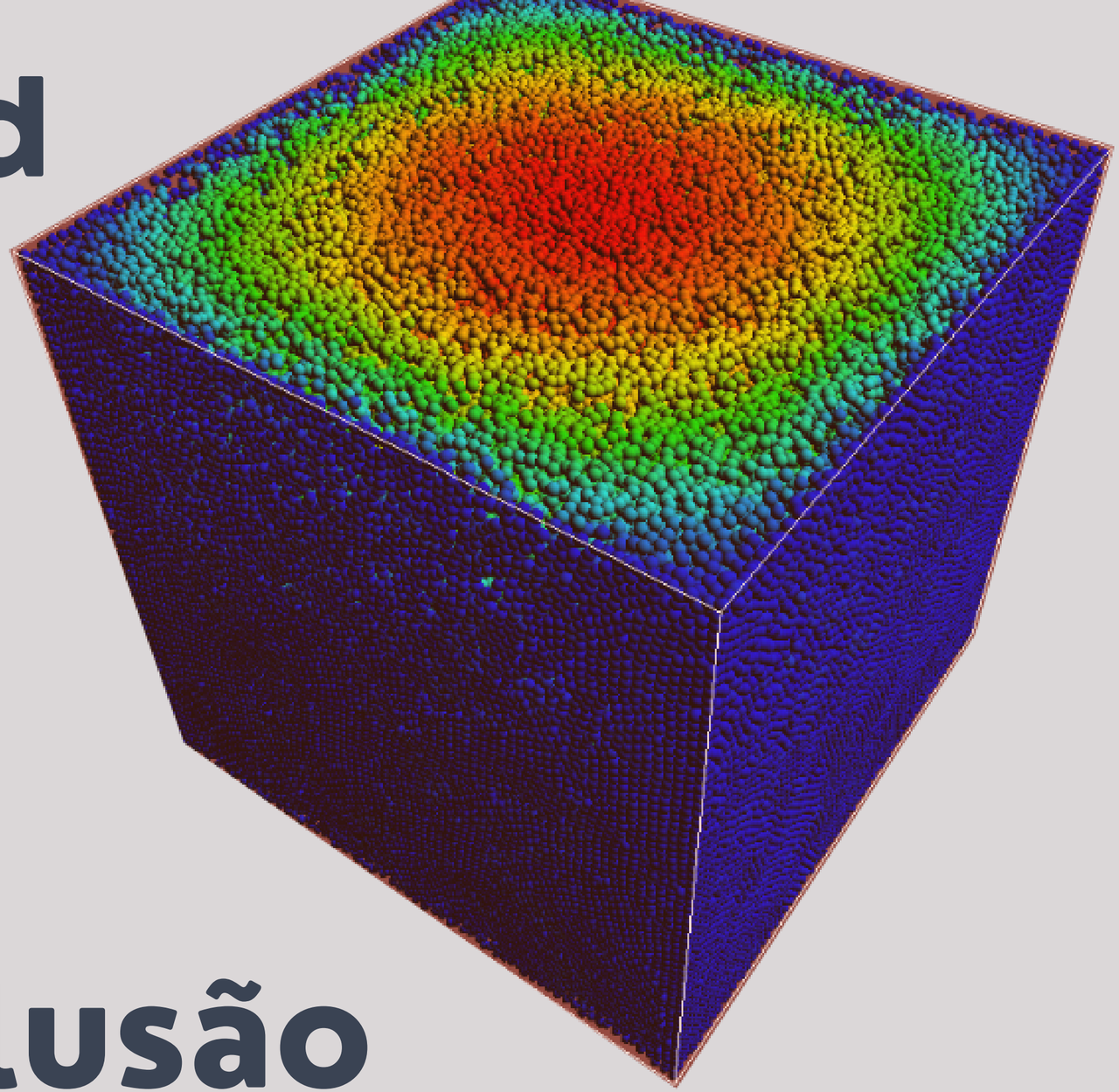


Benchmark GPUs

Threads por Bloco

BLOCK	Threads por bloco	Resultado	Runtime (s)
7	343	Incorreto	N/A
8	512	Correto	10.1
9	729	Correto	11
10	1.000	Correto	13.2
11	1.331	Erro	N/A

3dfluid



Conclusão

Universidade do Minho
Escola de Engenharia

TRABALHO PRÁTICO DE COMPUTAÇÃO PARALELA

Mestrado em Engenharia Informática

3DFLUID

Grupo

Rúben Silva pg57900

Pedro Oliveira pg55093

Henrique Faria a91637

2025-01-08