COC472 - Computação de Alto Desempenho

Terceiro Trabalho

Universidade Federal
do Rio de Janeiro

- Pedro Tubenchlak Boechat - DRE 119065050
  - pedroboechat@poli.ufrj.br

O código-fonte desse trabalho está disponível no meu GitHub
(https://github.com/pedroboechat/UFRJ/tree/main/COC472/Trabalho%203).

Rio de Janeiro - RJ
3 de Outubro de 2021 (2021.1)

Introdução

O código *laplace.cxx* foi alterado de forma que o valor de *nx* seja definido via *argv* e os valores de n_{iter} e *eps* foram fixados. Além disso, também foi alterado o formato de output para "RESULTADO;TEMPO". Baseado nos dados do Trabalho 2, temos que os hotspots do programa são as funções *LaplaceSolver::timeStep* e *SQR* e, portanto, os loops mais intensivos são os dois contidos na função *LaplaceSolver::timeStep*.

Modificações com o OpenMP

Foi criada uma cópia do código *laplace.cxx*, chamada *laplace_omp.cxx*, onde se foi adicionada a biblioteca <omp.h>. Nela, à função *LaplaceSolver::timeStep* foi adicionada a seguinte diretiva de OpenMP:

```
#pragma omp parallel for private(tmp, i, j) shared(u, dx2, dy2) reduction(+ : err)
```

Essa diretiva:

- Adiciona paralelismo para os *for-loops*;
- Utiliza variáveis privadas nas variáveis de iteração (*i, j, tmp*), que devem ser únicas para cada thread;
- Utiliza variáveis compartilhadas nas variáveis da função (*u, dx2, dy2*), que devem ser compartilhadas pelos threads;
- Adiciona uma redução para realizar a soma na variável *err* no final da região paralela.

Resultados

Foi criado o código Python *run.py*, que compila os arquivos com as flags necessárias e coleta o desempenho para diferentes configurações, ao fim salvando esses dados a um arquivo *run.csv*. Após rodar esse programa numa máquina com um processador Intel i7-10700 (com 16 threads) e sistema operacional Ubuntu 20.04.3 LTS, temos como resultados:

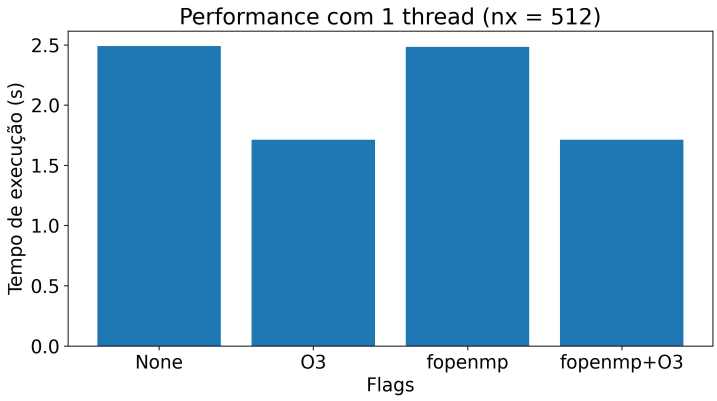
Flags	Threads	*nx*	Result	Elapsed (s)
None	1	512	0.0402737	2.49003
None	1	1024	0.0578281	10.2353
None	1	2048	0.0820855	46.3686
O3	1	512	0.0402737	1.71105
O3	1	1024	0.0578281	7.16366
O3	1	2048	0.0820855	33.1205
fopenmp	1	512	0.0402737	2.48315
fopenmp	1	1024	0.0578281	10.2418
fopenmp	1	2048	0.0820855	45.806
fopenmp	2	512	0.0402737	1.25295
fopenmp	2	1024	0.0578281	5.26486
fopenmp	2	2048	0.0820855	24.4493
fopenmp	3	512	0.0402737	0.865536
fopenmp	3	1024	0.0578281	3.64513
fopenmp	3	2048	0.0820855	17.7402
fopenmp	4	512	0.0402737	0.639756
fopenmp	4	1024	0.0578281	2.75869
fopenmp	4	2048	0.0820855	13.9678
fopenmp	5	512	0.0402737	0.519615
fopenmp	5	1024	0.0578281	2.22806
fopenmp	5	2048	0.0820855	11.5657

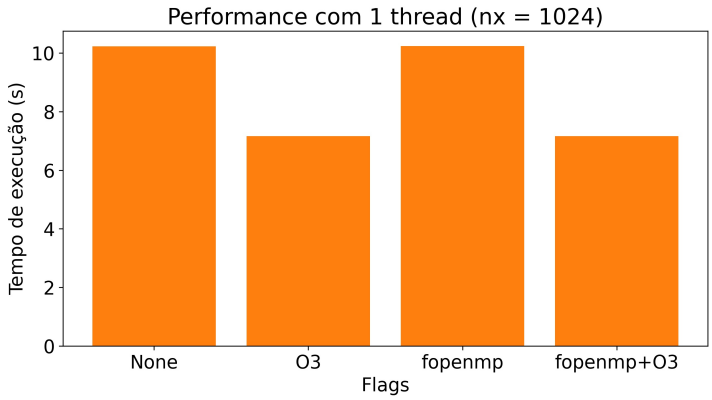
Flags	Threads	nx	Result	Elapsed (s)
fopenmp	6	512	0.0402737	0.436
fopenmp	6	1024	0.0578281	1.89229
fopenmp	6	2048	0.0820855	10.1939
fopenmp	7	512	0.0402739	0.419684
fopenmp	7	1024	0.0578281	1.84816
fopenmp	7	2048	0.0820855	9.74496
fopenmp	8	512	0.0402742	0.430951
fopenmp	8	1024	0.0578281	1.71465
fopenmp	8	2048	0.0820855	9.06975
fopenmp	9	512	0.0402744	0.524902
fopenmp	9	1024	0.0578282	2.13662
fopenmp	9	2048	0.0820855	9.44107
fopenmp	10	512	0.0402744	0.488344
fopenmp	10	1024	0.0578282	2.06291
fopenmp	10	2048	0.0820855	8.784
fopenmp	11	512	0.0402742	0.453176
fopenmp	11	1024	0.0578282	1.90241
fopenmp	11	2048	0.0820855	9.1597
fopenmp	12	512	0.0402738	0.422209
fopenmp	12	1024	0.0578282	1.81556
fopenmp	12	2048	0.0820855	8.24339
fopenmp	13	512	0.0402735	0.395625
fopenmp	13	1024	0.0578283	1.67951
fopenmp	13	2048	0.0820855	8.07432
fopenmp	14	512	0.040273	0.371718
fopenmp	14	1024	0.0578285	1.59355
fopenmp	14	2048	0.0820855	7.90296
fopenmp	15	512	0.0402728	0.348993
fopenmp	15	1024	0.0578287	1.52667
fopenmp	15	2048	0.0820855	8.06759
fopenmp	16	512	0.0402721	0.366673
fopenmp	16	1024	0.0578289	1.50596
fopenmp	16	2048	0.0820855	7.76724
fopenmp+O3	1	512	0.0402737	1.711
fopenmp+O3	1	1024	0.0578281	7.16758
fopenmp+O3	1	2048	0.0820855	33.1313
fopenmp+O3	2	512	0.0402737	0.862398
fopenmp+O3	2	1024	0.0578281	3.69996
fopenmp+O3	2	2048	0.0820855	17.65
fopenmp+O3	3	512	0.0402737	0.581425
fopenmp+O3	3	1024	0.0578281	2.54402
fopenmp+O3	3	2048	0.0820855	13.0349
fopenmp+O3	4	512	0.0402737	0.440296
fopenmp+O3	4	1024	0.0578281	1.94769
fopenmp+O3	4	2048	0.0820855	10.4107
fopenmp+O3	5	512	0.0402737	0.352063
fopenmp+O3	5	1024	0.0578281	1.57555
fopenmp+O3	5	2048	0.0820855	8.67275
fopenmp+O3	6	512	0.0402737	0.294171
fopenmp+O3	6	1024	0.0578281	1.33321
fopenmp+O3	6	2048	0.0820855	7.44909
fopenmp+O3	7	512	0.0402739	0.254115
fopenmp+O3	7	1024	0.0578281	1.16022
fopenmp+O3	7	2048	0.0820855	6.76792

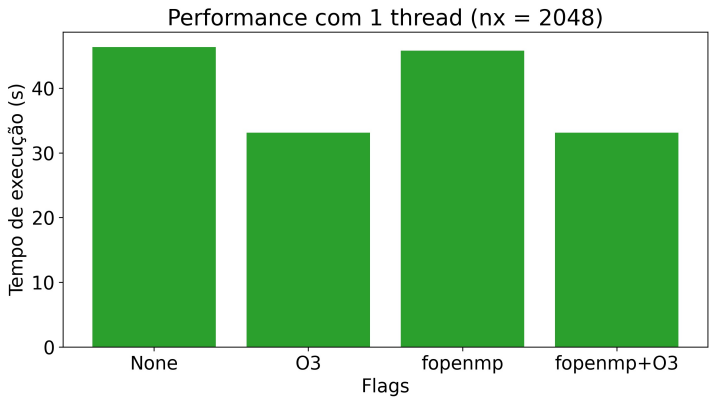
Flags	Threads	nx	Result	Elapsed (s)
fopenmp+O3	8	512	0.0402742	0.224113
fopenmp+O3	8	1024	0.0578281	1.03682
fopenmp+O3	8	2048	0.0820855	6.09922
fopenmp+O3	9	512	0.0402744	0.200494
fopenmp+O3	9	1024	0.0578282	0.935963
fopenmp+O3	9	2048	0.0820855	5.62835
fopenmp+O3	10	512	0.0402744	0.18072
fopenmp+O3	10	1024	0.0578282	0.860797
fopenmp+O3	10	2048	0.0820855	5.48426
fopenmp+O3	11	512	0.0402742	0.178782
fopenmp+O3	11	1024	0.0578282	0.840245
fopenmp+O3	11	2048	0.0820855	5.25915
fopenmp+O3	12	512	0.0402738	0.167027
fopenmp+O3	12	1024	0.0578282	0.786469
fopenmp+O3	12	2048	0.0820855	4.97736
fopenmp+O3	13	512	0.0402735	0.156356
fopenmp+O3	13	1024	0.0578283	0.740689
fopenmp+O3	13	2048	0.0820855	4.80849
fopenmp+O3	14	512	0.040273	0.146801
fopenmp+O3	14	1024	0.0578285	0.697875
fopenmp+O3	14	2048	0.0820855	4.62652
fopenmp+O3	15	512	0.0402728	0.138228
fopenmp+O3	15	1024	0.0578287	0.664036
fopenmp+O3	15	2048	0.0820855	4.49677
fopenmp+O3	16	512	0.0402721	0.155496
fopenmp+O3	16	1024	0.0578289	0.665205
fopenmp+O3	16	2048	0.0820855	4.46001

Comparação de desempenho

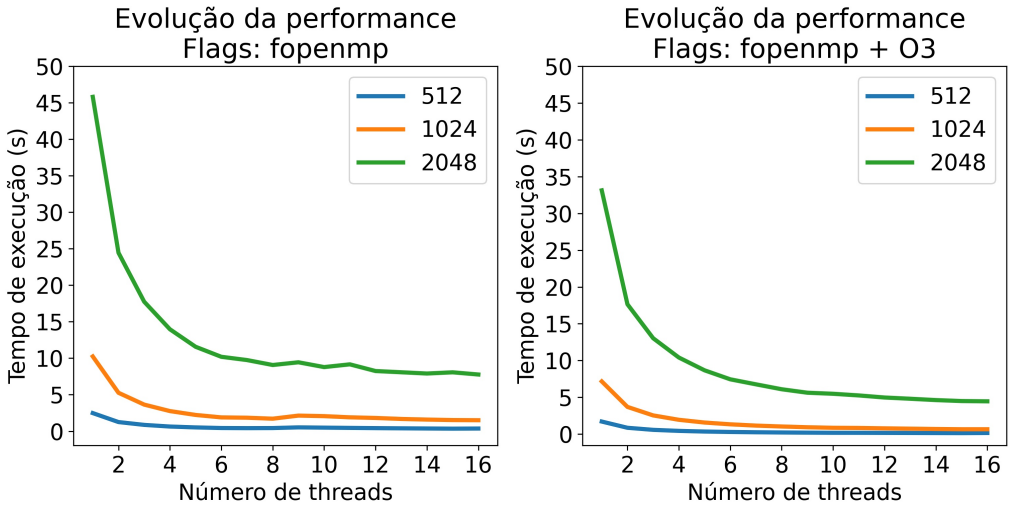
Após análise desses dados, foi constatado que a performance single thread varia significativamente apenas na adição da flag -O3.







Na análise da performance multi thread, é possível ver a melhora de performance de, em média, 52% com 2 threads com a flag -fopenmp e 36% com as flags -fopenmp e -O3. A performance continua melhorando com a adição de mais threads, mas começa a convergir a partir de 6 threads.



Conclusão

Fica evidente, após a análise, que a flag -O3 é bastante importante na melhora de performance do código *laplace.cxx*, tanto single thread como multi thread. Além disso, a flag -fopenmp não afeta o desempenho single thread, mas o paralelismo implementado pelo OpenMP exibe grandes melhorias, mesmo adicionando apenas mais uma thread.