COC 472 - Computação de Alto Desempenho
Trabalho 4

Paulo Mattos - DRE: 116013884

1 Identificação de Loops

Com base nas analises realizadas nos ultimos trabalhos, foi possivel identificar que os loops aninhados da função ***timeStep*** são os mais intensivos. E a otimização utilizando o OpenMP vai focar nesses loops.

2 Modificações

A fim de otimizar o código foram empregadas algumas instruções do OpenMP para a paralelização dos loops em questão. As instruções e suas justificativas se encontam abaixo:

- **FIRSTPRIVATE**: As variáveis *dx2*, *dy2*, *hoistedConstant* foram marcadas com FIRSTPRIVATE, pois são compartilhadas entre as threads, mas as threads só lêem o valor dessas variáveis. Usar FIRSTPRIVATE no lugar de SHARED evita que seja necessário acessar as variaveis desrreferenciando elas através de um ponteiro, evitando conflitos de cache.
- **SHARED**: O unico recurso compartilhado de forma não readonly de cada thread é a matriz u. Logo esse foi o unico marcado como SHARED;
- **REDUCTION**: A variavel que guarda o resultado final do método, *err*, precisa sofrer redução ao final do loop.

Além das instruções do OpenMP o programa foi modificado de forma a simplificar sua execução por um script.

3 Resultados

Flags	Nx	Resultado	Tempo (s)	Threads
	512	0.0402737	1.86697	1
	1024	0.0578281	8.67536	1
	2048	0.0820855	37.3417	1
-O3	512	0.0402737	1.28899	1
-O3	1024	0.0578281	5.46925	1
-O3	2048	0.0820855	26.6981	1
-fopenmp	512	0.0402737	1.87943	1
-fopenmp	1024	0.0578281	8.14334	1
-fopenmp	2048	0.0820855	37.6996	1
-fopenmp	512	0.0402737	1.02944	2
-fopenmp	1024	0.0578281	4.63884	2
-fopenmp	2048	0.0820855	21.3474	2
-fopenmp	512	0.0402737	0.834794	3
-fopenmp	1024	0.0578281	3.64381	3
-fopenmp	2048	0.0820855	16.4683	3
-fopenmp	512	0.0402737	0.496859	4
-fopenmp	1024	0.0578281	2.24251	4
-fopenmp	2048	0.0820855	12.4285	4
-fopenmp, -O3	512	0.0402737	1.31094	1
-fopenmp, -O3	1024	0.0578281	5.57365	1
-fopenmp, -O3	2048	0.0820855	28.5756	1
-fopenmp, -O3	512	0.0402737	0.648694	2
-fopenmp, -O3	1024	0.0578281	2.90814	2
-fopenmp, -O3	2048	0.0820855	14.8733	2
-fopenmp, -O3	512	0.0402737	0.43531	3
-fopenmp, -O3	1024	0.0578281	2.20776	3
-fopenmp, -O3	2048	0.0820855	12.0895	3
-fopenmp, -O3	512	0.0402737	0.376588	4
-fopenmp, -O3	1024	0.0578281	1.79827	4
-fopenmp, -O3	2048	0.0820855	11.1755	4

4 Conclusão

Como é possivel observar na tabela, o código que contem as instruções do OpenMP apresenta um tempo de execução menor que o código original. É possivel observar também que o tempo de execução diminui com o aumento do numero de threads utilizados.

Em relação a escalabilidade, os resultados mostram que o desempenho do código aumenta com o numero de threads utilizado, mas tem uma tendencia decrescente.

5 Código

O código desenvolvido para esse trabalho pode ser acessado em GitHub | Paulo Mattos