Computação Paralela
3dfluid - 2º Fase

Rúben Silva
pg57900
Mestrado Eng. Informática

Pedro Oliveira
pg55093
Mestrado Eng. Informática

Henrique Faria
a91637
Externo

I. Introdução

Na segunda fase do Trabalho Prático, usamos *OpenMP* para explorar o paralelismo da memória e reduzir o tempo do simulador com oito vezes mais partículas.

II. Análise e Implementação

A. Análise

Após o aumento do **SIZE** de 42 para 84, substituiu-se o solver pelo o (*Red-Black*). Foi realizado o profiling com esta nova versão:

Overhead	Samples	Command	Shared Object	Symbol
79.56%	108824	fluid_sim	fluid_sim	[.] lin_solve(int, int,
14.72%	20078	fluid_sim	fluid_sim	[.] advect(int, int, in
2.86%	3904	fluid_sim	fluid_sim	[.] project(int, int, i
1.46%	2004	fluid_sim	fluid_sim	[.] main
0.80%	1096	fluid_sim	fluid_sim	[.] set_bnd(int, int, i
0.02%	28	fluid_sim	[unknown]	[k] 0xffffffff8ac011d3

Fig. 1: Profiling

Como mostrado na *Fig. 1*, a função *lin_solve* continua a ser o **Hot-Spot**, mas as outras funções teem de ser optimizadas.

B. Implementação

Inicialmente melhoramos a função *lin_solve* e para explorar o paralelismo na função começou-se por melhorar a *Localidade Espacial e Temporal* do novo solver, ajustando a ordem dos ciclos dos *"fors"* para (*O-N-M*).

Inicialmente exploramos de uma maneira básica o Paralelismo, mas para melhorar a escalabilidade, fizemos:

Os ciclos Red e Black foram inseridos numa secção **parallel** para minimizar o overhead do **Paralelismo**. Adicionou-se também a cláusula (**reduction(max: max_c)**), que permite calcular o valor máximo de forma segura entre múltiplas threads, assegurando que a comparação do valor máximo (max_c) ocorre sem erros de concorrência, minimizando assim o overhead das **Sincronizações**.

Para mitigar o **Load Imbalance**, aplicou-se a cláusula **schedule(static)** nos ciclos do algoritmo *Red-Black*. Esta configuração distribui as iterações dos ciclos entre as threads, ajustando a carga de trabalho em tempo real.

Nas outras funções, a *Localidade Espacial e Temporal* já estava bastante bem optimizada tendo em conta os valores da primeira fase. Tendo em conta isto, a exploração do paralelismo foi feita usando simplesmente **parallel for schedule(dynamic)** para todos os ciclos "*fors*".

Toda a implementação na versão final, é fruto de diversas tentativas diferentes de atingir a melhor performance.

C. Resultados

Os seguintes resultados foram obtidos no Cluster da Universidade do Minho, na partição de *cpar* que contém 20 PUs.

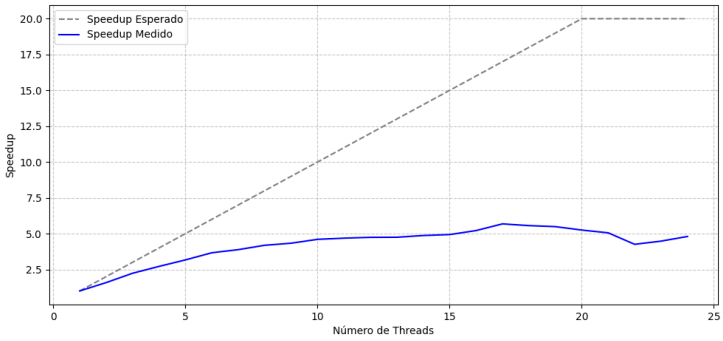

Fig. 2: Evolução do Speedup

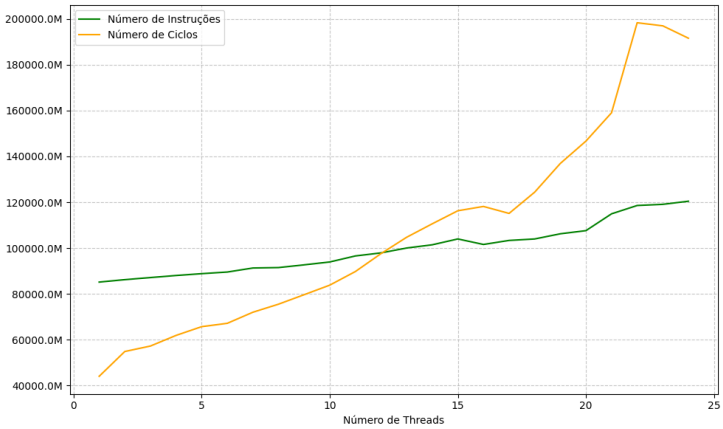

Fig. 3: Evolução do nº de Instruções e Ciclos

Foi obtida uma melhoria de $\approx$ 6x de Speedup. Também é notória o aumento acentuado do número de ciclos e um aumento mais ligeiro do número instruções.

A Versão Sequencial teve um runtime de $\approx$ 12.8s e a Versão Paralela com o melhor runtime de $\approx$ 2.35s.

Foi notado também que acima das 20 Threads, os valores dois gráficos saem fora do espectável. Isso, claramente consequência de a existência de só 20 PUs.