Universidade Federal de São João del-Rei
Departamento de Ciência da Computação
Curso de Ciência da Computação

Computação Paralela - 2024/2

Rodrigo José Zonzin Esteves

O uso de Open MP para a paralelização de um código sequencial em grafos

Descrição

Em relação à versão sequencial do Trabalho Prático 1 (TP1), a versão sequencial deste TP foi adaptada da seguinte maneira: i - toda a matriz que representa o grafo foi alocada contiguamente e ii - o cálculo dos vizinhos em comuns foi otimizado, somando-se apenas os elementos, coluna a coluna, por meio de uma expressão lógica.

Para a paralelização, utilizou-se as diretivas `#pragma omp parallel for` e `#pragma omp critical`, do padrão OpenMP em C. O Quadro 1 demonstra o processo de paralelização.

```
    #pragma omp parallel for
    for(int i =0; i< CONTROLADOR_TAM; i++){
        int k;
        for(int j = 0; j < CONTROLADOR_TAM; j++){
            //imprime só a matriz triangular inferior
            if(i >= j) continue;
            //determina o modulo do conjunto N(i)(intesec)N(j)
            k = determina_vizinhos(g, i, j);
              //printf("Thread %d processando vertices: %d %d\n", omp_get_thread_num(), i,
j);
            //se o número de vizinhos for inferior a 1, pula
            if(k <= 0) continue;
            #pragma omp critical
            fprintf(f_saida, "%d %d %d\n", i, j, k);
        }
    }
```

Quadro 1: Código utilizado para a paralelização do problema

Para os testes, utilizou-se um grafo *G(V, E)*, onde |V|= 6347 e *E* são arestas aleatórias, |E| = 5000. A Figura 1 apresenta um *plot* de *G* com propósito meramente ilustrativo.

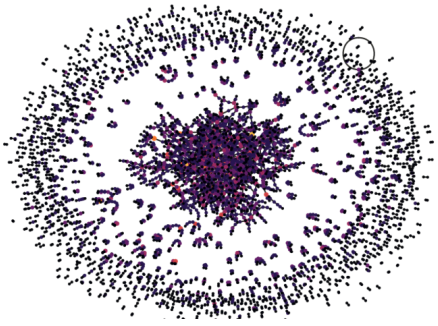

Figura 1: Grafo utilizado para teste

Resultados

A Tabela 1 apresenta os tempos de execução real obtidos pelo programa time, no Linux. Conforme se observa, o modo paralelo apresenta tempos consistentemente menores, o que indica a efetividade do paralelismo na redução do tempo necessário para o processamento.

Os dados ainda permitem observar uma tendência de melhora de desempenho à medida que *n* aumenta. Isso ocorre pois a parte inicial do código, não paralelizável, representa um overhead de processamento. Isto quer dizer que se gasta mais tempo com o processamento do arquivo de entrada e da construção de *G* do que com a tarefa de determinar vizinhos comuns.

n	Sequencial	Paralelo	Par./Seq.
100	0,466	0,434	93,13%
200	1,319	0,924	70,05%
400	3,246	2,112	65,06%
600	5,698	3,552	62,34%
800	5,698	4,95	86,87%
1000	9,056	6,657	73,51%
5000	281	148,588	52,84%

Tempo de execução (s)

A Tabela 2 apresenta os tempos de usuário, isto é, a somatória do tempo em CPU gasto por todos os processos. Naturalmente, o código paralelo apresenta um tempo de usuário maior, uma vez que mais threads ocuparam a CPU durante o processamento.

n	Sequencial	Paralelo
100	0,347	0,454
200	1,205	1,629
400	3,126	4,293
600	5,586	7,714
800	8,949	11,444
1000	13,515	18,486
5000	280,823	366,488

Tempo de usuário (s)

Conclusão

A implementação utilizada foi capaz de reduzir o tempo de processamento até pela metade. Além disso, a facilidade de implementação é um ponto favorável à utilização do OpenMP, pois apenas duas linhas de código destoam do código sequencial.