Trabalho 2 - Programação Paralela
Relatório de Experimentos

Geffté L. S. Caetano[1], **Arthur H. A. Farias**[1]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 549 – 79.070-900 – Campo Grande – MS – Brazil

{geffte.caetano,arthur.farias}@ufms.br

Abstract. *In this study, the arbitrary-sized clique counting algorithm on undirected graphs was implemented and tested in two parallelized versions using OpenMP. In version 1, the static, dynamic and guided scheduling types were explored to distribute the work among the threads. In version 2, the use of automatic scheduling was abandoned, and load balancing and load stealing were manually implemented, allowing idle threads to "steal" tasks from more overloaded threads. Comparisons between these approaches were made based on the execution time on graphs of different sizes and characteristics, aiming to analyze the efficiency gains or losses of each strategy.*

Resumo. *Neste estudo, o algoritmo de contagem de cliques de tamanho arbitrário em grafos não orientados foi implementado e testado em duas versões paralelizadas usando OpenMP. Na versão 1, foram explorados os tipos de agendamento static, dynamic e guided para distribuir o trabalho entre os threads. Já na versão 2, abandonou-se o uso de agendamentos automáticos, implementando-se manualmente o balanceamento de carga e o roubo de carga, permitindo que threads ociosas "roubassem" tarefas de threads mais sobrecarregadas. As comparações entre essas abordagens foram feitas com base no tempo de execução em grafos de diferentes tamanhos e características, visando analisar os ganhos ou perdas de eficiência de cada estratégia.*

1. Grafo

De acordo com [Feofiloff et al. 2011], um grafo é um par (V,A), onde V é um conjunto arbitrário e A é um subconjunto de V. V é chamado de **vértices** e seu subconjunto A é chamado de **arestas**. Ainda nessa linha, os elementos de A assumiram a forma v,w, onde v e w pertecem ao conjunto V.

1.1. Cliques, cliques máximas e de tamanhos arbitrários

Segundo [Feofiloff et al. 2011], dado um grafo G, uma **clique** (ou conjunto completo) de G é definido por ser um conjunto de vértices X dois a dois adjacentes, ou seja, deve haver pelo menos uma aresta que conecta todos os vértices de X.

Ainda nessa linha, se há uma clique máxima denotada por $\omega(G)$, é óbvio que haverá pelo menos uma clique de tamanho menor ou igual a k, onde $k \leq \omega(G)$. Assim, é possível concluir que, desde que haja uma clique máxima que atenda as condições acima, é possível obter cliques de tamanho menores que a máxima.

2. OpenMP

O OpenMP (Open Multi-Processing) é uma API (Interface de Programação de Aplicações) amplamente utilizada para programação paralela em sistemas com múltiplos processadores ou núcleos. Ele permite a paralelização de programas em C, C++ e Fortran, facilitando a criação de aplicações que podem executar múltiplas tarefas simultaneamente. O OpenMP oferece um conjunto de diretivas, bibliotecas e funções para dividir o trabalho entre threads, coordenar a execução paralela e gerenciar a sincronização entre elas. Sua principal vantagem é a simplicidade na adaptação de programas sequenciais para paralelos, sem exigir grandes mudanças no código original, sendo ideal para melhorar o desempenho em tarefas computacionais intensivas, como o processamento de grandes volumes de dados ou a execução de algoritmos complexos.

No OpenMP, os agendamentos static, dynamic e guided controlam como as iterações de um loop são distribuídas entre os threads. No static, as iterações são divididas de forma fixa e igual entre os threads no início. No dynamic, as iterações são distribuídas de forma dinâmica, com os threads pegando blocos conforme vão terminando suas tarefas, útil quando o tempo de execução das iterações varia. O guided começa com blocos maiores e vai diminuindo o tamanho conforme os threads completam as tarefas, buscando equilibrar a carga ao longo da execução. A escolha do agendamento depende do padrão de trabalho e da eficiência desejada.

2.1. Implementação do Algortimo de contagem de cliques de forma paralela usando OpenMP

A implementação do algoritmo de contagem de cliques de forma paralela utilizando OpenMP envolve a divisão do trabalho entre várias threads, permitindo que cada uma processe uma parte do grafo simultaneamente. Através de diretivas como #pragma omp parallel for, é possível executar loops em paralelo, o que melhora a eficiência do algoritmo. A escolha de estratégias de balanceamento de carga, como as métricas Static, Dynamic e Guided, também é crucial para otimizar o desempenho. Após a contagem, os resultados das diferentes threads precisam ser combinados de maneira adequada, garantindo a precisão dos cliques contados. Essa abordagem permite uma utilização mais eficiente dos recursos computacionais, especialmente em sistemas com múltiplos núcleos.

2.2. Implementação do Algortimo de contagem de cliques de forma paralela usando OpenMP e balanceamento de carga

Na implementação do algoritmo de contagem de cliques de forma paralela com OpenMP e balanceamento de carga manual, o foco está em otimizar a distribuição do trabalho entre as threads sem depender das estratégias de agendamento automáticas do OpenMP. Cada thread inicia sua tarefa de contagem e, quando termina, pode ”roubar” carga de trabalho de outras threads que ainda estão processando, garantindo uma utilização mais equilibrada dos recursos. A escolha de qual thread será roubada é aleatória, pois o problema é irregular e a carga de trabalho pode variar de maneira imprevisível entre as threads. Esse método permite que as threads mais rápidas assumam tarefas não concluídas, minimizando o tempo ocioso e melhorando a eficiência geral do algoritmo. A combinação de paralelismo com um balanceamento de carga controlado manualmente resulta em um desempenho superior, especialmente em cenários onde a carga de trabalho é irregular.

3. Resultados

Por fim, esta seção mostrará os resultados obtidos, com comparações realizadas variando o tamanho k das cliques e os datasets para cada algoritmo. Também foi contabilizado o tempo de execução de cada um para determinar se houve ganho ou perda. Além disso, houve uma variação no valor arbitrário de r cliques ”roubadas”, a fim de investigar se, a partir de um determinado valor, não há mais ganho de desempenho ao roubar carga de trabalho de outras threads.

A tabela abaixo apresenta os valores das cliques obtidas e os tempos de execução em segundos para o algoritmo paralelo, considerando as diferentes métricas do OpenMP (Static, Dynamic e Guided) e o algoritmo com balanceamento de carga. Podemos comparar os tempos das implementações em OpenMP com os tempos das implementações usando PThread. Todos os dados foram organizados em uma única tabela para facilitar a comparação.

Agora, segue abaixo a análise dos resultados:

Table 1. Comparativo de Tempo de Execução: Algoritmo Paralelo com OpenMP – Schedules vs. Balanceamento Manual

Dataset	Tamanho das Cliques (k)	Tempo (s) - Static	Tempo (s) - Dynamic	Tempo (s) - Guided	Tempo (s) - Balanceamento Manual
citeseer	3	0.0115	0.0073	0.0141	0.0123
	4	0.0155	0.0093	0.0053	0.0127
	5	0.0161	0.0113	0.0067	0.0141
	6	0.0183	0.0043	0.0053	0.0093
ca_astroph	3	3.0360	0.9081	4.889	0.7570
	4	48.6461	17.6549	75.5587	5.73739
	5	822.0420	177.571	768.534	50.4853
	6	7364.4700	1469.8	7506.9	357.1870
	7	+4hrs	+4hrs	+4hrs	1270.0200
dblp	3	5.8966	1.8043	8.9853	1.0673
	4	67.6689	13.8247	95.0512	9.8669
	5	2148.23	506.6000	2145.6200	178.3060
	6	+4hrs	6760.9600	+4hrs	1831.4300

A fim de comparação, também temos os resultados da experiência anterior, que contém a implementação em threads do C++. Pode ser observada na seguinte tabela:

Table 2. Comparativo de Tempo de Execução: Algoritmo Serial e Paralelo com Thread (com e sem balanceamento)

Dataset	Tamanho das Cliques (k)	Tempo (s) - Serial	Tempo (s) - Paralelo	Tempo (s) - Balanceado
citeseer	3	0.0522	0.0112	0.0297
	4	0.0650	0.0184	0.0303
	5	0.0687	0.0124	0.0406
	6	0.0699	0.0125	0.0407
ca_astroph	3	17.3828	12.9792	10.6048
	4	221.2280	165.0540	158.169
	5	2220.5400	1782.0200	1694.71
	6	+4hrs	+4hrs	+4hrs
	7	+4hrs	+4hrs	+4hrs
dblp	3	36.1096	18.9649	18.2536
	4	286.3000	200.9500	175.129
	5	5454.2800	1833.3700	1723.25
	6	+4hrs	+4hrs	+4hrs

Para analisar o impacto do agendamento dynamic com k=3 no dataset ca_astroph, realizamos uma série de experimentos variando o tamanho dos chunks de trabalho. Os chunks foram definidos como 1, 2, 4, 8, 16, 32 e 64, permitindo observar como diferentes divisões do trabalho influenciam o desempenho do algoritmo.

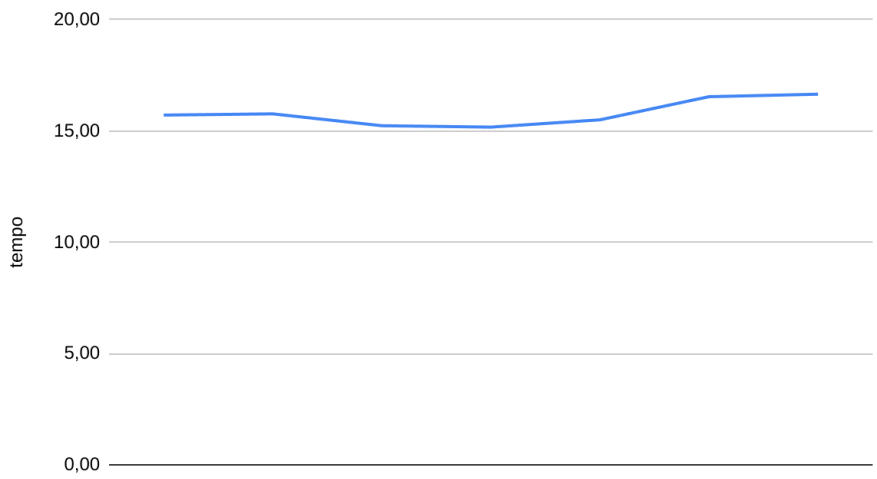

Figure 1. Tempo por variação de chunk

No experimento de roubo de carga utilizando OpenMP, variamos as configurações do roubo de carga para o mesmo dataset e o mesmo k. Os resultados obtidos foram os seguintes:

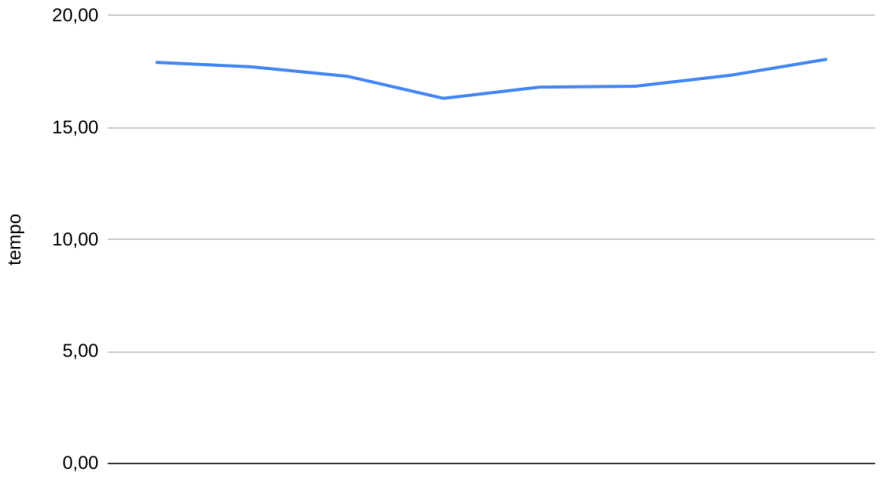

Figure 2. Tempo por variação de chunk

4. Conclusões

Analisando os chunks, podemos perceber pelo gráfico, mesmo em um exemplo pequeno e rápido, que aumentar os chunks indefinidamente, não necessariamente trará melhores resultados e/ou performance. O mesmo ocorreu com o gráfico do roubo de carga utilizando o OpenMP, que conseguiu se equiparar ao schedule Dynamic em tempo de execução.

References

Feofiloff, P., Kohayakawa, Y., and Wakabayashi, Y. (2011). Uma introdução sucinta à teoria dos grafos.