Trabalho Prático 1 - Computação Paralela

Rodrigo José Zonzin Esteves

Novembro de 2024

1 Versão Sequencial

O trabalho consiste em determinar, para um grafo $G = (V, E)$, a intersecção entre os conjuntos de vizinhança $N(u)$ e $N(v)$, u e $v \in V$. Inicialmente, deve-se tratar os requisitos de leitura e saída do programa. Para a modelagem adequada do grafo, faz-se a leitura e o armazenamento de todos os inteiros *inputados*. Em seguida, percorre-se elemento a elemento da lista para determinar o número de elementos únicos. Por exemplo, para a sequência $I = (1, 2, 1, 3, 1, 5)$, deve-se determinar 4 pois apenas 1, 2, 3 e 5 são elementos unitários em I. Esse procedimento é necessário para que se saiba a dimensão $|V|$ de G a ser alocada. Após a alocação de uma matriz de adjacências com dimensão $|V| \times |V|$, insere-se as arestas (u, v) *inputadas* pelo arquivo.

Considerando-se que o requisito consiste em determinar o número de elementos pertencentes ao conjunto $N(u) \cap N(v)$ e não o conjunto em si, calcula-se quantos elementos a_{ij} são iguais aos elementos b_{kj}, para a mesma coluna j. Por exemplo, para os vértices 1 e 2 em M, temos $|N(1) \cap N(2)| = 2$, pois apenas $a_{13} = b_{23}$.

$$M = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{1}$$

Em seguida, para cada vetor-linha $\mathbf{a_i}$ da matriz, calcula-se o conjunto intersecção com os $\mathbf{a_k}$ demais vetores-linha. Por se tratar de uma modelagem de um grafo não-direcionado, pode-se efetuar o cálculo apenas com a matriz triangular superior de M, evitando que o cálculo seja feito para ij e ji.

2 Versão Paralela

2.1 Introdução

Para a versão paralela, a modelagem do grafo por matriz de adjacência foi adaptada para um array linear de inteiros. Essa abordagem é necessária pois, para n processadores disponíveis, pode-se subdividir os $|V| \times |V|$ inteiros para n processos independentes.

Por exemplo, para matriz P a seguir, um processo *rank* 0 poderá processar as duas primeiras linhas P_1 e P_2 e outro processo *rank* 1 poderá processar as linhas P_3 e P_4.

$$P = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{2}$$

2.2 Protocolo de comunicação: abordagem utilizando broadcast

Inicialmente, o código mestre (rank 0) é o responsável por ler todos o arquivo de entrada e direcionar essa informação ao demais processos. Para o processamento inicial, deve-se enviar o tamanho do Grafo para todos os processos e o proprio rank 0 deve ter acesso à métrica.

Uma vez que o processo mestre tenha enviado o tamanho do grafo, os processos obedientes devem receber os dados da matriz de adjacência (um vetor linear). Em seguida, os processos fazem o cálculo dos vizinhos em comum da parte que lhes cabe e finalizam a execução.

Para o cálculo, define-se uma função que recebe o grafo alocado, o rank e o número total de processos. Cada processo deverá processar um número determinado de pares $(u, v) \in E$. Para determinar o número de pares por processo, considera-se o número total de pares (total $= n(n-1)/2$) admitidos em um grafo divido pelo número total de clusteres. Essa abordagem é menos ineficiente pois, no caso de um grafo esparso, haverá iterações sobre arestas inexistentes.

Cada processo deverá iniciar o processamento em um ponto proporcional ao seu rank. Por exemplo, o processo rank 0 deverá processar desde os vértices $0 \cdot \frac{\text{total}}{nprocs}$ até $\frac{\text{total}}{nprocs}$. Para o rank 1, de $1 \cdot \frac{total}{nprocs}$ até $2 \cdot \frac{total}{nprocs}$, ..., para o $n - esimo$ rank de $(n-1) \cdot \frac{total}{nprocs}$ até total.

3 Resultados

Os testes foram realizados utilizando-se dois dispositivos conectados à mesma rede: um i3-7020U com 2 núcleos e um i5-10400 com 6 núcleos. A disparidade de eficiência dos dispositivos computacionais é uma das limitações deste trabalho.

A Figura 1 apresenta os resultados para a execução de um arquivo com $|V| = 400$. Como esperado, o aumento no número de processadores diminui o tempo de execução do processo. Entretanto, a partir de 6 núcleos, o ganho de eficiência da abordagem paralela se mostrou menos expressivo em relação ao ganho inicial. Isso pode ocorrer diante de *overheads* causados por chamadas de sistema, condições bloqueantes dos processos etc.

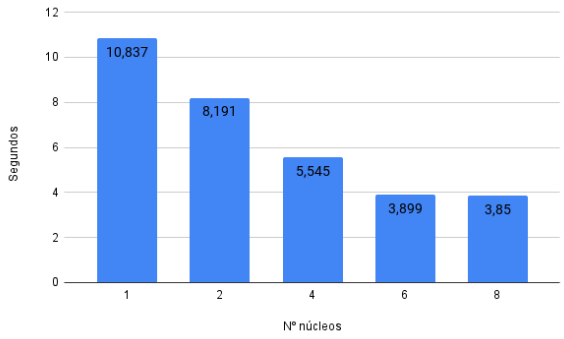

Figura 1: Resultado da execução em ambiente multiprocessado ($|V| = 400$)

De forma semelhante, a Figura 2 apresenta o resultado obtido para um arquivo com $|V| = 1000$. A tendência de diminuição do tempo de execução também é observada neste caso. Os dados estão

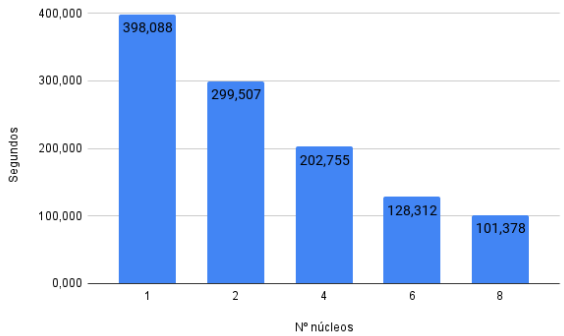

Figura 2: Resultado da execução em ambiente multiprocessado ($|V| = 1000$)

em linha com o esperado de uma abordagem multiprogramada, em que pese a versão sequencial ser mais rápida ($\approx +5971{,}42\%$). Isso ocorre, como discutido, pela implementação equivocada do processo paralelo, causando *overhead* de comunicação e/ou processamento.

4 Abordagem utilizando Scatter

Tentou-se empregar uma segunda abordagem utilizando a função MPI_Scatter( ). Essa função é utilizada para enviar dados para todos os processos em um comunicador. A Figura 3 representa essa funcionalidade.

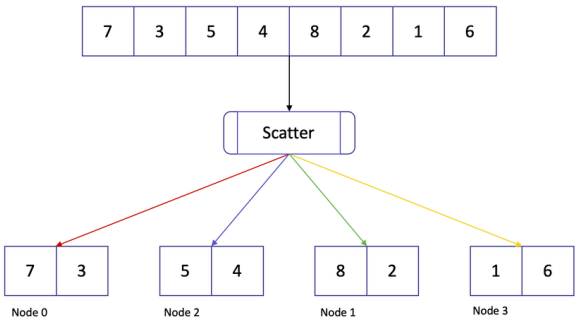

Figura 3: Exemplo de Scatter

A abordagem tentada consiste em subdividir as $|V|$ linhas de uma matriz de adjacência entre os n processos. Dessa forma, cada processo lidaria com $\frac{|V|}{n}$ (ou $\frac{|V|}{n} + k$, caso a divisão seja desbalanceada) linhas. Cada processo poderia utilizar a função que determina o número de vizinhos entre vértices uv para o seu subconjunto de vértices. A Figura 4 exemplifica o comportamento pretendido.

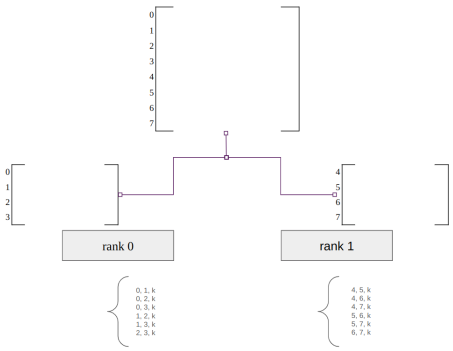

Figura 4: Abordagem alternativa: subdivisão das linhas de G

Por fim, ter-se-ia a necessidade de combinar vértices em ranks diferentes do comunicador para que a comparação de um vértice u ocorra com todos os $V - \{u\}$ vértices alcançáveis. Essa abordagem é promissora mas não foi concluída.

Uma terceira abordagem pode ser possível passando do processo mestre para os processos no comunicador uma coluna inteira, e acumulando a quantidade de *"matches"* entre cada par de vértices.