Universidade Federal de São João del-Rei
DEpartamento de Ciência da Computação
Programa de Graduação

Relatório sobre a implementação do algoritmo de Floyd-Warshall em paralelo usando a biblioteca OpenMPI

Aluno: Jardel Felipe de Carvalho
Professor: Rafael Saquetto Oliveira

Outubro
2019

Conteúdo

1 Descrição **1**

2 Resumo **1**
2.1 Floyd-Warshall 1
2.2 Fox 1
2.3 MPI 2

3 Abordagem **2**

4 Implementação **2**
4.1 Comunicação Entre Processos 2
4.2 Vizinhos Laterais 3

5 Vizinhos Verticais **3**

6 Análise dos Resultados **3**
6.1 SpeedUp 3
6.2 Tempos de Execução 4

7 Dificuldades Obtidas e Comentários **5**

Bibliografia **5**

Anexo **6**

1 Descrição

Este relatório é parte dos requisitos necessários para a conclusão do primeiro trabalho prático da disciplina de Computação Paralela oferecida pela Universidade Federal de São João del-Rei.

Para o trabalho em questão foi proposto a implementação do algoritmo de Floyd-Warshall utilizando a biblioteca OpenMPI com o intuito de comparar a eficiência de execução paralela e sequencial.

Neste relatório, os resultados obtidos assim como a descrição da metodologia será especificada e discutida.

2 Resumo

2.1 Floyd-Warshall

Entre as soluções para problemas de menor caminho entre dois vértices de um grafo se encontra o algoritmo de Floyd-Warshall. Ao contrário do algoritmo de Djikstra o algoritmo de Floyd-Warshall produz resultados plausíveis para grafos com arestas de peso negativo.

Seu funcionamento requer uma matriz ou lista de adjacência como entrada, mediante a isto é realizada a busca por vértices t que possam intermediar o acesso entre os vizinhos u e v e ao mesmo minimizar as suas distâncias dado os pesos das arestas.

O algoritmo de Floyd-Warshall também por ser executado sob uma aborgem parecida com uma multiplicação de matrizes em que menor distância entre cada vértice é dada pela matriz *Df* encontrada após multiplicações sucessivas de matrizes, sendo *D2 = D1 x D1*, *D4 = D2 x D2*, *D8 = D4 x D4* ... *Df = Dg x Dg*, $f \geq N$ e $g < N$. Ao contrario da multiplicação de matrizes convencional o algoritmo de Floyd-Warshall realiza operações de soma e mínimo ao invés de multplicação e soma.

2.2 Fox

O algoritmo de Fox permite a multiplicação de matrizes por meio da propagação de pedaços da mesma entre nós quaisquer. Seu funcionamento imita uma multiplicação convencional de matrizes, entretando, com o particionamento dos dados o algoritmo de Fox torna possível a atuação de múltiplos processos na busca por resultados.

As condições para que o algoritmo de Fox seja aplicado sobre uma matriz de dimensão N dado P processos disponíveis é que P seja um quadrado perfeito e que N mod $\sqrt{P}$ seja igual a zero.

As sua execução é separada em quatro partes. Primeiro é realizado o particionamento da matriz inicial em pedaços B de dimensão N / $\sqrt{P}$ que serão associados e enviados à cada um dos processos respectivamente. O particionamento organiza os processos em linhas de forma que suas numerações se dão na forma RTL (Right To Left Script). Cada processo inicialmente realiza uma cópia da matriz B denotada por R.

No segundo passo, é escolhido para cada linha um processo que será responsável por compartilhar a sua submatriz R com os seus vizinhos laterais. Feita a escolha é realizado o envio e recebimento dos dados pelos vizinhos, a matriz recebida é denotada por A.

No terceiro passo, é realizada a multplicação da matriz recebida A, pela matriz B residente em cada processo atualmente.

No quarto passo a matriz B é enviada para o processo logo acima, assim como é recebida a matriz B do processo da qual é aplicada na matriz B do processo atual.

Os procedimentos dois ao quatro são repetidos $\sqrt{P}$ vezes e o *rank* da matriz ecolhida no segundo passo é denotado por $(r + i) \mod \sqrt{P}$ em que r é a linha da qual está sendo verificada e i é o número da iteração atual.

2.3 MPI

O MPI (Message Passing Interface) é uma especificação que expressa um conjunto chamdas e estruturas de dados util para a comunicação entre processos de funcionamento paralelo e memória distribuída. Para este trabalho foi utilizada a implementação OpenMPI para a linguagem C.

3 Abordagem

Sabe-se que algoritmo de Floyd-Warshall pode também trazer resultados com a multplicação sucessiva de matrizes, sendo assim, para trazer a execução para o formato paralelo com OpenMPI, foi utilizado o algoritmo de Fox em cada uma das multiplicações.

4 Implementação

4.1 Comunicação Entre Processos

Foi optado por não fazer o uso de comunicadores na implementação proposta. Sendo assim, algumas operações foram realizadas com base no *rank* do

processo com o intuito cumprir a tarefa de identificação de nós destinatários e nó remetente.

4.2 Vizinhos Laterais

Cada processo escolhido para enviar sua submatriz a seus vizinhos laterais identificou os remetentes com base no seguinte cálculo.

$$q \times r \leq rank < q \times r + q \tag{1}$$

Onde $q = \sqrt{p}$, $p = num\_procs$ e r é a linha do processo escolhido.

Vale ressaltar que com esta inequação o processo escolhido enviará sua submatriz para si mesmo, portanto o recebimento de tal submatriz poderá ser realizado por todos os processos da linha, assim como por todos os processos da operação, visto que este compartilhamento de dados ocorre em todas as linhas de processos.

5 Vizinhos Verticais

Ao fim de cada iteração cada processo com número de identificação *my_rank* compartilha a matriz B com o processo diretamente acima assim como recebe do processo diretamente abaixo. A determinação do destinatário logo acima é dada pelo seguinte cálculo.

$$up = (my\_rank + q \times (q - 1))\%p \tag{2}$$

A determinação do remetente logo abaixo é dada pelo seguinte cálculo.

$$up = (my\_rank + q)\%p \tag{3}$$

Onde *my_rank* é o número de identificação do processo.

6 Análise dos Resultados

6.1 SpeedUp

A execução dos testes se tornou inviável para matrizes de dimensões mais altas, visto que os testes se tornaram muito demorados para matrizes de dimensão maior de 24 na execução paralela. É possível notar que nos testes realizados, não foi obtido SpeedUp.

O código implementado, para as matrizes testadas produziu resultados equivalentes às execuções sequenciais.

Abaixo, o gráfico exibe os resultados obtidos por meio de matrizes de dimensões 12x12 e 24x24. As execuções foram realizadas 10 vezes em cada matriz, ao final a média dos tempos de execução foi utilizada para cálculo do SpeedUp através da fórmula *T(1)/T(n)* onde n é o número de processos. É possível perceber que existiu quase nenhum SpeedUp.

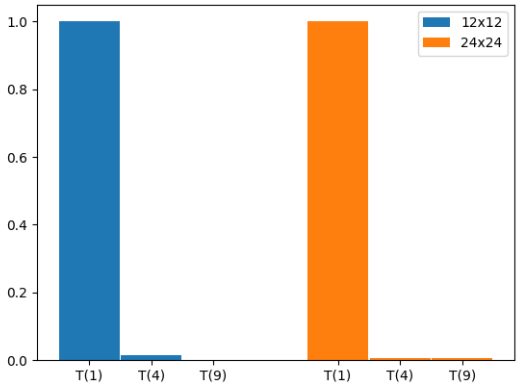

Figura 1: Compara4 e 9 processos em matrizes de 12x12 e 24x24.

6.2 Tempos de Execução

A tabela abaixo mostra os tempos de execução de cada processo assim como os seu desvio. É notório o aumento do tempo gasto nos casos em que $n > 1$. A tabela também mostra desvios mais altos para execuções com mais de um processo o que pode implicar em uma alta latência ou demais falhas na rede.

Caso de Teste	**Média**	**Desvio**
n=1, 12x12	0.0000604	0.0000074
n=1, 24x24	0.0005624	0.0000685
n=4, 12x12	0.0040997	**0.0085496**
n=4, 24x24	0.0812761	**0.1014220**
n=9, 12x12	0.0586874	**0.0044348**
n=9, 24x24	0.0906578	**0.0065898**

7 Dificuldades Obtidas e Comentários

Inicialmente, para o desenvolvimento deste trabalho, foi optado por solucionalo sem o auxílio de algoritmos já consolidados, entretanto, foi verificada uma grande dificuldade em se manusear todas as variáveis do problema, sendo estas, a multiplicação consistente de matrizes e o assincronismo entre processos.

Mediante a isto, recorreu-se ao algoritmo de Fox, do qual, simplificou consideravelmente a solução do problema. O autor do documento de base[**1**] para o entendimento do algoritmo menciona que a tarefa de implementação com o uso de comunicadores pode tornar o caminho mais fácil, entretanto, para este trabalho, foi observado que a identificação dos processos pertencentes a cada comunicador deveriam ser determinados mediante a algum cálculo, tendo em vista este aspecto, foi optado por realizar a comunicação somente através de chamadas *send* e *recv* utilizando os cálculos acima mencionados de maneira direta.

Em linhas gerais, para este trabalho, a maior dificuldade percebida foi a manipulação de todas as variáveis que compõe o problema, sendo estas o tempo levando em consideração o assincronismo dos processos, o particionamento e consistência dos dados como as transformações aplicadas no decorrer da execução e por fim a manutenção da lógica de multiplicação de matrizes.

Todo este conjunto precisa ser satisfeito para o resultado final ser correto e vale ressaltar que a inserção do fator tempo dificulta de maneira significativa o problema.

Bibliografia

[1] PACHECO, Peter S. A User's Guide to MPI $< https://www.lrz.de/services/software/parallel/mpi/mpi_guide.pdf >$

Anexo