Trabalho prático 1 - All-Pairs Shortest Paths - Dupla ou Individual

Professor: Rafael Sachetto Oliveira

20 de setembro de 2019

1 Objetivo

Permitir que os alunos adquiram um maior conhecimento prático do funcionamento de um ambiente de programação distribuído através da utilização da ferramenta de programação MPI.

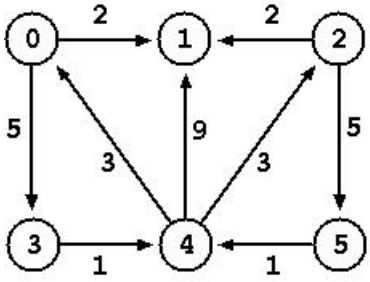
2 Descrição

O problema que se pretende resolver consiste em encontrar o conjunto de todos os caminhos mais curtos entre pares de vértices de um determinado grafo. Este problema é conhecido na literatura por *all-pairs shortest paths*.

A ideia é a seguinte: dado um grafo direcionado $G = (V, E)$ em que V é o conjunto de vértices e E é o conjunto de arestas, pretende-se determinar para todos os pares de vértices (v_i,v_j) o tamanho do menor caminho que começa em v_i e termina em v_j. Um caminho não é nada mais do que uma sequência de arestas, em que o vértice de destino e o vértice de origem de duas arestas consecutivas da sequência são os mesmos. Uma aresta pode ser vista como uma tupla da forma $< v_i, v_j, t >$ em que v_i é o vérice de origem, v_j é o vértice de destino e t é o peso da aresta entre os vértices. A soma dos pesos das arestas de um caminho definem o custo do caminho.

3 Implementação

Um grafo direcionado com V vértices pode ser representado por uma matriz $D1$ de dimensão $N \times N$ ($N = 6$ no exemplo abaixo) em que cada elemento (i, j) da matriz corresponde ao peso da aresta que liga o vértice v_i ao vértice



	0	1	2	3	4	5
0	0	2	0	5	0	0
1	0	0	0	0	0	0
2	0	2	0	0	0	5
3	0	0	0	0	1	0
4	3	9	3	0	0	0
5	0	0	0	0	1	0

v_j. A não existência de qualquer aresta entre um determinado par de vértices pode ser representada pelo valor zero.

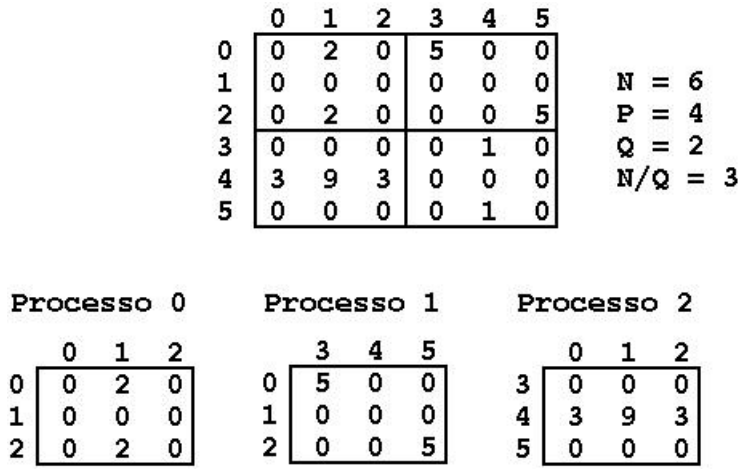
A solução do problema pode ser igualmente representada por uma matriz Df de dimensão $N \times N$ em que cada elemento (i, j) da matriz corresponde ao menor tamanho dos caminhos que começam em v_i e terminam em v_j.

	0	1	2	3	4	5
0	0	2	9	5	6	14
1	0	0	0	0	0	0
2	9	2	0	14	6	5
3	4	6	4	0	1	9
4	3	5	3	8	0	8
5	4	6	4	9	1	0

O algoritmo de Floyd-Warshall permite obter a matriz Df a partir da matriz $D1$ utilizando uma forma adaptada de multiplicação de matrizes, em que as operações de multiplicação e soma são substituídas pelas operações de soma e mínimo, respectivamente. Utilizando essa forma adaptada de multiplicação podemos obter a matriz Df construindo sucessivamente matrizes $D2 = D1 \times D1$, $D4 = D2 \times D2$, $D8 = D4 \times D4$, ..., $Df = Dg \times Dg$ em que $f \geq N$ e $g < N$.

A multiplicação de matrizes com a mesma dimensão pode ser eficientemente implementada em ambientes distribuídos utilizando o algoritmo de Fox. No entanto, a aplicação do algoritmo de Fox nem sempre é possível pois depende da dimensão das matrizes a multiplicar e do número de processos disponíveis. Para matrizes de dimensão $N \times N$ e P processos disponíveis no sistema, deve existir um inteiro Q tal que $P = Q^2$ e $N \, mod \, Q = 0$. O algoritmo de Fox divide as matrizes de dimensão $N \times N$ em P sub-matrizes de dimensão $(N/Q) \times (N/Q)$ e atribui cada uma delas aos P processos disponíveis no sistema.

Para calcular o resultado da multiplicação da sua sub-matriz, cada pro-



cesso, apenas necessita de trocar informação com os processos da mesma linha e da mesma coluna de sub-matrizes que a sua, o que permite reduzir o tamanho e número de mensagens a trocar entre os processos envolvidos (consulte as páginas 29 e 30 do A User's Guide to MPI para uma descrição mais detalhada do algoritmo de Fox). A aplicação sucessiva do algoritmo de Fox permite calcular as matrizes $D2$, $D4$, ..., Df do algoritmo de Floyd-Warshall e obter assim a matriz que soluciona o nosso problema.

4 Entrada e saída

A aplicação deverá aceitar como entrada de dados um arquivo com a descrição do grafo. Essa descrição deverá seguir o seguinte padrão: a primeira linha indica o inteiro N que representa o número de vértices do grafo, enquanto que as seguintes N linhas indicam a matriz que representa as arestas entre todos os pares de vértices do grafo. A descrição do grafo do exemplo seria a seguinte:

```
6
0 2 0 5 0 0
0 0 0 0 0 0
0 2 0 0 0 5
0 0 0 0 1 0
3 9 3 0 0 0
0 0 0 0 1 0
```

No caso da dimensão da matriz e do número de processos disponíveis não serem compatíveis com os requisitos do algoritmo de Fox, deverá ser apresentada uma mensagem de erro e a execução deve ser cancelada.

Como resultado a aplicação deverá escrever as N linhas da matriz que representa o menor tamanho dos caminhos entre todos os pares de vértices do grafo. Para o grafo do exemplo, o resultado seria o seguinte:

```
0 2 9 5 6 14
0 0 0 0 0 0
9 2 0 14 6 5
4 6 4 0 1 9
3 5 3 8 0 8
4 6 4 9 1 0
```

5 Especificação

Criar uma versão paralela com memória distribuída do algoritmos descrito anteriormente. Espera-se que sejam explorados todos os potenciais de melhora na execução do algoritmo. As implementações devem ser feitas em linguagem C utilizando MPI. Não devem ser utilizados pacotes ou bibliotecas específicas, a menos que autorizadas pelo professor. Os algoritmos devem ser compiláveis com o gcc e executáveis em sistemas Linux.

6 O Que Entregar (via moodle)?

Um arquivo compactado, com seu nome, que inclua:

- Arquivos com o código da aplicação.
- Um makefile que gera o executável da aplicação (utilizar o nome floyd).
- Um documento contendo o relatório do trabalho onde se descreve de forma sucinta e clara:
  - A implementação (ideia base do algoritmo, estruturas de dados utilizadas, funções auxiliares utilizadas, ...);
  - As dificuldades encontradas na implementação (se alguma);
  - Os tempos de execução (em segundos e sem contabilizar as operações de input/output) e speedups (em relação à execução com 1 processo) obtidos para 1, 4 e 8 processos.