Computação Paralela e Distribuída

Trabalho Académico

Sistema de Recomendação

Versão 1.0

Segunda-feira, 04/03/2024

Conteúdo

1. Introdução

O propósito deste trabalho académico é ganhar experiência em programação paralela em sistemas *UMA (Uniform Memory Access)* e multicomputadores, usando OpenMP e MPI, respectivamente. Para tal, os estudantes devem escrever uma implementação serial e duas paralelas de um algoritmo de sistemas de recomendação.

2. Descrição do Problema

Existem muitos serviços que fazem recomendações para o utilizador, sejam músicas, filmes, livros, etc. Uma abordagem é combinar a actividade anterior de diferentes utilizadores e, em seguida, sugerir itens que utilizadores com perfis semelhantes selecionaram e deram uma avaliação alta.

Dado um conjunto de itens, I_0 até I_m, e um conjunto de utilizadores, U_0 até U_n, podemos construir uma matriz com todas as avaliações que os utilizadores deram aos itens. Para um grande número de itens e/ou utilizadores, esta será normalmente uma matriz

esparsa, contendo o subconjunto de itens que os utilizadores avaliaram. Por exemplo, com três utilizadores e cinco itens, a matriz A:

$$\boldsymbol{A} = \begin{array}{c|ccccc} & I_0 & I_1 & I_2 & I_3 & I_4 \\ U_0 & 2 & 0 & 3 & 0 & 0 \\ U_1 & 0 & 0 & 3 & 0 & 1 \\ U_2 & 0 & 5 & 3 & 4 & 0 \end{array}$$

representa uma situação em que o utilizador U_0 avaliou dois itens, I_0 e I_2, com 2 e 3 (digamos que em uma escala de 1 a 5, 0s indicam nenhuma avaliação). Da mesma forma para os outros utilizadores.

Para advinhar o quão atraente é um item ainda não selecionado pelo utilizador, vamos estimar os valores dos zeros nesta matriz. Para tanto, assumimos que os itens possuem características, porém não sabemos quais são nem quantas devemos considerar. Digamos que para o exemplo acima temos dois características, F_0 e F_1. Então podemos considerar duas matrizes auxiliares, uma que associa a importância de uma característica para cada utilizador, L, a outra como a característica está presente em cada item, R:

$$\boldsymbol{L} = \begin{array}{c|cc} & F_0 & F_1 \\ U_0 & & \\ U_1 & & \\ U_2 & & \end{array} \qquad \boldsymbol{R} = \begin{array}{c|ccccc} & I_0 & I_1 & I_2 & I_3 & I_4 \\ F_0 & & & & & \\ F_1 & & & & & \end{array}$$

A abordagem é então encontrar as matrizes L e R de modo que a matriz B = L × R tenha todas as mesmas entradas diferentes de zero que A.

Para o exemplo actual, obtemos:

$$\boldsymbol{L} = \begin{bmatrix} 1.48 & 1.29 \\ 1.36 & 1.38 \\ 1.98 & 0.92 \end{bmatrix} \qquad \boldsymbol{R} = \begin{bmatrix} 0.72 & 2.02 & 0.93 & 1.49 & 0.25 \\ 0.73 & 1.09 & 1.26 & 1.14 & 0.48 \end{bmatrix}$$

$$\boldsymbol{B} = \boldsymbol{L} \times \boldsymbol{R} = \begin{bmatrix} 2.00 & 4.38 & 3.00 & 3.70 & 0.99 \\ 1.98 & 4.23 & 3.00 & 3.60 & 1.00 \\ 2.09 & 5.00 & 3.00 & 4.00 & 0.94 \end{bmatrix}$$

Este resultado sugere que o utilizador U_0 é altamente aconselhado a tentar o item I_1 e deve evitar o item I_4.

3. Factorização de Matriz

Utilizaremos um método iterativo para encontrar as matrizes auxiliares **L** e **R**. Começamos por inicializar essas matrizes com valores aleatórios, obtendo um valor inicial para a solução aproximada, $\boldsymbol{B}^{(0)} = \boldsymbol{L}^{(0)} \times \boldsymbol{R}^{(0)}$. Em seguida, examinamos cada valor diferente de zero da matriz A e melhoramos essa estimativa inicial minimizando a soma dos erros quadráticos:

$$\min \sum_{i,j} \Delta_{ij}, \qquad \Delta_{ij} = (A_{ij} - B_{ij})^2$$

As novas estimativas para os elementos de **L** e **R** são obtidas como se segue. Para cada elemento diferente de zero A_{ij} da matriz **A**, actualiza os valores L_{ik} e R_{kj} usando as seguintes equações:

$$L_{ik}^{(t+1)} = L_{ik}^{(t)} - \alpha \sum_j \frac{\partial \Delta_{ij}}{\partial L_{ik}^{(t)}} \qquad R_{kj}^{(t+1)} = R_{kj}^{(t)} - \alpha \sum_i \frac{\partial \Delta_{ij}}{\partial R_{kj}^{(t)}}$$

Onde

$$\frac{\partial \Delta_{ij}}{\partial L_{ik}^{(t)}} = 2(A_{ij} - B_{ij}^{(t)})(-R_{kj}^{(t)}) \qquad \frac{\partial \Delta_{ij}}{\partial R_{kj}^{(t)}} = 2(A_{ij} - B_{ij}^{(t)})(-L_{ik}^{(t)})$$

O parâmetro α permite o ajuste da taxa de convergência. Para simplificar, usaremos um valor fixo para α para cada instância do problema.

O método pode iterar até que o erro fique abaixo de um determinado limite, ou definindo um número máximo de iterações, que é o que usaremos.

4. Detalhes de Implementação

4.1. Dados de Entrada

Seu programa deve permitir **um único parâmetro** de linha de comando, um nome de ficheiro com todas as informações da instância a ser executada.

O formato deste arquivo de texto é o seguinte:

1. a primeira linha é um número inteiro, que indica o número de iterações a serem executadas;
2. a segunda linha é um número de ponto flutuante, o valor de α;
3. a terceira linha é um número inteiro, o número de características latentes a considerar;
4. a quarta linha contém três inteiros separados por um espaço, o número de linhas e colunas, respectivamente, e o número de elementos diferentes de zero da matriz de entrada;
5. as demais linhas possuem a matriz, um elemento da matriz por linha, cada linha com um conjunto de três valores: dois inteiros indicando a linha e a coluna do elemento; e o elemento real, um valor de ponto flutuante no intervalo de 1 a 5.

4.2. Dados de Saída

A saída do programa deverá ser o item recomendado para cada utilizador. Para cada linha da matriz **B**, o programa deve gerar o índice (um inteiro) do item de maior valor sem levar em consideração aqueles que foram avaliados na matriz dada **A**. Assim, a saída consiste em `nU` linhas com um único inteiro , onde `nU` é o número de utilizadores.

Por exemplo, considerando a matriz A dada na Seção 2, o programa deve gerar:

```
$ sisRecom dado0.in
1
1
0
```

Os programas submetidos devem enviar estas linhas de saída (e **nada mais**!) para a saída padrão, para que o resultado possa ser validado em relação à solução correcta.

ATÊNCÃO: O projecto **não pode ser avaliado** a menos que siga rigorosamente estas regras de entrada e saída!

3.3. Notas de implementação

Considere o seguinte conjunto de notas:

- Para minimizar erros numéricos devido a arredondamentos, utilize o tipo double para os números de ponto flutuante.
- Use a rotina do Apêndice A para a inicialização das matrizes L e R.
- Observe que precisará de duas cópias de L e R, pois precisará de uma cópia estável ao calcular a nova iteração.

O programa deve aderir a estas regras para que possamos validar os seus resultados!

5. Parte 1 – Implementação Serial

Escreva uma implementação serial do algoritmo em C (ou C++). Nomeie o ficheiro de código-fonte desta implementação como `sisRecom.c`. Conforme declarado, seu programa deve esperar um único parâmetro de entrada.

ATÊNÇÃO: Esta será a base para comparações e espera-se que seja o mais eficiente possível.

5. Parte 2 – Implementação OpenMP

Escreva uma implementação OpenMP do algoritmo, com as mesmas regras e descrições de entrada/saída. Nomeie este código-fonte como `sisRecom-omp.c`.

Pode começar simplesmente adicionando diretivas OpenMP, mas é livre e incentivado a modificar o código para tornar a paralelização **mais eficaz** e **mais escalável**.

ATÊNÇÃO: Tenha cuidado com a sincronização e o balanceamento de carga!

Nota importante: para testar a escalabilidade, executaremos este programa atribuindo valores diferentes à variável shell `OMP_NUM_THREADS`. Se substituir esse valor em seu programa, não poderemos avaliar adequadamente a escalabilidade de seu programa.

6. Parte 3 – Implementação MPI

Escreva uma implementação MPI do algoritmo como para o OpenMP e resolva os mesmos problemas. Nomeie este código-fonte como `sisRecom-mpi.c`.

Para MPI, precisará modificar seu código substancialmente. Além da sincronização e balanceamento de carga, precisará levar em consideração a **minimização do impacto dos custos de comunicação**. É encorajado a explorar diferentes abordagens para a decomposição do problema.

ATÊNÇÃO: Créditos extras serão dados aos grupos que apresentarem uma implementação combinada de MPI + OpenMP.

7. O Que Entregar e Quando Entregar

Deve eventualmente enviar as versões sequenciais e paralelas do seu programa (**por favor, use os nomes dos ficheiros indicados acima**) e os tempos para executar as versões paralelas nos dados de entrada que serão disponibilizados (para 1, 2, 4 e 8 tarefas paralelas para ambos OpenMP e MPI, e adicionalmente 16, 32 e 64 para MPI). Observe que não usaremos nenhum nível de optimizações do compilador para avaliar o desempenho de seus programas, então também não deveria usar.

Também deve enviar um breve relatório sobre os resultados (1-2 páginas) que discuta:

- Qual a abordagem usada para paralelização?
- Que decomposição foi usada?
- Quais foram as preocupações de sincronização e porquê?
- Como foi tratado o balanceamento de carga?
- Quais são os resultados de desempenho? São o que esperava?

Entregará, inicialmente, a versão serial. Entregará a versão serial e a versão paralela do OpenMP na primeira data de entrega, com o relatório resumido, e depois a versão serial novamente (esperamos a mesma) e a versão paralela do MPI na segunda data de entrega, com um relatório actualizado.

Tanto o código quanto o relatório serão enviados para o email do docente (joao.costa@isptec.co.ao) em um ficheiro zip. Nomeie esses ficheiros como `g<n>serial.zip`, `g<n>omp.zip` e `g<n>mpi.zip`, onde `<n>` é o número do seu grupo. Também deve partilhar o repositório **GitHub** do projecto com o docente, através da *username* `joaojdacosta`. **Obs.:** Não deve incluir os ficheiros de instâncias.

Data de entrega (serial + OMP): **06 de Abril de 2024, até as 17h.**

- Obs: seu projecto será testado na aula prática logo após a data de entrega.

Data de entrega (serial + MPI): **18 de Maio de 2024, até as 17h.**

- Obs: seu projeto será testado na aula prática logo após a data de entrega.

A – Rotina PREENCHE_ALEATORIO_LR

```
#include <stdlib.h>

#define ALEATORIO ((double)random() / (double)RAND_MAX)

void preenche_aleatorio_LR(int nU, int nI, int nF)
{
     srandom(0);

     int i, j;

     for(i = 0; i < nU; i++)
          for(j = 0; j < nF; j++)
               L[i][j] = ALEATORIO / (double) nF;

     for(i = 0; i < nF; i++)
          for(j = 0; j < nI; j++)
               R[i][j] = ALEATORIO / (double) nF;
}
```

Para garantir que sua implementação esteja correcta, as matrizes iniciais $\boldsymbol{L}^{(0)}$ e $\boldsymbol{R}^{(0)}$ para o exemplo da Secção 2 devem ser:

$$\boldsymbol{L} = \begin{bmatrix} 0.420094 & 0.197191 \\ 0.391550 & 0.399220 \\ 0.455824 & 0.098776 \end{bmatrix} \quad \boldsymbol{R} = \begin{bmatrix} 0.167611 & 0.384115 & 0.138887 & 0.276985 & 0.238699 \\ 0.314435 & 0.182392 & 0.256700 & 0.476115 & 0.458098 \end{bmatrix}$$

Bom trabalho!