CE 265 2011

Exercício 4

1. Matriz Esparsa

Uma matriz é denominada esparsa quando possui poucos elementos não nulos. Matrizes esparsas são armazenadas em estrutura de dados específica que contém apenas com os elementos não nulos, composta por três arrays unidimensionais.

A matriz é percorrida por linhas e extirpada de elementos nulos. Os dois primeiros arrays (*val* e *col*) armazenam, respectivamente, o elemento não nulo e sua coluna. Por coerência com a linguagem utilizada na implantação, colunas são numeradas a partir de 1 em Fortran e a partir de 0 em C.

O terceiro array (*lin*) armazena o índice, nos dois arrays anteriores, do primeiro elemento não nulo da linha. Caso a linha seja nula, armazena o índice do primeiro elemento da próxima linha não nula. Caso não existam linhas posteriores não nulas, armazena o índice posterior ao último índice dos dois arrays anteriores. O índice inicial muda conforme a linguagem base.

Por exemplo, seja a matriz esparsa A_{6x6} representada na Tabela 1:

Tabela 1: Matriz A

10	0	0	0	7	0
0	0	0	5	8	0
0	0	0	0	0	0
0	0	0	0	0	0
2	9	0	6	0	0
0	0	0	0	0	0

A estrutura de dados em Fortran é composta pelos vetores *val, col e lin* representados nas Tabelas 2 e 3. Observe como *lin* representa as linhas nulas (2, 3 e 6).

Tabela 2: Vetores *val* e *col* em Fortran

índice	1	2	3	4	5	6	7
val	10	7	5	8	2	9	6
col	1	5	4	5	1	2	4

Tabela 3: Vetor *lin* em Fortran

índice	1	2	3	4	5	6
lin	1	3	5	5	5	8

Para reconstituir a matriz esparsa a partir dos três vetores, seja $A(n,m)$ a matriz esparsa declarada em Fortran, representada pelos três vetores com dimensões $val(nonZero)$, $col(nonZero)$ e $lin(n)$, onde $nonZero$ é o número de elementos não nulos de A. Então, para cada linha i com $1 \le i < n$ e para todo o $k \le nonZero$ tal que $lin(i) \le k < lin(i+1)$, a posição $val(k)$ armazena o valor de $A(i, col(k))$. A última linha (n) deve ser tratada de forma similar.

A estrutura de dados em C é similar à de Fortran, respeitados os índices iniciais diferentes. A estrutura de dados em C para armazenar a matriz da Tabela 1 é representada nas Tabelas 4 e 5.

Tabela 4: Vetores *val* e *col* em C

índice	0	1	2	3	4	5	6
val	10	7	5	8	2	9	6
col	0	4	3	4	0	1	3

Tabela 5: Vetor *lin* em C

índice	0	1	2	3	4	5
lin	0	2	4	4	4	7

O procedimento para reconstituir a matriz esparsa a partir dos três vetores em C é idêntico ao de Fortran, respeitadas as diferenças de índice inicial.

2. Norma 2

Dado um vetor x_N com $N \geq 1$ elementos, a norma 2 desse vetor é o escalar representado por $\|x\|_2$ e definido por

$$\|x\|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}$$

3. Exercício

Seja A_{NxN} uma matriz esparsa e quadrada. Seja x_N um vetor denso de *N* elementos. Matriz e vetor com elementos reais em precisão simples. Seja o cálculo recursivo $x^{j+1} = Ax^j$, com $j = 1,...,J$. Implante esse cálculo recursivo em linguagem de sua escolha, representando a matriz esparsa pela estrutura de dados acima descrita. Paralelize o cálculo utilizando OpenMP. Faça o exercício no Crow.

Os dados de entrada do programa são o nome de um arquivo com a matriz esparsa e o número de iterações (*J*). Inicialize $x^1 = 1$. Calcule o produto da matriz esparsa pelo vetor denso, armazenando o resultado em vetor denso. Implante o cálculo recursivo por um laço com exatamente *J* iterações – não tente eliminar a recursão. Meça o tempo de execução desse laço (*wall clock*). Calcule a norma 2 do vetor resultante ($\|x^{J+1}\|_2$). Encerre o programa imprimindo o nome do arquivo, o número de threads, o tempo de execução e a norma 2 do vetor resultante. Utilize uma única seção paralela contendo o laço com as *J* iterações.

Forneço três arquivos com matrizes esparsas, em formato descrito abaixo. Execute o programa variando o número de threads de 1 a 8 para cada uma das três matrizes. Forneço o número de iterações para cada matriz. Faça uma tabela com os tempos de execução e speed-up correspondentes em função do número de threads.

Encontre no *site* do curso o arquivo *tarball* Esparsas.tgz. Transfira o *tarball*. A execução de "tar –xzvf Esparsas.tgz" produz três arquivos (Esparsa_X.txt, com X=1,2,3), uma matriz esparsa por arquivo. São arquivos ASCII com *nonZero+1* linhas. Informações em cada linha são separadas por brancos. A primeira linha contém inteiros que representam o número de linhas (*n*), o número de colunas (*m*) e o número de não nulos (*nonZero*) da matriz, nessa ordem. Cada uma das demais *nonZero* linhas contém os dados de um elemento não nulo da matriz: o número da linha (inteiro) seguido pelo número da coluna (inteiro) seguido pelo valor do

elemento não nulo (ponto flutuante). Os elementos são armazenados por linhas. Linhas e colunas são contadas a partir de 1.

Inicie seu trabalho certificando-se que implantou corretamente o cálculo e o paralelismo solicitado. Teste seu programa em matrizes e no número de iterações de sua escolha. Garanta a correção seqüencial e paralela.

Em seguida execute o programa nos arquivos fornecidos, utilizando o número de iterações (J) reportado na Tabela 6 e variando o número de threads conforme descrito acima.

Tabela 6: Arquivos e iterações

arquivo	Esparsa_1.txt	Esparsa_2.txt	Esparsa_3.txt
J	20.000	1.300	180

4. Crow

Obtenha do site do curso o arquivo JogoDaVidaNoCrow.tgz. Extraia o arquivo XmitTempo.sh e o Makefile. Se quiser, use esses dois arquivos com as modificações abaixo:

1. XmitTempo.sh: Mude o nome do executável no arquivo para o nome de sua escolha. Insira a linha "cd ${DirBase}" após a seqüência de linhas "#PBS" para que a execução inicie-se no diretório corrente, possibilitando o uso de caminhos relativos para obter arquivos de entrada.
2. Makefile: Adapte o Makefile para seu programa. O conjunto de compiladores default no Cray é o conjunto da Portland (pgcc e pgf90), identificados por cc e ftn, respectivamente. Nos dois compiladores é necessário compilar seu fonte com a opção "-mp" para reconhecer e gerar código OpenMP.

5. Relatório

Empacote seu programa fonte, Makefile, script de execução e os arquivos de saída de cada execução contendo a linha impressa em um único arquivo .tgz (use tar –czvf <arquivo.tgz> <lista de arquivos a empacotar>). Faça relatório sucinto, de no máximo duas páginas, contendo:

1. O conjunto de testes que utilizou para assegurar-se da correção de seu programa (tente convencer-me da correção);
2. Tabela com os tempos de execução, speed-up e eficiência medidos;
3. Analise do ganho paralelo com o número de threads.

Entregue o relatório (pdf) e o arquivo empacotado pelo site do curso até a meia noite de 11 de abril.

Para verificar a correção de sua resposta poderei compilar e executar seu programa fonte em dados de minha escolha.

Atribuirei nota em função da correção, da redução do tempo de execução pelo paralelismo e da análise dos resultados. Sua nota será crescente com o ganho paralelo nos dados fornecidos e com a qualidade da análise dos resultados.