Міністерство освіти і науки України

Національний університет "Львівська політехніка"

Кафедра ЕОМ

****

**Звіт**

з лабораторної роботи № 1

з дисципліни: “ Паралельні та розподілені обчислення”

на тему: “ **ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОЇ ДЕКОМПОЗИЦІЇ ДЛЯ РОЗВ’ЯЗКУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ.**”

Варіант 20

Виконав: студент .гр. КІ-33

Харченко М.В.

Прийняв: асистент каф. ЕОМ

Козак Н.Б.

Львів 2020

**Мета роботи:** Вивчити методи декомпозицій задач. Набути навиків розв’язування задач з використанням функціональної декомпозиції.

**Завдання**

Використовуючи метод функціональної декомпозиції, розробити алгоритм обчислення запропонованого матрично-векторного виразу, який би враховував можливість паралельного виконання і був оптимальним з точки зору часових затрат. На основі створеного алгоритму написати програму яка дозволяє обчислити вираз та ілюструє проведену декомпозицію.

Вираз, який слід обрахувати, заданий наступним чином:

 ( Варіант 20 )

При чому елементи  визначаються згідно правил:

, де bi=20/(i3+20), і=1,2,...n



, де Cij=20/(i3-j3+2)

**Аналіз завдання.**

Вхідні дані: розмірність матриць – n, матриці ; вектори-стовпці .

Вектор-стовпець  та матриця  обраховуються, виходячи з уведеної розмірності, зауважимо, що значення їх елементів завжди менші одиниці і різко спадають зі збільшенням розмірності.

Згідно з поставленою задачею, в обчисленні загального виразу приймають участь три різні елементи – два вектори стовпці  та матриця .

Оскільки, згідно правил матричних обчислень, добуток не є комутативною операцією, всі множення слід виконувати в тій послідовності, яка задана. Результатом множення рядка на стовпець є число, а матриці на матрицю – матриця, рядок × матрицю = рядок, матрицю × стовпець = стовпець, рядок × стовпець = число, число × матрицю = матриця.

Декомпозиція задачі.

Однозначно, всі обчислення безпосередньо залежать від розмірності даних, тому найперше, слід забезпечити ввід змінної n, що визначає цю розмірність. Далі, можна паралельно виконувати обчислення значень вектора b та матриці С2, оскільки вони незалежні від інших параметрів. Крім того, на тому ж рівні декомпозиції слід визначати вхідні дані, тобто вводити з клавіатури, або генерувати випадковим чином матриці  та вектори-стовпці . Наступний рівень декомпозиції – це знаходження елементів виразу. Значення  залежить від введеної матриці А та обрахованого вектора b. Значення  залежить від введеної А1 та різниці векторів b1 і c1, тому знайти його можна лише після обчислення (b1 - c1). Зауважимо, що множення на константу не є окремою операцією, як і транспонування векторів. Аналогічно, знаходимо . Подальша декомпозиція відбувається згідно заданої послідовності операцій та врахування залежностей отриманих на кожному рівні даних. Порядок виконання програми умовно розбито на 8 етапів. Повна схема декомпозиції обчислення заданого виразу приведена нижче.

Об’єднання частин виразу проведено безпосередньо у схемі декомпозиції, оскільки воно однозначно визначається порядком обчислень.

**Схема декомпозиції задачі**

****

**Текст програми**

#include "stdafx.h"

#include "stdlib.h"

#include "conio.h"

#include "math.h"

FILE \*OutFile=fopen("output.txt","w");

int rows;

int Option(void)

{

char select;

while ( select!='y' && select!='Y' && select!='N' && select!='n' )

{

printf("\b");

select=getche();

}

if (select=='y' || select=='Y') return 1;

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Memory allocation \*\*\*\*\*/

float\*\* AllocateMatrix(int row)

{

int i;

float \*\*A;

A=new float\*[row];

if (A==NULL)

{

puts("Memory allocation error."); exit(0);

};

for (i=0; i<rows; i++)

{

A[i]=new float[row];

if (A[i]==NULL) {

puts("Memory allocation error."); exit(0); };

};

return A;

};

float\* AllocateVector(int row)

{

float \*A;

A=new float[row];

if (A==NULL)

{

puts("Memory allocation error."); exit(0);

};

return A;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Memory freeing \*\*\*\*\*/

void DeleteMatrix( float \*\*A, int row)

{ int i;

for (i=0; i<row; i++) delete A[i];

delete A;

};

void DeleteVector( float \*A)

{ delete A; };

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Randomization \*\*\*\*\*/

void BuildMatrix(float \*\*A, int row,int column) //

{

int i,j;

for (i=0; i<row; i++)

for (j=0; j<column; j++) A[i][j]=rand()%10;

};

void BuildVector(float \*A,int column)

{

int j;

for (j=0; j<column; j++) A[j]=rand()%10;

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Printing / File out \*\*\*\*\*/

void PrintMatrix(float \*\*A)

{

int i,j;

for (i=0; i<rows; i++)

{

for (j=0; j<rows; j++) printf("%12.2e",A[i][j]);

printf("\n");

};

printf("-------\n");

};

void PrintMatrix\_F(float \*\*A)

{

int i,j;

for (i=0; i<rows; i++)

{

for (j=0; j<rows; j++) fprintf(OutFile,"%12.2e",A[i][j]);

fprintf(OutFile,"\n");

};

fprintf(OutFile,"-------\n");

};

void PrintColumn(float \*A)

{

int j;

for (j=0; j<rows; j++) printf("%12.2e \n",A[j]);

printf("-------\n");

};

void PrintColumn\_F(float \*A)

{

int j;

for (j=0; j<rows; j++) fprintf(OutFile,"%12.2e \n",A[j]);

fprintf(OutFile,"-------\n");

};

void PrintRow(float \*A)

{

int j;

for (j=0; j<rows; j++) printf("%12.2e ",A[j]);

printf("\n-------\n");

};

void PrintRow\_F(float \*A)

{

int j;

for (j=0; j<rows; j++) fprintf(OutFile,"%12.2e ",A[j]);

fprintf(OutFile,"\n-------\n");

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Operations \*\*\*\*\*/

void MultMatrixByNumber(float \*\*A, float number)

{

int i,j;

for (i=0; i<rows; i++)

for (j=0; j<rows; j++)

A[i][j]=A[i][j]\*number;

};

void MultMatrixByMatrix(float \*\*A, float \*\*B, float \*\*C)

{

int i,j,k;

float x;

for (i=0; i<rows; i++)

{

for (j=0; j<rows; j++)

{

x=0;

for (k=0; k<rows; k++) x=x+A[i][k]\*B[k][j];

C[i][j]=x;

};

};

};

void MultRowByColumn(float \*A, float \*B, float \*C)

{

int j;

float x;

x=0;

for (j=0; j<rows; j++) x=x+A[j]\*B[j];

\*C=x;

};

void MultMatrixByColumn(float \*\*A, float \*B, float \*C)

{

int i,j;

float x;

for (i=0; i<rows; i++)

{

x=0;

for (j=0; j<rows; j++) x=x+A[i][j]\*B[j];

C[i]=x;

};

};

void MultRowByMatrix(float \*A, float \*\*B, float \*C)

{

int i,j;

float x;

for (j=0; j<rows; j++)

{

x=0;

for (i=0; i<rows; i++) x=x+A[i]\*B[i][j];

C[i]=x;

};

};

void AddMatrix(float \*\*A, float \*\* B, float \*\*C)

{

int i,j;

for (i=0; i<rows; i++)

for (j=0; j<rows; j++) C[i][j]=A[i][j]+B[i][j];

};

void AddVectors(float \*A, float \* B, float \*C)

{

int i;

for (i=0; i<rows; i++) C[i]=A[i]+B[i];

};

void main(void)

{

int n, i, j, manual\_input, show\_on\_display, write\_to\_file;

puts("Initial expression: (y1' \* (y2'\*y2\*Y3) + y2'\*Y3^3 + y2'\*Y3) \* Y3\*y1");

cputs("Input n (matrix resolution): ");

scanf("%d",&n);

rows=n;

float \*\*A1 =AllocateMatrix(rows);

float \*\*A2 =AllocateMatrix(rows);

float \*\*A =AllocateMatrix(rows);

float \*\*B2 =AllocateMatrix(rows);

float \*\*C2 =AllocateMatrix(rows);

float \*b =AllocateVector(rows);

float \*b1 =AllocateVector(rows);

float \*c1 =AllocateVector(rows);

float \*\*T1 =AllocateMatrix(rows);

float \*\*T2 =AllocateMatrix(rows);

float \*\*T3 =AllocateMatrix(rows);

float \*t1 =AllocateVector(rows);

float \*t2 =AllocateVector(rows);

float \*t3 =AllocateVector(rows);

float \*t4 =AllocateVector(rows);

float \*t5 =AllocateVector(rows);

float \*ch1 =new float;

for (i=0;i<rows;i++)

{

b[i] = 20/(pow(i,3)+20); // building vector b

for (j=0;j<rows;j++)

C2[i][j] = ( 20 / (pow(i,3) - pow(j,3) + 2) ); // matrix C

}

cputs("Manual input? (y/n) [ ]\b");

manual\_input = Option();

cputs("]; Output to screen? [ ]\b");

show\_on\_display = Option();

cputs("]; Output to file? [ ]\b");

write\_to\_file = Option();

if ( manual\_input == 1 ) // manual input

{

printf("\nEnter the elements of matrix B2\n");

for (i=0;i<rows;i++)

for (j=0;j<rows;j++) scanf("%f",&B2[i][j]);

printf("Enter the elements of matrix A2\n");

for (i=0;i<rows;i++)

for (j=0;j<rows;j++) scanf("%f",&A2[i][j]);

printf("Enter the elements of matrix A1\n");

for (i=0;i<rows;i++)

for (j=0;j<rows;j++) scanf("%f",&A1[i][j]);

printf("Enter the elements of matrix A\n");

for (i=0;i<rows;i++)

for (j=0;j<rows;j++) scanf("%f",&A[i][j]);

printf("Enter the elements of vector b1\n");

for (i=0;i<rows;i++) scanf("%f",&b1[i]);

printf("Enter the elements of vector c1\n");

for (i=0;i<rows;i++) scanf("%f",&c1[i]);

}

else // randomization

{

BuildMatrix(B2,rows,rows); BuildMatrix(A2,rows,rows);

BuildVector(b1,rows); BuildVector(c1,rows);

BuildMatrix(A1,rows,rows); BuildMatrix(A,rows,rows);

};

// Stage 1

MultMatrixByNumber(B2,-1.0); // B2 = B2\*(-1)

MultMatrixByMatrix(A2,C2,T1); // T1 = A2\*C2

for (i=0;i<rows;i++) t1[i]=20\*b1[i]-c1[i]; // t1 = 20\*b1 - c1

if ( show\_on\_display == 1 )

{ printf("\nMatrix A2\*C2 :\n");

PrintMatrix(T1);

printf("\nColumn 20\*b1-c1 :\n");

PrintColumn(t1);

};

if ( write\_to\_file == 1 )

{ fprintf(OutFile,"\nMatrix A2\*C2 :\n");

PrintMatrix\_F(T1);

fprintf(OutFile,"\nColumn 20\*b1-c1 :\n");

PrintColumn\_F(t1);

};

// Stage 2

AddMatrix(T1,B2,T2); // T2 = A2 \* C2 - B2

MultMatrixByColumn(A1,t1,t2); // t2 = A1\*(20\*b1-c1)

MultMatrixByColumn(A,b,t1); // t1 = A\*b

if ( show\_on\_display == 1 )

{

printf("\nMatrix Y3=A2 \* C2 - B2 :\n"); PrintMatrix(T2);

printf("\nColumn y2=A1\*(20\*b1-c1) :\n"); PrintColumn(t2);

printf("\nColumn y1=A\*b :\n"); PrintColumn(t1);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nMatrix Y3=A2 \* C2 - B2 :\n");PrintMatrix\_F(T2);

fprintf(OutFile,"\nColumn y2=A1\*(20\*b1-c1) :\n");PrintColumn\_F(t2);

fprintf(OutFile,"\nColumn y1=A\*b :\n"); PrintColumn\_F(t1);

};

// Stage 3

MultMatrixByMatrix(T2,T2,T1); // T1 = Y3^2

if ( show\_on\_display == 1 )

{

printf("\nMatrix Y3^2 :\n"); PrintMatrix(T1);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nMatrix Y3^2 :\n"); PrintMatrix\_F(T1);

};

// Stage 4

MultRowByColumn(t2,t2,ch1); // ch1= y2'\*y2

MultMatrixByMatrix(T1,T2,T3); // T3 = Y3^3

if ( show\_on\_display == 1 )

{

printf("\nNumber y2'\*y2 : %f",\*ch1);

printf("\nMatrix Y3^3 :\n"); PrintMatrix(T3);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nNumber y2'\*y2 : %f",\*ch1);

fprintf(OutFile,"\nMatrix Y3^3 :\n"); PrintMatrix\_F(T3);

};

// Stage 5

MultMatrixByColumn(T2,t1,t3); // t3 = Y3\*y1

MultRowByMatrix(t2,T3,t4); // t4 = y2'\*Y3^3

MultRowByMatrix(t2,T2,t5); // t5 = y2'\*Y3

MultMatrixByNumber(T2,\*ch1); // T2 = y2'\*y2\*Y3

if ( show\_on\_display == 1 )

{

printf("\nColumn Y3\*y1 :\n"); PrintColumn(t3);

printf("\nRow y2'\*Y3^3 :\n"); PrintRow(t4);

printf("\nRow y2'\*Y3 :\n"); PrintRow(t5);

printf("\nMatrix y2'\*y2\*Y3 :\n"); PrintMatrix(T2);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nColumn Y3\*y1 :\n"); PrintColumn\_F(t3);

fprintf(OutFile,"\nRow y2'\*Y3^3 :\n"); PrintRow\_F(t4);

fprintf(OutFile,"\nRow y2'\*Y3 :\n"); PrintRow\_F(t5);

fprintf(OutFile,"\nMatrix y2'\*y2\*Y3 :\n"); PrintMatrix\_F(T2);

};

// Stage 6

MultRowByMatrix(t1,T2,t2); // t2 = y1'(y2'\*y2\*Y3)

AddVectors(t4,t5,t1); // t1 = y2'\*Y3^3 + y2'\*Y3

if ( show\_on\_display == 1 )

{

printf("\nRow y1'(y2'\*y2\*Y3) :\n"); PrintRow(t2);

printf("\nRow y2'\*Y3^3 + y2'\*Y3 :\n"); PrintRow(t1);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nRow y1'(y2'\*y2\*Y3) :\n"); PrintRow\_F(t2);

fprintf(OutFile,"\nRow y2'\*Y3^3 + y2'\*Y3 :\n"); PrintRow\_F(t1);

};

// Stage 7

AddVectors(t1,t2,t4); // t4 = y1'(y2'\*y2\*Y3) + y2'\*Y3^3+y2'\*Y3

if ( show\_on\_display == 1 )

{

printf("\nRow y1'(y2'\*y2\*Y3) + y2'\*Y3^3+y2'\*Y3 :\n");

PrintRow(t4);

};

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\nRow y1'(y2'\*y2\*Y3) + y2'\*Y3^3+y2'\*Y3 :\n");

PrintRow\_F(t4);

};

// Stage 8 (final)

MultRowByColumn(t4,t3,ch1);

if ( show\_on\_display == 1 )

printf("\nResult: (y1'\*(y2'\*y2\*Y3)+y2'\*Y3^3+y2'\*Y3)\*Y3\*y1 = %.3e\n",\*ch1);

if ( write\_to\_file == 1 )

{

fprintf(OutFile,"\n--- Result ---");

fprintf(OutFile,"\n(y1'\*(y2'\*y2\*Y3)+y2'\*Y3^3+y2'\*Y3)\*Y3\*y1 = %.3e",\*ch1);

};

printf("Press any key to exit...\n",\*ch1);

fclose(OutFile);

DeleteMatrix(A,rows); DeleteMatrix(A1,rows); DeleteMatrix(A2,rows);

DeleteMatrix(B2,rows); DeleteMatrix(C2,rows); DeleteMatrix(T1,rows);

DeleteMatrix(T2,rows); DeleteVector(b1); DeleteVector(c1);

DeleteVector(b); DeleteVector(t1); DeleteVector(t2);

DeleteVector(t3); DeleteVector(t4); DeleteVector(t5);

DeleteVector(ch1);

getch();

}

**Результати виконання програми**

Initial expression: (y1' \* (y2'\*y2\*Y3) + y2'\*Y3^3 + y2'\*Y3) \* Y3\*y1. [V 20]

Input n (matrix resolution): 4

Manual input? (y/n) [y]; Output to screen? [y]; Output to file? [y]

Matrix B2

5 5 5 5

5 5 5 5

5 5 5 5

5 5 5 5

Matrix A2

5 10 15 20

25 30 35 40

45 50 55 60

65 70 75 80

Matrix A1

5 5 5 5

5 5 5 5

5 5 5 5

5 5 5 5

Matrix A

5 10 15 20

25 30 35 40

45 50 55 60

65 70 75 80

Vector (column) b1

5

10

15

20

Vector (column) c1

5

10

15

20

Column b:

9.52e-001

7.14e-001

4.26e-001

2.38e-001

Matrix C2:

1.00e+001 -4.00e+000 -8.33e-001 -3.28e-001

2.22e+000 1.00e+001 -1.18e+000 -3.70e-001

7.14e-001 9.52e-001 1.00e+001 -5.71e-001

3.08e-001 3.45e-001 5.13e-001 1.00e+001

Matrix A2\*C2 :

8.91e+001 1.01e+002 1.44e+002 1.86e+002

3.54e+002 2.47e+002 3.14e+002 3.61e+002

6.19e+002 3.93e+002 4.84e+002 5.35e+002

8.84e+002 5.39e+002 6.55e+002 7.10e+002

-------

Column 20\*b1-c1 :

9.50e+001

1.90e+002

2.85e+002

3.80e+002

-------

Matrix Y3=A2 \* C2 - B2 :

8.41e+001 9.62e+001 1.39e+002 1.81e+002

3.49e+002 2.42e+002 3.09e+002 3.56e+002

6.14e+002 3.88e+002 4.79e+002 5.30e+002

8.79e+002 5.34e+002 6.50e+002 7.05e+002

-------

Column y2=A1\*(20\*b1-c1) :

4.75e+003

4.75e+003

4.75e+003

4.75e+003

-------

Column y1=A\*b :

2.30e+001

6.97e+001

1.16e+002

1.63e+002

-------

Matrix Y3^2 :

2.85e+005 1.82e+005 2.26e+005 2.51e+005

6.16e+005 4.02e+005 5.03e+005 5.64e+005

9.47e+005 6.22e+005 7.80e+005 8.77e+005

1.28e+006 8.42e+005 1.06e+006 1.19e+006

-------

Number y2'\*y2 : 90250000.00

Matrix Y3^3 :

4.47e+008 2.93e+008 3.67e+008 4.13e+008

9.97e+008 6.53e+008 8.18e+008 9.19e+008

1.55e+009 1.01e+009 1.27e+009 1.42e+009

2.10e+009 1.37e+009 1.72e+009 1.93e+009

-------

Column Y3\*y1 :

5.43e+004

1.19e+005

1.83e+005

2.48e+005

-------

Row y2'\*Y3^3 :

2.42e+013 1.58e+013 1.98e+013 2.23e+013

-------

Row y2'\*Y3 :

9.15e+006 5.99e+006 7.49e+006 8.42e+006

-------

Matrix y2'\*y2\*Y3 :

7.59e+009 8.68e+009 1.26e+010 1.63e+010

3.15e+010 2.19e+010 2.79e+010 3.21e+010

5.54e+010 3.50e+010 4.33e+010 4.79e+010

7.93e+010 4.82e+010 5.86e+010 6.36e+010

-------

Row y1'(y2'\*y2\*Y3) :

2.17e+013 1.36e+013 1.68e+013 1.85e+013

-------

Row y2'\*Y3^3 + y2'\*Y3 :

2.42e+013 1.58e+013 1.98e+013 2.23e+013

-------

Row y1'(y2'\*y2\*Y3) + y2'\*Y3^3+y2'\*Y3 :

4.59e+013 2.95e+013 3.66e+013 4.08e+013

-------

Result: (y1'\*(y2'\*y2\*Y3)+y2'\*Y3^3+y2'\*Y3)\*Y3\*y1 = 2.282e+019

**Висновок:** Виконуючи дану лабораторну роботу, я ознайомився із використанням паралелізму на рівні підзадач. Обмін даними відбувається через використання спільних змінних. Присутня залежність даних між різними рівнями декомпозиції, але в межах одного рівня її немає. Є залежність за керуванням, оскільки послідовність обчислювального процесу наперед однозначно відома. Залежність за ресурсами та вводом/виводом може бути визначена лише у відношенні до певної обчислювальної системи.