Mirette boules

ID:5515

Grp2

Lab2 repot

Matrix multiplication

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**//function that do multiplication (a thread for every row)**

void \*multbyrow(void \*th)

{

int core = step\_i++;

// Each thread computes 1 row of matrix multiplication

for (int i = core \* row1 /row1; i < (core + 1) \* row1 / row1; i++)

for (int j = 0; j < column2; j++)

for (int k = 0; k <column1; k++)

matrix[i][j] += mat1[i][k] \* mat2[k][j];

pthread\_exit(NULL);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**//funtction that do multiplication (element by element)for each element a thread)**

void \*multbyelement(void \*th)

{

int k,sum=0;

for(k=0;k<column1; k++)

sum+=mat1[i][k]\*mat2[k][j];

matrix2[i][j]=sum;

if(i<row1)

{

if(j<column2)

{

j++;

}

else

{

i++;

j=0;

}

}

pthread\_exit(NULL);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{ //reading matrices from input file

FILE \*f,\*fout;

f=fopen("input.txt","r");

int i,j;

clock\_t start,end;

fscanf(f,"%d %d \n",&row1,&column1);

for(i=0; i<row1; i++)

for(j=0; j<column1; j++)

fscanf(f,"%d ",&mat1[i][j]);

fscanf(f,"%d %d \n",&row2,&column2);

for(i=0; i<row2; i++)

for(j=0; j<column2; j++)

fscanf(f,"%d ",&mat2[i][j]);

//creating threads for each row and waiting for them

int k,v;

start=clock();

pthread\_t multhrea[row1];

for(v=0; v<row1; v++)

{

pthread\_create(&multhrea[v],NULL,multbyrow,NULL);

}

for(k=0; k<row1; k++)

pthread\_join(multhrea[k],NULL);

end=clock();

time1=((float)(end-start))/CLOCKS\_PER\_SEC;

//creating threads for each element and waiting for them

int y,x,z=row1\*column2+2,c;

start=clock();

pthread\_t multhread2[z];

for(c=0;c<z; c++)

pthread\_create(&multhread2[c],NULL,multbyelement,NULL);

int n;

for(n=0; n<z; n++)

pthread\_join(multhread2[n],NULL);

end=clock();

time2=((float)(end-start))/CLOCKS\_PER\_SEC;

//printing the resultant matix and elapsed time in an output file

fout=fopen("output.txt","w+");

fprintf(fout,"result matrix row by row: \n");

for(i=0; i<row1; i++)

{

for(j=0; j<column2; j++)

fprintf(fout,"%d ",matrix[i][j]);

fprintf(fout,"\n");

}

fprintf(fout,"elappsed time= %f",time1);

fprintf(fout,"\n");

fprintf(fout,"result matrix element by element: \n");

for(i=0; i<row1; i++)

{

for(j=0; j<column2; j++)

fprintf(fout,"%d ",matrix2[i][j]);

fprintf(fout,"\n");

}

fprintf(fout,"elappsed time= %f\n",time2);

fprintf(fout,"the difference in time is = %f ",(time2-time1));

fclose(f);

fclose(fout);

return(0);

}

Output file:



Merge sort

//struct to hold first and last index

struct node {

int left;

int right;

} ;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Function to merge back array in a sorted way**

void merge(int arr[], int l, int m, int r)

{

int n1 = m - l + 1;

int n2 = r - m;

// Create temp arrays

int L[n1], R[n2];

// Copy data to temp arrays L[] and R[]

for (int i = 0; i < n1; i++)

L[i] = arr[l + i];

for (int j = 0; j < n2; j++)

R[j] = arr[m + 1 + j];

// Merge the temp arrays back into arr[l..r]

// Initial index of first subarray

int i = 0;

// Initial index of second subarray

int j = 0;

// Initial index of merged subarray

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

// Copy the remaining elements of

// L[], if there are any

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

// Copy the remaining elements of

// R[], if there are any

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**//function that handle multithreaded merge sort**

void \* mergeSort(void \*a){

pthread\_t threadleft;

pthread\_t threadright;

node \*p = (node \*)a;

node n1, n2;

int m = (p->left+p->right)/2;

if(p->left>= p->right){

return NULL;//returns recursively

}

n1.left = p->left;

n1.right =m;

n2.left= m+1;

n2.right =p->right;

//hena anadiha b thread 1st half w a join

pthread\_create(&threadleft, NULL, mergeSort,&n1);

pthread\_join(threadleft, NULL);

//anadiha b thread tani half w a join

pthread\_create(&threadright, NULL, mergeSort,&n2);

pthread\_join(threadright, NULL);

//merging the 2 threads

merge(arr,p->left,m,p->right);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{ int i =0;

int x;

ifstream is("input.txt");

int size;

is >> x;

size=x;

arr=(int\*)malloc(sizeof(int)\*size);

while (is >> x)

// and read integer from file

arr[i++] = x;

is.close();

node m;

m.left=0;

m.right=size-1;

pthread\_t thread;

pthread\_create(&thread, NULL, mergeSort,&m);

pthread\_join(thread, NULL);

// displaying sorted array

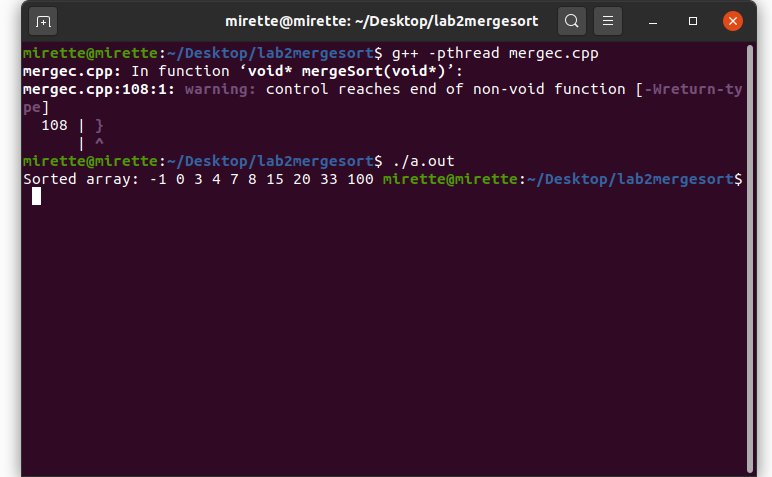
cout << "Sorted array: ";

for (int i = 0; i < size ; i++)

cout << arr[i] << " ";

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*