Info

Exam in IN4200 spring semester 2022

Assistance: One two-sided A4 page with hand-written notes, plus a calculator. (A calculator is also available in Inspera.) No other assistance is allowed.

All the exam questions should be answered with keyboard and mouse. No need to use sketching paper.

Weighting of questions:
Question 1: 5 points
Question 2: 10 points
Question 3: 10 points
Question 4: 10 points
Question 5: 10 points
Question 6: 5 points
Question 7: 10 points
Question 8: 8 points
Question 9: 12 points
Question 10: 10 points
Question 11: 10 points

1 Cache line

Give a short explanation about "cache line".

Fill in your answer here

Format

Words: 0

Maximum marks: 5

2 Performance improvement

Please name one data-access optimization technique that is very effective for improving the performance of the following code when n is large. Explain your answer.

```
for (i=0; i<n; i++) {
  double t = 0.0;
  for (j=0; j<n; j++)
     t += a[i][j]*b[j];
  c[i] = t;
}
```

We assume that a is a row-major 2D array of dimension $n \times n$, while b, c are 1D arrays of length n.

Fill in your answer here

Format

Words: 0

Maximum marks: 10

3 Estimating time usage

Assume that we have a CPU with 40 GB/s as the theoretical memory bandwidth and 100 GFLOP/s (in double precision) as the theoretical peak floating-point performance. It is also known that each cache line is 64 bytes in length. Can you estimate how much time the following code needs on this CPU, when $n = 10^5$? (If needed, you can make additional assumptions about the CPU.)

```
#define stride 9


double s = 0.0;
for (i=0; i<n; i++)
  for (j=0; j<n; j+=stride)
     s += a[i][j]*a[i][j];
```

We assume that a is a row-major 2D array of dimension $n \times n$ and contains double-precision values.

Fill in your answer here

Format

Words: 0

Maximum marks: 10

4 Difficulty for straightforward OpenMP parallelization

Explain why the following code segment cannot be directly parallelized by only inserting "**#pragma omp parallel for**" before either of the two for-loops.

```
for (i=1; i<n-1; i++)
  for (j=1; j<n-1; j++)
    a[i][j] = 0.25*(a[i-1][j]+a[i][j-1]+a[i][j+1]+a[i+1][j]);
```

We assume that a is a row-major 2D array of dimension $n \times n$

Fill in your answer here

Format

Words: 0

Maximum marks: 10

5 OpenMP parallelisation

Parallelize the code in the previous question by using OpenMP. (**Hint**: You need to first restructure the code and identify "wave fronts". Remember to use comments for explaining the parallelization.)

Fill in your answer here

```
1
```

Maximum marks: 10

6 Speedup discussion

Discuss the achievable speedup as a function of the number of threads.
Fill in your answer here

Format

Words: 0

Maximum marks: 5

7 Understanding an MPI program

What will be the result of running the following MPI program using 8 MPI processes? (See the attachment for a quick reference of important MPI functions.)

```
#include<stdio.h>
#include<mpi.h>
int main(int argc, char *argv[])
{
  int rank, size;

  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  int sum = rank;
  int temp, stride = 1;

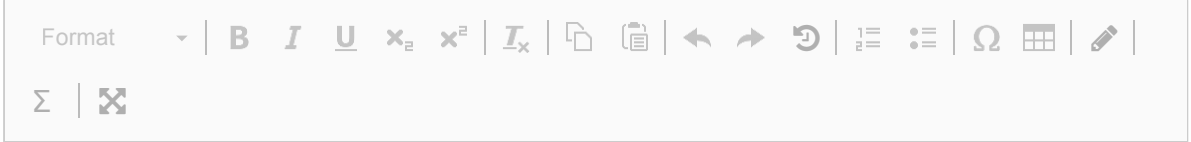
  while(stride <= size/2) {
    if (rank<size-stride) {
      MPI_Send(&sum, 1, MPI_INT, rank+stride,0,MPI_COMM_WORLD);
    }

    if (rank>=stride) {
      MPI_Recv(&temp, 1, MPI_INT, rank-stride,0,MPI_COMM_WORLD,&status);
      sum += temp;
    }

    stride = 2 * stride;
  }

  printf("On rank <%d>, sum=%d\n", rank, sum);
  MPI_Finalize();
  return 0;
}
```

Fill in your answer here



Words: 0

Maximum marks: 10

8 Implementation of function find_min_with_location

Write a serial C function with the following syntax:

```
void find_min_with_index (const int *input_array, const int length, int *min_value, int *index);
```

The purpose of the function is to find the minimum value inside **input_array**, as well as the index of the minimum value found. Here is a concrete small example of usage:

```
int an_array[6] = {19, 3, -2, 5, -2, 0};
int min_v, ind;
find_min_with_index (an_array, 6, &min_v, &ind);
printf("minimum value=%d, index=%d\n", min_v, ind);
```

The result of **printf** will be
minimum value=-2, index=2

Note: if there are multiple values having the same minimum value, the lowest index is to be found.

Fill in your answer here

```
1
```

Maximum marks: 8

9 OpenMP parallelization of 'find_min_with_index'

Implement an OpenMP parallelization of function 'find_min_with_index' from the previous question:

Fill in your answer here

```
1
```

Maximum marks: 12

10 Denoising a color image

We want to extend the second mandatory project such that a color image can be denoised. More specifically, we assume that there is one 3D array of single-precision numerical values named "color_image_data_bar" and another 3D array named "color_image_data". The first two dimensions are the same as in the second project, that is, as the number of vertical pixels and horizontal pixels in the image. The third dimension is 3, containing the "red", "green", "blue" components of each pixel. Both the 3D arrays have an underlying 1D contiguous storage.

You are required to implement a serial function as follows:

```
void denoising_iteration (int m, int n, float ***color_image_data_bar, float ***color_image_data, float kappa)
```

which carries out one iteration of isotropic diffusion. The effect is that the red, green, blue components are denoised by

$$\bar{u}_{i,j,r} = u_{i,j,r} + \kappa(u_{i-1,j,r} + u_{i,j-1,r} - 4u_{i,j,r} + u_{i,j+1,r} + u_{i+1,j,r})$$
$$\bar{u}_{i,j,g} = u_{i,j,g} + \kappa(u_{i-1,j,g} + u_{i,j-1,g} - 4u_{i,j,g} + u_{i,j+1,g} + u_{i+1,j,g})$$
$$\bar{u}_{i,j,b} = u_{i,j,b} + \kappa(u_{i-1,j,b} + u_{i,j-1,b} - 4u_{i,j,b} + u_{i,j+1,b} + u_{i+1,j,b})$$

where the indices are r=0, g=1, b=2.

Fill in your answer here

```
1
```

Maximum marks: 10

11 Programming MPI communication

Now we want to use MPI processes to parallelize the work of "one_denoising_iteration". Assume that a color image is already partitioned vertically such that each MPI process is assigned with 'my_m' rows of pixels (including two ghost rows). The numbers of pixel columns per process is still 'n'.

Implement the necessary MPI communication needed before every MPI process can call 'one_denoising_iteration' restricted to its partitioned data (that is, 'my_m' replaces 'm', 'my_color_image_data_bar' replaces 'color_image_data_bar', and 'my_color_image_data'

'my_color_image_data_bar' replaces 'color_image_data_bar', and 'my_color_image_data' replaces 'color_image_data').

(See the attachment for a quick reference of important MPI functions.)

If needed, please use comments to provide necessary explanations.

Fill in your answer here

```
1
```

Maximum marks: 10