TDT4200 Problem set 1

Pass/fail evaluation

Deadline: 30/08, 20:00

Answers should be submitted on It's Learning

Task 1 - Theory

1. Draw a 4D hypercube. Label the nodes with bit patterns (e.g. '0100') such that neighbouring nodes differ only in one bit.
2. Map the hypercube to a 4-level binary tree. Label the root '0000', and label the remaining nodes such that a left child has the same label as its parent, and a right node is labeled by one of the parents neighbours in the hypercube. All the leaf nodes should have different labels.

Task 2 - C-programing

In this task, you should expand 'ps1.c' from the provided archive 'ps1.tar.gz'. You should not use C99 support for complex numbers (from `<complex.h>`).

1. Create a `struct` for complex numbers called `complex_t` (use `typedef`), with integer fields for the real and imaginary parts.
2. Implement the function `multiply_complex`. The function takes two complex numbers as arguments, and should return their product. Complex multiplication is defined as:

$$(a + bi)(c + di) = (ac - bd) + (bc + ad)i$$

3. Implement the function `divide_complex`. The function should store the quotient of the two first arguments in the complex number pointed to by the third. Complex division is defined as:

$$\frac{a + bi}{c + di} = \left(\frac{ac + bd}{c^2 + d^2}\right) + \left(\frac{bc - ad}{c^2 + d^2}\right) i$$

4. Implement the function `create_random_complex_array`. The function takes a single integer as argument, and should return an array of complex numbers, with that number of elements (use `malloc()`). The function shoud initialize the real and imaginary parts of the complex numbers to values between -1000 and 1000.
5. Implement the function `multiply_complex_arrays`. The function takes two equaly sized arrays of complex numbers, as well as their size as input, and should return a new array, where each element is the product of the corresponding elements of the two input arrays. That is, the function should perform pairwise multiplication, the n'th element of the output array should be

the product of the n'th elements in the input arrays. Use `multiply_complex` for the multiplication.

6. Implement the function `divide_complex_arrays`. The function takes two equaly sized arrays of complex numbers, as well as their size as input, and should return a new array, where each element is the quotient of the corresponding elements of the two input arrays. That is, the function should perform pairwise division, the n'th element of the output array should be the quotient of the n'th elements in the input arrays. Use `divide_complex` for the division.
7. Make sure to free all allocated memory.

Task 3 - MPI-programing

In this task, you should expand 'pi.c' from the provided archive 'ps1.tar.gz'. You should compile and run the program on clustis (clustis3.idi.ntnu.no). For more information about how to access clustis, and compile and run MPI programs, see the recitation slides.

The program finds an approximation of π by evaluating the first terms of the infinite sum:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \sum_{n=0}^{\infty} (-1)^n \frac{1}{2n+1}$$

Each process calculates 1000 terms from the sequence. That is, rank 0 evaluates the sum for values of n from 0 to 999, rank 1 evaluates for n between 1000 and 1999 and so on.

Your task is to complete the program by doing the following:

1. Insert appropriate code for doing MPI initialization and shutdown.
2. Insert code for collecting the sum of all partial calculations on rank 0 (Using `MPI_Send` and `MPI_Recieve`). The collected sum should be put in the variable `result`.

The program should work correctly with 8 processes (`mpiexec -n 8 pi`).