FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Parallel algorithms for the analysis and synthesis of data on the assignments No.12, 13, 14

Performed by
Ivan Dubinin
J4132c

St. Petersburg
2021

Assignment 12

MPI allows for non-blocking operations to form whole packets of requests forcommunication operations MPI_Send_init and MPI_Recv_init, which are started by theMPI_Start or MPI_Startall functions. Checking for completion of execution is performed byconventional means using the functions of the WAIT and TEST families.

Find and fix errors in Assignment12.c, add the for loop. When should you use a loop?

Listing of the program

Cascade variant

Description

In functions MPI_Send_init, MPI_Recv_initand MPI_Request_free we need to pass not Request object, but pointer to it (adress in our case is taken by &).The same we need to do with rbuf and sbuf in MPI_Send_init and MPI_Recv_init. (See here: https://www.mpich.org/static/docs/v3.1.x/www3/MPI_Send_init.html) In first for loop I have decided to add determination of I variable.

We can use loops for i freeing requests.Function MPI_Startall and MPI_Waitall can be also in cycle, depending on our purposes (So I have left them in a cycle).

Example of launch parameters and output

Not here :)

Assignment 13

Find out which process will perform the multiplication of two 500x500 square matrices faster.Complete the code Assignment13.c. You can use the necessary code from the previous assignments.

Listing of the program

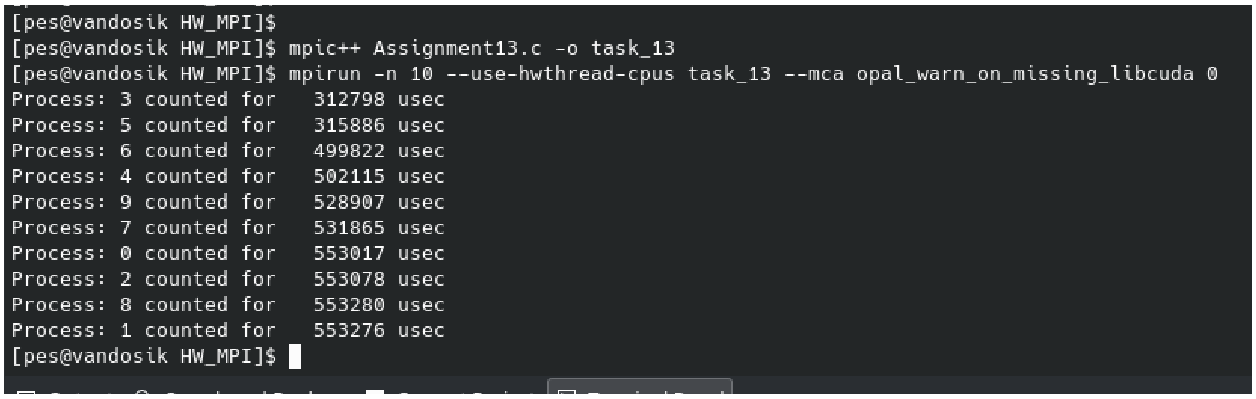
See it in my github repo

Description

Each proc prepares matrix 500*500 with random integers in such a way that matrixes are identical for all the processes. This is because function rand() generates the same values in each proc (seed() was not called).

After preparations all the procs are taken togeather at MPI_Barrier, so each of them starts calculations simultaneously with others. After calculations each proc prints to console his rank and elapsed time in usec.

Example of launch parameters and output



```
[pes@vandosik HW_MPI]$
[pes@vandosik HW_MPI]$ mpic++ Assignment13.c -o task_13
[pes@vandosik HW_MPI]$ mpirun -n 10 --use-hwthread-cpus task_13 --mca opal_warn_on_missing_libcuda 0
Process: 3 counted for   312798 usec
Process: 5 counted for   315886 usec
Process: 6 counted for   499822 usec
Process: 4 counted for   502115 usec
Process: 9 counted for   528907 usec
Process: 7 counted for   531865 usec
Process: 0 counted for   553017 usec
Process: 2 counted for   553078 usec
Process: 8 counted for   553280 usec
Process: 1 counted for   553276 usec
[pes@vandosik HW_MPI]$ 
```

Assignment 14

Understand the new functions in Assignment14.c.

Create your own global function for finding the maximum element, compare the correctness of execution with the MPI_MAX operation in the MPI_Reduce() function.

Listing of the program

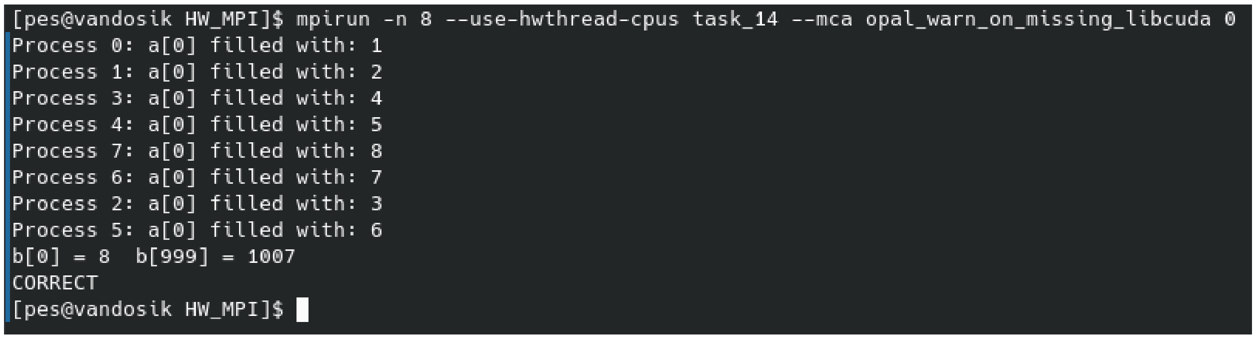
See it in my github repo

Description

Program created n processes. Each process fills his own array on integers. Than MPI_Reduce with max_of_pair() function as operator. This function finds tha maximum between elements of two arrays with same index. So after calling MPI_Reduce the root proc must have an array of all maximum elements with distinct index among all the arrays (in fact it is array with max rank).

The obtained results were checked with ones got using MPI_MAX operator. As one can see on the picture they are correct.

Example of launch parameters and output



```
[pes@vandosik HW_MPI]$ mpirun -n 8 --use-hwthread-cpus task_14 --mca opal_warn_on_missing_libcuda 0
Process 0: a[0] filled with: 1
Process 1: a[0] filled with: 2
Process 3: a[0] filled with: 4
Process 4: a[0] filled with: 5
Process 7: a[0] filled with: 8
Process 6: a[0] filled with: 7
Process 2: a[0] filled with: 3
Process 5: a[0] filled with: 6
b[0] = 8  b[999] = 1007
CORRECT
[pes@vandosik HW_MPI]$ 
```