Homework 3: MPI Programming

Introduction to Big Data Systems course

Due: March 21, 2021 23:59 China time. Late submission results in lower (or even no) scores.

For questions or concerns, contact TA (Jiping Yu) by WeChat. Or send an email to yjp19@mails.tsinghua.edu.cn if you could not use WeChat.

Overview

In this assignment, you will implement a reduction algorithm on a cluster of nodes. You should use **MPI** to implement it.

Background

`MPI_Reduce()` combines the elements provided in the input buffer of each process in the group, using the operation `op` , and returns the combined value in the output buffer of the process with rank root.

For example, there are 3 processes in a group, ranking from 0 to 2, and the process 0 is the root process, each process has an array, the length of the array is 5:

```
Array of process 0: [ 1 2 3 4 5 ]
Array of process 1: [ 1 1 1 1 1 ]
Array of process 2: [ 2 2 2 2 2 ]
```

Use the array as input and use `MPI_SUM` as the operation of `MPI_Reduce()` , after reducing, the result is:

```
Result array in process 0: [ 4 5 6 7 8 ]
```

Environment

You will need to run code on the server machine finally for this assignment.

You can refer to `server.pdf` about how to connect to the cluster.

For this homework, you need to program in **C++** to use MPI directly. The compiler is GNU C++ compiler (`g++`) version 9.3.0. We use Open MPI 4.0.5 for MPI libraries.

You must load the MPI C++ compiler before compiling and running MPI programs. We use the Spack package manager for loading MPI.

After login to the server, you should load Spack by

```
source /opt/spack/share/spack/setup-env.sh
```

And then load the MPI environment with Spack, by

```
spack load openmpi
```

You can test it with

```
which mpicxx
```

If it displays a path like `/opt/spack/opt/spack/...something...` , you loaded the MPI environment successfully. Otherwise, for example if it displays `/usr/bin/mpicxx` , you did not load it correctly.

Files

We have placed the example code at `/data/hw3_src` . You should copy it to your home (by `cp -r /data/hw3_src ~` for example). Alternatively, you can use `hw3_src.zip` in your local computer.

To compile the code, simply type `make` (after you correctly loaded the MPI environment), resulting in an executable file `reduce` . You can run the command `srun -n 4 ./reduce` , to run the program on 4 nodes, for example.

There is no data file for this homework. Instead, the data is randomly generated. You may view `main.cpp` for details.

Description

Let NP denotes the number of processes (i.e. the number of nodes). You are only required to handle the cases of NP = 2, 4, and 8. It is totally acceptable if your program is incorrect or even crashes for other NP values.

In addition, you may assume the length of the array is one of (1, 16, 256, 4096, 65536, 1048576, 16777216, 268435456) integers. For the 4 smaller sizes (up to 4096), you should make sure your implementation is correct but need not report their performance, since the time needed is very short. For the 4 larger sizes (beginning with 65536), you should make sure the correctness, and also report the time.

You can implement `YOUR_Reduce` in a serial, single-threaded program. Performing a multi-threaded reducing is optional (see Bonus). `YOUR_Reduce` should produce the same result (in process 0) as a normal `MPI_Reduce` would. See the example implementation in `your_reduce.cpp` for more information.

Since you should write your own version to reduce the integers, you must NOT directly use `MPI_Reduce` or similar functions (such as `MPI_Allreduce`) provided by MPI. The only MPI communication functions you may use are `MPI_Send`, `MPI_Recv`, `MPI_Sendrecv`, and asynchronous versions `MPI_Isend` and `MPI_Irecv`. Of course, you are free to use functions which are unrelated to data, such as `MPI_Comm_size`, `MPI_Comm_rank`, and `MPI_Barrier`. Contact TA if you are unsure about whether you could use a certain function.

Hand-in

Files

You should submit a single ZIP file, strictly following the format.

For example, if your cluster username is `2020123456`, you should submit a ZIP file exactly named `hw3_2020123456.zip`. Inside it, there should not be any subdirectories, but it should **only contain these two files**:

- `hw3_2020123456_report.pdf`. Your report in PDF format.
- `hw3_2020123456_your_reduce.cpp`. Your version of the reduction algorithm.

You do not submit other source code, since `your_reduce.cpp` is the only file you may modify (for hand-in). You should make sure we can compile your code, by simply replace the `your_reduce.cpp` and type `make`. The compilation should not issue any errors, and you should make sure your code works correctly.

Submitting

If you have access, submit the ZIP file to the web learning homeworks. Otherwise, email the ZIP file to TA. You should submit before **March 21, 2021** 23:59 China time. Late submission results in lower (or even no) scores.

Scoring

Correctness (20%)

You will get full marks for correctness, if your results are correct.

Performance (40%)

The faster your implementation is, the more scores you can get (if your results are correct).

Report (40%)

Your report should at least contains:

- Detailed description of your implementation.
- Time result of 3 configurations of NP (2, 4, and 8 nodes), and for 4 different array sizes (64K, 1M, 16M, and 256M integers). You should report 3 x 4 = 12 time results. If

you feel the time usage is unstable, run multiple times and report the median.

Bonus (optional, up to +10%)

Normally, each MPI process can utilize 1 CPU core. However, each node has 4 cores, 8 threads.

For the bonus, you should implement a multi-threaded version to reduce the integers. You can use OpenMP or other methods of multi-threading (such as pthread).

Describe your multi-thread version implementation in your writeup. Report 12 time results for 3 NP values (2, 4, and 8 nodes), and for 4 different numbers of threads (1, 2, 3, and 4 threads per node). You may not change the array size; using only 256M is fine. Submit the extra source file of multi-threaded version named like `hw3_2020123456_your_reduce_multithread.cpp` in the ZIP file (so that it contains 3 files in total).

Note: The bonus is extra. You should still implement and submit a single-thread version, and report 24 time results for it. Otherwise you will lose the regular scores!