HPC Tools 2024, Exercise 5 solutions

Bruno Oliveira Cattelan, Laurent Chôné

Problem 1

For this exercise we were interested in comparing formatted vs unformatted output. This is done by using "fwrite" for unformatted and "fprintf" for formatted.

Since "fprintf" needs a hard limit on the data to be printed, we choose to do variable by variable. Other implementations might write the whole array at once. Both are correct, as long as the format vs unformat is respected and the same process is used for both cases.

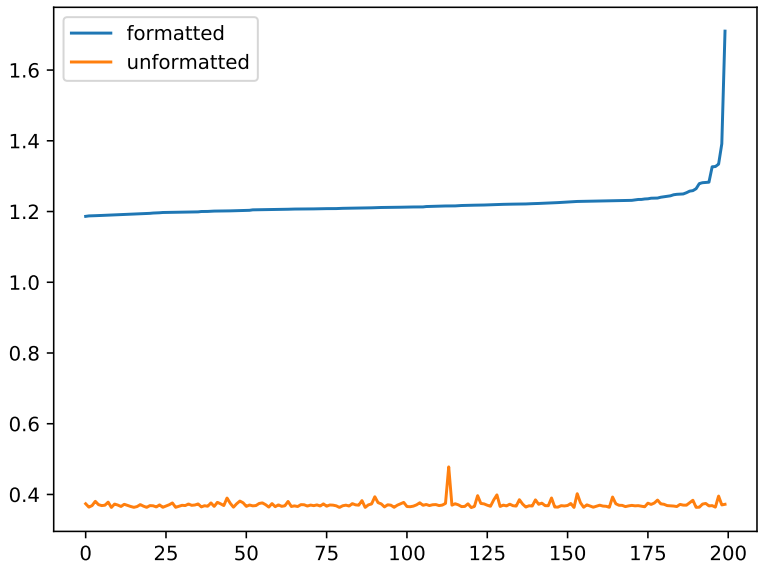

Figure 1: Formatted vs Unformatted output to file

Problem 2

On the y-axis we have the mean runtime for the different cases. On the x-axis as well as by the color we have the different cases analysed in this problem.

We see a considerable difference between omitting the frame and not for optimization level 0. However, as the optimization level increases, this difference reduces to the point of being negligible for level 2.

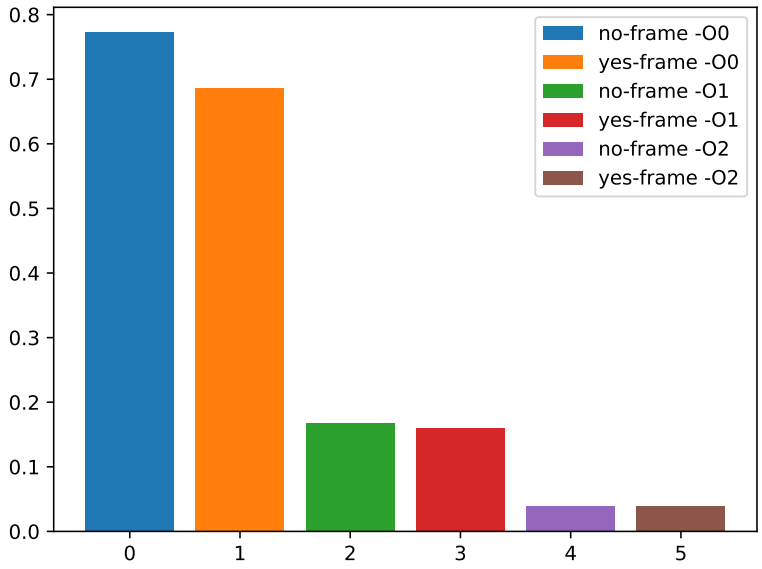

Figure 2: Omit frame results - 1000 repetitions

Problem 3

In this problem, we simply run a parallel program provided. Each MPI rank $\neq 0$ sends the total number of ranks to rank 0.

```
if (id != 0) {
 ...
 rc=MPI_Send(msg,2,MPI_INT,dest_id,tag,MPI_COMM_WORLD);
} else {
  for (i=1; i < ntasks; i++) {
    rc=MPI_Recv(msg,2,MPI_INT,MPI_ANY_SOURCE,tag,MPI_COMM_WORLD,&status);
    ...
  }
}
```

The output produces is of the form:

```
host ukko3-805.local.cs.helsinki.fi

host ukko3-805.local.cs.helsinki.fi
message: 2 4 sender: 2
message: 1 4 sender: 1
message: 3 4 sender: 3

host ukko3-805.local.cs.helsinki.fi

host ukko3-805.local.cs.helsinki.fi
```

Notice that the messages are not received in order, nor are the `print` statements. In particular, the constant `MPI_ANY_SOURCE` causes rank 0 to receive the messages in the order they were emitted. Replacing it with `i` makes it so rank 0 reads the messages in order of rank number.

Problem 4

In this problem we write a program which sends an integer back and forth, incrementing it every time it is received. An example of how to obtain this result in a similar fashion to the previous problem follows (main loop, in C++):

```
for ( int idx = 0 ; idx < exchangeSize * 2 ; idx++)
{
    if ( mpiIdx == idx%2 )
    {
        MPI_Send(&counter, 1, MPI_INT, (idx+1)%2, tag, MPI_COMM_WORLD);
    }
    else if  ( mpiIdx == (idx+1)%2 )
    {
        MPI_Recv(&counter, 1, MPI_INT, idx%2, tag, MPI_COMM_WORLD, &status);
        counter += 1;
    }
}
```

Or equivalently in Fortran:

```
do idx = 0, exchangeSize * 2 - 1
    if ( mpiIdx == MOD(idx,2) ) then
        call mpi_send(counter, 1 , mpi_integer, MOD(idx+1,2), tag,
           mpi_comm_world, mpiError)
    else if ( mpiIdx == MOD(idx+1,2) ) then
        call mpi_recv(counter, 2, mpi_integer, MOD(idx,2), tag,
           mpi_comm_world, status, mpiError)
        counter = counter + 1
    end if
end do
```