Parallel Programing: HW2

Andres Imperial

January 24, 2020

Implementation

I implemented my code by first having srand() init my random number generator. Then I had process 0 start the bomb counter with random number and then send the bomb to a random process. Then all processes are in a loop waiting to receive something. When they receive the bomb they announce it and decrements its counter and send it to another process that is not themselves. If the counter is decremented to zero then the bomb explodes and that process loses and then terminates the rest of the processes. I also put in a small sleep to allow the processes to write out to stdout in the order they received the bomb.

Code:

Listing 1: main.cpp

```
#include <chrono>
#include <ctime>
#include <iostream>
#include <mpi.h>
#include <stdlib.h>
#include <thread>
#include <unistd.h>

#define MCW MPI_COMM_WORLD

using namespace std;

int main(int argc, char **argv){
  // Init random seed
  srand(time(NULL));

  // Variables
  int rank, size;
  int bomb;
  MPI_Init(&argc, &argv);
```

```
MPI_Comm_rank(MCW, &rank);
MPI_Comm_size(MCW, &size);

// process zero will start the bomb
if (rank == 0) {
  bomb = rand() % 100;
  cout << "process 0 generated a bomb with timer at" << bomb << endl;

  while (true) {
    // find next sucker
    int addr = rand() % size;
    // don't send it to yourself
    addr = addr == rank ? (addr + 1) % size : addr;
    MPI_Send(&bomb, 1, MPI_INT, addr, 0, MCW);
    MPI_Recv(&bomb, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW, MPI_STATUS_IGNORE);
    this_thread::sleep_for(std::chrono::milliseconds(10));
    --bomb;

    if (bomb == 0) {
      cout << "BOOM! -- process " << rank << " loses\n";
      MPI_Abort(MCW, 0);
    }
    cout << "process 0 holds the bomb with timer at " << bomb
         << " and tossed it\n";
  }
} else {
  while (true) {
    MPI_Recv(&bomb, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW, MPI_STATUS_IGNORE);
    this_thread::sleep_for(std::chrono::milliseconds(10));
    --bomb;

    if (bomb == 0) {
      cout << "BOOM! -- process " << rank << " loses\n";
      MPI_Abort(MCW, 0);
    }
    cout << "process " << rank << " holds the bomb with timer at " << bomb
         << " and tossed it\n";

    // find next sucker
    int addr = rand() % size;
    // don't send it to yourself
    addr = addr == rank ? (addr + 1) % size : addr;
    MPI_Send(&bomb, 1, MPI_INT, rand() % size, 0, MCW);
  }
}
```

```
    MPI_Finalize();

    return 0;
}
```

How to run:

```
mpic++ main.cpp -o main
mpirun -np 4 -oversubscribe ./main
```

Output

```
process 0 generated a bomb with timer at20
process 1 holds the bomb with timer at 19 and tossed it
process 0 holds the bomb with timer at 18 and tossed it
process 2 holds the bomb with timer at 17 and tossed it
process 0 holds the bomb with timer at 16 and tossed it
process 1 holds the bomb with timer at 15 and tossed it
process 1 holds the bomb with timer at 14 and tossed it
process 1 holds the bomb with timer at 13 and tossed it
process 3 holds the bomb with timer at 12 and tossed it
process 0 holds the bomb with timer at 11 and tossed it
process 1 holds the bomb with timer at 10 and tossed it
process 3 holds the bomb with timer at 9 and tossed it
process 1 holds the bomb with timer at 8 and tossed it
process 2 holds the bomb with timer at 7 and tossed it
process 1 holds the bomb with timer at 6 and tossed it
process 3 holds the bomb with timer at 5 and tossed it
process 1 holds the bomb with timer at 4 and tossed it
process 1 holds the bomb with timer at 3 and tossed it
process 1 holds the bomb with timer at 2 and tossed it
process 2 holds the bomb with timer at 1 and tossed it
BOOM! -- process 1 loses
--------------------------------------------------------------------------
MPI_ABORT was invoked on rank 1 in communicator MPI_COMM_WORLD
with errorcode 0.

NOTE: invoking MPI_ABORT causes Open MPI to kill all MPI processes.
You may or may not see output from other processes, depending on
exactly when Open MPI kills them.
--------------------------------------------------------------------------
```

Order of lines may vary due to parallel processing variability. MPI ABORT kills all other processes that were not taken out by the bomb.