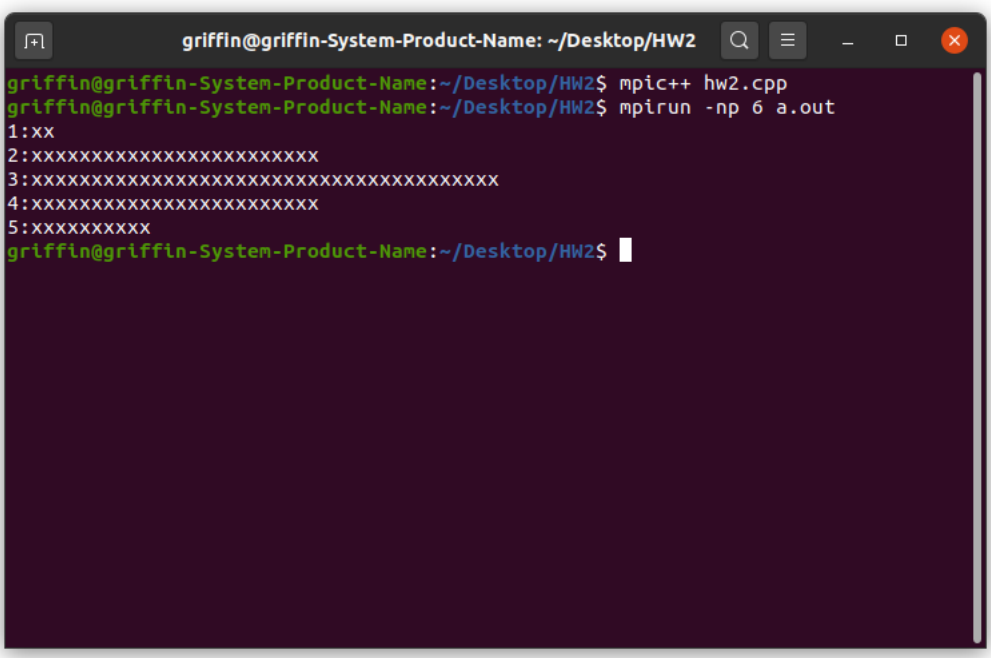
CS 5500 HW2

griffinhackley

January 2021

I realized after coding the entire project that the instructions say to make each process handle a column. This is not how I went about coding the pachinko board, so I hope that is okay. The way I coded this project was to set up a leader and workers. The leader sends each worker a ball and the worker will simulate the ball falling through the board. When a ball gets to the bottom the worker sends a message to the leader telling the leader which bucket the ball landed in. The leader then gives the worker another ball to simulate until there are no balls left. After the leader has sent out all of the balls it waits for the results from each process and records which ones are finished. After all of the processes are finished the leader tells the workers to stop. Afterwards the leader will print out the results.

Sample execution:



Leader Code:

```
void honorableLeader(Ball balls[], int length, int size, int boardSize){
    int score[1+(boardSize*2)];
    fill_n(score, 1+(boardSize*2), 0);
    int done[size];
    int data;
    MPI_Status status;
    int go = 1;

    //if not enough tasks for workers only assign as many as needed
    if(length < size-1){
        cout << "not enough tasks" << endl;
        for(int i = 1; i < length; i++){
            data = i;
            MPI_Send(&data,1,MPI_INT,i,0,MCW);
        }
        for(int i = length; i < size; i++){
            data = -10;
            MPI_Send(&data,1,MPI_INT,i,0,MCW);
            done[i] = 1;
        }
    }

    //give work to all workers
    for(int i = 1; i < size; i++){
        data = i;
        MPI_Send(&data,1,MPI_INT,i,0,MCW);
    }

    //wait for workers to be done and assign more work
    for(int i = size; i<=length-1;  i++){
        MPI_Recv(&data, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW,&status);
        score[data+boardSize]++;
        data=i;
        MPI_Send(&data,1,MPI_INT,status.MPI_SOURCE,0,MCW);
    }

    //wait for all workers to be done
    while(go){
        MPI_Recv(&data, 1, MPI_INT, MPI_ANY_SOURCE, 0, MCW,&status);
        done[status.MPI_SOURCE] = 1;
        score[data+boardSize]++;

        //wait until the last one finishes
        go = 0;
        for(int i = 1; i<size;  i++){
            if(done[i] != 1){
                go = 1;
            }
            data=i;
        }
    }

    //send workers home
    data = -10;
    for(int i = 1; i < size; i++){
        MPI_Send(&data,1,MPI_INT,i,0,MCW);
    }

    //print results
    int counter = 0;
    for(int i = 1; i < (sizeof(score)/sizeof(score[0]))-2; i++){
        if(i%2 != 0){
            i++;
        }

        // divide by 2 because if the number of times the ball needs to fall is even then it will al
        cout << i/2 << ":";

        // comment out to not print number of ball in each bucket
        // cout << score[i];

        for(int j = 1; j<=score[i]; j++){
            cout << "x";
            counter++;
        }
        cout << endl;
    }
}
```

Worker Code:

```c
void gloriousWorker(Ball balls[], int rank, int boardSize){
    int data;
    //give work to glorious workers
    while(1){
        //if terminate code, end process
        MPI_Recv(&data, 1, MPI_INT, 0, 0, MCW,MPI_STATUS_IGNORE);
        if(data == -10){
            break;
        }
        int score = simulate(balls[data], boardSize);
        MPI_Send(&score,1,MPI_INT,0,0,MCW);
    }
}
```