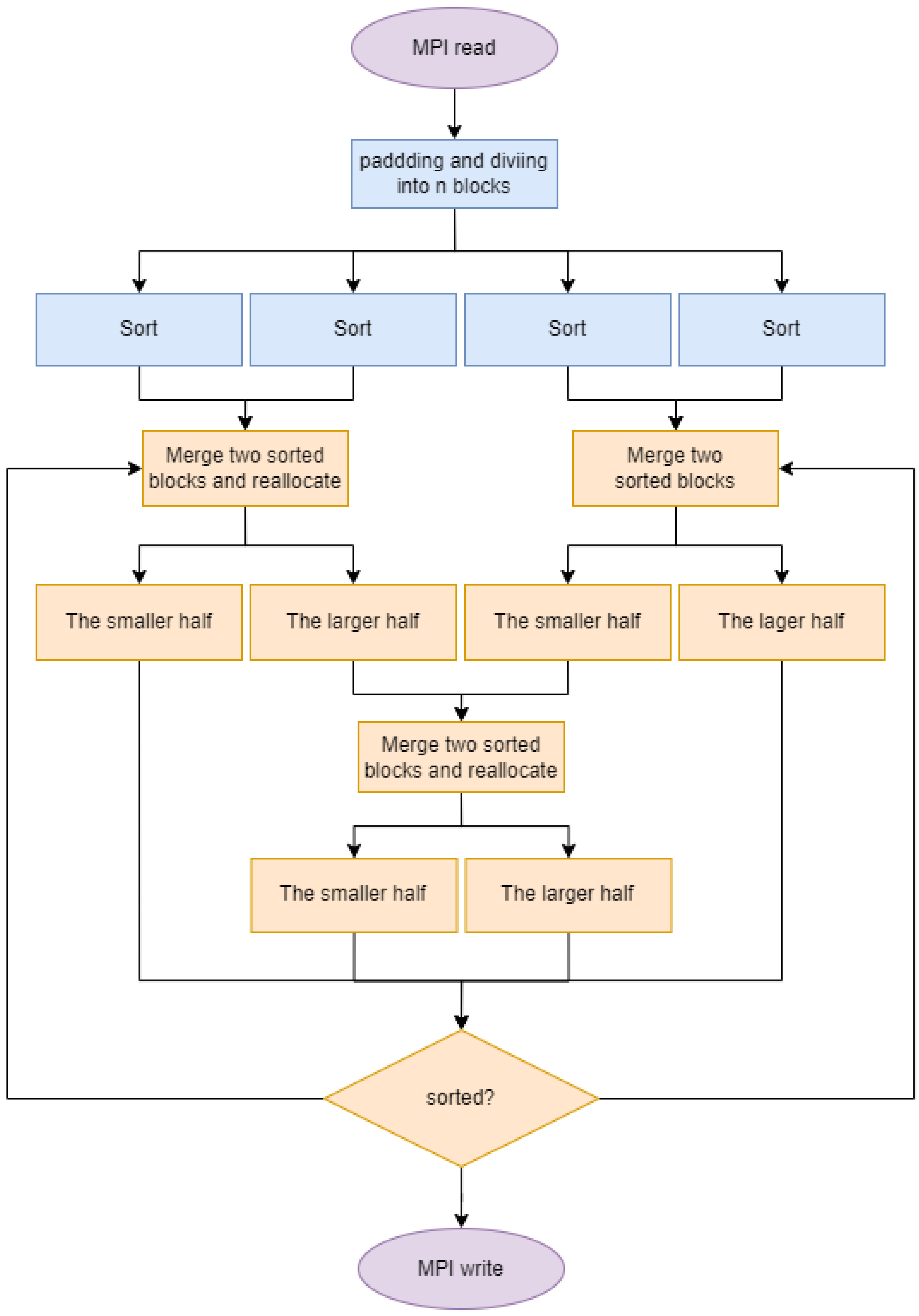
Title: Odd-Even Sort

Name: 曹寓恆

Student ID: 0008-110521167

Implementation

1. Flow chart (assume # Process is 4)



2. **How do you handle an arbitrary number of input items and processes?**
   (1) If the problem size is smaller than lower bound, than it is solved with the root process.
   (2) If the size of the problem is not divisible, then padding several FLT_MAX(defined in cfloat.h) elements.

3. **How do you sort in your program?**
   After using the idea of "merge sorting" to deal with adjacent blocks, then compare to the minimum and maximum of each block to determine if termination or not.

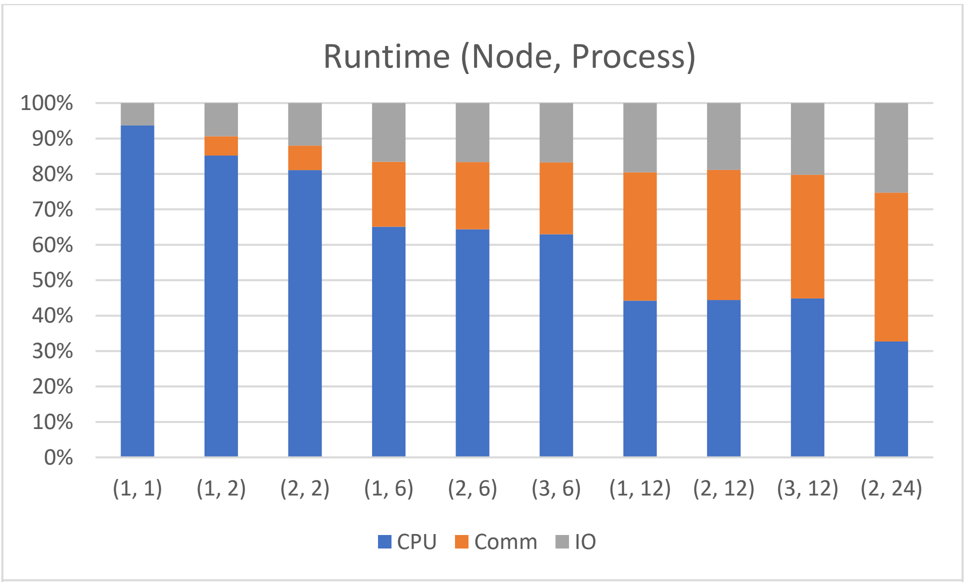
4. **Other efforts you've made in your program.**
   (1) I have tried several MPI APIs such as Isend, Send and so on, and compared their performance, but still can't accelerate further.
   (2) Avoid unnecessary duplication in the "merge" function and get a little acceleration.

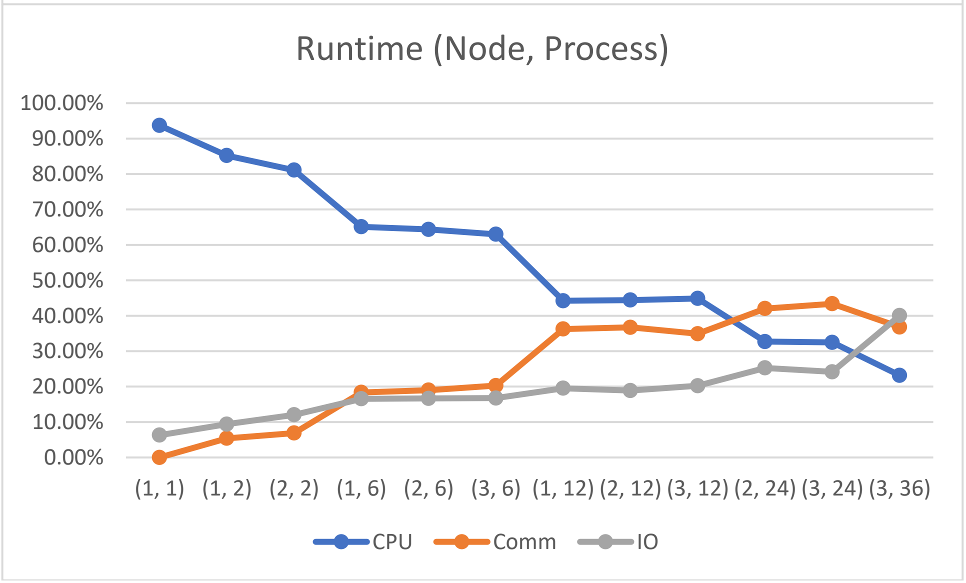
Experiment & Analysis

1. System spec

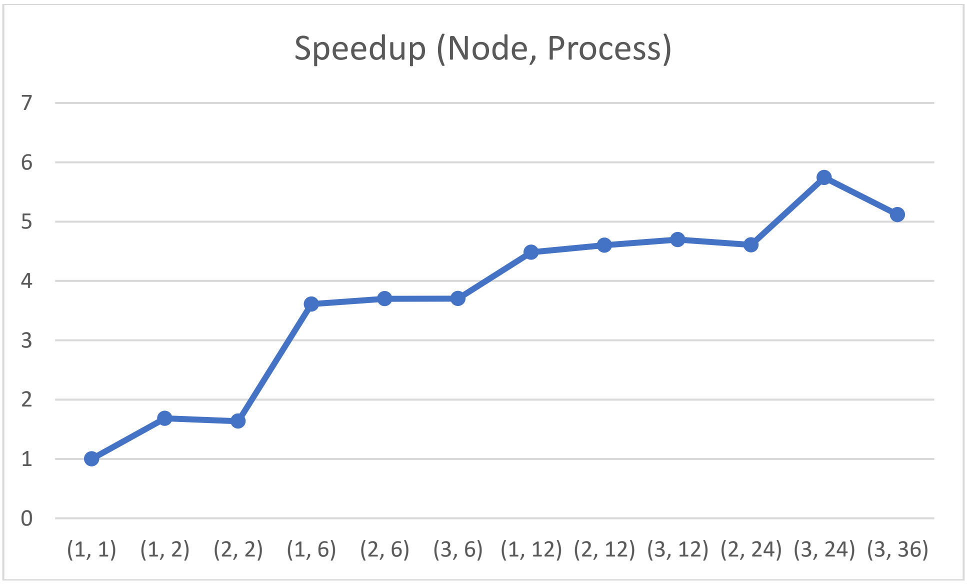
(1) 19 nodes for this course (apollo31 - 48, 50)
(2) Intel X5670 2x6 cores @ 2.93GHz (Hyper threading disabled)
(3) 96GB RAM (each node)
(4) 5.5TB shared RAID5 disk
(5) QDR Infiniband (40 Gb/s)

2. Performance metrics (experiment with 40.in)



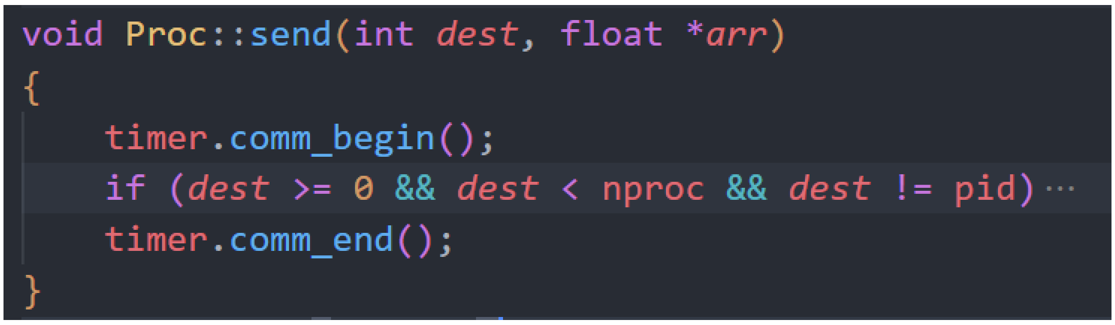


Runtime composition

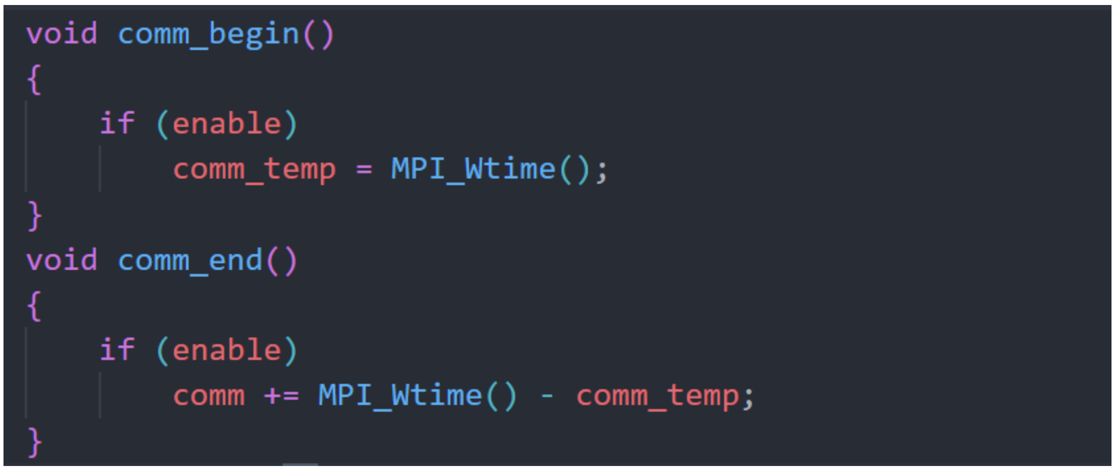


Speed up

3. Measurement method



```
void Proc::send(int dest, float *arr)
{
    timer.comm_begin();
    if (dest >= 0 && dest < nproc && dest != pid)···
    timer.comm_end();
}
```



```
void comm_begin()
{
    if (enable)
        comm_temp = MPI_Wtime();
}
void comm_end()
{
    if (enable)
        comm += MPI_Wtime() - comm_temp;
}
```

Call MPI_Wtime at the beginning and end of the function

4. Discussion

As more resources are deployed, CPU time as a percentage of execution time will decrease significantly, but communication time will increase as the number of resources deployed increases. Look at the graph Runtime Composition, comparing the results of the experiment with 12 cores and 24 cores, the factor that dominates the execution time is changed from CPU time to communication time. And, what confuses me a little bit is that the IO time seems to be unstable and irregular.

5. Bottleneck

In my opinion, my program is scalable because the communication time has not increased drastically. However, to be honest, I don't have any more ideas to improve it. I think we can discuss this during the demo if possible.

6. **Bottleneck (updated on 2022/10/24, experiment with 40.in)**

   The bottleneck in the submitted version is sorting the data for each process, a step that takes about 71.4% of CPU time. For this problem, I see it as a possible solution to divide and conquer until the cost of communication exceeds the cost of computation.

Experiences / Conclusion

In this assignment, I learned how to communicate between the process and implemented a simple sorting algorithm. This is really cool, thank you all.