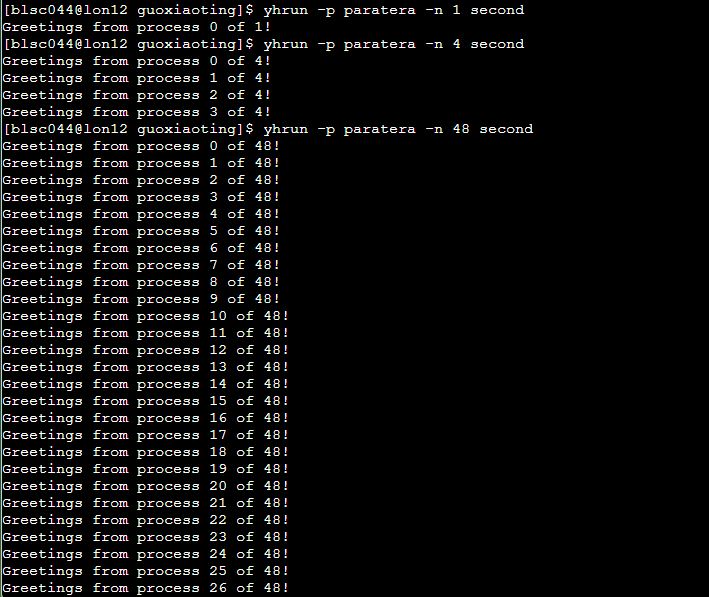
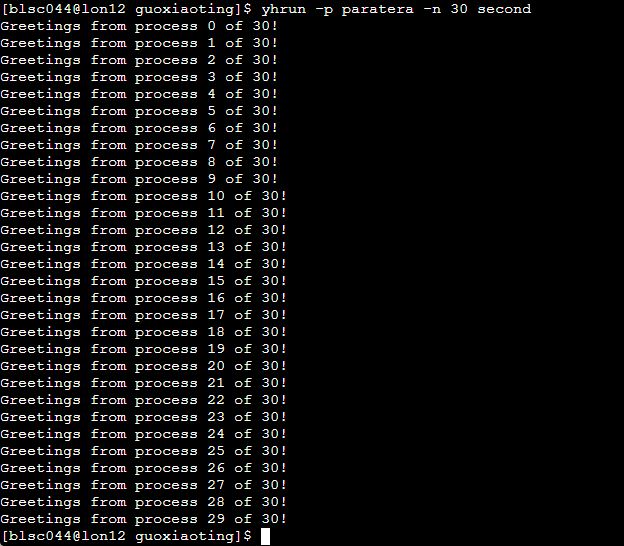
第二次作业

3.6.1 the running result of basic “greeting” program on one and different numbers of processors:

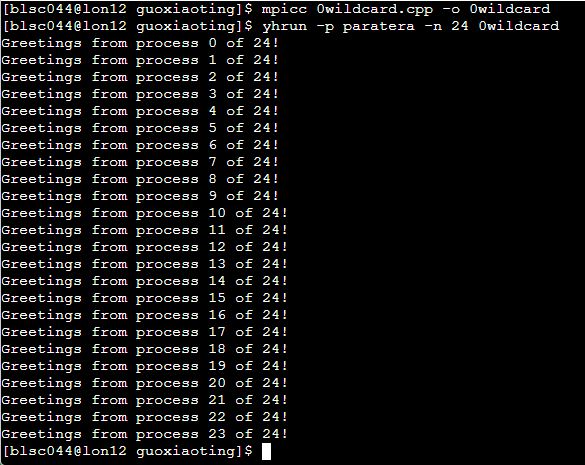
Theoretically, we can use any number of processors as long as we don’t specify a number of nodes.

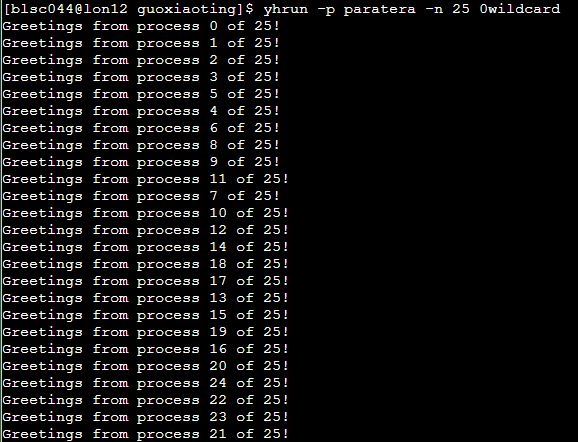


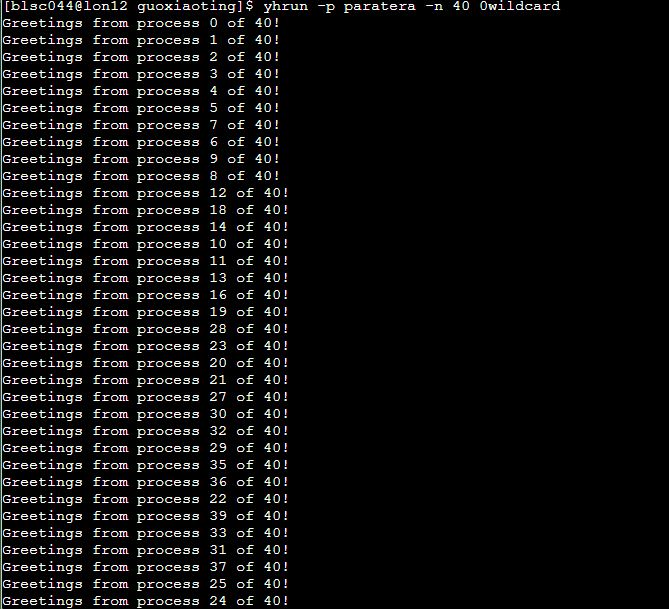




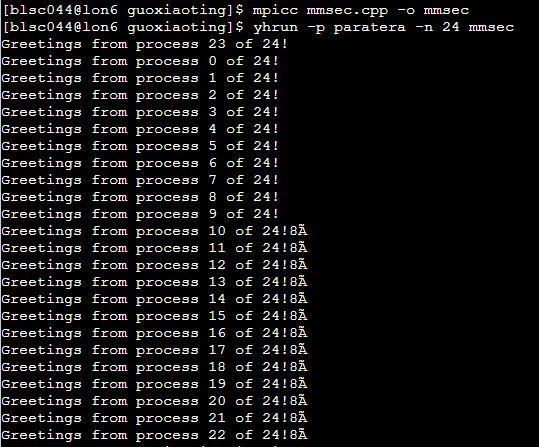
3.6.2 there is no difference in the output of the program for processes less than 24, when the number of processes is more than 24, it will print in a random order.





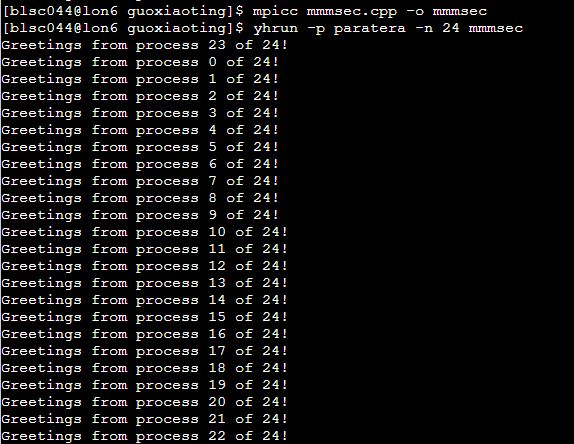


3.6.3 replace strlen(greeting)+1 with strlen(greeting): source file is named by mmsec.cpp

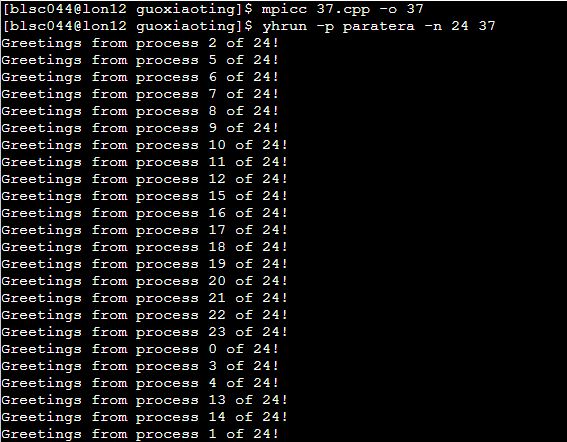


3.6.4 process p-1 prints the message it receives from all the processes other than itself.

It can only print its own greetings directly, rather than using mpi\_send and mpi\_recv. Source file is named by mmmsec.cpp



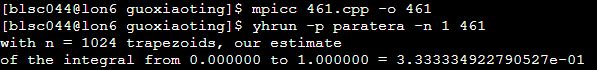
3.7 according to the result of running 37.cpp, we can find that there are no restrictions on the order in sending the message or receive the message for each process. But if we reverse order of MPI\_Send with MPI\_Recv in the program, we’ll find the process is hanged.

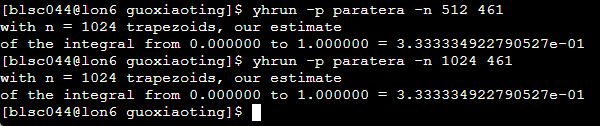


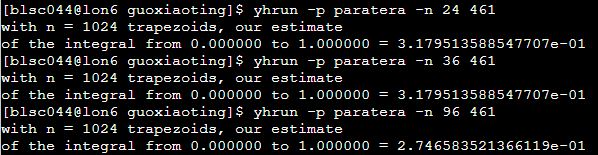
There will no execute result when running on only one process.



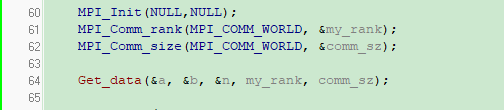
4.6.1There is no significant difference between one process and multiple processes. But by running 461.cpp, we can find if comm\_size cannot be divisible by n (such as –n 24, 36, 96), the final result is imprecise.

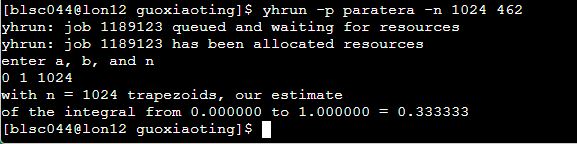




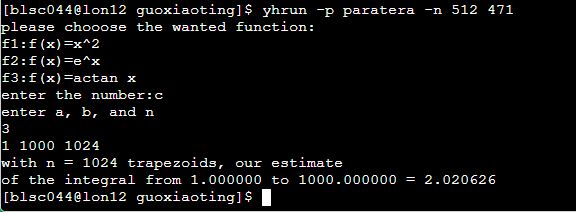


4.6.2 we should call function Get\_data after initializing my\_rank and comm\_sz;

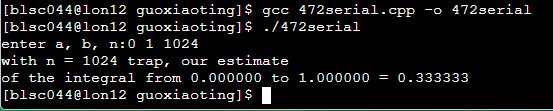




4.7.1 the source file is named by 471.cpp



4.7.2 the source file of the serial program is named by 472serial.cpp



The source file of the parallel program is named by 472parallel.cpp:

