Tianyang Li

24833444

Distributed Event Modelling

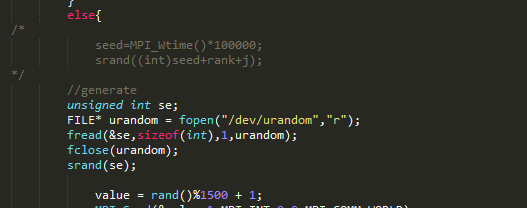
FIT5174 Assignment 2 Part B

This code is written in C language which can compile with a CNU C compiler under Linux OS. It is a MPI program which calculates the number of strikes under specified naval fleet model. In this model, there are a naval fleet patrols 1500 location, each vessel in fleet have a random location at a time. For a strike been generated, it needs four or more than four vessels been the same place at a time.

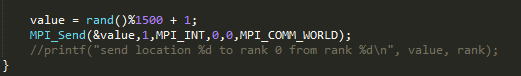
# Program structure

To simulate the fleet model and calculate the number of strikes, in this program each process other than rank 0 process represents one vessel and generate a random integer in the range of 1 to 1500, this integer represent the location for the vessel. For rank 0 process, it collecting all integers generate by other processes and store it in a inter array, and work out how many strike in a time. After all, repeat this for 60 seconds and get how many repeat times and the overall number of strikes in 60 seconds.

In particular, for processes other than rank 0 process, each of them generate a random integer with the “urandom” file under “dev” directory in Linux OS:



The random number assign to integer “value”, and send to process rank 0 with MPI\_Send function:

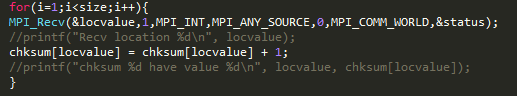


And for process rank 0, first wipe the memory of “chksum” int array to all 0 by using “memset”:

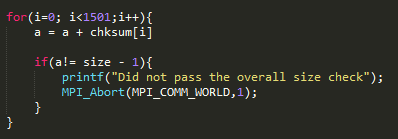




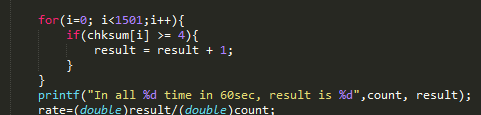
Then receive integer value “value” from other processes with MPI\_Recv function, assign it to integer “locvalue” every time. For each value received from other processes, make the chksum array increase 1 in the place of received value:



After rank 0 process received all values from other processes, if add up all integers in chksum array, it should have the same value of size-1. So I construct a size check to ensure process rank 0 received all values from other processes and the chksum array stored all values:

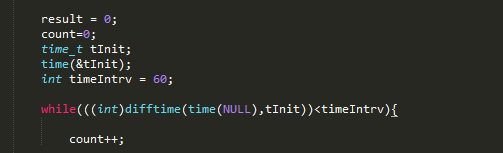


If the program pass the check, it means the chksum array worked properly. So count the number of chksum array have the value equal or large than 4 and assign to int “result”.



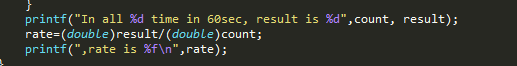
The integer “result” is the number of strikes in one time.

Repeat this “one time” progress with while loop.



While the time is not later than initial time + 60 seconds, the “one time” progress keep repeating. And the integer “count” is counting how many times it repeated.

Calculate the strike rate with counted overall time and the number of strikes.



# Inter-process communication scheme

The data flow of this program is not very complicate, it is using centralized structure.

Simply in “one instant of time”, the processes other than rank 0 generate an integer and send the integer data to rank 0. And process rank 0 receive from others and produce the number of strike. The program repeat the “one instant of time” progress for 60 seconds.

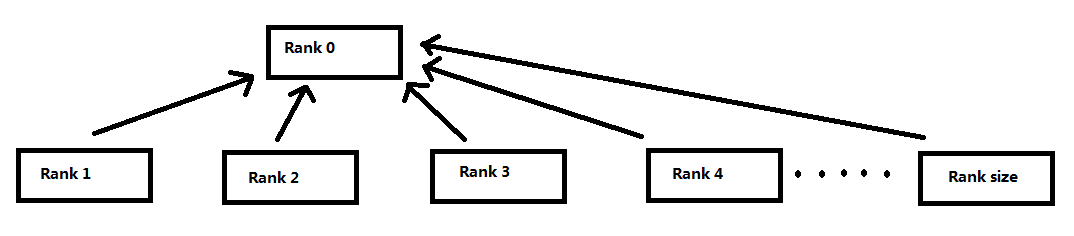
Process rank 0 receive value from other processes for size-1 times with MPI\_Recv function, because rank 0 did not send to itself a value.



Other processes send value to rank 0 with MPI\_Send function.



The overall data flow for “one instant time” is:

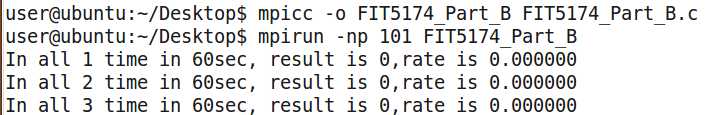


# Performance metrics

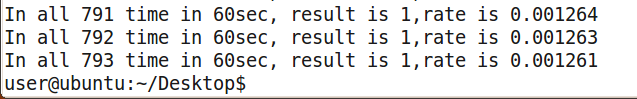
The average number of launches generated by the program over a minute is changing with the size of the program because increases in fleet size will increase the probability of strike and when there are more processes the program become slower.

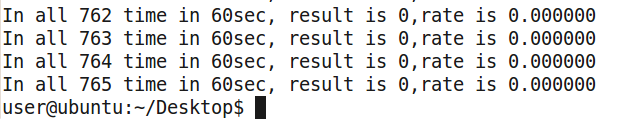
## Fleet size is 100

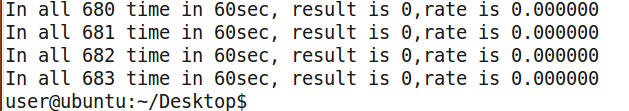
If set the size to 101 which is means the fleets have 100 vessels:



The result shows the “one instant of time” have repeat for 793, 765 and 683 times in 60 seconds, and the number of strikes is 1, 0 and 0.



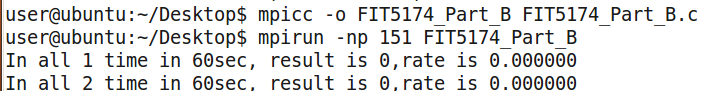




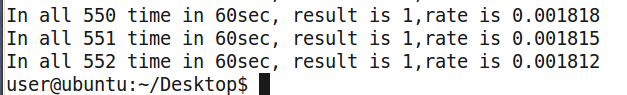
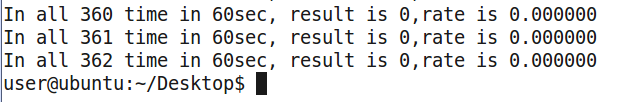
So the average number of strike launches is 0.333 when the fleet size is 100. The average times in 60 seconds is 747, and from this we can calculate the time for program run “one instant of time”, is 60/747 = 0.08032 s.

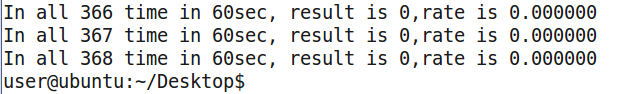
## Fleet size is 150

If set the size to 151 which is means the fleets have 150 vessels:



The result shows the “one instant of time” have repeat for 552, 362 and 368 times in 60 seconds, and the number of strikes is 1, 0 and 0.



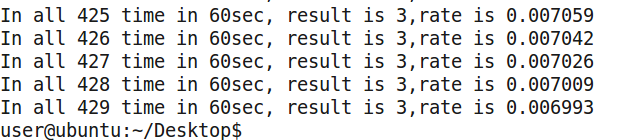
So the average number of strike launches is 0.333 when the fleet size is 150. The average times in 60 seconds is 427.33, and from this we can calculate the time for program run “one instant of time”, is 60/427.33 = 0.14041 s.

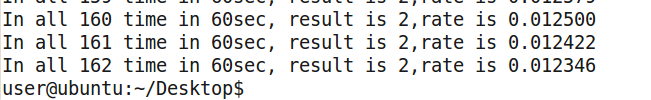
## Fleet size is 170

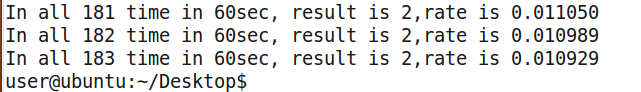
If set the size to 171 which is means the fleets have 170 vessels:



The result shows the “one instant of time” have repeat for 429, 162 and 183 times in 60 seconds, and the number of strikes is 3, 2 and 2.



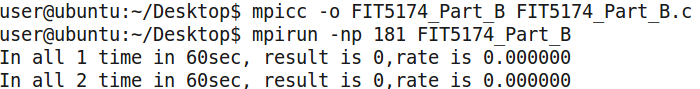




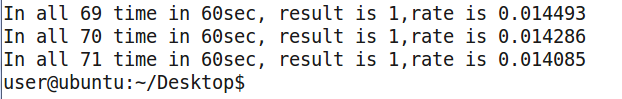
So the average number of strike launches is 2.333 when the fleet size is 170. The average times in 60 seconds is 258, and from this we can calculate the time for program run “one instant of time”, is 60/258 = 0.23256 s.

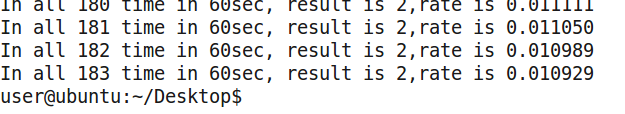
## Fleet size is 180

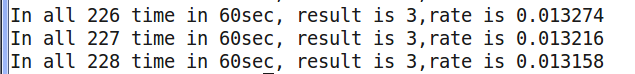
If set the size to 181 which is means the fleets have 180 vessels:



The result shows the “one instant of time” have repeat for 71, 183 and 228 times in 60 seconds, and the number of strikes is 1, 2 and 3.







So the average number of strike launches is 2 when the fleet size is 180. The average times in 60 seconds is 160.667, and from this we can calculate the time for program run “one instant of time”, is 60/160.667 = 0.37344 s.

# Summary

From the results above, we can find the average number of strikes for 60 seconds have the highest value when the fleet size is 170, which is 2.333. And when fleet size smaller or larger than 170, the average number of strikes is smaller than 2.333: when the fleet size is 150, the average number of strikes is 0.333 and when the fleet size is 180, the average number of strikes is 2. So we can say when fleet size is 170 the program have the largest strike rate.