Project Option 2: Distributed Event Modelling with MPI

Gradient Communication Scheme

Objectives

The more vessels used in the program, the higher possible that it may fulfill the launch rules. However, the inter-processor communication and expense and statistic time will increase as well. Since the probability of each location for one vessel is fixed (1/1100), the round (each loop) number will decide significantly the final launches given a number of vessels. To achieve the highest loop times, the program has to minimize the time of each loop. I designed a communication schema to distribute the message handing over all the tasks in the communicator to achieve shortest loop time.

Program Structure

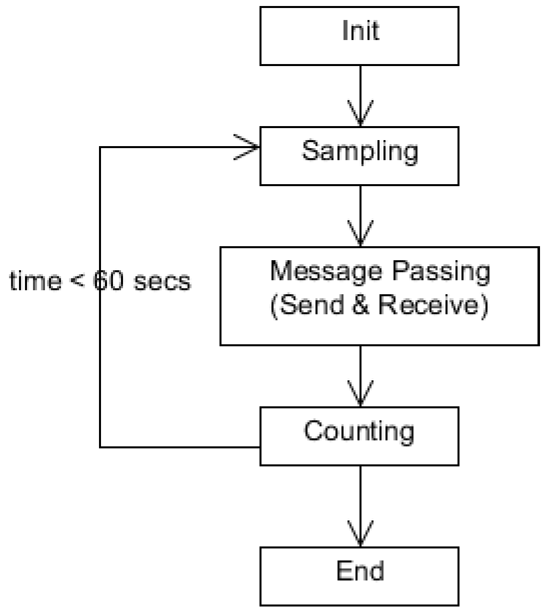

Figure 1: Program Structure

The program structure can be seen in Figure 1. In the 'Init' steps, the MPI environment is initialized. To ensure the random allocation of location value, the program will initialize distinctive random seed on each task using the combination of time and rank:

srand(time(NULL) + rank);

The 'time' ensure the program will get different value each time running the program. Then 'rank' ensure the values are different on each processor or vessel.

Then the program will record the time point and start a loop to execute the following major steps:

(1) Sampling

Each task, presenting one vessel will be allocated a location value from 0 to 1099 inclusively (1100 locations).

loc = rand() % N;

(2) Message Passing

The location values on all tasks must be finally collected together to proceed launch counting. This is the primary part which is significant to the performance of the whole program and thus the strike rate. Routing all messages directly to one task will lead to severe traffic overhead. My provision will be explained in the following section.

(3) Counting or Statistic

The occurrences of all possible location distribution (0-1099) will be calculated in this step. Each occurrence which is greater or equal to 3 will be counting as one launch. This part involves multiple loops over arrays so it has effect on the overall performance. Putting all the counting in one task cannot utilize the advantage of parallel computing. My provision helps distribute this process.

The loop will run until the execution time exceeds 60 seconds. Then the total launches can be pertained as required strike rate.

Provisions for Efficiency

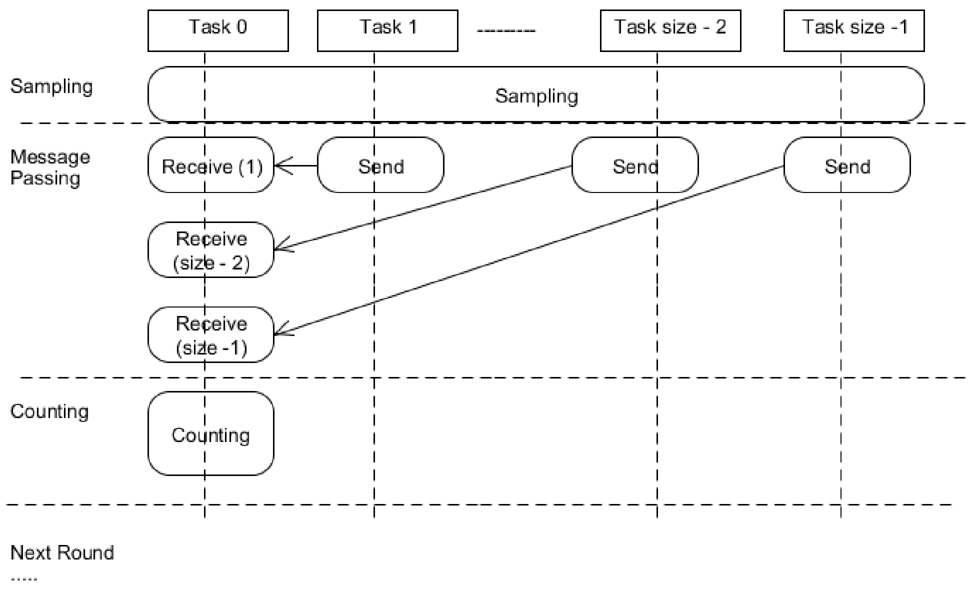

Figure 2: The Intuitive Strategy (Poor Efficiency)

The intuitive, or most primitive, strategy can be seen from Figure 2. After sampling, all vessels except the first one send location to task 0. On task 0, it executes a loop (of size -1) to receive messages one by one and record them into occurrence array. After collecting all messages, it runs counting process to collect success launches. There are two obvious problems here:

(1) Message traffic overhead on task 0. When it receives messages iteratively, other tasks have to idle.

(2) Counting is executed on task 0. Each time, task 0 has to at least execute two loops (of 1100) to clean previous data and calculate new launches. Similar to (1), more idle time on other tasks. However, this consequence is not as severe as (1).

To solve these two problems, I designed a gradient communication strategy in my work. The structure can be seen from Figure 3. Each task will still do sampling separately. After that, each task, except task 0, will calculate the interim occurrence and launches locally then send this interim data to the previous task (rank-1). Then every task, except last one, only need to receive one message from next task (rank+1), computing the interim occurrence and launches, then send it to the previous one. Through this strategy, the program is able to:

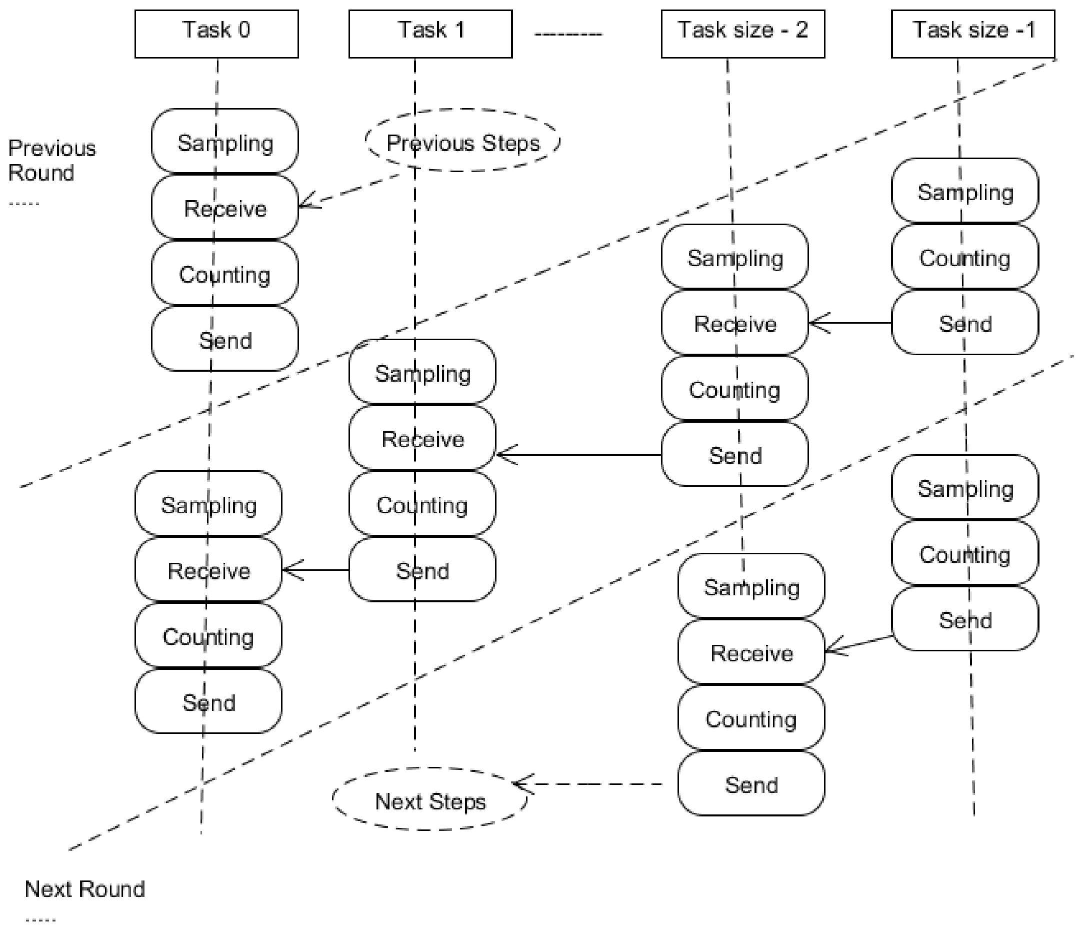

Figure 3: Gradient Structure (High Efficiency)

(1) Distribute the repetitive receive work over all tasks (except last one). Each task (except last one) only need to receive once and send once. After that, it can start work on next round as soon as possible. As a result, the processing is far more compact and there are very little idle time.

(2) Divide the counting work into all tasks. Besides displaying progress message, task 0 do no more than other tasks. Meanwhile, since each task only need to handle one location value, there is no need to iterate 1100 times to clean previous data and counting launches can be achieve in one step.

Through these provisions, the performance is highly increased. Both the number of rounds (loops) and launches are improved by many times (See details in Performance Metric).

Inter-Process Communication Schema

The inter-process communication schema is like this:

(1) Execute sampling using random function on each task

(2) On last task, update occurrence array with location value (reset old values and record new one). Then send occurrence array to the previous step (rank – 1)

(3) On each task, except the last one, begin receive occurrence array. After receiving, check whether the value at 'loc' slot equals to 2. If yes, a new launch is detected and increase the interim launches stored at the last slot in occurrence array. Then update its own location value to the slot in array. At last, send the occurrence array to previous step (rank – 1)

(4) Each task except the last one repeat (3) recursively until task 0. At the end, check the launches on task 0 and update it to the final launches variable.

(5) Repeat (1) – (4) until the time exceeding 60 seconds. Then display the total launches to the output.

Performance Metric

The first metric we used is the average launches over a minutes. Since the probability of each location value is fixed (1/1100), it means the more rounds (loops) completed the higher launches it can achieve given a fixed vessel numbers. Meanwhile, this round number can reflect the increasing of communication expense. So I use round numbers as another metric.

To evaluate the performance of my provisions against the intuitive strategy, I implemented these two programs separately. As discussed before, as the fleet size increase, the occurrences of more than 3 identical location values will increase as well. However, the communication cost increases along with the fleet size. As a result, I can expect a turning point in the figure of launches. So I choose to use increasing vessel numbers from 10, 40, 70, 80, 90, 100, and 120 to collect the respective launches. I witnessed an obvious turning point around 80 and 90 vessels. Then I work into it, further collect data using 85 vessels and it shows to be a relatively highest figure.

I run the two programs on each selected vessel number seven times, collect the round numbers and total launches, excluding the highest and lowest one to reduce abnormal data, then calculate the average round numbers and launches of the left five data. The comparison is shown in Figure 4 and 5.

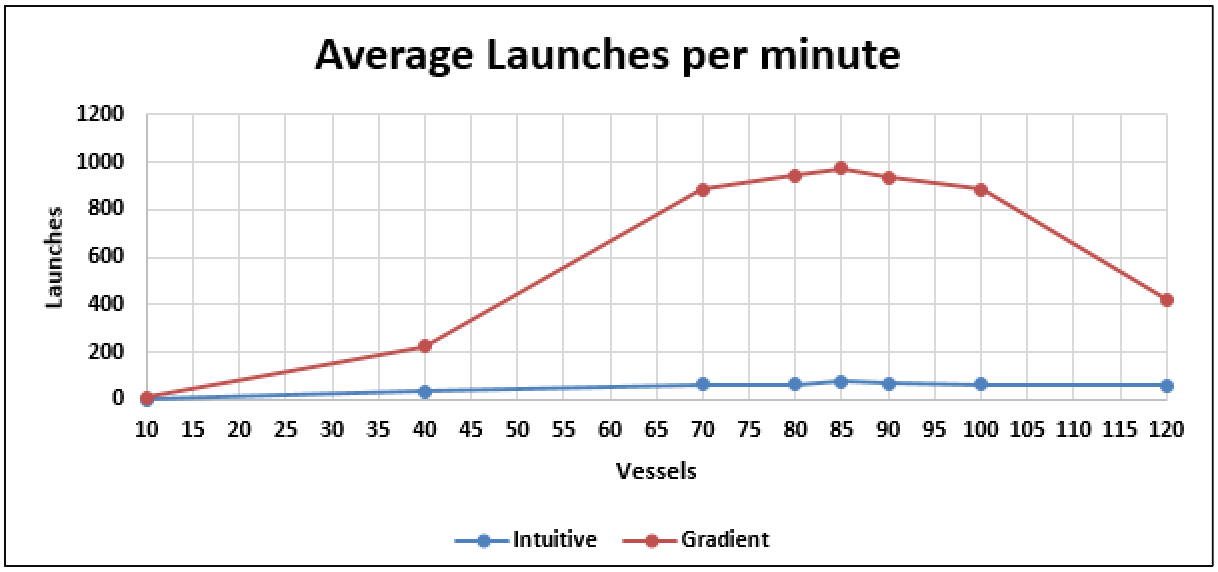

Figure 4: Average Launches per minute (Intuitive vs. Gradient)

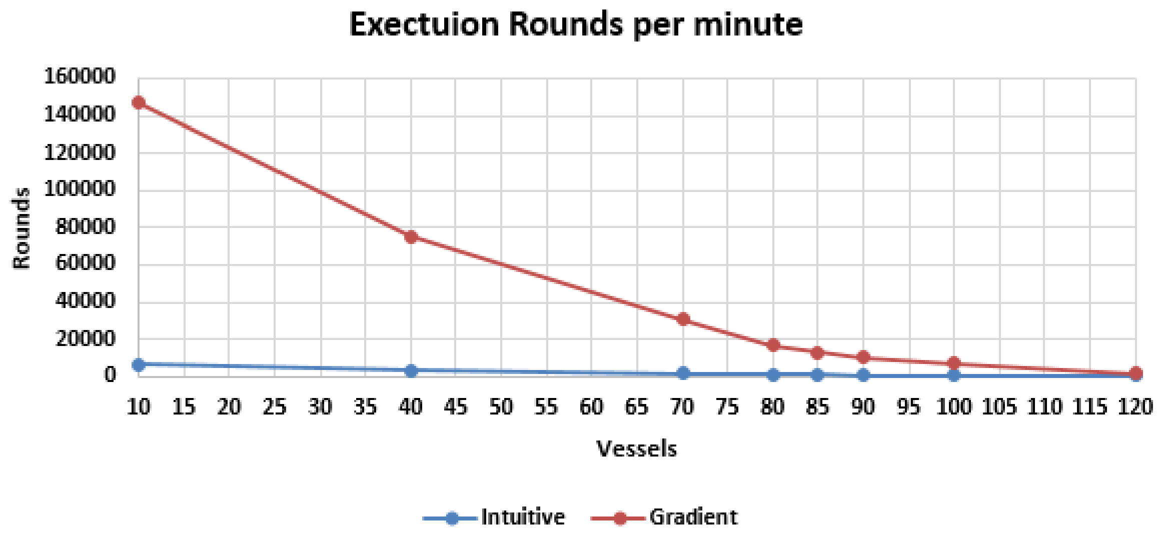

Figure 5: Execution Rounds (Intuitieve vs. Gradient)

It can be seen from the figures that the proposed Gradient communication schema dramatically reduce the communication expense and leads to an extremely large numbers of rounds compared with the intuitive schema. As a result, the highest average launches is almost 10 times more than the intuitive one.