Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Отчёт по учебному курсу «Распределенные системы»

Ершов Никита,
4 курс, группа 424

Москва
2021

Содержание

Постановка задачи

Требуется сделать следующее:

- В транспьютерной матрице размером 5*5, в каждом узле которой находится один процесс, необходимо выполнить операцию сбора данных (длиной 4 байта) от всех процессов для одного (MPI_GATHER) - процесса с координатами (0,0).

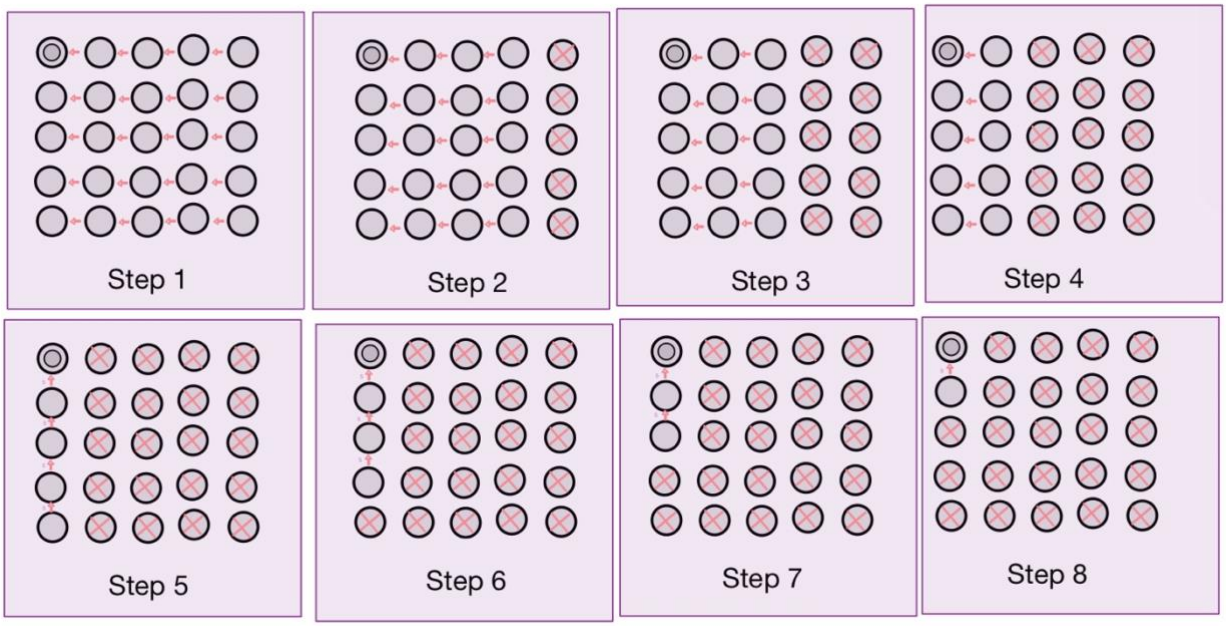
- Доработать MPI-программу, реализованную в рамках курса “Суперкомпьютеры и параллельная обработка данных”. Добавить контрольные точки для продолжения работы программы в случае сбоя. Реализовать один из 3-х сценариев работы после сбоя: а) продолжить работу программы только на “исправных” процессах; б) вместо процессов, вышедших из строя, создать новые MPI-процессы, которые необходимо использовать для продолжения расчетов; в) при запуске программы на счет сразу запустить некоторое дополнительное количество MPI-процессов, которые использовать в случае сбоя.

Получить временную оценку работы алгоритма. Оценить сколько времени потребуется для выполнения операции MPI_GATHER, если все процессы выдали ее одновременно. Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

Реализация MPI_Gather и оценка её сложности

В транспьютерной матрице размером 5*5, в каждом узле которой находится один процесс, необходимо выполнить операцию сбора данных (длиной 4 байта) от всех процессов для одного (MPI_GATHER) - процесса с координатами (0,0).

Минимальное время оценивается через минимальное расстояние между двумя самыми дальними процессами в матрице. В нашем случае, чтобы пройти от процесса с координатами (0, 0) к процессу с координатами (4, 4), необходимо сделать 8 шагов. Это количество шагов является минимальным, так как есть алгоритм, реализующий операцию сбора данных за 8 шагов.



Данный алгоритм был реализован с помощью функций MPI_Send и MPI_Recv. Получение топологии в виде транспьютерной матрицы произведено с помощью функции MPI_Cart_rank.

Оценим время работы алгоритма. Если время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1), то время выполнения операции рассчитывается следующим образом:

$$time = num\_steps \cdot (T\,s + n \cdot T\,b)\ ,$$

где n - размер передаваемого сообщения в байтах).

Таким образом, при n = 4, получаем:

$$time = 8 \cdot (100 + 4 \cdot 1) = 832$$

Добавление в программу возможности её продолжения в случае сбоя.

Для того, чтобы при сбое одного из процессов программа не завершалась с ошибкой, а продолжала своё выполнение, необходимо написать обработчик ошибок, который будет срабатывать в таких ситуациях. Для этого в стандарте MPI существует специальные функции MPI_Comm_create_errhandler и MPI_Comm_set_errhandler. Однако стандарт не позволяет определить, в каком именно процессе произошла ошибка. Это можно сделать, используя расширение MPI – ULFM [1].

Реализован сценарий а) продолжить работу программы только на “исправных” процессах;

Обработчик ошибок verbose_errhandler вычисляет процессы, вышедшие из строя.

Корневой процесс пересчитывает участки данных, которые были предназначены для вышедших процессов.

Список литературы

1. User Level Failure Mitigation. http://fault-tolerance.org/.
2. Github repository with code: https://github.com/enkeess/distributed_networks_mpi.git