МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по учебному курсу « Распределенные системы»

Отчет

студента 428 группы

факультета ВМК МГУ

Фомина Сергея Александровича

2020 год

Оглавление

Постановка задачи

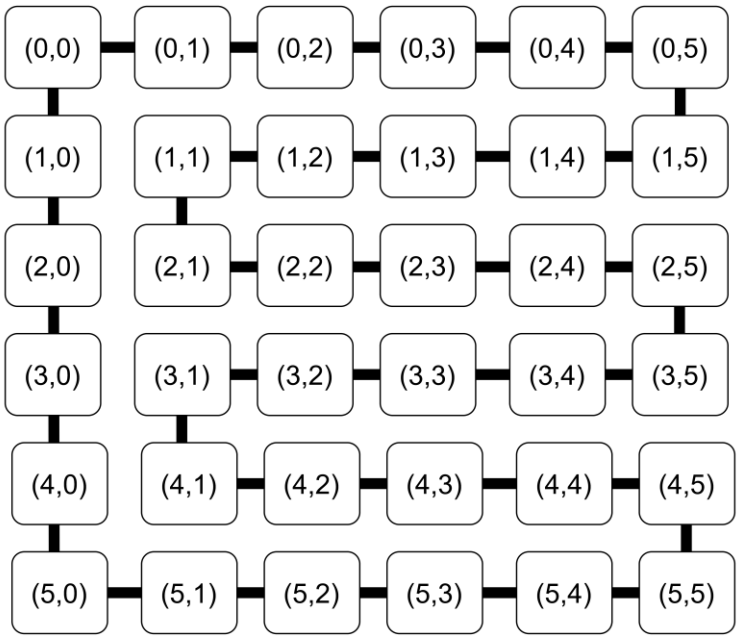
Требуется сделать следующее:

- Реализовать программу для выбора координатора среди 36 процессов, находящихся в узлах транспьютерной матрицы размером 6*6, использующую круговой алгоритм. Все необходимые межпроцессорные взаимодействия реализовать при помощи пересылок MPI типа точка-точка. Получить временную оценку работы алгоритма. Оценить сколько времени потребуют выборы координатора, если 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.

- 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.

Реализация кругового алгоритма выбора координатора

Для реализации кругового алгоритма необходимо неявно выделить в транспьютерной матрице цикл, по которому будут передаваться сообщения. В матрице с 36 процессами был выбран следующий порядок передачи сообщений:



Для имитации сбоя процессы выходят из строя с вероятностью 0.5. Перед началом алгоритма выбирается некоторый процесс инициатор, который запускает рассылку сообщений.

Каждый процесс ждет два вида сообщений: ВЫБОРЫ и КООРДИНАТОР.

Если процессу приходит сообщение ВЫБОРЫ, он проверяет, есть ли в полученном массиве его номер, и отправляет сообщение КООРДИНАТОР дальше (с выбором нового координатора), если это так. Иначе добавляет свой номер в массив и отправляет сообщение ВЫБОРЫ дальше. В случае, если процесс больше секунды не получает подтверждения от следующего процесса на сообщение ВЫБОРЫ, принимается решение о выходе из строя этого процесса и он пропускается (сообщение повторяется следующему за ним процессу до тех пор, пока не найдется работающий процесс). Если было получено сообщение КООРДИНАТОР, то процесс узнает нового координатора в матрице и отправляет сообщение следующему работающему процессу.

В итоге за два прохода цикла передачи сообщений удалось выбрать нового координатора.

Оценка времени

Оценим время работы алгоритма в предположении, что время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1) и процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

В худшем случае, если ни один процесс не выйдет из строя будет отправлено 36 сообщений ВЫБОРЫ и 36 сообщений КООРДИНАТОР. Каждое сообщение ВЫБОРЫ содержит массив длиной 36 целых чисел (36*4 байта). Каждое сообщение КООРДИНАТОР содержит одно целое (4 байта). По следующей формуле подсчитаем время работы алгоритма:

$$Time = num * (Ts + n * Tb)$$

Где num – количество сообщений, n – количество байтов.

В нашем случае получим следующую оценку:

$$Time = 36 * (100 + 36 * 4 * 1) + 36 * (100 + 4 * 1) = 8784 + 3744 = 12528$$

Доработка MPI программы для продолжения работы программы в случае сбоя

Чтобы при сбое программа не завершалась с ошибкой, был выбран сценарий с резервным процессом, который в случае выхода из строя одного из процессов заменяет его и продолжает выполнять обязанности сломанного процесса.

В программу была добавлена функция ***MPI_Comm_set_errhandler(MPI_COMM_WORLD, MPI_ERRORS_RETURN)***, чтобы в случае обращения к сломанному процессу функции передачи сообщений возвращали код ошибки, а не завершали свою работу.

В качестве резервного процесса используется последний процесс, который на протяжении всего времени работы программы ожидает два сообщения от соседей сломанного процесса, которые обнаружили выход из строя своего соседа.

Параллельная программа содержит два вложенных цикла. Внешний цикл отвечает за количество прохождений массива каждым процессом. Внутренний – за прохождение каждой строки массива. Контрольные точки сохраняются так, что можно восстановить работу программы на любой итерации внешнего и внутреннего цикла. Файлы контрольных точек сохраняются в директорию ***CP*** и имею вид ***DATA_{id}_{k}_{i}.txt***. В каждый файл сохраняется срока массива ***i*** обработанная процессом ***id***, на итерации внешнего цикла ***k***.

В начале файла программы можно найти переменные, которые настраивают, в какой момент выполнения программы произойдет имитация сбоя.

С каждым процессом взаимодействуют два соседних процесса: получение информации от предыдущего и отправление ее следующему в блокирующем режиме. Таким образом, в случае выхода из строя одного из процессов, два его соседа смогут обнаружить сбой и отправить сообщение о восстановлении последнему процессу (с нужной информацией о восстановлении). Последний процесс, получив два сообщения, читает две контрольные точки (посчитанную сломанным процессом перед сбоем и посчитанную предыдущим процессом), отправляет подтверждение соседям о вступлении в роль старого процесса и с помощью оператора ***goto*** начинает выполнение с нужного места.

Программа после восстановления продолжает работать на том же количестве процессов.

Заключение

Таким образом, был реализован круговой алгоритм выбора координатора на транспьютерной матрице при помощи пересылок MPI типа точка-точка и оценено время его работы.

MPI программа seidel-2d, была доработана так, чтобы работа программы продолжала выполнение после выхода из строя одного из процессов. Для этого один из процессов изначально считается резервным и используется только в случае выхода из строя одного из процессов.

Код двух программ прилагается к отчету и доступен по ссылке:
https://github.com/TotalChest/DistributedSystems