

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



Отчет по курсу «Распределенные системы»

428 группа,
Первушина Александра

Москва, 2021

Содержание

1. Постановка задачи	3
2. Реализация операции MPI_Reduce_scatter и оценка ее сложности	4
3. Реализация отказоустойчивости в задаче Gauss	7
Заключение	8

1. Постановка задачи

Реализовать программу, моделирующую выполнение операции `MPI_Reduce_scatter` для транспьютерной матрицы размером 4×4 при помощи пересылок MPI типа точка-точка. В каждом узле транспьютерной матрицы запущен один MPI-процесс, который для массива из 16 чисел определяет максимальное значение (среди всех процессов) и рассылает полученный результат i -ый процесс получает i -ый элемент результирующего массива.

Оценить сколько времени потребуется для выполнения операции `MPI_Reduce_scatter`, если все процессы выдали эту операцию редукции одновременно. Время старта равно 100, время передачи байта равно 1 ($T_s=100, T_b=1$). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

Доработать программу Gauss для подсчета определителя матрицы путем приведения ее к верхнетреугольному виду таким образом, чтобы она могла продолжать работу после выхода из строя одного или нескольких процессов.

2. Реализация операции `MPI_Reduce_scatter` и оценка ее сложности

В транспьютерной матрице размером 4×4 , в каждом узле которой находится один процесс, необходимо выполнить операцию нахождения максимума среди 16 чисел (в каждом процессе находится свой массив). Найденное максимальное значение для каждого из элементов с индексом i должно быть отправлено на i -ый процесс.

Для выполнения `MPI_Reduce_scatter` сначала необходимо выполнить команду `MPI_Reduce`. Данные собираются на центральных процессах с координатами (1,1), (1,2), (2,1), (2,2), с пошаговым сравнением для нахождения максимальных элементов для всех индексов при каждой пересылке.

После этого моделируется выполнение команды `MPI_scatter`. С центральных процессов за 2 шага отсылается не весь массив, а лишь необходимые остальным процессам координаты найденного на предыдущих шагах вектора максимумов.

Данный алгоритм был реализован с помощью функций `MPI_Send` и `MPI_Recv`. Создание топологии и получение координат процессов в матрице было сделано с помощью функций `MPI_Cart_create` и `MPI_Cart_coords/ MPI_Cart_rank`.

Схема пересылки данных представлена ниже:

Шаг 1

0,0 ↓	0,1 ↓	0,2 ↓	0,3 ↓
1,0	1,1	1,2	1,3
2,0 ↑	2,1 ↑	2,2 ↑	2,3 ↑
3,0	3,1	3,2	3,3

Шаг 2

0,0	0,1	0,2	0,3
1,0 →	1,1	1,2 ←	1,3
2,0 →	2,1	2,2 ←	2,3
3,0	3,1	3,2	3,3

Шаг 3

0,0	0,1	0,2	0,3
1,0	1,1 ↑ ↓	1,2 ↑ ↓	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

Шаг 4

0,0	0,1	0,2	0,3
1,0	1,1 ↔	1,2	1,3
2,0	2,1 ↔	2,2	2,3
3,0	3,1	3,2	3,3

Шаг 5

0,0	0,1 ↑	0,2 ↑	0,3
1,0 ←	1,1	1,2 →	1,3
2,0 ←	2,1 ↓	2,2 →	2,3
3,0	3,1	3,2	3,3

Шаг 6

0,0 ←	0,1 ↑	0,2 →	0,3
1,0	1,1	1,2	1,3
2,0	2,1 ↓	2,2	2,3
3,0 ←	3,1	3,2 →	3,3

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

Для первых четырех шагов объем пересылаемых данных равен $16 * 4 = 64$ байта. Для 5 и 6 шагов - 4 байта.

Таким образом итоговая сложность алгоритма:

$$T = 4 * (T_s + 64 * T_b) + 2 * (T_s + 4 * T_b) = 864$$

3. Реализация отказоустойчивости в задаче Gauss

Для реализации отказоустойчивости был реализован механизм сохранения обработанной части матрицы и перераспределения нагрузки после падения одного или нескольких процессов после между работающими процессами.

Для реализации данного функционала были реализованы следующие функции:

- `save_checkpoint`

Данная функция сохраняет состояние обработанной части матрицы после каждой успешной итерации цикла на процессе с рангом 0. Остальные процессы ждут завершения сохранения с помощью `MPI_Barrier`.

- `load_checkpoint`

Данная функция осуществляет загрузку данных о матрице после сбоя одного или нескольких процессов.

- `verbose_errhandler`

Данная функция является обработчиком ошибок. Она вызывается в случае падения процесса и осуществляет исключение неработающих процессов с помощью `MPIX_Comm_shrink` на всех рабочих процессах, после чего на них вызывается `load_checkpoint`.

Реализация чекпоинтов осуществляется с помощью добавления дополнительного флага, который принимает значение `true` после падения процесса. Все итерации цикла для вычисления определителя помещаются в дополнительный цикл `while`, который будет повторять вычисления до тех пор, пока не выполнит их без ошибки.

Заключение

Была реализована программа выполняющая операцию нахождения максимума среди 16 чисел и моделирующая выполнение операции MPI_Reduce_scatter и проведена оценка времени работы такой программы согласно заданным условиям. Также была модифицирована программа Gauss таким образом, чтобы она могла продолжать работу в случае выхода из строя одного из процессов.