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



Отчет по курсу «Распределенные системы»

Содержание

1. Постановка задачи.	3
2. Реализация операции MPI_Reduce и оценка ее сложности.	4
3. Запуск	7
Заключение	8

1. Постановка задачи.

Глобально требуется реализовать программу, моделирующую выполнение операции MPI_REDUCE на транспьютерной матрице (используя cartesian grid топологию) при помощи пересылок MPI типа точка-точка а также оценить сколько времени потребуется для её выполнения, если все процессы выдают операцию редукции 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.

2. Реализация операции MPI_Reduce и оценка ее сложности.

В транспьютерной матрице размером 5*5, в каждом узле которой находится один процесс, необходимо выполнить операцию нахождения максимума среди 25 чисел (каждый процесс имеет свое число). Найденное максимальное значение должно быть получено на процессе с координатами (0,0).

Минимальное время работы зависит от размера транспьютерной матрицы, это время можно оценить с помощью «минимального расстояния» между двумя самыми дальними процессами в матрице (здесь такими процессами являются процесс [0,0] и процесс [4,4]). В данном случае нам потребуется 5 шагов, чтобы собрать максимум на процессе [0,0].

Такое число обусловлено необходимостью участия в операции ровно двух процессов (мы сравниваем число одного процесса с числом другого процесса), таким образом получается что на каждом шаге мы можем сравнить только N div 2 чисел, где N – четное число процессов.

Тогда в нашем случае (для 5 * 5 = 25 процессов) получается:

- 1. $25 \, \text{div } 2 + 25 \, \text{mod } 2 = 13 процессов останется после первого шага$
- 2. 13 div 2 + 13 mod 2 = 7
- 3. 7 div 2 + 7 mod 2 = 4
- 4. 4 div 2 + 4 mod 2 = 2
- 5. 2 div 2 + 2 mod 2 = 1 останется единственный процесс, на котором и соберется максимальное значение (процесс [0, 0])

Один из способов такой пересылки изображен ниже, этот же способ и был реализован в программе.

Ular 1

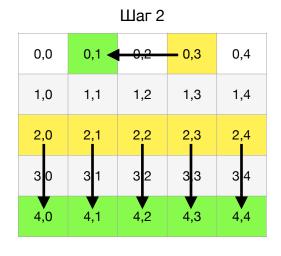
0,0 0,1 0,2 0,3 0,4

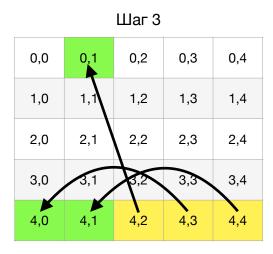
1,0 1,1 1,2 1,3 1,4

2,0 2,1 2,2 2,3 2,4

3,0 3,1 3,2 3,3 3,4

4,0 4,1 4,2 4,3 4,4





Шаг 4					
0,0	— 0,1	0,2	0,3	0,4	
1,0	1,1	1,2	1,3	1,4	
2,0	2,1	2,2	2,3	2,4	
3,0	3,1	3,2	3,3	3,4	
4,0	 4,1	4,2	4,3	4,4	

Шаг 5						
0,0	0,1	0,2	0,3	0,4		
10	1,1	1,2	1,3	1,4		
20	2,1	2,2	2,3	2,4		
30	3,1	3,2	3,3	3,4		
4,0	4,1	4,2	4,3	4,4		

Схемы работы алгоритма

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.

Для упрощения установки координат на «смежный» процесс (процесс, с которым данный собирается общаться через *MPI_Send/MPI_Recv*) и отправки/получения значений были написаны макросы.

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:

$$time = Nsteps * (Ts + n * Tb)$$

где n - размер передаваемого сообщения в байтах. В нашем случае сообщением является число, размер которого может быть равен, к примеру, 4 байтам.

Таким образом, при n = 5, получаем:

$$time = 5 * (100 + 4 * 1) = 520$$

3. Запуск

Запуск производился на обычном ноутбуке. Конфигурация системы:

Darwin Chromatica.local 21.1.0 Darwin Kernel Version 21.1.0: Wed Oct 13 17:33:23 PDT 2021; root:xnu-8019.41.5~1/RELEASE_X86_64 x86_64, open-mpi libs version 4.1.1 (brew packages for Darwin/Linux OSes), Apple clang version 13.0.0 (clang-1300.0.29.30) from Xcode 13.2 (13C5066c).

Компиляция производилась с помощью mpicc-обертки для clang.

При запуске дополнительно добавлены флаги:

- -np 25 (количество процессов = 25)
- -mca shmem mmap (замена аллокатора общей памяти на встроенный mmap)
- —use-hwthread-cpus (использование виртуальных потоков вместо реальных физических ядер при определении количества нод)
- —oversubscribe (поскольку ни физических, ни реальных ядер не хватает, чтобы выделить по ядру на процесс)

Заключение

Была реализована программа выполняющая операцию нахождения максимума среди 25 чисел и моделирующая выполнение операции MPI_REDUCE, найденное максимальное значение образуется на одном процессе – на процессе с координатами [0,0]. Также была составлена примерная оценка времени работы такой программы согласно заданным условиям.