Отчет по курсу “Распределенные системы”

Вахрушев Вадим, 428

Москва, 2020

Содержание

Постановка задачи

Данное задание состоит из двух частей:
1. В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо выполнить операцию редукции MPI_MINLOC, определить глобальный минимум и соответствующих ему индексов. Каждый процесс предоставляет свое значение и свой номер в группе. Для всех процессов операция редукции должна возвратить значение минимума и номер первого процесса с этим значением.
Реализовать программу, моделирующую выполнение данной операции на транспьютерной матрице при помощи пересылок MPI типа точка-точка. Оценить сколько времени потребуется для выполнения операции редукции, если все процессы выдали эту 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. Доработать MPI-программу, реализованную в рамках курса "Суперкомпьютеры и параллельная обработка данных". Добавить контрольные точки для продолжения работы программы в случае сбоя. Вместо процессов, вышедших из строя, создать новые MPI-процессы, которые необходимо использовать для продолжения расчетов.

Описание первой части задания

В данном задании необходимо было реализовать операцию MPI_MINLOC на транспьютерной сетке 4 на 4 и оценить время ее выполнения.
Реализованный алгоритм работает следующим образом:

1 шаг – процессы с первой координатой 0 посылают свои значения процессам с первой координатой 1 и процессы с первой координантой 3 посылают значения процессам с первой координатой 2

↓	↓	↓	↓
↑	↑	↑	↑

2 шаг – процессы с первой координатой 1 передают значения процессам с первой координантой 2

↑	↑	↑	↑

3 шаг – процессы с первой координатой 2 передают значения процессам с первой координатой 1

↓	↓	↓	↓

4 шаг – процессы с первой координатой 2 посылают значения процессам с первой координатой 3, и процессы с первой координатой 1 посылают значения процессам с первой координатой 0.

↑	↑	↑	↑
↓	↓	↓	↓

Для горизонтального распространения проводятся по сути аналогичные действия:

→			←
→			←
→			←
→			←

	→		
	→		
	→		
	→		

		←	
		←	
		←	
		←	

	←	→	
	←	→	
	←	→	
	←	→	

Оценим временную сложность данного алгоритма. Будем считать, что размер MPI_INT = 4.
При каждом взаимодействии процессов происходит две пересылки : пересылается текущий минимум и координаты процесса, на котором вычислен текущий минимум(может быть случай, когда на двух процессах вычислены одинаковые минимумы и тогда, в соответствии с семантикой minlock нужно брать процесс с минимальными координатами).
Так как координанты – это два числа, а значение – одно, на каждом этапе пересылки необходимо
Ts + 4 * Tp + Ts + 2 * 4 * Tp времени. Так как этапов 8, то суммарно необходимо 8 * (2 * Ts + 12 * Tp) единиц времени, или, в нашем случае, 8 * (2 * 100 + 12 * 1) = 1696 единиц времени.

Описание второй части задания

В данном задании необходимо было добавить механизм отказоустойчивости и контрольных точек в программу, разработанную в рамках курса по суперкомпьютерам. В данном случае отказоустойчивость осуществляется путем добавления в программу обработчика ошибок MPI-функций. В случае, если какой-то процесс вышел из строя, коммуникатор урезается, отбрасывая мертвые процессы и перераспределяя ранги, далее порождается новый процесс и интеркоммуникатор нового и старых процессов преобразуется в интракоммуникатор, добавляя новый процесс “в конец”(присваивая ему максимальный ранг). Контрольные точки реализуются в виде файлов, куда, каждые N итераций записывается полученная информация, которая вычислена в цикле(текущая итерация и текущее вычесленное значение). Таким образом, в случае возникновения ошибки, при повторных вычислениях нет необходимости вычислять все заново-достаточно прочитать сохраненные данные из соответствующего файла и довыполнить цикл. Имя файла привязано к рангу процесса, поэтому в результате перераспределения рангов процессы, вообще говоря, могут открывать файлы, созданные не ими, но это поведение приводит к корректному результату, так как цикл, который каждый процесс выполняет, также зависит только от его ранга, поэтому процесс, прочитав данные из файла, получит корректные данные, относящиеся к циклу, который ему предстоит завершить.

Заключение

В рамках данной работы были изучены механизмы реализации MPI операции на транспьютерной матрице путем выполнения операций типа “точка-точка”, проведена оценка ее времени выполнения, а также изучены механизмы обеспечения отказоустойчивости MPI-программ путем порождения новых процессов в случае отказа одного из рабочих процессов.