МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

«Распределенные системы»

ОТЧЕТ

о выполненном задании

студента 427 учебной группы факультета ВМК МГУ

Казанцевой Варвары Денисовны

гор. Москва
2023 год

Постановка задачи

Доработать MPI-программу, реализованную в рамках курса “Суперкомпьютеры и параллельная обработка данных” (умножение матриц с помощью ленточного алгоритма). 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.

Решение

Из доступных на старте программы процессов будем оставлять некоторое количество в качестве запасных. Когда какой-нибудь из работающих процессов упадет, будем выделять один из запасных процессов на его место, а итерацию перезапускать на новом коммуникаторе, полученном из старого путем удаления упавших процессов.

Чтобы падения процессов не приводили к падению программы, зарегистрируем обработчик на возврат кодов ошибок из MPI_ функций взаимодействия. На возвращаемые значения функций можно не смотреть, т. к. на каждой итерации у нас происходит сбор всех процессов в MPI_Barrier и, если есть упавшие процессы, будем запускать на новом коммуникаторе. MPI_Comm_shrink используем для “очистки” от упавших процессов.

Операция MPI_Comm_shrink может перенумеровать процессы, поэтому после её вызова нужно обновлять ранки.

Для удобства вводим новые типы (WIDTH – ширина полосы, SIZE - размер матрицы): lane_type (полоса WIDTH*SIZE), column_type (столбец SIZE*WIDTH), elem_type (квадрат WIDTH * WIDTH). Читать матрицу A будем полосами, читать матрицу B будем столбцами, а писать в матрицу C = A*B квадратами.

Чтобы сохранять промежуточные вычисления, после каждой итерации для всех работающих процессов будем заполнять матрицу C в файле. В начале итерации будем читать файлы, соответствующие процессу с указанным рангом.

В отсутствии сбоев каждый рабочий процесс будет отправлять столбец следующему процессу и получать от предыдущего используя функции MPI_Isend и MPI_Recv. В случае сбоя будем необходимые строки и столбцы считывать из файлов с матрицами A и B.

Для заполнения матриц A и B создана программа create_matrix.c, для вывода результата - print_matrix.c

Производительность

Указанные улучшения надежности приводят к следующим негативным с точки зрения производительности эффектам:

1) На перезапуски итераций в случае падения процессов уходит дополнительное время

2) Запасные процессы по сути простаивают, в то время как они могли бы выполнять некоторую работу

3) В худшем случае программа будет перезапускаться на каждой итерации, пока не исчерпает свой лимит запасных процессов, после чего все равно упадет.

Компиляция и запуск

```
mpicc  create_matrix.c -o create_matrix
mpiexec -np 1 ./create_matrix

mpicc  task2.c -o task2
mpiexec -np 6 --with-ft ulfm –oversubscribe  ./task2

mpicc  print_matrix.c -o print _matrix
mpiexec -np 1 ./print_matrix
```