Филиал МГУ имени М.В. Ломоносова в городе Сарове

Направление подготовки
«Фундаментальная информатика и информационные технологии»

Чернышов Михаил Михайлович

Программирование для распределенных систем

ОТЧЕТ

Саров, 2023

Содержание

Задание 1

1.1 Формулировка

16 процессов, находящихся в узлах транспьютерной матрицы размером 4*4, одновременно выдали запрос на вход в критическую секцию.

С помощью технологии MPI-реализовать программу, использующую централизованный алгоритм на транспьютерной матрице для прохождения всеми процессами критических секций.

Критическая секция:

```
<проверка наличия файла “critical.txt”>;
if (<файл “critical.txt” существует>) {
<сообщение об ошибке>;
<завершение работы программы>;
} else {
<создание файла “critical.txt”>;
sleep (<случайное время>);
<уничтожение файла “critical.txt”>;
}
```

Для межпроцессорных взаимодействий использовать средства MPI.

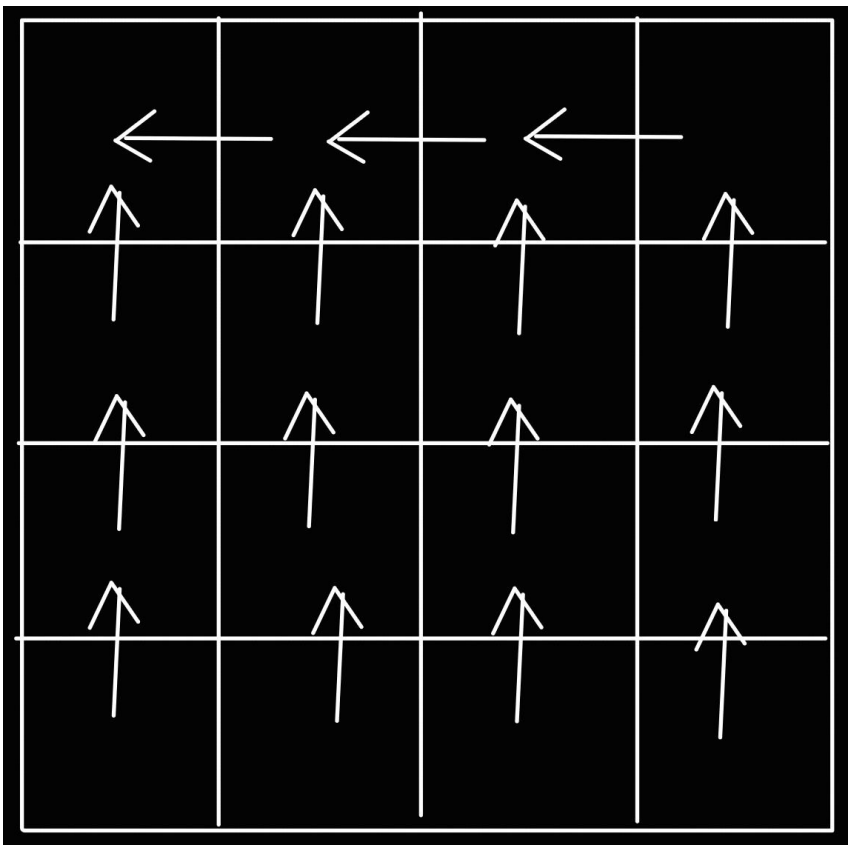
Получить временную оценку работы алгоритма. Оценить сколько времени потребуется для прохождения всеми критических секций, если координатор расположен в узле (0,0)? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

1.2 Описание алгоритма

Централизованный алгоритм предполагает, что все процессы запрашивают у координатора разрешение на вход в критическую секцию и ждут этого разрешения. Координатор обслуживает запросы в порядке поступления. Получив разрешение процесс входит в критическую секцию.

При выходе из нее он сообщает об этом координатору. Количество сообщений на одно прохождение критической секции - 3.

Таким образом, необходимо реализовать централизованный алгоритм на транспьютерной матрице, но нельзя из любого процесса запросить у (0, 0) доступ к критической секции напрямую. Нужно передавать запрос последовательно через соседний процесс на транспьютерной матрице. Также нужно поступать с разрешением на вход и с сообщением о выходе из критической секции. Один из способов пересылки представлен на картинке.



Слева сверху расположен нулевой процесс. Сообщение о разрешении на вход в критическую секцию отправляется по обратном пути.

1.3 Временная оценка

Для отправления сообщения с каждого процесса главному нужно совершить 2*1 + 3*2 + 4*3 + 3*4 + 2*5 + 6 = 48 переходов. Так как централизованный алгоритм предполагает по 3 сообщения на каждый процесс, то результат получается равным 3*48*(Ts+Tb).

Если учесть многонаправленность передачи, то после получения сообщения от ближнего процесса, не надо будет ждать сообщения от следующих, так как они будут готовы к принятию за время отправки 2-го сообщения (разрешение на вход) и принятия 3-го (критический ресурс освобожден). Дальнейшие коммуникации не будут ускорены, так как действовать можно лишь после освобождения критической секции и получения об этом сообщения. Таким образом, главный процесс начнет работу через Ts+Tb. Далее он будет отправлять сообщение и ждать обратной связи. Следовательно, временная оценка станет равна (1 + 2*48)*(Ts+Tb).

Задание 2

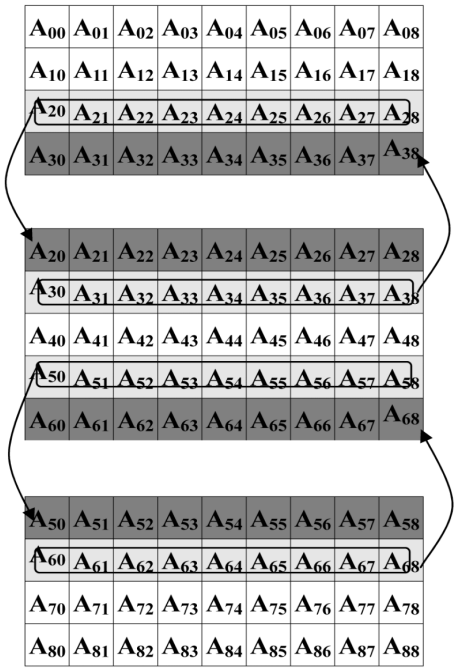
2.1 Формулировка

Для программы, которая была разработана в прошлом семестре, необходимо реализовать ее MPI-версию. Предусмотреть в программе контрольные точки, которые позволят продолжить расчет в случае сбоя/снятия программы с выполнения. С помощью технологии ULFM реализовать один из 3-х сценариев работы программы после сбоя:

1) продолжить работу программы только на “исправных” процессах;
2) вместо процессов, вышедших из строя, создать новые MPI-процессы, которые необходимо использовать для продолжения расчетов;
3) при запуске программы на счет сразу запустить некоторое дополнительное количество MPI-процессов, которые использовать в случае сбоя.

2.2 Описание MPI реализации

Трехмерный массив представляет собой k двумерных матриц i строк на j столбцов. Каждый процесс работает с определенной частью двумерных матриц. К сожалению, картинка показывает лишь двумерный случай



На картинке представлен двумерный случай. В моей программе даже первый и последний процесс имеют по 2 теневые грани, так как двумерная матрица при минимальном и максимальном k нулевая.

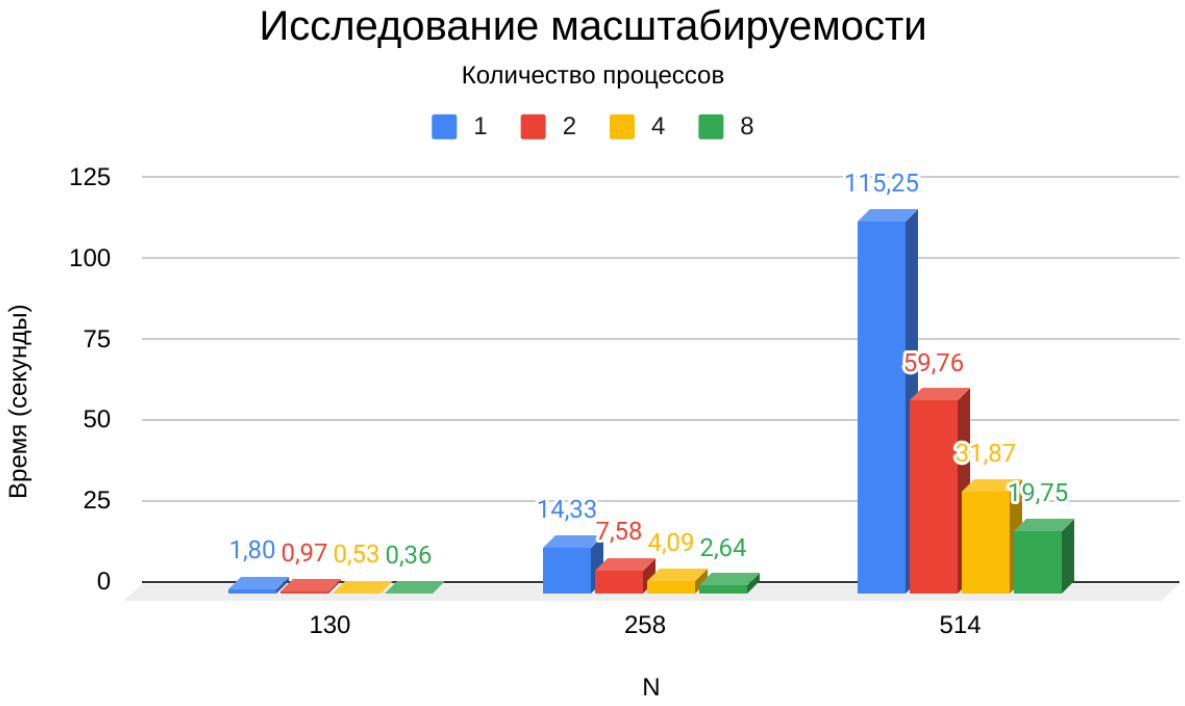
Перед relax на каждой итерации происходит обновление теневых граней. Каждый процесс имеет ksize двумерных матриц. Эта последовательность двумерных матриц разбивается на блоки по строкам. Обработка блока происходит следующим образом:

1) процесс получает верхнюю теневую грань (двумерный массив) блока от верхнего процесса;
2) процесс выполняет relax этого блока;
3) процесс отправляет нижнюю двумерную матрицу (не теневую грань) нижнему процессу.

Следует заметить, что каждый процесс хранит в памяти лишь необходимую часть трехмерного массива и две теневые грани.

Трехмерный массив реализован как двумерный, одномерные массивы которого имеют размер N*N (как двумерный), для того, чтобы отправлять и получать двумерные массивы за 1 операцию MPI.

2.3 Эффективность MPI реализации



2.4 Описание ULFM реализации

Я выбрал пункт предполагающий продолжение программы на исправных процессах.

Каждые ITER итераций происходит запись контрольной точки в файл.

При возникновении сбоя создается новый коммуникатор без поврежденных процессов с помощью MPIX_Comm_shrink. Процессы освобождают память и выделяют её заново, так как на каждый процесс теперь будет больше данных. Далее они возвращаются на итерацию, следующую за контрольной точкой, читают данные из файла и продолжают работу.

На каждой итерации процесс с максимальным rank с шансом CHANCE убивается. Таких инцидентов может произойти не больше чем NERRORS.