Метод Флойда нахождения кратчайших путей

Постановка задачи

1) Реализовать последовательный алгоритм поиска кратчайших путей методом Флойда.

2) Реализовать программу параллельного поиска кратчайших путей с помощью MPI.

3) Провести ряд тестов. Сравнить ускорение параллельного и не параллельного алгоритма.

Задача состоит в том, чтобы для имеющегося графа G найти минимальные длины путей между каждой парой вершин графа. В качестве метода, решающего задачу поиска кратчайших путей между всеми парами пунктов назначения, используется алгоритм Флойда (Floyd).

Исходной информацией для задачи поиска кратчайших путей является взвешенный граф G = (V, R), содержащий n вершин (|V|= n), в котором каждому ребру графа приписан неотрицательный вес. Граф будем полагать ориентированным, т.е., если из вершины i есть ребро в вершину j, то из этого не следует наличие ребра из j в i. В случае, если вершины все же соединены взаимообратными ребрами, то веса, приписанные им, могут не совпадать. Для поиска минимальных расстояний между всеми парами пунктов назначения Флойд предложил алгоритм, сложность которого имеет порядок n^3 . В общем виде данный алгоритм может быть представлен следующим образом:

```
// Serial Floyd algorithm

for (k = 0; k < n; k++)

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++)

A[i,j] = min(A[i,j],A[i,k]+A[k,j]);

}
```

(реализация операции выбора минимального значения min должна учитывать способ указания в матрице смежности несуществующих дуг графа). Как можно заметить, в ходе выполнения алгоритма матрица смежности A изменяется, после завершения вычислений в матрице A будет храниться требуемый результат - длины минимальных путей для каждой пары вершин исходного графа.

Параллельная схема

Как следует из общей схемы алгоритма Флойда, основная вычислительная нагрузка при решении задачи поиска кратчайших путей состоит в выполнении операции выбора минимальных значений. Данная операция является достаточно простой и ее распараллеливание не приведет к заметному ускорению вычислений. Более эффективный способ организации параллельных вычислений может состоять в одновременном выполнении нескольких операций обновления значений матрицы A.

Как результат, необходимые условия для организации параллельных вычислений обеспечены и, тем самым, в качестве базовой подзадачи может быть использована операция обновления элементов матрицы A (для указания подзадач будем использовать индексы обновляемых в подзадачах элементов). Выполнение вычислений в подзадачах становится возможным только тогда, когда каждая подзадача (i,j) содержит необходимые для расчетов элементы Aij, Aik, Akj- матрицы A. Для исключения дублирования данных разместим в подзадаче (i,j) единственный элемент Aij-, тогда получение всех остальных необходимых значений может быть обеспечено только при помощи передачи данных. Таким образом, каждый элемент Akj- строки k матрицы A должен быть передан всем подзадачам (k,j), 1 < j < n, а каждый элемент Aik столбца k матрицы A должен быть передан всем подзадачам (i,k), 1 < i< n.

Возможный способ укрупнения вычислений состоит в использовании ленточной схемы разбиения матрицы A - такой подход соответствует объединению в рамках одной базовой подзадачи вычислений, связанных с обновлением элементов одной или нескольких строк (горизонтальное разбиение) или столбцов (вертикальное разбиение) матрицы A. Эти два типа разбиения практически равноправны - учитывая дополнительный момент, что для алгоритмического языка C++ массивы располагаются по строкам, будем рассматривать далее только разбиение матрицы A на горизонтальные полосы.

Следует отметить, при таком способе разбиения данных на каждой итерации алгоритма Флойда потребуется передавать между подзадачами только элементы одной из строк матрицы A.