Решение задачи обращения матрицы с помошью блочного мтеода Гаусса с использованием MPI интерфейса

Локтев. 410

Август 2022

1 Постановка задачи

Пусть нам дана матрица вещественнх чисел размера $n * n$, разделенная на блоки размером $m * m$. Задача подразумевает нахождение обратной матрицы с помощью блочного алгоритма Гаусса, с помощью нескольких изолированных процессов в количестве p штук, с использованием MPI интерфейса для передачи информации от одного процесса к другому.

2 Хранение матрицы в памяти

Матрица хранится построчно и распределена равномерно между всеми процессами(кроме того случая, если количество процессов превышает количество строк, тогда учавствовать в работе будут только первые n/m процессов). Матрица расспределена между всеми процессами в шахматном порядке. Так для процесса с номером k будут выделены строки $k, k+p, ..., k+p*i$. Таким образом, обеспечивается равномерное хранение матрицы, что в будущем даст равномерное распределение нагруззки на все процессы.

3 Описание работы алгоритма

Для упрощения расчетов, будем считать что матрица поделилась нацело, и блоки с меньшей размерностью чем $m * m$ отсутствуют.

3.1 Распределение матрицы

Если матрица получается из формулы, то тогда каждый процесс в зависимости от своего ранга самостоятельно заполняет свой кусок матрицы. Если матрица получается из файла, то тогда процесс с рангом 0, считывает таковую из файла, после чего рассылает куски матрицы соответствующим им процессами.

3.2 Прямой ход Гаусса

Рассмотрим прямой ход Гаусса локально, относительно потока с номером k. Будем считать, что прямой ход метода Гаусса сейчас находится на своеь i-ом шаге.

Будем называть loc - локальные координаты для процесса, a glob - глобальные координаты для исходной матрицы.

Для начала процессам необходимо получить строку-лидер, т.е. строку с номером j такую, что $\rho(A_{j,i}^{-1}) \rightarrow min$(здесь ρ - матричная норма). Для этого, каждый процесс производит обращение блоков $A_{i_{loc},j}$ и выбирают среди них блок с наименьшей нормой обратного к ним блока, и подготавливает информацию об этом блоке к пересылке в виде структуры $int\_double = \{k, 1.0/norma\}$. Если процессу не удалось найти такой блок, то подготавливается структура $int\_double = \{-1, -1\}$. После чего, процессы пользуясь функцией MPI интерфейса MPI_Allreduce с использованием MPI_MINLOC получают номер процесса, который владеет нужной строкой. Если процессам не удалось найти ни одного обратимого блока, то в таком случае потоки получают что норма блока -1 и номер этого блока -1, что в свою очередь, указывает на то что матрица необратима и процессы завершают свою работу.

Допустим, что обратимый блок нашелся в строке с глобальным номером j. Если $i = j$, то тогда процесс содержащий эту строку, производит обращение блока $A_{i,i}$ и пользуясь MPI_Broadcast рассылает обратный блок всем остальным процессам. Если же нет, то перед этим производится обмен строк между процессом содержащим строку i и процессом содержащим строку j. После чего, пользуясь MPI_Scatterv, процесс равномерно распределяет эту строку между всеми остальными процессами. После того как все процессы получили свою часть строки, они производят умножение этой строки $A_{i,j} = A_{i,i}^{-1} * A_{i,j}$. После чего, с помощью MPI_Allgatherv уже нормализованная строка собирается всеми процессами. Аналогичная процедеру производится для присоединенной матрицы.

После этого, начинается непосредственное вычитание строки из нижележащих. Для этого процессы вычитают из всех принадлежащих им строк(с глобальным номером >i) $A_{l_{loc},j} = A_{l_{loc},j} - ARowBuffer_j * A_{i_{loc},i}$. Здесь, мы игнорируем индексы j < i, для экономии процессорного времени, поскольку изменение этих блоков не помешает в дальнейшем при обращении матрицы. Аналогичное производим для присоединенной матрицы, за тем лишь исключением что мы игнорируем блоки с индексом j > i, по той же самой причине.

После этого, потоки синхронизируются, и переходят к следующему шагу, вплоть до последней строки матрицы.

3.3 Обратный ход Гаусса

Обратный ход Гаусса идентичен прямому ходу Гаусса, за тем лишь исключением, что процессы не занимаются поиском строки-лидера и не взаимо-

действуют с исходной матрицей.

Вновь рассматриваем i-ый шаг алгоритма. Процесс обладающий строкой с номером i рассылает ее через MPI_Broadcast всем остальным процессам. После чего процессы производят вычитание строки из буффера из принадлежащих им строк(с индексом $<$ i): $B_{l_{loc},j} = B_{l_{loc},j} - ARowBuffer_j * B_{i_{loc},j}$

4 Связь между координатами

Каждый процесс, в виду особенности распределения памяти, будет иметь свою собственную локальную систему координат. Тогда для процесса с номером k связь между глобальными и локальными координатами устанавливается формулой

$i_{glob} = i_{loc} * p + i \;\; ; \;\; i_{loc} = i_{glob}/p$

5 Количество пересылок информации

Для простоты рассчетов, предположим что у нас имеется p процессов матрица поделена на целые блоки $l = m/n$, тогда рассчитаем количество пересылыаемых данных в худшем случае составит:

1. Во время прямого хода, процессам придется произвести пересылку 2 строк, для их обмена. Это дает 2*m*n*8*l байт информации.
2. Произвести передачу p*12*l байт информации для установления строки лидера.
3. m*n*8*2*l байт будет передано во время нормализации строки
4. Во время обратного хода будет произведено m*n*8*l байт информации.

5.1 Оценка сложности алгоритма

Рассчитаем количество математических операций которые необходимо произвести одному процессу с номером k, для выполнения этого алгоритма. Учтем, что обращение блока требует $8/3m^3$ операций, а умножение блоков требует $2 * m^2 - m$, тогда:

1. Во время поиска обратной матрицы придется произвести $\sum_{i=0}^{l}(l-i)/p$ операций обращения матрицы. Итого: $8/3 * m^3 \sum_{i=0}^{l}(l-i)/p$ математических операций.
2. Во время нормализации строки придется произвести $\sum_{i=0}^{l}((l-i)/p + i/p)$ операций умножения. Итого: $(2 * m^2 - m) \sum_{i=0}^{l}((l-i)/p + i/p)$ математических операций.

3. Во время вычитания строк придется произвести $\sum_{i=0}^{l}(l-i)*(l-i)/p$ операций умножения и вычитания. Итого: $(2*m^2-m+m*m)\sum_{i=0}^{l}(l-i)*(l-i)/p$ математических операций.

4. Во время обратного хода, потребуется $\sum_{i=0}^{l} l*(l-i)/p$ операций умножения и вычитания. Итого: $(2*m^2-m+m*m)\sum_{i=0}^{l} l*(l-i)/p$

Что в сумме дает нам: $8/3*m^3\sum_{i=0}^{l}(l-i)/p+(2*m^2-m)\sum_{i=0}^{l}((l-i)/p+i/p)+(2*m^2-m+m*m)\sum_{i=0}^{l}(l-i)*(l-i)/p+(2*m^2-m+m*m)\sum_{i=0}^{l} l*(l-i)/p = \frac{l(l+1)m\left(5l(3m-1)+8m^2+15m-7\right)}{6p} = -((7n)/(6p)) + (5mn)/(2p) + (4m^2n)/(3p) + (5n^2)/p-(2n^2)/(mp)+(4mn^2)/(3p)-(5n^3)/(6m^2p)+(5n^3)/(2mp)$ математических операций.