УДК 681.3

Ю.Я. ЛЕДЯНКИН

СПОСОБЫ ПАРАЛЛЕЛЬНОГО РЕШЕНИЯ СИСТЕМ $Ax = b$ В ЕДИНОМ ТЕХНОЛОГИЧЕСКОМ ПОТОКЕ РЕШЕНИЯ ЗАДАЧ МАТЕМАТИЧЕСКОЙ ФИЗИКИ

Анотація. *Пропонуються способи вирішення систем $Ax = b$ для спецпроцесора паралельного типу на базі процесорних елементів (ПЕ) зі скалярним помножувачем (СП) у складі кожного ПЕ. Факторизацію матриці розміром $n * n$ пропонується виконувати методами, які дозволяють скоротити в n-раз кількість операцій ділення на прямому ході виключення. На зворотному ході обчислення невідомих пропонується виключити ділення, виконуючи в ПЕ на СП тільки операції множення і складання.*

Ключові слова: *рішення систем $Ax = b$, факторизація матриці, множення векторів, спецпроцесор паралельного типу, процесорний елемент, скалярний помножувач.*

Аннотация. *Предлагаются способы решения систем $Ax = b$ для спецпроцессора параллельного типа на базе процессорных элементов (ПЭ) со скалярным умножителем (СУ) в составе каждого ПЭ. Факторизацию матрицы размером $n * n$ предлагается выполнять методами, которые позволяют сократить в n-раз количество операций деления на прямом ходе исключения. На обратном ходе вычисления неизвестных предлагается исключить деление, выполняя в ПЭ на СУ только операции умножения и сложения.*

Ключевые слова: *решение систем $Ax = b$, факторизация матрицы, умножение векторов, спецпроцессор параллельного типа, процессорный элемент, скалярный умножитель.*

Abstract. *The ways of solving $Ax = b$ systems for special parallel type processor on the base of processing elements (PE) with a scalar multiplier (SM) within each PE are suggested. Matrix factorization of $p * p$ size is suggested to perform by the methods allowing to reduce the number of divisions operations on the direct route of exceptions in n-times. On the return route of calculation of indeterminate it is proposed to exclude a division, performing in PE on SM only the operations of multiplication and totalling.*

Keywords: *$Ax = b$ solving systems, matrix factorization, vectors multiplication, special parallel type processors, processing element, scalar multiplier.*

1. Введение

Для моделирования и решения сложных задач в различных областях науки, техники, экономики требуются вычислительные системы (ВС) высокой производительности. Обычно спецпроцессоры (СП) являются неотъемлемой частью таких ВС. Наглядным примером достижений последних лет в области специализированных структур является создание гетерогенных систем. В основе их архитектуры лежит соединение СП, разработанных для решения графических задач (GPU), совместно с кластерными ВС на базе многоядерных процессоров классической структуры (CPU). Такое сочетание позволило фирмам nVIDIA и AMD с доработками технологий CUDA и STREAM, соответственно, обеспечить их применение для решения научно-технических и других (общих) задач. Это обеспечило пользователей сверхвысокопроизводительным инструментом. Привело к резкому сокращению времени решения задач при одновременном снижении стоимости всех затрат.

Процесс разработки архитектур ВС и принципов построения их на базе параллельных СП для решения задач математической физики (МФ) начался задолго до создания GPU. Но в силу объективных и субъективных причин создание специализированных кластерных структур (КС) на базе потоковых процессоров (ПП) по 512 и более опередил создание аналогичного типа параллельных СП для решения задач МФ.

ISSN 1028-9763. Математичні машини і системи, 2013, № 1

В настоящей статье предлагается реализация одного из наиболее ответственных этапов решения задач МФ – решение систем линейных алгебраических уравнений (СЛАУ) с помощью СП, реализующего единый технологический (вычислительный) поток (ЕТП). Развивается идея архитектуры, заложенная в пионерском препринте 89-57 [1] и других, более ранних публикациях, которая идет дальше идеологий существующих гетерогенных КС.

Замечание

Потоковый процессор в существующих кластерных спецпроцессорах фирмы nVIDIA в своей основе представляет собой арифметико-логическое устройство (АЛУ). В различных вариациях с обрабатывающими устройствами параллельной структуры СП из ПП выполняет арифметические операции с целыми и вещественными числами и представляет собой векторный RISC-процессор.

В работе [1] предлагается вести обработку сложных структур данных (ССД) – полиномов, записанных в виде регулярного матричного представления (РМП), в процессорном элементе (ПЭ) (аналоге ПП), основой которого является скалярный умножитель (СУ). Запись полиномов и обработка их в виде ССД предполагает параллелизм на уровне процедуры, выполняемой как операция. Такая запись позволяет совместить матричную и скалярную алгебры: матричная – обеспечивает одновременную обработку всего массива чисел (повышенную производительность), а скалярная – ускоренную обработку пары чисел. Одновременность следует из структуры исходных данных, подлежащих обработке. А вычислительный процесс обеспечивает структура СП, реализующая ЕТП от ввода исходных данных до получения псевдо решения, включая вывод результатов на устройство визуализации.

Цель работы

Ниже предлагаются способы решения СЛАУ с помощью СП в режиме ЕТП. В основе архитектуры СП лежит параллельная КС из процессорных элементов. В структуру каждого ПЭ заложена структура СУ. В целом СП должен обеспечить процедуру факторизации матрицы с последующим определением неизвестных СЛАУ. Основная цель – в СП переложить тяжесть вычислительного алгоритма (прямого и обратного ходов по обработке массива данных) на выполнение операций скалярного умножения, сокращая до минимума выполнение операций деления. А еще лучше – заменить операции деления на операции сдвига или на выполнение операции взятия обратной величины от числа, отличного (заведомо известно) от нуля. При этом исключить определение на ноль ведущего коэффициента в процессе факторизации матрицы.

Постановка задачи

Известна процедура факторизации матрицы

$$A = L * U \qquad (1)$$

по методу Гаусса с делением на прямом ходе исключения. Для матрицы $A = \|a_{ik}\|$ $(i = 1,2,...,m; k = 1,2....,n)$, ($A$ порядка n) она сводится к приведению A к верхней треугольной (U) и заключается в почленном вычитании из второго уравнения коэффициентов первого уравнения, умноженных на a_{21} / a_{11}, из третьего – первое, умноженное на a_{31} / a_{11}, из k-го первое, умноженное на $m_{k1} = a_{k1} / a_{11}$, k =1,2, … , n. Это приводит оставшиеся $(i-1)$ уравнений к исключению под диагональных элементов первого столбца. Далее по аналогии процедура исключения под диагональных элементов повторяется. Исклю-

чения (прямой ход) продолжают до $(n-1)$-го шага, на котором получают верхнюю (U) и нижнюю (L) треугольные матрицы. Процесс нарушается, когда $a_{kk}^{(k)}=0$. Поэтому должно выполняться условие $a_{kk}^{(k)}\neq 0$. Всего для матриц размерности $n*n$ необходимо на прямом ходе факторизиции выполнить $n^2/2$ операций деления.

После приведения к верхней треугольной матрице U процесс продолжают в обратном порядке (обратный ход), по которому значения, определенные по известному алгоритму обратного хода, подставляют в матрицу, начиная с последней строки. Для этого требуется n операций деления.

Прямой ход известного алгоритма Гаусса при исключении коэффициентов в каждом из под диагональных столбцов требует выполнения операции однократной, но для каждой строки, операции деления. А всего $n^2/2$ делений на прямом ходе. На каждом этапе требуются также и проверка главного элемента на ноль, выполнение соответствующих перестановок строк (или столбцов). Естественно, что дополнительные n операций деления на обратном ходе усложняют процесс, увеличивают время выполнения каждого шага и решения СЛАУ в целом.

Кроме того, что выполнение операции деления требует больше времени, чем операция умножения (во всех, практически, ЭВМ), погрешность округления при делении больше погрешности округления при умножении, она не вписывается в идеологию ЕТП и в целом увеличивает время получения решения.

Предлагается для системы уравнений

$$Ax=b, \tag{2}$$

где x и $b-n$-мерные векторы – столбцы, A – неособенная матрица n-го порядка, положительно определенная, самосопряженная. Матрицу можно разложить в произведение LDU. Симметричность матрицы может сократить вычислительный процесс, но она не обязательна. Возможно, что выполнено масштабирование коэффициентов матрицы (обычно эти требования предъявляют к коэффициентам и они вытекают из процедуры аппроксимации уравнений методами конечных разностей (МКР) или конечных элементов (МКЭ)). Известно, что процесс решения системы (2) можно выполнять в виде прямого и обратного ходов. На прямом ходе исключения неизвестных матрица A приводится к верхней треугольной матрице U. На обратном ходе подстановкой значений коэффициентов, вычисленных на прямом ходе, определяется вектор неизвестных x.

2. Прямой ход

Пусть $A^{(k)}$ означает начальную матрицу k-го шага, $A^{(1)}\equiv A$, $A^{(k+1)}\equiv U$ и $a_{ij}^{(k)}$ есть элемент i-й строки и j-го столбца матрицы $A^{(k)}$.

Процедура исключения первого неизвестного выглядит следующим образом:

$$\left[\begin{array}{llll|l} a_{11} & a_{12} & a_{13} & a_{14} & b_1 \\ (0=a_{21}a_{11}-a_{11}a_{21}) & (a_{22}a_{11}-a_{12}a_{21}) & (a_{23}a_{11}-a_{13}a_{21}) & (a_{24}a_{11}-a_{14}a_{21}) & (b_2a_{11}-b_1a_{21}) \\ (0=a_{31}a_{11}-a_{11}a_{31}) & (a_{32}a_{11}-a_{12}a_{31}) & (a_{33}a_{11}-a_{13}a_{31}) & (a_{34}a_{11}-a_{14}a_{31}) & (b_3a_{11}-b_1a_{31}) \\ (0=a_{41}a_{11}-a_{11}a_{41}) & (a_{42}a_{11}-a_{12}a_{41}) & (a_{43}a_{11}-a_{13}a_{41}) & (a_{44}a_{11}-a_{14}a_{41}) & (b_4a_{11}-b_1a_{41}) \end{array}\right]. \tag{3}$$

При вычислениях в (3) использованы выражения:

$$a_{ij}^{(k+1)}=a_{ij}^{(k)}\,a_{kk}^{(k)}-a_{ik}^{(k)}a_{kj}^{(k)},\quad j=(k+1),\ldots,n,$$

$$b_j^{(k+1)} = b_j^{(k)} a_{kk}^{(k)} - a_{ik}^{(k)} b_k^{(k)}, \quad i = (k+1),...,n. \tag{4}$$

После $(n-1)$-го исключения получают верхнюю треугольную матрицу (U) с диагональными элементами, отличными от единицы и не равными нулю:

$$U = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ & a_{22}^{(2)} & a_{23}^{(2)} & a_{24}^{(2)} \\ & & a_{33}^{(3)} & a_{34}^{(3)} \\ & & & a_{44}^{(4)} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2^{(2)} \\ b_3^{(3)} \\ b_4^{(4)} \end{bmatrix}. \tag{5}$$

Из (3)–(5) видно, что в вычислительном плане алгоритм (с точки зрения реализации устройством) исключения построен по аналогии с процедурами решения задач по методу приращений (МП) [2] и матричного умножения, где элементы вектора-столбца умножаются на коэффициент с последующим сложением столбцов.

Алгоритм приведения матрицы $Ax = b$ к верхней треугольной без деления на прямом ходе по (3)–(5)

Назовем ведущей строкой первую k-ю строку, элементы которой в процессе преобразования остаются без изменения; ведущим столбцом – столбец, элементы которого расположены под ведущим (диагональным) элементом ведущей строки.

Шаг 1. Осуществим вызов элементов ведущей i-й строки из памяти.

Шаг 2. Осуществим вызов элементов ведущего (и последующего за ним) редуцируемого столбцов, подлежащих операции преобразования. Запишем их в СУ для покомпонентного умножения (без накопления).

Шаг 3. Все элементы ведущего столбца (a_{ik}), $i = (k+1),...,n$ умножают на j-й (a_{ij}), $j = (k+1),...,n$ элемент ведущей строки. Полученные таким образом произведения поэлементно вычитают из элементов j-го, $j = (k+1),...,n$ столбцов, умноженных на ведущий элемент ведущей строки, т.е. из элементов второго столбца, умноженных на ведущий элемент, вычитаются элементы первого столбца, умноженные на второй элемент ведущей строки, затем (или одновременно) в работу вступает пара (третий-первый) столбцы, j-й – первый столбец и т.д. до n-го с первым.

Замечание

В спецпроцессоре, построенном на базе СУ с реализацией ЕТП [1], процесс можно дополнительно распараллелить. Для этого необходимо взять не один, а n столбцов (матрицу) умножителей, каждый j-й из которых (запоминает ведущий столбец) умножается на j-й элемент ведущей строки.

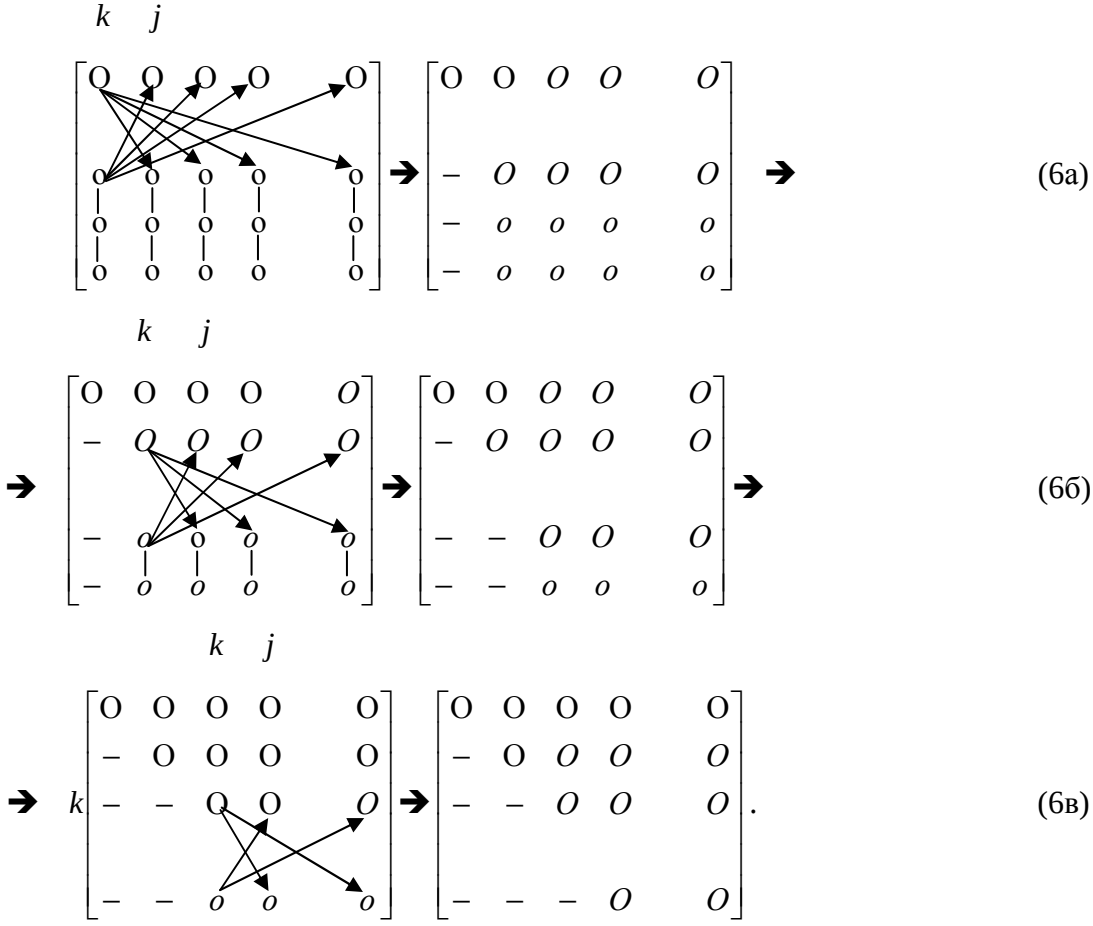
Шаг 4. Ведущий (диагональный) элемент ведущей строки оставляем без изменения. В ячейки памяти для хранения элементов (a_{ik}) ведущего столбца (под ведущим элементом) записываем нули (как результат в преобразуемой (U) матрице).

Замечание

1. Поскольку из i-х ($i = (k+1),...,n$) произведений j-го столбца ($j = (k+1),...,n$) – (уменьшаемое) надо вычесть i-е ($i = (k+1),...,n$) (вычитаемое) произведение ведущего столбца на j-е элементы ведущей строки ($j = (k+1),...,n$), то достаточно поменять знак только у одного сомножителя (например, у элементов ведущей строки), чтобы заменить вычитание столбцов на суммирование их (с учетом знаков, естественно).

2. Если применять метод умножения чисел младшими разрядами обоих сомножителей вперед [4, 5] , то операции умножения столбцов на соответствующий коэффициент парой СУ можно объединить с их сложением, выполняя процедуру вычисления текущего коэффициента матрицы как одну операцию.

Представим, для наглядности, алгоритм прямого исключения без деления в виде граф – схемы:



(6а)

(6б)

(6в)

В граф-схемах (6а–6в) стрелками определены пары сомножителей.

При этом коэффициент матрицы A, стоящий на пересечении k-й строки с k-м столбцом (и является выходным концом нисходящей стрелки графа), последовательно умножается на j-е ($j = k-1,...,n$) коэффициенты ниже лежащей $(k-1)$-й строки (куда входит нисходящая стрелка графа). Коэффициент, стоящий на пересечении $(k-1)$-й строки с k-м столбцом (выходной конец восходящей стрелки графа), последовательно умножается на $(k-1)$-е коэффициенты верхней k-й строки матрицы (куда приходит восходящая стрелка графа), и одновременно выполняется попарное вычитание получаемых произведений в соответствии с (3–5).

В пределах матрицы граф-схемы большими кружками обозначены коэффициенты, вычисленные в процессе факторизации, малыми – коэффициенты, которые подлежат пересчету, прочерками – коэффициенты, равные нулю. Из (3)–(5) видно, что в вычислительном плане алгоритм исключения (с точки зрения реализации устройством) построен по аналогии с процедурами решения задач по методу приращений (МП) [2] и матричного умножения [3], где элементы вектора-столбца (коэффициенты) умножаются на коэффициент (с возможным последующим сложением столбцов) с помощью СУ.

3. Обратный ход алгоритма

Возможен процесс, когда полученное ранее значение неизвестного x (в общем случае это значение x_i) является множителем для всех элементов n-го (j-го), $j=n,...,1$ столбца матрицы U. Оптимальной будет организация вычислений по аналогии с той, которую выполняют в процессе исключения при прямом ходе решения.

Назовем ведущим столбцом такой j ($j=n,...,2$), в котором во время обратного хода последовательно вычисляют неизвестные $\left(x_j\right)$ СЛАУ.

Назовем вычисленным ведущим столбцом такой j-й ($j=n,...,2$), который получен путем умножения ведущего столбца на вычисленное значение соответствующего ему неизвестного $\left(x_j\right)$.

Назовем вектором правых частей вектор, который получают на каждом цикле путем вычитания из вектора свободных членов вычисленного ведущего столбца. Тогда алгоритм можно продолжить следующим образом.

Алгоритм обратного хода

1. Определим из последней строки неизвестное

$$x_n = b_n / u_{nn}. \tag{7}$$

2. Умножив i-е ($i=\left(n-1\right),...,1$) элементы $\left(u_{in}\right)$ ведущего столбца на x_n:

$$\begin{bmatrix} | \\ U_{in} \\ | \end{bmatrix} * x_n = \begin{bmatrix} | \\ U_{in} - x_n \\ | \end{bmatrix}, \quad i=(n-1),...,1, \tag{8}$$

получим вычисленный ведущий столбец.

3. Из вектора правых частей (уменьшаемое) вычтем вектор вычисленного ведущего столбца (вычитаемое):

$$\begin{bmatrix} b_i \end{bmatrix} - \begin{bmatrix} | \\ U_{in} * x_n \\ | \end{bmatrix} = \begin{bmatrix} | \\ (b_i - U_{in} * x_n) \\ | \end{bmatrix}. \tag{9}$$

Получим новое значение вектора правых частей, которое и запомним.

4. Из уравнения в $\left(n-1\right)$-й строке матрицы U, где выполняется равенство

$$\begin{bmatrix} | \\ U_{n-1,n} \\ | \end{bmatrix} * x_{n-1} = \begin{bmatrix} | \\ b_{n-1} - U_{nn} * x_n \\ | \end{bmatrix}, \tag{10}$$

определим очередное $\left(n-1\right)$-е значение неизвестного:

$$x_{n-1} = (b_{n-1} - U_{nn} * x_n) / U_{n-1,n-1}. \tag{11}$$

5. Делаем $\left(n-1\right)$-й столбец ведущим. Умножим его элементы $U_{i,n-1}$ ($i=(n-2),\ldots,1$) на значение x_{n-1}. Получим значения коэффициентов ведущего $\left(n-1\right)$-го столбца матрицы U:

$$x_{n-1} * \begin{bmatrix} U_{i,n-1} \end{bmatrix} = \begin{bmatrix} | \\ U_{i,n-1} * x_{n-1} \\ | \end{bmatrix}, \quad i = (n-1),...,1 \tag{12}$$

и запоминаем их взамен значений коэффициентов предыдущего столбца.

6. Вычислим вектор правых частей. Для i-й $(i = (n-2),...,1)$ строки обратного хода он равен

$$\begin{bmatrix} b_i^{(j)} \end{bmatrix} = \begin{bmatrix} | \\ (b_i^{(j)} - \sum_{j=i+1}^{n} U_{ij} * x_j) \\ | \end{bmatrix}, \quad j = (n-1),..., 2,\ i = (n-2),..., 1. \tag{13}$$

7. Вычислим текущее i-е значение неизвестного x_i для i-й строки:

$$x_i^{(j)} = (b_i^{(j-1)} - \sum_{j=i+1}^{n} U_{ij} * x_j)/U_{ij}, \quad j = (n-1),...,2, \quad i = (n-2),...,1 \tag{14}$$

и так далее.

Недостатком вычислительной части описанного алгоритма для обратного хода является задержка на время выполнения операции деления U_{ij} при определении текущего значения неизвестного x из (14), так как в этот момент нельзя получить вычисленное значение ведущего столбца по (10).

Возможно ускорение процесса за счет его распараллеливания с упреждением.

На этапе прямого хода исключения, получив ведущую строку, сразу в параллель определим обратную величину

$$L_{ii} = 1/U_{ii} = U_{ii}^{-1} \tag{15}$$

от вычисленного значения ведущего элемента матрицы U_{kk}. Тогда в процессе обратного хода, взамен (7), (14), надо соответственно выполнить

$$x_n = b_n * L_{nn}, \tag{16а}$$

$$x_i^{(j)} = b_i^{(j)} * L_{ii}, \tag{16б}$$

где $b = (b_i^{(j-1)} - \sum_{j=i+1}^{n} U_{ij}\, x_j), \quad j = (n-1),...,2, \quad i = (n-2),...,1.$

Для решения системы $Ax = b$ надо выполнить n операций деления. Если операцию деления (взятия обратной величины от коэффициента) выполнить на прямом ходе, то процесс обратного хода упрощается, n значений величин L_{ii} (15) будут вычислены. Тогда процедура обратного хода выполняется без деления, становится однотипной с процедурой для прямого хода и вписывается в ЕТП.

Модификация обратного хода алгоритма

$$x_i = (b_i * L_{ii}), \tag{17}$$

– вычислить вектор правых частей:

$$b_i^{(j)} = (b_i^{(j-1)} - \sum_{j=i+1}^{n} U_{ij} * \cdot x_j); \tag{18}$$

– вычислить текущее значение неизвестного:

$$x_i^j = b_i^j * L_{ii} \tag{19}$$

и т.д.

Преимущество применения предлагаемого решения для задач МФ состоит в предварительном масштабировании коэффициентов $B(x_i, x_k^{(i)})$ при сеточных функциях $y(x_k^{(i)})$ [1]. Масштабирование коэффициентов заложено в соблюдение принципа максимума, который лежит в основе решения эллиптических и др. задач. Он определяет, что значения сеточных функций во внутренних узлах области решения не могут быть больше значений сеточных функций на ее границе. При этом узловой микропроцессор (или ПЭ) реализует скалярное произведение, работая с числами меньше единицы (в формате представления их с фиксированной запятой (ФЗ) перед старшим значащим разрядом). Поскольку умножение двух чисел, каждое из которых меньше единицы, дает результат меньше единицы, то переполнение разрядной сетки исключается.

Но для матриц большой размерности умножение чисел меньше единицы может привести к потере значности чисел (выход их за пределы разрядной сетки). Если в СУ СП заложен, например, метод обработки чисел с ФЗ, то решением проблемы потери значности может быть сдвиг кода мантиссы в сторону старшего разряда (во всем массиве чисел, подлежащих пересчёту) на каждом k-м цикле $(k = 2,...,m)$. Операция равносильна умножению на величину, равную основанию системы счисления (представления кода обрабатываемых чисел). Это обеспечивает компенсацию сдвига, на которую уменьшаются два числа меньше единицы при их умножении. Реализация в СУ СП представления чисел в форме с плавающей точкой (ПЗ) упрощает процедуру нормализации.

Замечание

Описанный выше способ факторизации матриц был предложен и разработан в конце 80-х годов на этапе подготовки работ по созданию СП с ЕТП и пионерского препринта [1] к печати. Но в силу форс-мажорных обстоятельств все публикации, касающиеся спецпроцессора, были приостановлены. В 2011 году, когда было принято решение опубликовать методы (численные и арифметические), касающиеся разработки СП, сюрпризом всемирной паутины, в которую мы окунулись в последние годы, оказалось обнаружение алгоритма, опубликованного в 1967 году Барейсом [8]. Эта работа, посвященная приведению матрицы к треугольному виду и решению СЛАУ, в силу объективных обстоятельств оказалась тогда неизвестной.

В работе [8] Барейс (на примере вычисления определителя матрицы путем приведения её к треугольному виду с использованием только целочисленной арифметики) также предложил исключить операцию деления на первом этапе прямого хода. По аналогии с изложенным выше он исключил операцию деления, заменив её операцией умножения на коэффициент a_{ii}. Но, поскольку многократное умножение коэффициентов строк матрицы большого размера на её элементы может привести к переполнению разрядной сетки в пределах используемого формата, Барейс предложил алгоритм, при котором коэффициенты матрицы остаются в формате представления.

С этой целью правая часть уравнения

$$a_i^{\rightarrow} \leftarrow a_i^{\rightarrow} - \frac{a_{ij}}{a_{jj}} * a_j^{\rightarrow} \tag{20}$$

приводится к виду

$$a_i^{\rightarrow} \leftarrow a_i^{\rightarrow} * a_{jj} - a_{ij} * a_j^{\rightarrow}, \tag{21}$$

где a_{ij} – элемент матрицы, $i = 0,..., N-1$ – номер строки, $j = 0,..., N-1$ – номер столбца. Для сокращения записи строки матрицы используется обозначение $a_i^{\rightarrow}$ и с ними можно производить операции, аналогичные операциям с векторами. После этого, во избежание потери значности в вычислениях, автор предложил делить (и доказал возможность деления) правую часть выражения (21) нацело на $a_{j-1,j-1}$ (а не на a_{jj}, как это выполняется в алгоритме Гаусса с делением на прямом ходе в обозначениях, принятых в настоящей статье (в выражении (4) – это $a_{kk}^{(k)}$)) с предположением, что $a_{-1,-1} \equiv 1$:

$$a_i^{\rightarrow} \leftarrow \frac{a_{jj} * a_i^{\rightarrow} - a_{ij} * a_j^{\rightarrow}}{a_{j-1,j-1}}. \qquad (22)$$

Замечание

Под делением нацело понимается возможность деления чисел с меньшей разрядностью, чем требуется для хранения числителя.

В результате получается, что определитель исходной матрицы, подлежащей факторизации, равен правому нижнему элементу треугольной по Барейсу матрицы (в обозначениях, принятых в данной статье, это a_{nn}).

Но применение данного алгоритма при определении текущих значений коэффициентов также оставляет операцию деления и, как метод Гаусса, вносит прерывание в вычислительный поток умножений с помощью СУ.

Решением данной проблемы может быть вычисление $a^{-1}{}_{j-1,j-1}$ (обратного значения от диагонального элемента, полученного на предыдущем $(j-1)$-м шаге):

$$L_{j-1,j-1} = 1/a_{j-1,j-1} = a^{-1}{}_{j-1,j-1}. \qquad (23)$$

Его вычисление можно осуществлять с помощью Host-процессора, который параллельно с СП обслуживает ПЭ, где хранится коэффициент $a_{j-1,j-1}$.

Предлагается алгоритм (22) изменить и процедуру факторизации матрицы на прямом ходе выполнять с помощью выражения

$$a_i^{\rightarrow} \leftarrow (a_{jj} * a_i^{\rightarrow} - a_{ij} * a_j^{\rightarrow}) * a_{j-1,j-1}^{-1}. \qquad (24)$$

В целом процесс упрощается. По сути вычислений. На обратном ходе решения задачи по предложенному ранее алгоритму вычисление $a^{-1}{}_{j-1,j-1}$ необходимо для определения неизвестных. Поэтому вычисление $a^{-1}{}_{j-1,j-1}$ на прямом ходе снимает потребность в его выполнении при обратном ходе алгоритма, реализующего выражение (4). Но $a^{-1}{}_{j-1,j-1}$ надо запомнить в кэш-памяти ПЭ, где хранится массив чисел, которые данный ПЭ будет использовать при выполнении обратного хода вычислений. А всего в процессе факторизации матрицы на прямом и обратном ходе следует выполнить лишь $(n-1)$ операций вычисления обратного значения величин (равносильно делению).

4. Выводы

1. Решение систем $Ax = b$ с помощью параллельной КС с ПЭ на базе СУ в режиме ЕТП возможно. При реализации выражения (24) на СУ добавляется операция умножения на $a^{-1}{}_{j-1,j-1}$, но сокращаются операции деления.

2. При факторизации матрицы и выполнении текущего цикла пересчета коэффициентов в каждой строке:

– по Гауссу с делением на прямом ходе для определения текущего значения каждого коэффициента необходимо выполнять одну операцию деления и операции FMA ($A * B + C$);

– по предлагаемому способу без деления на прямом ходе следует выполнять операции FM + FMA ($A * B + C * D$) и сдвиг (полученного результата на один разряд в сторону старшего при нормализации мантисс редуцируемого массива чисел);

– по модифицированному алгоритму Барейса необходимо для реализации процедуры прямого хода выполнить операции вида FM+FMA+FM [$(A * B + C * D) * L$] с элементами строки факторизуемой матрицы. Операция получения обратной величины L от диагонального элемента, равносильная делению, выполняется на прямом ходе. Но её выполнения требует и процедура обратного хода вычисления. В целом на решение уравнения $Ax = b$ приходится всего $(n-1)$ операция деления.

3. По поводу выбора ведущего элемента на прямом ходе исключения.

Задачи МФ, описываемые дифференциальными уравнениями в частных производных, после аппроксимации их с помощью МКР или МКЭ, приводятся к СЛАУ с главным диагональным преобладанием.

Операция умножения не исключает операцию умножения на ноль, в то время как для деления это является исключением в силу определения самой операции деления.

Если учесть, что для реализации обратного хода решения системы $Ax = b$ вычисление значений $a^{-1}{}_{j-1,j-1}$ (23) обязательно, то применение описанных способов будет и более рациональным по сравнению с традиционным методом Гаусса.

Использование максимального (по своему значению) в качестве ведущего элемента всегда дает лучшие результаты. Однако все известные автору «плохие» примеры, приведенные Дж.Форсайтом, Райсом, Беклемишевым, Барейсом и др. специалистами, были решены автором без выбора ведущего элемента и следующих за ним соответствующих перестановок строк.

Заключение

1. В сфере разработки и создания архитектур ВС сверхвысокой производительности наметился явный идейный возврат. Например, от создания кластеров на базе одно- и многоядерных процессоров классической структуры переходят к гетерогенным кластерным структурам с включением в свой состав графических спецпроцессоров (фирм nVIDIA и AMD с доработками технологий CUDA и STREAM соответственно). Но любая серьезная модернизация существующих систем (как и любого серийно выпускаемого оборудования) связана с затратами, гораздо большими, чем создание новых систем (или оборудования). Поэтому дешевле, эффективней и целесообразней обогатить гамму спецпроцессоров кластерных систем на базе CPU разработкой и (главное) созданием СП для решения общенаучных и технических проблем. В частности, создание КС СП для задач МФ с использованием полученных научных и технических заделов разработки архитектуры с идеологией ЕТП и обработкой ССД может послужить основой создания универсальных СП (или целого класса СП) для решения общих задач. Естественно, с учетом опыта создания и применения имеющихся кластерных архитектур ВС на базе GPU и CPU с доработками для решения общих задач.

2. Применение изложенных алгоритмов хорошо вписывается в общий вычислительный процесс решения задач МФ с помощью однотипных математических выражений и в структуру СП, позволяет упростить расчеты в целом и выполнять их на однотипных ПЭ с СУ в его основе. Это решает проблему загрузки и использования системы, поскольку исходные данные формируются в самом процессорном элементе, а далее участвуют в решении задачи в целом.

3. Предложенные и описанные способы решения СЛАУ вписываются в вычислительный поток обработки данных на аппаратуре с применением МКЭ. Аппроксимация с помощью МКЭ при регулярном матричном представлении (как показано в [1]), позволяет от последовательной обработки чисел переходить к обработке полиномов за время, соизмеримое со временем обработки пары чисел. А архитектура (кластерный СП из ПЭ с СУ в их основе) обеспечивает переход к поточной параллельной обработке сложных структур данных (полиномов) с распараллеливанием вычислительного процесса на уровне операции (над ССД) и в целом – всей задачи.
4. После вычисления глобальной матрицы жесткости значения коэффициентов уже находятся в памяти ПЭ. Исключается простой матричного устройства и вычислительного процесса. В целом повышается производительность СП. Особо следует отметить, что такая организация структуры СП из ПЭ в режиме ЕТП исключает затраты по времени на обращения за исходными данными к глобальной памяти устройства.
5. Проблему факторизации матрицы можно решать с помощью GPU и кластерных CPU (на базе универсальных ЭВМ) с соответствующими доработками. Преимущество применения предлагаемого способа в том, что решение задач МФ можно выполнять в режиме ЕТП. Применение МКЭ сокращает размерность решаемых систем, дает матрицу СЛАУ с высокой степенью заполнения, что оправдывает использование прямых методов решения систем, которые быстрее итерационных.
6. Возможность получения решений с высокой точностью (при работе с числами удвоенной значности над той частью выражения, которая стоит в скобках (24)) может представлять особый интерес, если учесть имеющиеся методы умножения чисел в прямом и дополнительном кодах.
7. Предлагаемые решения подходят для методов последовательных приближений, итерационных уточнений, обращения матриц, вычисления невязок, при исправлении элементов приближенной матрицы и др. случаях решения СЛАУ.
8. Предлагаемый алгоритм полезен и при решении разного рода задач, когда в расчетах приходится вычислять значения неизвестных в системах малой размерности, где коэффициенты подконтрольны экспериментатору. А при работе с целочисленными значениями вычисления по прямому алгоритму Барейса, и с коррекцией по (24), обеспечат получение результата.
9. Работа с полиномами, представленными в виде РМП, предполагает запись матрицы в виде верхней треугольной (типа U), где на главной диагонали стоит один и тот же коэффициент. Это означает единственность деления за весь процесс факторизации матрицы. А часто этот коэффициент равен единице. В таких случаях вычисление обратной матрицы $\left(U^{-1}\right)$ упрощается. Все это позволяет упростить процесс определения неизвестных в выражении, где полином стоит в знаменателе. Можно уйти от операции деления, сведя её к умножению на обратную матрицу типа U^{-1}, и выполнять с помощью СУ.

В заключение считаю приятным долгом выразить свою благодарность И.В. Фрязинову и М.Ф. Яковлеву за полезные советы и замечания, высказанные при подготовке статьи к печати.

СПИСОК ЛИТЕРАТУРЫ

1. Ледянкин Ю.Я. Единый технологический поток в организации вычислений – способ повышения производительности параллельных структур на процессорных элементах транспьютерного типа / Ледянкин Ю.Я. – Киев, 1989. – 20 с. – (Препринт / АН УССР. Ин-т кибернетики им В.М. Глушкова; 89-57).
2. Ледянкин Ю.Я. Метод приращений для решения уравнений $Ay = f$ / Ледянкин Ю.Я. // ЖВМ и МФ. – 1979. – Т. 19, № 4. – С. 1043 – 1047.

3. А.с. 1619354 СССР. Скалярный умножитель векторов / Ю.Я. Ледянкин, В.А. Вышинский. – Опубл. в Б.И. 1989.
4. Ледянкин Ю.Я. Ускоренный метод умножения чисел, представленных в последовательном дополнительном коде / Ледянкин Ю.Я. // Автоматика. – 1989. – № 3. – С. 76 – 82.
5. Ledyankin Yu.Ya. Accelerated Method of Multipying Numbers Represented in Sequential Code / Yu.Ya. Ledyankin // Soviet Journal of Automation and Information Sciens. – 1989. – N 3. – P. 88 – 94.
6. Адинец А. Графический вызов суперкомпьютерам / А. Адинец, В. Воеводин // Открытые системы. – 2008. – № 4. – С. 35 – 41.
7. Губайдуллин Д.А. Об особенностях использования архитектуры гетерогенного кластера для решения задач механики сплошных сред / Д.А. Губайдуллин, А.И. Никифоров, Р.В. Садовников // Вычислительные методы и программирование. – 2011. – № 12. – С. 450 – 460.
8. Bareiss E.H. Sylvester`s Identity and Multistep Integer-Preserving Gaussian Elimination / E.H. Bareiss // Mathematics of Computation. – 1968. – Vol. 22, N 103. – P. 565 – 578.

Стаття надійшла до редакції 27.09.2012