Многопоточная параллельная реализация итерационного алгоритма решения систем линейных уравнений с динамическим распределением нагрузки по нитям вычислений

Г.Б. Сушко, С.А. Харченко

В процессе решения задач вычислительной гидродинамики в пакете “FlowVision” с использованием неявных численных методов возникает необходимость решения систем линейных уравнений, описываемых неструктурированными разреженными матрицами большой размерности. В данной работе для решения таких систем уравнений предлагается вычислять неполное разложение матрицы высокого порядка точности (ICH2/ILU2) и затем решать систему уравнений с использованием предобусловленных итерационных алгоритмов типа подпространства Крылова. Для уменьшения времени вычислений этот алгоритм распараллеливается по общей памяти с динамическим распределением нагрузки на основе технологий OpenMP и Intel® Threading Building Blocks (TBB). Для явного выделения параллелизма в зависимых вычислениях факторизации и при решении систем уравнений с треугольными матрицами использовалось упорядочивание вложенных сечений Nested Dissection (ND). Приводятся результаты численных экспериментов по масштабируемости предложенных алгоритмов на многоядерных процессорах для систем линейных уравнений, возникающих при моделировании задач вычислительной гидродинамики в пакете “**FlowVision**”.

1. Введение

В процессе решения задач вычислительной гидродинамики в пакете “**FlowVision**” с использованием неявных численных методов возникает необходимость решения систем линейных уравнений

$$Ax = b \tag{1}$$

с неструктурированными сильно разреженными матрицами большой размерности. В настоящее время для эффективного решения таких систем линейных алгебраических уравнений (СЛАУ) традиционно используются предобусловленные итерационные методы.

В данной работе в качестве предобуславливателя используется предобуславливатель ILU2, являющийся несимметричным обобщением предобуславливателя ICH2[1]. В работе [1] доказано существование и устойчивость этого неполного разложения для произвольных симметричных положительно-определенных матриц, а также получены оценки скорости сходимости метода сопряженных градиентов с этим предобуславливанием. На практике предобуславливатель ICH2 и ILU2 показали свою высокую надежность и эффективность при решении разнообразных систем линейных уравнений. В качестве итерационной схемы используются методы типа подпространства Крылова: метод сопряженных градиентов CG [2] в симетричном случае, алгоритм GMRES [3] и алгоритм Ланцоша [2] в несимметричном случае.

В настоящее время большая часть производимых процессоров состоит более чем из одного вычислительного ядра. Ведущие производители процессоров, такие как компания Intel, аннонсируют появление в ближайшее время внутренне хорошо масштабируемых процессоров с большим (>=16) числом вычислительных ядер. Учитывая все возрастающую потребность решения задач большой размерности, возникает необходимость научиться эффективно использовать новые возможности вычислительной техники.

Существуют различные варианты использования многоядерных процессоров. Многоядерные процессоры можно рассматривать как многопроцессорную систему с распределенной памятью, учитывая накопленное количество программного обеспечения, написанного на стандарте MPI. Недостатком такого подхода является необходимость заранее сбалансировать вычисления и минимизировать обмены, поскольку основанные на MPI алгоритмы используются и на чисто распределенных системах, а значит должны быть там эффективны. С другой стороны, явное использование коллективного доступа к общей памяти в

многоядерных процессорах дает уникальную возможность динамического выбора ядра для текущих необходимых вычислений. Для хорошо внутренне масштабируемых многоядерных процессоров это означает потенциальную возможность добиться хорошей сбалансированности загрузки ядер за счет явной динамической балансировки нагрузки при вычислениях.

В данной работе описывается параллельная реализация алгоритма решения СЛАУ с предобуславливатилем ICH2/ILU2 и итерационными схемами типа подпространства Крылова для компьютеров с общей памятью с динамической балансировкой зависимых вычислений. Предлагаемые в данной работе подходы к распараллеливанию алгоритмов решения СЛАУ для компьютеров с общей памятью могут применяться и при распараллеливании других способов предобуславливания типа неполной треугольной факторизации.

2. Зависимости по данным алгоритма решения СЛАУ

Предобусловленный итерационный алгоритм решения СЛАУ включает в себя: построение предобуславливателя (неполная факторизация), умножение матрицы на вектор, решение систем уравнений с верхней и нижней треугольными матрицами и векторные операции. Рассмотрим некоторые детали соответствующих алгоритмов чтобы прояснить зависимости по данным.

2.1 Неполная факторизация

Рассмотрим кратко алгоритм (точной) треугольной факторизации – разложения матрицы A в произведение нижней L и верхней U треугольных матриц. В блочном виде рекуррентное соотношение этой операции можно записать следующим образом:

$$\begin{bmatrix} A_{11} & a_{12} & A_{13} \\ a_{21} & a_{22} & a_{23} \\ A_{31} & a_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ L_{31} & l_{32} & L_{33} \end{bmatrix} \cdot \begin{bmatrix} U_{11} & u_{12} & U_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & U_{33} \end{bmatrix}. \quad (2)$$

Здесь блок a_{22} имеет размер 1, блоки $a_{21}, a_{23}, l_{21}, u_{23}$ - строки, а $a_{12}, a_{32}, l_{32}, u_{12}$ - столбцы. Рекуррентный обход матрицы производится последовательно по строкам U и столбцам L сверху вниз, т.е. блоки, L_{i1}, U_{1j}, $i, j = 1,\ldots,3$, уже вычислены, блоки $l_{22}, l_{32}, u_{22}, u_{23}$ рассчитываются на данном шаге, а блоки L_{33} и U_{33} пока не известны. На каждом шаге вычислений необходимо:

1. Из уравнения $l_{22}u_{22} = a_{22} - l_{21}u_{12}$ вычислить значения l_{22}, u_{22}.
2. Из уравнений $u_{23} = (a_{23} - l_{21}U_{13})l_{22}^{-1}$ и $l_{32} = (a_{32} - L_{31}u_{12})u_{22}^{-1}$ найти значения u_{23} и l_{32}.

Различные варианты неполных разложений, в том числе и ICH2/ILU2 [1], получаются из описанного выше алгоритма путем некоторой модификации шагов 1 и 2.

Зависимость по данным в этом алгоритме для текущей строки U показана на Рис. 1, для столбца L зависимость по данным аналогична. Таким образом, при вычислении текущей строки U/столбца L используются некоторые ненулевые значения из уже посчитанных строк U/столбцов L.

Рис. 1 Зависимость по данным при факторизации строки U

2.2 Умножение матрицы на вектор

Запишем формулу вычисления результата умножения матрицы на вектор:

$$y_i = \sum_{j=0}^{n-1} A_{ij}x_j, i = \overline{0, n-1}. \quad (3)$$

Из формулы (3) следует, что в вычислениях результата умножения рекуррентная зависимость по данным отсутствует. Вместо этого имеет место зависимость по данным только от некоторых данных входного вектора, а именно только для тех данных, для которых соответствующие коэффициенты в текущей строке разреженной матрицы A ненулевые.

2.3 Решение СЛАУ с треугольной матрицей

Для определения зависимостей при решении треугольных СЛАУ запишем рекуррентное соотношение для решения треугольной системы с матрицами L и U:

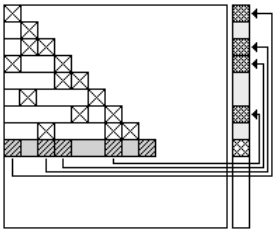

Рис. 2 Зависимость по данным при решении системы L

$$\mathrm{L}: x_i = \frac{f_i - \sum_{j=0}^{i-1} L_{ji} x_j}{L_{ii}}, i = \overline{0, n-1}; \qquad (4)$$

$$\mathrm{U}: x_i = \frac{f_i - \sum_{j=i+1}^{n-1} U_{ji} x_j}{U_{ii}}, i = \overline{n-1, 0}. \qquad (5)$$

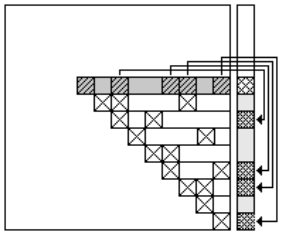

Рис. 3 Зависимость по данным при решении системы с U

Из соотношений видно, что вычисление величины x_i не может быть проведено до вычисления тех x_j, для которых значение коэффициентов по i-й строке L/i-й строке U не равно нулю, см. Рис. 2-3. Таким образом структура разреженности внедиагональных элементов матрицы определяет зависимости по данным в вычислениях неизвестных вектора решения.

3. Явное выделение параллелизма и упорядочивание вложенных сечений

Для явного выделения параллелизма вычислений предлагается использовать подходящую нумерацию переменных, при которой явным образом выделяются независимые и зависимые вычисления [7, 8]. Пусть для некоторой нумерации неизвестных и некоторого целого p>1 матрица системы уравнений имеет следующий блочный вид:

$$A = \begin{bmatrix} A_1 & & 0 & C_1 \\ & \ddots & & \vdots \\ 0 & & A_p & C_p \\ B_1 & \cdots & B_p & D \end{bmatrix}. \qquad (6)$$

В таком случае мы будем говорить, что матрица системы уравнений имеет p-блочно окаймленный вид, неявно предполагается, что диагональные блоки $A_1, \ldots, A_p$ - квадратные.

Блочное представление (6) интересно тем, что в нем явно выделяются независимые вычисления для p процессоров. Действительно, при факторизации строк U/столбцов L из подматрицы A_i данные факторизации из других подматриц A_j, $j \neq i$, не участвуют. Этот факт легко следует из Рис.1, поскольку в представлении (6) вне главной блочной диагонали в блочной позиции (i,j) и (j,i) стоят нулевые матрицы, $1 \le i \le p$, $1 \le j \le p$, $i \neq j$. Легко показать, что блочная разреженность неполных факторов L и U в представлении (6) не выходит за рамки блочной разреженности, указанной в (6). По этой причине при решении треугольных систем уравнений с L и U снова явно выделяются блоки независимых вычислений. Междублоковая зависимость в вычислениях факторизации и при решения систем уравнений с треугольными матрицами появляются только при обработке общего окаймления, отвечающего диагональному блоку D.

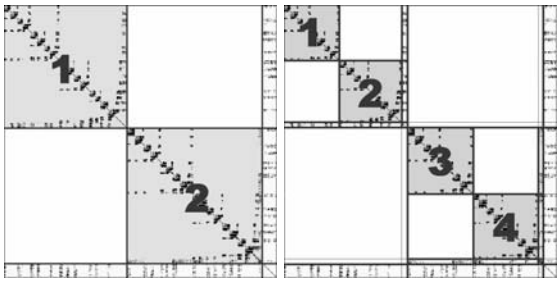

Рис. 4 ND разбиение на 1 и 2 уровня рекурсии

Представление (6) мы будем использовать в частном случае p=2. Хорошо известное упорядочивание вложенных сечений Nested Dissection (ND) [4] строит представление (6) для p=2 с минимальным размером разделителя – размером матрицы D, и максимально близкими размерами блоков A_1 и A_2. Далее это представление снова строится уже для

диагональных блоков A_j, и т.д. В результате получается упорядочивание, показанное на Рис. 4, на рисунке показаны первый и второй уровни рекурсии, можно выделять и дальнейшие уровни. Уровням рекурсии ND соответствует некоторое бинарное дерево. На Рис. 5 показан пример бинарного дерева, которому соответствует разбиение матрицы до трех уровней рекуррентного биения. Корень этого дерева соответствует самому внешнему окаймлению, листья дерева на верхнем уровне – самым внутренним диагональным блокам, элементы дерева на промежуточных уровнях – соответствующим окаймлениям.

4. Параллельные алгоритмы

На основе упорядочения по вложенным сечениям и соответствующего ему бинарного дерева можно описать зависимости вычислений и предложить динамические параллельные алгоритмы неполной факторизации и решения систем с треугольными факторами.

4.1 Неполная факторизация и решение СЛАУ с треугольными матрицами

Каждому узлу бинарного дерева соответствует некоторый набор строк/столбцов упорядоченной по ND матрицы коэффициентов. Зависимость по данным – от листьев дерева к корням. Сначала можно в произвольном порядке вычислять неполную факторизацию данных, соответствующих листьям верхнего уровня бинарного дерева зависимостей. Неполную факторизацию блоков, соответствующих остальным узлам дерева зависимостей, можно вычислять как только готова пара данных выше по дереву. Таким образом, имеет место возможность асинхронных параллельных вычислений при построении неполной факторизации. При этом последующее вычисление можно проводить по мере готовности результатов вычислений для предыдущих двух узлов дерева вне зависимости от состояния остальных узлов дерева.

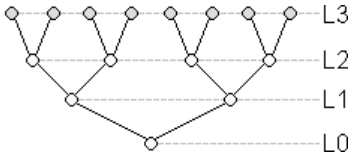

Рис. 5 Бинарное дерево зависимых вычислений

При решении треугольной системы с L имеют место зависимости от листьев дерева к корням. Сначала в произвольном порядке можно вычислить неизвестные, соответствующие листьям верхнего уровня бинарного дерева зависимостей. Затем можно вычислять неизвестные любого узла дерева зависимостей как только закончены вычисления с двумя узлами, от которых этот узел зависит.

При решении треугольной системы с U имеют место зависимости от корней дерева зависимостей к листьям. Сначала нужно вычислить неизвестные, соответствующие корню дерева, затем в произвольном порядке данные для узлов, исходящих из корня, и т. д. При этом два последующих вычисления можно проводить в любом порядке по мере готовности результата для предыдущего узла дерева вне зависимости от состояния остальных узлов дерева.

4.2 Data Parallelism vs. Task Parallelism

При реализации параллельных алгоритмов естественно выделять следующие типы параллельных вычислений:

- набор вычислений, притом такой, что каждое вычисление никак не зависит от результатов других вычислений, будем условно называть *независимыми вычислениями*;
- набор вычислений, среди которых имеются вычисления, зависящие от результатов других вычислений из этого же набора, будем условно называть *зависимыми вычислениями*.

Примером независимых вычислений являются умножение матрицы на вектор и операции с векторами в итерационной схеме. Примером зависимых вычислений являются неполная факторизация и решение систем с неполными треугольными факторами. С другой стороны, в алгоритмах неполной факторизации и решения систем уравнений с неполными треугольными факторами вычисления с различными диагональными блоками представления (6) являются независимыми между собой в контексте блочных вычислений; зависимыми в блочном контексте для этих алгоритмов являются вычисления с окаймлением.

С каждым типом параллельных вычислений естественно связать соответствующую методологию распараллеливания. Методология Data Parallelism – это способ распараллеливания независимых вычислений, при котором выделяются независимые

подзадачи, каждая из которых может исполняться параллельно. Примером программной реализации методологии Data Parallelism является цикл #pragma omp parallel for в стандарте параллельного программирования OpenMP, который является фактически директивой компилятору о создании параллельных потоков и автоматическом разбиении цикла на части с выбором (в зависимости от опций цикла) какой поток какую часть цикла исполняет. Методология Task Parallelism – это способ распараллеливания зависимых вычислений, при котором выделяются зависимые подзадачи, часть из которых может исполняться параллельно. При программной реализации такого подхода выделяется так называемый *планировщик заданий*, который, используя граф зависимостей, определяет, какое из вычислений может выполняться в следующий момент времени. На многоядерных вычислительных системах каждый поток исполнения, называемый thread (нить) в англоязычной литературе, обращаясь к этому планировщику, получает от него задание, выполняет его и сообщает планировщику о его исполнении. Таким образом осуществляется динамическое распределение заданий по потокам, что позволяет, как ожидается, сбалансированно распределять нагрузку по потокам.

4.3 Параллельная реализация

Для реализации параллельной обработки данных на компьютерах с общей памятью используются библиотеки работы с потоками. В данной работе использовались технологии OpenMP и Intel® Threading Building Blocks (TBB). Концепция Task Parallelism, реализуемая в алгоритмах параллельной неполной факторизации и решения треугольных СЛАУ, напрямую поддерживается в библиотеке Intel® TBB, и не поддерживается в OpenMP. С другой стороны концепция Data Parallelism, более подходящая для независимых вычислений типа умножения матрицы на вектор и обработки векторов, удобнее реализована в OpenMP.

Для реализации в OpenMP зависимых вычислений по методологии Task Parallelism была предпринята попытка использовать стандартный для OpenMP механизм lock синхронизации потоков. В стандарте OpenMP отсутствует механизм снятия блокировки lock'а потоком, отличным от потока, установившего lock. Из-за этого не удается избежать бесконечных циклов опрашивания lock'а потоком, для которого нет работы. По этой причине вместо lock-ов стандарта OpenMP при реализации планировщика задач использовался механизм event операционной системы, реализованный на основе WinAPI (Windows) или pthreads (Unix/Linux).

5. Результаты

Для демонстрации возможностей текущей реализации параллельных алгоритмов решения систем уравнений была выбрана существенно трехмерная тестовая задача о течении воздуха в смесителе. Геометрия задачи показана на Рис. 6. Для этой задачи была построена расчетная сетка с количеством расчетных ячеек N = 405600. Декомпозиция тестовой задачи на 2, 4 и 8 процессоров на основе упорядочивания вложенных сечений показана на Рис.6.

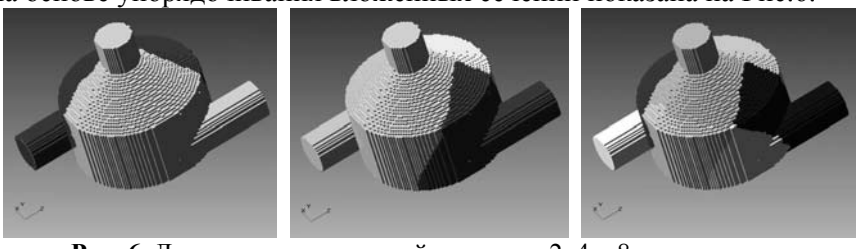

Рис. 6. Декомпозиция тестовой задачи на 2, 4 и 8 процессоров.

Для тестовой задачи рассматривалась система линейных уравнений для давления на некотором шаге по времени. Эксперименты проводились на двухпроцессорном компьютере. Каждый процессор - Intel 2xDualCore Intel Xeon 5310 - четырех- ядерный процессор с тактовой частотой 1.6ГГц. Шина памяти работает на частоте 1066 МГц. Память FB DIMM DDR2-667 объемом 8Гб. Общее количество доступных ядер 2x4=8.

Ускорение при решении тестовой системы линейных уравнений при распараллеливании по OpenMP представлены на Рис. 7. Результаты для TBB оказались аналогичными и не приводятся для экономии места. На Рис. 7 слева представлены результаты в случае полной оптимизации программы при компиляции (опция компиляции "–O3"), на Рис. 7 справа представлены результаты для отключенной оптимизации программы при компиляции (опция "–O0").

Представленные результаты численных экспериментов позволяют сделать следующие выводы. Прежде всего, неплохое ускорение программы (больше 5 на 8 процессорах) в случае отключенной оптимизации при компиляции означает хорошую масштабируемость собственно параллельного алгоритма на тестовых данных. С другой стороны, низкая масштабируемость той же программы при полной оптимизации означает, что процессоры считают очень быстро, а данные для них не успевают поступать, особенно на итерациях. Учитывая, что при любых опциях оптимизации данные в процессор из оперативной памяти поступают в среднем с одной и той же скоростью получается, что на тестовом компьютере с общей памятью узким местом с точки зрения масштабируемости параллельных вычислений является шина общей памяти.

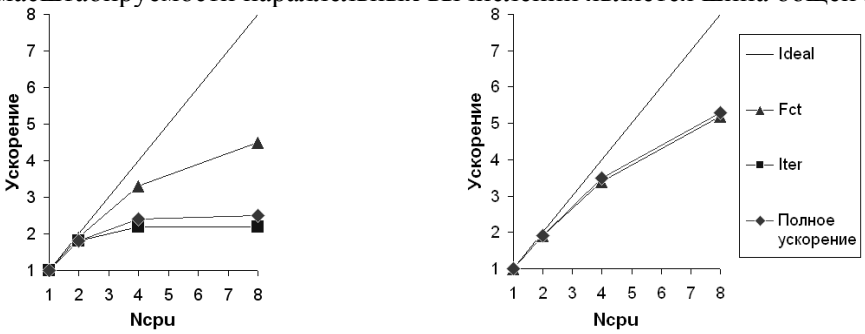

Рис. 7. Ускорение для тестовой задачи при компиляции с полной оптимизацией "–O3"(слева) и при компиляции без оптимизации "-O0" (справа).

6. Заключение

Целью данного исследования является разработка масштабируемого параллельного алгоритма решения систем линейных уравнений на многоядерных процессорах. Представленные результаты показывают, что предложенные в работе алгоритмы распараллеливания по общей памяти обладают большим ресурсом параллелизма и демонстрируют результаты, близкие к оптимальным. К сожалению, динамическая балансировка нагрузки при распараллеливании по общей памяти не улучшает существенным образом масштабируемость алгоритмов. Из результатов экспериментов следует, что это связано с недостаточной пропускной способностью шины общей памяти, которая не позволяет обеспечить вычислительной нагрузкой высокопроизводительные ядра. Авторы предполагают, что дальнейшее развитие многоядерных процессоров ведущими производителями приведет к таким технологическим решениям, которые снимут проблему производительности механизмов взаимодействия многоядерных процессоров при работе с общей памятью.

Литература

1. I.E.Kaporin. "High Quality Preconditioning of a General Symmetric Positive Definite Matrix Based on its $U^TU+U^TR+R^TU$ decomposition". Numer. Linear Algebra Appl., 5, 483-509 (1998).
2. Тыртышников Е.Е. Краткий курс численного анализа. Москва: ВИНИТИ. 1994.
3. Saad Y., Schultz M.H., "GMRES: A generalized minimum residual algorithm for solving non-symmetric linear systems". SIAM J. Sci. Comput. 1986. 7. 856-869.
4. George A., Liu J.W., "Computer Solution of Large Sparse Positive Definite Systems", Prentice Hall, 1981.
5. OpenMP Application Program Interface - 2.5 / OpenMP Architecture Review Board – 2005.
6. Intel Threading Building Blocks Tutorial – 1.6 / Intel Corp. 2007.
7. Харченко С.А. "Параллельная реализация алгоритма решения систем линейных уравнений в пакете FlowVision", в сборнике "Прикладные исследования в механике", труды конференции "Инженерные системы -2007", 135-144, 2007.
8. А.А. Дядькин, С.А. Харченко. "Алгоритмы декомпозиции области и нумерации ячеек с учетом локальных адаптаций расчетной сетки при параллельном решении систем уравнений в пакете FlowVision". Труды Всероссийской научной конференции "Научный сервис в сети Internet: многоядерный компьютерный мир", 201-206, 2007.