Параллельный метод LU-SGS для трехмерных задач газовой динамики со сложной геометрией на вычислительных системах с многими графическими ускорителями

П.В. Павлухин[1], И.С. Меньшов[1,2]

Московский Государственный Университет им М.В. Ломоносова[1], Институт прикладной математики им М.В. Келдыша Российской академии наук[2]

В статье представлен эффективный алгоритм параллельного расчета трехмерных газодинамических течений со сложной геометрией на вычислительных системах с многими графическими ускорителями. Неявная схема, лежащая в основе алгоритма, приводит к большим линейным системам с сильно разреженной матрицей, которые решаются методом приближенной факторизации LU-SGS (Lower-Upper Symmetric Gauss-Seidel). Параллельный алгоритм точно воспроизводит работу последовательного и обладает высокой степенью масштабируемости. Для описания геометрии расчетной области используется метод VOF (Volume Of Fluid).

1. Введение

В механике жидкостей и газов в рамках модели уравнений Навье-Стокса задачи прямого численного моделирования требуют для решения значительных вычислительных и временных ресурсов. Экстенсивное наращивание производительности вычислительных систем за счет повышения тактовой частоты процессоров закончилось несколько лет тому назад, когда она вплотную приблизилась к отметке в 4 Ггц, поэтому единственной возможностью вести расчеты за приемлемое время осталось одновременное использование большого числа вычислительных ядер. Но здесь возникают две фундаментальные проблемы. Первая из них – это построение корректного параллельного алгоритма: результат его работы должен быть идентичен работе последовательного. Многие параллельные реализации численных методов (как правило, неявных) соблюдают эту идентичность лишь на определенных подобластях исходной расчетной области. Как результат, в процессе вычислений накапливаются ошибки, связанные с различиями в последовательной и параллельной версиях алгоритма. Дополнительно, сама архитектура вычислительной системы накладывает ограничения на структуру параллельного алгоритма. Вторая проблема – масштабируемость параллельных программ. Как правило, они могут эффективно использовать лишь порядка сотни процессорных ядер и 2 – 3 графических ускорителя, установленных внутри одного узла. Падение эффективности при задействовании бóльшего числа ресурсов связано как со скоростными характеристиками сетей передачи данных вычислительной системы, так и с ограниченной масштабируемостью самого параллельного алгоритма (безотносительно его конкретных реализаций).

Все чаще современные суперкомпьютеры используют гибридную структуру – в дополнение к «классическим» процессорным ядрам устанавливаются графические ускорители, которые и дают основной вклад в суммарную производительность системы (на многих приложениях производительность GPU на 1 – 2 порядка превосходит производительность многоядерных центральных процессоров). Однако сложность массивно-параллельной архитектуры графических плат значительно затрудняет написание программ для них. В абсолютном большинстве случаев программы для GPU используют только явные схемы с простыми вычислительными ядрами и способны задействовать лишь несколько ускорителей, находящихся внутри одного узла. Помимо сложностей программной модели, архитектура мультитредовых графических процессоров обуславливает трудности при построении алгоритмов для них – в частности, отсутствие механизма глобальной синхронизации потоков является одной из основной причин отсутствия реализаций неявных методов для задач механики жидкостей и газов на GPU.

Проблема построения алгоритмов одновременного счета на узлах системы с распределенной памятью без единого адресного пространства уже является общей для CPU и GPU архитек-

тур. Для неявных схем она решается, как правило, лишь частичным соблюдением работы последовательного алгоритма на подобластях расчетной области, т.е., вообще говоря, параллельный алгоритм нарушает работу последовательного. Версии параллельных алгоритмов с идентичной последовательной работой, как, например, в [1], имеют уже теоретическое ограничение по масштабируемости, которая еще более ухудшается при программной реализации под конкретную вычислительную систему.

Наряду с рассмотренной выше проблемой особую важность имеет также подготовка исходных данных задачи, а именно построение сеточной геометрии расчетной области. В задачах со сложной геометрией этот процесс трудно автоматизируется, поэтому «ручное» построение высококачественных расчетных сеток по времени зачастую сравнивается с полным счетом всей задачи. Альтернативный предлагаемый подход основан на методе Volume of fluid (VOF) [2 – 4], в котором используются простые структурированные сетки, а вся информация о геометрии (поверхностях) расчетной области описывается т.н. характеристической функцией. Данный подход позволяет вести эффективный расчет трехмерных задач со сложной геометрией на GPU без затрат времени на построение сеток.

Большинство программных реализаций численных методов, во-первых, ограничивают размер расчетной области суммарной памятью доступных графических плат, и во-вторых, для запуска задач с начальными данными объемом более десятка гигабайт минимально требуется от двух и более ускорителей. Эти два ограничения связаны с тем, что для хранения данных в процессе счета (за исключением пересылаемых между ускорителями т.н. ghost-ячеек) используется только видеопамять GPU, составляющая в последних моделях не более 5 - 6 ГБ, что более чем на порядок меньше оперативной памяти в вычислительных узлах современных кластеров. Между тем, карты серии Nvidia Tesla поддерживают возможность одновременного выполнения трех операций: счет вычислительного ядра и копирование данных между оперативной памятью и памятью GPU в обоих направлениях. Благодаря этому можно снять ограничение по объему видеопамяти, выполняя одновременно *выгрузку* из GPU уже обработанных данных, *расчет* над предварительно загруженными данными и *загрузку* в GPU для последующего расчета новой порции. Таким образом, если в этой схеме время перемещения данных по шине pci-express будет меньше соответствующего времени расчета, то без потерь производительности можно задействовать для хранения данных всю доступную оперативную память, не ограничиваясь лишь объемом видеопамяти.

В настоящей работе будет рассмотрен параллельный алгоритм неявного метода LU-SGS (Lower-Upper Symmetric Gauss-Seidel) [5 – 7] для трехмерных задач газовой динамики в областях со сложной геометрией, описываемой VOF-методом. Его реализация выполняется с помощью технологии CUDA и MPI для multi-GPU кластеров с возможностью использования всей доступной оперативной памяти.

2. Численный метод

Рассматривается модель сжимаемой жидкости, определяемая системой уравнений Навье-Стокса в декартовых координатах, которые дискретизируются по пространственным переменным методом конечного объема. Применяемая явно-неявная схема интегрирования по времени приводит к системе дискретных уравнений, решаемой методом установления по псевдовременной переменной с использованием неявной дискретизации и ньютоновских итераций. Таким образом, для определения итерационного инкремента $\delta^s \vec{q}$ получается следующая линейная система:

$$(D+L+U)\delta^s \vec{q}_i = -\vec{R}_i^{n+1,s} \tag{1}$$

где D – блочно-диагональная матрица, а L и U - блочно-диагональные нижняя и верхняя треугольные. Уравнение (1) преобразуется к следующему виду:

$$(D+L)D^{-1}(D+U)\delta^s \vec{q}_i = -\vec{R}_i^{n+1,s} - LD^{-1}U \tag{2}$$

Для систем такого вида в методе LU-SGS выполняется приближенная факторизация левой части уравнений (1), которая получается, если в уравнениях (2) пренебречь последним слагае-

мым правой части (которое фактически пропорционально Δt^2). После факторизации уравнения распадаются на две подсистемы:

$$\begin{aligned}(D+L)\delta^s\vec{q}_i^{\,*} &= -\vec{R}_i^{n+1,s}\\ (D+U)\delta^s\vec{q}_i &= D\delta^s\vec{q}_i^{\,*}\end{aligned} \tag{3}$$

Первая система уравнений в (3) имеет нижне-треугольную блочную матрицу, каждый элемент которой представляется блоком размерностью (5 x 5), а вторая – соответственно верхнетреугольную. Такая структура матриц систем позволяет эффективно вычислить их решения за два расчетных цикла по ячейкам: первый - в прямом направлении (от первой ячейки к последней), а второй - в обратном. При этом необходимо обращать только диагональные блоки, т. е., матрицы размером (5 x 5). Получающаяся при этом итерационная невязка $\delta^s\vec{q}$ служит для обновления итерационного вектора, после чего процедура (3) повторяется. Более подробное описание метода приведено в [5 – 7]

Для счета задач со сложной геометрией применяется VOF-метод [2 – 4], который, в частности, используется для отслеживания поверхностей-границ в многокомпонентных течениях. В расчетной области предполагается наличие непроницаемых твердых тел со сложной геометрией. Изначально вся область погружается в т.н. расчетный куб, для которого строится декартова сетка (со сгущением координатных линий там, где это необходимо). Затем вычисляется характеристическая функция, определяющая для каждой ячейки куба объемную долю твердого тела, содержащегося в ней:

$$f_i = \frac{V(c_i \cap G)}{V(c_i)} \tag{4}$$

где c_i - i-*ая* ячейка, G - область твердого тела, V - объем. Следовательно, $0 \le f_i \le 1$, и $f_i = 1$ только для ячеек, полностью находящихся в области твердого тела, а $f_i = 0$ - для ячеек внутри расчетной жидкости. Непосредственно в расчетном цикле исходная информация о геометрии уже не используется. Она восстанавливается для ячеек с $0 < f_i < 1$ по локальным значениям характеристической функции в текущей и соседних ячейках. В результате процедуры восстановления исходная поверхность твердого тела, пересекающего ячейку, приближается проходящей через ячейку плоскостью. Восстановление поверхности выполняется одним из ранее разработанных методов - SLIC, CM (центральных масс), CD (центральных разностей), LVIRA/ELVIRA [4].

Таким образом, выполняется сквозной однородный расчет методом LU-SGS по всем ячейкам куба с последующей коррекцией значений вектора состояния в тех ячейках, где $0 < f_i < 1$, методом внутренней границы [8]. Данный подход эффективно реализуем для работы на массивно-параллельных архитектурах при условии корректного счета предлагаемой параллельной версией метода LU-SGS.

3. Параллельная версия метода LU-SGS

Если рассматривать алгоритм LU-SGS как объект для распараллеливания, то его последовательный вариант выглядит как два прохода (прямой и обратный) по ячейкам расчётной области. Оба прохода осуществляются по одной последовательности ячеек, но в разных направлениях. В общем случае значения искомых функций в ячейке на следующем временном слое зависят от всех ячеек расчётной области на текущем временном слое, что затрудняет построение параллельного алгоритма.

При обращении к геометрически соседней ячейке вычислительные операции в текущей зависят от того, где располагается в выполняемом обходе эта соседняя ячейка. Если она находится до текущей (т.е. уже была обсчитана), то выполняется один блок вычислений, если после текущей (т.е. еще не была обсчитана), то выполняется другой блок вычислений, и при этом происходит обновление данных в соседней ячейке. При построении параллельного алгоритма необходимо сохранить эту линейную упорядоченность ячеек, т.е. гарантировать, чтобы каждая соседняя ячейка была всегда уже обсчитанной или еще не обсчитанной при обращении к ней.

Особенностью рассматриваемого метода является возможность произвольного выбора порядка обхода ячеек – не обязательно, чтобы две последовательные ячейки в обходе были бы геометрически соседними. Этим мы и воспользуемся для построения специального обхода. Будем рассматривать трехмерные расчетные области с структурированными сетками, где каждая расчетная ячейка может иметь 6 ячеек-соседей – по числу своих граней.

Расчетная область разбивается на блоки с примерно равным числом ячеек, каждый из которых будет считаться на отдельном GPU. Их взаимное расположение аналогично структуре ячеек в сеточном разбиении – «стык в стык». Затем выполняем раскраску блоков двумя цветами так, чтобы два блока с общей гранью были разных цветов. В итоге получаем трехмерное «шахматное» разбиение блоков на два типа – «черные» и «белые».

В каждом блоке выделим две части ячеек – граничные, прилегающие к граням блока и остальные - внутренние. Построим глобальный обход ячеек в блоках следующим образом. Сначала обойдем все внутренние в каждом из «черных» блоков, затем половину внутренних и граничные в «белых», потом граничные в «черных» и, наконец оставшуюся вторую половину в «белых» (схема обхода показана на Рис. 1). Данный обход позволяет вести параллельный счет в блоках идентично соответствующему последовательному прототипу. При этом передача ghost-ячеек между соседними блоками совмещается со счетом. Доказательство корректности такой параллельной схемы представлено в [9].

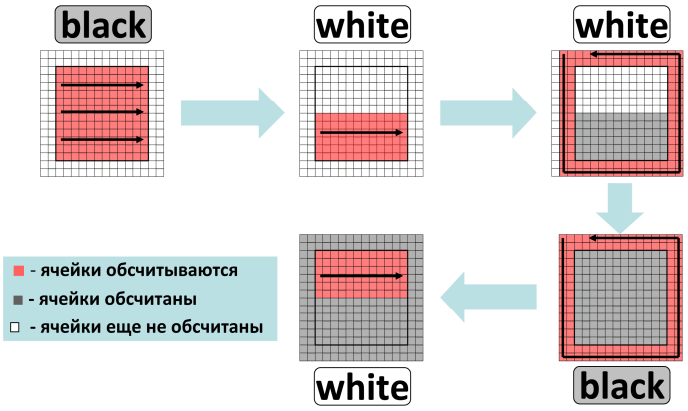

Рис. 1. Порядок обхода ячеек в «черных» и «белых» блоках

Представленный выше алгоритм описывает корректный счет на уровне разделения блоков по нескольким GPU, дополнительно необходимо сохранить корректность при счете и внутри каждого GPU. Средств глобальной синхронизации, с помощью которых это можно было бы достичь, в современных графических ускорителях нет. Однако есть способ обойтись без нее – разбить граничные и внутренние части блоков на несколько подмножеств, внутри каждого из которых ячейки считаются независимо.

Разбиение ячеек на подмножества основано на их раскраске разными цветами так, чтобы соседние по общей грани ячейки всегда были разных цветов. В случае трехмерных структурированных сеток в такой раскраске достаточно использовать только два цвета, при этом получается т.н. red-black разбиение [10], образующее, как и в случае разбиения блоков, «объемную шахматную» доску.

Будем выполнять последовательный обход сначала всех ячеек из black, затем – всех оставшихся из red. Тогда эквивалентный параллельный счет будет состоять из двух этапов – сначала одновременный счет всех ячеек из black и после – из red. Нетрудно видеть, что в этом случае сохраняется линейная упорядоченность между всеми соседними ячейками, поэтому результат работы построенного параллельного алгоритма будет совпадать с результатом работы последовательного.

Таким образом, корректный параллельный алгоритм метода LU-SGS для систем с многими графическими ускорителями полностью построен.

4. Реализация и результаты

Программный код для расчета трехмерных задач реализуется с помощью библиотеки MPI и CUDA ToolKit. Из-за вычислительной сложности ядра его полная сборка возможна только с версиями 4.2 и 5.0 компилятора nvcc, при использовании более старых версий компиляция завершалась с ошибкой на этапе конечной оптимизации ассемблерного кода. Red и black ячейки хранятся в отдельных массивах для более эффективного обращения к ним. Для совмещения счета и передачи данных ячеек во времени использовались неблокирующие функции приема/передачи MPI и несколько CUDA-«потоков» (Stream). В одном из них выполнялось вычислительное ядро (kernel), в других параллельно шло копирование данных. Для синхронизации вычислений использовались функции MPI_Wait и функции работы с событиями (event) в API CUDA. Все вычисления проводились на GPU, ядро центрального процессора использовалось лишь для копирования данных между оперативной памятью и памятью ускорителя, а также для обмена данными между процессами посредством MPI.

Чтобы не ограничиваться только памятью GPU, реализуется следующая схема счета. Блок ячеек делится на части, по объему занимающие 1/3 памяти одного GPU. Ячейки последовательно обрабатываются на GPU: предыдущая (обсчитанная) часть ячеек копируется из видеопамяти в память CPU, текущая часть используется вычислительным ядром на GPU, и из памяти CPU в видеопамять копируется следующая часть ячеек. Эти три операции выполняются одновременно. Таким образом, для счета задач на GPU возможно использовать всю доступную оперативную память, не ограничиваясь лишь малым объемом видеопамяти.

Представленная на конференции ПАВТ-2012 реализация двумерного кода [11] показала эффективность предлагаемых алгоритмов и подходов при расчете задач с использованием большого числа GPU (Рис. 2). Реализация трехмерного кода ведется в настоящее время. Результаты multi-GPU расчетов будут доложены на конференции. Полученные предварительные результаты показывают, что время счета типовой задачи газодинамики (трехмерная фокусировка ударной волны) с разрешением 100x100x100 оказалось на одном GPU (Nvidia Tesla C2070) в 30 раз меньше по сравнению с одним ядром CPU (Intel Xeon X5670).

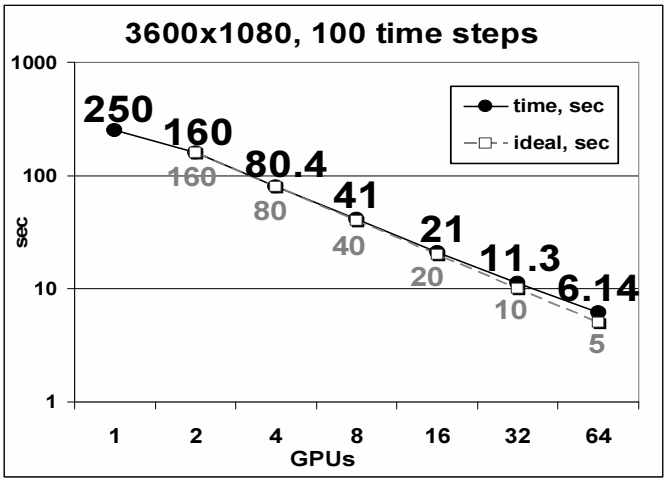

Рис. 2. Время счета двумерной задачи с расчетной сеткой 3600x1080 с разным числом GPU

5. Заключение

В работе представлен параллельный алгоритм, реализующий метод LU-SGS для трехмерных задач газовой динамики со сложной геометрией, описываемой VOF-методом, на кластерной вычислительной системе распределенных графических ускорителей. Показана корректность этого алгоритма, описана его реализация с помощью технологий MPI и CUDA, позволяющая задействовать всю доступную оперативную память, не ограничиваясь объемом видеопа-

мяти. Двумерная версия кода показала масштабируемость, близкую к линейной. Предварительные результаты для трехмерной версии дают 30-кратный рост производительности относительно одного процессорного ядра.

Литература

1. И. В. Семенов, И. Ф. Ахмедьянов. Разработка параллельного алгоритма LU-SGS для решения многомерных задач вычислительной газодинамики // Материалы Четвертой Сибирской школы-семинара по параллельным и высокопроизводительным вычислениям. Изд-во Томск: Дельтаплан, 2008. -- С. 122-129, 2008.
2. Hirt, C.W.; Nichols, B.D. (1981), "Volume of fluid (VOF) method for the dynamics of free boundaries", Journal of Computational Physics 39 (1): 201–225
3. W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (2) (1998) 112–152.
4. J.E. Pilliod Jr., E.G. Puckett, Second-order accurate volume-of-fluid algorithms for tracking material interfaces // Journal of Computational Physics 199 (2004) 465–502
5. A. Jameson, E. Turkel Implicit schemes and LU decomposition // Math.of Comp., v.37, № 156, pp.385-397, 1981.
6. I. Menshov, Y. Nakamura On implicit Godunov's method with exactly linearized numerical flux //Computers & Fluids, 29 (6), pp. 595 – 616, 2000.
7. I. Menshov, Y. Nakamura Hybrid Explicit-Implicit, Unconditionally Stable Scheme for Unsteady Compressible Flows //AIAA Journal,vol. 42, № 3, pp. 551-559, 2004.
8. И. С. Меньшов. Метод свободной границы для расчета задач динамики газа и твердого тела. Материалы XXII Научно-технической конференции по аэродинамики ЦАГИ, 3-4.03.2011, п. Володарского МО, с.108-109, 2011.
9. П.В. Павлухин, И.С. Меньшов. Эффективная реализация метода LU-SGS для задач газовой динамики. //Научный вестник МГТУ ГА, Т.165, 3, с.46 - 55, 2011
10. T. Iwashita, M. Shimasaki, Block Red-Black Ordering Method for Parallel Processing of ICCG Solver // 4th International Symposium, ISHPC 2002 Kansai Science City, Japan, May 15–17, 2002 Proceedings, pp 175-189, 2006
11. П. В. Павлухин. Реализация параллельного метода LU-SGS для задач газовой динамики на кластерных системах с графическими ускорителями //Труды международной научной конференции Параллельные вычислительные технологии (ПаВТ’2012, Новосибирск, 26 – 30 марта 2012 г.), с. 627-634, 2012