

ВНУТРИУЗЛОВАЯ ДИНАМИЧЕСКАЯ И МЕЖУЗЛОВАЯ ЧАСТИЧНАЯ АДАПТИВНАЯ БАЛАНСИРОВКА ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ПРИ РЕШЕНИИ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ НА СУПЕРКОМПЬЮТЕРАХ С НЕОДНОРОДНЫМ ДОСТУПОМ К ПАМЯТИ

Г.Б. Сушко, С.А. Харченко

Введение. Моделирование задач вычислительной аэро- и гидродинамики является актуальной проблемой многих отраслей промышленности. Параллельная версия программного комплекса “FlowVision” [1-2], разрабатываемого в ООО “ТЕСИС”, используется для моделирования широкого круга промышленных задач, включая задачи с подвижными телами и свободными поверхностями для областей со сложной геометрией. В программном комплексе “FlowVision” используются неявные схемы аппроксимации, что приводит к необходимости решать соответствующие системы линейных уравнений. Для адекватного воспроизведения тонких физических эффектов в геометрически сложных трехмерных областях требуются подробные расчетные сетки, содержащие от сотен тысяч до десятков и даже сотен миллионов расчетных ячеек. Моделирование подобных задач требует огромных вычислительных ресурсов и может быть проведено только на самой современной мощной параллельной вычислительной технике.

Современные суперкомпьютерные вычислительные системы как правило имеют неоднородную архитектуру. С одной стороны, имеется набор вычислительных узлов с распределенной памятью, обмен данными между которыми может быть осуществлен по быстрой обменной сетке. С другой стороны, каждый узел представляет собой многопроцессорный/многоядерный компьютер с общим доступом к оперативной памяти. В предыдущих работах [3-7] авторами были предложены алгоритмы распараллеливания вычислений при решении систем уравнений для случая параллельных вычислений с распределенной [3-5] на основе MPI и с общей [6] памятью на основе OpenMP/TBB [8-9]. В работе [7] предложена комбинированная MPI+TBB реализация алгоритма решения систем линейных уравнений. Алгоритмы, описанные в этих работах, основаны на единых подходах к распараллеливанию вычислений, базируются на рекурсивной геометрической декомпозиции расчетной сетки на основе упорядочиваний типа вложенных сечений Nested Dissection (ND) [10], используют специальные подходы к распараллеливанию вычислений с поверхностными межпроцессорными границами [5], и при этом учитывают адаптацию расчетной сетки [4].

Эффективная хорошо масштабируемая организация параллельных вычислений при большом общем числе доступных процессоров/ядер предполагает хорошую балансировку вычислений на различных процессорах между собой. Чем больше число задействованных процессоров/ядер, тем более точная балансировка вычислений требуется для поддержания заданного уровня масштабируемости. С другой стороны, моделирование задач с неструктурированными локально адаптированными расчетными сетками приводит к заранее непредсказуемым различиям в объеме вычислений даже при условии обработки одинакового количества ячеек. Все это приводит к необходимости использовать специальные методики, позволяющие балансировать вычислительную нагрузку для достижения максимальной эффективности параллельных вычислений.

В работе описывается параллельная реализация алгоритма решения СЛАУ с предобуславливателем ICN2/ILU2 и итерационной схемой типа подпространства Крылова для компьютеров с неоднородным доступом к памяти. Для уменьшения времени вычислений этот алгоритм распараллеливается по распределенной памяти (по узлам вычислений) с использованием стандарта MPI, а по общей памяти узла распараллеливание по процессорам/ядрам осуществляется с динамическим распределением нагрузки на основе технологии Intel® Threading Building Blocks (TBB) [9]. Для улучшения межузловой балансировки нагрузки на каждом шаге по физическому времени собирается статистика о CPU временах счета для каждого домена расчетной области и при взаимодействии данных соседних доменов между собой. На следующем шаге по физическому времени на основе этой информации производится частичное по возможности максимально близкое к исходному перераспределение расчетных доменов по узлам суперкомпьютера для оптимизации загрузки. По существу, внутриузловая динамическая балансировка внутри узла осуществляется по принципу клиент – сервер (client-server), а межузловая частичная адаптивная балансировка осуществляется по принципу децентрализованной сети (peer-to-peer). В работе приводятся некоторые результаты численных экспериментов по масштабируемости предложенных алгоритмов для систем линейных уравнений, возникающих при моделировании во “FlowVision” задач вычислительной гидродинамики.

Внутриузловая динамическая балансировка. Внутриузловая динамическая балансировка осуществляется во “FlowVision” на основе комбинированной MPI+TBB технологии организации параллельных вычислений. Подробности детально изложены в работе [7]. В данной работе упомянем только некоторые ключевые моменты этого подхода.

Основная идея внутриузловой динамической балансировки состоит в том (Рис. 1), чтобы на каждом этапе вычислений по мере выполнения предыдущих вычислений каждое ядро внутри текущего узла

динамически осуществляло выбор следующего возможного вычисления и производило его. В каждом узле с общей памятью при распараллеливании по процессорам/ядрам для организации синхронизаций по общей памяти используется технология Intel® Threading Building Blocks (TBB). При распараллеливании вычислений между узлами для организации обменов и синхронизаций используется стандарт обмена сообщениями MPI. При этом предполагается, что на каждом узле имеется только один MPI процесс, который затем порождает на этом узле нужное количество одновременно работающих потоков вычислений.

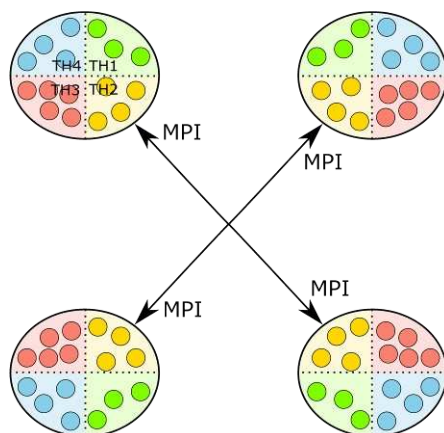


Рис. 1. Комбинированная MPI+threads организация параллельных вычислений.

Для явного выделения параллелизма в алгоритмах решения систем уравнений используется специальное упорядочивание рекурсивного типа, аналогичное упорядочиванию вложенных сечений ND. На основе этого упорядочивания строятся блочное биение системы уравнений и выбирается распределение данных по узлам суперкомпьютера. При распараллеливании вычислений с поверхностными межпроцессорными границами используется специальная методика распараллеливания этих вычислений [5]. При этом структура взаимодействия всех узлов и процессоров/ядер между собой при решении СЛАУ полностью описывается матрицей зависимостей блочных вычислений [7]. На Рисунке 2 для некоторой тестовой задачи приведен пример структуры разреженности и блочного биения матрицы СЛАУ, а также матрица зависимостей блочных вычислений для 64 узлов*процессоров*ядер.

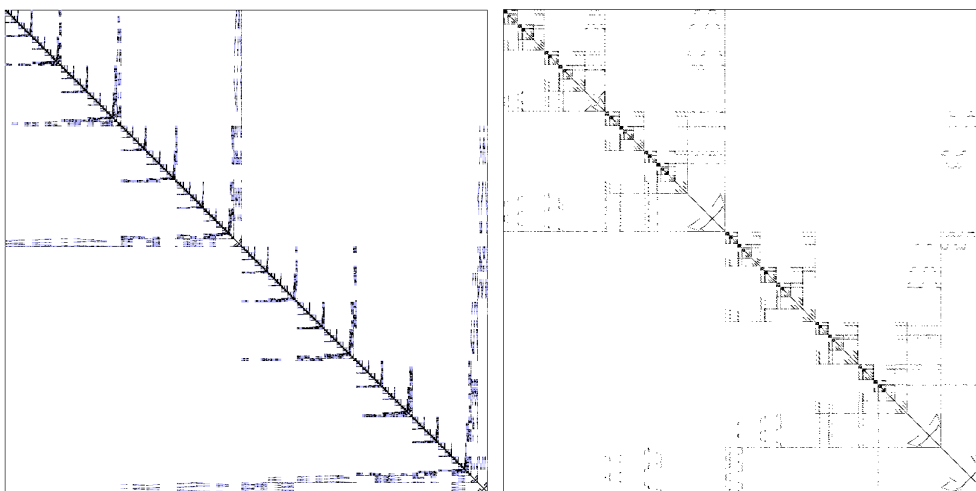


Рис. 2. Структура разреженности и блочное биение (слева) и матрица зависимостей блочных вычислений (справа) для тестовой задачи при распараллеливании на 64 узлов*процессоров*ядер.

Параллельные вычисления на компьютерах с неоднородным доступом к памяти при решении СЛАУ организованы следующим образом. Пусть имеется s узлов и t процессоров/ядер в каждом узле, итого всего $p = s * t$ процессоров/ядер. Производится параллельная декомпозиция задачи (в том числе построение блочного биения матрицы системы уравнений и матрицы зависимостей блочных вычислений) на число доменов, равное $p * q$, где $q > 1$ - целое число, обозначающее запас параллелизма для автоматической балансировки вычислений внутри узла. В данной работе использовалось значение $q = 4$. Далее, на каждый узел распределяются все данные, соответствующие $t * q$ доменам этого узла. Все параллельные вычисления проводятся в соответствии с блочными структурами соответствующих матриц на основе матрицы зависимостей блочных вычислений.

Межузловая частичная адаптивная балансировка. Современные суперкомпьютеры обладают огромным числом вычислительных узлов, соединенных друг с другом быстрой обменной сеткой. С этой точки зрения при использовании сверхбольшого числа ядер только внутриузловой динамической балансировки вычислений недостаточно для достижения хорошей масштабируемости параллельных вычислений. Необходимо каким-то образом балансировать между собой также и вычисления на большом числе узлов.

Эффективная балансировка параллельных вычислений предполагает перенос хотя бы части вычислений с одних вычислительных устройств на другие. При внутриузловой динамической балансировке это перенос вычислений с одних ядер на другие, при этом за счет использования общей памяти в узле собственно явное перебрасывание данных не требуется, необходимо лишь правильно синхронизовать данные в общей памяти узла. При межузловой балансировке необходимо перебросить также и данные с одного вычислительного узла на другой, поскольку оперативная память в этом случае – распределенная. С этой точки зрения естественным образом возникает вопрос о минимизации объемов перебрасываемых данных при осуществлении балансировки. С другой стороны, необходимо найти некоторую базовую информацию, на основе которой будет осуществляться балансировка вычислений. В случае общей памяти все просто – как только ядро освободилось, его можно использовать для следующего вычисления по принципу клиент – сервер, при этом клиентами выступают все ядра узла, а сервером – некоторый объект в общей памяти, изменяемый всеми ядрами через семафоры доступа. В случае распределенных вычислений подобная организация вычислений также возможна, но она будет иметь недопустимые в нашем случае накладные расходы.

В случае межузловой балансировки по распределенной памяти предлагается использовать предысторию параллельных вычислений для оптимизации последующих. Действительно, например при моделировании во времени динамики движения жидкости и газа проводятся вычисления на множестве последовательных шагов по физическому времени. Для осуществления балансировки в этом случае будем использовать предположение, что проводимые вычисления не сильно меняются от одного шага по физическому времени к другому. В этом случае можно провести замер CPU времен выполнения каждого этапа вычислений для каждого домена, а также CPU времен вычислений соответствующих взаимодействии соседних доменов между собой. Предлагается измерять именно CPU время, поскольку именно это время наименьшим образом подвержено задержкам из-за способа организации параллельных вычислений, и при этом показывает именно вычислительные затраты каждого ядра при обработке данных домена. На основе информации о геометрических связях доменов между собой можно составить матрицу, описывающую для каждого домена все его соседние домены. С другой стороны, просуммировав все замеренные CPU времена для каждого домена и CPU времена для связей домена с каждым соседом можно составить весовую матрицу, описывающую полные CPU времена затраченные на вычисления с самими доменами и на вычисления при взаимодействии соседних доменов. На основе этой информации можно на текущем шаге по физическому времени адаптивно перераспределить домены по узлам суперкомпьютера, начиная с некоторого исходного распределения.

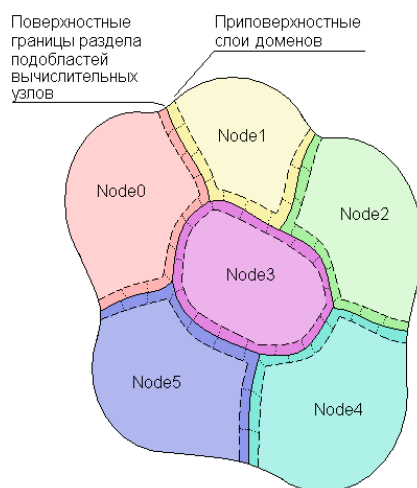


Рис. 3. Домены из областей, близких к доменным поверхностным границам узлов.

Для минимизации затрат на перенос данных предлагается использовать идею, аналогичную идеям из работы [11]. А именно, для большинства доменов фиксируется предыдущее распределение доменов по узлам суперкомпьютера. Перенести расчеты с некоторыми доменами на другие узлы разрешается только для доменов из областей, близких к доменным поверхностным границам узлов, см. Рисунок 3. Вычислять подобное перераспределение можно рекурсивно параллельным алгоритмом, аналогичным описанному в работе [11]. При этом, с одной стороны, по построению ограничены объемы данных, которые возможно придется перебросить с одних узлов на другие по межузловой обменной сетке. А с другой стороны, за счет предыстории вычислений делается попытка максимально точно предсказать CPU времена вычислений для каждого расчетного домена на

текущем шаге по физическому времени. На основе этой информации предполагается управляемым образом адаптивно частично улучшить межузловую балансировку параллельных вычислений на текущем шаге по физическому времени.

Результаты численных экспериментов. Для проверки эффективности алгоритмов балансировки в данной работе рассматривалась двумерная задача о сверхзвуковом обтекании крылового профиля с числом Маха $M=2$. Геометрия крыла, адаптированная расчетная сетка и декомпозиция задачи на 16 доменов показаны на Рисунке 4. Расчетная сетка сильно адаптируется к граничной линии контура крыла, число уровней адаптации было равно 5. Декомпозиция, приведенная на рисунке 4, производилась из условия близости числа расчетных ячеек во всех доменах при минимальной междоменной границе. Расчетная сетка содержала $N=964.364$ ячеек. Структура полученного решения в терминах модуля скорости показана на Рисунке 5.

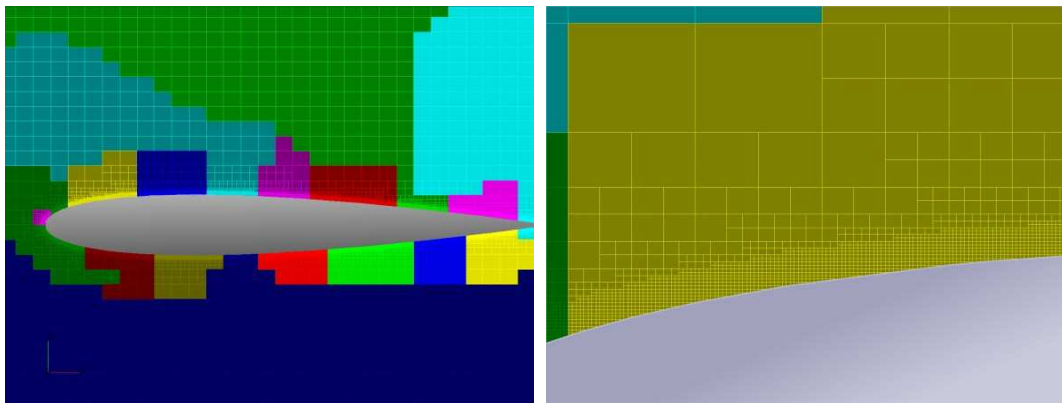


Рис. 4. Геометрия тестовой задачи, декомпозиция на 16 доменов (слева) и адаптация сетки вблизи граничной линии профиля крыла (справа).

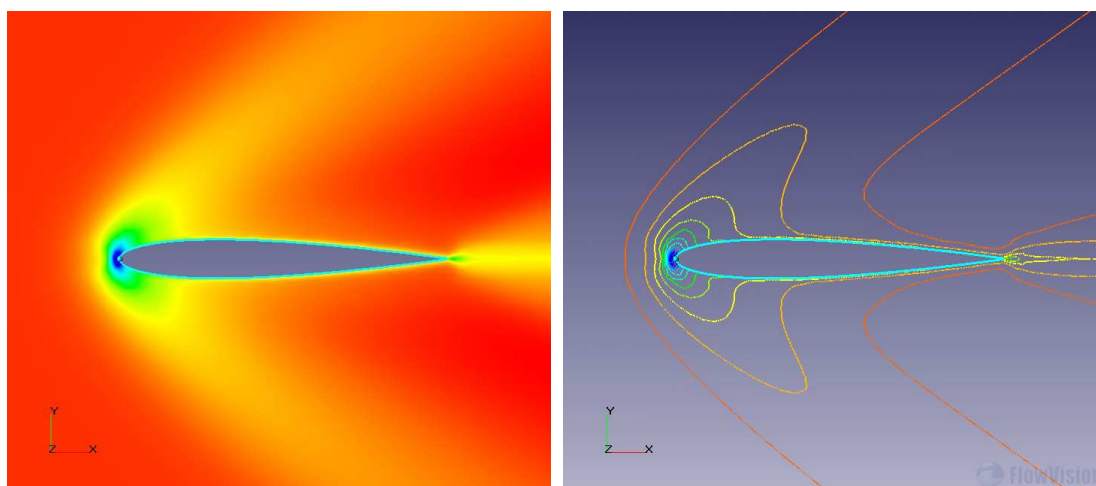


Рис. 5. Структура решения в терминах модуля скорости: заливка (слева) и изолинии (справа).

Численные эксперименты проводились на кластере, составленном из 8 узлов. Каждый узел содержал по 2 четырехядерных процессора Intel(R) Xeon(R) CPU E5430, 2.66GHz, каждый узел содержал 16 Gb DDR2 памяти на узел. Узлы связаны между собой обменной сетью Mellanox Technologies MT25208 InfiniHost III Ex.

Результаты экспериментов по масштабируемости представлены на Рисунке 6.

Для тестовой задачи с сильно локально адаптированной расчетной сеткой эксперименты показывают значительное положительное влияние внутриузловой динамической балансировки на масштабируемость вычислений.

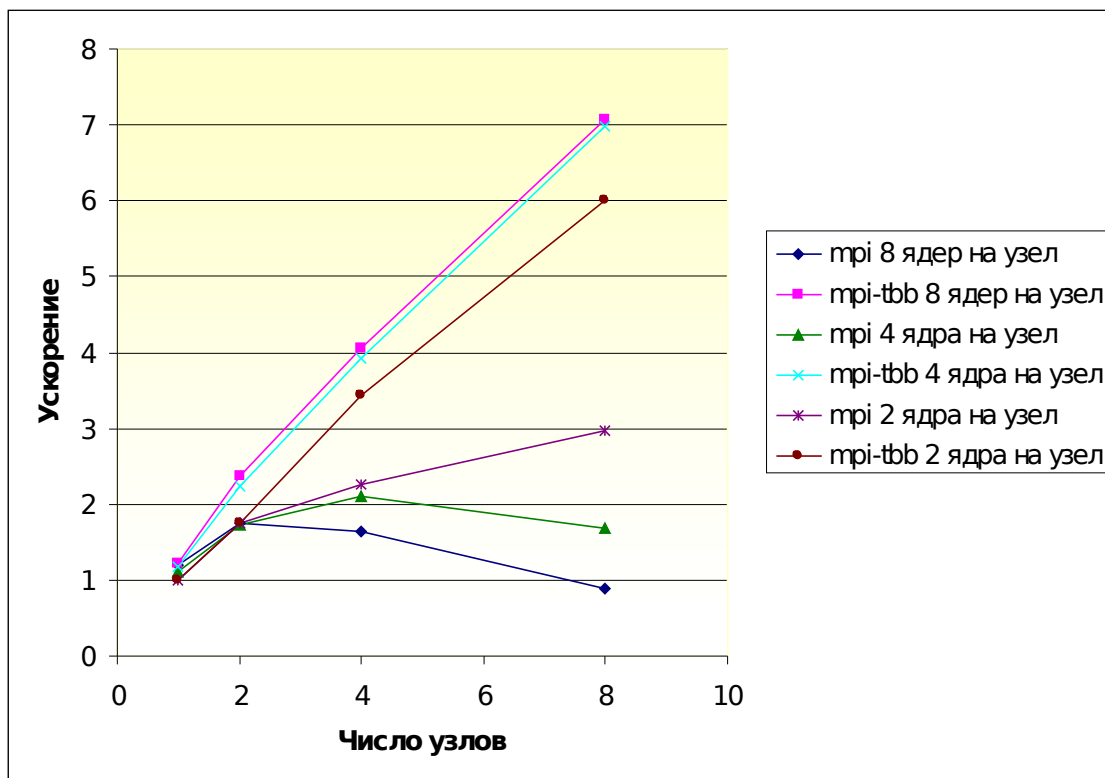


Рис. 6. Масштабируемость при решении тестовой задачи в терминах числа используемых узлов.

ЛИТЕРАТУРА:

1. Aksenov A., Dyadkin A., Pokhilko V., "Overcoming of Barrier between CAD and CFD by Modified Finite Volume Method", Proc 1998 ASME Pressure Vessels and Piping Division Conference, San Diego, ASME PVP-Vol 377-1, 1998.
2. Aksenov A.A., Kharchenko S.A., Konshin V.N., Pokhilko V.I., "FlowVision software: numerical simulation of industrial CFD applications on parallel computer systems", Parallel CFD 2003 conference, Book of abstracts, p. 280-284, 2003.
3. 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.
4. Дядькин А.А., Харченко С.А. "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.
5. Харченко С.А. "Влияние распараллеливания вычислений с поверхностными межпроцессорными границами на масштабируемость параллельного итерационного алгоритма решения систем линейных уравнений на примере уравнений вычислительной гидродинамики" // Труды международной научной конференции Параллельные вычислительные технологии (ПаВТ'2008), Санкт-Петербург, 28 января – 1 февраля 2008 г. Челябинск, Изд. ЮУрГУ, 2008, с.494-499.
6. Сушко Г.Б., Харченко С.А. "Многопоточная параллельная реализация итерационного алгоритма решения систем линейных уравнений с динамическим распределением нагрузки по нитям вычислений" // Труды международной научной конференции Параллельные вычислительные технологии (ПаВТ'2008), Санкт-Петербург, 28 января – 1 февраля 2008 г. Челябинск, Изд. ЮУрГУ, 2008, с.452-457.
7. Сушко Г.Б., Харченко С.А. "Экспериментальное исследование на СКИФ МГУ "Чебышев" комбинированной MPI+threads реализации алгоритма решения систем линейных уравнений, возникающих во FlowVision при моделировании задач вычислительной гидродинамики" // Труды международной научной конференции Параллельные вычислительные технологии (ПаВТ'2009), Нижний Новгород, 30 марта – 3 апреля 2009 г. Челябинск, Изд. ЮУрГУ, 2009, с.316-324.
8. OpenMP Application Program Interface - 2.5 / OpenMP Architecture Review Board – 2005.
9. Intel Threading Building Blocks Tutorial – 1.6 / Intel Corp. 2007.
10. George A., Liu J.W., "Computer Solution of Large Sparse Positive Definite Systems" // Prentice Hall, 1981.

11. Аксенов А.А., Дядькин А.А., Харченко С.А. “Исследование эффективности распараллеливания расчета движения подвижных тел и свободных поверхностей во FlowVision на компьютерах с распределенной памятью”. Вычислительные методы и программирование, т. 10, 2009 г., с. 132-140.