S. A. Sukov, Methods for improving and evaluating the performance of unstructured CFD-algorithms,
Mat. Model., 2023, Volume 35, Number 2, 30–42

https://www.mathnet.ru/eng/mm4440

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ 2023 год, том 35, номер 2, стр. 30-42

МЕТОДЫ ПОВЫШЕНИЯ И ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ АЛГОРИТМОВ МОДЕЛИРОВАНИЯ ГАЗОДИНАМИЧЕСКИХ ТЕЧЕНИЙ НА НЕСТРУКТУРИРОВАННЫХ СЕТКАХ

© *2023г.* ***С.А. Суков***

Институт прикладной математики им. М.В. Келдыша РАН
ssoukov@gmail.com

DOI: 10.20948/mm-2023-02-03

Обсуждается проблема повышения производительности неструктурированных алгоритмов моделирования газодинамических течений на многоядерных процессорах. Рассматриваются детали применения алгоритмов уменьшения ширины ленты матрицы смежности и реализации схемы вычислений с асинхронным интегрированием по времени. Показана возможность использования симулятора множественно-ассоциативного кэш для сравнительной оценки методов упорядочивания данных. Описана процедура тестирования быстродействия конечно-объемного алгоритма моделирования уравнений Навье-Стокса на примере расчета задачи сверхзвукового обтекания сферы на трех смешанных сетках, содержащих от 270 тыс. до 17 млн ячеек.

Ключевые слова: вычислительная газовая динамика, неструктурированная смешанная сетка, оптимизация доступа к данным.

METHODS FOR IMPROVING AND EVALUATING THE PERFORMANCE OF UNSTRUCTURED CFD-ALGORITHMS

S.A. Soukov

Keldysh Institute of Applied Mathematics of RAS

The problem of improving the performance of unstructured CFD-algorithms on multicore processors is discussed. The details of the application of algorithms for reducing the adjacency matrix bandwidth and the implementation of the calculation scheme with asynchronous time integration are considered. The possibility of using a set-associative cache simulator for a comparative evaluation of data ordering methods is shown. A procedure for testing the speed of a finite-volume algorithm for modeling the Navier-Stokes equations is described using the example of calculating the problem of supersonic flow around a sphere on three mixed meshes containing from 270 thousand to 17 million cells.

Key words: computational fluid dynamics, unstructured mixed mesh, data access optimization.

1. Введение

Два последних десятилетия в области компьютерного моделирования ведутся исследования в направлении разработки и программной реализации численных алгоритмов расчета течений жидкости и газа на неструктурированных сетках [1-3]. Для дискретизации многосвязных областей с криволинейными границами широко используются конформные смешанные сетки, а также построенные на их основе многоблочные перекрывающиеся сетки и адаптивные сетки с висячими узлами. Подробные дискретные модели областей для моделирования нестационарных течений у поверхностей сложной формы содержат десятки миллионов ячеек. Поэтому расчеты проводятся на высокопроизводительных системах.

По соотношению стоимости, энергопотребления, производительности, удобства программирования и переносимости кода на данный момент лидируют гетерогенные кластеры. Их узлы состоят из многоядерных процессоров общего назначения (CPU) и массивно-параллельных ускорителей (GPU). Теоретическая пиковая производительность вычислителей соотносится примерно как 1:5. Но для рассматриваемого класса задач отношение реальной производительности вычислений к пиковой производительности устройства у центральных процессоров оказывается примерно в два раза выше [4]. То есть доля производительности CPU-части систем с симметричными узлами составляет до 30% от вычислительной мощности кластера, и использование процессоров исключительно как платформы для управления запуском GPU-ядер и обменов MPI-сообщениями неэффективно. Следовательно, задача повышения скорости вычислений на CPU не теряет практического значения.

На уровне прикладного программиста адаптация кода к конкретной модели процессора часто ограничивается изменением размера группы OpenMP-нитей, хотя устройства разных моделей и поколений имеют более серьезные отличия. Это характеристики оперативной памяти, число уровней, тип и размер кэш, алгоритмы замещения данных и их спекулятивной загрузки из оперативной памяти. Поэтому длительное сопровождение программ должно включать периодическую оценку производительности, оптимизацию, а в крайних случаях пересмотр алгоритмов в соответствии с новой архитектурой вычислителя.

В данной работе на примере явного конечно-объемного неструктурированного алгоритма моделирования уравнений Навье-Стокса рассматриваются существующие методы повышения производительности вычислений на неструктурированных сетках и возможные способы оценки производительности программного обеспечения.

2. Повышение производительности неструктурированных алгоритмов

Программные реализации неструктурированных CFD-алгоритмов характеризуются низкой плотностью вычислений, то есть небольшим числом арифметических операций на единицу данных. Поэтому основным способом повышения быстродействия программ становится поиск оптимальных методов представления, упорядочивания и доступа к данным в оперативной памяти. В настоящей статье эффективность возможных подходов к организации вычислений обсуждается на примере реализации конечно-объемного алгоритма моделирования систем уравнений Эйлера и Навье-Стокса с определением значений сеточных функций в центрах масс ячеек смешанной сетки, линейной реконструкцией переменных и явным интегрированием по времени [4].

Код газодинамического ядра состоит из трех функциональных блоков: вычисление коэффициентов линейной реконструкции поведения сеточных функций внутри ячеек, определение потоков функции распределения через грани контрольных объемов и обновление значений газодинамических переменных на новом слое по времени. Значения переменных $\mathbf{Q}_i = (\rho_i, \rho u_i, \rho v_i, \rho w_i, E_i)$, координаты центров ячеек $\mathbf{x}_i = (x_i, y_i, z_i)$, их объемы V_i и полиномиальные коэффициенты $\nabla\mathbf{Q}_i = (\nabla\rho_i, \nabla\rho u_i, \nabla\rho v_i, \nabla\rho w_i, \nabla E_i)$ хранятся в виде линейных массивов вещественных чисел двойной точности. Внутри массивов данные группируются по элементам сетки. Коэффициенты линейной реконструкции вычисляются по формуле Грина-Гаусса в цикле по контрольным объемам. Списки индексов соседних элементов и весовые коэффициенты при значениях функций хранятся с использованием аналога CSR-формата представления разреженных матриц. Потоки функции распределения $\mathbf{F}_n$ через общую грань $\mathbf{cf}_n.cL$-го и $\mathbf{cf}_n.cR$-го элементов вычисляются один раз в цикле по граням и буферизуются в локальный линейный массив. Помимо индексов ячеек структура с параметрами грани $\mathbf{cf}_n$ содержит координаты ее центра $\mathbf{cf}_n.\mathbf{x}$, площадь $\mathbf{cf}_n.S$ и компоненты векторов локальной системы координат $\mathbf{cf}_n.\mathbf{lcs}$. Суммирование потоков и обновление значений газодинамических переменных на новом слое по времени выполняется в цикле по сеточным ячейкам. Списки модифицированных индексов граней, принадлежащих ячейкам, хранятся в массиве **idxFace**. Указатели на первые элементы списков считываются из массива **idxOffset**. Знак (направление) потока во время суммирования соответствует знаку модифицированного индекса ($idxFace_l = n+1 > 0$ для ячейки $\mathbf{cf}_n.cL$ и $idxFace_r = -n-1 < 0$ для ячейки $\mathbf{cf}_n.cR$).

В табл.1 приводятся фрагменты кода циклов по вычислению конвективных потоков с повышенным порядком точности и обновлению значений переменных на новом шаге по времени. В качестве функции *EulerSolver* подставляется реализация одной из двух схем [5, 6] решения задачи Римана о распаде произвольного разрыва. В случае вычисления потоков с первым порядком точности по пространству тело цикла сокращается до простого вызова функции от осредненных значений переменных $EulerSolver(\mathbf{Q}_{cf_i.cL}, \mathbf{Q}_{cf_i.cL}, \mathbf{cf_i.lcs}, \mathbf{FF})$ с последующей буферизацией. Витки циклов не имеют зависимости по данным. Для их распараллеливания между процессорными ядрами ставится OpenMP-директива «`#pragma omp parallel for`».

Таблица 1.

Вычисление потоков

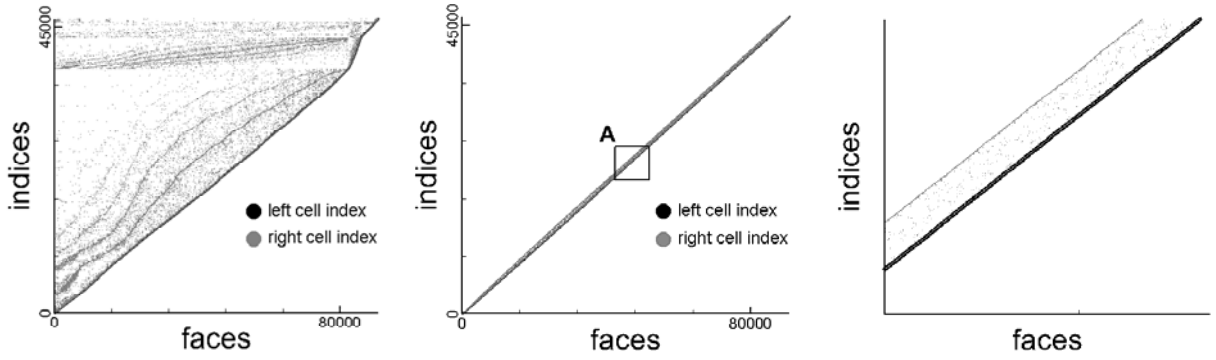
```
#pragma omp parallel for
for(i=0; i<Nface; i++) {
double r_L[3], r_R[3], QF_l[5], QF_R, FF[5];
// расстояния до центра грани
r_L = cf_i.x − x_{cf_i·cL}; r_R = cf_i.x − x_{cf_i·cR};
// значения переменных на грани
QF_L = Q_{cf_i·cL} + r_L ∇Q_{cf_i·cL};
QF_R = Q_{cf_i·cR} + r_R ∇Q_{cf_i·cR};
// вычисление потока
EulerSolver(QF_L, QF_R, cf_i.lcs, FF);
// буферизация потока
F_i = FF * cf_i.S;
} // for i
```

Обновление переменных

```
#pragma omp parallel for
for(i=0; i<Ncell; i++) {
double CE[5] = {0.0, 0.0, 0.0, 0.0, 0.0};
int j, n;
for(j=idxOffset[i]; j<idxOffset[i+1]; j++) {
if(idxFace[j] > 0) {int idFace = idxFace[j] - 1;
for(n=0; n<5; n++) CE[n] -= F[idFace*5 + n]; }
else { int idFace = -idxFace[j] - 1;
for(n=0; n<5; n++) CE[n] += F[idFace*5 + n]; }
} // for j
for(n=0; n<5; n++) Q[i*5+n] += CE[n]*DT/V[i];
} // for i
```

Основной метод ускорения доступа к данным в реализациях неструктурированных CFD-алгоритмов состоит в поиске оптимальной индексации сеточных элементов. Новая индексация генерируется с использованием одного из алгоритмов оптимизации матрицы смежности [7, 8]. В результате переупорядочивания значения $\mathbf{Q}_i$ и $\nabla\mathbf{Q}_i$ в соседних по граням ячейках располагаются в оперативной памяти более компактно.

На рис.1 представлены примеры распределения $\mathbf{cf}_n.cL$ и $\mathbf{cf}_n.cR$ индексов ячеек неструктурированной сетки в массиве граней до (рис.1а) и после (рис.1б) оптимизации индексации алгоритмом Катхилла-Макки [7]. Структуры с описанием геометрии граней отсортированы внутри $\mathbf{cf}$ по возрастанию индексов сеточных ячеек (лексикографическое упорядочивание с предварительным условием $\mathbf{cf}_n.cL < \mathbf{cf}_n.cR$). Вне зависимости от конкретного вида индексации при чтении $\mathbf{Q}_{cf_n.cL}$ всегда реализуется последовательный доступ к данным. На иллюстрациях видно, что переупорядочивание ячеек приводит к уменьшению ширины ленты матрицы смежности, а обращения к значениям $\mathbf{Q}_{cf_n.cR}$ ложатся преимущественно на вторую линию загрузки данных по возрастанию индексов контрольных объемов. Однако чтение последовательных блоков данных чередуется с обращениями к произвольным ячейкам памяти (рис.1в). Таким образом, применение методов оптимизации матрицы смежности теоретически повышает эффективность работы алгоритмов спекулятивной загрузки, хранения и замещения данных в сверхоперативной памяти процессора, но не гарантирует полное исключение пропусков кэш. Аналогичные по своей сути выводы будут справедливы и относительно повышения производительности вычислительных циклов по сеточным ячейкам и их соседям.



а) исходная индексация б) преобразованная индексация в) индексы в зоне A

Рис.1. Распределение индексов ячеек в массиве структур с описанием граней.

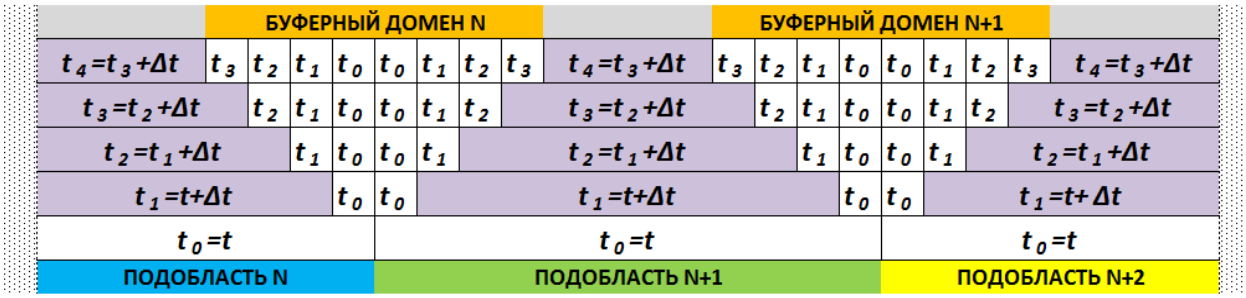
Следует отметить, что коэффициент ускорения вычислений зависит от стартовой индексации ячеек. Оптимизация матрицы смежности явно или

неявно присутствует в алгоритмах генерации сеток. Поэтому переупорядочивание элементов не всегда приводит к заметному повышению производительности последовательных программ, но оказывается эффективным методом для ускорения и стабилизации быстродействия параллельных MPI-приложений. В случае балансировки загрузки методом геометрического параллелизма MPI-процессы обрабатывают локальные подобласти сетки, внутри которых требуется заново проиндексировать ячейки оптимальным образом. Лексикографическая сортировка считается наиболее простым и распространенным подходом к упорядочиванию данных для вычислений в циклах по граням контрольных объемов. В ряде работ [напр., 9] приводится описание более сложных алгоритмов составления массива граней. По результатам проведенных автором исследований эффективность и целесообразность их использования для рассматриваемого класса задач сильно зависит от топологических параметров конкретной сетки, а замена метода сортировки граней приводит к росту времени работы модуля обновления переменных.

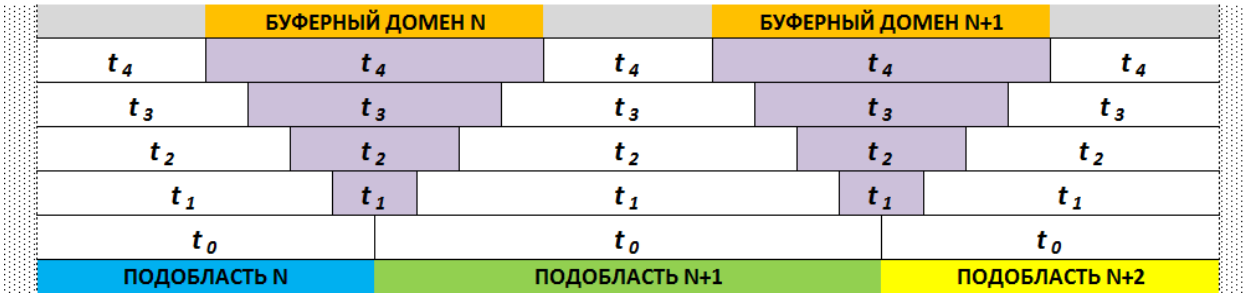
Второй подход к ускорению явных алгоритмов решения эволюционных задач математической физики состоит в реализации схемы вычислений с асинхронным интегрированием по времени [10]. Стандартная синхронная схема предполагает, что значения газодинамических переменных в массиве $\mathbf{Q}$ относятся к некоторому моменту модельного времени t, а переход $\mathbf{Q}_i(t) \Rightarrow \mathbf{Q}_i(t+\Delta t)$ выполняется одновременно во всех сеточных ячейках. При асинхронном интегрировании значения $\mathbf{Q}_i$ могут относиться к разным моментам модельного времени. Снятие ограничений дает возможность перейти к новой последовательности вычислений, которая обеспечивает лучшую локализацию обрабатываемых данных в кэш процессора.

На рис.2 показан пример асинхронной схемы вычислений для метода контрольного объема первого порядка точности в 1D постановке. Расчетная область разбивается на подобласти по пространственной оси. Параметры декомпозиции подбираются таким образом, чтобы данные для расчета локальных задач в пределах сеточных подобластей полностью помещались в кэш. Интегрирование по времени идет блоками из N_{step} шагов. Выполнение блока делится на два этапа. На первом этапе в произвольном порядке запускаются процедуры моделирования локальных задач. Локализация данных исключает доступ к значениям переменных в ячейках соседних подобластей. Поэтому на каждом шаге интегрирования по времени ширина зоны вычислений сокращается на две ячейки. И таким образом вокруг границ разбиения образуются буферные домены, где значения $\mathbf{Q}_i$ относятся к разным

моментам модельного времени (рис.2а). На втором этапе запускаются процедуры расчета локальных задач в буферных доменах. В данном случае зона вычислений расширяется на две ячейки после каждого шага (рис.2б). По завершению второго этапа все значения переменных в массиве **Q** относятся к одному моменту модельного времени $t + \Delta t \cdot N_{step}$.



а) локальное моделирование внутри подобластей



б) локальное моделирование внутри буферных зон

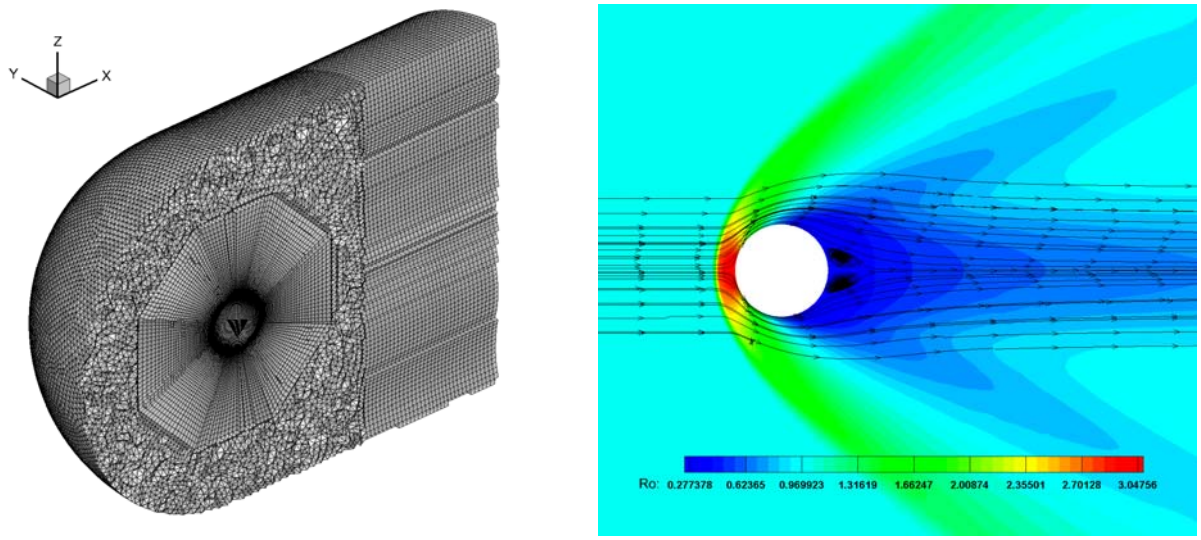
Рис.2. Пример асинхронной схемы вычислений при $N_{step} = 4$.

С чисто алгоритмической точки зрения описанная асинхронная схема интегрирования по времени переносится на рассматриваемый в статье неструктурированный CFD-алгоритм, но быстродействие ее программной реализации вызывает вопросы. Рост производительности должен достигаться в первую очередь за счет повторяющихся вычислений на локализованном наборе данных. При среднем расходе оперативной памяти 750 Кбайт на тысячу ячеек сетки в L1 и L2 кэш процессорного ядра могут быть размещены значения параметров для 250 – 500 контрольных объемов. По статистике задача декомпозиции неструктурированной сетки на компактные подобласти минимального размера в большом числе случаев решается некорректно, что исключает вариант локализации данных в сверхоперативной памяти первого и второго уровня. В разделяемом L3-кэш в зависимости от модели процессора могут храниться значения переменных в 5 – 20 тысячах ячеек. По результатам экспериментов циклический запуск функциональных блоков внутри сеточных подобластей среднего размера не дает выигрыша в производительности. То есть простое ограничение размера задачи не приносит ожидаемого

эффекта, и требуется изменение формата представления данных по аналогии с [11]. Эта задача эквивалентна поиску оптимальной индексации и порядка обхода ячеек и на текущий момент не имеет решения, которое воспроизводит алгоритм последовательного доступа к данным. Поэтому модификация кода и логики работы вычислительного ядра для перехода к асинхронной схеме вычислений лишена большого практического смысла. В то же время упрощенная реализация асинхронной схемы интегрирования по времени без разбиения на подобласти и изменения формата представления данных может применяться для улучшения масштабируемости параллельных программ за счет сокращения числа точек синхронизации MPI-процессов.

3. Оценка быстродействия

В качестве теста для оценки быстродействия программного обеспечения рассматривается задача моделирования сверхзвукового течения у поверхности сферы (M=2, Re=300) на трех смешанных сетках, состоящих из тетраэдров, треугольных призм, четырехугольных пирамид и гексаэдров. Постановка задачи, технология построения сетки и результаты расчетов более подробно обсуждаются в [12]. Расчетная область, структура сетки и картина течения показаны на рис.3.



а) расчетная область и структура сетки

б) распределение плотности и мгновенные траектории отмеченных частиц в сечении $Y=0$

Рис.3. Сверхзвуковое течение у поверхности сферы.

Базовая сетка M1 построена по заданным геометрическим параметрам генератором GAMBIT [13]. Сетки M2 и M3 получены в результате однократного (M2) и двукратного (M3) согласованного разбиения элементов M1 с добавлением новых узлов на середины ребер, в центры масс четырех-

угольных граней и центры гексаэдров. На каждом шаге измельчения число сеточных элементов увеличивается примерно в восемь раз. Параметры сеток приводятся в табл.2.

Таблица 2.

Параметр	Сетка		
	M1	M2	M3
Число вершин	132497	1039620	8236906
Число элементов	268633	2160156	17347800
Число внутренних граней	647550	5168122	41329140

Алгоритм тестирования сфокусирован на анализе зависимости между временем вычислений и методом упорядочивания данных. Рассматривается три варианта индексации: начальная, оптимизированная и негативная. Начальная индексация (IGEN) соответствует порядку перечисления элементов на выходе сеточного генератора или программы равномерного измельчения сеток. Оптимизированная индексация (IRCM) строится обратным алгоритмом Катхилла-Макки [7]. Алгоритм выбора псевдопериферийной вершины графа взят из [14]. Негативная индексация (IWRCM) генерируется путем присваивания разрозненных индексов $(0, N-1, N/2, N/4, 3N/4, \ldots)$ элементам сетки IRCM. Преобразование IRCM → IWRCM нарушает упорядоченность обращений к значениям $\mathbf{Q}_{\mathbf{cf_n}.cR}$ в цикле по граням контрольных объемов и локализацию данных относительно ячеек сетки в массиве потоков **F**.

Сравнение времени расчетов для вариантов индексации IGEN и IRCM отражает коэффициент ускорения вычислений только на конкретных сетках. Индексация IWRCM используется для наглядной демонстрации заметного снижения производительности вычислений на неупорядоченных данных.

На первом этапе тестирования газодинамический код запускается в режиме моделирования уравнений Эйлера с первым порядком точности по пространству. Конвективный поток вычисляется по схеме Лакса-Фридрихса. Здесь представлены результаты экспериментов, проводившихся на рабочей станции с 64-разрядной ОС Windows, процессором Intel Core i7-6700K (4.2 GHz, 4 ядра, 8 потоков, активирован режим "Turbo Boost") и ОЗУ размером 64 Гбайт. Для создания исполняемых файлов использовались средства Microsoft Visual Studio 2015. Поддержка набора avx инструкций отключена в настройках компилятора. Для каждого варианта индексации фиксировалось минимальное время выполнения 64 шагов численного алгоритма в серии из десяти запусков программы. Результаты тестирования представлены в табл.3. Времена работы программных модулей в последовательном ре-

жиме даны в секундах. Вариативность времени выполнения всех тестов не превышает 2.5%.

Таблица 3.

Функциональный блок	Сетка	Индексация		
		IGEN	IRCM	IWRCM
Вычисление потоков	M1	1.40	1.26	1.68
	M2	10.65	9.63	13.05
	M3	87.87	80.85 (79.60)	115.64
Обновление переменных	M1	0.38	0.28	0.51
	M2	3.09	2.26	4.76
	M3	22.19	24.51 (19.20)	47.90

Скорость работы программных модулей на разрозненных данных IWRCM по сравнению с индексацией IRCM оказывается в среднем в 1.37 раза ниже при вычислении потоков и в 1.95 раза ниже в процессе обновления переменных. Общее время расчета увеличивается на 42-55%. При замене схемы Лакса-Фридрихса на схему CUSP время вычисления потоков увеличивается в 2.2 раза для индексации IRCM и только в 1.82 раза для индексации IWRCM. То есть с ростом плотности вычислений зависимость между скоростью работы программы и алгоритмом доступа к данным прогнозируемо снижается.

Преобразование индексации IGEN методом Катхилла-Макки повышает общее быстродействие на 4.5 - 15.6 %. Однако при запуске теста на сетке M3 модуль перехода на новый временной слой замедляется в 1.1 раза, хотя модуль вычисления потоков работает, наоборот, в 1.09 раза быстрее.

Проблема локального падения производительности решается применением алгоритма переупорядочивания данных с декомпозицией. Сетка равномерно делится на заданное число подобластей, после чего в каждой из них вводится новая индексация элементов методом Катхилла-Макки. В результате индексы ячеек *k*-й подобласти находятся на числовом отрезке $\left[I^k_{first}, I^k_{last} \right]$, $I^0_{first} = 0$, $I^k_{last} + 1 = I^{k+1}_{first}$.

Времена выполнения функциональных блоков для варианта индексации IRCM с разбиением M3 на 8 подобластей добавлены в табл.3 в скобках. По сравнению с первым результатом модуль вычисления потоков работает быстрее в 1.016 раза, модуль обновления значений переменных – в 1.28 раза. Отметим, что применение переупорядочивания с декомпозицией дает положительный результат только для сеток среднего и большого размера. Дополнительное разбиение M1 и M2 не влияет на скорость работы программы.

Последний пример показывает актуальность разработки простого средства для сравнительной оценки и выбора наилучшего метода упорядочивания данных непосредственно внутри приложения до запуска газодинамического ядра. В качестве такого инструмента был реализован симулятор множественно-ассоциативного кэш процессора с алгоритмом замещения LRU (Least Recently Used). Симулятор используется для моделирования обращений исключительно к значениям газодинамических переменных в цикле по граням и обращений к буферизованным потокам в цикле по ячейкам без учета прочих массивов данных. Параметры симулятора соответствуют характеристикам кэш процессора. В табл.4 приводится полученное распределение пропусков L3-кэш. В зависимости от функционального блока значение параметра нормируется на тысячу граней или тысячу ячеек.

Таблица 4.

Функциональный блок	Сетка	Индексация		
		IGEN	IRCM	IWRCM
Вычисление потоков	M1	277	265	348
	M2	318	264	834
	M3	314	263 (270)	844
Обновление переменных	M1	1539	1506	2897
	M2	1750	1495	3106
	M3	1869	2437 (1508)	3656

Результаты в табл.4 в целом воспроизводят соотношение времен выполнения функциональных блоков. Видно, что использование алгоритма индексации с разбиением на подобласти приводит к значительному снижению пропусков кэш. Индексация IRCM характеризуется практически одинаковым значением параметра вне зависимости от сетки. То есть быстродействие вычислений на разных сетках теоретически должно быть постоянным, что подтверждается результатами экспериментов. Скорость вычисления потоков меняется в пределах 8030 – 8385 граней в секунду. Число обновляемых за секунду ячеек составляет от 14118 до 14990.

Результаты моделирования обращений к данным в сверхоперативной памяти первого и второго уровня принципиально расходятся с реальными временами выполнения функциональных блоков. Так, после преобразования индексации IREG→IRCM число пропусков кэш стабильно возрастает в 1.5 – 2 раза, но скорость работы модулей увеличивается.

На втором этапе тестирования проведена оценка эффективности распараллеливания вычислений на общей памяти. Газодинамический код запускался в режиме моделирования уравнений Навье-Стокса с повышенным порядком точности на сетке M2. В табл.5 приводятся времена выполнения

трех функциональных блоков для различного числа нитей OpenMP и двух вариантов индексации. Выполнение витков циклов равномерно распределяется между нитями (директива

«`#pragma omp parallel for schedule(static) num_threads(N)`»).

Предварительно задается привязка OpenMP-нитей к процессорным ядрам.

Таблица 5.

Число нитей OpenMP	Функциональный блок		
	Полиномиальная реконструкция	Вычисление потоков	Обновление переменных
IRCM			
1	5.32	37.44	2.35
4	2.91	10.97	1.90
8	3.00	7.52	2.03
IWRCM			
1	8.51	47.75	4.84
4	5.16	17.69	3.05
8	6.02	13.64	3.20

При запуске 4 нитей OpenMP эффективность распараллеливания составляет от 31 до 85%. Наилучший показатель соответствует циклу вычисления потоков с максимальной плотностью вычислений, наихудший – модулю обновления переменных, который характеризуется минимальным числом арифметических операций на единицу данных. В режиме гипертрединга с запуском 8 нитей OpenMP дополнительно ускоряется только модуль вычисления потоков. Похожим образом ведет себя параллельная эффективность вычислений на разрозненных данных с индексацией IWRCM, но время расчета оказывается в среднем в 1.65 раза выше.

4. Заключение

В статье рассмотрены вопросы оценки производительности и оптимизации программной реализации явного конечно-объемного алгоритма моделирования уравнений Навье-Стокса для многоядерных процессоров. Основным методом ускорения неструктурированных кодов остается переупорядочивание сеточных элементов с использованием алгоритмов уменьшения ширины ленты матрицы смежности. Корректировка индексации ячеек позволяет стабилизировать скорость вычислений вне зависимости от вида стартовой индексации и числа сеточных элементов. Для сравнения различных методов упорядочивания данных могут использоваться симуляторы множественно-ассоциативной сверхоперативной памяти процессора.

СПИСОК ЛИТЕРАТУРЫ

1. *A. Gorobets, P. Bakhvalov.* Heterogeneous CPU+GPU parallelization for high-accuracy scale-resolving simulations of compressible turbulent flows on hybrid supercomputers // Computer Physics Communications, 2022, v.271, 108231. https://doi.org/10.1016/j.cpc.2021.108231.
2. *V.E. Borisov, A.A. Davydov, I.Yu. Kudryashov, A.E. Lutsky, I.S. Men'shov.* Parallel Implementation of an Implicit Scheme Based on the LU-SGS Method for 3D Turbulent Flows // Mathematical Models and Computer Simulations, 2015, v.7, №3, p.222–232.
3. *M.M. Krasnov, P.A. Kuchugov, M.E. Ladonkina et al.* Discontinuous Galerkin method on three-dimensional tetrahedral grids: Using the operator programming method // MM & CS, 2017, v.9, №5, p.529-543. https://doi.org/10.1134/S2070048217050064.
4. *A. Gorobets, S. Soukov, P. Bogdanov.* Multilevel parallelization for simulating turbulent flows on most kinds of hybrid supercomputers // Computers and Fluids, 2018, v.173, p.171-177. https://doi.org/10.1016/j.compfluid.2018.03.011.
5. *P.D Lax*. Weak Solutions of Nonlinear Hyperbolic Equations and their Numerical Computation // Comm. Pure and Applied Mathematics, 1954, v.7, p.159-193.
6. *A. Jameson.* Positive Schemes and Shock Modelling for Compressible Flow // Int. J. Numerical Methods in Fluids, 1995, 20, p.743-776.
7. *E. Cuthill, J. McKee.* Reducing the bandwidth of sparse symmetric matrices // In Proc. ACM Nat. Conf., 1969, p.157-172.
8. *L.P. King*. An automatic reordering scheme for simultaneous equations derived from network problems // Intern. J. for Numerical Methods in Engineering, 1970, v.2, p.523-533.
9. *A.L.G.A. Coutinho, M.A.D. Martins, R.M. Sydenstricker, R.N. Elias*. Performance comparison of data-reordering algorithms for sparse matrix-vector multiplication in edge-based unstructured grid computations // Int. J. Numer. Meth. Eng., 2006, 66: p.431-460. https://doi.org/10.1002/nme.1557.
10. *В.Д. Левченко*. Асинхронные параллельные алгоритмы как способ достижения эффективности вычислений // Информац. технологии и выч. системы, 2005, т.1, с.68;
V.D. Levchenko. Asinkhronnye parallelnye algoritmy kak sposob dostizheniia effektivnosti vychislenii // Informatsionnye tekhnologii i vychislitelnye sistemy, 2005, t.1, s.68;
11. *А.Ю. Перепёлкина, В.Д. Левченко, И.А. Горячев*. Трехмерный кинетический код CFHall для моделирования замагниченной плазмы // Мат. модел., 2013, т.25, №11, с.98–110;
A.Iu. Perepelkina, V.D. Levchenko, I.A. Goriachev. Trekhmernyi kineticheskii kod CFHall dlia modelirovaniia zamagnichennoi plazmy // Matem. model., 2013, t.25, №11, s.98–110.
12. *S. Soukov*. Parallel CFD-Algorithm on Unstructured Adaptive Meshes // Math. Models & Computer Simul., 2022, v.14, №1, p.19-27. https://doi.org/10.1134/ S2070048222010197.
13. GAMBIT, http://www.ansys.com
14. *S.W. Sloan.* A Fortran program for profile and wavefront reduction // International Journal for Numerical Methods in Engineering, 1989, v.28, p.2651-2679.

Поступила в редакцию 23.06.2022
После доработки 31.10.2022
Принята к публикации 14.11.2022