ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 152 за 2018 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Горобец А.В., Нейман-заде М.И., Окунев С.К., Калякин А.А., Суков С.А.

Производительность процессора Эльбрус-8С в суперкомпьютерных приложениях вычислительной газовой динамики

Рекомендуемая форма библиографической ссылки: Производительность процессора Эльбрус-8С в суперкомпьютерных приложениях вычислительной газовой динамики / А.В.Горобец [и др.] // Препринты ИПМ им. М.В.Келдыша. 2018. № 152. 20 с. doi:10.20948/prepr-2018-152
URL: http://library.keldysh.ru/preprint.asp?id=2018-152

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

А. В. Горобец, М. И. Нейман-заде,
С. К. Окунев, А. А. Калякин, С. А. Суков

Производительность процессора Эльбрус-8С в суперкомпьютерных приложениях вычислительной газовой динамики

Москва — 2018

Горобец А. В.[1], Нейман-заде М. И.[2], Окунев С. К.[2], Калякин А. А.[2], Суков С. А.[1]

Производительность процессора Эльбрус-8С в суперкомпьютерных приложениях вычислительной газовой динамики

В работе исследуется производительность вычислений на отечественном многоядерном процессоре Эльбрус-8С в суперкомпьютерных приложениях вычислительной газовой динамики. Рассматриваются параллельные программные комплексы на основе методов повышенной точности на неструктурированных сетках для численного моделирования турбулентных течений. Описаны особенности архитектуры Эльбрус, а также подходы к адаптации и оптимизации программ. Производительность исследована как для алгоритмов в целом, так и для основных составляющих алгоритмы операций в отдельности. Представлены результаты сравнительного тестирования с различными многоядерными процессорами Intel и AMD.

Ключевые слова: высокопроизводительные вычисления, Эльбрус, вычислительная газовая динамика, CFD, суперкомпьютерное моделирование

Andrey Vladimirovich Gorobets, Murad Iskender ogly Neyman-zade, Sergey Konstantinovich Okunev, Aleksandr Alekseyevich Kalyakin, Sergey Aleksandrovich Soukov

Performance of Elbrus-8C CPU in supercomputer CFD applications

The present work is devoted to performance evaluation of a multicore CPU Elbrus-8C in supercomputer computational fluid dynamics (CFD) applications. Parallel simulation codes based on higher-accuracy methods on unstructured meshes for modelling of turbulent flows are considered. Main features of the Elbrus architecture are described. Approaches for adaptation and optimization of computing software are presented. Performance has been investigated for the whole algorithms and for their main operations separately. Benchmarking results in comparison with various Intel and AMD multicore CPUs are presented.

Key words: HPC, Elbrus processor, CFD, supercomputer simulations

Работа выполнена при поддержке Программы фундаментальных исследований Президиума РАН № 26 по приоритетным направлениям, определяемым президиумом РАН, на 2018 год «Фундаментальные основы создания алгоритмов и программного обеспечения для перспективных сверхвысокопроизводительных вычислений». Авторы благодарят Б. Н. Четверушкина и А. К. Кима за помощь и внимание к данной работе.

[1] ИПМ им. М. В. Келдыша РАН, 125047, Москва, Миусская пл., д. 4
[2] АО "МЦСТ", 119991, Москва, Ленинский проспект, д. 51

Введение

Задачи вычислительной газовой динамики являются одним из наиболее распространенных классов суперкомпьютерных приложений. Такие расчеты широко востребованы в различных областях науки и промышленности, в особенности в авиационно-космической отрасли. При этом вихреразрешающее численное моделирование турбулентного течения жидкости или газа в трехмерной расчетной области является, как правило, крайне ресурсоемким вычислительным экспериментом. Нестационарный расчет обтекания отдельных элементов летательного аппарата, например секции крыла, шасси, лопасти винта вертолета, отсека вооружения, может требовать порядка $10^{17} – 10^{18}$ FLOP (floating point operation – арифметическая операция с числами с плавающей точкой). Расчеты сложных конфигураций, таких как крыло с механизацией, несущий винт вертолета с учетом влияния фюзеляжа, могут требовать порядка 10^{19} FLOP. Крупные расчеты фундаментальных задач методом прямого численного моделирования могут потреблять 10^{20} FLOP и более. Под задачи такого класса выделяются десятки миллионов процессорных часов.

Интенсивный рост производительности суперкомпьютеров открывает все более широкие возможности применения численного моделирования. Производительность процессоров в настоящее время повышается в основном за счет увеличения числа ядер и числа операций, выполняемых ядром за такт.

Увеличение числа операций с плавающей точкой за такт достигается различными способами. Во-первых, это расширение векторных регистров процессора. Например, векторное FMA (fused multiply-add – совмещенное умножение-сложение) устройство системы команд AVX-512 современного процессора Intel Xeon может выполнять за один такт до 16 FLOP с аргументами двойной точности (64-битный формат FP64). Ключевую роль в эффективном использовании таких SIMD (single instruction, multiple data – одиночный поток команд, множественный поток данных) расширений играют возможности автоматической векторизации компилятора и средства программирования для векторизации вычислений.

Во-вторых, это расширение суперскалярности, то есть увеличение числа исполнительных устройств, которые могут одновременно выполнять независимые по данным операции. Например, процессоры Intel Xeon серии Platinum и Intel Xeon Phi серии 7200 имеют по два 512-битных FMA устройства, что позволяет одному ядру выполнять до 32 FP64 операций за такт. В этом случае ключевую роль играют возможности эффективного планирования исполнения независимых по данным команд в рамках SISD (single instruction, single data – одиночный поток команд, одиночный поток данных) параллелизма.

В современных процессорах Intel и AMD загрузку множественных исполнительных устройств обеспечивает ядро процессора за счет возможности внеочередного исполнения команд. Дополняет этот механизм поддержка одновременной многопоточности, которая позволяет ядру выполнять инструкции сразу из нескольких потоков (например, технология Hyper-

Threading процессоров Intel – до 4 потоков). За счет этого увеличивается число поступающих на исполнение независимых команд и повышается коэффициент загрузки множественных исполнительных устройств ядра.

В отличие от большинства распространенных западных аналогов, отечественные процессоры Эльбрус являются представителями архитектуры VLIW (very long instruction word – очень длинная машинная команда). В такой архитектуре задача распределения независимых команд по исполнительным устройствам возложена на компилятор. VLIW команда содержит в себе сразу несколько операций для параллельного выполнения суперскалярным ядром процессора. Ядро процессора Эльбрус-8С имеет шесть 64-битных FMA устройств, что позволяет выполнять до 12 FP64 FLOP за такт.

Для оценки производительности процессоров широко используются различные эталонные тестовые алгоритмы – бенчмарки. Наиболее распространенным является бенчмарк LINPACK [1], на котором основан мировой рейтинг суперкомпьютеров TOP500. В этом тесте решается СЛАУ с плотной матрицей прямым методом на основе LU разложения. Тест имеет вычислительную стоимость $O(N^3)$ FLOP и потребление оперативной памяти $O(N^2)$ байт, где N – число неизвестных в системе. Операции с плотными матрицами, на которых основан тест, очень удобны для процессора, что позволяет получать высокий процент от пиковой производительности, достигающий 90%. Однако данный тест сложно назвать репрезентативным. Он очень далек по своему паттерну вычислений и доступа к памяти от подавляющего большинства реальных суперкомпьютерных приложений, достигаемая производительность в которых во много раз ниже.

Другой крайностью является тест HPCG [2], в котором СЛАУ с сильно разреженной матрицей решается итерационным методом сопряженных градиентов с предобуславливателем на основе многосеточного метода. Тест имеет одинаковую асимптотику вычислительной стоимости и потребления памяти $O(N)$. В этом тесте используются весьма проблематичные для процессора операции с разреженными матрицами нерегулярной структуры, характерными для дискретизации на трехмерной неструктурированной сетке. Таким операциям свойственны крайне низкая вычислительная плотность (порядка 1 FLOP на FP64 аргумент из памяти) и нерегулярный доступ к памяти, сопряженный с частыми кэш промахами. Этот тест намного ближе к реальным приложениям, чем LINPACK, но представляется, наоборот, излишне жестоким по отношению к процессору. Достигаемая на нем производительность в пределах всего нескольких процентов от пика, как правило, в разы ниже, чем в реальных приложениях.

Также существуют различные бенчмарки из наборов репрезентативных операций, характерных для задач трехмерного моделирования, как, например, NAS Parallel Benchmarks [3]. Исследование архитектуры Эльбрус на таком наборе тестов представлено, например, в [4]. Аналогичные результаты для отечественных процессоров КОМДИВ для сравнения можно найти в [5].

Результаты таких тестов более информативны, поскольку включают операции различных типов. Тем не менее, такие тестовые операции являются сильно упрощенными, а получаемая производительность даже для близких по структуре алгоритмов зачастую далека от реальных приложений, имеющих гораздо более сложную логику и разнородные вычислительные операции.

В данной работе исследование производительности выполнено именно на примере реальных суперкомпьютерных приложений. Используются программные комплексы для трехмерного моделирования задач газовой динамики на неструктурированных сетках. Алгоритмы такого класса ориентированы на промышленные задачи. Использование неструктурированных сеток позволяет моделировать течения в областях сложной геометрии или вокруг тел сложной формы, таких как планер летательного аппарата, крыло с механизацией, винт вертолета, вентилятор авиадвигателя и т.д. Далее будут рассмотрены различные типы алгоритмических операций, оценена производительность процессора Эльбрус относительно западных аналогов Intel и AMD, приведены примеры оптимизации вычислений.

1. Расчетные коды

Рассматриваемые программные комплексы предназначены для вихреразрешающего трехмерного моделирования задач газовой динамики и аэроакустики на неструктурированных гибридных сетках. Можно отметить следующие общие характерные особенности:

1. Достаточно низкая вычислительная интенсивность, примерно 1–2 FLOP на байт из памяти, при которой производительность главным образом ограничена пропускной способностью памяти. Это, с одной стороны, заметно выше, чем на операциях линейной алгебры с разреженными матрицами (около 1/8 FLOP на байт), но, с другой стороны, в несколько раз ниже, чем соотношение пиковой производительности и пропускной способности памяти современных процессоров (около 10).
2. Большой объем обрабатываемых данных, ассоциированных с элементами расчетной области – узлами сетки, гранями контрольных объемов, сеточными элементами и т.д., при котором входные данные основных ресурсоемких операций многократно превышают объем кэш памяти процессора.
3. Нерегулярный доступ к памяти, косвенная адресация при доступе к инцидентным элементам расчетной области, что обусловлено дискретизацией на неструктурированной сетке. Такой паттерн доступа приводит к существенному влиянию на производительность кэш промахов и латентности доступа к памяти. Соответственно, от процессора требуется эффективная работа кэш памяти и механизмов сокрытия латентности.

Для распараллеливания используется двухуровневый подход [6], сочетающий модели параллельных вычислений с распределенной и общей памятью и соответствующий архитектуре суперкомпьютера на основе многоядерных процессоров. На верхнем уровне используется MPI (Message passing interface – интерфейс передачи сообщений) для объединения многоядерных процессоров в рамках модели с распределенной памятью, для распараллеливания по ядрам процессора используется OpenMP (Open multiprocessing – открытое мультипроцессирование).

Кроме того, выполненные на языке C++ программные реализации репрезентативны с точки зрения сравнения производительности процессоров, поскольку представляют некоторый компромисс между трудоемкостью разработки и вычислительной оптимизацией. С одной стороны, достаточно много внимания уделено повышению производительности вычислений, оптимальной организации структуры данных, повышению эффективности доступа к памяти и так далее. С другой стороны, в кодах не используется низкоуровневая оптимизация, снижающая переносимость, сильно повышающая трудоемкость, осложняющая последующую модификацию кода и снижающая удобство работы с кодом. В кодах не применяются ассемблерные вставки, интринсики (встроенные функции), зависимые от компилятора директивы, явная развертка циклов и т.д.

1.1. Программный комплекс NOISEtte

Основной областью приложений NOISEtte [7] является суперкомпьютерное моделирование задач аэродинамики и аэроакустики, характерных главным образом для авиационно-космической отрасли. Типичный пример вихреразрешающего расчета показан на рис. 1.

Течение вязкого идеального газа описывается системой уравнений Навье–Стокса. Для моделирования турбулентности система дополняется необходимыми источниковыми членами или уравнениями в рамках одного из следующих подходов: прямое численное моделирование, осредненные по Рейнольдсу уравнения (RANS), метод крупных вихрей (LES), гибридные RANS-LES методы.

Для численного решения уравнений Навье–Стокса используется конечно-объемный метод на неструктурированных сетках с определением переменных в узлах. Дискретизация по пространству осуществляется с помощью экономичных реберно-ориентированных схем повышенной точности на основе квазиодномерной реконструкции [8]. Для расчета потока через грани контрольных объемов реализованы различные методы решения задачи Римана о распаде разрыва. Обзор таких методов представлен в [9]. Для интегрирования по времени используются явные схемы Рунге–Кутты (до 4-го порядка) или неявные схемы на основе линеаризации по Ньютону (до 2-го порядка). В случае неявной схемы для решения СЛАУ с матрицей Якоби используется предобусловленный метод Bi-CGSTAB [10]. Программный

комплекс NOISEtte имеет комбинированное MPI+OpenMP распараллеливание для кластерных систем на основе многоядерных процессоров. Более подробно параллельный алгоритм и показатели параллельной эффективности в расчетах на различных суперкомпьютерах представлены в [11].

Для анализа производительности в алгоритме были выделены следующие основные ресурсоемкие операции.

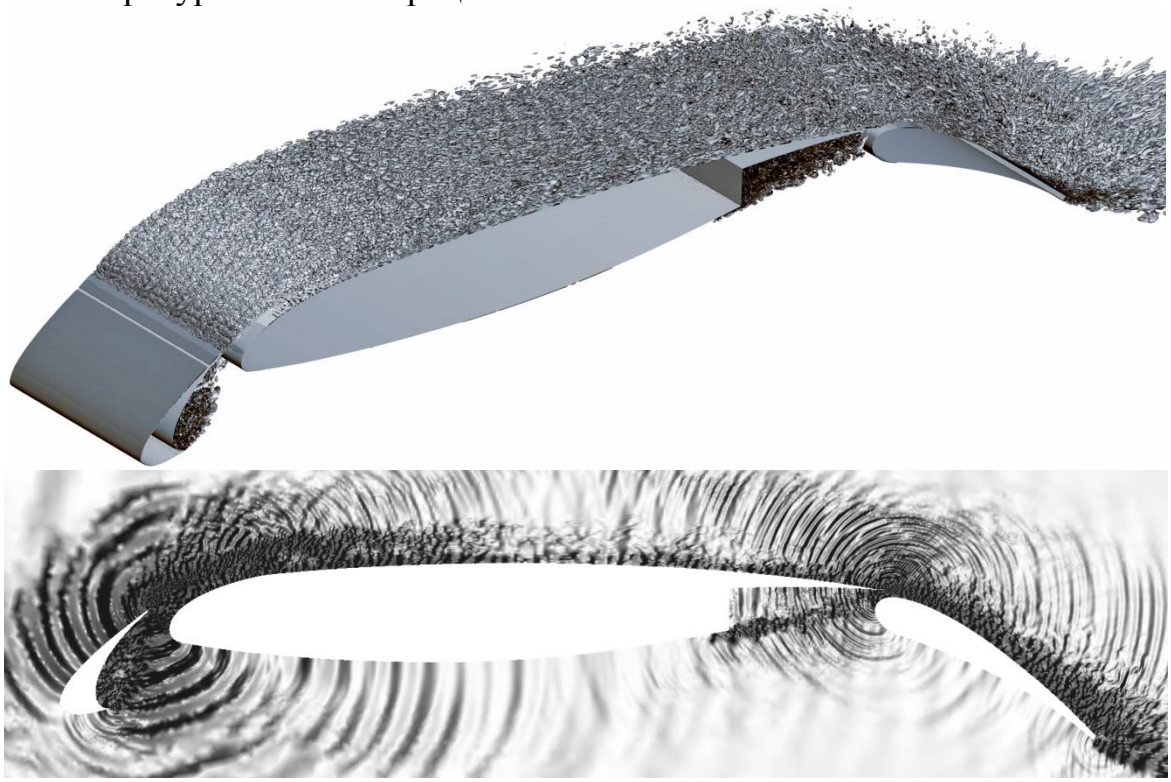

Рис. 1. Пример вихреразрешающего расчета. Обтекание секции крыла с выпущенной механизацией: турбулентные структуры (сверху) и генерируемые ими акустические поля (снизу).

Расчет конвективных потоков. Операция представляет собой цикл по множеству внутренних граней ячеек, ассоциированных с ребрами сетки. Для расчета потока через грань f между ячейками вокруг узлов i и j (рис. 2) по заданным в узлах значениям переменных (плотность, компоненты вектора скорости, давление) выполняется реконструкция значений "слева" и "справа" от центра грани, L и R. Для этого используется интерполяционная конструкция вдоль прямой e, проходящей через узлы i и j, которая включает узлы из трех уровней соседства. В трехмерном случае шаблон состоит из 14 узлов. Затем для наборов значений L и R решается вычислительно-интенсивная задача Римана о распаде разрыва для нахождения результирующего потока через грань f. При этом в случае неявной схемы также рассчитывается вклад грани в коэффициенты приближённой матрицы Якоби, в портрет которой входят только прямые связи между узлами через общую грань.

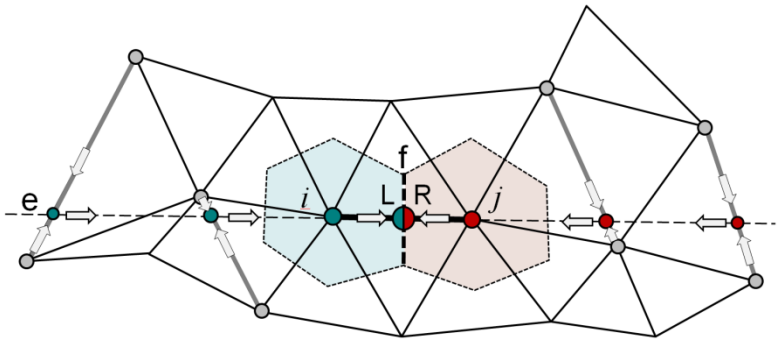

Рис. 2. Шаблон схемы расчета потока через грань *f* между ячейками *i* и *j*

Особенностью данной операции является выборка данных из узлов сетки путем косвенной адресации к данным, расположенным в памяти произвольным образом, зачастую на расстоянии, превышающем длину кэш линии. Также на этапе расчета задачи Римана используется сложная логика с большим количеством ветвлений и несколькими уровнями вызовов подпрограмм.

На вход для каждой грани из узлов поступают 70 вещественных значений переменных (по 5 из 14 узлов), 10 значений суммарных потоков в узлах (по 5 из двух узлов), на выходе записываются 10 значений суммарных потоков (по 5 в два узла). В случае неявной схемы также считываются и записываются 100 значений коэффициентов в матрицу Якоби (4 блока по 25 коэффициентов). Вклад данной операции в общее время вычислений составляет около 20%.

Расчет вязких потоков. Для данной операции используется обратная сеточная топология, т.е. граф, в котором вершинам соответствуют узлы сетки, а ребрам между двумя вершинами соответствует сеточный элемент, в который входят соответствующие вершинам узлы. Операция представлена в виде цикла по ребрам графа обратной топологии, в котором для каждого ребра рассчитывается вклад в суммарные потоки в узлах и в матрицу Якоби. На вход из узлов поступают 10 значений переменных и 10 значений суммарных потоков (по 5 из двух узлов), на выходе также 10 значений суммарных потоков и 100 значений коэффициентов в матрицу Якоби в случае неявной схемы. В этой операции логика работы менее сложна, чем в случае конвективных потоков, однако ресурсоемкость также высока. Много вычислений затрачивается на расчет коэффициентов локальных матриц размера 5×5 и алгебраические операции с этими матрицами. Вклад расчета вязких потоков в общее время составляет около 30%.

В случае неявной схемы с линеаризацией по Ньютону на каждой итерации решается СЛАУ с матрицей Якоби. Для этого используется предобусловленный метод Bi-CGSTAB [10], в котором основными операциями являются матрично-векторное произведение с разреженной матрицей (МВП), скалярное произведение, линейная комбинация векторов. Из них наиболее ресурсоемкой операцией является МВП.

Матрично-векторное произведение. Для представления разреженной матрицы нерегулярной структуры используется блочный построчно-

разреженный формат CSR (Compressed Sparse Row). Размер блока равен числу физических переменных (5 в трехмерном случае). Операция реализована в цикле по узлам сетки. На вход для каждого узла поступают коэффициенты соответствующей строки матрицы. Число ненулевых блоков в строке зависит от типа сеточных элементов, для тетраэдральной сетки оно в среднем около 15. Для этого числа объем данных на вход составляет 375 значений коэффициентов, расположенных в памяти компактно, и 75 значений входного вектора в 15 блоках, расположенных в памяти произвольным образом. В выходной вектор записывается 5 значений. Вклад данной операции в общее время вычислений составляет около 20%.

1.2. Программный комплекс Tapir

Tapir предназначен для расчета дозвуковых и сверхзвуковых течений вязкого сжимаемого газа. Дискретизация уравнений Навье–Стокса выполняется на основе метода конечного объема с определением значений сеточных функций в центрах масс элементов гибридной неструктурированной сетки.

Конвективный поток через грани контрольных объемов определяется с использованием одной из схем (HLLE, HLLC, AUSM, схема Роу и т.д., см. [9]) решения задачи Римана о распаде произвольного разрыва. Повышение порядка пространственной аппроксимации достигается за счет полиномиальной реконструкции сеточных функций внутри многогранников. Необходимые для расчета диффузионного потока значения первых производных вычисляются путем осреднения компонент вектора градиента в примыкающих к грани ячейках. Коэффициенты линейных или квадратичных реконструкционных полиномов определяются на основе дискретного аналога формулы интегрального представления градиента или метода наименьших квадратов соответственно. Интегрирование по времени выполняется явным методом первого или повышенного порядка точности.

Программная реализация одного шага интегрирования по времени включает в себя последовательный вызов процедур определения полиномиальных коэффициентов, расчета градиентов, вычисления потоков и обновления значений переменных. Каждая процедура содержит внутри себя цикл либо по ячейкам расчетной сетки, либо по их граням. Витки циклов не имеют зависимости по данным. Более развернутое описание вычислительного алгоритма и его программной реализации для различных платформ можно найти в работе [12].

Для анализа производительности были выбраны две основные ресурсоемкие операции: расчет градиентов сеточных функций и вычисление потоков через грани ячеек.

Расчет градиентов. Операция выполняется в цикле по ячейкам. На вход для каждой ячейки поступает от 40 (для тетраэдра) до 56 (для гексаэдра) вещественных значений: 5 физических переменных в ячейке, по 5 переменных из соседних ячеек, по 3 геометрических коэффициента для самой ячейки и ее

соседей. При этом геометрические коэффициенты для ячейки расположены в памяти линейно, а доступ к значениям переменных в соседних ячейках осуществляется путем косвенной адресации с использованием целочисленного массива, хранящего адреса соседних ячеек. На выходе записывается 15 значений градиентов переменных. Данная операция имеет низкую вычислительную интенсивность, близкую к матрично-векторному произведению. Вклад данной операции в общее время составляет 10–20%.

Расчет потоков через грани. Это основная операция алгоритма, представленная в виде цикла по внутренним граням ячеек. Для каждой грани сначала определяются реконструированные значения газодинамических переменных с двух сторон от центра масс грани, затем вычисляются конвективный и вязкий потоки, для чего вызываются соответствующие относительно вычислительноемкие подпрограммы. На вход поступают 12 вещественных значений параметров грани (координаты центра масс, площадь, компоненты нормали и т.д.) и 2 целочисленных значения адресов инцидентных грани ячеек. Из этих двух ячеек поступает 46 вещественных значений: по 3 координаты центра массы, 5 значений переменных и 15 значений градиентов. На выходе записывается 5 значений результирующего потока через грань. Эта операция имеет более высокую вычислительную стоимость, и ее вклад в общее время составляет 60–80%.

2. Особенности архитектуры процессора Эльбрус-8С

Восьмиядерный процессор Эльбрус-8С принадлежит семейству архитектур VLIW. Каждое ядро может исполнять до 25 различных элементарных операций в одном такте. Структура широкой команды (ШК) позволяет разместить:

- 1 команду передачи управления: переход, вызов, возврат;
- 3 команды предикатной логики;
- до 6 арифметико-логических операций: целочисленная, побитовая, сдвиговая и вещественная арифметика (в том числе комбинированная), обращения к памяти, сравнения, пересылки и тернарные операторы;
- 4 команды обращения к линейно-регулярным данным, включая инкремент соответствующего указателя;
- 1 команду управления цикловым счетчиком;
- 16 байт константных данных;
- команды управления регистровым окном.

Для вещественной арифметики имеется 6 арифметико-логических устройств с полностью конвейеризированными устройствами умножения и сложения формата FP32 и FP64, позволяющих запускать и завершать исполнение fmul*, fadd*/fsub*, комбинированные (состоящие из двух операций) fmul_add*/sub*/rsub* каждый такт. Также имеется одно частично конвейеризированное устройство деления и квадратного корня, позволяющее

запускать одну из операций fdiv*, fsqrt* один раз в 2 такта. Более подробно описание архитектуры представлено в [13, 14].

Подсистема памяти посредством 4 каналов DDR3 1600 обеспечивает предельный темп работы 51.2 GB/s. Иерархия кэш-памяти представлена

- кэшем данных первого уровня на каждое ядро, 64 KB, 4-way;
- кэшем инструкций первого уровня на каждое ядро, 128 KB, 4-way;
- кэшем второго уровня на каждое ядро, 512 KB, 4-way;
- инклюзивным общим для 8 ядер кэшем третьего уровня, 16 MB, 16-way.

Процессоры можно объединять в NUMA-систему до 4 процессоров, для связи между которыми используются линки пропускной способностью до 8 GB/s в каждом направлении. Более подробное описание работы подсистемы памяти процессора представлено в [15].

Традиционно для архитектур VLIW, обеспечение производительности исполнения кода во многом переносится на этап оптимизирующей компиляции. Система программирования включает оптимизирующий компилятор lcc собственной разработки для языков C/C++/Fortran, предназначенный для выявления параллелизма операций и планирования широких команд, а также использования аппаратных возможностей процессоров Эльбрус. При этом для удобства использования обеспечена совместимость с системой программирования GNU Compiler Collection версии 5.5.0 по опциям и основным расширениям стандартов языков.

3. Адаптация программы к архитектуре Эльбрус

Основными особенностями оптимизации кода для VLIW архитектуры являются:

- повышенные требования к возможности инлайн-подстановки процедур с целью расширения контекста для обнаружения параллелизма операций;
- более высокий эффект от межмодульной (или полнопрограммной) оптимизации по сравнению с другими архитектурами;
- двухфазная компиляция с генерацией и использованием профиля исполнения задачи;
- более высокая отзывчивость к разрешению конфликтов между операциями обращения к памяти.

Анализ производительности кодов NOISEtte и Tapir на вычислительном комплексе Эльбрус-801 показал несколько схожих проблем:

- из-за объёма данных кэши не могут полностью обеспечить быстрый доступ, есть проблема с блокировками по времени чтения из памяти;
- есть плохо разрешаемые зависимости между указателями;
- в горячих участках кода присутствуют циклы с малым переменным числом итераций, не позволяющие провести ни оптимизацию полной раскрутки, ни эффективную программную конвейеризацию;

– есть межмодульные вызовы процедур небольшого размера, инлайн-подстановка которых была бы полезна.

Для борьбы с этими негативными эффектами были использованы два подхода: настройка параметров оптимизирующего компилятора и точечные модификации исходного кода в наиболее важных с точки зрения производительности участках.

Использовались следующие настройки компилятора:

– опция -fcache-opt, включающая режим конвейеризации с дополнительным увеличением расстояния от операций чтения до момента использования их результатов;
– опции -frestrict-all, -frestrict-params, которые необходимы для установки спецификатора restrict на локальные указатели и параметры процедуры;
– режим работы межмодульной оптимизации -fwhole в сочетании с точечной расстановкой атрибутов функций __attribute((always_inline));
– опции -fprofile-generate и -fprofile-use для двухфазной компиляции с получением (путем тестового запуска приложения) и использованием счетчиков исполнения линейных участков;
– подкачка данных в кэш в сочетании с опцией -fset-ld-delay=13 для выдерживания дистанции при получении данных из кэш;
– опция выбора целевой архитектуры -mcpu=elbrus-v4;
– стандартные флаги оптимизации -O4 -ffast -ffast-math.

Поскольку у VLIW архитектуры за планирование загрузки множественных исполнительных устройств отвечает компилятор, некоторая помощь компилятору со стороны программиста позволяет существенно повысить производительность вычислений.

Большинство модификаций исходного кода NOISEtte и Tapir было связано с подстановкой подкачки данных путем обращения к встроенной функции __builtin_prefetch, с помощью чего снижаются потери на ожидание поступления данных из оперативной памяти.

Для помощи компилятору в разрешении зависимости между указателями точечно добавлены ключевые слова __restrict.

Выполнены точечные модификации кода для устранения зависимостей при доступе к локальным данным. Пример устранения конфликта между двумя фрагментами кода, мешавшего обоим фрагментам выполняться одновременно на разных исполнительных устройствах, приведен в таблице 1.

Выполнена точечная ручная модификация малоитерационных циклов для выравнивания количества их итераций для последующей автоматической полной раскрутки циклов на этапе компиляции. Пример модификации циклов приведен в таблице 2. Устранение условного выхода из цикла сделало его доступным для полной раскрутки. Перестановка порядка обхода n-k в гнездах необходима для разрыва зависимостей после раскрутки гнёзд и более плотной упаковки кода внешнего цикла.

Таблица 1

Пример устранения зависимости между участками кода

Исходная версия	Модифицированная версия

`double dXYZ[3];`
 	
`for(j=0; j<3; ++j)`
`  dXYZ[j] = I[i].fMC[j]`
`  - CE[I[i].eL*3+j];`
`for(j=0; j<5; ++j)`
`  QL[j] = Qph[I[i].eL].Q[j]`
`  + dQph[I[i].eL].dQ[j][0]*dXYZ[0]`
`  + dQph[I[i].eL].dQ[j][1]*dXYZ[1]`
`  + dQph[I[i].eL].dQ[j][2]*dXYZ[2];`
 	
`//зависимость по dXYZ между частями`
 	
`for(j=0; j<3; ++j)`
`  dXYZ[j] = I[i].fMC[j]`
`  - CE[I[i].eR*3+j];`
`for(j=0; j<5; ++j)`
`  QR[j] = Qph[I[i].eR].Q[j]`
`  + dQph[I[i].eR].dQ[j][0]*dXYZ[0]`
`  + dQph[I[i].eR].dQ[j][1]*dXYZ[1]`
`  + dQph[I[i].eR].dQ[j][2]*dXYZ[2];`
	`{`
	` double dXYZ[3];`
	` for(j=0; j<3; ++j)`
	`  dXYZ[j] = I[i].fMC[j]`
	`  - CE[I[i].eL*3+j];`
	` for(j=0; j<5; ++j)`
	`  QL[j] = Qph[I[i].eL].Q[j]`
	`  + dQph[I[i].eL].dQ[j][0]*dXYZ[0]`
	`  + dQph[I[i].eL].dQ[j][1]*dXYZ[1]`
	`  + dQph[I[i].eL].dQ[j][2]*dXYZ[2];`
	`}`
	`{`
	` double dXYZ[3];`
	` for(j=0; j<3; ++j)`
	`  dXYZ[j] = I[i].fMC[j]`
	`  - CE[I[i].eR*3+j];`
	` for(j=0; j<5; ++j)`
	`  QR[j] = Qph[I[i].eR].Q[j]`
	`  + dQph[I[i].eR].dQ[j][0]*dXYZ[0]`
	`  + dQph[I[i].eR].dQ[j][1]*dXYZ[1]`
	`  + dQph[I[i].eR].dQ[j][2]*dXYZ[2];`
	`}`

Таблица 2

Пример модификации циклов

Исходная версия	Модифицированная версия

`//цикл с переменной границей`
`//от 4 до 6 (посредством break)`
`for(int j=0; j<6; ++j){`
`  int ic = IC[i*6+j];`
`  if(ic == -1) break;`
`  for(int n=0; n<5; ++n)`
`  for(int k=0; k<3; ++k)`
`  dQ[n][k] +=`
`  Qph[ic].Q[n]*kXYZ[k];`
`}`
	`//цикл с постоянной границей`
	`//лишние слагаемые зануляются`
	`//дополнительным множителем`
	`double C=1.0;`
	`for(int j=0; j<6; ++j){`
	`  int ic = IC[i*6+j];`
	`  if(ic == -1){ ic=0; C=0.0;}`
	`  for(int k=0; k<3; ++k)`
	`  for(int n=0; n<5; ++n)`
	`  dQ[n][k] +=`
	`  C*Qph[ic].Q[n]*kXYZ[k];`
	`}`

Все вышеуказанные модификации совместимы с компиляторами GCC и Intel. Для компилятора Microsoft Visual Studio требуется добавить пустые определения для отсутствующих ключевых слов и функций.

Изменения в исходном коде NOISEtte и Tapir суммарно затронули всего несколько десятков строк. Для кода NOISEtte модификации дали ускорение на архитектуре Эльбрус около 1.3 раз, для кода Tapir – более 2 раз. При этом изменения не оказали заметного влияния на время вычислений на процессорах Intel, имеющих внеочередное исполнение команд.

4. Сравнение производительности

В тестовых расчетах моделируется течение вязкого сжимаемого газа. Используются небольшие сетки, число элементов в которых примерно соответствует обычной характерной нагрузке на процессор в расчетах больших задач на кластерных системах.

Тесты кодом NOISEtte выполняются на неструктурированной сетке с преобладанием тетраэдров, состоящей из 119 тыс. узлов и 679 тыс. элементов. Используется неявная схема по времени 2-го порядка и вершинно-центрированная схема EBR5 [8] повышенной точности по пространству. Для решения задачи распада разрыва на гранях используется схема Poy [16].

Для кода Tapir выбрана сетка из 273 тыс. узлов и 445 тыс. элементов. Используется схема с полиномиальной реконструкцией (линейные полиномы) по пространству и явная схема Эйлера 1-го порядка по времени. Для распада разрыва также используется схема Poy.

Измеряются следующие основные алгоритмические операции:

– *NOISEtte Total* – расчет кодом NOISEtte, весь алгоритм;
– *NOISEtte CFlux* – операция расчета конвективных потоков;
– *NOISEtte VFlux* – операция расчета вязких потоков;
– *NOISEtte SpMV* – МВП с разреженной матрицей в решателе;
– *Tapir Total* – расчет кодом Tapir, весь алгоритм;
– *Tapir Grad* – операция расчета градиентов сеточных функций;
– *Tapir Flux* – операция расчета потоков.

Описание этих алгоритмических операций приведено в разделе 1.

Для сравнительного тестирования выбраны несколько западных аналогов процессора Эльбрус-8С. Среди них два близких по характеристикам процессора, также работающих с памятью DDR3 и выполненных по техпроцессу 22 нм и более: AMD Opteron 6276, ядро Interlagos; Intel Xeon E5-2650v2, ядро Ivy Bridge. Также присутствуют процессоры Intel из числа наиболее современных на данный момент, работающих с памятью DDR4 и сделанных по техпроцессу 14 нм: Intel Xeon E5-2683v4, ядро Broadwell, и Intel Xeon Platinum 8160, ядро Skylake. В таблице 3 приведены количество ядер, тактовая частота (ГГц), пиковая производительность (GFLOPS), пропускная способность памяти (GB/s), энергопотребление (Вт), техпроцесс (нм).

Таблица 3

Основные характеристики процессоров

CPU	ядер	ГГц	GFLOPS	GB/s	Вт	нм
Эльбрус-8С	8	1.3	125	51	80	28
AMD Opteron 6276 (Interlagos)	16	2.3	147	51	115	32
Intel Xeon E5-2650v2 (Ivy Bridge)	8	2.6	166	60	95	22
Intel Xeon E5-2683v4 (Broadwell)	16	2.1	294	77	120	14
Intel Xeon Platinum 8160 (Skylake)	24	2.1	1612	128	150	14

Параллельное ускорение. Данное измерение показывает, во сколько раз ускоряется расчет на многоядерном процессоре в многопоточном режиме с OpenMP распараллеливанием относительно последовательного исполнения на том же процессоре. Результаты ускорения на 8 ядрах представлены в таблице 4. Ускорение на процессоре Эльбрус-8С, 5–6 раз, в целом хорошо соответствует западным аналогам. У большинства процессоров можно отметить слабое ускорение операций с низкой вычислительной интенсивностью, ограниченных пропускной способностью памяти (SpMV, Grad). При этом Skylake демонстрирует одинаково высокое ускорение на всех операциях, благодаря более мощной подсистеме памяти, имеющей 6 каналов DDR4.

Таблица 4

Параллельное ускорение, 8 нитей OpenMP на 8 ядрах

Операция	Эльбрус-8С	AMD 6276	Intel Ivy Bridge	Intel Broadwell	Intel Skylake
NOISEtte CFlux	6.4	6.1	6.3	5.9	7.7
NOISEtte VFlux	6.7	5.3	5.5	5.6	7.5
NOISEtte SpMV	4.4	1.7	3.7	4.7	7.0
NOISEtte Total	**5.8**	**3.5**	**5.2**	**5.4**	**7.2**
Tapir Grad	1.8	3.4	4.4	4.3	7.2
Tapir Flux	6.7	5.3	6.8	6.2	7.6
Tapir Total	**5.0**	**4.6**	**6.2**	**5.8**	**7.5**

Сравнение одиночного ядра. В этом тесте сравнивается производительность вычислений в последовательном режиме. Результаты, представленные в таблице 5, показывают соотношение производительности с процессором Эльбрус-8С, скорость работы которого принята за единицу.

На коде NOISEtte разница с процессорами Intel составила около 3 раз. В пересчете производительности на такт это соответствует разнице около полутора раз (поскольку у процессоров Intel частота примерно вдвое выше).

Относительно AMD проигрыш составил около 1.4 раз.

Важно отметить, что NOISEtte имеет существенно более сложную реализацию и логику работы (вычислительная часть порядка десятка тысяч строк), чем Tapir (порядка тысячи строк). Для большинства ресурсоемких операций NOISEtte не выполнялось никакой специальной адаптации к архитектуре Эльбрус. Для Tapir была выполнена сравнительно несложная адаптация вычислений под архитектуру Эльбрус, описанная выше в разделе 3.

На коде Tapir разница с Intel составила всего около полутора раз. Таким образом, на данном приложении Эльбрус-8С демонстрирует более высокую производительность на такт, чем аналоги Intel. Ядро AMD оказалось медленнее Эльбруса примерно в полтора раза.

Таблица 5

Производительность относительно Эльбрус-8С: одно ядро

Операция	AMD 6276	Intel Ivy Bridge	Intel Broadwell	Intel Skylake
NOISEtte CFlux	1. 40	3.43	3.63	2.97
NOISEtte VFlux	1.20	2.75	2.53	1.80
NOISEtte SpMV	0.90	2.14	1.73	1.82
NOISEtte Total	**1.27**	**2.82**	**2.80**	**2.28**
Tapir Grad	0.52	1.13	1.30	0.84
Tapir Flux	0.63	1.50	1.67	1.31
Tapir Total	**0.68**	**1.58**	**1.75**	**1.34**

Сравнение производительности на 8 ядрах. В этом тесте сравнивается производительность вычислений в многопоточном режиме на одинаковом с Эльбрус-8С числе ядер. В параллельном режиме уже существенное влияние на результат оказывает производительность подсистемы памяти, поскольку 8 потоков могут полностью исчерпывать пропускную способность памяти. Результаты по соотношению с Эльбрусом показаны в таблице 6. Сравнивая с таблицей 5, можно отметить, что на коде NOISEtte соотношение изменилось в пользу Эльбруса. В этом тесте Эльбрус обгоняет AMD на обоих кодах. Для кода Tapir разница с Intel составила около 2 раз.

Таблица 6

Производительность относительно Эльбрус-8С: 8 ядер

Операция	AMD 6276	Intel Ivy Bridge	Intel Broadwell	Intel Skylake
NOISEtte CFlux	1.34	3.42	3.33	3.60
NOISEtte VFlux	0.94	2.25	2.13	2.02
NOISEtte SpMV	0.34	1.80	1.85	2.88
NOISEtte Total	**0.77**	**2.55**	**2.63**	**2.87**
Tapir Grad	1.01	2.81	3.22	3.48
Tapir Flux	0.50	1.53	1.54	1.51
Tapir Total	**0.62**	**1.95**	**2.01**	**2.00**

Сравнение всего процессора. В этом тесте сравнивается производительность вычислений в многопоточном режиме на всех ядрах процессора. Результаты приведены в таблице 7. Процессор AMD смотрится заметно слабее Intel. 8-ядерный Эльбрус-8С обгоняет на коде Tapir этот 16-ядерный процессор. 16- и 24-ядерные процессоры Intel за счет высокого OpenMP ускорения заметно прибавили относительно 8-ядер, разница с Эльбрусом составила уже примерно 3–7 раз.

Таблица 7

Производительность относительно Эльбрус-8С: весь процессор

Операция	AMD 6276	Intel Ivy Bridge	Intel Broadwell	Intel Skylake
NOISEtte CFlux	2.56	3.42	6.41	10.25
NOISEtte VFlux	1.87	2.25	2.25	5.72
NOISEtte SpMV	0.61	1.80	2.53	4.86
NOISEtte Total	**1.39**	**2.55**	**4.17**	**6.82**
Tapir Grad	0.81	2.81	3.63	7.25
Tapir Flux	0.88	1.53	2.90	4.13
Tapir Total	**0.89**	**1.95**	**3.30**	**4.94**

Производительность вычислений. Для измерения фактической производительности было посчитано число арифметических операций в коде Tapir. Также в качестве примера операции, сильно ограниченной пропускной способностью памяти, была выбрана операция NOISEtte SpMV, имеющая наименьшую вычислительную интенсивность из рассматриваемых – около 0.2 FLOP на байт, что примерно в 10 раз ниже, чем у кода Tapir. Результаты представлены в таблице 8, в которой также приводится процент от теоретического пика устройств.

Таблица 8

Производительность, GFLOPS, и процент от пика

	Эльбрус-8С	AMD 6276	Intel Ivy Bridge	Intel Broadwell	Intel Skylake
NOISEtte SpMV					
1 ядро	1.0 (6.5%)	0.9 (9.9%)	2.2 (10%)	1.8 (9.5%)	1.8 (2.7%)
8 ядер	4.4 (3.6%)	1.5 (2.1%)	8 (4.8%)	8.2 (5.6%)	12.8 (2.4%)
процессор	4.4 (3.6%)	2.7 (1.8%)	8 (4.8%)	11.2 (3.8%)	21.6 (1.3%)
Tapir Total					
1 ядро	3.1 (20%)	2.1 (23%)	4.9 (23%)	5.4 (29%)	4.1 (6.1%)
8 ядер	15.5 (12%)	9.5 (13%)	30 (18%)	31 (21%)	31 (5.8%)
процессор	15.5 (12%)	13.7 (9%)	30 (18%)	59 (17%)	77 (4.7%)

Из результатов видно, что на SpMV достигается заметно меньший процент от пика, поскольку пик по числу арифметических операций у процессоров многократно превосходит пропускную способность памяти. Соотношение производительности и пропускной способности можно оценить по таблице 3.

Также можно отметить низкий процент от пика у Skylake, который имеет учетверенное число операций на такт по сравнению с предшественником Broadwell. Можно сделать вывод, что на рассматриваемом типе алгоритмов, в основном в силу ограничений пропускной способности памяти, удвоение числа

векторных арифметических устройств и расширение в 2 раза векторных регистров не дало прироста производительности.

Заключение

Производительность вычислений на многоядерных процессорах Эльбрус-8С была исследована на реальных приложениях вычислительной газовой динамики. Использовались два программных комплекса для моделирования сжимаемых течений на неструктурированных сетках, NOISEtte [6] и Tapir [11].

Рассматривались несколько моделей процессоров Intel Xeon, от моделей 5-летней давности до наиболее современных. Чудес, конечно, не бывает, Эльбрус ожидаемо оказался медленнее процессоров Intel. Проигрыш по производительности ядра составил в среднем 2.6 раза для кода NOISEtte и 1.5 раза для кода Tapir. Это представляется достаточно хорошим результатом, учитывая, что тактовая частота Эльбрус-8С примерно вдвое ниже, т. е. в пересчете на такт Эльбрус не уступает Intel. Кроме того, проигрыш около двух раз на данном классе приложений является типичным даже для процессоров AMD, основного конкурента Intel. По производительности всего процессора проигрыш относительно Intel на коде NOISEtte составил от 2.5 раз против 8-ядерного Ivy Bridge до 6.8 раз против 24-ядерного Skylake, а на коде Tapir – от 2 до 5 раз, соответственно.

Для сравнения был рассмотрен 16-ядерный процессор AMD Opteron 6276 примерно 5-летней давности. Этот процессор проиграл соответствующему по времени 8-ядерному Intel Ivy Bridge примерно в 2 раза. На коде NOISEtte 8-ядерный Эльбрус-8С с частотой 1.3 ГГц оказался медленнее 16-ядерного процессора AMD с частотой 2.3 ГГц всего в 1.4 раза, а на коде Tapir Эльбрус обогнал AMD на 12%.

Также стоит отметить, что у процессоров Intel на данном типе приложений наблюдается отсутствие роста производительности ядра. Более того, современное ядро Intel Skylake оказалось на 20% медленнее ядра Intel Ivy Bridge 5-летней давности. Расширение векторных регистров и удвоение числа арифметических устройств не дало прироста производительности (с использованием только средств автоматической векторизации компилятора), в силу ограниченной пропускной способности памяти. Рост производительности современных процессоров связан в основном с увеличением числа ядер.

В то же время у процессора Эльбрус-8С производительность ядра относительно предыдущего поколения Эльбрус-4С выросла примерно в полтора раза. Это позволяет надеяться, что с выходом следующего поколения процессоров Эльбрус-16С отставание еще сократится. Ожидается, что следующая модель, выпуск которой запланирован на 2022 год, будет иметь 16 ядер, работающих на частоте 2 ГГц. Предполагается использование от 4 до 8 каналов памяти DDR4-2666, что может увеличить пропускную способность подсистемы памяти более чем в 3 раза. Также существенный вклад в рост

производительности может внести дальнейшее совершенствование оптимизирующего компилятора.

Список литературы

1. Dongarra J. The LINPACK Benchmark: An explanation. // ICS 1987: Supercomputing. Lecture Notes in Computer Science. Vol 297. P. 456–474. DOI 10.1007/3-540-18991-2_27
2. Dongarra J., Heroux M. A., Luszczek P. HPCG Benchmark: a New Metric for Ranking High Performance Computing Systems. // Technical Report, Electrical Engineering and Computer Science Department, Knoxville, Tennessee, UT-EECS-15-736, November, 2015. URL: http://www.hpcg-benchmark.org/
3. Bailey D., Barszcz E., Barton J.T., Browning D.S., Carter R.L., Dagum D., Fatoohi R.A., Frederickson P., Lasinski T.A, Schreiber R., Simon H., Venkatakrishnan V., Weeratunga K. The NAS Parallel Benchmarks. // International Journal of High Performance Computing Applications. 5. 63-73. 1991. DOI: 10.1177/109434209100500306.
4. Тютляева Е.О., Конюхов С.С., Московский А.А., Одинцов И.О. Оценка потенциала использования платформы Эльбрус для высокопроизводительных вычислений // Суперкомпьютерные дни в России. Труды международной конференции. 2016. С. 373–385.
5. Богданов П.Б., Сударева О.Ю. Производительность процессоров КОМДИВ на ряде типовых расчётных задач // Информационные Технологии и Вычислительные Системы. 2017. Т. 4. С. 104–111.
6. Горобец А.В. Параллельная технология численного моделирования задач газовой динамики алгоритмами повышенной точности // Журнал вычислительной математики и математической физики. 2015. Т. 55. №4. С. 641–652. DOI:10.7868/S0044466915040067
7. Абалакин И.В., Бахвалов П.А., Горобец А.В., Дубень А.П., Козубская Т.К. Параллельный программный комплекс NOISETTE для крупномасштабных расчетов задач аэродинамики и аэроакустики // Вычислительные методы и программирование. Т.13. 2012. С. 110–125.
8. Бахвалов П.А., Козубская Т.К. О построении реберно-ориентированных схем, обеспечивающих точность на линейной функции, для решения уравнений Эйлера на гибридных неструктурированных сетках // Ж. вычисл. матем. и матем. физ. 2017. Т. 57. № 4. С. 92–111.
9. Toro E. F. Riemann Solvers and Numerical Methods for Fluid Dynamics. // Springer-Verlag Berlin Heidelberg 2009. DOI: 10.1007/b79761
10. Van der Vorst H. A. A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. // SIAM Journal on Scientific and Statistical Computing. 13 (2), 631–644 (1992). DOI: 10.1137/0913035

11. Gorobets A. Parallel Algorithm of the NOISEtte Code for CFD and CAA Simulations. // Lobachevskii Journal of Mathematics. 39(4). 524–532 (2018). DOI: 10.1134/S1995080218040078
12. Gorobets A., Soukov S., Bogdanov P. Multilevel parallelization for simulating turbulent flows on most kinds of hybrid supercomputers // Computers and Fluids, 2018, In Press. https://doi.org/10.1016/j.compfluid.2018.03.011
13. Kozhin A.S, Polyakov N.Y., Alfonso D.M., Demenko R.V., Klishin P.A., Kozhin E.S., Slesarev M.V., Smirnova E.V., Smirnov D.A., Smolyanov P.A., Kostenko V.O., Gruzdov F.A., Tikhorskiy V.V., Sakhin Y.K. The 5th Generation 28nm 8-Core VLIW Elbrus-8C Processor Architecture // Proceedins of the 2016 International Conference on Engineering and Telecommunication (EnT-2016). Moscow, 2016, pp. 85–89.
14. Альфонсо Д.М., Деменко Р.В., Кожин А.С., Кожин Е.С., Колычев Р.Е., Костенко В.О., Поляков Н.Ю., Смирнова Е.В., Смирнов Д.А., Смольянов П.А., Тихорский В.В. Микроархитектура восьмиядерного универсального микропроцессора «Эльбрус-8С» // Вопросы радиоэлектроники. – 2016. – № 3 – Сер. ЭВТ. — С. 6–13.
15. Кожин А.С., Нейман-заде М.И., Тихорский В.В. Влияние подсистемы памяти восьмиядерного микропроцессора «Эльбрус-8С» на его производительность // Вопросы радиоэлектроники. – 2017. – № 3 – Сер. ЭВТ. — С. 13–21.
16. Roe P.L. Approximate Riemann Solvers, Parameter Vectors and Difference Schemes // J. Comput. Phys. 1981. 43, N2. 357–372.

Оглавление