УДК 519.6

АДАПТАЦИЯ МЕТОДИКИ "ЭГАК" К СЧЕТУ НА ГИБРИДНЫХ ЭВМ С ГРАФИЧЕСКИМИ УСКОРИТЕЛЯМИ

А. М. Ерофеев, Е. А. Сизов, И. Ю. Турутина, И. Н. Чистякова
(ФГУП "РФЯЦ-ВНИИЭФ", г. Саров Нижегородской области)

Представлены результаты работы по адаптации методики ЭГАК к гибридным параллельным ЭВМ, содержащим графические ускорители. Методика ЭГАК предназначена для численного моделирования многомерных задач механики сплошной среды на неподвижной счетной сетке. В ней активно используется адаптивно-встраиваемая дробная сетка, вносящая нерегулярность в сеточную структуру.

Последовательно описаны подходы к решению проблем, возникших в процессе создания программы, приспособленной к счету на графических ускорителях. Рассматриваются вопросы выбора программной реализации при адаптации существующих алгоритмов методики, вопросы обмена информацией при использовании в расчетах нескольких графических ускорителей. Затронута проблема одновременного задействования всех счетных ресурсов рабочего узла (одновременный параллельный счет на всех доступных ядрах центральных процессоров и всех доступных графических ускорителях). Представлены показатели полученного ускорения при использовании гибридного вычислительного узла с графическими ускорителями относительно узла только с центральными процессорами на характерных для методики тестовых задачах.

Ключевые слова: ЭГАК, ЭГИДА-ТЕСТ, GPU, ускорение, распараллеливание, CUDA, MPI, OpenMP, эффективность, газовая динамика, адаптивно-встраиваемая дробная сетка.

Введение

Вычислительные устройства с архитектурой графических ускорителей (GPU) давно перестали быть только средствами отображения графики и прочно заняли свое место в качестве вычислительных устройств благодаря высокой степени параллелизма. Более того, GPU уже захватили лидерство в области высокопроизводительных вычислений. Активно ведутся работы по исследованию и применению данной архитектуры к решению прикладных задач [1—3]. На рынке GPU пока лидирующие позиции довольно прочно удерживает фирма Nvidia. Для использования GPU этой фирмы, как правило, применяется разработанная ею же технология CUDA [4].

В данной статье представлен первый вариант реализации разработанной и эксплуатируемой в РФЯЦ-ВНИИЭФ методики ЭГАК [5] для обеспечения счета задач с применением GPU.

Областью применения методики ЭГАК является моделирование ударно-волновых течений многокомпонентной сплошной среды с большими деформациями на многомерной пространственной неподвижной сетке с использованием ALE-подхода. Ключевой особенностью методики ЭГАК является возможность ведения счета на адаптивно-встраиваемой дробной сетке [6].

Для адаптации к счету на GPU выбран код программы ЭГИДА-ТЕСТ [7], включенный в систему тестов РФЯЦ-ВНИИЭФ для исследования параметров высокопроизводительных систем. Это упрощенная реализация методики ЭГАК, в которой присутствует лишь основной набор программ, позволяющих рассчитывать процесс газовой динамики. Расчет процесса выполняется в два этапа — лагранжев и эйлеров, решение на каждом из которых получается с помощью явных схем. В процессе адаптации к счету с использованием GPU для более широкого охвата тестовых задач методики ЭГАК в про-

грамму ЭГИДА-ТЕСТ была добавлена возможность ведения счета на адаптивно-встраиваемой дробной сетке. Это внесло элемент нерегулярности как в расчетную сетку, так и в структуры данных.

В процессе адаптации к счету на GPU понадобились глубокий анализ и существенная переработка текстов программ газовой динамики. Результатом стали новые коды, позволяющие вести работу как на одном, так и на нескольких устройствах GPU. Для более эффективного использования возможностей GPU некоторые программы были переписаны полностью с изменением алгоритма.

Работоспособность и эффективность полученной адаптированной программы ЭГИДА-ТЕСТ-GPU подтверждается результатами численных экспериментов, приведенными в статье.

Анализ существующего кода ЭГИДА-ТЕСТ и пути адаптации к счету на GPU

Адаптация программы к счету на GPU начинается с профилирования и анализа существующего кода. Результатом этих действий является набор особенностей, которые определяют дальнейший процесс адаптации.

Профилирование кода программы ЭГИДА-ТЕСТ показало, что "горячие пятна" (небольшая часть программы, занимающая более 50 % времени от общего исполнения) в коде отсутствуют. Поэтому простейший путь адаптации только горячих пятен в данном случае не подходит и необходимо адаптировать для GPU весь счетный код программы.

Анализ кода, реализующего формирование и хранение данных решаемых задач, привел к выводу о нецелесообразности реализации на GPU аналогичного способа. При запуске в программе ЭГИДА-ТЕСТ производится расчет необходимого количества памяти для решаемой задачи. Весь необходимый объем памяти ЭВМ выделяется единым фрагментом. Работа с данным фрагментом памяти организована через сложную структуру, содержащую внутри себя как другие структуры, так и указатели на отдельные участки этого фрагмента. Работа с этой структурой реализована через систему так называемых функций доступа.

При адаптации можно скопировать весь выделяемый фрагмент памяти на GPU, но тогда появится необходимость в организации на GPU структуры, аналогичной той, что имеется для CPU. При этом простого копирования заполненной структуры недостаточно, так как конечными элементами данной структуры в основном являются указатели, а они на каждом устройстве свои. Смысл в копировании на GPU указателей на память CPU отсутствует. Кроме того, для GPU адаптируется только код счетных программ, поэтому достаточно большая часть информации, хранящаяся в размеченном фрагменте памяти ЭВМ на CPU, не будет востребована на GPU. Таким образом, организовывать подобную структуру на GPU нецелесообразно.

Исходя из указанных соображений в памяти GPU выделены только те массивы, которые нужны для работы счетных программ. Размер и структура таких массивов соответствуют их расположению на CPU. В случае необходимости есть возможность обмена данными между CPU и GPU для каждого массива отдельно единым куском памяти, целиком содержащим этот массив.

При реализации счетных модулей методики ЭГАК в ЭГИДА-ТЕСТ, как правило, используются программы расчета одной ячейки. Порядок расчета ячеек, обмен данными между ячейками, рассчитываемыми разными MPI-процессами, организует типовая схема [8, 9]. Типовая схема реализована в виде функции, одним из параметров которой также является функция, выполняющая расчет ячейки. Поэтому при адаптации на GPU в качестве процесса, рассчитываемого на каждой GPU-нити, в большинстве случаев выбран расчет одной ячейки. Каждая GPU-нить ассоциируется с номером ячейки. При текущей реализации программы ЭГИДА-ТЕСТ-GPU авторы отказались от единой типовой схемы, ориентированной на использование CPU. Вместо вызова функции типовой схемы выполняется вызов ядра GPU.

Решаемая система дифференциальных уравнений на лагранжевом этапе имеет следующий вид:

$$\frac{d\vec{u}}{dt} = -\frac{1}{\rho}\,\mathrm{grad}\,P;$$
$$\frac{d\rho_i}{dt} = -\rho_i\,\mathrm{div}\,\vec{u}_i;$$
$$\frac{d\beta_i}{dt} = \beta_i\,(\mathrm{div}\,\vec{u}_i - \ \mathrm{div}\,\vec{u})\,;$$
$$\frac{de_i}{dt} = -\frac{P_i}{\rho_i}\,\mathrm{div}\,\vec{u}_i.$$

Здесь $\vec{u}$ — скорость; ρ — плотность; e — удельная внутренняя энергия; P — давление; β — объемная концентрация; i — номер компонента. Скорость определена в узлах счетной сетки, скалярные величины ρ_i, e_i, P_i, P, $\beta_i = V_i/V$ определены в центрах ячеек, где V_i и V — объемы i-го компонента вещества и всей ячейки соответственно. Индекс i в выражении для дивергенции относится не к скорости, а к дивергенции в целом.

Данная система замыкается уравнением состояния компонентов среды

$$P_i = P_i\left(\rho_i, e_i\right).$$

На эйлеровом этапе счетная сетка возвращается в исходное до лагранжева этапа состояние и осуществляется пересчет величин за счет потоков через грани ячеек (здесь используется расщепление по направлениям).

Каждый этап разделен на подэтапы, что очень удобно и для последовательной реализации, и для анализа эффективности распараллеливания.

Лагранжев этап содержит:

1) расчет предвычисленных давлений;
2) расчет ускорений на гранях ячеек;
3) пересчет ускорения и расчет скорости в узлах сетки;
4) решение уравнений неразрывности и энергии.

Эйлеров этап включает в себя расчеты:

1) потоков объема;
2) потоков масс, объемных концентраций и плотности;
3) энергии;
4) потоков импульса;
5) скорости.

Каждый из этих подэтапов адаптирован к счету на GPU независимо от остальных. Подэтап на CPU, как правило, реализован одним или несколькими вызовами функции типовой схемы. На GPU вместо функций типовой схемы вызывается реализованная функция, своя для каждого подэтапа, организующая работу одного или нескольких последовательно вызванных ядер GPU. Впоследствии объединение таких функций позволило перевести весь счет на GPU. На CPU производится работа, связанная со считыванием и записью данных. Это первоначальная инициализация задачи при считывании файлов с начальными данными, а также выдача полученных значений результатов счета в файлы.

На рис. 1 представлена схема выполнения программы ЭГИДА-ТЕСТ-GPU.

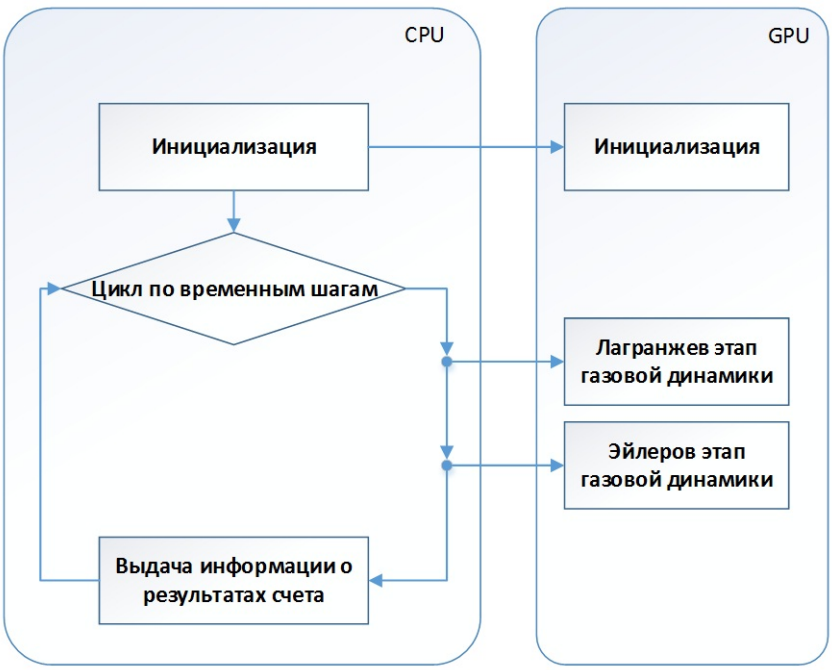

Рис. 1. Схема выполнения программы ЭГИДА-ТЕСТ-GPU при использовании одного устройства GPU

Особенности адаптации программ к счету на GPU

Коды функций расчета одной ячейки на CPU и коды ядер GPU схожи между собой. Отличия заключаются в отказе от функций доступа, используемых на CPU, в пользу прямых ссылок на используемые массивы, выделенные на GPU. Кроме того, при разработке ядер GPU авторы стремились максимально избавиться от дивергентных ветвлений (условных операторов, заставляющих разные GPU-нити выполнять неидентичную работу). Также ветвления приводят к замедлению работы вследствие того, что GPU-нити одного варпа (группа GPU-нитей, выполняющих свою работу физически одновременно) не могут одновременно рассчитывать разные ветви условного оператора.

Работа по избавлению от дивергентного ветвления на GPU схожа с векторизацией кода на CPU.

На рис. 2 показан пример, позволяющий понять необходимость отказа от дивергентного ветвления на GPU. В примере ведется расчет восьми ячеек, но в одной из них (это ячейка номер четыре) по каким-либо критериям реализуется простой случай, позволяющий сократить вычисления до нескольких операторов. GPU, благодаря своему параллелизму, рассчитывает сразу все ячейки одного варпа. Но сначала считаются все ячейки, кроме четвертой, попавшие в первую часть ветвления. После этого считается вторая часть ветвления (четвертая ячейка). Таким образом, на GPU условный оператор формально может превратиться в последовательный. Если частота срабатывания критерия реализации простого случая невелика, то лучше не сокращать вычисления в коде.

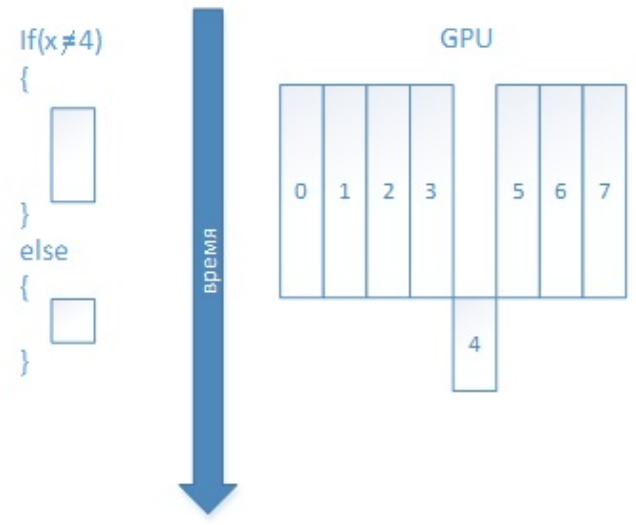

Рис. 2. Дивергентное ветвление

Примером дивергентного ветвления в коде ЭГИДА-ТЕСТ может служить ветвление при расчете средней плотности смешанной (несколько веществ) и чистой (только одно вещество в данный момент) ячеек. В смешанной ячейке выполняется суммирование произведений объемных концентраций на удельную плотность каждого вещества ячейки. В чистой ячейке одна из объемных концентраций равна единице, а остальные — нулю. Поэтому при вычислении средней плотности вещества в чистой ячейке суммирование не выполняется, а ведется поиск присутствующего вещества и берется значение его плотности. Если такой алгоритм оставить при реализации на GPU, то GPU-нити, рассчитывающие смешанные и чистые ячейки, даже в пределах одного варпа будут работать не одновременно. Поэтому при реализации на GPU в данном случае все ячейки рассматриваются как смешанные (в чистой ячейке суммируются преимущественно нулевые слагаемые).

Другая особенность, напоминающая дивергентное ветвление, но при этом используемая для решения проблемы декомпозиции вычислительной работы при зависимости по данным, — это маркировка признаков, определяющих, считать ту или иную величину в данной ячейке или ее можно не считать, а брать уже посчитанное значение из соседней ячейки. Такая маркировка часто используется в программе ЭГИДА-ТЕСТ. Ее применение является следствием избыточности хранящейся информации для обеспечения независимого расчета ячейки. Маркировка применяется при расчете смежных величин, определенных на грани, ребре или узле и принадлежащих сразу нескольким ячейкам. На CPU такая величина в каждой ячейке помечается маркером, обозначающим, что расчет для этой ячейки уже произведен. Перед началом расчета очередной ячейки анализируется маркер соседней ячейки, и, если он находится в состоянии *посчитано*, расчет не проводится, а значение величины берется из соседней ячейки.

На GPU для указанной реализации необходимо прибегать к медленным атомарным функциям, да и не всегда это возможно. В таких случаях приходится несколько видоизменять алгоритм, избавляясь от маркировки, как от дивергентного ветвления. Если для расчета маркиру-

емой величины требуются небольшие вычисления, то они выполняются для каждой ячейки со смежной величиной и одна и та же работа дублируется на разных GPU-нитях. Если для расчета маркируемой величины требуются значительные вычисления, то одно GPU-ядро заменяется двумя ядрами. В первом ядре для всех ячеек рассчитывается смежная величина только на одной фиксированной по расположению грани (ребре, узле) ячейки. Во втором ядре копируется уже посчитанная информация от соседней ячейки.

На рис. 3 показан схематичный пример последовательности расчета скорости в узлах двумерной сетки. Приведен простейший случай фрагмента сетки, где ни одна из представленных ячеек не находится на границе расчетной области. Закрашенным кружком помечены посчитанные значения величины, незакрашенный кружок означает, что значение данной величины в текущей ячейке скопировано из соседней ячейки.

Использование нескольких GPU

Для возможности использования при расчете задачи нескольких устройств GPU в код внесена привязка MPI-процесса к GPU. Остальные MPI-процессы ничего об этом устройстве "не знают". По сути при такой реализации GPU является сопроцессором для MPI-процесса. Такой подход позволяет оставить неизменным алгоритм MPI-распараллеливания, реализованный на CPU. Вследствие этого без дополнительных трудозатрат появляется возможность производить расчеты в гетерогенном режиме, т. е. с одновременным использованием всех доступных CPU- и GPU-ресурсов для одного исполняемого файла.

Для экономии трудозатрат авторы используют существующие в ЭГИДА-ТЕСТ программы обмена, выполняемые при вызове типовой схемы. Таким образом, реализован способ обмена информацией через CPU с помощью MPI-интерфейса типовой схемы ЭГИДА-ТЕСТ, в которой отключен расчет ячеек. Для корректности такого обмена необходимо обеспечить наличие данных в граничных ячейках на CPU.

До и после вызова типовой схемы производится копирование данных граничных точек по каналам GPU—CPU с использованием промежуточного массива. Размер промежуточного массива определяется как максимум из количества граничных и *внешнепроцессорных* ячеек (типы B (boundary) и E (external) соответственно [8]). Ячейки типа E являются фиктивными ячейками, точными копиями ячеек типа B с других MPI-процессов. На рис. 4 ячейки типа B выделены жирными линиями, типа E — пунктиром. Для наглядности на рисунке представлена двумерная сетка, а не трехмерная, используемая в ЭГИДА-ТЕСТ.

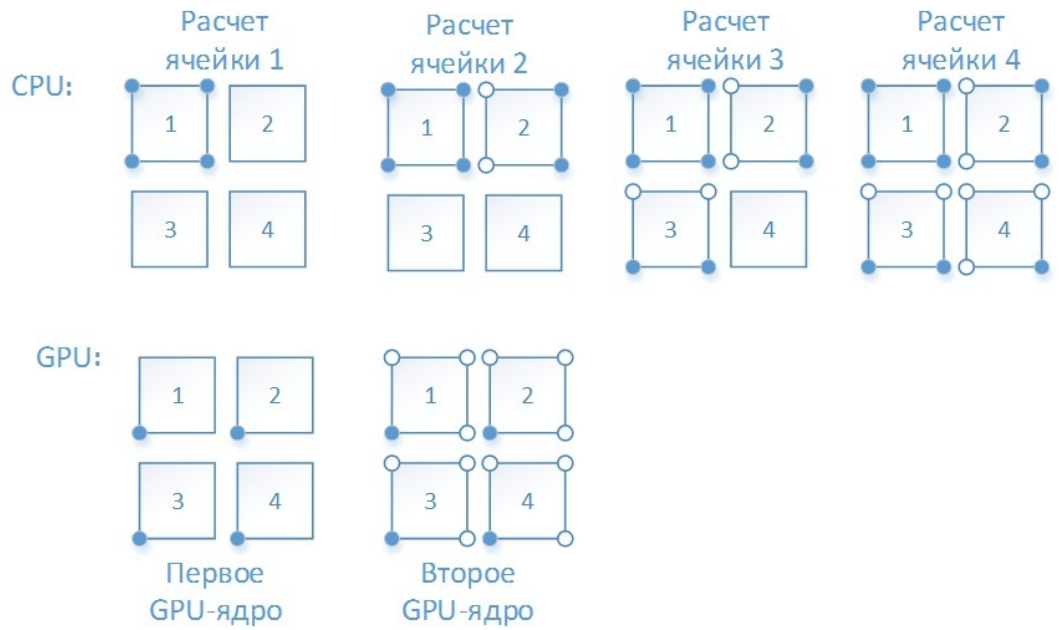

Рис. 3. Последовательности расчета скорости на CPU и GPU

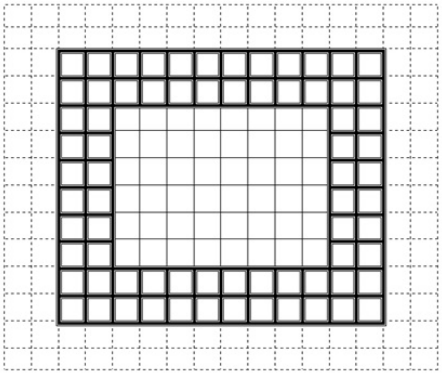

Рис. 4. Типы ячеек пространственной сетки

Для осуществления обмена значениями величин в граничных ячейках выполняются следующие действия:

1. Запуск ядра GPU, собирающего эти значения из граничных ячеек в промежуточный массив.
2. Передача данных промежуточного массива с GPU на CPU.
3. Выполнение программы, копирующей данные из промежуточного массива в реальные места хранения этой информации в системе памяти ЭГИДА-ТЕСТ на CPU.
4. Работа типовой схемы по MPI-обмену без вызова расчетных функций.
5. Выполнение программы, копирующей данные, полученные с других MPI-процессов и находящиеся в системе памяти ЭГИДА-ТЕСТ, в промежуточный массив на CPU.
6. Передача данных промежуточного массива с CPU на GPU.
7. Запуск ядра GPU, "раскладывающего" новые значения величин из промежуточного массива в граничные ячейки на GPU.

На рис. 5 показана используемая в ЭГИДА-ТЕСТ-GPU схема MPI-обменов.

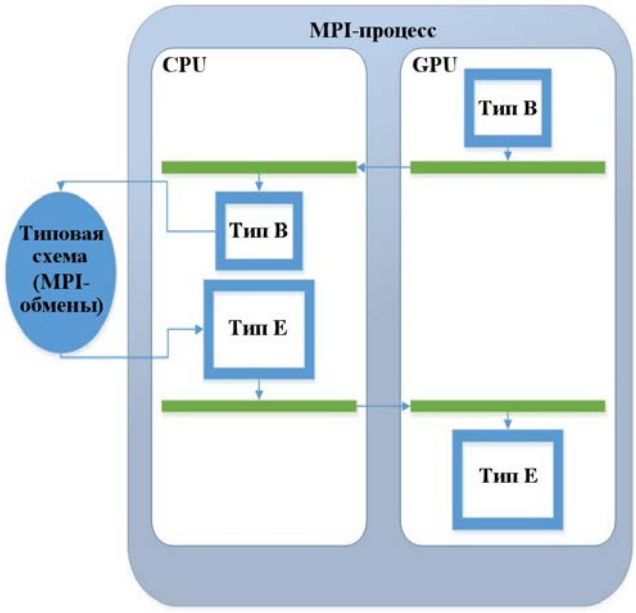

Рис. 5. Схема MPI-обменов

Реализация адаптивно-встраиваемой дробной сетки на GPU

Для повышения точности расчетов на неподвижной сетке и экономии ресурсов в методике ЭГАК активно используется адаптивно-встраиваемая дробная сетка (далее просто *дробная сетка*). Она вводится либо в начальный момент времени с привязкой к пространству или веществу, либо в силу срабатывания некоторых условий или критериев.

Применение дробной сетки означает, что в процессе счета в конце каждого временного шага любая счетная ячейка независимо от других может быть раздроблена на восемь частей. Вновь полученные ячейки также могут быть раздроблены вплоть до пятого уровня дробления (программное ограничение, связанное с выделением памяти). При необходимости ранее полученные дробные ячейки могут удаляться. Вычисления выполняются на полученной нерегулярной сетке до следующего изменения ее состояния.

Расчет каждого этапа ведется в цикле по уровням дробления, начиная от самых мелких ячеек. Ячейки основной, регулярной, сетки называются ячейками нулевого уровня дробления.

Для того чтобы была возможность в процессе счета создавать новые ячейки, в ЭГИДА-ТЕСТ существует список номеров ячеек, участвующих в расчете в данный момент. Номера ячеек, которые в данный момент в расчете не участвуют, хранятся в стеке свободных ячеек. Создание и удаление ячеек в конце счетного шага подразумевает: – проверку каждой ячейки на соответствие критериям дробления; – изменение списков ячеек, участвующих в счете; – установление соседства между ячейками как одного, так и разных уровней дробления; – заполнение вновь созданных ячеек физическими данными.

Работа по изменению списков ячеек, участвующих в счете, не является параллельной. Вообще, как правило, работы по формированию раз-

личных списков плохо адаптируются на GPU. Поэтому в данном случае авторы отошли от парадигмы адаптации всего кода на GPU. Проведен поиск "горячих пятен" в программах создания и удаления дробных ячеек. Такой оказалась только одна программа, которая устанавливает признаки дробления для ячеек, попадающих в переходную зону дробления с заданной шириной. *Ширина переходной зоны дробления* задается исполнителем расчета в количестве ячеек и определяет радиус шаров, центрами которых являются все дробные ячейки. Нераздробленные ячейки, попадающие в объединение всех таких шаров, дробятся и формируют переходную зону дробления.

На рис. 6 показана схема выполнения программы при использовании дробной сетки. Последним этапом является заполнение вновь созданных ячеек физическими данными. Этот этап не занимает много времени, но использует большое количество информации о физическом состоянии рассчитываемой системы. Данная информация в актуальном состоянии хранится на GPU. Поэтому в целях минимизации передаваемой информация по каналу CPU—GPU работа по заполнению физическими данными (усреднением от соседей) вновь созданных ячеек выполняется на GPU.

Постановка тестовых задач

Тест 1. Движение сферической системы. Данный тест представляет собой трехмерную задачу, в которой двенадцать веществ находятся в сферических слоях толщиной 1 см, имеющих общий центр [10].

Тест 2. Седовский (точечный) взрыв. Данный тест является трехмерной задачей о сферически-симметричном взрыве. Постановка задачи взята из работы [11].

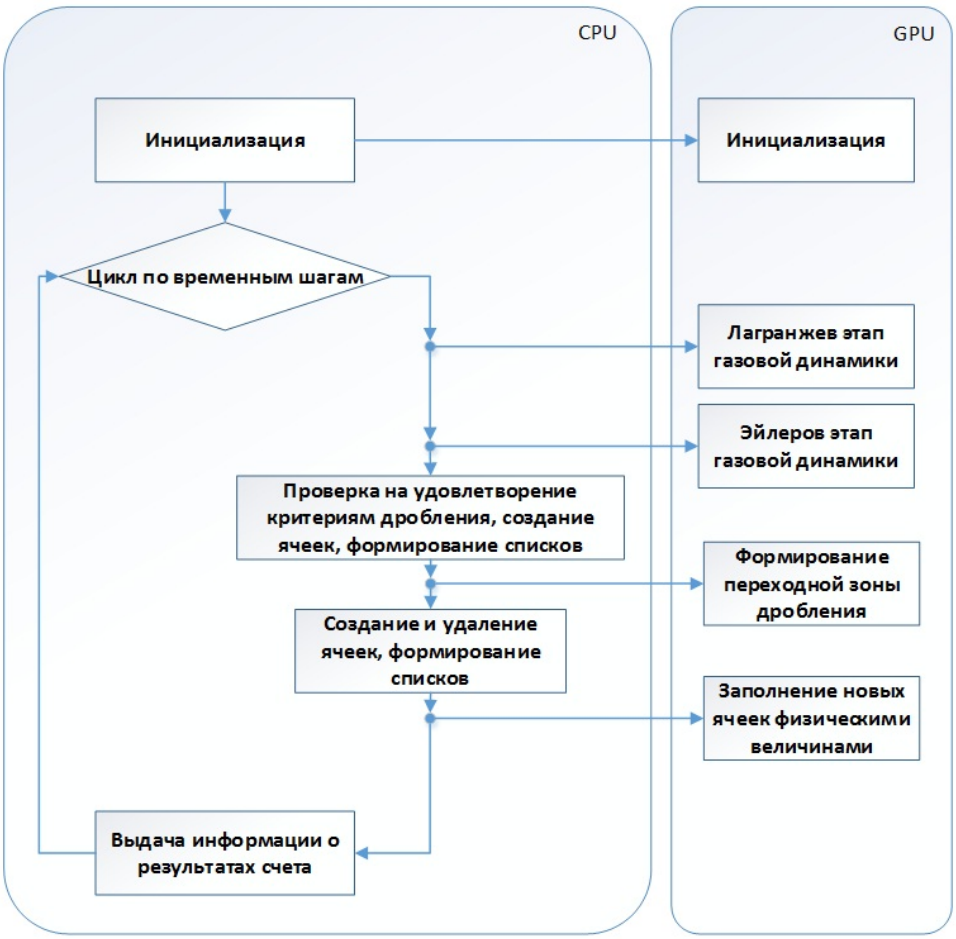

Рис. 6. Схема выполнения программы при использовании дробной сетки

Тест 2 характеризуется малой областью возмущения. Тем самым он является хорошим примером необходимости использования дробной сетки.

Использование дробной сетки в расчете определяется наличием критериев дробления. Наиболее часто в качестве критерия дробления используется номер вещества: ячейки, содержащие вещество с этим номером, дробятся. Кроме того, вокруг уже раздробленных ячеек создается переходная зона дробления с заданной шириной. В тесте 1 дробятся ячейки, содержащие вещества 1, 2 и 3, в тесте 2 — ячейки, содержащие возмущенное вещество.

Количество ячеек нулевого уровня дробления задается при старте задачи. Количество ячеек других уровней дробления в задачах варьировалось с помощью ширины переходной зоны дробления.

Результаты тестирования

Тестирование проводилось на гибридной ЭВМ, содержащей два процессора Intel Xeon CPU E5-2690 v4 2.6 GHz и два графических ускорителя Nvidia Tesla V100. Наилучший режим загрузки для программы ЭГИДА-ТЕСТ без использования ускорителей достигается с двумя MPI-процессами и двадцатью восемью OpenMP-нитями (MPI×OMP = 2 × 28). Данный режим использовался для всех тестов, рассчитываемых без применения GPU.

При использовании двух GPU запуски проводились в таком же режиме. Заметим, что OMP-распараллеливание необходимо, так как процесс создания и удаления дробных ячеек рассчитывается на GPU не полностью.

Полученные результаты расчетов всех вариантов тестов по программам ЭГИДА-ТЕСТ и ЭГИДА-ТЕСТ-GPU в пределах ста временных шагов полностью между собой согласуются.

Далее представлены результаты тестирования различных вариантов тестов 1 и 2. Показательным является среднее время выполнения второго временного шага в расчете (на первом шаге часть времени занимает создание и обмен начальной информации).

В табл. 1 показаны результаты выполнения теста 1 на двух CPU, а также одном и двух GPU. На CPU запуск теста производится по программе ЭГИДА-ТЕСТ без использования GPU, он и является отправной точкой при расчете ускорения программы. Показанные в таблице результаты на двух GPU получены при запуске теста 1 в двух режимах масштабирования — деления (на сетке $150 \times 100 \times 100$) и умножения (на сетках $300 \times 100 \times 100$ и $150 \times 200 \times 100$). При использовании режима умножения в обоих случаях пространственная декомпозиция выполнялась вдоль первого пространственного направления. Приведенные в табл. 1 времена соответствуют выполнению лагранжева и эйлерова этапов процесса газовой динамики, этапа создания и удаления дробных ячеек, а также выполнению временного шага.

Общее количество ячеек в задаче немного варьируется в зависимости от декомпозиции, так как учитываются все используемые ячейки (в том числе типы B и E). Основным показателем расчета является количество ячеек (в млн), посчитанных за секунду (R). Исходя из этого по-

Таблица 1

Результаты выполнения теста 1

Параметры выполнения	Количество ячеек		Время, мс				R, млн/с	S	E, %
	0-й уровень	1-й уровень	Лагр. этап	Эйлер. этап	Дроб. ячейки	Временной шаг			
$150 \times 100 \times 100$:									
2 CPU	1 540 000	111 232	160	686	333	1 179	1,4	–	–
1 GPU	1 500 000	103 664	53	277	194	524	3,1	2,2	–
2 GPU	1 540 000	111 232	45	248	111	404	4,1	2,9	66
$300 \times 100 \times 100$:									
2 GPU	3 040 000	177 120	71	379	211	661	4,9	3,5	79
$150 \times 200 \times 100$:									
2 GPU	3 080 000	192 832	89	478	219	786	4,2	3	68

казателя считаются ускорение (S) и эффективность (E).

Аналогичные результаты для теста 2 приведены в табл. 2.

Пиковая производительность гибридной ЭВМ без учета GPU составляет $\sim$ 1,2 Тфлопс, а пиковая производительность используемого GPU $\sim$ 7,1 Тфлопс. Таким образом, ускорение от использования одного устройства GPU (2,2—2,7) приблизительно в два раза проигрывает отношению пиковых производительностей (7,1/1,2 = = 5,9 раз). Такие показатели являются удовлетворительным результатом.

Однако ускорение от использования двух устройств GPU (2,9—4,4) составляет лишь $\sim$ 25 % от соотношения пиковых производительностей (14,2/1,2 = 11,8), что подтверждается значениями эффективности использования двух GPU (66—81 %).

В табл. 3, 4 представлены суммарные времена полезной арифметической нагрузки (kernel), передачи данных между CPU и GPU (CPU—GPU) и MPI-обменов (MPI) на разных этапах расчета тестов 1 и 2, а также их доли от общего времени расчета этапов. На этапе создания и удаления дробных ячеек производится некоторая работа на CPU, ее время учтено вместе с MPI (CPU+ +MPI).

В табл. 3, 4 явно не показано, но просматривается, что эффективность распараллеливания полезной нагрузки при переходе от одного GPU к двум в среднем находится на уровне 90 %. Проблему составляет низкая общая эффективность при передаче информации как по каналам CPU—GPU, так и между MPI-процессами. Такие передачи занимают до 50 % времени выполнения программы. Необходима дальнейшая работа для достижения асинхронности полезной нагрузки и

Таблица 2

Результаты выполнения теста 2

Параметры выполнения	Количество ячеек		Время, мс				R, млн/с	S	E, %
	0-й уровень	1-й уровень	Лагр. этап	Эйлер. этап	Дроб. ячейки	Временной шаг			
150 × 150 × 150:									
2 CPU	3 456 000	2 512	286	689	761	1 736	2	–	–
1 GPU	3 375 000	2 512	99	141	384	624	5,4	2,7	–
2 GPU	3 465 000	2 512	78	174	209	461	7,5	3,8	70
300 × 150 × 150:									
2 GPU	6 840 000	2 512	127	240	404	771	8,9	4,4	81
150 × 300 × 150:									
2 GPU	6 930 000	2 512	154	346	416	916	7,6	3,7	69

Таблица 3

Время и доля выполнения операций для теста 1 при использовании GPU

Этап	Тип операции	1 GPU	2 GPU					
		150 × 100 × 100	150 × 100 × 100		300 × 100 × 100		150 × 200 × 100	
		Время, мс	Время, мс	Доля, %	Время, мс	Доля, %	Время, мс	Доля, %
Лагранжев	kernel	53	28	62	54	76	54	61
	CPU—GPU		12	27	12	17	26	29
	MPI		5	11	5	7	9	10
Эйлеров	kernel	277	161	65	292	77	304	63
	CPU—GPU		66	27	66	17	132	28
	MPI		21	8	21	6	42	9
Дробные	kernel	100	53	48	103	49	105	48
ячейки	CPU—GPU	65	32	29	65	31	65	30
	CPU+MPI	29	26	23	43	20	49	22

Таблица 4

Время и доля выполнения операций для теста 2 при использовании GPU

Этап	Тип операции	1 GPU	2 GPU					
		150 × 150 × 150	150 × 150 × 150		300 × 150 × 150		150 × 300 × 150	
		Время, мс	Время, мс	Доля, %	Время, мс	Доля, %	Время, мс	Доля, %
Лагранжев	kernel	99	53	68	102	80	106	69
	CPU—GPU		18	23	18	14	34	22
	MPI		7	9	7	6	14	9
Эйлеров	kernel	141	93	54	159	66	187	54
	CPU—GPU		60	34	60	25	114	33
	MPI		21	12	21	9	45	13
Дробные ячейки	kernel	226	118	56	232	57	237	57
	CPU—GPU	85	41	20	85	21	85	20
	CPU+MPI	73	50	24	87	22	94	23

обменов, что влечет за собой разработку новой или совершенствование старой типовой схемы в программе ЭГИДА-ТЕСТ-GPU.

Стоит отметить, что на этапе создания и удаления дробных ячеек практически не используется типовая схема. Обмен CPU—GPU здесь является необходимым для осуществления работы со списками на CPU. Такой обмен присутствует даже при использовании одного GPU. Поэтому эффективность распараллеливания на двух GPU данного этапа находится на приемлемом уровне ($\sim 90\,\%$).

Перед выполнением дальнейших работ по оптимизации использования нескольких GPU в расчете необходимо иметь результаты "в нулевом приближении", за которые можно принять результаты первой версии работающей программы ЭГИДА-ТЕСТ-GPU. Такие результаты представлены в табл. 5, 6 для разных размеров сеток. В таблицах указаны размеры счетных сеток нулевого уровня. Ширина переходной зоны дробления подбиралась таким образом, чтобы общее число ячеек в вариантах задачи приблизительно совпадало. Данный параметр сильно влияет на время выполнения этапа создания и удаления дробных ячеек. Также указано количество ячеек на каждом из используемых уровней дробления и общее количество ячеек на всех уровнях дробления. По этому параметру можно понять, какой максимальный уровень дробления использовался в данном варианте теста. Значения ускорений, приведенные в табл. 6, получены путем отношения времени выполнения в режиме CPU (без использования GPU) ко времени выполнения с использованием двух GPU.

Использование GPU позволило получить на тесте 1 общее ускорение от 3 до 5 раз, на тесте 2 — от 4 до 6 раз. Данные результаты яв-

Таблица 5

Параметры сеток для тестов 1 и 2

Сетка	Ширина переход. зоны	Кол-во ячеек				
		0-й уровень	1-й уровень	2-й уровень	3-й уровень	Всего
Тест 1:						
226 × 160 × 160	–	5 785 600	–	–	–	5 785 600
210 × 156 × 156	16	5 207 904	616 608	–	–	5 824 512
194 × 140 × 140	6	3 880 800	247 440	1 659 648	–	5 787 888
Тест 2:						
200 × 200 × 200	–	8 000 000	–	–	–	8 000 000
198 × 198 × 198	16	7 919 208	26 560	–	–	7 945 768
196 × 196 × 196	16	7 683 200	33 360	212 480	–	7 929 040
190 × 190 × 190	12	7 003 400	17 040	118 696	819 712	7 958 848

Таблица 6

Ускорения при выполнении этапов тестов 1 и 2

Сетка	Лагранжев этап	Эйлеров этап	Дробные ячейки	Временной шаг
Тест 1:				
$226 \times 160 \times 160$	4,4	3,2	–	3,4
$210 \times 156 \times 156$	4,1	3,2	5,8	5,2
$194 \times 140 \times 140$	3,5	2,9	2,3	2,8
Тест 2:				
$200 \times 200 \times 200$	3,9	4,4	–	4,2
$198 \times 198 \times 198$	4,2	4,8	5,9	5,8
$196 \times 196 \times 196$	4,1	4,7	5,7	5,5
$190 \times 190 \times 190$	4,5	5,2	4,9	4,9

ляются удовлетворительными, так как соотношение между пиковой производительностью используемых устройств составляет около 12 раз.

Гетерогенный режим загрузки счетного узла

Схема MPI-обменов, используемая в ЭГИДА-ТЕСТ-GPU, позволяет вести счет в режиме, когда один или несколько MPI-процессов работают каждый со своим GPU, а другие выполняются только на CPU. То есть для расчета задачи можно использовать все счетные ресурсы узла в рамках одного исполняемого файла. Для этого в программу ЭГИДА-ТЕСТ внедрена статическая балансировка по первому пространственному направлению. В итоге получилась рабочая версия кода, позволяющая вести счет в режиме (MPI× ×OpenMP)+(MPI×CUDA).

В течение разработки программы ЭГИДА-ТЕСТ-GPU гетерогенный режим загрузки поддерживается в работоспособном состоянии, но не развивается и не оптимизируется: основной целью исследования является эффективное освоение только нескольких устройств GPU.

Счет в гетерогенном режиме загрузки исследовался на тесте 1 с размером счетной сетки $226 \times 160 \times 160$ без использования дробной сетки: исследования проводились сразу после введения возможности счета с использованием нескольких MPI-потоков, когда возможность использования дробной сетки еще не была реализована.

Так как устройства GPU значительно мощнее ядер CPU и тестирование проводилось на ЭВМ, содержащей только два GPU, то оптимальный вариант декомпозиции решаемой задачи можно представить в виде двух столбиков вдоль любого (выбрано первое) пространственного направления, в основании которых находятся MPI-процессы с GPU (рис. 7).

Такой вид декомпозиции позволяет балансировать счетную нагрузку MPI-процессов только вдоль первого пространственного направления (вдоль второго направления нагрузка всегда делится на две части поровну, вдоль третьего направления нет разбиения) за счет распределения *слоев* ячеек (или, другими словами, количества ячеек вдоль первого пространственного направления) на MPI-процессах. Количество слоев ячеек на всех MPI-процессах, работающих с CPU, совпадает между собой (аналогично на MPI-процессах, работающих с GPU).

В табл. 7 приведены средние времена выполнения временного шага для задачи с разными

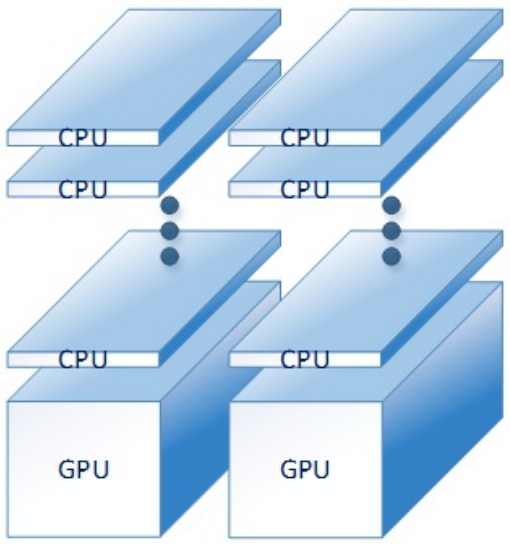

Рис. 7. Декомпозиция решаемой задачи

Таблица 7

Среднее время (в мс) выполнения одного шага при различных режимах загрузки сервера

MPI × OMP	Количество слоев ячеек					
	4	6	8	10	12	14
4 × 6	1 022	957	940	933	989	–
4 × 14	1 048	989	981	970	1 016	–
6 × 2	991	1 040	1 011	1 089	1 108	1 207
6 × 4	953	999	926	966	**904**	952
6 × 5	960	996	926	969	**897**	961
6 × 6	951	996	993	961	961	945
6 × 7	943	995	924	963	**889**	982
8 × 3	1 020	932	920	909	966	–
8 × 4	1 012	932	920	**896**	947	–
8 × 5	1 007	932	907	**883**	917	–
8 × 6	1 059	992	961	939	960	–
16 × 2	1 074	1 163	1 344	1 456	–	–

параметрами. Во всех указанных вариантах два MPI-процесса работают с GPU, остальные выполняются на CPU. Результаты представлены для разного количества слоев ячеек (от 4 до 14), расчет которых производится на каждом MPI-процессе, содержащем CPU.

Время счета шага с применением только двух устройств GPU на момент тестирования составляло 906 мс. При гетерогенном режиме получено несколько комбинаций, при которых время счета, хотя и незначительно, но меньше (в табл. 7 выделено жирным шрифтом). Тем самым показана принципиальная возможность получения выигрыша от гетерогенного режима загрузки, когда задействуются все вычислительные возможности устройств на узле, хотя на текущий момент вычисления по программе ЭГИДА-ТЕСТ-GPU эффективнее вести только на GPU.

Заключение

Приведено описание исследований и их результатов по адаптации к счету на гибридной ЭВМ, содержащей GPU, программы ЭГИДА-ТЕСТ. Данная программа реализует алгоритмы лагранжева и эйлерова этапов газовой динамики методики ЭГАК с возможностью ведения счета на адаптивно-встраиваемой дробной сетке.

Результатом работ стала первая версия программы ЭГИДА-ТЕСТ-GPU, позволяющая вести счет задач газовой динамики на адаптивно-встраиваемой дробной сетке с применением GPU. Получено хорошее согласие результатов расчета тестовых задач на гибридной параллельной ЭВМ с результатами, полученными на многопроцессорной ЭВМ (без GPU).

При использовании двух устройств GPU на разных тестовых задачах получено ускорение счета от 3 до 6 раз по сравнению со счетным узлом с двумя CPU.

Анализ полученных результатов показывает необходимость продолжения работы по оптимизации и повышению эффективности при использовании нескольких GPU, например, за счет прямых обменов между устройствами. Кроме того, по мнению авторов, следует сосредоточиться на адаптации к счету на GPU программ теплопроводности методики ЭГАК с применением дробных сеток.

Исследование выполнено в рамках научной программы Национального центра физики и математики г. Сарова, направление № 1 "Национальный центр исследования архитектур суперкомпьютеров".

Список литературы

1. *Быков А. Н., Гордеев Д. Г., Куделькин В. Г., Сизов Е. А., Фёдоров А. А.* Методика РАМЗЕС-КП на гибридных параллельных ЭВМ с графическими ускорителями // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2017. Вып. 3. С. 70—76.

Bykov A. N., Gordeev D. G., Kudelkin V. G., Sizov E. A., Fyedorov A. A. Metodika RAMZES-KP na gibridnykh parallelnykh EVM s graficheskimi uskoritelyami // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskukh protsessov. 2017. Vyp. 3. S. 70—76.

2. *Геллер О. В., Васильев М. О., Холодов Я. А.* Построение высокопроизводительного вычислительного комплекса для моделирования задач газовой динамики // Компьютерные исследования и моделирование. 2010. Т. 2, № 3. С. 309—317.
   Geller O. V., Vasilyev M. O., Kholodov Ya. A. Postroenie vysokoproizvoditelnogo kompleksa dlya modelirovaniya zadach gazovoy dinamiki // Kompyuternye issledovaniya i modelirovanie. 2010. T. 2, № 3. S. 309—317.
3. *Анисов В. О., Вазиев Э. М., Ушаков Д. А.* Реализация метода решения двумерного уравнения теплопроводности на гибридной архитектуре (CPU+GPU) // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2021. Вып. 1. С. 39—52.
   Anisov V. O., Vaziev E. M., Ushakov D. A. Realizatsiya metoda resheniya dvumernogo uravneniya teploprovodnosti na gibridnoy arkhitekture (CPU+GPU) // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskukh protsessov. 2021. Vyp. 1. S. 39—52.
4. *Боресков А. В., Харламов А. А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2010.
   Boreskov A. V., Kharlamov A. A. Osnovy raboty s tekhnologiey CUDA. M.: DMK Press, 2010.
5. *Шанин А. А., Янилкин Ю. В.* Комплекс программ ЭГАК. Газодинамические разностные схемы в эйлеровых переменных // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 1993. Вып. 1. С. 24—30.
   Shanin A. A., Yanilkin Yu. V. Kompleks programm EGAK. Gazodinamicheskie raznostnye skhemy v eylerovykh peremennykh // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskukh protsessov. 1993. Vyp. 1. S. 24—30.
6. *Янилкин Ю. В., Беляев С. П., Городничев А. В., Воронов Е. Г., Гужова А. Р., Дегтяренко Л. И., Жарова Г. В., Кучерова П. А., Стадник А. Л., Ховрин Н. А.* Комплекс программ ЭГАК++ для моделирования на адаптивно-встраивающейся дробной счетной сетке // Вопросы атомной науки и техники. Сер. Методики и программы численного решения задач математической физики. 2003. Вып. 1. С. 20—28.
   Yanilkin Yu. V., Belyaev S. P., Gorodnichev A. V., Voronov E. G., Guzhova A. P., Degtyarenko L. I., Zharova G. V., Kucherova P. A., Stadnik A. L., Khovrin N. A. Kompleks programm EGAK++ dlya modelirovaniya na adaptivno-vstraivayushcheysya drobnoy raschyetnoy setke // Voprosy atomnoy nauki i tekhniki. Ser. Metodiki i programmy chislennogo resheniya zadach matematicheskoy fiziki. 2003. Vyp. 1. S. 20—28.
7. *Алексеев А. В., Беляев С. П., Бочков А. И., Быков А. Н., Ветчинников М. В., Залялов А. Н., Нуждин А. А., Огнев С. П., Самсонова Н. С., Чистякова И. Н., Янилкин Ю. В.* Методические прикладные тесты РФЯЦ-ВНИИЭФ для численного исследования параметров высокопроизводительных систем // Там же. Сер. Математическое моделирование физических процессов. 2020. Вып. 2. С. 86—100.
   Alekseev A. V., Belyaev S. P., Bochkov A. I., Bykov A. N., Vetchinnikov M. V., Zalyalov A. N., Nuzhdin A. A., Ognev S. P., Samsonova N. S., Chistyakova I. N., Yanilkin Yu. V. Metodicheskie prikladnye testy RFYaTs-VNIIEF dlya chislennogo issledovaniya parametrov vysokoproizvoditelnykh system // Tam zhe. Ser. Matematicheskoe modelirovanie fizicheskukh protsessov. 2020. Vyp. 2. S. 86—100.
8. *Беляев С. П.* Метод мелкозернистого распараллеливания с динамической балансировкой на примере задачи газовой динамики и вычислительные эксперименты на параллельной системе // Там же. 2000. Вып. 1. С. 45—49.
   Belyaev S. P. Metod melkozernistogo rasparallelivaniya s dinamicheskoy balansirovkoy zadachi gazovoy dinamiki i vychislitelnye eksperimenty na parallelnoy sisteme // Tam zhe. 2000. Vyp. 1. S. 45—49.
9. *Колобянин В. Ю., Фёдоров А. А., Антипина Н. Р.* Двухуровневое распараллели-

вание явных разностных схем методики ЭГАК // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2017. Вып. 3. С. 62—69.
Kolobyanin V. Yu., Fyedorov A. A., Antipina N. R. Dvukhurovnevoe rasparallelivanie yavnykh raznostnykh skhem metodiki EGAK // Voprosy atomnoy nauki i tekhniki. Ser. Matematicheskoe modelirovanie fizicheskukh protsessov. 2017. Vyp. 3. S. 62—69.

10. *Колобянин В. Ю., Чистякова И. Н.* Программа ЭГИДА-ТЕСТ для численного исследования параметров высокопроизводительных вычислительных систем // Там же. 2023. Вып. 1. С. 42—50.
Kolobyanin V. Yu., Chistyakova I. N. Programma EGIDA-TEST dlya chislennogo issledovaniya parametrov vysokoproizvoditelnykh vychislitelnykh system // Tam zhe. 2023. Vyp. 1. S. 42—50.
11. *Седов Л. И.* Методы подобия и размерности в механике. М.: Наука, 1965.
Sedov L. I. Metody podobiya i razmernosti v mekhanike. M.: Nauka, 1965.

Статья поступила в редакцию 22.08.22.