УДК 519.6

ИСПОЛЬЗОВАНИЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ ПЛАТФОРМ В ЗАДАЧАХ ТОМОГРАФИЧЕСКОЙ ЦИФРОВОЙ ТРАССЕРНОЙ ВИЗУАЛИЗАЦИИ

В. А. Ложкин[1], **Ю. А. Ложкин**[1], **М. П. Токарев**[1]

Томографический метод цифровой трассерной визуализации (Tomo PIV; Tomographic Particle Image Velocimetry) — это новый метод изучения газовых и жидкостных потоков, позволяющий измерять трехмерные распределения скорости и, на их основе, рассчитывать другие характеристики потока в заданном объеме. Одной из основных сложностей в использовании этого метода измерений является высокая ресурсоемкость используемых алгоритмов обработки данных. В настоящей статье рассматриваются способы увеличения производительности метода за счет использования различных высокопроизводительных вычислительных платформ, таких как многопроцессорные серверы, вычислительные кластеры и графические процессорные устройства (ГПУ). Описаны проблемы, возникшие при переносе алгоритма томографической реконструкции и корреляционного алгоритма расчета полей скорости на платформы с множеством вычислительных ядер, а также способы решения этих проблем. Рассматриваются особенности переноса алгоритма томографической реконструкции на графический ускоритель с применением технологии OpenCL. Приводится оценка времени обработки одного эксперимента с использованием различных вычислительных платформ, а также целесообразности использования той или иной платформы в терминах условной стоимости обработки одного эксперимента. Работа выполнена при поддержке Министерства образования и науки РФ в рамках реализации Федеральной целевой программы “Научные и научно-педагогические кадры инновационной России” на 2009–2013 гг.

Ключевые слова: томография, цифровая трассерная визуализация, Tomo PIV, графические ускорители, вычислительный кластер, OpenCL, MPI, высокопроизводительные вычисления.

1. Введение. Томографический метод цифровой трассерной визуализации (Tomographic Particle Image Velocimetry, Tomo PIV) [1] используется для измерения скорости в объеме потока жидкости или газа в гидроаэродинамическом эксперименте. В отличие от стереоскопического метода цифровой трассерной визуализации Stereo PIV, являющегося в настоящее время практически стандартным инструментом диагностики потоков [2, 3], где три компоненты скорости измеряются в плоском сечении потока, метод TomoPIV позволяет измерять мгновенные распределения трех компонент скорости в объеме течения. Принцип измерения скорости подобен традиционным PIV-методам. В исследуемое течение добавляются частицы-трассеры, поток дважды, с контролируемой временно́й задержкой, освещается лазерным импульсом, но при этом область освещения не является локализованной в плоскости. Рассеянный частицами свет регистрируется на три или более цифровые камеры, ориентированные под различными направлениями к области измерения. На фотоматрицу цифровой камеры проецируется распределение света, рассеянного частицами в данном направлении. Положения трассеров в объеме потока восстанавливаются методами томографической реконструкции по полученным изображениям частиц с использованием информации о пространственной калибровке камер. Мгновенная скорость течения определяется по смещениям частиц за время между вспышками лазера, которые рассчитываются посредством корреляции реконструированных объемных изображений трассеров между собой [4].

Одним из основных недостатков метода Tomo PIV является высокая ресурсоемкость вычислительных процедур, которые используются для обработки данных. Для получения одного трехмерного поля скорости на основе исходных экспериментальных данных с использованием “офисного” персонального компьютера требуется примерно один час. Один типичный эксперимент содержит две тысячи измерений, что приводит к двум месяцам непрерывных вычислений, а это значительно ограничивает область применения метода. Ресурсоемкими операциями являются как расчет трехмерных распределений трассеров,

[1] Институт теплофизики СО РАН (ИТ СО РАН), просп. акад. Лаврентьева, 1, 630090, г. Новосибирск; В. А. Ложкин, лаборант, e-mail: v.a.lozhkin@gmail.com, Ю. А. Ложкин, инженер, e-mail: lozhkin@itp.nsc.ru, М. П. Токарев, науч. сотр., e-mail: mtokarev@itp.nsc.ru

так и построение итоговых трехмерных полей скорости. Корреляционные алгоритмы расчета полей скорости могут быть легко модифицированы для использования ресурсов многоядерных процессоров, что позволяет увеличивать производительность процедур расчета трехмерных полей скорости. С другой стороны, задачи малоракурсной вычислительной томографии гораздо меньше подходят для оптимизации под многоядерные процессорные архитектуры. Актуальной задачей является снижение общего времени обработки данных методом Tomo PIV за счет использования вычислительных ресурсов различных высокопроизводительных систем, таких как вычислительные кластеры, многопроцессорные серверные системы и графические процессорные устройства (ГПУ).

2. Алгоритмы обработки изображений в методе Tomo PIV. Основным шагом алгоритма построения поля скорости в методе Tomo PIV является восстановление трехмерного распределения интенсивности по нескольким двумерным проекциям алгоритмом итерационной алгебраической реконструкции (рис. 1).

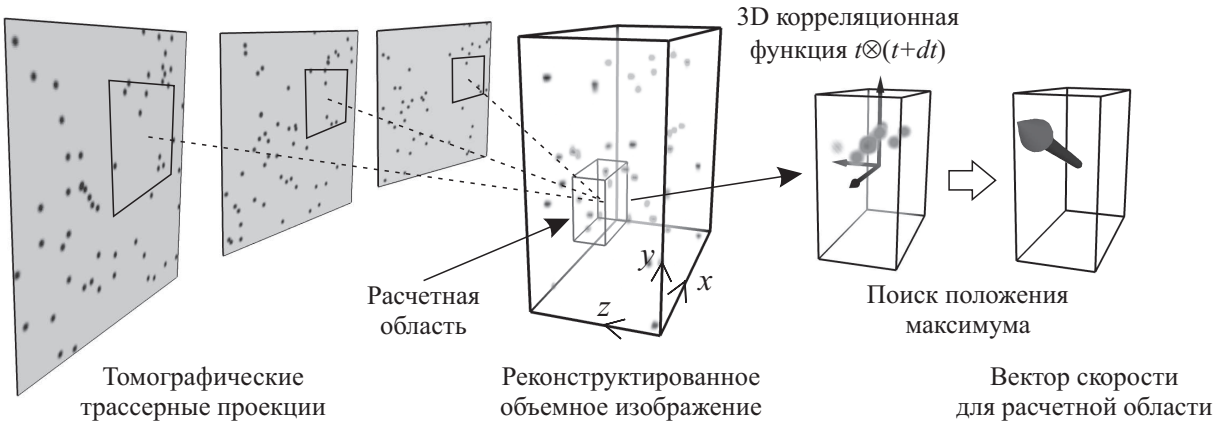


Рис. 1. Принцип метода Tomo PIV

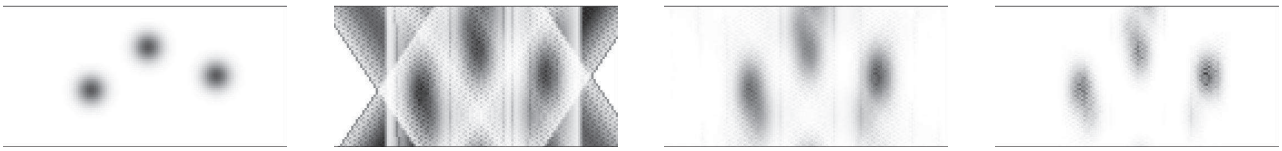
Задача реконструкции сводится к задаче решения недоопределенной системы линейных алгебраических уравнений (СЛАУ), для решения которой известно несколько итерационных методов. Одним из эффективных алгоритмов для задачи восстановления объемной интенсивности света, рассеянного частицами, по скорости оценки и точности реконструкции на сегодняшний день является алгоритм SMART (Simultaneous Multiplicative Algebraic Reconstruction Technique) [5]. Принцип итерационного алгебраического метода томографической реконструкции заключается в итерационном уточнении интенсивности в каждом элементе (вокселе) реконструируемого объема f_j^{k+1} с учетом интенсивности каждого пикселя p_i на зарегистрированных изображениях-проекциях:

$$W_{ij}\, f_j = p_i, \quad f_j^{k+1} = f_j^k \sqcap_i^{N_j} \left[{p_i}^{\mu W_{ij}} \left(\sum_{l \in L_i} W_{il} f_l^k \right)^{-\mu W_{ij}} \right]^{1/N_j}.$$

Здесь μ — коэффициент релаксации; W_{ij} — разреженная весовая матрица, учитывающая вклад интенсивности j-го вокселя в интенсивность i-го пикселя; N_j — количество пикселей, на которые спроецирован j-й воксель; L_i — множество вокселей, участвующих в формировании изображения i-го пикселя. На рис. 2 в плоском сечении $z = 64$ объемного изображения размером $512\times128\times128$ вокселей представлены модель (а) и результаты ее реконструкции по трем проекциям, зарегистрированным с направлениями ракурса наблюдения ($\theta = 90°$, $\phi = 125°$), ($\theta = 90°$, $\phi = 90°$), ($\theta = 90°$, $\phi = 55°$), после одной (б), пяти (в) и десяти (г) итераций алгоритма. Азимутальный угол наблюдения ϕ отсчитывался между проекцией оптической оси камеры и осью y (рис. 1), полярный угол наблюдения θ отсчитывался между оптической осью камеры и осью x (рис. 1). На рис. 2 размеры частиц специально увеличены для наглядности.

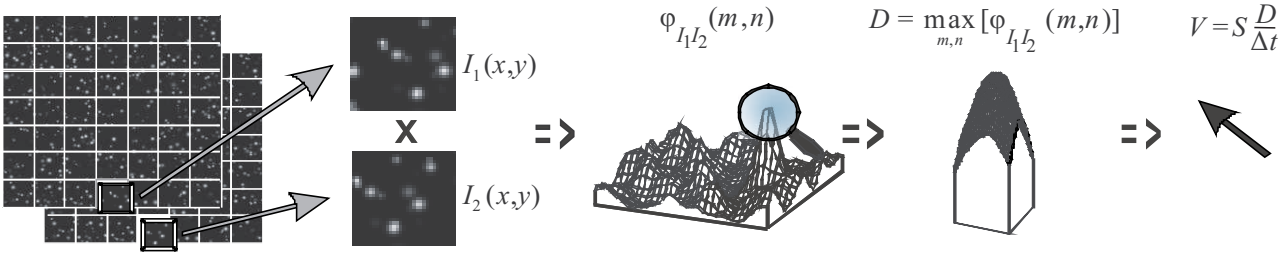
Для расчета трехмерного поля скорости на основе двух последовательных по времени реконструированных трехмерных трассерных картин используются трехмерные корреляционные алгоритмы. Основные принципы алгоритма не отличаются от двумерного случая, поэтому рассмотрим здесь принцип работы метода на примере двумерного стандартного кросс-корреляционного алгоритма [6]. На входе алгоритма есть два изображения образов частиц, полученных в два последовательных момента времени с известной временно́й задержкой между ними. Необходимо получить поле скорости, соответствующее мгновенной скорости потока в измерительной области. Для этого все изображение разбивается на элементарные

ячейки (расчетные области) таким образом, чтобы в каждую расчетную область попало несколько частиц (рис. 3). Среднее смещение частиц внутри элементарной ячейки за время между первым и вторым импульсами лазера может быть найдено путем вычисления функции пространственной корреляции для каждой расчетной области. После вычисления корреляционной функции производится поиск максимального корреляционного пика, положение которого отвечает наиболее вероятному смещению частиц в области. Зная временну́ю задержку между вспышками лазера Δt и рассчитав наиболее вероятное перемещение частиц D в данной элементарной ячейке, можно вычислить скорость: $V = S\,\dfrac{D}{\Delta t}$, где S — масштабный коэффициент для пересчета скорости в м/сек.



а) б) в) г)

Рис. 2. Иллюстрация процесса томографической реконструкции в сечении. Точное распределение интенсивности (а), рассчитанное распределение после 1 итерации SMART (б), после 5 итераций SMART (в), после 10 итераций SMART (г)



а) б) в) г) д)

Рис. 3. Кросс-корреляционный алгоритм вычисления вектора скорости по изображениям частиц в потоке: а) разбиение изображений на соответственные расчетные области, б) расчет корреляционной функции “степени серого” для пары областей, в) поиск максимумов корреляционной функции, г) определение координат максимумов корреляционной функции с подпиксельной точностью, д) расчет вектора скорости

3. Использование многоядерных компьютерных систем. В настоящее время центральные процессоры общего назначения идут по пути увеличения числа процессорных ядер. Многоядерными процессорами оборудуются практически любые компьютерные системы общего назначения от смартфонов до узлов вычислительных кластеров. Поэтому оптимизация алгоритмов для архитектур с большим числом процессорных ядер является обязательным этапом при разработке высокопроизводительных алгоритмов как для вычислительных кластеров, так и для серверов с общей памятью. Для исследования и оптимизации производительности метода Tomo PIV были использованы двухпроцессорный сервер на базе процессоров AMD Opteron 6168 (два 12-ядерных процессора, 16 Гб ОЗУ) и один узел кластера — блейд-сервер HP BL2x220 G6 (два 4-ядерных процессора Intel Xeon E5540, 16 Гб ОЗУ). Первый опыт применения систем с большим числом процессорных ядер для задач томографической трассерной визуализации показал, что для эффективного использования таких вычислительных систем необходима переработка и дополнительная оптимизация кода алгоритмов. При расчетах с использованием большого числа (до 24) вычислительных ядер был выявлен ряд проблем масштабируемости алгоритмов, которые не проявлялись при меньшем числе (до восьми процессорных ядер для расчета одного трехмерного поля скорости).

Целью алгоритма томографической реконструкции является восстановление интенсивности в каждой дискретной точке объема (вокселе) на основе записанных фотокамерами проекций этого распределения и калибровочных уравнений. Из анализа алгоритма можно установить, что интенсивность каждого вокселя восстанавливается независимо от других вокселей объема. В таком случае наиболее эффективным и простым решением будет подвергнуть распараллеливанию процесс обхода массива вокселей. Следует отметить, что, несмотря на независимость расчета значения интенсивности каждого вокселя, между па-

раллельными потоками могут возникать конфликты записи и чтения общей памяти, которых следует избегать (например, используя примитивы синхронизации потоков). Конфликты записи могут происходить в силу того, что разные воксели могут проецироваться на один и тот же пиксель объема. В алгоритме расчета трехмерных полей скорости весь объем разбивается на небольшие области, после чего рассчитывается корреляционная функция двух соответствующих областей последовательных (по времени регистрации) реконструированных трехмерных изображений. В этом случае наиболее ресурсоемкой операцией является расчет корреляционных функций. Расчет вектора скорости в каждой области проходит полностью независимо от других областей. В соответствии с этим распараллеливанию подвергается та часть алгоритма, которая отвечает за расчет корреляционных функций. При этом нет необходимости заботиться о синхронизации потоков, так как конфликты одновременной записи и чтения по одинаковым адресам памяти из параллельных потоков полностью исключены.

В алгоритме томографической реконструкции трехмерного распределения интенсивности было отмечено, что время реконструкции увеличивается при увеличении числа используемых процессорных ядер больше четырех–шести. Анализ кода алгоритма выявил, что это падение производительности связано с синхронизацией потоков при записи рассчитанных значений интенсивности пикселей в структуру данных, представляющую собой репроецированные изображения. При малом числе потоков исполнения время ожидания потоками возможности доступа на запись было мало́, однако при увеличении числа потоков они стали тратить существенное время в ожидании своей очереди на запись. Кроме того, в случае малого числа потоков синхронизация потоков вообще могла быть отключена, так как вероятность конкурентного доступа на запись к одному пикселю была невелика, что не выполнялось в случае большого числа потоков. Для решения этой проблемы был использован подход, позволивший полностью отказаться от синхронизации потоков. Для этого была предложена новая схема разбиения задачи реконструкции на потоки. Изначально каждый поток обрабатывал первый еще не обработанный воксель объема, а так как два соседних вокселя могут репроецироваться на один и тот же пиксель, то конфликты записи были неизбежны. В новой схеме разделения работы между потоками каждому потоку выделяется некоторый подобъем, в котором он последовательно обрабатывает каждый пиксель. Таким образом, конкурентный доступ на запись к пикселям репроецированных изображений возможен только на границах подобьемов, однако потоки обрабатывают свою часть вокселей с примерно равной скоростью, и ситуация конкурентного доступа менее вероятна.

Кроме того, следует отметить, что если число неправильных значений интенсивности пикселей на репроецированных изображениях, появляющихся в результате конкурентной записи двумя потоками, не превышает нескольких процентов, то полученное трехмерное распределение интенсивности все еще остается корректным. Примененная оптимизация позволила увеличить производительность процедуры реконструкции в несколько раз и получить прирост производительности с увеличением числа процессорных ядер, используемых алгоритмом. На рис. 4 приведены графики зависимости реального и ожидаемого времени реконструкции для одного узла кластера и многоядерного сервера. Ожидаемое время реконструкции — это время реконструкции при использовании одного вычислительного ядра, поделенное на число вычислительных ядер, т.е. время реконструкции в случае идеальной масштабируемости. Для одного узла кластера (8 процессорных ядер) время реконструкции линейно уменьшалось с увеличением числа потоков исполнения от одного до восьми, т.е. была достигнута масштабируемость, близкая к идеальной. Для реконструкции на двухпроцессорном 24-ядерном сервере линейная зависимость не сохранялась, что может быть следствием различных особенностей как аппаратного, так и программного обеспечения. Тем не менее, на серверной платформе вычислительные ресурсы также использовались эффективно, о чем свидетельствует близкое расположение кривых для ожидаемого и реального времени реконструкции, в особенности в крайних точках.

В алгоритме расчета трехмерных полей скорости при использовании 24 процессорных ядер также было выявлено, что производительность практически не изменяется при увеличении числа используемых процессорных ядер, начиная с некоторого значения. Причиной этого послужило изменение соотношения времени, потраченного в параллельной и последовательной секциях кода, в соответствии с законом Амдала $S_N = \dfrac{1}{\alpha + (1-\alpha)N^{-1}}$, где S_N — ускорение при использовании N вычислителей (процессорных ядер) и α — доля последовательных вычислений. В этом алгоритме параллелизации подверглась только основная часть алгоритма — вычисление корреляционной функции расчетных областей, как наиболее ресурсоемкая. Кроме того, алгоритм включает в себя различные необязательные процедуры фильтрации векторных полей. Изначально время исполнения всех дополнительных процедур было несущественным по сравнению со временем основной части, поэтому и при распараллеливании алгоритма дополнительным процедурам внимания не уделялось. Такое соотношение времени исполнения параллельных и последова-

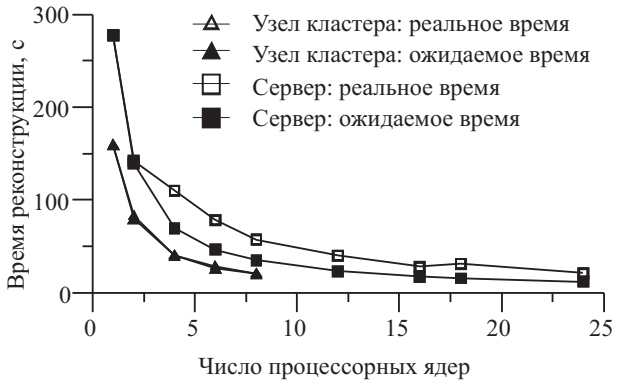


Рис. 4. Масштабируемость алгоритма реконструкции в зависимости от числа процессорных ядер на одном из узлов кластера и на многоядерном сервере

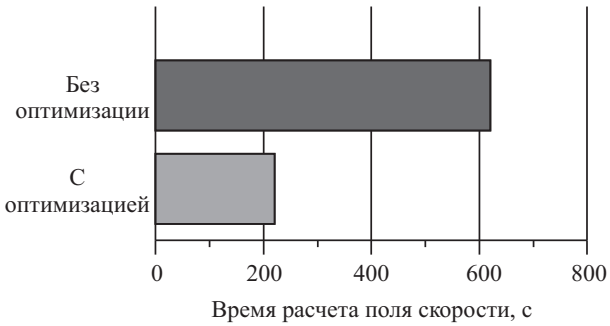


Рис. 5. Время расчета одного поля скорости до и после оптимизации для многоядерной архитектуры. Расчет производился с использованием двух AMD Opteron 6168 (всего 24 ядра)

тельных секций кода приблизительно сохранялось, когда число потоков обработки было сравнительно небольшим (при использовании до 4 процессорных ядер). Однако когда процедуре обработки было доступно 24 ядра, время основной (параллельной) части алгоритма стало в несколько раз меньше времени исполнения последовательной части (дополнительных процедур), производительность алгоритма расчета трехмерных полей скорости масштабировалась нелинейно. В данном случае решением проблемы стало использование стандарта OpenMP, с помощью которого были параллелизованы циклы процедур фильтрации. Производительность алгоритма после оптимизации увеличилась примерно в 3 раза (рис. 5).

4. Использование вычислительного кластера. Вычислительные кластеры являются одним из наиболее широко распространенных средств ускорения вычислительных задач. Высокая цена кластерных решений компенсируется их высокой вычислительной мощностью. Наиболее часто вычислительные кластеры используются в задачах моделирования различных явлений в физике, химии, биологии. В задачах моделирования применение вычислительных кластеров позволяет снизить время расчетов в сотни раз. Обработка экспериментальных данных на вычислительных кластерах не является распространенной практикой и ограничивается небольшим количеством случаев. При создании программ для обработки экспериментальных данных для кластерных систем необходимо учитывать ряд особенностей, свойственных этой задаче. При обработке экспериментальных данных требуется передавать на кластер и копировать обратно большие объемы данных — сотни гигабайт. Время копирования данных на кластер и обратно может оказаться сравнимым со временем обработки этих данных, что иногда может сделать использование кластера бессмысленным. С другой стороны, каждый эксперимент содержит до нескольких тысяч измерений, которые можно обрабатывать независимо друг от друга. Это позволяет не терять время на сетевое взаимодействие между узлами кластера при обработке каждого поля скорости, а рассчитывать на каждом узле только одно поле скорости.

С использованием интерфейса MPI, стандартного де-факто средства разработки параллельных программ для вычислительных кластеров, на основе ранее разработанного программного обеспечения для управления обработкой данных на системах с общей памятью была реализована система распределения задач между узлами и балансировки нагрузки при обработке данных на кластере. Принцип работы разработанной программы следующий: исходное задание на обработку представляет собой список имен файлов для обработки. Этот список разбивается на группы-задания, каждое задание содержит только имена файлов, относящихся к одному измерению (трассерному изображению, или полю скорости), т.е. являющихся единицей обработки данных. Копии этого списка хранятся у всех узлов-обработчиков и у управляющего узла. Далее узлы-обработчики через интерфейс MPI запрашивают номер задания у управляющего узла, который выдает номер еще не обработанного задания. Каждый из узлов самостоятельно производит запись результатов расчетов в постоянное хранилище. Благодаря такой схеме работы естественным образом нагрузка балансируется между узлами на уровне заданий. Узлы, справляющиеся с работой быстрее, будут обрабатывать больше данных, чем более медленные узлы. Это в особенности актуально для кластеров, состоящих из различных по техническим характеристикам узлов. В дальнейшем такая схема позволит естественным образом ввести транзакции по обработке данных, таким образом обеспечив устойчивость к аппаратным сбоям узлов кластера.

Для разработки, тестирования и последующей обработки экспериментальных данных в работе ис-

пользовался вычислительный кластер Информационно-вычислительного цетра Новосибирского государственного университета (ИВЦ НГУ), состоящий из блейд-серверов Hewlett-Packard. В настоящее время пиковая производительность этого кластера составляет 23 ТФлоп/с, с подробными техническими характеристиками кластера можно ознакомиться на сайте ИВЦ НГУ [7]. В работе авторам было доступно одновременно не более 16 вычислительных узлов. В силу использованной стратегии распараллеливания, которая не требует практически никакого обмена данными между узлами, с ростом числа задействованных в обработке данных узлов производительность системы растет линейно. Нелинейная зависимость производительности от числа используемых узлов кластера может появиться в силу ограничений подсистемы хранения данных.

5. Использование графических ускорителей. Одним из наиболее перспективных способов аппаратного увеличения производительности вычислений является использование для расчетов графических процессоров вместо центральных процессоров общего назначения. Графические процессоры позволяют достигать производительности до нескольких сотен миллиардов операций в секунду. Для достижения такой же производительности с применением центральных процессоров требуется создавать сложные и дорогостоящие вычислительные кластеры из нескольких десятков вычислительных узлов. В литературе представлены успешные результаты использования графических ускорителей для решения задач медицинской томографии (увеличение производительности итерационных методов реконструкции) [8–10].

На данный момент существуют две основные технологии создания программ для вычислений на графических процессорах: NVIDIA CUDA [11] и OpenCL [12]. Первая технология является закрытой и специфичной для устройств производства компании NVIDIA Corporation. Технология OpenCL является открытым стандартом, разработанным организацией Khronos Group совместно с различными производителями вычислительного оборудования. Технология OpenCL позволяет создавать программы не только для графических процессоров, но и для любых гетерогенных вычислительных систем. Существуют реализации OpenCL для центральных процессоров Intel, AMD, графических чипов NVIDIA, ATI (AMD), S3 Graphics (VIA) и векторных процессоров IBM Cell. Обе технологии позволяют составлять программы на языке Си с использованием специальных расширений.

В представленной работе была использована технология OpenCL в силу того, что она позволяет задействовать большее количество вычислительных устройств и платформ, а также не привязана к конкретному производителю оборудования. Из минусов этой технологии можно отметить то, что в некоторых аспектах она отстает от CUDA, так как появилась на несколько лет позже. В качестве ускорителя была использована видеокарта NVIDIA GeForce GTX 480 с драйвером, поддерживающим OpenCL версии 1.1.

При использовании графических процессоров для ускорения вычислений следует учитывать ряд специфических особенностей этих типов вычислительных устройств. Их важной архитектурной особенностью является то, что все потоки объединяются в группы, для каждой группы существует один счетчик команд, соответственно все потоки одной группы выполняют одну инструкцию, оперируя при этом разными данными. Это ведет к тому, что ветвления в исполняемом коде негативно сказываются на производительности программы для ГПУ, так как всеми потоками будет исполнен сначала один вариант ветвления, затем другой. Таким образом, для получения максимальной эффективности вычислений на ГПУ необходимо модифицировать код так, чтобы минимизировать число ветвлений в коде. Другой особенностью графических чипов является их способность исполнять число потоков, многократно превосходящее реальное число вычислительных блоков без потерь производительности на переключение контекста исполнения. Более того, для получения наиболее высокой производительности необходимо, чтобы число потоков исполнения данных существенно превосходило число вычислительных блоков. Еще одна особенность графических процессоров заключается в том, что обычно время передачи данных между ОЗУ компьютера и ОЗУ графического процессора достаточно велико. Поэтому имеет смысл использовать графические адаптеры для ускорения тех алгоритмов, которым не требуются частые обмены данных между устройством и центральным процессором.

Из двух алгоритмов обработки данных в методе Tomo PIV более подходящим для полного переноса на графический процессор является алгоритм томографической реконструкции трехмерных изображений частиц. Программный код этого алгоритма достаточно прост и компактен, а значит, может быть полностью перенесен на графический процессор. Это, в свою очередь, избавит от необходимости частого обмена данными между центральным процессором и графическим ускорителем. Каждый воксель объема обрабатывается данным алгоритмом независимо от других вокселей, а их число составляет от нескольких миллионов до нескольких миллиардов, соответственно размерность задачи многократно превышает число вычислительных блоков графического процессора. Алгоритм расчета полей скорости также может быть ускорен за счет использования графических процессоров путем переноса наиболее ресурсоемких операций,

таких как преобразование Фурье, на графический процессор, однако это требует существенной переработки кода алгоритма для получения выигрыша в производительности. При переносе на графический процессор алгоритма томографической реконструкции SMART удалось полностью избавиться от ветвлений за счет использования механизмов работы с изображениями, реализованных в библиотеке функций OpenCL. Основными объектами, с которыми работает алгоритм томографической реконструкции, являются изображения. Встроенные в OpenCL механизмы работы с ними автоматически обрабатывают случаи чтения по координатам, выходящим за границы изображения. Это позволяет обойтись без дополнительных проверок краевых случаев, которые, как правило, приводят к ветвлениям в коде алгоритма. Кроме того, при работе с объектами-изображениями на ГПУ доступна аппаратная реализация процедуры билинейной интерполяции интенсивности, которая используется алгоритмом SMART для расчета интенсивности в точке проекции вокселя. Работа с объектами-изображениями по сравнению с простыми буферами делает программный код в целом более понятным и наглядным, так как входные данные алгоритма являются изображениями.

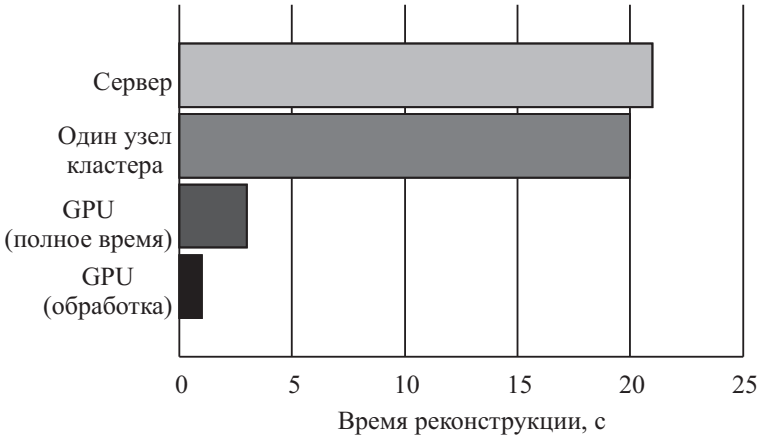


Рис. 6. Время реконструкции на различных платформах

На рис. 6 приведено среднее время реконструкции одного трехмерного изображения размером $2048 \times 2048 \times 256$ вокселей с учетом времени на запись и чтение данных с графического ускорителя и без учета этого времени, а также приведено время расчета такого же объема на кластере и многоядерном сервере. Реконструированные с помощью графического процессора объемы сравнивались с объемами, полученными с использованием центрального процессора. Сравнение показало полную идентичность этих результатов реконструкции.

Сравнительная таблица эффективности использования различных компьютерных платформ для задач томографической реконструкции

Платформа	Время реконструкции объема $2048 \times 2048 \times 256$ вокселей, с	Время расчета одного эксперимента (6000 измерений, 2 объема, 15 итераций реконструкции)	Стоимость обработки одного эксперимента, руб.
Кластер	20	35 часов (16 узлов кластера)	13 700
Сервер	21	21 день	12 000
Офисный ПК	116	4 месяца	3 300
ГПУ	3	3 дня	100

6. Заключение. В рамках данной работы были спроектированы и реализованы параллельные версии расчетных процедур, входящих в состав метода Tomo PIV для измерения трехмерных полей скорости в газовых и жидкостных потоках. Были созданы реализации алгоритмов для многоядерных серверных решений (в том числе узлов кластера) и графических адаптеров, разработана программа для управления обработкой и балансировки нагрузки на кластере. Разработанное программное обеспечение имеет линейную масштабируемость производительности при увеличении числа используемых узлов кластера и количества используемых процессорных ядер на одном узле. Показан существенный выигрыш в производительности при использовании высокопроизводительных вычислительных платформ. При использовании вычислительной платформы с 24 процессорными ядрами в алгоритмах были выявлены и успешно

решены проблемы производительности, которые не проявлялись при меньшем числе процессорных ядер. В таблице представлено сравнение эффективности использования каждой из высокопроизводительных платформ для обработки данных, полученных методом Tomo PIV. Для сравнения в таблице указано время обработки данных с использованием стандартного "офисного" ПК. Видно, что наиболее эффективным способом обработки данных является использование графических адаптеров. Использованные вычислительные системы имели следующие конфигурации:

— кластер ИВЦ НГУ: 2 Intel Xeon 5535, 16 Гб ОЗУ, ОС SUSE Linux Enterprise Server 11;

— многоядерный сервер: 2 AMD Opteron 6168 (12 ядер), 16 Гб ОЗУ, ОС Windows 7;

— офисный ПК: AMD Athlon x2 64 5600+, 3 Гб ОЗУ, ОС Windows 7;

— графические процессорные устройства: NVIDIA GeForce GTX 480, 1536 Гб DDR5.

Немаловажным критерием для выбора того или иного типа платформы обработки данных являются финансовые затраты. Для оценки финансовой эффективности можно рассмотреть, например, критерий "стоимость обработки одного эксперимента". Оценка по данному критерию осуществлялась следующим образом. Для кластерных решений стоимостью считается NPt/T, где N — число доступных одному пользователю узлов кластера, P — средняя цена одного узла кластера, t — расчетное время обработки эксперимента и T — время окупаемости, взятое равным одному году. Для серверных решений стоимостью считается Pt/T, где P — сметная стоимость серверной системы, а параметры t и T имеют тот же смысл, что и для кластеров. Для офисного ПК и ГПУ расчеты производились по той же формуле, что и для серверной системы, при этом стоимость офисного ПК была взята равной 10 000 руб, для графического процессора — средней рыночной стоимости использованного в тестировании ускорителя. Исходя из проведенных оценок (см. таблицу) можно сделать вывод, что наименее затратным способом обработки данных является решение на графических процессорах.

Список литературы

1. *Elsinga G.E., Scarano F., Wieneke B., van Oudheusden B.W.* Tomographic particle image velocimetry // Experiments in Fluids. 2006. **41**. 933–947.
2. *Stanislas M., Okamoto K., Kahler C.J., Westerweel J.* Main results of the third international PIV challenge // Experiments in Fluids. 2008. **45**. 27–71.
3. *Adrian R.J.* Twenty years of particle image velocimetry // Experiments in Fluids. 2005. **39**. 159–169.
4. *Бильский А.В., Ложкин В.А., Маркович Д.М., Токарев М.П., Шестаков М.В.* Оптимизация и тестирование томографического метода измерения скорости в объеме потока // Теплофизика и аэромеханика. 2011. **18**, № 4. 1–12.
5. *Atkinson C., Soria J.* An efficient simultaneous reconstruction technique for tomographic particle image velocimetry // Experiments in Fluids. 2009. **47**. 553–568.
6. *Бильский А.В., Маркович Д.М., Токарев М.П.* Адаптивные алгоритмы обработки изображений частиц для расчета мгновенных полей скорости // Вычислительные технологии. 2007. **12**, № 3. 109–131.
7. Информационно-вычислительный центр НГУ. Описание комплекса (http://nusc.ru).
8. *Jang B., Kaeli D., Do S., Pien H.* Multi GPU implementation of iterative tomographic reconstruction algorithms // Proc. IEEE Int. Symposium on Biomedical Imaging. Boston, 2009. 185–188.
9. *Xu F., Mueller K.* Towards a unified framework for rapid 3D computed tomography on commodity GPUs // Nuclear Science Symposium Conference Record. 2003. **4**. 2757–2759.
10. *Maar S., Batenburg K., Sijbers J.* Experiences with Cell-BE and GPU for tomography // Proc. of the 9th Int. Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation. Berlin: Springer, 2009. 298–307.
11. NVIDIA Corporation. CUDA C Programming Guide Version 4, 2011 (http://nvidia.com).
12. Khronos OpenCL Working Group. The OpenCL Specification Version 1.1, 2011 (http://www.khronos.org/opencl/).

Поступила в редакцию
02.02.2012