Адаптация решателя динамического напряженно-деформированного состояния методом дискретных элементов для неоднородных вычислительных систем

С.С. Алексеев, В.В. Гетманский, А.Е. Андреев

Волгоградский государственный технический университет

Рассматривается перенос решателя динамического напряженно-деформированного состояния (НДС), использующего модель дискретных элеметов, для неоднородных вычислительных платформ с использованием стандарта OpenCL. Для эффективного использования системы, оснащенной разнородными вычислителями, прежде всего такими как GPU и FPGA, необходим перенос всего решателя либо отдельных его компонент на ускоритель. Описаны особенности OpenCL реализации. Приведены результаты тестирования решателя на GPU, в сравнении с расчетом на двух центральных процессорах.

Ключевые слова: OpenCL, неоднородные вычислители, междисциплинарное моделирование, метод дискретных элементов

1. Введение

Разработка решателя динамического напряженно-деформированного состояния использует представление модели в виде набора дискретных элементов со связями [1]. Основное свойство элемента – то, что он унифицированный, то есть вычисления, связанные со всеми элементами одинаковы, что очень хорошо отображается на SIMT-архитектуры (Single Instruction Multiple Threads), которые характерны для GPGPU. Таким образом, перенос расчетного модуля на такие архитектуры актуален и рассматривается далее. Для этого выбран наиболее универсальный открытый стандарт OpenCL. Расчетный модуль разрабатывается для CAE пакета моделирования многотельной динамики ФРУНД (Формирование и Решение Уравнений Нелинейной Динамики) [2]. В решателе ранее была реализована оптимизация под архитектуры современных процессоров с векторными инструкциями [3-4], включая широкие 512-битные векторные регистры, появившиеся в сопроцессорах Xeon Phi, а затем в центральных процессорах Xeon Phi второго поколения и новых семействах Intel Xeon. Для дальнейшего ускорения расчета, а также в качестве альтернативы рассматриваются платформы GPGPU и FPGA, поэтому для реализации была выбрана технология гетерогенных вычислений OpenCL.

2. Описание задачи и анализ существующих решений

Пакет моделирования многотельной динамики ФРУНД позволяет проводить инженерный анализ при проектировании агрегатов и конструкций. В список его возможностей входит решение задачи напряженно-деформированного состояния для определения прочности конструкций. Кроме того, расчет может проводиться в комплексе с другими типами расчетов в системе, например, с тепловым решателем в рамках междисциплинарного моделирования. Для её решения была использована технология OpenCL для GPU.

Метод дискретных элементов (МДЭ), примененный в решателе, обычно не используется для решения НДС, в большинстве CAE систем используется метод конечных элементов, где тело аппроксимируется сеткой. Этот метод весьма производителен, однако, имеет ряд недостатков, связанных, в частности, с некоторыми расхождениями с практикой, например, с точки зрения оценки долговечности нагруженных элементов системы в динамике. В методе дискретных элементов тело аппроксимируется регулярной ортогональной сеткой элементов, упруго связанных между собой.

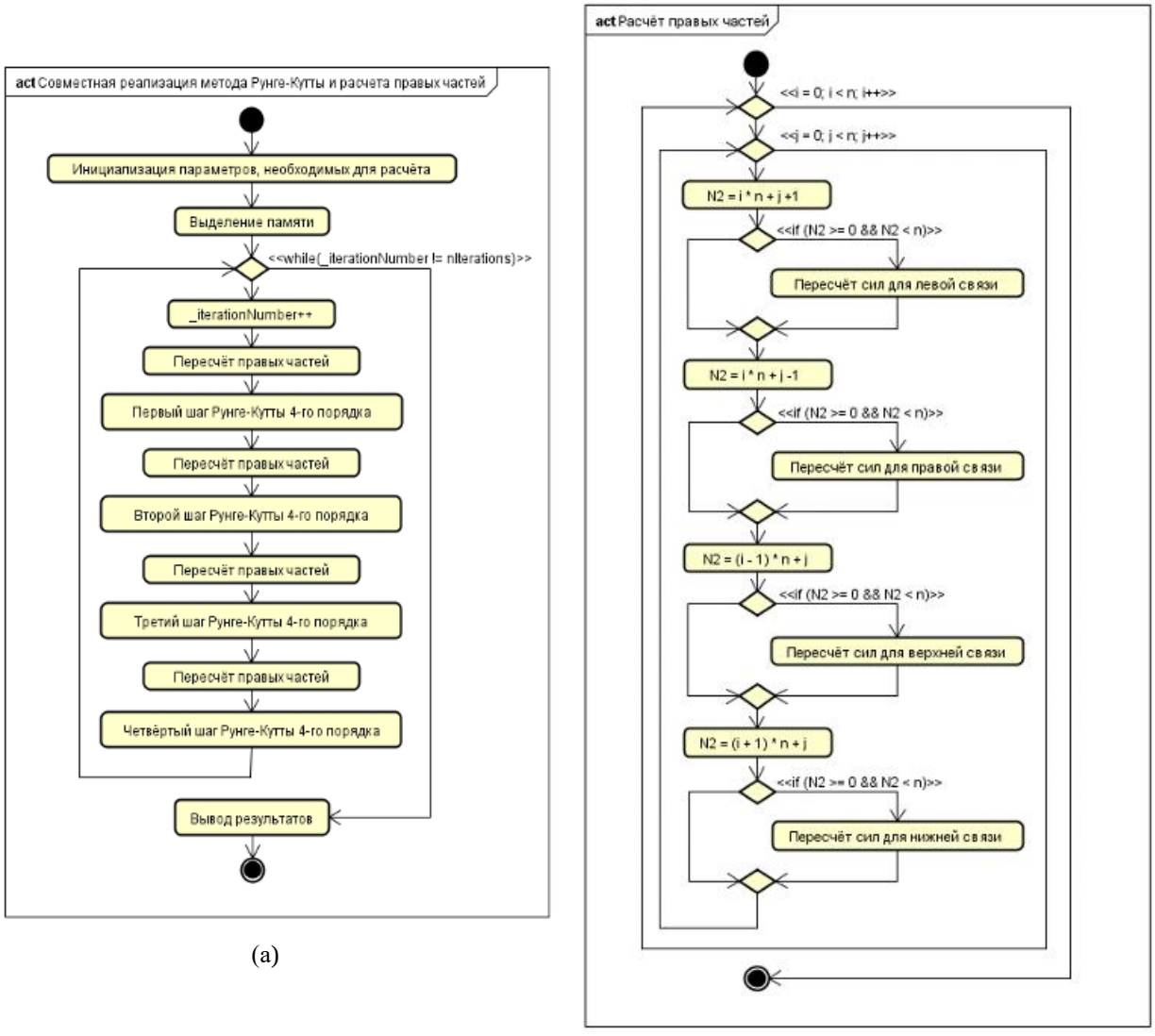
Сам алгоритм расчета НДС с помощью МДЭ состоит из трёх затратных этапов. Все этапы происходят в рамках временного моделирования внутри минорной и мажорной итераций. Мажорная итерация содержит в себе указанное при старте алгоритма количество минорных итераций. После каждой мажорной итерации происходит сохранение промежуточных данных.

Первый этап - вычисление сил на основе сдвигов, как линейных, так и угловых, элементов исходя из закона Гука и граничных условий, иначе говоря, правых частей дифференциальных уравнений. При этом в отдельный массив сохраняется тензор напряжений. Эта часть многократно используется при интегрировании между этапами метода Рунге-Кутты 4 порядка точности. На этом этапе производится чтение данных о сдвигах для текущего и всех соседних элементов. Запись происходит только в секцию данных для текущего элемента.

Второй этап – итерация метода Рунге-Кутты 4 порядка. Интегрирование производится по линейным и угловым степеням свободы, т. е. на один элемент приходится 6 неизвестных. На этом этапе производится чтение и запись данных только в секцию текущего элемента.

Третий этап - расчет матриц поворота для элемента. Этот этап выполняется после каждого интегрирования и отвечает за то, чтобы на основе углов вычислить матрицы поворота, которые используются на этапе расчета сил для их трансформации в системы координат отдельных элементов.

Схематично алгоритм описан на рисунке 1.



(а)

(б)

Рис. 1. Алгоритм реализации метода Рунге-Кутты с перерасчетом правых частей (а), включая расчет сил по связям с соседними дискретными элементами (б)

Метод дискретных элементов с некоторыми оговорками можно считать похожим на метод решения задачи N тел. В частности, в предлагаемом методе расчета НДС используется численное интегрирование методом Рунге-Кутты 4 порядка, который зачастую используется при решении задачи N тел. В то же время расчеты, выполняемые на каждом шаге интегрирования, характер взаимодействий отличаются.

На каждый элемент в модели НДС приходится 6 и менее расчетов сил, в то время как в базовой реализации метода N тел на каждый элемент приходится N расчетов сил. Сложность алгоритма N тел $O(N^2)$, алгоритма расчета НДС с МДЭ O(N). Отметим, что существуют оптимизированные реализации решения задачи N тел с сложностью O(N).

Также стоит отметить, что существующий способ уменьшения количества расчета сил между элементами, обусловленный соотношением из третьего закона Ньютона, может быть применен в обеих задачах. Однако данный способ влечет за собой большие потери на синхронизацию данных, таким образом, нивелируя выигрыш от уменьшения количества рассчитываемых взаимодействий.

Тема эффективности алгоритмов расчета задачи N тел давно и хорошо проработана и используется в практических целях [5 - 7]. Из этих результатов можно выделить несколько особенностей алгоритма, относящихся также и к представленному решателю НДС.

Хорошая масштабируемость. В первую очередь это относится к оптимизированным версиям алгоритмов решения задачи N тел. Такие версии могут приближаться к сложности O(N), приближаясь к сложности решателя НДС. Последний, в свою очередь, также относительно хорошо масштабируется. Это связано с отсутствием зависимостей по данным внутри одной итерации цикла интегрирования. Однако всё ещё возникают ограничения на скорость сохранения данных об элементах между итерациями.

Чувствительность к доводке и векторизации. Архитектуры, рассчитанные на массовый параллелизм, хорошо отзываются на доводку и оптимизацию, которые сокращают время расчета. Более того, время расчета может изменяться при использовании разных компиляторов, например, для CPU Intel более высокую эффективность относительно GCC показал фирменный компилятор, хоть выигрыш составил не более 5%, на большом размере модели это может заметно сократить время расчета.

3. Реализация расчета на OpenCL

Для распараллеливания использована технология OpenCL. Это связано с тем, что OpenCL позволяет запускать высокооптимизированный параллельный код на большом количестве устройств, включая видеоускорители от AMD, nVidia, Intel, а также использовать ПЛИС. Кроме того, технология предоставляет возможность сборки и отладки ядер под центральные процессоры с использованием наборов команд AVX и SSE. Такая универсальность открывает перспективы для гетерогенных вычислений.

Однако в первую очередь собственная реализация вычислений ориентирована под GPU. Выбор обусловлен высокой производительностью этого типа устройств и акцентом на векторные операции. Рассматриваемая нами задача использует большое количество векторных операций и при этом хорошо декомпозируется при большом числе дискретных элементов, для подобных классов задач известно множество реализаций [6-7]. Ранее с участием авторов уже предпринималась попытка переноса алгоритмов работы решателя на OpenCL, но она выполнялась для существенно упрощенного прототипа решателя.

Написание кода на OpenCL под GPU связано с рядом специфических особенностей архитектуры как всей системы, так и видеокарт в частности. Так как GPU соответствуют по таксономии Флинна к классу SIMD, в них возникают соответствующие особенности, которые могут негативно сказаться на производительности алгоритма. В первую очередь это сильное замедление при обработке сложных ветвлений алгоритма. Это ограничение сказывается на том, как учитываются граничные условия. Ветвления, где применяются эти граничные условия, вынесены на хост. В вычислительном ядре OpenCL расчет производится из предположения, что для каждого дискретного элемента существуют все граничные условия. Однако значащие условия появляются только там, где они есть фактически в самой модели.

В частном виде применение граничных условий выглядит следующим образом. Перед исполнением ядра на хосте рассчитываются значения применяемых условий для всех элементов и всех типов граничных условий. Так, для граничного условия (1) на i-том элементе в буфер слагаемых со второй производной будет записано либо условие по умолчанию, т. е. ноль, либо реальное условие, если оно фактически есть :

$$\vec{u}_{tt}(\vec{x}_i, t) = \vec{\mu}(t) \quad . \qquad (1)$$

Аналогично для условия (2), с той лишь разницей, что условие хранится в виде множителя для координаты :

$$\vec{u}(\vec{x}_i, t) = 0 \quad . \qquad (2)$$

Граничное условие фактически означает блокировку степени свободы. По умолчанию степень свободы разблокирована. Так как массив содержит именно множители, по умолчанию значения в этом массиве равны единицам. Для случая, когда степень свободы блокируется, значения в массиве для блокируемых элементов приравниваются к нулю и играют роль маски, накладываемой на степень свободы.

Вторая ситуация, в которой должны применяться ветвления, согласно алгоритму, состоит в определении соседних элементов относительно текущего элемента для расчета напряжений и сил. Так, если одна из граней элемента не соединена ни с одним из других элементов модели, расчет сил и напряжений не должен производиться. У элемента в модели расчета имеется шесть граней, для каждой из которой может как существовать, так и не существовать соседний. Для корректной обработки каждого элемента в ядре введен массив флагов, отображающих наличие или отсутствие соседнего относительно текущей грани. Флаги выступают в качестве множителя. В случае наличия соседнего элемента он имеет значение единицы, иначе — ноль. Так как расчет производится в любом случае, в случае отсутствия соседа по умолчанию соседним элементом считается нулевой. После применения множителя связи влияние этой связи становится нулевым. Так минимизируется количество условных блоков на этапе применения граничных условий.

Кроме этого, стоит указать на возможность уменьшить количество итераций на расчет сил и напряжений для элемента, которая является следствием третьего закона Ньютона. Так как силы, действующие между дискретными элементами разнонаправленные, но одинаковые по амплитуде, эти силы можно вычислять один раз для каждой пары элементов. Однако в текущей реализации эта оптимизация не применяется из-за небольшого относительного прироста производительности. Максимальный выигрыш при использовании этого приема без учета трат на синхронизацию составляет двукратное уменьшение времени расчета сил. Но по причине необходимости применять приемы синхронизации вычислений, либо редукцию, в конечном итоге прирост производительности частично нивелируется, хотя такой поход можно рассматривать как небольшой резерв для достижения прироста производительности в будущем.

Можно выделить критерий, по которому происходило избавление от условных блоков, которые находятся внутри циклов. Условие можно считать допустимым, если оно позволяет напрямую развернуть цикл. Допустимыми здесь остались условия, зависящие только от текущего значения числового итератора цикла. Это связано с оптимизацией компилятором циклов, когда циклы разворачиваются в случае, когда известно количество итераций. Кроме того, в стандарте OpenCL 2.0 введена директива компилятора, явно указывающая на необходимость развернуть цикл. В зависимости от компилятора такие условия могут быть также раскрыты и распределены по итерациям.

Другим важным элементом при работе с OpenCL на GPU остается ограничение шины между хостом и устрйоством. Дело в том, что PCI Express является «бутылочным горлышком» системы такого рода. Исходя из этого критически важно минимизировать объем пересылаемых данных между GPU и CPU.

В таблице 1 приведены главные буферы, участвующие в расчете. Пересылка буфера Stress необходима для расчета общего напряжения по фон Мизесу на хосте. Буфер DataInternal содержит позиции, повороты, первые и вторые производные указанных параметров. Синхронизация буфера DataInternal необходима для корректного расчета и необходима для обнаружения ошибок и переполнений, так как проверки производятся на стороне хоста.

Пересылка Stress производится каждый раз, когда хост рассчитывает промежуточный результат напряжений. Это происходит с некоторым промежутком во время мажорной итерации. DataInternal же необходимо отсылать на устройство и обратно каждый раз, так как обнаружения переполнений производится каждую итерацию, кроме того, сам процесс моделирования сохраняется в файл, где хранятся напряжения и координаты каждого узла между мажорными итерациями. Буферы BoudaryFixed, BoudaryForces и LinkedElements отсылаются каждый раз, когда происходит изменение модели. Таким образом создается задел под будущее расширение функциональности, например, для удаления связей при превышении напряжений, моделируя разрушение детали.

Все второстепенные буферы и константы загружаются один раз при инициализации и перечислены в таблице 1.

Таблица 1. Буферы для хранения и пересылки данных между хостом и устройством

Название буфера	Флаги памяти хоста	Содержимое
DataInternal	Read/Write	Линейные и угловые координаты, первые и вторые производные
Stress	Read/Write	Напряжения
BoudaryFixed	Write	Граничные условия для координат в виде флагов
BoudaryForces	Write	Граничные условия для вторых производных в виде слагаемых
LinkedElements	Write	Номера соединенных элементов по шести граням, нуль если сосед отсутствует

4. Результаты

Замеры скорости работы производились на нескольких устройствах. Первая платформа, выбранная в качестве опорной, это двухпроцессорная система на Intel Xeon E5 2660 без Hyper Threading и с использованием команд AVX. Для этой платформы использовалась отдельная версия программы, специально написанная под процессоры с использованием команд AVX. На второй и третьей платформе использовалась версия под OpenCL. Это видеокарты AMD RX 460 и nVidia Tesla K20. Первая значительно медленнее второй и её использование связано с оценкой относительной производительности разных ускорителей чтобы определить, как сильно влияют затраты, связанные с выполнением служебных операций, на производительность.

Исходя из результатов, приведенных на рис. 2, можно сделать следующие выводы. На основании того, что скорость выполнения расчета на Tesla K20 в пике превосходит скорость расчета на RX 460 в 4 раза, а в минимуме в 2 раза (при том, что заявленная пиковая производительность устройств соответственно 1.17 TFLOPS и 122 GFLOPS - при операциях с двойной точностью различается почти в 10 раз), можно сделать вывод о том, что издержки на служебные операции и пересылки между хостом и устройством растут очень быстро. Действительно, как говорилось выше, некоторые буферы активно пересылаются между каждой итерацией вычислений. Это закономерно замедляет работу всей программы из-за простоя как GPU, так и CPU в ожидании завершения пересылки.

Относительный выигрыш по времени от использования GPU относительно двух процессоров Xeon E5 2660 составляет в пике преимущество всего 1.5 раза, что также свидетельствует о больших издержках на обмен данными. Если сравнить пиковую производительность GPU и CPU, то заявленная производительность используемого центрального процессора составляет около 200 GFLOPS при операциях с двойной точностью, то есть при грубом приближении два Xeon E5 2660 обеспечивают производительность меньше 400 GFLOPS, а один GPU TESLA K20 – около 1,2 TFLOPS, то есть в 3 раза выше.

Можно отметить то, что при малом количестве элементов время расчета на GPU и на CPU практически не отличается. Это связано с затратами на подготовку данных для отправки на

GPU и издержки, связанные с обменом данными при инициализации. Хотя затраты на этот этап небольшие, они растут с ростом числа элементов и на крупных моделях могут достигать существенных величин.

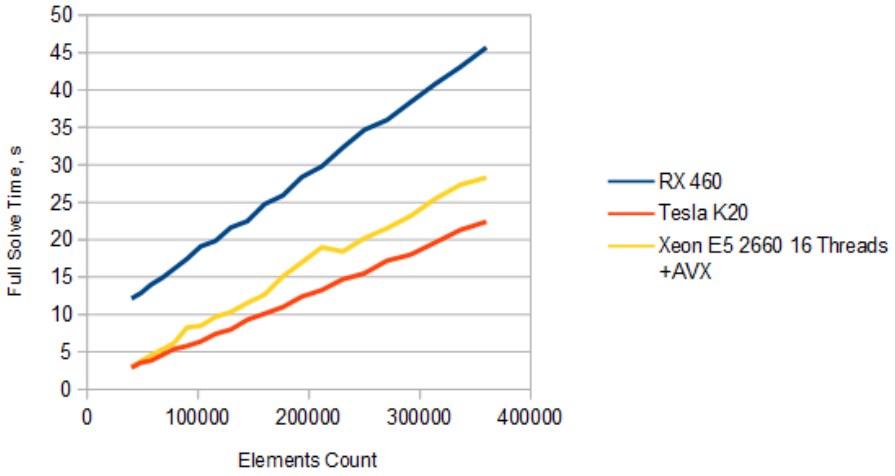


Рис. 2. График зависимости времени расчета от количества элементов в модели на AMD RX 460, Nvidia Tesla K20 и двумя Intel Xeon E5 2660

Ускорение расчета на GPU относительно двух CPU, как видно из рисунка 2, достигает 1.5 раз. Можно сделать вывод о недостаточной эффективности работы алгоритма (исходя из соотношения пиковой производительности GPU K20 и двух CPU можно было бы ожидать ускорения, близкого к 3). Это, в первую очередь, связано с неэффективной пересылкой данных между итерациями. Отдельно проведенное решение модели размером миллион элементов подтверждает предел ускорения данной версии решателя.

На рисунке 3 можно наблюдать визуализацию напряжений в тестовой модели. Тестовая модель — плоская пластина. Она закреплена в части правой грани. Сила прикладывается к углу пластины.

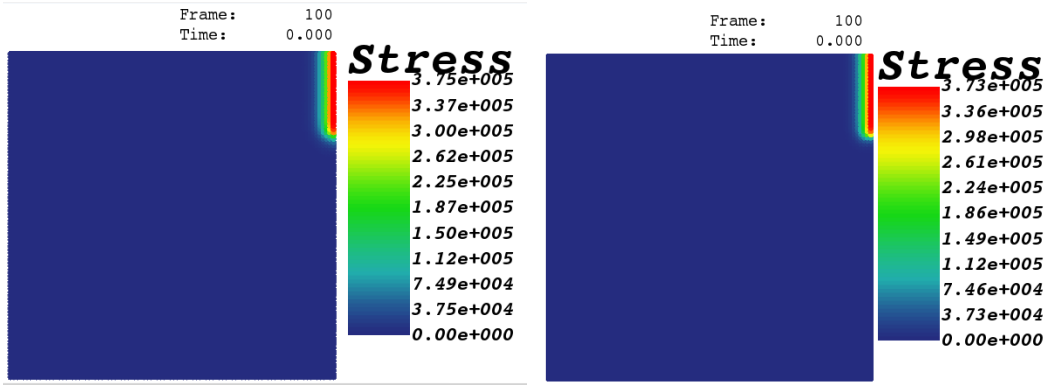


Рис. 3. Визуализация результата расчета НДС тестовой модели (слева – расчет на CPU, справа – на GPU/OpenCL)

Результаты визуализации (как и сами результаты расчета) совпадают с визуализацией результатов при расчете на CPU.

5. Выводы и перспективы дальнейших исследований

Относительный прирост производительности при использовании GPU заметно ниже максимально возможного. Как видно из результатов, это связано с большими издержками на пересылку буферов данных, которую необходимо совершать для проведения второстепенных расчетов, не перенесенных на OpenCL. В первую очередь для дальнейшего ускорения расчетов необходимо переносить оставшиеся алгоритмы на GPU. В идеальном случае количество пересылаемых данных должно быть минимально возможным и пересылка должна происходить только при инициализации расчетов. Однако производить считывание данных только при окончании расчета не получится — необходимо сохранять в файл данные, получаемые при вычислениях между итерациями для наблюдения динамики.

Потенциально можно повысить производительность, применяя в модели третий закон Ньютона, это позволит вдвое сократить количество вычислений на расчет сил. Однако для этого нужно будет реализовать алгоритм синхронизации данных на GPU с помощью редукции. Это повлечет за собой заметные потери, особенно на большом количестве элементов. Применять традиционные приемы синхронизации на GPU неэффективно — общая скорость работы в таком случае будет низкой.

Кроме модификации реализации алгоритма расчета следует исследовать возможность перевести весь расчет на использование чисел одинарной точности. Это может повлечь за собой заметное ускорение вычислений. Кроме того, что производительность GPU на числах одинарной точности в разы выше, между хостом и устройством будет передаваться почти в 2 раза меньше данных. Однако стоит рассмотреть проблему сопоставимого увеличения ошибки численного метода интегрирования.

В целом реализация и тестирование OpenCL версии расчетного модуля показали, что вычисления в решателе могут ускоряться с помощью GPGPU, кроме того имеется некоторый потенциал для повышения производительности. Получено ускорение до 1,5 раз по сравнению с наилучшей на данный момент версией расчета с векторизацией для двух Intel Xeon E5 в одной системе.

Дальнейшая оптимизация также может быть связана с переносом OpenCL кода на FPGA и исследованием решения задачи на реконфигурируемых вычислителях.

Работа выполнена при финансовой поддержке РФФИ - проекты №№ 16-47-340385, 16-07-00534, 15-01-04577, 15-07-06254 и Администрации Волгоградской области – проект № 16-47-340385.

Литература

1. Гетманский В.В., Горобцов А.С., Измайлов Т.Д. Распараллеливание расчёта напряжённо-деформированного состояния тела в многотельной модели методом декомпозиции расчётной области // Известия ВолгГТУ. Серия "Актуальные проблемы управления, вычислительной техники и информатики в технических системах". Вып. 16, ВолгГТУ, № 8 (111), 2013, с. 5-10.
2. Горобцов А.С., Гетманский В.В., Андреев А.Е., Doan D.T. Simulation and Visualization Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings / ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015, p. 379-391.
3. Гетманский В.В., Андреев А.Е, Мовчан Е.О. Особенности применения различных наборов векторных инструкций для оптимизации кода расчёта динамики систем тел / // Суперкомпьютерные дни в России = Russian Supercomputing Days : тр. междунар. конф. (г. Москва, 26-27 сентября 2016 г.) / Суперкомпьютерный консорциум университетов России, Федеральное агентство науч. организаций России, МГУ им. М.В. Ломоносова. - Москва, 2016. - С. 365-372.

4. Гетманский, В.В. Ускорение расчёта динамического напряжённо-деформированного состояния с помощью наборов векторных инструкций / В.В. Гетманский, Е.О. Мовчан, А.Е. Андреев // Известия ЮФУ. Технические науки. - 2016. - № 11 (184). - С. 27-39.

5. Швец П.А., Адинец А.В. Исследование возможностей ручной и автоматической оптимизации типовых алгоритмических структур для графических ускорителей// Информационные технологии Вестник Нижегородского университета им. Н.И.Лобачевского. - 2013. - № 1. - с. 289-294.

6. Адинец А. В. Анализ эффективности решения задачи n тел на различных вычислительных архитектурах // Вестник ННГУ. - 2009. - №5 — С.219-229.

7. В.Е.Баранов, В.Г. Макарян Программный комплекс для численного моделирования дискообразной самогравитирующей системы неупругих частиц с массивным центром // Известия Самарского научного центра Российской академии наук. - т. 15. - 2013 - №6(3). - С. 584-588.

Adaptation of dynamic stress strain solver based on discrete elements method for heterogeneous computing hardware

S.S. Alekseev, V.V. Getmanskiy, A.E. Andreev

Volgograd State Technical University

Transfer of the dynamic stress-strain state solver based on discrete elements approach to heterogeneous computing platforms using OpenCL is considered. It is performed to run the solver on heterogeneous computing hardware, such as GPGPU and FPGA. The features of the OpenCL implementation are described. The results of the solver testing on GPU in comparison with two CPUs use are presented.

Keywords: OpenCL, heterogeneous computing hardware, Multiphysics simulation, discrete elements method

References

1. Getmanskiy V.V., Gorobcov A.S., Izmaylov T.D. Rasparallelivanie rascheta napryazhenno-deformirovannogo sostoyaniya tela v mnogotel'noy modeli metodom dekompozicii raschetnoy oblasti // Izvestiya VolgGTU. Seriya "Aktual'nye pro-blemy upravleniya, vychislitel'noy tekhniki i informatiki v tekhnicheskih siste-mah". Vyp. 16, VolgGTU, № 8 (111), 2013, pp. 5-10.
2. Gorobcov A.S., Getmanskiy V.V., Andreev A.E., Doan D.T. Simulation and Visualiza-tion Software for Vehicle Dynamics Analysis Using Multibody System Approach // Creativity in Intelligent Technologies and Data Science. CIT&DS 2015: Proceedings / ed. by A. Kravets et. al., Springer International Publishing, Switzerland, 2015, p. 379-391.
3. Getmanskiy V.V., Andreev A.E, Movchan E.O. Osobennosti primeneniya razlichnyh naborov vektornyh instrukciy dlya optimizacii koda rascheta dinamiki sistem tel / // Superkomp'yuternye dni v Rossii = Russian Supercomputing Days : tr. mezh-dunar. konf. (g. Moskva, 26-27 sentyabrya 2016 g.) / Superkomp'yuternyy konsorcium universitetov Rossii, Federal'noe agentstvo nauch. organizaciy Rossii, MGU im. M.V. Lomonosova. - Moskva, 2016. - pp. 365-372.
4. Getmanskiy, V.V. Uskorenie rascheta dinamicheskogo napryazhenno-deformirovannogo sostoyaniya s pomoshch'yu naborov vektornyh instrukciy / V.V. Getmanskiy, E.O. Movchan, A.E. Andreev // Izvestiya YUFU. Tekhnicheskie nauki. - 2016. - № 11 (184). - pp. 27-39.
5. Shvec P.A., Adinec A.V. Issledovanie vozmozhnostej ruchnoj i avtomaticheskoj optimizacii tipovyh algoritmicheskih struktur dlja graficheskih uskoritelej// Informacionnye tehnologii Vestnik Nizhegorodskogo universiteta im. N.I.Lobachevskogo. - 2013. - # 1. - pp. 289-294.
6. Adinec A. V. Analiz ehffektivnosti resheniya zadachi n tel na razlichnyh vychislitel'nyh arhitekturah // Vestnik NNGU. - 2009. - №5 — pp.219-229.
7. V.E.Baranov, V.G. Makaryan Programmnyj kompleks dlya chislennogo modelirovaniya diskoobraznoj samogravitiruyushchej sistemy neuprugih chastic s massivnym centrom // Izvestiya Samarskogo nauchnogo centra Rossijskoj akademii nauk. - t. 15. - 2013 - №6(3). - pp. 584-588.