# CUDA-MPI реализация метода BiCGStab для решения СЛАУ в задачах моделирования больших пластических деформаций*

Ю.В. Халевицкий, А.В. Коновалов, А.С. Партин

Институт машиноведения УрО РАН

Рассмотрено применение гетерогенных кластеров с графическими ускорителями для моделирования больших высокотемпературных пластических деформаций методом конечных элементов. Вычислительная система задействуется для ускорения решения системы линейных алгебраических уравнений с разреженной матрицей большой размерности. Данный этап является наиболее ресурсоемкой фазой моделирования. Решение системы производится с помощью переупорядоченного итерационного метода BiCGStab. Применена разработанная авторами вычислительная процедура умножения распределенной разреженной матрицы на распределенный вектор, оптимизированная для уменьшения времени передачи информации об элементах вектора по высокопроизводительной сети. Приводятся результаты вычислительных экспериментов.

## 1. Введение

Задачи моделирования больших пластических деформаций при высоких температурах особенно требовательны к вычислительным ресурсам. Учет существенной физической и геометрической нелинейности задачи требует многократного решения последовательностей систем линейных алгебраических уравнений (СЛАУ) с разреженными матрицами, имеющими особую структуру, которая обсуждается во втором разделе работы. Количество решений систем линейных алгебраических уравнений, возникающих во время решения одной задачи, может исчисляться тысячами. Это делает затруднительным использование сеток большой размерности, уже ставшее привычным в тех областях, где применимы линейные модели.

Современные библиотеки прямых методов решения СЛАУ для систем с общей памятью позволяют использовать в упругопластических задачах сетки, порождающие системы размерностью порядка миллионов. При этом требуется использовать дорогое вычислительное устройство коллективного пользования с сотнями гигабайт оперативной памяти, а поиск окончательного решения может занимать несколько часов. Это не только затрудняет исследование свойств моделируемых процессов, но и делает невозможным проведение многих экспериментов на существующих вычислительных системах.

Моделирование процессов деформирования тел со структурными неоднородностями, таких, как представительные объемы композитных материалов, требует использование мелкого разбиения сетки на границе раздела сред. Большие поверхности разделов сред, возникающие при комплексном моделировании сложных процессов механики деформаций и разрушения, потребуют создания сеток, порождающих СЛАУ, факторизации матриц которых не удастся разместить в разделяемой оперативной памяти. Итерационные методы позволяют не хранить факторизованную матрицу, что смягчает требования к доступному объему оперативного запоминающего устройства, однако результат их работы сильно зависит от числа обусловленности матрицы СЛАУ.

Число обусловленности матрицы жесткости в упругопластической задаче растет по мере перехода моделируемой системы в фазу пластичности, и, кроме этого, зависит от размерности матрицы. Применение итерационных решателей для упругопластических задач с достаточным для практических целей количеством элементов в конечно-элементной сетке требует как использования мощных вычислительных систем, так и правильной организации хранения матрицы, а также подбора предобуславливателя.

* Работа выполнена в рамках Программы №7 УрО РАН, проект № 15-7-1-17 при частичной поддержке РНФ (проект №14-19-01358) в части применения результатов к моделированию деформаций ММК

Для преодоления ограничений по объему оперативной памяти и времени вычисления задачи целесообразно использовать системы с распределенной памятью. Работы по переносу алгоритмов моделирования больших упругопластических деформаций на системы с распределенной памятью ведутся уже несколько лет. В частности, были созданы программы для решения упругопластических задач на гомогенных кластерных вычислителях с использованием библиотеки lis [1], исследован ряд итерационных методов решения СЛАУ, а также осуществлено сравнение применимости различных предобуславливателей. Для дальнейшего развития предназначенных для систем с распределенной памятью программных кодов, необходимо выполнить перенос существующих наработок на системы с ускорителями.

Применение итерационных алгоритмов решения СЛАУ для гетерогенных кластерных систем до сих пор не нашло широкого распространения. Единственной универсальной библиотекой такого рода, известной авторам, является AmgX [2]. Однако детали реализации данной библиотеки раскрыты неполно. Существует работа [3], посвященная реализации итерационного метода обобщенных минимальных невязок для произвольных задач с разреженными матрицами. Концепции организации вычислений, приведенные в работе [3] в некоторых аспектах схожи с использованными в данном исследовании, однако при проектировании авторы принимают ряд принципиальных решений, которые сильно разграничивают области применения разработанных подходов. Сходства и различия методов организации вычислений обсуждаются в разделе 3.

Итерационными решателями для гетерогенных кластерных систем оснащены некоторые пакеты конечно-элементного анализа, такие как ANSYS и CAE Fidesys.

## 2. Упругопластические задачи

Упругопластическая задача с большими деформациями решается последовательными шагами по приложению нагрузки [4]. На каждом шаге решения задачи методом конечных элементов вычисления проводятся в три этапа:

На первом этапе рассчитываются локальные матрицы жесткости и векторы правой части. Глобальная матрица жесткости получается суммированием расширенных локальных матриц жесткости, имеющими размерность глобальной, но содержащими лишь локальные вклады соответствующих элементов. Расширенная локальная матрица в значительной степени разрежена, и для ее хранения и передачи используется координатный формат.

На втором этапе происходит решение полученной СЛАУ. В силу природы метода конечных элементов матрица жесткости полученной СЛАУ структурно-симметрична и имеет блочно-ленточную структуру со значительной разреженностью внутри ленты и размером блока, соответствующим размерности задачи. В частности, в трехмерном моделировании возникают блоки размера 3x3. При этом сама матрица не является ни симметричной, ни положительно определенной.

На третьем этапе происходит вычисление напряженно-деформированного состояния конечных элементов в конце шага нагружения.

Для удовлетворения условию пластичности с приемлемой точностью шаги 2 и 3 выполняются 10-15 раз. При этом можно пользоваться собранной матрицей жесткости и заранее подготовленной матрицей предобуславливателя. Кроме этого возможно использовать в качестве начальной аппроксимации решения следующего шага применения нагрузки результат, полученный на предыдущем шаге.

## 2. Реализация

Основой для реализации итерационных методов решения СЛАУ в упругопластической задаче послужили эффективные структуры данных, предложенные авторами в работе [5]. Данный подход задействует специальную схему распределенного хранения вектора. При этом вектор разбивается на дискреты, называемые авторами порциями. Локально хранятся только дискреты, участвующие в операции умножения разреженной матрицы на вектор. Специальный вариант соответствующей процедуры, разработанный авторами, позволяет применять при вычислениях одновременно разреженные матрицы и разреженные векторы. Использование такого подхода

позволяет снизить количество необходимых транзакций как внутри узла, так и по высокоскоростной сети суперкомпьютера.

Альтернативным подходом, предложенным в работе [3], является использование сжатия данных вместе с перестановкой строк локальной матрицы на каждом из ускорителей. Это позволяет задействовать вычислительные примитивы из готовых библиотек для систем с единственным ускорителем, таких как CUSPARSE [6].

Каждый из подходов обладает преимуществами и недостатками. Несмотря на некоторое увеличение количества вычислений, которые необходимо произвести внутри ускорителя, подход авторов позволяет значительно сократить затраты времени работы основного процессора при передаче данных. Это позволяет использовать возможности CUDA Aware MPI с помощью технологии GPUDirect [7]. Кроме этого, подход авторов позволяет не хранить значительную часть элементов вектора на каждом из ускорителей, что позволяет органично использовать основную память ограниченного объема.

Для предложенных ранее структур данных были созданы примитивы линейной алгебры, необходимые для конструирования реализаций основанных на подпространствах Крылова методов решения СЛАУ. Экспериментальное обоснование эффективности созданных методов приводится в работе [5].

Из разработанных примитивов был собран модифицированный предобусловленный вариант стабилизированного метода бисопряженных градиентов (англ. Biconjugate gradient stabilized method, BiCGStab), предложенный Б.И. Краснопольским в работе [8]. Метод позволяет естественным образом сочетать вычисления с передачей данных по высокопроизводительной сети кластера, сокращая количество необходимых синхронизаций. В качестве критерия останова метода было выбрано соотношение $\|r_i\|/\|r_0\| < 10^{-9}$, где $\|r_i\|$ — 2-норма вектора невязки текущего шага, $\|r_0\|$ — 2-норма вектора невязки первого шага. Для того, чтобы избежать глобальной синхронизации, проверка критерия останова на каждой итерации выполнялась только один раз.

Для предобуславливания использовалась комбинация блочного метода Якоби [9], при этом для каждого из диагональных блоков строилось ILU(0) [10] разложение. Размер диагонального блока совпадал с размером порции вектора.

Операции построения матрицы предобуславливателя, а также операции над порциями вектора выполнялись на центральном процессоре. Для операций, требующих суммирования промежуточных результатов, таких, как скалярное произведение и норма векторов, использовалась неблокирующая редукция MPI. Операция умножения матрицы на вектор также была выполнена в неблокирующем асинхронном виде.

## 3. Вычислительные эксперименты

Для подтверждения эффективности предложенных подходов были проведены вычислительные эксперименты. Использовались матрицы и векторы правой части, возникающие в упругопластической задаче сжатия параллелепипеда плоскими плитами. Область задачи разбивалась гексаэдральными восьмиузловыми конечными элементами. Граничные условия включали в себя три плоскости симметрии и полное прилипание на одной из граней параллелепипеда. Наборы экспериментальных данных обозначены, как d70, d80 и d90, где число обозначает количество разбиений параллелепипеда в направлении каждой из координатных осей. Информация о наборах экспериментальных данных приводится в таблице 1.

**Таблица 1.** Характеристики наборов данных, используемых в вычислительных экспериментах

| Параметр набора данных | d70 | d80 | d90 |
|---|---|---|---|
| Количество конечных элементов | 343000 | 512000 | 729000 |
| Размерность матрицы | 1073733 | 1594323 | 2260713 |
| Количество ненулевых элементов в матрице | 82224307 | 121549113 | 173028555 |
| Объем памяти в координатном формате хранения, мегабайт | 1254 | 1854 | 2640 |

Вычислительные эксперименты проводились на гетерогенной части кластера «Уран» Института математики и механики им. Н.Н. Красовского УрО РАН. Узлы суперкомпьютера оснащены ускорителями NVIDIA Tesla M2050 и NVIDIA Tesla M2090, при этом на каждый из них установлено 8 ускорителей [11]. Для обмена сообщениями MPI используется высокопроизводительная сеть Infiniband 4x DDR. В вычислительных экспериментах использовались только узлы, оснащенные ускорителями NVIDIA Tesla M2090.

Часть кода, выполняющаяся на ускорителях, была реализована на языке CUDA C/C++. Компиляция производилась с использованием nvcc (Cuda compilation tools, release 4.1, V0.2.1221), использовались флаги компиляции -O3 -gencode code=sm_20,arch=compute_20. Остальной код, реализованный на языке C++ компилировался с помощью Intel Compiler, версия 14.0.0 20130728. Использовался флаг компиляции -O3. Операции над порциями вектора, включая BLAS уровня 1, вычисление и применение предобуславливателя выполнялись средствами Intel MKL, версия 8.0.1. Для организации обмена сообщениями использовалась MVAPICH2.

Производились замеры астрономического времени (англ. wall-clock time) $\tau$, необходимого для проведения 10 итераций BiCGStab, затем полученное время соотносилось со временем работы алгоритма на двух ускорителях $\tau_0$.

В первом наборе тестов для каждого из процессов MPI выделялся один ускоритель и одно ядро процессора. При этом для случаев с количеством процессов $p \le 8$ все вычисления выполнялись в пределах одного узла. Относительное ускорение $s = \tau_0 / \tau$ в каждом из случаев приведено в таблице 2 и изображено на рис 1. Для удобства в таблице 2 для каждого из экспериментов приводится отношение количества ускорителей $p$ к исходному количеству ускорителей $p_0 = 2$. Это число выступает в качестве теоретической оценки масштабируемости алгоритма.

**Таблица 2.** Относительное ускорение $s$ в зависимости от количества ускорителей $p$ для первой серии экспериментов

| **Набор данных** | $p$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| | $p / p_0$ | | | | | | | | |
| | **1,00** | **1,50** | **2,00** | **2,50** | **3,00** | **3,50** | **4,00** | **4,50** | **5,00** |
| **d70** | 1,00 | 1,31 | 1,66 | 1,76 | 2,00 | 2,14 | 2,15 | 1,88 | 1,89 |
| **d80** | 1,00 | 1,44 | 1,87 | 2,08 | 2,24 | 2,43 | 2,66 | 2,44 | 2,54 |
| **d90** | 1,00 | 1,77 | 2,35 | 2,46 | 2,79 | 2,87 | 3,12 | 2,95 | 2,94 |

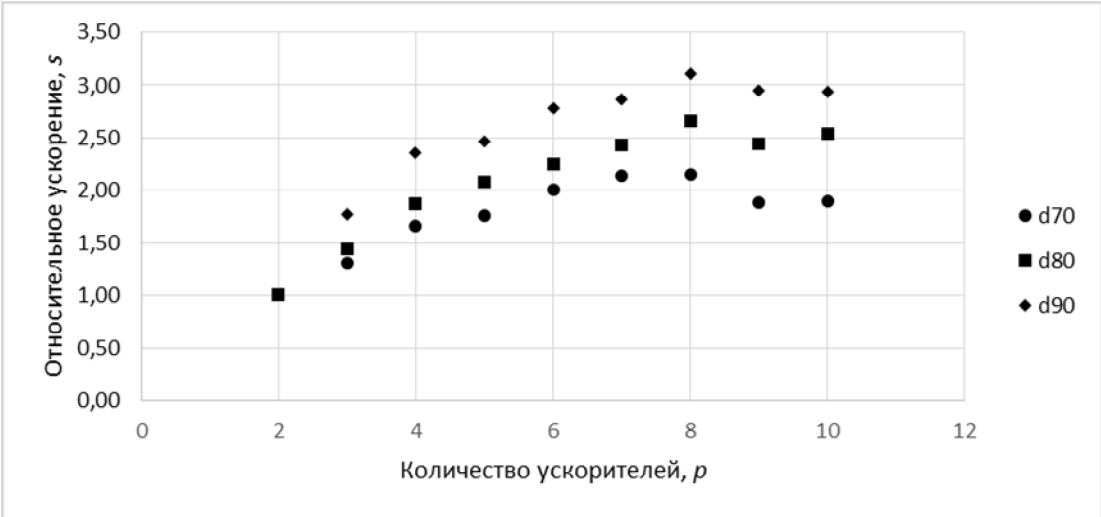


**Рис. 1**. Изменение ускорения решения $s$ в зависимости от количества ускорителей $p$ для первой серии экспериментов

Во втором наборе тестов для каждого из процессов MPI выделялся один ускоритель и два ядра процессора. При этом часть ускорителей на узле оставалась незадействованной, а процессы MPI распределялись по большему числу узлов кластерной системы. Вычислительные эксперименты проводились только для набора данных d90. Результаты экспериментов приводятся в таблице 3 и отображены на рис 2.

**Таблица 3.** Относительное ускорение $s$ в зависимости от количества ускорителей $p$ для второй серии экспериментов с набором данных d90

| $p$ | **2** | **4** | **6** | **8** | **10** | **12** |
|---|---|---|---|---|---|---|
| $p/p_0$ | 1,00 | 2,00 | 3,00 | 4,00 | 5,00 | 6,00 |
| $s$ | 1,00 | 1,57 | 1,88 | 2,70 | 3,87 | 4,41 |

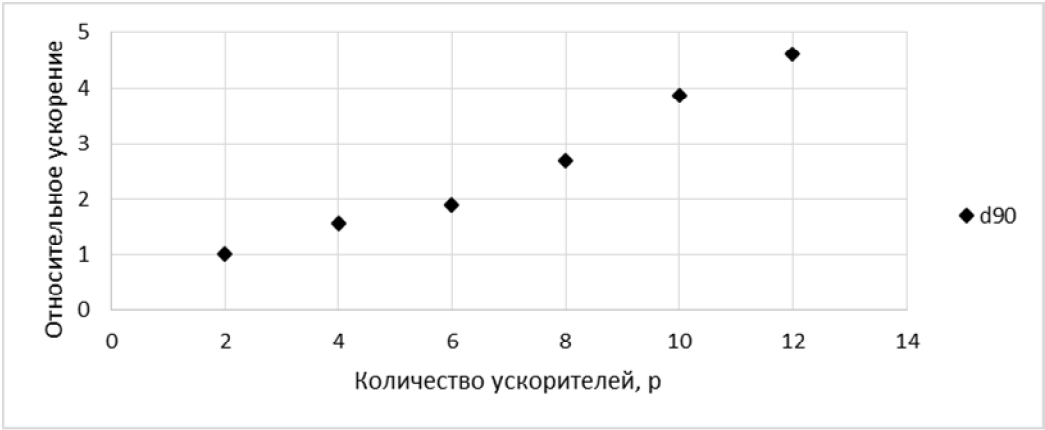


**Рис. 2**. Изменение ускорения решения $s$ в зависимости от количества ускорителей $p$ для второй серии экспериментов с набором данных d90

## 4. Обсуждение результатов

Из таблицы 2 и рис. 1 видно, что при больших значениях размерности матрицы коэффициентов системы и небольших значениях количества ускорителей относительное ускорение растет быстрее, чем увеличивается количество вычислителей. Такое поведение вызвано выбором ILU0 предобуславливателя: с уменьшением количества элементов в порции вектора уменьшается и размер диагонального блока. Это приводит к уменьшению количества элементов в матрице ILU0-предобуславливателя и ускорению решения системы уравнений с треугольной матрицей коэффициентов на шаге предобуславливания.

Выбор достаточно маленькой порции вектора позволит использовать значительно более требовательные к вычислительным ресурсам предобуславливатели, например ILUT [12]. При достаточно мелком размере блока возможно выполнение полного LU-разложения. Современные реализации алгоритмов LU-разложений предназначены для параллельного выполнения построения единственной факторизации, но не для параллельного построения множества факторизаций. Вероятно, целесообразно выбирать размер блока блочного предобуславливателя независимо от размера порции вектора.

При использовании восьми ускорителей на каждом из узлов (таблица 2 и рис. 1) насыщение достигается до того момента, как процесс начинает выходить за пределы узла. Причиной подобного поведения, вероятно, является быстрое насыщение пропускной способности шины PCI-E. Это подтверждает второй набор тестов, который показывает значительное улучшение масштабируемости и позволяет достигнуть меньшего общего времени решения. Использование четырех ускорителей на каждом узле (таблица 3 и рис. 2) позволяет задействовать в решении не менее 12 ускорителей, причем тренд роста относительного ускорения все еще не изменяется.

Для конкретного кластера, на котором проводились эксперименты, разумным подходом является использование реализаций для системы с общей памятью и ускорителями. Предложенный в [6] подход к организации вычислений подходит и для этого случая, однако реализация такого программного обеспечения не является темой данной работы. Для оптимальной ра-

боты подобных программ целесообразно выбирать вычислители со сбалансированным количеством ускорителей и сетевых карт высокопроизводительной сети на одной шине PCI-E.

## 5. Выводы

Предложенные авторами ранее подходы к организации вычислений при решении упругопластических задач на гетерогенных кластерных системах были реализованы в виде решателя, использующего конкретный итерационный метод решения СЛАУ (BiCGStab) и предобуславливатель (блочный метод Якоби с ILU0 внутри блока). Полученные результаты позволяют решать упругопластические задачи с бóльшим количеством конечных элементов, чем это было возможно ранее, при использовании гомогенных кластерных систем. Разработанные программы показывают бóльшую масштабируемость в случае, если количество ускорителей на каждом из узлов невелико. При соотношении в два ускорителя на один сокет возможно производительное использование по крайней мере 12 ускорителей.

Дальнейшим развитием исследования может стать выбор и реализация более подходящего предобуславливателя, а также адаптация программ для использования третьей версии GPUDirect. Это позволит добиться масштабирования разработанных программ на большее количество узлов, особенно при использовании кластеров с одним, двумя или четырьмя ускорителями на узел.

Предложенный подход может быть обобщен на использование различных кластерных систем с ускорителями, включая кластеры Intel Xeon Phi.

## Литература


1. Толмачев А.В., Коновалов А.В., Партин А.С. Исследование производительности ряда итерационных методов решения СЛАУ в упругопластической задаче // Программные продукты и системы 108. С 180-187
2. Eaton J. AmgX: Multi-Grid Accelerated Linear Solvers for Industrial Applications. URL: http://tinyurl.com/AmgX-pfa (дата обращения: 30.11.2014)
3. Ziane Khodja, L., et al. Parallel sparse linear solver with GMRES method using minimization techniques of communications for GPU clusters. // The Journal of Supercomputing 69(1) P. 200-224.
4. Поздеев А.А., Трусов П.В., Няшин Ю.И. Большие упругопластические деформации. М.: Наука, 1986. 232 с.
5. Халевицкий Ю.В., Коновалов А.В. Эффективные структуры данных для решения упругопластических задач на гетерогенных кластерных вычислителях // Материалы XIV Международной конференции «Высокопроизводительныепараллельные вычисления на кластерных системах», Пермь, 10-12 ноября — Пермь. 2014. С. 446-452
6. CUSPARSE LIBRARY. URL: http://docs.nvidia.com/cuda/pdf/CUSPARSE_Library.pdf (дата обращения: 01.12.2014)
7. NVIDIA GPUDirect URL: https://developer.nvidia.com/gpudirect (дата обращения: 30.11.2014)
8. Krasnopolsky, B. The reordered BiCGStab method for distributed memory computer systems // Procedia Computer Science 1(1) P. 213-218.
9. Li, R. and Y. Saad GPU-accelerated preconditioned iterative linear solvers // The Journal of Supercomputing 63(2) P. 443-466.
10. Akhunov, R. R., et al. Optimization of the ILU(0) factorization algorithm with the use of compressed sparse row format. Journal of Mathematical Sciences 191(1) P. 19-27.
11. Параллельные вычисления в УрО РАН URL: http://parallel.uran.ru/ (дата обращения: 30.11.2014)
12. Saad, Y. ILUT: A dual threshold incomplete LU factorization // Numerical Linear Algebra With Applications 1(4) P. 387-40