Реализация итерационных методов решения СЛАУ для гибридных вычислительных систем и их использование при моделировании диссипативных процессов в задачах физики плазмы*

П.А. Кучугов[1,2], Г.А. Багдасаров[1,3], А.С. Болдарев[1,3], О.Г. Ольховская[1], Ю.А. Повещенко[1,3]

Институт прикладной математики им. М.В. Келдыша РАН[1], Физический институт им. П.Н. Лебедева РАН[2], Национальный исследовательский ядерный университет "МИФИ"[3]

Многие задачи физики, в том числе физики плазмы, требуют решения систем линейных уравнений с большими разреженными матрицами. Важным в этом вопросе является возможность использования всех вычислительных ресурсов, предоставляемых современными гибридными кластерами, для минимизации временных затрат при моделировании различных диссипативных процессов. В данной работе предлагается вариант реализации некоторых итерационных методов подпространств Крылова с использованием технологий CUDA и MPI. Представлены результаты замеров производительности решения тестовых задач, относящихся к различным физическим процессам, при использовании структурированных и неструктурированных сеток. На основе этих эмпирических данных сделана оценка числа строк матрицы, оптимального для обработки конфигурацией, состоящей из одного CPU и одного GPU.

Ключевые слова: Итерационные методы, СЛАУ, математическое моделирование, физика плазмы, CUDA, MPI, гибридные вычислительные системы.

1. Введение

Потребность в проведении все более подробных численных расчетов подталкивает исследователей и разработчиков численных кодов к использованию дополнительных ресурсов, предоставляемых современными вычислительными кластерными системами, чтобы минимизировать общие временные затраты на проведение моделирования поставленных задач. В первую очередь речь идет о разного рода сопроцессорах к CPU, в частности о GPU применительно к данной статье. GPU-устройства, изначально ориентированные на решение специфических задач, связанных с обработкой и отображением графической информации, на сегодняшний день могут успешно выполнять общие массивно-параллельные вычисления (GPGPU). Наибольшее распространение для этих целей получила технология CUDA [1], позволяющая получить доступ к GPU-устройству с помощью набора функций, позиционируемая как расширение языка Си.

Хорошо известно, что при моделировании широкого спектра различных физических и не только процессов и явлений возникает необходимость решения системы линейных алгебраических уравнений (СЛАУ) с большими разреженными матрицами, что занимает большую часть вычислительного времени. Этот факт выделяет поиск эффективных с точки зрения производительности методов решения СЛАУ в отдельную задачу, которой занимается значительное число исследователей по всему миру. Подходы к решению могут быть различны. Они включают в себя как разработку новых алгоритмов решения СЛАУ, обладающих большей степенью параллелизма, так и поиск эффективной реализации в рамках одного и того же алгоритма. В данной работе авторы будут придерживаться второго под-

*Работа выполнена при частичной финансовой поддержке РФФИ, гранты № 16-31-60101-мол_а_дк и № 16-31-00350-мол_а.

хода. В качестве исследуемого метода был выбран часто используемый на текущий момент стабилизированный метод бисопряженных градиентов (BiCGStab) с предобуславливанием [2, 3], являющийся представителем итерационных решателей подпространств Крылова. Несколько слов следует сказать о выборе предобуславливателя. Предобуславливание исходной матрицы улучшает ее спектральные свойства и влияет на скорость сходимости итерационного метода, однако при параллельной реализации решателей СЛАУ также следует обращать внимание на степень параллелизма операции предобуславливания и, возможно, поступиться скоростью сходимости в разумных пределах в обмен на ощутимое снижение времени, приходящегося на одну итерацию. Из этих соображений нами был выбран полиномиальный предобуславливатель в смысле наименьших квадратов. Так, для рассматриваемых в данной работе задач переход от ранее использованного предобуславливания по методу Гаусса–Зейделя привел к увеличению числа итераций в 1.5-2 раза, что в абсолютных значениях составляет 4-5 итераций вместо 2-3. Представленный в последующих разделах анализ времени выполнения одной итерации позволяет говорить, что данный факт не приводит к существенным потерям в производительности.

Безусловно, на сегодняшний день существует достаточно широкий спектр библиотек как с открытым кодом, так и проприетарных, реализующих разнообразные итерационные методы подпространств Крылова с предобуславливанием и без, такие как CUSP [4], CULA [5], PARALUTION [6], PETSc [7], CUB [8], gpu_sparse [9], ViennaCL [10] и другие. Традиционно с коммерческими решателями сложно работать из-за невозможности внесения даже минимальных изменений, которые рано или поздно потребуются, а открытые библиотеки в своем большинстве не реализуют работу с распределенными матрицами с одновременным использованием GPU-ресурсов для локальных, т.е. относящихся к конкретному процессу, вычислений. Также требуется адаптация типов данных, используемых в основной программе, к типам данных, с которыми оперирует библиотека, что занимает дополнительное вычислительное время. Таким образом, авторы пришли к выводу о необходимости создания собственной реализации решателя СЛАУ в рамках пакета прикладных программ MARPLE3D [11], предназначенного для моделирования задач высокотемпературной плазмы и физики высоких плотностей энергии.

Наиболее ресурсоемкой и интенсивно используемой (особенно с учетом выбранного предобуславливателя) при решении СЛАУ является операция вычисления произведения разреженной матрицы на плотный вектор (SpMV). Для GPU-устройств ситуация сильно усугубляется нерегулярным доступом к памяти, осуществляемым в соответствии с портретом разреженной матрицы, который в свою очередь зависит от способа аппроксимации исходных дифференциальных уравнений в частных производных и от метода дискретного представления области моделирования, т.е. расчетной сетки. На текущий момент предложено несколько достаточно эффективных способов вычисления SpMV [12–16], позволяющих более или менее нивелировать эффект нерегулярного и некогерентного доступа к памяти. На наш взгляд на сегодняшний день ключевой стратегией реализации SpMV является подбор значений различных параметров для того или иного вида разреженных матриц и классов GPU-устройств. Также можно экспериментировать с форматами хранения матриц для организации когерентного доступа к памяти или с переупорядочиванием элементов матрицы для группировки ненулевых элементов и улучшения локализованности данных, однако, и тот, и другой способ приводят к дополнительным временным затратам, связанным с описанными операциями подготовки матрицы.

2. Программная реализация

Обратимся к описанию выполненой реализации метода BiCGStab с полиномиальным предобуславливателем в рамках программного комплекса MARPLE3D. Данный численный код естественным образом поддерживает проведение параллельных вычислений на основе

метода геометрического распараллеливания. Для разбиения области на домены используется пакет ParMetis [17]. Для решения СЛАУ без привлечения GPU-устройств используется библиотека Aztec [18], поэтому логичным представляется использование той же структуры данных для работы с распределенными матрицами, что и раньше. Таким образом, для хранения разреженной матрицы используется формат DMSR [18], в локальном представлении схожий с форматом CSR. В наиболее общем случае при каждом вычислении произведения глобальной матрицы на вектор необходимо обновлять внешние элементы вектора, относящиеся к другим MPI-процессам. Однако, выбранный алгоритм позволяет сократить до минимума обменные операции и выполнять их один раз за итерацию после нахождения следующего приближения решения.

Для локального вычисления SpMV нами была использована модификация алгоритма, предложенного в [12]. Размер блока тредов и размер группы тредов, обрабатывающих одну строку матрицы, были сделаны переменными, а также введен дополнительный параметр, определяющий число строк матрицы, обрабатываемых конкретной группой тредов. Несколько шагов по варьированию этих параметров позволяет подобрать оптимальные значения для данного GPU-устройства и класса матриц.

Для выполнения практически всех операций с векторами, таких как скалярное произведение векторов, произведение вектора на скаляр, сумма векторов и некоторых других была использована библиотека cuBLAS [19], являющаяся частью CUDA Toolkit.

Достаточно распространенной стратегией при написании CUDA-программ является стратегия по объединению нескольких вычислительных ядер в одно, где это возможно, чтобы сократить издержки на запуск ядер. Нами было замечено, что алгоритм часто использует операцию $\alpha x + \beta y + \gamma z$, поэтому было реализовано соответствующее ядро вместо двукратного вызова функции $y = \alpha x + y$ из библиотеки cuBLAS.

При реализации метода BiCGStab в данной работе также была применена оптимизация, основывающаяся на использовании ключевого слова `__restrict__` для неперекрывающихся массивов, что позволяет компилятору значительно оптимизировать код, не производя лишних проверок при выполнении программы.

3. Результаты численных расчетов

В данном разделе представлены результаты тестирования[1] выполненной программной реализации метода BiCGStab, использующей ресурсы GPU-ускорителей, на примере двух задач разной физической природы и решаемых различными численными методами. Это позволяет рассмотреть разреженные матрицы, обладающие различной структурой, не абстрагируясь от реальных приложений.

3.1. Постановки тестовых задач

3.1.1. ЭМГ-волна

В качестве первой тестовой задачи (будем обозначать T1) рассмотрим задачу о распространении ЭМГ-волны [21]. Сущность этой задачи заключается в ускоренной диффузии магнитного поля в неоднородных по плотности областях за счет токовых электронов. Аналогичный эффект возникает и в случае криволинейных линий магнитного поля в однородных областях. Используя конфигурацию, представленную на рис. 1, можно получить

[1]Все приведенные в статье результаты были получены для двойной точности с использованием гибридного кластера К-100 Института прикладной математики им. М. В. Келдыша [20]. Данная вычислительная система состоит из 64 узлов, объединенных коммуникационной системой "МВС-Экспресс", каждый из которых оснащен 2-мя процессорами Intel Xeon X5670 (6 ядер) и 3-мя видеокартами nVidia Fermi C2050.

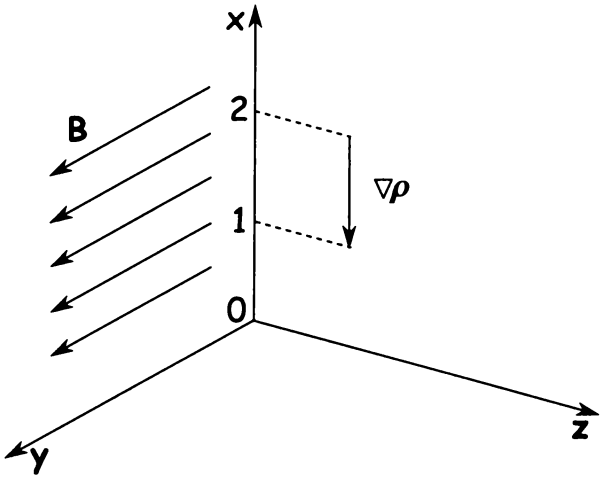


Рис. 1. Постановка задачи о распространении ЭМГ-волны.

следующее дифференциальное уравнение для диффузии:

$$\frac{\partial B}{\partial t} + \kappa B \frac{\partial B}{\partial z} = D \frac{\partial^2 B}{\partial z^2},$$

где $B \equiv B_y$ является компонентой магнитного поля, $D = c^2/4\pi\sigma$, $\kappa = (cm_i/4\pi Ze)\left(\partial n^{-1}/\partial x\right)$ или $cm_i/4\pi\rho Ze$. Здесь c – это скорость света, σ – проводимость, m_i – масса иона, Z – заряд иона, e – элементарный заряд. Для тестовых запусков были использованы следующие значения параметров: $D = 1/4\pi$, $\kappa = 8.25 \cdot 10^{-18}/\rho$, $x|y|z \in [0, 3]$ в программных единицах MARPLE3D. Распределение плотности задавалось выражением $\rho = 2\rho_0$ для $x < 1$, $\rho = 2\rho_0/x$ для $1 < x < 2$ и ρ_0 в остальных случаях, $\rho_0 = 10^{-18}$. Начальные и граничные условия следующие:

$$B(z_{min}, t) = 1/\sqrt{4\pi}, \quad B(z_{max}, t) = 0, \quad \left.\frac{\partial B}{\partial s}\right|_{\partial} = 0, s = x|y.$$

Построение дискретного аналога исходного дифференциального уравнения выполнялось с использованием проекционной схемы [22]. Тестовые расчеты проводились на ортогональной сетке, состоящей из прямоугольных параллелепипедов.

3.1.2. Тепловая волна

В качестве второй тестовой задачи (T2) рассмотрим задачу о распространении тепловой волны в среде с нелинейным коэффициентом теплопроводности [23]. Соответствующее дифференциальное уравнение имеет вид:

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial s}\left(\kappa_0 \cdot T^{\alpha} \cdot \frac{\partial T}{\partial s}\right),$$

где T – это неизвестная температура, κ_0, α – произвольные коэффициенты, $s \equiv x\,|\,y\,|\,z$. Начальные и граничные условия задаются выражениями:

$$T(0,\ t) = \left[\frac{\alpha D}{\kappa_0}\left(Dt + s_0\right)\right]^{\frac{1}{\alpha}}, t > 0, \quad \text{и} \quad T(s,\ 0) = \begin{cases} \left[\dfrac{\alpha D}{\kappa_0}\left(s_0 - s\right)\right]^{\frac{1}{\alpha}}, s \leqslant s_0, \\ 0, s > s_0, \end{cases}$$

где D – скорость тепловой волны, s_0 – произвольный параметр.

Для аппроксимации данного дифференциального уравнения использовался метод опорных операторов [24,25]. Тестовые запуски проводились для значений параметров $\alpha = 2.0$, $\kappa_0 = 0.5$, $s_0 = 0.5$, $D = 5.0$ и $s \in [0, 3]$ с использованием структурированных и неструктурированных разностных сеток и постоянного шага по времени $\tau = 2 \cdot 10^{-4}$.

3.2. Результаты

На первом этапе измерений на примере задачи T1 был проведен подбор параметров ядра, вычисляющего SpMV. Была использована сетка $100 \times 100 \times 10$, что в результате привело к глобальной матрице с числом строк $N = 3 \cdot 10^5$ и числом ненулевых элементов $NNZ \approx 2.2 \cdot 10^7$, которые оказались сгруппированными вокруг нескольких диагоналей. Отметим, что и для задачи T2 даже при использовании неструктурированной сетки, состоящей из тетраэдров, портрет матрицы меняется незначительно. Это верно и для других задач, описывающих диссипативные процессы, что позволяет распространить полученные результаты на весь класс задач. Итак, варьировались значения трех основных параметров – размера блока (`BLOCK_SIZE`), размера группы тредов, обрабатывающих одну строку матрицы (`GROUP_SIZE`), и количество строк, обрабатываемых каждой группой тредов (`ROWS_COUNT`).

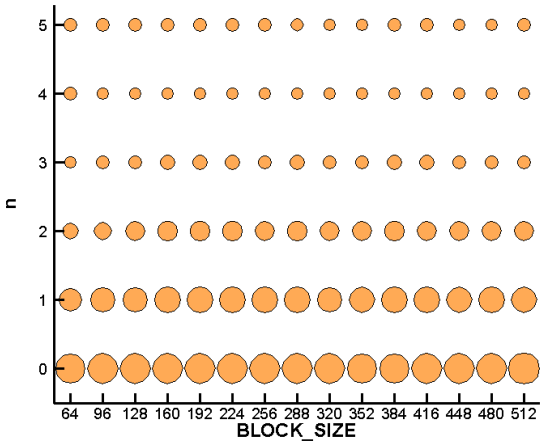


Рис. 2. Зависимость времени вычисления произведения разреженной матрицы на вектор (диаметр кругов) от размера блока и числа тредов в группе, где `GROUP_SIZE` $= 2^n$, при `ROWS_COUNT` $= 1$.

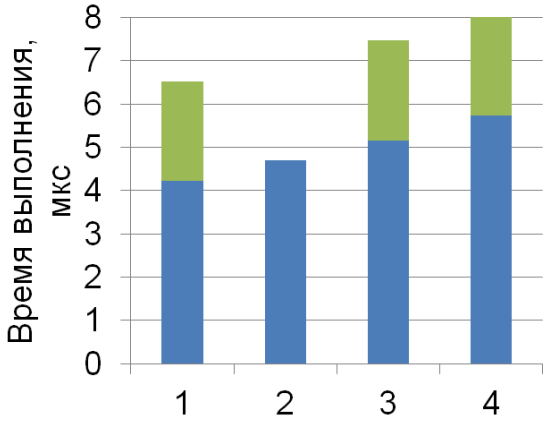


Рис. 3. Время вычисления SpMV (мкс) в различных реализациях: основное время + время, затрачиваемое на преобразование одного формата хранения разреженной матрицы в другой. 1 – CUSPARSE, 2 – собственная реализация, 3 – CUSP, 4 – CUB.

На рис. 2 схематически представлена зависимость времени вычисления SpMV от размера блока и размера группы при `ROWS_COUNT` $= 1$. Чем меньше диаметр круга, тем более эффективно выполняется данная операция на GPU. При этом видно, что в широком диапазоне размеров блока тредов ситуация, когда число тредов в группе равно размеру варпа, т.е. `GROUP_SIZE` $= 32$, соответствующая основополагающей работе [12], не является наилучшей. Наименьшее время соответствует `GROUP_SIZE` $= 16$. Увеличение значения параметра

`ROWS_COUNT` приводит к уменьшению размера сетки блоков, что может отрицательно сказываться на загруженности GPU-устройства при достаточно больших значениях. Однако, при варьировании параметра от 1 до 32 тесты показали, что время выполнения операции SpMV меняется слабо.

Также было проведено сравнение времени вычисления SpMV в наилучшем нашем варианте с некоторыми сторонними библиотеками. Результаты представлены на рис. 3. Видно, что эффективность нашей реализации незначительно уступает CUSPARSE [26] и несколько превосходит CUSP и CUB, но в сочетании с отсутствием необходимости проведения какого-либо преобразования матрицы из одного формата в другой, существенно лучше выглядит на фоне всех рассмотренных реализаций.

Дальнейшие исследования были направлены на определение ускорения и эффективности реализованного итерационного метода на примере задач T1 и T2. Для вычисления ускорения использовалось выражение $S = t_N / T_N$, а для эффективности – $E = T_1 / (N T_N)$, где t_N – время выполнения одной итерации решателя N CPU-процессами, T_N - то же, только с дополнительным использованием ресурсов одного GPU-устройства для каждого CPU-процесса. Соответствующие зависимости представлены на рис. 4 и 5.

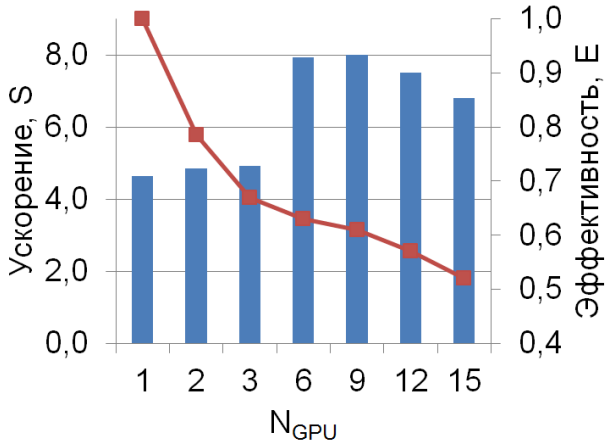
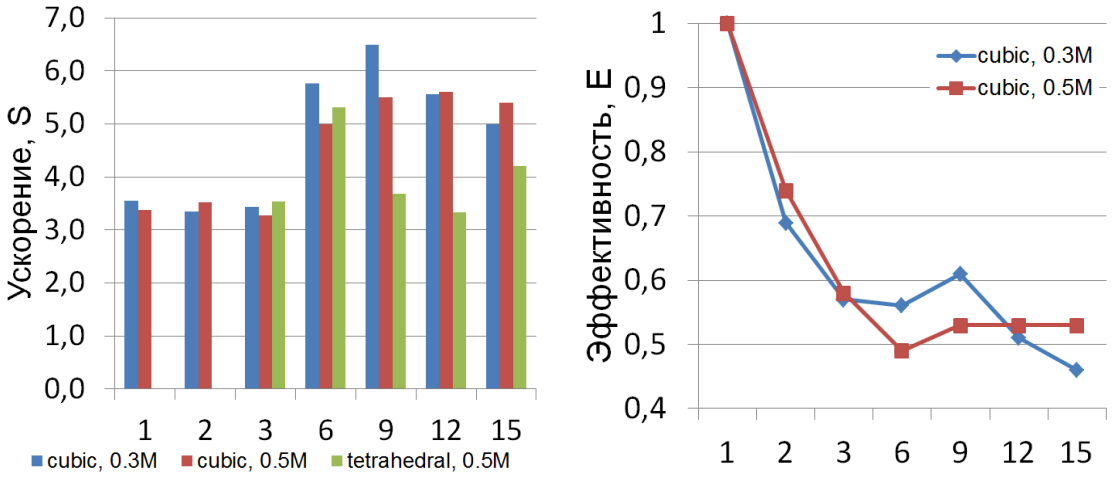


Рис. 4. Ускорение S и эффективность при фиксированном размере задачи E для задачи T1 в зависимости от числа используемых GPU-устройств.



(а) (б)

Рис. 5. Ускорение S (а) и эффективность при фиксированном размере задачи E (б) для задачи T2 в зависимости от числа используемых GPU-устройств при использовании различных сеток.

Видно, что наилучшее ускорение достигается в случае, когда одно GPU-устройство обраба-

тывает от 30 до 50 тысяч строк глобальной матрицы. Сравнивая рис. 4 и рис. 5а также можно отметить, что мы получаем более высокое ускорение в случае большего числа ненулевых элементов, в среднем приходящихся на одну строку матрицы. На наш взгляд это связано с лучшей загруженностью GPU-устройства. Наблюдаемое снижение эффективности выполнения одной итерции при увеличении числа используемых GPU-устройств в основном определяется накладными расходами на обменные операции между MPI-процессами.

3.3. Планируемые исследования

Несмотря на то, что мы постарались до минимума сократить количество обменов между CPU и GPU, они все еще занимают значительные временные ресурсы, особенно на старте решателя на каждой новой временной итерации, практически сводя на нет достигнутое ускорение. Здесь можно идти по пути переноса большего количества вычислений на GPU, например, процедуры заполнения матрицы СЛАУ. С выходом CUDA 6.0 еще одна возможность заключается в использовании так называемой Unified Memory, которая создает пул общей для CPU и GPU памяти, объекты из которой доступны по одним и тем же указателям. Ключевым моментом является то, что система автоматически обменивается такими данными между CPU и GPU.

Еще одной перспективной доработкой является перенос части вычислений, относящихся к итерационному решателю, на CPU и огранизация балансировки нагрузки между CPU и GPU. Это позволит исключить простой CPU при выполнении GPU вычислений, а также использовать все имеющиеся на узле ядра вне зависимости от числа доступных GPU-устройств.

4. Заключение

В работе проведена оптимизация вычисления произведения разреженной матрицы на вектор на одном GPU-устройстве для класса матриц, получаемых при моделировании диссипативных процессов в 3D геометрии. Рассмотрен подход к реализации итерационных методов решения СЛАУ на гибридных вычислительных системах. Соответствующая реализация метода BiCGStab с полиномиальным предобуславливателем продемонстрировала на тестовых задачах среднее ускорение до 8 раз за счет использования ресурсов GPU-ускорителей. При этом было показано, что пиковые значения ускорения достигаются в случае, когда на одно GPU-устройство приходится от 30 до 50 тысяч строк исходной матрицы СЛАУ.

Литература

1. NVIDIA Corp. CUDA C Programming Guide. Design Guide, PG-02829-001_v7.0. 2015.
2. Saad Y. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003. 537 P.
3. van der Vorst H. Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, 2009. 236 P.
4. Dalton S., Bell N., Olson L., Garland M. Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations. URL: http://cusplibrary.github.io/ (дата обращения: 01.02.2016).
5. Humphrey J. R., Price D. K., Spagnoli K. E., Paolini A. L., Kelmelis E. J. CULA: Hybrid GPU Accelerated Linear Algebra Routines // Modeling and Simulation for Defense Systems and Applications V, SPIE Proceedings. 2010. Vol. 7705. P. 770502.

6. PARALUTION Labs. User manual, PARALUTION Version 1.1.0. URL: https://www.paralution.com/user-manual/ (дата обращения: 01.02.2016).

7. Balay S., Abhyankar S., Adams M. F., Brown J., Brune P., Buschelman K., Dalcin L., Eijkhout V., Gropp W. D., Kaushik D., Knepley M. G., McInnes L. C., Rupp K., Smith B. F., Zampini S., Zhang H. PETSc Web page. URL: http://www.mcs.anl.gov/petsc/ (дата обращения: 01.02.2016).

8. CUB Web page. URL: https://nvlabs.github.io/cub/ (дата обращения: 01.02.2016).

9. Губайдуллин Д. А., Никифоров А. И., Садовников Р. В. Библиотека gpu_sparse для численного решения задач механики сплошных сред на гибридной вычислительной системе // Вестник Нижегородского университета им. Н. И. Лобачевского. Информационные технологии. 2011. № 2(1). С. 190–196.

10. ViennaCL Web page. URL: http://viennacl.sourceforge.net/ (дата обращения: 01.02.2016).

11. Гасилов В. А., Болдарев А. С., Дьяченко С.В., Ольховская О. Г., Карташева Е. Л., Болдырев С. Н., Гасилова И. В., Бояров М. С., Шмыров В. А. Пакет прикладных программ MARPLE3D для моделирования на высокопроизводительных ЭВМ импульсной магнитоускоренной плазмы // Математическое моделирование. 2012. Т. 24. № 1. С. 55–87.

12. Bell N., Garland M. Efficient Sparse Matrix-Vector Multiplication on CUDA. Technical Report NVR-2008-004, NVIDIA Corporation. 2008.

13. Guo P., Wang L. Auto-Tuning CUDA Parameters for Sparse Matrix-Vector Multiplication on GPUs // In Computational and Information Sciences (ICCIS'10). 2010. P. 1154–1157.

14. Sørensen H. H. B., High-Performance Matrix-Vector Multiplication on the GPU // Euro-Par 2011: Parallel Processing Workshops. 2012. Vol. 7155. P. 377–386.

15. Reguly I., Giles M., Efficient sparse matrix-vector multiplication on cache-based GPUs // Innovative Parallel Computing (InPar). 2012. P. 1–12.

16. Xu S., Xue W., Lin H. X. Performance modeling and optimization of sparse matrix-vector multiplication on NVIDIA CUDA platform // The Journal of Supercomputing. 2013. No. 63(3). P. 710–721.

17. Karypis G., Kumar V. Parmetis: Parallel graph partitioning and sparse matrix ordering library, version 4.0. Technical report, University of Minnesota, Department of Computer Science and Engineering. 2013.

18. Tuminaro R. S., Heroux M., Hutchinson S. A., Shadid J. N., Official Aztec User's Guide, SAND99-8801J, 1999.

19. NVIDIA Corp. CUBLAS Library. User guide, DU-06702-001_v7.0. 2015

20. Гибридный вычислительный кластер K-100. URL: http://www.kiam.ru/MVS/resourses/k100.html (дата обращения: 01.02.2016).

21. Kingsep A. S., Mokhov Y. V., Chukbar K. V. Nonlinear skin phenomena in plasma // Proceedings of the 2nd International Workshop on Nonlinear and Turbulent Processes in Physics (Kiev, USSR, 10-25 October 1983), 1983. Vol. 1. P. 267–275.

22. Багдасаров Г. А. Трехмерное моделирование магнитоускоренной импульсной плазмы с учетом эффектов, обусловленных обобщенным законом Ома. Диссертация на соискание степени кандидата физико-математических наук, Институт прикладной математики им. М. В. Келдыша РАН. 2012.

23. Самарский А. А., Соболь И. М. Примеры численного расчета температурных волн // Журнал вычислительной математики и математической физики. 1963. Т. 3. № 4. С. 702–719.

24. Самарский А. А., Колдоба А. В., Повещенко Ю. А., Тишкин В. Ф., Фаворский А. П. Разностные схемы на нерегулярных сетках. ЗАО "Критерий", 1996. 274 С.

25. Гасилов В. А., Гасилова И. В., Клочкова Л. В., Повещенко Ю. А., Тишкин В. Ф. Разностные схемы на основе метода опорных операторов для задач динамики флюидов в коллекторе, содержащем газогидраты // Журнал вычислительной математики и математической физики. 2015. Т. 55. № 8. С. 1341–1355.

26. NVIDIA Corp. CUSPARSE Library, DU-06709-001_v7.0. 2015.

The implementation of iterative methods for solving linear systems on heterogeneous computing systems and their application in modeling of dissipative processes in problems of plasma physics

P.A. Kuchugov[1,2], G.A. Bagdasarov[1], A.S. Boldarev[1], O.G. Ol'khovskaya[1], Yu.A. Poveshchenko[1]

Keldysh Institute of Applied Mathematics RAS[1], P.N. Lebedev Physical Institute RAS[2]

Many problems in physics, including physics of plasmas require solving systems of linear equations with large sparse matrices. The use of all computing resources provided by modern heterogeneous clusters is important in this issue to minimize overall time spent on modeling of various dissipative processes. This paper proposes realization of some Krylov subspace iterative methods with using CUDA and MPI. The results of the performance measurements on test problems related to different physical processes, using structured and unstructured grids are presented. On the basis of these empirical data it has been estimated optimal number of rows to be processed by the configuration consisting of one CPU and one GPU.

Keywords: Iterative methods, linear systems, mathematical modeling, plasma physics, CUDA, MPI, heterogeneous computing systems.

References

1. NVIDIA Corp. CUDA C Programming Guide. Design Guide, PG-02829-001_v7.0. 2015.
2. Saad Y. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, 2003. 537 P.
3. van der Vorst H. Iterative Krylov Methods for Large Linear Systems. Cambridge University Press, 2009. 236 P.
4. Dalton S., Bell N., Olson L., Garland M. Cusp: Generic Parallel Algorithms for Sparse Matrix and Graph Computations. URL: http://cusplibrary.github.io/ (accessed: 01.02.2016).
5. Humphrey J. R., Price D. K., Spagnoli K. E., Paolini A. L., Kelmelis E. J. CULA: Hybrid GPU Accelerated Linear Algebra Routines // Modeling and Simulation for Defense Systems and Applications V, SPIE Proceedings. 2010. Vol. 7705. P. 770502.
6. PARALUTION Labs. User manual, PARALUTION Version 1.1.0. URL: https://www.paralution.com/user-manual/ (accessed: 01.02.2016).
7. Balay S., Abhyankar S., Adams M. F., Brown J., Brune P., Buschelman K., Dalcin L., Eijkhout V., Gropp W. D., Kaushik D., Knepley M. G., McInnes L. C., Rupp K., Smith B. F., Zampini S., Zhang H. PETSc Web page. URL: http://www.mcs.anl.gov/petsc/ (accessed: 01.02.2016).
8. CUB Web page. URL: https://nvlabs.github.io/cub/ (accessed: 01.02.2016).
9. Gubaydullin D. A., Nikiforov A. I., Sadovnikov R. V. Biblioteka gpu_sparse dlya chislennogo resheniya zadach mekhaniki sploshnykh sred na gibridnoy vychislitel'noy sisteme [gpu_sparse library for the numerical solution of continuum mechanics problems on hybrid computing system] // Vestnik Nizhegorodskogo universiteta

im. N. I. Lobachevskogo. Informatsionnye tekhnologii [Bulletin of the Lobachevsky University]. 2011. No. 2(1). P. 190–196.

10. ViennaCL Web page. URL: http://viennacl.sourceforge.net/ (accessed: 01.02.2016).

11. Gasilov V. A., Boldarev A. S., D'yachenko S. V., Olkhovskaya O. G., Kartasheva E. L., Boldyrev S. N., Bagdasarov G. A., Gasilova I. V., Boyarov M. S., Shmyrov V. A. Program package MARPLE3D for simulation of pulsed magnetically driven plasma using high performance computing // Matem. Mod. 2012. Vol. 24. No. 1. P. 55–87.

12. Bell N., Garland M. Efficient Sparse Matrix-Vector Multiplication on CUDA. Technical Report NVR-2008-004, NVIDIA Corporation. 2008.

13. Guo P., Wang L. Auto-Tuning CUDA Parameters for Sparse Matrix-Vector Multiplication on GPUs // In Computational and Information Sciences (ICCIS'10). 2010. P. 1154–1157.

14. Sørensen H. H. B., High-Performance Matrix-Vector Multiplication on the GPU // Euro-Par 2011: Parallel Processing Workshops. 2012. Vol. 7155. P. 377–386.

15. Reguly I., Giles M., Efficient sparse matrix-vector multiplication on cache-based GPUs // Innovative Parallel Computing (InPar). 2012. P. 1–12.

16. Xu S., Xue W., Lin H. X. Performance modeling and optimization of sparse matrix-vector multiplication on NVIDIA CUDA platform // The Journal of Supercomputing. 2013. No. 63(3). P. 710–721.

17. Karypis G., Kumar V. Parmetis: Parallel graph partitioning and sparse matrix ordering library, version 4.0. Technical report, University of Minnesota, Department of Computer Science and Engineering. 2013.

18. Tuminaro R. S., Heroux M., Hutchinson S. A., Shadid J. N., Official Aztec User's Guide, SAND99-8801J, 1999.

19. NVIDIA Corp. CUBLAS Library. User guide, DU-06702-001_v7.0. 2015

20. Gibridnyy vychislitel'nyy klaster K-100 [Heterogeneous computing cluster K-100]. URL: http://www.kiam.ru/MVS/resourses/k100.html (accessed: 01.02.2016).

21. Kingsep A. S., Mokhov Y. V., Chukbar K. V. Nonlinear skin phenomena in plasma // Proceedings of the 2nd International Workshop on Nonlinear and Turbulent Processes in Physics (Kiev, USSR, 10-25 October 1983), 1983. Vol. 1. P. 267–275.

22. Bagdasarov G. A. Tryekhmernoe modelirovanie magnitouskorennoy impul'snoy plazmy s uchyetom effektov, obuslovlennykh obobshchyennym zakonom Oma [Three-dimensional modeling of magnetically accelerated pulsed plasma, taking into account effects due to the generalized Ohm's law]. PhD thesis, Keldysh Institute of Applied Mathemtics. 2012.

23. Samarskii A. A., Sobol' I. M. Examples of numerical calculation of temperature waves // Zh. Vychisl. Mat. Mat. Fiz. 1963. Vol. 3. No. 4. P. 702–719.

24. Samarskii A. A., Koldoba A. V., Poveshenko Yu. A., Tishkin V. F., Favorskii A. P. Raznostnye skhemy na neregulyarnykh setkakh [Difference Schemes on Irregular Meshes]. JSC "Kriteriy", 1996. 274 P.

25. Gasilov V. A., Gasilova I. V., Klochkova L. V., Poveshchenko Yu. A., Tishkin V. F. Difference schemes based on the support operator method for fluids dynamics problems in a collector containing gas hydrates // Zh. Vychisl. Mat. Mat. Fiz. 2015. Vol. 55. No. 8. P. 1341–1355.

26. NVIDIA Corp. CUSPARSE Library, DU-06709-001_v7.0. 2015.