# Алгоритмы решения некоторых прикладных задач на многопроцессорных системах с применением параллельных технологий MPI, OpenCL и CUDA*

А.С. Айриян[1], Э.А. Айрян[1], С.А. Багинян[1], Я. Буша[2], Я. Буша (мл.)[2], Л. Вальова[1,2], И.А. Гогин[3], М.И. Зуев[1], В.В. Иванов[1], О.И. Стрельцова[1]

[1]Лаборатория информационных технологий ОИЯИ, Дубна

[2]Технический университет г. Кошице, Словакия

[3]МИРЭА, Москва

При решении широкого круга вычислительных задач возникает проблема существенного сокращения расчетного времени, которая может быть решена распараллеливанием вычислений на многоядерных и/или многопроцессорных системах, а также с использованием графических процессоров. В частности, распараллеливание вычислений при решении уравнений математической физики методом сеток [1] для различных прикладных задач является актуальной задачей [2]. Наиболее распространенными подходами, применяемыми для параллельных вычислений на таких системах, являются технология Message Passing Interface (MPI) [3], стандарт Open Computing Language (OpenCL) [4] и программно-аппаратная архитектура Compute Unified Device Architecture (CUDA) [5].

В работе [6], совместно с сотрудниками Технического университета (Кошице, Словакия), представлена реализация на OpenCL алгоритма вычисления доступной площади поверхности и объема макромолекулы с учетом внутримолекулярных полостей. Разработанная программа предназначена для решения задач молекулярного моделирования и основана на модификациях алгоритма, позволяющих использовать возможности параллельных вычислений на OpenCL. К преимуществам созданной программы можно отнести ее универсальность: вычисления можно проводить на всех устройствах, поддерживающих стандарт OpenCL. Ранее данный алгоритм, использующий стереографическое проектирование сфер на плоскость, был реализован на языке FORTRAN. Проведено сравнение результатов работы этих программ и показаны преимущества реализации алгоритма на графическом процессоре NVIDIA GeForce GTX285 [7].

В таблице 1 приведено сравнение расчетного времени на графической карте (GPU) и на центральном процессоре (CPU). Из таблицы видно, что при небольшом числе атомов вычисления на GPU идут заметно медленнее по сравнению с CPU. С ростом числа атомов GPU позволяет добиться существенного выигрыша во времени вычислений по сравнению с CPU.

Таблица 1: Сравнение времени вычислений (20 запусков)

| Протеин | Число сфер | Расчетное время, сек GPU | CPU |
|---|---|---|---|
| s26a | 26 | 10.634 | 1.493 |
| s52a | 52 | 50.545 | 13.861 |
| 1j4m | 236 | 38.011 | 23.865 |
| 2lyz | 1001 | 42.814 | 178.062 |
| 2brd | 1738 | 44.584 | 384.100 |
| 1rr8 | 5003 | 44.716 | 694.965 |

В работе [8] разработан параллельный алгоритм для решения волновых уравнений в одномерном случае. Дискретизация методом конечных разностей по временной и пространственной переменным на каждом временном слое приводит к системе линейных алгебраических уравнений (СЛАУ) ленточной структуры. Параллельное решение полученных СЛАУ проводилось путем разбиения системы на блоки, число которых выбирается равным числу процессоров многопроцессорной системы [9].

В [8, 10] разработаны программы для параллельного решения начально-краевых задач для неоднородного волнового уравнения и квазилинейного волнового уравнения с применением технологии MPI. Вычисления проводились на кластере параллельных вычислений Центрального информационно-вычислительного комплекса (ЦИВК) ОИЯИ и на кластере `alicepc100.jinr.ru` – `alicepc103.jinr.ru`. Проведен анализ эффективности параллельных вычислений в зависимости от размерности сеточной задачи и числа процессоров.

Были разработаны параллельные алгоритмы решения задачи Дирихле для двумерного уравнения Пуассона, которая определяется как задача нахождения функции $u(x, y)$, удовлетворяющей в области $D$ уравнению

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -f(x, y), \quad (x, y) \in D \qquad (1)$$

*Работа поддержана грантами РФФИ №10-01-00467, №11-01-00278

и принимающей значение $g(x,y)$ на границе $D_0$ области $D$:

$$u(x,y) = g(x,y), \quad (x,y) \in D_0, \tag{2}$$

где $f(x,y)$ и $g(x,y)$ – заданные в $D$ и на $D_0$ соответственно функции. Задача (1,2) рассмотрена в прямоугольной области $D = \{(x,y) : a \le x \le b, a \le y \le b\}$, для которой строилась равномерная сетка по обеим переменным с шагом $h = (b-a)/N$, где $N$ – число узлов по каждому направлению, $N \times N$ – общее число узлов сетки (размерность задачи).

Разностное уравнение, получаемое после дискретизации задачи (1,2) методом конечных разностей, решалось разными итерационными методами: методом простой итерации, методом Гаусса-Зейделя и методом верхней релаксации. Разработана соответствующая программа на языке C/C++ с применением технологии параллельного программирования MPI [11, 12].

Схема организации параллельных вычислений с применением технологии MPI представлена на рис. 1.

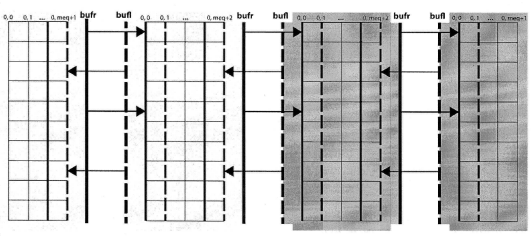


Рис. 1: Схема обмена данными между процессорами при параллельном решении задачи с применением технологии MPI

Параллельно вычислялись блоки размерности $N \times Pr$, где $Pr$ – число процессоров. Особо отметим организацию обмена данными между блоками (изображены на рисунке стрелками): пересылаются только крайние столбцы блоков с предварительной буферизацией и синхронизацией во избежание непроизвольного перезаписывания данных до момента их передачи соседним блокам.

Проведены многочисленные вычислительные эксперименты и выполнен анализ эффективности параллельных вычислений. На рис. 4 представлены зависимости расчетного времени и ускорения, как отношения расчетного времени решения задачи $T_1$ на одном процессоре к расчетному времени решения той же задачи $T_{Pr}$ на системе из $Pr$ процессоров.

Из рисунка видно, что для всех значений размерности матрицы (сетки) расчетное время сокращается с увеличением числа используеых процессоров. Так, например, для массива размерностью $800 \times 800$ ускорение расчетов достигает 9 раз при использовании 10 процессоров.

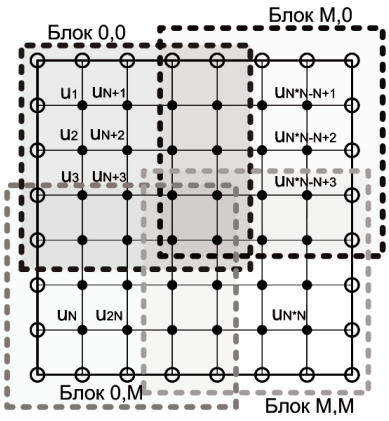


Рис. 2: Схема разбиения неизвестных на блоки при вычислениях на GPU

Для решения задачи (1,2) написана программа с использованием технологии CUDA [11, 12]. В качестве примера, на рис. 2 приведена схема разбиения на блоки двумерного массива неизвестных $u_{ij} = u(x_i, y_j)$, размерности $N \times N$, при использовании двумерной иерархии блоков, размерности $M \times M$. Пунктирной линией показана область элементов, которую каждый блок использует для подсчетов своих компонентов. Все граничные элементы области в каждом блоке не подсчитываются (они считаются в соседних блоках или заданы граничными условиями), блоки к ним обращаются для подсчета своих компонентов. Переопределение элементов матрицы происходит параллельно после очередного приближения.

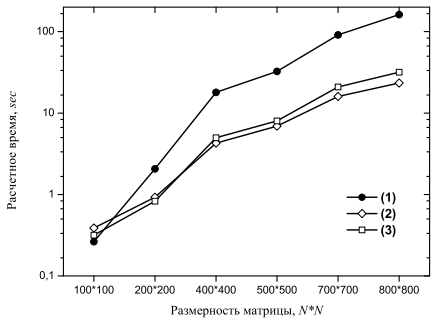


Рис. 3: График зависимости расчетного времени от размерности матрицы при решения задачи методом верхней релаксации

Программы с применением технологии CUDA тестировались и запускались на рабочей станции (AMD Phenom II X6 1055T, 8Gb RAM, NVIDIA GeForce GTX 480). Организация вычислений с использованием GPU основывалась на работе с двумя типами памяти (глобальной и разделяемой [5]) и с разными схемами ограничизации параллельных вычислений.

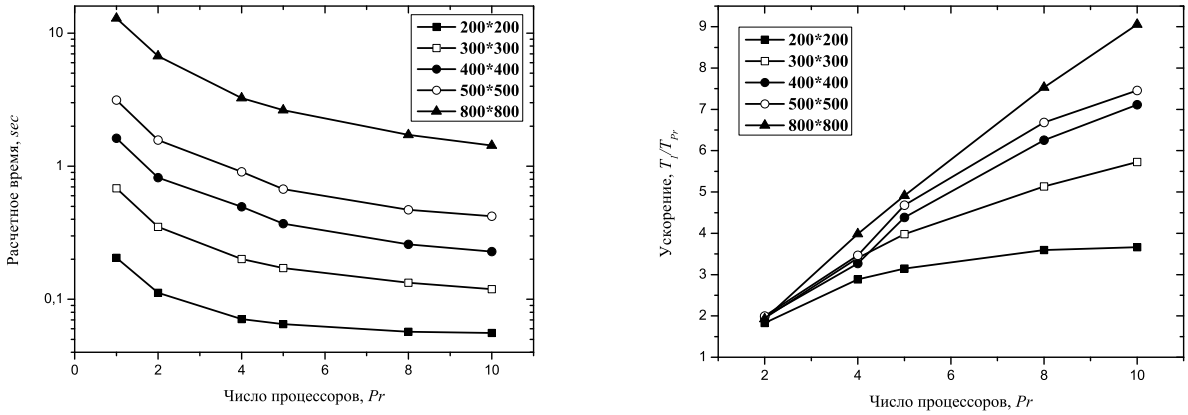


Рис. 4: Графики зависимости расчетного времен (слева) и ускорения расчетов (справа) от числа процессоров, построенные для различных значений размерности матрицы, при вычислениях с применением технологии MPI

Вычисления проводились для различных значений $N$. Был проведен анализ зависимости расчетного времени от размерности блоков и получено, что оптимальной является размерность блока в $8 \times 8$ потоков. Это объясняется тем, что число одновременно исполняемых блоков в CUDA варьируется в зависимости от их размерности: можно запустить 1024 потока одним блоком или 32 блока по 32 потока. Однако в первом случае расчет будет проходить с большими задержками, потому что 1024 потока будут считаться в 32 варпа (группа из 32 потоков, являющаяся минимальным объемом данных, обрабатываемым мультипроцессорами CUDA). Из-за этого разработчики не рекомендуют использовать большие размерности блоков [5].

На рис. 3 представлены графики зависимости расчетного времени от размерности массивов. Для сравнения на рисунке показано время вычисления программы в однопроцессорном режиме (1) и время вычислений на GPU с использованием глобальной (2) и разделяемой (3) памяти. Из рисунка видно, что расчетное время задачи уменьшилось в 7 раз при использовании глобальной памяти и в 5 раз — при использовании разделяемой памяти для массива размерностью $800 \times 800$.

Из полученных результатов можно сделать следующие выводы:

- Ускорение параллельных вычислений растет при увеличении размерности задачи для всех проведенных расчетов. Отметим, что при необходимости решения задач для больших размерностей может быть достигнуто и большее ускорение вычислений. При достаточно больших размерностях сетки распараллеливание вычислений позволяет существенно сократить расчетное время.
- Разработанные подходы к параллелизации итерационных процессов для вычислений на многоядерных и/или многопроцессорных системах и с использованием графических процессоров могут быть применены при решении широкого круга задач и перенесены на гибридные вычислительные системы (CPU+GPU).

## Список литературы


[1] Самарский А.А. Теория разностных схем. Издание третье, исправленное. Москва. Издательство Наука, 1989, 616 стр.

[2] Ayriyan A., Ayryan E., Donets E., Pribiš J. Numerical Simulation of Heat Conductivity in Composite Object with Cylindrical Symmetry. Proceedings of the International Conference MMCP 2011, "Mathematical Modeling and Computational Science", Lecture Notes in Computer Science, vol. 7125, 2011, pp. 264–269 (in print).

[3] Лупин С.А., Посыпкин М.А. Технологии параллельного программирования. – М.: ИД "ФОРУМ". ИНФРА-М. 2008, 208 стр.

[4] The OpenCL Specification. Khronos OpenCL Working Group, Editor: Aaftab Munshi, 2011.

[5] Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. М: ДМК Пресс, Москва, 2010, 232 стр.

[6] Buša (Jr.) Ján, Buša Ján, Hayryan Edik. Using GPU for Scientific Computations. International Conference on Applied Electrical Engineering and Informatics 2010, Editor: Liberios Vokorokos, 2010, pp. 112–116.

[7] JINR NEWS (2011) No. 1, p. 8.

[8] Зуев М.И. Дипломный проект. Москва, МИРЭА. Руководитель С.А. Багинян, 2010.

[9] Austin T.M., Berndt M., Moulton D. A Memory Efficient Parallel Tridiagonal Solver, Preprint LA-VR-03-4149, 2004.

[10] Зуев М.И. и др. Параллельные алгоритмы решения трехточечных разностных уравнений. XV научная конференция молодых ученых и специалистов. Труды конференции, 2011, стр. 236-239.

[11] Гогин И.А. Дипломный проект. Москва, МИРЭА. Руководитель С.А. Багинян, 2011.

[12] Zuev M., at al. Two-Dimensional Poisson Equation on Multiprocessor Systems with MPI and CUDA Technologies. MMCP2011. Book of Abstracts. FEEI TU Kosice, 2011. p. 79.