PARALLELIZING THE SOLUTION OF THE NONLINEAR HEAT CONDUCTION PROBLEM WITH THE APPLICATION OF THE OPENCL LIBRARY

L. F. Spevak, O. A. Nefedova*

Institute of Engineering Science, Ural Branch of the Russian Academy of Sciences,
34 Komsomolskaya St., Ekaterinburg, Russian Federation

*Corresponding author. E-mail: nefedova@imach.uran.ru;
address for correspondence: 34, ul. Komsomolskaya, Ekaterinburg, Russian Federation.
Tel.: +7 (343) 375 35 92; fax: +7 (343) 3745330

The paper deals with the application of parallel computation methods to the numerical solution of the nonlinear boundary value problem for the degenerate two-dimensional differential heat conduction equation. The nonlinearity of the problem stems from the power dependence of the thermal conductivity coefficient on temperature. The solution algorithm is based on the boundary element method with the application of the dual reciprocity method enabling all the computations to be brought to the boundary of the problem solution domain. A program has been developed from the presented computational algorithm. To accelerate the computation as much as possible, we use parallel programming processes and graphics processors. The program is written in the C++ programming language with the use of the OpenMP and OpenCL open standards. An example is considered to illustrate the work of the algorithm and the program; the calculation accuracy and the calculation speed are analyzed.

Keywords: *parallel computation, OpenMP, OpenCL, nonlinear heat conduction problem, boundary element method, analytical integration, radial basis functions.*

DOI: 10.17804/2410-9908.2016.6.080–091

References

1. Aguilar–Leal O., Fuentes–Aguilar R.Q., Chairez I., Garcia–Gonzalez A., Huegel J.C. Distributed parameter system identification using finite element differential neural networks. *Applied Soft Computing*, 2016, vol. 43, pp. 633–642. DOI: 10.1016/j.asoc.2016.01.004.
2. Petaccia G., Leporati F., Torti E. OpenMP and CUDA simulations of Sella Zerbino Dam break on unstructured grids. *Computational Geosciences*, 2016, vol. 20, no. 10, pp. 1123–1132. DOI: 10.1007/s10596-016-9580-5.
3. Sundarapandian M., Kalpathi R., Siochi R.A.C., Kadam A.S. Lung diaphragm tracking in CBCT images using spatio-temporal MRF. *Computerized Medical Imaging and Graphics*, 2016, vol. 53, pp. 9–18. DOI: 10.1016/j.compmedimag.2016.07.001.
4. Li K.L., Yang W.D., Li K.Q. A Hybrid Parallel Solving Algorithm on GPU for Quasi–Tridiagonal System of Linear Equations. *IEEE Transactions on Parallel and Distributed Systems*, 2016, vol. 27, no. 10, pp. 2795–2808. DOI: 10.1109/TPDS.2016.2516988.
5. Witz C., Treffer D., Hardiman T., Khinast J. Local gas holdup simulation and validation of industrial–scale aerated bioreactors. *Chemical Engineering Science*, 2016, vol. 152, pp. 636–648. DOI: 10.1016/j.ces.2016.06.053.
6. Tredak P., Rudnicki W.R., Majewski J.A. Efficient implementation of the many-body Reactive Bond Order (REBO) potential on GPU. *Journal of Computational Physics*, 2016, vol. 321, pp. 556–570. DOI: 10.1016/j.jcp.2016.05.061.
7. Jia S.Y., Zhang W.Z., Yu X.K., Pan Z.K. CPU-GPU mixed implementation of virtual node method for real-time interactive cutting of deformable objects using OpenCL. *International Journal of Computer Assisted Radiology and Surgery*, 2015, vol. 10, no. 9, pp. 1477–1491. DOI: 10.1007/s11548-014-1147-0.

8. *CUDA Zone. NVIDIA Developer*. Available at: https://developer.nvidia.com/cuda–zone (accessed 12.06.2016).
9. *OpenCL – The Open Standard for Parallel Programming of Heterogeneous Systems*. Available at: https://www.khronos.org/opencl (accessed 05.06.2015).
10. Munshi A., Gaster B.R., Mattson T.G., Fung J., Ginsburg D. *OpenCL Programming Guide*, Addison-Wesley, Upper Saddle River, NJ, Boston, Indianapolis, San Francisco, New York, Toronto, Montreal, London, Munich, Paris, Madrid, Cape Town, Sydney, Tokyo, Singapore, Mexico City, 2012, 603 p. ISBN 13: 978-0321749642, ISBN 10: 0321749642.
11. *OpenCL Optimization Guide*. Available at: http://developer.amd.com/tools–and–sdks/opencl–zone/amd–accelerated–parallel–processing–app–sdk/opencl–optimization–guide/ (accessed 23.01.2015).
12. Gaster B.R., Howes L., Kaeli D.R., Mistry P., Schaa D. *Heterogeneous Computing with OpenCL*, Morgan Kaufmann, Amsterdam, Boston, Heidelberg, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sydney, Tokyo, 2012, 277 p. ISBN 978-0-12-87766-6.
13. Han T.D., Abdelrahman T.S. Reducing Branch Divergence in GPU Programs. In: *GPGPU-4: Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*, Newport Beach, California, USA, March 05, 2011, article no. 3. DOI: 10.1145/1964179.1964184.
14. *What is OpenMP? PARALLEL.RU*. Available at: https://parallel.ru/tech/tech_dev/openmp.html (accessed 11.02.2015).
15. OpenMP. Available at: http://www.openmp.org/ (accessed 11.02.2015).
16. Chandra R., Dagum L., Kohr D., Maydan D., McDonald J., Melon R. *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers, San Francisco, 2001, 231 p. ISBN 1-55860-671-8.
17. *OpenACC. Directives for Accelerators*. Available at: http://www.openacc.org/ (accessed 21.04.2015).
18. Fedotov V.P., Spevak L.F. One approach to the derivation of exact integration formulae in the boundary element method. *Engineering Analysis with Boundary Elements*, 2008, vol. 32, no. 10, pp. 883–888. DOI: 10.1016/j.enganabound.2008.03.001.
19. Fedotov V.P., Spevak L.F., Nefedova O.A. Elastic-plastic deformation processes simulated by the modified boundary element method. *Programmnye producty i sistemy*, 2013, vol. 104, no. 4, pp. 253–257. DOI: 10.15827/0236-235X.104.253-257. (In Russian).
20. Fedotov V.P., Spevak L.F., Nefedova O.A. A software package designed to solve problems of the potential theory by the boundary element method. *Programmnye producty i sistemy*, 2014, vol. 108, no. 4, pp. 178–182. DOI: 10.15827/0236-235X.108.178-182. (In Russian).
21. Kazakov A.L., Spevak L.F. Numerical and analytical studies of a nonlinear parabolic equation with boundary conditions of a special form. *Applied Mathematical Modelling*, 2013, vol. 37, iss. 10–11, pp. 6918–6928. DOI: 10.1016/j.apm.2013.02.026.
22. Spevak L.F., Nefedova O.A. Solving the nonlinear heat conduction equation by the dual reciprocity boundary element method. *Mezhdunarodnyy zhurnal prikladnykh i fundamentalnykh issledovaniy*, 2015, no. 12–1. Available at: http://www.applied–research.ru/ru/article/view?id=7813 (accessed: 05.12.2016). (In Russian).
23. Kazakov A.L., Spevak L.F. An analytical and numerical study of a nonlinear parabolic equation with degeneration for the cases of circular and spherical symmetry. *Applied Mathematical Modelling*, 2015, vol. 40, iss. 2, pp. 1333–1343. DOI: 10.1016/j.apm.2015.06.038.
24. Vazquez J.L., ed. *The Porous Medium Equation: Mathematical Theory*. Clarendon Press, Oxford, 2006, 648 p. ISBN 978-0-19-856903-9. DOI: 10.1093/acprof:oso/9780198569039.001.0001.
25. Brebbia C.A., Telles J.F.C., Wrobel L.C. *Boundary Element Techniques*. Springer-Verlag, Berlin, Neidelberg, New-York, Tokyo, 1984, 466 p. ISBN: 978-3-642-48862-7 (Print), 978-3-642-48860-3 (Online). DOI: 10.1007/978-3-642-48860-3.

26. Nardini D., Brebbia C.A. A New Approach to Free Vibration Analysis using Boundary Elements. *Applied Mathematical Modelling*, 1983, vol. 7, no. 3, pp. 157–162. DOI: 10.1016/0307-904X(83)90003-3.
27. *GSL-GNU Scientific Library*. Available at: http://www.gnu.org/software/gsl/ (accessed 23.05.2016).
28. *Boost C++ Libraries*. Available at: http://www.boost.org.

Подана в журнал: 13.12.2016
УДК 517.958:[536.2+ 539.219.3]:004.272:004.032.24
DOI: 10.17804/2410-9908.2016.6.080–091

РАСПАРАЛЛЕЛИВАНИЕ РЕШЕНИЯ НЕЛИНЕЙНОЙ ЗАДАЧИ ТЕПЛОПРОВОДНОСТИ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENCL

Л. Ф. Спевак, О. А. Нефедова*

Федеральное государственное бюджетное учреждение науки Институт машиноведения Уральского отделения Российской академии наук, ул. Комсомольская, 34, Екатеринбург, Российская Федерация

*Ответственный автор. Электронная почта: nefedova@imach.uran.ru;
адрес для переписки: ул. Комсомольская, 34, Екатеринбург, Российская Федерация.
Тел.: +7 (343) 375–35–92; факс: +7 (343) 374–53–30

Статья посвящена применению методов параллельных вычислений к численному решению нелинейной краевой задачи для двумерного дифференциального уравнения теплопроводности с вырождением. Нелинейность задачи обусловлена степенной зависимостью коэффициента теплопроводности от температуры. Алгоритм решения основан на методе граничных элементов с использованием метода двойственной взаимности, позволяющего свести все вычисления на границу области решения задачи. На основе представленного вычислительного алгоритма разработана программа. Для максимального ускорения счета задействованы технологии параллельного программирования и графические процессоры. Программа написана на языке программирования C++ с использованием открытых стандартов OpenMP и OpenCL. Рассмотрен пример, иллюстрирующий работу алгоритма и программы, выполнен анализ точности и скорости счета.

Ключевые слова: параллельные вычисления, OpenMP, OpenCL, нелинейная задача теплопроводности, метод граничных элементов, аналитическое интегрирование, радиальные базисные функции.

1. Введение

В последнее время наблюдается рост использования графических процессоров (GPU) для решения прикладных задач в различных областях науки [1–7]. Рекламно-**маркетинговые усилия** компаний-производителей графических чипов и мировая конкуренция привели к существенному снижению стоимости самих видеокарт, а также к снижению совокупной стоимости владения данными аппаратными устройствами. Всё это позволило перевести GPU в разряд действительно массового продукта. В 2013–2015 гг. использование видеокарт получило широкую популярность в связи с генерацией цифровых денег – биткойнов. Код для ядра процессора представлял собой ряд простых логических операций с конечной схемой ветвлений, и привлечение GPU с возможностью выполнения параллельных вычислений для этих целей было идеальным вариантом. Однако для проведения точных математических расчетов требовалось использование типа данных double, возможности полной поддержки которого на GPU не было еще продолжительное время. Также не было единого стандарта для написания программ параллельных вычислений, который регламентировал бы операции загрузки микрокода и данных в процессор видеокарты.

На протяжении последних десяти лет многие разработчики программного обеспечения, ученые и исследователи активно используют набор инструментов CUDA SDK [8], который является частной собственностью фирмы производителя видеокарт NVIDIA и работает только на оборудовании этой компании. Заметим, что аппаратная поддержка типа данных double появилась в графических процессорах компании AMD заметно раньше, а их продукты

всегда были более привлекательными по цене для рядовых программистов. Первые версии открытой библиотеки OpenCL [9] были разработаны приблизительно в то же время, что и архитектура CUDA. Использование открытого стандарта OpenCL дает возможность запускать одну и ту же программу, выполняющую параллельные вычисления, как на центральных многоядерных процессорах, так и на графических процессорах [10]. Несмотря на то, что реализация приложений на OpenCL не зависит от типа оборудования и производителя, данный стандарт почему-то продолжительное время не привлекал особого внимания специалистов, и сейчас этот факт легко отследить по количеству статей, посвященных конкретной реализации распараллеливания [1–7].

В связи с тем, что проведение математических вычислений на GPU становится всё более популярным, нам хотелось бы отметить два значимых нюанса применения данной технологии. Первый состоит в том, что в отличие от центрального процессора (CPU), ядра в графическом процессоре не являются полностью потоконезависимыми, т.е. GPU не может выполнять циклические алгоритмы в параллельных участках кода независимо. Ветвления, как такового, здесь нет, и видеопроцессор всегда выполняет обе ветки условного оператора, по крайней мере, на группе ядер. Это свойство, а также отсутствие полноценного конвейера для каждого ядра при приеме и обработке информации приводят к тому, что выполнение циклов на GPU становится менее эффективным, чем на CPU [11–13]. А поскольку большинство численных методов решения прикладных задач широко используют циклы, от количества повторений которых зависит погрешность метода, неэффективным становится выполнение всего алгоритма на GPU. Выходом из создавшегося положения может стать перенос повторяющихся участков кода на CPU. Но тогда возникает второй нюанс, определяемый скоростью работы контроллера памяти и шины передачи данных вычислительного устройства. Память центрального процессора и память видеокарты независимы для большинства высокопроизводительных устройств, поэтому время, расходуемое на передачу массива аргументов на видеокарту, может многократно перекрывать недостатки конвейера GPU.

К открытым стандартам для распараллеливания программ относится также OpenMP [14, 15]. Фреймворк OpenMP описывает совокупность директив компилятора, библиотечных процедур и переменных окружения, предназначенных для программирования многопоточных приложений на многопроцессорных системах с общей памятью, и разработан для программ, написанных на языках программирования C, C++, Fortran [16]. В основе стандарта OpenMP лежат декларативные элементы и константы, подсказывающие компилятору возможности переноса независимых участков кода на разные ядра в многопроцессорной системе.

Для проведения параллельных вычислений этот метод является самым эффективным, но и самым затратным, поскольку создание большой многоядерной системы, проектирование и организация арбитража многих процессоров к общей памяти являются дорогостоящими задачами. Крупные производители суперкомпьютеров пытались спроектировать стандарт OpenACC [17], по синтаксису схожий с OpenMP, но предназначенный уже для систем, основанных на GPU, или даже модифицировать OpenMP для поддержки им вычислительных ускорителей. На сегодняшний день часть компиляторов языка C ограниченно поддерживает стандарт OpenACC, но ввиду большого разнообразия самих видеокарт, отличающихся архитектурой, даже в линейке одного производителя, использование директив для распараллеливания на GPU представляется малоэффективным.

Таким образом, использование GPU для выполнения математических расчетов при решении прикладных задач – это, несомненно, тенденция, которая будет становиться все более популярной, в частности по экономическим причинам. Однако графическим процессорам присущи определенные и неустранимые недостатки, указанные ранее. Следовательно, для эффективного и относительно бюджетного использования GPU немаловажным фактором является выбор класса и метода решения задачи.

Высокоинтенсивные процессы, сопровождающиеся значительным изменением температуры за малые промежутки времени, широко используются в топливно-энергетической,

химической, металлургической и других отраслях промышленности. При математическом моделировании процессов теплопроводности, протекающих в большом интервале изменения температур, необходимо учитывать зависимость коэффициента теплопроводности от температуры. В этом случае для определения распределения температурного поля в образце в любой момент времени необходимо решать нелинейную краевую задачу для дифференциального уравнения параболического типа. Точное решение нелинейных краевых задач связано с большими трудностями, поэтому специалисты часто обращаются к численным методам решения.

В данной работе представлен итерационный алгоритм решения двумерной нелинейной задачи в случае степенной зависимости коэффициента теплоповодности от температуры. В основе алгоритма лежит методика применения метода граничных элементов (МГЭ) [18], основанная на точном вычислении интегралов по граничным элементам с помощью полученных авторами аналитических формул и параллельных вычислений на каждом этапе решения. Разработанный подход и созданные на его основе программы показали свою эффективность для решения классических задач математической физики [19, 20], а также одномерных задач рассматриваемого в настоящей работе типа [21–23].

2. Математическая постановка задачи

Рассмотрим двумерное дифференциальное уравнение параболического типа в случае степенной зависимости коэффициента теплопроводности от температуры [24]

$$\frac{\partial\theta}{\partial\tau} = div\left[\lambda(\theta) grad\theta\right],\ \lambda(\theta) = \beta\theta^{\sigma}, \tag{1}$$

соответствующее нелинейной задаче теплопроводности. Здесь функция $\theta = \theta(\tau, x, y)$ – температура в точке (x, y) в момент времени $\tau > 0$; $\lambda(\theta)$ – коэффициент теплопроводности; div, $grad$ – дифференциальные операторы дивергенции и градиента соответственно; β и σ – положительные коэффициенты.

После введения в уравнение (1) новых переменных $u = \theta^{\sigma}$, $t = \beta\tau$, получим:

$$u_t = u(u_{xx} + u_{yy}) + \frac{1}{\sigma}(u_x^2 + u_y^2). \tag{2}$$

Здесь u_t, u_x, u_y – частные производные первого порядка от функции $u(t, x, y)$ по переменным t, x и y соответственно; u_{xx}, u_{yy} – частные производные второго порядка от функции $u(t, x, y)$ по соответствующим переменным.

Зададим краевое условие в следующем виде:

$$u\big|_{g(t,x,y)=0} = 0. \tag{3}$$

Здесь уравнение $g(t, x, y) = 0$ в каждый момент времени определяет нулевой фронт тепловой волны $S^{(t)}$ – замкнутую гладкую линию, ограничивающую область $V^{(t)}$, содержащую начало координат. Предполагается, что если $t_1 < t_2$, то $V^{(t_1)} \subset V^{(t_2)}$. Задача состоит в определении функции $u = u(t, x, y)$ в области $t \in [0, t_*]$, $(x, y) \in V^{(t)}$, где $V^{(t)}$ – область, ограниченная $S^{(0)}$ и $S^{(t)}$. Можно показать, что из условия (3) следует:

$$q\big|_{g(t,x,y)=0} = \frac{\sigma g_t(t,x,y)}{\sqrt{g_x^2(t,x,y)+g_y^2(t,x,y)}}. \quad (4)$$

Здесь $q = \frac{\partial u}{\partial n}$ – тепловой поток; $n = (n_x, n_y)$ – вектор внешней нормали к границе рассматриваемой области в момент времени t.

3. Алгоритмы решения методом граничных элементов

Решение задачи (2)–(4) выполнялось по шагам по времени на основе метода граничных элементов (МГЭ) [25]. На каждом шаге решается задача для уравнения

$$u_{xx} + u_{yy} = \frac{1}{u}\left(u_t - \frac{1}{\sigma}\left(u_x^2 + u_y^2\right)\right) \quad (5)$$

с граничными условмями (3), (4).

В соответствии с МГЭ, для произвольной точки $\xi \in V^{(t)}$ справедливо интегральное уравнение

$$u(\xi) = \int_S \left[q(x)u^*(\xi,x) - u(x)q^*(\xi,x)\right]dS(x) - \int_{V^{(t)}} \frac{1}{u}\left(u_t - \frac{u_x^2+u_y^2}{\sigma}\right)u^*(\xi,x)dV(x), \quad (6)$$

где $S = S^{(0)} \cup S^{(t)}$, $u^*(\xi,x)$, $q^*(\xi,x)$ – фунции влияния [25]. Для граничной точки $x_0 \in S$ справедливо

$$\frac{1}{2}u(x_0) = \int_S \left[q(x)u^*(x_0,x) - u(x)q^*(x_0,x)\right]dS(x) - \int_{V^{(t)}} \frac{1}{u}\left(u_t - \frac{u_x^2+u_y^2}{\sigma}\right)u^*(x_0,x)dV(x). \quad (7)$$

Записав уравнение (7) для каждого узла граничного элемента, получим систему линейных алгебраических уравнений, решение которой определит граничные значения температуры и потока, не заданные условиями (3) и (4). Решение проводилось с использованием аналитического вычисления интегралов по граничным элементам [18], использовались прямолинейные элементы с постоянной аппроксимацией.

Для вычисления интегралов по области $V^{(t)}$, стоящих в правых частях уравнений (6) и (7), применялся метод двойственной взаимности [26]. Представим входящий в подынтегральные выражения множитель $\frac{1}{u}\left(u_t - \frac{u_x^2}{\sigma}\right)$ в виде:

$$\frac{1}{u}\left(u_t - \frac{u_x^2}{\sigma}\right) = \sum_{i=1}^{N} \alpha_i f_i(x), \quad (8)$$

где для функций f_i существуют такие функции $\hat{u}_i$, что $f_i = \Delta\hat{u}_i$. В качестве функций f_i удобно использовать радиальные базисные функции (РБФ), значения которых зависят от расстояния между текущей точкой и заданными точками коллокации $x_1, x_2, ..., x_n$, лежащими в области $V^{(t)}$: $f_i(x) = f_i(|x - x_i|)$. С учетом разложения (8) интегралы могут быть вычислены следующим образом:

$$\int_{V^{(t)}} \frac{1}{u}\left(u_t - \frac{u_x^2+u_y^2}{\sigma}\right)u^*(\xi,x)dV(x) = \sum_{i=1}^{N}\alpha_i \int_{V^{(t)}} f_i(x)u^*(\xi,x)dV(x) = \sum_{i=1}^{N}\alpha_i \int_{V^{(t)}} \Delta\hat{u}_i(x)u^*(\xi,x)dV(x) =$$

$$= \sum_{i=1}^{N}\alpha_i\left(-\hat{u}_i(\xi) + \int_S \left[\hat{q}_i(x)u^*(\xi,x) - \hat{u}_i(x)q^*(\xi,x)\right]dS(x)\right), \qquad (9)$$

где $\hat{q}_i(x) = \frac{\partial\hat{u}_i(x)}{\partial n}$. Все вычисления, таким образом, сводятся на границу области решения задачи, и основное преимущество МГЭ – уменьшение размерности задачи – сохраняется. Полученные соотношения позволяют итерационно решить исходную задачу (2)–(4) на каждом шаге по времени.

4. Программная реализация

Представленный вычислительный алгоритм был реализован в виде программного модуля. Программа была написана на языке программирования C++ с использованием стандарта OpenMP для реализации параллельных вычислений с привлечением многопоточности на многопроцессорных многоядерных системах, имеющих общую память. Для выполнения параллельных вычислений на графических процессорах или на центральных процессорах, с использованием набора инструкций SSE/AVX, была задействована библиотека OpenCL.

Первоначально программа была отлажена на машине разработчика в окружении компилятора GCC 5.1.0 (TDM-GCC), последней версии драйверов OpenCL и технологии APP-AMD-SDK. Для выполнения численных расчетов были использованы библиотека GSL [27] и собрание библиотек классов BOOST [28]. Далее, после компиляции, отладки и проверки работоспособности, программный код копировался на суперкомпьютер «Уран» ИММ УрО РАН, оснащенный графическими укорителями NVIDIA Tesla M2050 GPU. Затем, уже в окружении вычислительной среды суперкомпьютера, была выполнена повторная компиляция программы. Компиляция производилась с использованием компилятора Intel C++ версии 14.0 и стандартных версий библиотек OpenMP и OpenCL, установленных на суперкомпьютере.

5. Пример

Тестирование программы, анализ точности и скорости счета были проведены на модельной задаче теплопроводности с вырождением для симметричного случая, когда нулевой фронт имеет форму окружности

$$g(t,x,y) = x^2 + y^2 - r^2(t),\ r(t) = R\left(1 + \frac{\mu t}{C}\right)^{\alpha}, \qquad (10)$$

где $\mu = 4 + \frac{4}{\sigma}$, $\alpha = \frac{1}{2\sigma+2}$. Точное решение в этом случае имеет вид:

$$u = -\frac{x^2+y^2}{C+\mu t} + \frac{C^{k-1}R^2}{(C+\mu t)^k}, \qquad (11)$$

где $k = \frac{4}{\mu}$.

Возможности программы позволили оперативно изменять входные данные, а именно: шаг по времени, количество граничных элементов, количество внутренних точек коллокации, а также вид РБФ, и производить компиляцию непосредственно перед выполнением расчета. Для этих целей использовался текстовой редактор Emacs. Значения температуры вдоль радиуса окружности, полученные с помощью МГЭ, оказались близки к аналитическому

решению (11). Для примера на рис. 1 приведено сравнение точного решения и решения МГЭ с шагом по времени $h = 0{,}05$ при $\sigma = 3;\ C = 10$ и $R = 1$ в различные моменты времени.

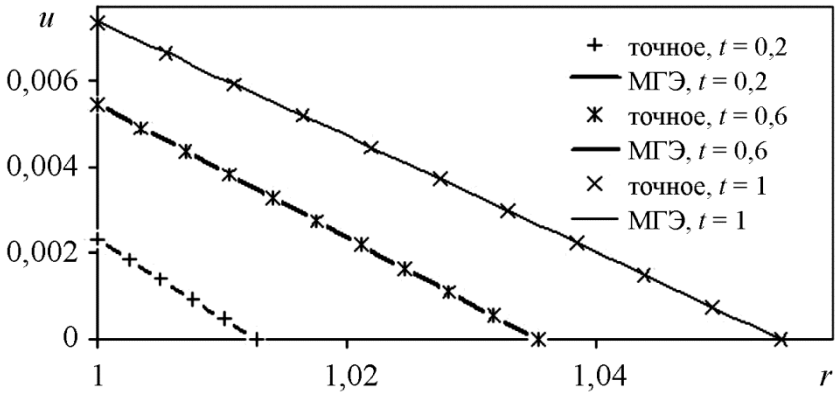

Рис. 1. Сравнение решения МГЭ и точного решения вдоль радиуса окружности

В табл. 1 представлено сравнение времени выполнения последовательного алгоритма и двух реализаций параллельного алгоритма с использованием OpenMP. Расчет выполнялся на двухъядерном процессоре Intel(R) Core(TM) Duo CPU E8300 и на четырехъядерном процессоре Intel(R) Core(TM)2 Quad CPU Q8400. Решение задачи проводилось для интервала времени $t \in [0,1]$ с шагом по времени $h = 0{,}1$.

Таблица 1 – Время расчета задачи при различных реализациях алгоритма с использованием OpenMP

Количество граничных элементов	Последовательный алгоритм, с	Параллельный алгоритм, использование OpenMP, 2 ядра, с	Параллельный алгоритм, использование OpenMP, 4 ядра, с
400	373	160	16
800	1065	445	99
1000	1531	645	190
1300	2396	1001	405

В табл. 2 представлено сравнение времени выполнения трех реализаций параллельного алгоритма с использованием OpenCL. Расчет выполнялся на двухъядерном процессоре Intel(R) Core(TM) Duo CPU E8300 с SSE, на графическом процессоре ATI Radeon HD 5750 GPU, имеющем 720 ядер, и на вычислительном модуле NVIDIA Tesla M2050 GPU. Решение задачи проводилось для интервала времени $t \in [0,1]$ с шагом по времени $h = 0{,}1$.

Таблица 2 – Время расчета задачи при различных реализациях алгоритма с использованием OpenCL

Количество граничных элементов	Параллельный алгоритм, реализация OpenCL на CPU с SSE, с	Параллельный алгоритм, реализация OpenCL на ATI Radeon HD 5750 GPU, с	Параллельный алгоритм, реализация OpenCL на NVIDIA Tesla M2050, с
400	3	3	2
800	10	6	6
1000	17	11	9
1300	38	27	19

Обобщая результаты, полученные в ходе выполнения программы на разных аппаратных платформах с использованием разных методов распараллеливания и при различном количестве граничных элементов, мы приходим к следующим выводам.

1. Как видно из таблицы, использование параллельных алгоритмов приводит к значительному сокращению времени счета.

2. Предложенный для распараллеливания алгоритм на основе МГЭ может полностью масштабироваться на количество ядер при различном количестве граничных элементов.

3. При решении нестационарной задачи с помощью МГЭ основное время затрачивается на подсчет коэффициентов системы линейных уравнений, вычисление которых может выполняться независимо, тогда, как решение самой системы не требует больших аппаратных ресурсов.

4. На обычных компьютерах платформа OpenCL более эффективна, чем OpenMP. Адаптация программы под нее может ускорить вычисления в несколько раз, даже без использования графических процессоров. Таким образом, имеет смысл включать ее во все новые проекты, где возможно распараллеливание алгоритма.

5. Подключение GPU позволяет еще больше ускорить счет, причем не столько важно поколение используемой видеокарты, как количество ядер на графическом процессоре.

6. На графическом процессоре имеет смысл выполнять только циклы с фиксированным количеством повторений, давая возможность компилятору раскрыть их в линейный код.

6. Заключение

В ходе работы была использована методика распараллеливания алгоритма на основе метода граничных элементов для открытых стандартов OpenMP и OpenCL и создана ее программная реализация. Представлено сравнение времени расчета на примере нелинейной краевой задачи для двумерного дифференциального уравнения теплопроводности с вырождением при различных реализациях алгоритма. На основе полученных результатов можно сделать вывод о целесообразности привлечения GPU для распараллеливания решения задач рассмотренного класса.

Благодарность

Работа выполнена при частичной поддержке Комплексной программы УрО РАН, проект № 15-7-1-17.

Список литературы

1. Distributed parameter system identification using finite element differential neural networks / O. Aguilar–Leal, R. Q. Fuentes–Aguilar, I. Chairez, A. Garcia–Gonzalez, J. C. Huegel // Applied Soft Computing. – 2016. – Vol. 43. – P. 633–642. – DOI: 10.1016/j.asoc.2016.01.004.
2. Petaccia G., Leporati F., Torti E. OpenMP and CUDA simulations of Sella Zerbino Dam break on unstructured grids // Computational Geosciences. – 2016. – Vol. 20, no. 10. – P. 1123–1132. – DOI: 10.1007/s10596-016-9580-5.
3. Lung diaphragm tracking in CBCT images using spatio–temporal MRF / M. Sundarapandian, R. Kalpathi, R. A. C. Siochi, A. S. Kadam // Computerized Medical Imaging and Graphics. – 2016. – Vol. 53. – P. 9–18. – DOI: 10.1016/j.compmedimag.2016.07.001.
4. Li K. L., Yang W. D., Li K. Q. A Hybrid Parallel Solving Algorithm on GPU for Quasi-Tridiagonal System of Linear Equations // IEEE Transactions on Parallel and Distributed Systems. – 2016. – Vol. 27, no. 10. – P. 2795–2808. – DOI: 10.1109/TPDS.2016.2516988.
5. Local gas holdup simulation and validation of industrial-scale aerated bioreactors / C. Witz, D. Treffer, T. Hardiman, J. Khinast // Chemical Engineering Science. – 2016. – Vol. 152. – P. 636–648. – DOI: 10.1016/j.ces.2016.06.053.

6. Tredak P., Rudnicki W. R., Majewski J. A. Efficient implementation of the many-body Reactive Bond Order (REBO) potential on GPU // Journal of Computational Physics. – 2016. – Vol. 321. – P. 556–570. – DOI: 10.1016/j.jcp.2016.05.061.
7. CPU–GPU mixed implementation of virtual node method for real–time interactive cutting of deformable objects using OpenCL / S. Y. Jia, W. Z. Zhang, X. K. Yu, Z. K. Pan // International Journal of Computer Assisted Radiology and Surgery. – 2015. – Vol. 10, no. 9. – P. 1477–1491. – DOI: 10.1007/s11548-014-1147-0.
8. CUDA Zone. NVIDIA Developer [Электронный ресурс]. – URL: https://developer.nvidia.com/cuda–zone (дата обращения: 12.09.2016).
9. OpenCL. The Open Standard for Parallel Programming of Heterogeneous Systems [Электронный ресурс]. – Khronos Group. – URL: https://www.khronos.org/opencl/ (дата обращения: 05.06.2015).
10. OpenCL Programming Guide / A. Munshi, B. R. Gaster, T. G. Mattson, J. Fung, D. Ginsburg. – Upper Saddle River, NJ, Boston, Indianapolis, San Francisco, New York, Toronto, Montreal, London, Munich, Paris, Madrid, Cape Town, Sydney, Tokyo, Singapore, Mexico City : Addison Wesley Professional, 2012. – 603 p. – ISBN 13 978-0321749642. – ISBN 10 0321749642.
11. OpenCL. Optimization Guide [Электронный ресурс]. – URL: http://developer.amd.com/tools–and–sdks/opencl–zone/amd–accelerated–parallel–processing–app–sdk/opencl–optimization–guide/ (дата обращения: 23.01.2015).
12. Heterogeneous Computing with OpenCL / B. R. Gaster, L. Howes, D. R. Kaeli, P. Mistry, D. Schaa. – Amsterdam, Boston, Heidelberg, London, New York, Oxford, Paris, San Diego, San Francisco, Singapore, Sydney, Tokyo : Morgan Kaufmann, 2012. – 277 p. – ISBN 978-0-12-387766-6.
13. Han T. D., Abdelrahman T. S. Reducing Branch Divergence in GPU Programs // The Fourth Workshop on General Purpose Processing on Graphics Processing Units “GPGPU-4”, Newport Beach, California, USA, March 05, 2011 : proceedings. – Article no. 3. – DOI: 10.1145/1964179.1964184.
14. Что такое OpenMP? PARALLEL.RU [Электронный ресурс]. – URL: https://parallel.ru/tech/tech_dev/openmp.html (дата обращения: 11.02.2015).
15. OpenMP [Электронный ресурс]. – URL: http://www.openmp.org/ (дата обращения: 11.02.2015).
16. Parallel Programming in OpenMP / R. Chandra, L. Dagum, D. Kohr, D. Maydan, J. McDonald, R. Melon. – San-Francisco : Morgan Kaufmann Publishers, 2001. – 231 p. – ISBN 1–55860–671–8.
17. OpenACC. Directives for Accelerators [Электронный ресурс]. – URL: http://www.openacc.org/ (дата обращения: 21.04.2015).
18. Fedotov V. P., Spevak L. F. One approach to the derivation of exact integration formulae in the boundary element method // Engineering Analysis with Boundary Elements. – 2008. – Vol. 32, no. 10. – P. 883–888. – DOI: 10.1016/j.enganabound.2008.03.001.
19. Федотов В. П., Спевак Л. Ф., Нефедова О. А. Моделирование процессов упругопластического деформирования модифицированным методом граничных элементов // Программные продукты и системы. – 2013. – Т. 104, № 4. – С. 253–257. – DOI: 10.15827/0236-235X.104.253-257.
20. Федотов В. П., Спевак Л. Ф., Нефедова О. А. Программный комплекс для решения задач теории потенциала методом граничных элементов // Программные продукты и системы. – 2014. – Т. 108, № 4. – С. 178–182. – DOI: 10.15827/0236-235X.108.178-182.
21. Kazakov A. L., Spevak L. F. Numerical and analytical studies of a nonlinear parabolic equation with boundary conditions of a special form // Applied Mathematical Modelling. – 2013. – Vol. 37, iss. 10–11. – P. 6918–6928. – DOI: 10.1016/j.apm.2013.02.026.
22. Спевак Л. Ф., Нефедова О. А. Решение нелинейного уравнения теплопроводности методом граничных элементов с использованием метода двойственной взаимности [Электрон-

ный ресурс] // Международный журнал прикладных и фундаментальных исследований. – 2015. – № 12–1. – С. 50–55. – URL: http://www.applied–research.ru/ru/article/view?id=7813 (дата обращения: 05.12.2016).
23. Kazakov A. L., Spevak L. F. An analytical and numerical study of a nonlinear parabolic equation with degeneration for the cases of circular and spherical symmetry // Applied Mathematical Modelling. – 2015. – Vol. 40, iss. 2. – P. 1333–1343. – DOI: 10.1016/j.apm.2015.06.038.
24. The Porous Medium Equation: Mathematical Theory / edited by J. L. Vazquez. – Oxford : Clarendon Press, 2006. – 648 p. – ISBN 978-0-19-856903-9. – DOI: 10.1093/acprof:oso/9780198569039.001.0001.
25. Brebbia C. A., Telles J. F. C., Wrobel L. C. Boundary Element Techniques. – Berlin, Neidelberg, New-York, Tokyo : Springer-Verlag, 1984. – 466 p. – ISBN 978-3-642-48862-7. – DOI: 10.1007/978-3-642-48860-3..
26. Nardini D., Brebbia C. A. A New Approach to Free Vibration Analysis using Boundary Elements // Applied Mathematical Modelling. – 1983. – Vol. 7, no. 3. – P. 157–162. – DOI: 10.1016/0307-904X(83)90003-3.
27. GSL-GNU Scientific Library [Электронный ресурс]. – URL: http://www.gnu.org/software/gsl/ (дата обращения: 23.05.2016).
28. Boost C++ Libraries [Электронный ресурс]. – URL: http://www.boost.org/ (дата обращения: 07.04.2016).