ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ
INFORMATION TECHNOLOGY, COMPUTER SCIENCE AND MANAGEMENT

УДК 004.272.22
https://doi.org/10.23947/2687-1653-2023-23-3-329-339

Научная статья

Разработка модели параллельно-конвейерного вычислительного процесса для решения системы сеточных уравнений

В.Н. Литвинов[1], **Н.Б. Руденко**[2], **Н.Н. Грачева**[2]
[1] Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация,
[2] Азово-Черноморский инженерный институт, ДГАУ, г. Зерноград, Российская Федерация
✉ LitvinovVN@rambler.ru

Аннотация

Введение. Экологические проблемы, возникающие на мелководных водоёмах и вызываемые как природными, так и техногенными факторами, ежегодно наносят существенный ущерб аквасистемам и прибрежным территориям. Своевременно определить эти проблемы, а также пути их устранения возможно с использованием современных вычислительных систем. Но проведённые ранее исследования показали, что ресурсов вычислительных систем, использующих только центральный процессор, недостаточно для решения больших научных задач, в частности, по прогнозированию крупных экологических происшествий, оценке нанесенного ими ущерба и определению возможностей их устранения. Для этих целей предлагается использовать модели вычислительной системы и декомпозиции расчётной области для разработки алгоритма параллельно-конвейерных вычислений. Целью данной работы является создание модели параллельно-конвейерного вычислительного процесса для решения системы сеточных уравнений модифицированным попеременно-треугольным итерационным методом с использованием декомпозиции трёхмерной равномерной расчётной сетки, учитывающей технические характеристики используемого для расчетов оборудования.

Материалы и методы. Разработаны математические модели вычислительной системы и расчётной сетки. Модель декомпозиции расчётной области выполнена с учётом характеристик гетерогенной системы. Предложен параллельно-конвейерный метод решения системы сеточных уравнений модифицированным попеременно-треугольным итерационным методом.

Результаты исследования. На языке CUDA C написана программа, реализующая параллельно-конвейерный метод решения системы сеточных уравнений модифицированным попеременно-треугольным итерационным методом. Проведённые эксперименты показали, что с увеличением числа потоков время вычислений уменьшается и при декомпозиции расчётной сетки рациональным является разбиение на фрагменты по координате z на величину, не превышающую 10. Результаты экспериментов подтвердили эффективность разработанного параллельно-конвейерного метода.

Обсуждение и заключение. По итогам проведенных исследований разработана модель параллельно-конвейерного вычислительного процесса на примере одного из самых трудоёмких этапов решения системы сеточных уравнений модифицированным попеременно-треугольным итерационным методом. Её построение основано на моделях декомпозиции трёхмерной равномерной расчётной сетки, учитывающей технические характеристики используемого в расчетах оборудования. Применение программы позволит ускорить процесс расчёта и равномерно по времени загрузить программные потоки. Проведенные численные эксперименты подтвердили математическую модель декомпозиции расчётной области.

Ключевые слова: параллельный алгоритм, вычислительный процесс, сеточные уравнения

Благодарности: авторы выражают благодарность редакционной коллегии журнала и рецензенту за профессиональный анализ и рекомендации для корректировки статьи.

Финансирование. Работа выполнена при поддержке Российского научного фонда (проект № 21–71–20050).

Для цитирования. Литвинов В.Н., Руденко Н.Б., Грачева Н.Н. Разработка модели параллельно-конвейерного вычислительного процесса для решения системы сеточных уравнений. *Advanced Engineering Research (Rostov-on-Don).* 2023;23(3):329–339. https://doi.org/10.23947/2687-1653-2023-23-3-329-339

Original article

Model of a Parallel-Pipeline Computational Process for Solving a System of Grid Equations

Vladimir N. Litvinov[1], Nelli B. Rudenko[2], Natalya N. Gracheva[2]

[1] Don State Technical University, Rostov-on-Don, Russian Federation

[2] Azov-Black Sea Engineering Institute, Don State Agrarian University, Zernograd, Russian Federation

✉ LitvinovVN@rambler.ru

Abstract

Introduction. Environmental problems arising in shallow waters and caused by both natural and man-made factors annually do significant damage to aquatic systems and coastal territories. It is possible to identify these problems in a timely manner, as well as ways to eliminate them, using modern computing systems. But earlier studies have shown that the resources of computing systems using only a central processor are not enough to solve large scientific problems, in particular, to predict major environmental accidents, assess the damage caused by them, and determine the possibilities of their elimination. For these purposes, it is proposed to use models of the computing system and decomposition of the computational domain to develop an algorithm for parallel-pipeline calculations. The research objective was to create a model of a parallel-conveyor computational process for solving a system of grid equations by a modified alternating-triangular iterative method using the decomposition of a three-dimensional uniform computational grid that takes into account technical characteristics of the equipment used for calculations.

Materials and Methods. Mathematical models of the computer system and computational grid were developed. The decomposition model of the computational domain was made taking into account the characteristics of a heterogeneous system. A parallel-pipeline method for solving a system of grid equations by a modified alternating-triangular iterative method was proposed.

Results. A program was written in the CUDA C language that implemented a parallel-pipeline method for solving a system of grid equations by a modified alternating-triangular iterative method. The experiments performed showed that with an increase in the number of threads, the computation time decreased, and when decomposing the computational grid, it was rational to split into fragments along coordinate z by a value not exceeding 10. The results of the experiments proved the efficiency of the developed parallel-pipeline method.

Discussion and Conclusion. As a result of the research, a model of a parallel-pipeline computing process was developed using the example of one of the most time-consuming stages of solving a system of grid equations by a modified alternating-triangular iterative method. Its construction was based on decomposition models of a three-dimensional uniform computational grid, which took into account the technical characteristics of the equipment used in the calculations. This program can provide you for the acceleration of the calculation process and even loading of program flows in time. The conducted numerical experiments validated the mathematical model of decomposition of the computational domain.

Keywords: parallel algorithm, computational process, grid equations

Acknowledgements: the authors would like to thank the editorial board of the journal and the reviewer for their professional analysis and recommendations for correcting the article.

Funding information. The research was done with the support of the Russian Science Foundation (project No. 21–71–20050).

For citation. Litvinov VN, Rudenko NB, Gracheva NN. Model of a Parallel-Pipeline Computational Process for Solving a System of Grid Equations. *Advanced Engineering Research (Rostov-on-Don).* 2023;23(3):329–339. https://doi.org/10.23947/2687-1653-2023-23-3-329-339.

Введение. В последнее время на территории Ростовской области отмечается возникновение ряда серьезных экологических проблем. К ним, в частности, относится эвтрофикация вод Азовского моря и Цимлянского водохранилища, которая вызывает рост вредоносных и токсичных видов популяций фитопланктона [1]. Инженерные работы в акваториях рек и морей приводят к загрязнению прилегающих территорий, изменению популяционной структуры биоты и ухудшению условий воспроизводства ценных и промысловых рыб. Изменение климата на юге России привело к увеличению количества случаев затопления некоторых территорий в районе Таганрогского залива и поймы реки Дон, вызванных сгонно-нагонными явлениями. В последнее

десятилетие в летний период несколько раз наблюдалось практически полное осушение русла реки Дон, что приводило к полной остановке судоходства. Чтобы спрогнозировать возникновение и развитие подобных случаев, спланировать пути устранения их последствий, оценить нанесенный ими ущерб, требуются современные программные комплексы, построенные с использованием высокоточных математических моделей, численных методов, алгоритмов и структур данных [2].

В основе математических моделей, используемых при прогнозировании природных и техногенных катастроф, лежат системы дифференциальных уравнений в частных производных, например уравнения Пуассона, Навье-Стокса, диффузии-конвекции-реакции, теплопроводности. Численное решение таких систем приводит к необходимости оперативного хранения больших объёмов данных (в массивах различной структуры) и решения систем сеточных уравнений высокой размерности, превышающих 10^9. Объём оперативной памяти, требуемой для хранения массивов данных при численном решении только одного уравнения Пуассона для трёхмерной области размерностью $10^3 \times 10^3 \times 10^3$ попеременно-треугольным итерационным методом, составляет более 64 Гб. В случае численного решения комбинированных задач требуются сотни гигабайт оперативной памяти, которые могут быть доступны лишь при использовании суперкомпьютерных систем.

Проведенное ранее исследование показало, что ресурсов вычислительной системы, использующей только CPU, недостаточно для решения подобных научных задач [3]. Увеличение мощности и видеопамяти GPU позволило использовать для расчетов ресурсы видеоадаптеров [4]. Эффективность использования GPU зависит от применения параллельных алгоритмов для решения вычислительно-трудоемких задач водной экологии [5–7]. Частично решить проблемы нехватки памяти и вычислительной мощности на рабочих станциях можно установкой дополнительных видеоадаптеров в слоты PCI-E X16 непосредственно и в слоты PCI-E X1 с помощью переходников PCI-E X1–PCI-E X16. Таким образом, количество видеоадаптеров, установленных на одной рабочей станции, можно довести до 12 [8–11].

Всё большую популярность в научном сообществе приобретают гетерогенные вычислительные системы, которые позволяют использовать ресурсы CPU и GPU совместно. Применение таких систем дает возможность уменьшить время расчета научных задач [12–14]. Однако эффективное использование гетерогенной вычислительной среды предполагает модернизацию математических моделей, алгоритмов и программ, их численно реализующих. Гетерогенная система позволяет организовать процесс вычислений в параллельном режиме. При этом должны быть учтены принципиальные различия в построении программных систем, использующих совместно CPU и GPU.

Материалы и методы. Опишем предложенные математические модели вычислительной системы, расчетной сетки, а также метод декомпозиции расчетной области.

Пусть D — множество технических характеристик вычислительной системы, тогда:

$$D = D^1 \bigcup D^2 \bigcup D^3, \tag{1}$$

где D^1 — подмножество характеристик центральных процессоров (CPU) вычислительной системы; D^2 — подмножество характеристик видеоадаптеров (GPU) вычислительной системы; D^3 — подмножество характеристик оперативной памяти.

$$D^1 = \left\langle d^{1,1}, d^{1,2}, d^{1,3}, d^{1,4} \right\rangle, \tag{2}$$

где $d^{1,1}$ — суммарное число ядер CPU; $d^{1,2}$ — количество потоков, одновременно обрабатываемых одним ядром процессора CPU; $d^{1,3}$ — тактовая частота, МГц; $d^{1,4}$ — частота шины центрального процессора, МГц.

$$D^2 = \bigcup_{k_{GPU} \in K_{GPU}} D^2_{k_{GPU}} = \left\{ d^2 \mid \exists k_{GPU} \in K_{GPU}, d^2 \in D^2_{k_{GPU}} \right\}, \tag{3}$$

где $K_{GPU} = \{1, ..., N_{GPU}\}$ — множество индексов видеоадаптеров; N_{GPU} — количество видеоадаптеров вычислительной системы; k_{GPU} — индекс видеоадаптера.

Каждый видеоадаптер представим в виде кортежа:

$$D^2_{k_{GPU}} = \left\langle d^{2,1}_{k_{GPU}}, d^{2,2}_{k_{GPU}} \right\rangle, \tag{4}$$

где $d^{2,1}_{k_{GPU}}$ — объём видеопамяти видеоадаптера с индексом k_{GPU}, Гб; $d^{2,2}_{k_{GPU}}$ — количество потоковых мультипроцессоров.

$$D^3 = \left\langle d^{3,1}, d^{3,2} \right\rangle, \tag{5}$$

где $d^{3,1}$ — суммарный объём оперативной памяти, Гб; $d^{3,2}$ — тактовая частота оперативной памяти, МГц.

Пусть S — множество программных потоков, задействованных в вычислительном процессе, тогда:

$$\begin{aligned} &S = S^1 \bigcup S^2, \\ &S^1 = \{1,...,N_{S^1}\}, \\ &S^2 = \bigcup_{k_{GPU} \in K_{GPU}} S^2_{k_{GPU}}, \quad S^2_{k_{GPU}} = \left\{1,...,N_{S^2_{k_{GPU}}}\right\}, \end{aligned} \tag{6}$$

где S^1 — подмножество программных потоков, реализующих процесс расчета на CPU; S^2 — подмножество потоковых блоков CUDA, реализующих процесс расчета на потоковых мультипроцессорах GPU; N_{S^1} — количество задействованных программных потоков CPU; $S^2_{k_{GPU}}$ — подмножество потоковых блоков CUDA, реализующих процесс расчета на потоковых мультипроцессорах GPU с индексом k_{GPU}; $K_{GPU} = \{1,...,N_{GPU}\}$ — множество индексов GPU; N_{GPU} — количество GPU в вычислительной системе; $N_{S^2_{k_{GPU}}}$ — количество задействованных потоковых блоков CUDA, реализующих процесс расчета на потоковых мультипроцессорах GPU с индексом k_{GPU}.

Пусть E — множество идентификаторов программных потоков. Тогда для идентификации программных потоков в вычислительной системе каждому элементу множества программных потоков S поставим в соответствие кортеж e из двух элементов:

$$\forall s \in S \;\; \exists e \in E: \; e = \langle n_d, n_t \rangle, \tag{7}$$

где n_d — индекс вычислительного устройства в вычислительной системе; n_t — индекс программного потока CPU или потокового блока GPU.

$$n_d = \begin{cases} 0, & s \in S^1 \\ K_{GPU}, & s \in S^2 \end{cases}, \tag{8}$$

$$n_t = \begin{cases} K_{S^1}, & s \in S^1 \\ K_{S^2_{k_{GPU}}}, & s \in S^2_{k_{GPU}} \end{cases}. \tag{9}$$

Возьмём расчётную область со следующими параметрами: l_x — характерный размер по оси Ox; l_y — по оси Oy; l_z — по оси Oz. Сопоставим с указанной областью равномерную расчётную сетку следующего вида:

$$\begin{aligned} &W = \left\{ x_i = ih_x, y_j = jh_y, z_k = kh_z; \right. \\ &\overline{i = 0, n_x - 1}, \overline{j = 0, n_y - 1}, \overline{k = 0, n_z - 1}; \\ &\left. (n_x - 1)h_x = l_x, (n_y - 1)h_y = l_y, (n_z - 1)h_z = l_z \right\}, \end{aligned} \tag{10}$$

где h_x, h_y, h_z — шаги расчётной сетки по соответствующим пространственным направлениям; n_x, n_y, n_z — количество узлов расчётной сетки по соответствующим пространственным направлениям.

Тогда множество узлов расчётной сетки представим в виде:

$$\begin{aligned} &G = \left\{ g_{i,j,k}, \overline{i = 0, n_x - 1}, \overline{j = 0, n_y - 1}, \overline{k = 0, n_z - 1} \right\}, \\ &g_{i,j,k} = \langle x_i, y_j, z_k \rangle, \end{aligned} \tag{11}$$

где $g_{i,j,k}$ — узел расчётной сетки.

Число узлов расчётной сетки N_G вычисляется по формуле:

$$N_G = n_x \cdot n_y \cdot n_z. \tag{12}$$

Под подразделом расчётной сетки $G^{k_1} \subset G$ (далее — подраздел) будем понимать подмножество узлов расчётной сетки G.

$$G = \bigcup_{k_1 \in K_{k_1}} G^{k_1} = \left\{ g^{k_1} \mid \exists k_1 \in K_{k_1}, g^{k_1} \in G_{k_1} \right\}, \; \bigcap_{k_1 \in K_{k_1}} G^{k_1} = \varnothing, \tag{13}$$

где $K_{k_1} = \{1,...,N_{k_1}\}$ — множество индексов подразделов G^{k_1} расчётной сетки G; N_{k_1} — количество подразделов G^{k_1}; $K_{k_1}, N_{k_1} \subset N$; N — множество натуральных чисел; k_1 — индекс подраздела G^{k_1}.

Так как $G^{k_1} \subset G$, тогда:

$$G^{k_1} = \left\{ g^{k_1}_{i,\tilde{j},k}, \overline{i = 0, n_x - 1}, \overline{\tilde{j} = 0, n^{k_1}_y - 1}, \overline{k = 0, n_z - 1} \right\}, \tag{14}$$

где $g^{k_1}_{i,\tilde{j},k}$ — узел подраздела k_1; знак $\sim$ обозначает принадлежность к подразделу; $\tilde{j}$ — индекс узла подраздела k_1 по координате y; $n^{k_1}_y$ — количество узлов в подразделе k_1 по координате y.

$$\begin{aligned} &g^{k_1}_{i,\tilde{j},k} = \langle x_i, y_j, z_k \rangle, \\ &x_i = ih_x, y_j = \left(\sum_{b_1=1}^{k_1 - 1} n^{b_1}_y + \tilde{j} \right) \cdot h_y, z_k = kh_z, \end{aligned} \tag{15}$$

где $n_y^{b_1}$ — количество узлов по координате y b_1-го раздела.

Под блоком расчётной сетки G^{k_1,k_2} (далее — блок) будем понимать подмножество узлов расчётной сетки подраздела G^{k_1}.

$$G^{k_1} = \bigcup_{k_2 \in K_{k_1,k_2}} G^{k_1,k_2} = \left\{ g^{k_1,k_2} \mid \exists k_2 \in K_{k_1,k_2},\, g^{k_1,k_2} \in G^{k_1,k_2} \right\},\ \bigcap_{k_2 \in K_{k_1,k_2}} G^{k_1,k_2} = \varnothing, \tag{16}$$

где $K_{k_1,k_2} = \left\{1,..., N_{k_1,k_2}\right\}$ — множество индексов блоков G^{k_1,k_2} подраздела G^{k_1}; N_{k_1,k_2} — количество блоков G^{k_1,k_2}; $K_{k_1,k_2}, N_{k_1,k_2} \subset N$; k_2 — индекс блока G^{k_1,k_2} подраздела G^{k_1}.

Так как $G^{k_1,k_2} \subset G^{k_1}$, тогда:

$$G^{k_1,k_2} = \left\{ g_{i,\hat{j},k}^{k_1,k_2}, i = \overline{0, n_x - 1},\ \hat{j} = \overline{0,\, n_y^{k_1,k_2} - 1}, k = \overline{0, n_z - 1} \right\}, \tag{17}$$

где $g_{i,\hat{j},k}^{k_1,k_2}$ — узел блока k_1,k_2; знак ^ обозначает принадлежность к блоку; $\hat{j}$ — индекс узла блока k_1,k_2 по координате y; $n_y^{k_1,k_2}$ — количество узлов в блоке k_1,k_2 по координате y.

$$\begin{aligned} & g_{i,\hat{j},k}^{k_1,k_2} = \left\langle x_i, y_j, z_k \right\rangle, \\ & x_i = ih_x,\ \ y_j = \left(\sum_{b_1=1}^{k_1-1} \sum_{b_2=1}^{N_{k_1,k_2}} n_y^{b_1,b_2} + \hat{j} \right) \cdot h_y,\ \ z_k = kh_z, \end{aligned} \tag{18}$$

где $n_y^{b_1,b_2}$ — количество узлов блока b_1,b_2.

Под фрагментом расчётной сетки G^{k_1,k_2,k_3} (далее — фрагмент) будем понимать подмножество узлов расчётной сетки блока G^{k_1,k_2} подраздела G^{k_1}.

$$\begin{aligned} & G^{k_1,k_2} = \bigcup_{k_3 \in K_{k_3}} G^{k_1,k_2,k_3} = \left\{ g^{k_1,k_2,k_3} \mid \exists k_3 \in K_{k_3},\, g^{k_1,k_2,k_3} \in G^{k_1,k_2,k_3} \right\}, \\ & \bigcap_{k_3 \in K_{k_3}} G^{k_1,k_2,k_3} = \varnothing, \end{aligned} \tag{19}$$

где $K_{k_1,k_2,k_3} = \left\{1,..., N_{k_1,k_2,k_3}\right\}$ — множество индексов фрагментов G^{k_1,k_2,k_3} блока G^{k_1,k_2} подраздела G^{k_1}; N_{k_1,k_2,k_3} — количество фрагментов G^{k_1,k_2,k_3}; $K_{k_1,k_2,k_3}, N_{k_1,k_2,k_3} \subset N$; k_3 — индекс фрагмента G^{k_1,k_2,k_3} блока G^{k_1,k_2} подраздела G^{k_1}.

Каждому индексу k_3 фрагмента G^{k_1,k_2,k_3} поставим в соответствие кортеж индексов $\left\langle k_4, k_5 \right\rangle$, предназначенный для хранения координат фрагмента в плоскости xOz, где k_4 — индекс фрагмента по координате x; k_5 — индекс фрагмента по координате z.

$$k_3 = k_4 + K_{k_4} \cdot k_5, \tag{20}$$

где k_4 — индекс фрагмента по координате x; k_5 – индекс фрагмента по координате z.

Количество фрагментов G^{k_1,k_2,k_3} блока G^{k_1,k_2} вычислим по формуле:

$$K_{k_3} = K_{k_4} \cdot K_{k_5}, \tag{21}$$

где K_{k_4} — количество фрагментов по оси Ox; K_{k_5} — количество фрагментов по координате z.

Так как $G^{k_1,k_2,k_3} \subset G^{k_1,k_2}$, тогда:

$$G^{k_1,k_2,k_3} = \left\{ \breve{g}_{\breve{i},\hat{j},\breve{k}},\ \ \breve{i} = \overline{0, \breve{n}_x - 1},\ \ \hat{j} = \overline{0, \hat{n}_y - 1},\ \ \breve{k} = \overline{0, \breve{n}_z - 1} \right\}, \tag{22}$$

где $\breve{g}_{\breve{i},\hat{j},\breve{k}}$ — узел фрагмента; знак ◡ обозначает принадлежность к фрагменту; $\breve{i}, \breve{k}$ — индексы узла фрагмента по координатам x, z; $\breve{n}_x$, $\breve{n}_z$ — количество узлов расчётной сетки в фрагменте по координатам x, z; $\breve{l}_x$, $\breve{l}_z$ — размеры фрагмента по координатам x, z.

$$\begin{aligned} & \breve{g}_{\breve{i},\hat{j},\breve{k}} = \left\langle x_i, y_j, z_k \right\rangle, \\ & x_i = \left(\sum_{b=1}^{k_4-1} \breve{n}_b + \breve{i} \right) h_x,\ y_j = \hat{j} h_y,\ z_k = \left(\sum_{b=1}^{k_5-1} \breve{n}_b + \breve{k} \right) h_z, \end{aligned} \tag{23}$$

где $\breve{n}_b$ — количество узлов b-го фрагмента.

Введем множество сопоставлений блоков расчётной сетки программным потокам M^1

$$M^1 = \bigcup_{k_1 \in K_{k_1}} \left(\bigcup_{k_2 \in K_{k_2}} M_{k_1,k_2}^1 \right), \tag{24}$$

где M_{k_1,k_2}^1 — элемент множества M^1.

Пусть $M^1_{k_1,k_2}$ — сопоставление блока G^{k_1,k_2} программному потоку s_{k_1,k_2}, тогда:

$$M^1_{k_1,k_2} = \left\langle G^{k_1,k_2}, s_{k_1,k_2} \right\rangle, \tag{25}$$

где $s_{k_1,k_2} \in S$ — программный поток, вычисляющий блок G^{k_1,k_2}.

В процессе решения задач гидродинамики на трехмерных расчетных сетках большой размерности необходимы высокопроизводительные вычислительные системы и огромные объемы памяти для хранения данных. Ресурсов одного вычислительного устройства недостаточно для вычислений и хранения трехмерной расчетной сетки со всеми ее данными. Для решения этой проблемы предложены различные способы декомпозиции расчетных сеток с последующим применением параллельных алгоритмов расчета в гетерогенных вычислительных средах [15].

Для декомпозиции расчётной сетки необходимо учитывать производительность вычислительных устройств, участвующих в расчётах. Под производительностью будем понимать количество узлов расчётной сетки, рассчитываемых с помощью заданного алгоритма в единицу времени.

Предположим, что все вычислительные устройства используются для расчётов. Тогда суммарная производительность вычислительной системы P_Σ вычисляется по формуле:

$$P_\Sigma = P_{CPU} \cdot N_{S^1} + \sum_{b=1}^{N_{GPU}} P^b_{GPU} \cdot N^b_{S^2}, \tag{26}$$

где P_{CPU} — производительность одного потока CPU; N_{S^1} — число программных потоков, реализующих процесс расчета на CPU; P^b_{GPU} — производительность GPU с индексом b на одном потоковом мультипроцессоре; $N^b_{S^2}$ — количество потоковых блоков CUDA, реализующих процесс расчета на потоковых мультипроцессорах GPU.

Тогда рассчитать количество узлов расчётной сетки n^b_y в подразделе по координате y для каждого GPU с индексом b можно по формуле:

$$n^b_y = \left\lfloor \frac{P^b_{GPU}}{P_\Sigma} \right\rfloor \cdot n_y. \tag{27}$$

В процессе вычисления по формуле (27) получим остаток — некоторое количество узлов расчётной сетки. Эти узлы будут располагаться в оперативной памяти. Количество оставшихся узлов n^b_y по координате y рассчитывается по формуле:

$$n^b_{yL} = n_y - \sum_{b=1}^{N_{GPU}} n^b_y. \tag{28}$$

Для вычисления количества узлов по координате y в блоках расчётной сетки, обрабатываемых потоковыми мультипроцессорами GPU, воспользуемся формулами:

$$\begin{gathered} n^b_{yGT} = \left\lfloor \frac{n^b_y}{N^b_{S^2} - 1} \right\rfloor, \quad b = \overline{1, N^b_{S^2} - 1}, \\ n^b_{yGTL} = n^b_y - \sum_{b=1}^{N^b_{S^2}-1} n^b_{yGT}, \end{gathered} \tag{29}$$

где n^b_{yGT} — количество узлов по координате y в блоках расчётной сетки, обрабатываемых потоковыми мультипроцессорами GPU с индексом b, кроме последнего блока; n^b_{yGTL} — количество узлов по координате y в последнем блоке расчётной сетки, обрабатываемом потоковыми мультипроцессорами GPU с индексом b.

Для вычисления количества узлов расчётной сетки по координате y в блоках, обрабатываемых программными потоками, реализующими процесс расчета на CPU, воспользуемся формулами:

$$\begin{gathered} n_{yCT} = \left\lfloor \frac{n^{CPU}_y}{N_{S^1} - 1} \right\rfloor, \\ n_{yCTL} = n^{CPU}_y - n_{yCT} \cdot \left(N_{S^1} - 1 \right), \end{gathered} \tag{30}$$

где n_{yCT} — количество узлов расчётной сетки по координате y, обрабатываемых программными потоками CPU, кроме последнего потока; n_{yCTL} — количество узлов расчётной сетки по координате y, обрабатываемых программными потоками CPU, в последнем потоке.

Рассчитаем количество фрагментов расчётной сетки по координате y:

$$N^f_y = N_{S^1} + \sum_{b=1}^{N_{GPU}} N^b_{S^2}. \tag{31}$$

Пусть задано количество фрагментов N^f_x и N^f_z по координатам x и z соответственно. Тогда количество узлов расчётной сетки по координате x вычисляется по формулам:

$$n_x^f = \left\lfloor \frac{n_x}{N_x^f - 1} \right\rfloor, \quad n_x^{fL} = n_x - n_x^f \cdot \left(N_x^f - 1\right), \tag{32}$$

где n_x^f — количество узлов расчётной сетки по координате x во всех фрагментах, кроме последнего; n_x^{fL} — количество узлов расчётной сетки по координате x в последнем фрагменте.

Аналогично вычисляется количество узлов расчётной сетки по координате z:

$$n_z^f = \left\lfloor \frac{n_z}{N_z^f - 1} \right\rfloor, \quad n_z^{fL} = n_z - n_z^f \cdot \left(N_z^f - 1\right), \tag{33}$$

где n_z^f — количество узлов расчётной сетки по координате z во всех фрагментах, кроме последнего; n_z^{fL} — количество узлов расчётной сетки по координате z в последнем фрагменте.

Опишем модель параллельно-конвейерного метода. Пусть на M^1 необходимо организовать параллельный процесс вычислений некоторой функции F, причем вычисления в каждом фрагменте G^{k_1,k_2,k_3} зависят от значений в соседних фрагментах, каждый из которых имеет хотя бы один из индексов по координатам x, y и z на единицу меньший, чем у текущего (рис. 1).

Для организации параллельно-конвейерного метода введём множество кортежей A, задающих соответствия a между идентификаторами программных потоков e, обрабатывающих фрагменты G^{k_1,k_2,k_3}, номерам шагов параллельно-конвейерного метода r:

$$\forall e \in E \ \exists a \in A: \ a = \left\langle e, G^{k_1,k_2,k_3}, r \right\rangle, \tag{34}$$

где $r = \overline{1, N_r}$ — номер шага параллельно-конвейерного метода; N_r — число шагов параллельно-конвейерного метода, вычисляемое по формуле:

$$N_r = N_x^f N_z^f + N_y^f - 1. \tag{35}$$

Полная загрузка всех вычислителей в предлагаемом параллельно-конвейерном методе начинается с шага $r_{100START} = N_y^f$ и заканчивается на шаге $r_{100STOP} = N_x^f N_z^f$. При этом общее количество шагов с полной загрузкой вычислителей N_{rPAR} составит:

$$N_{rPAR} = r_{100STOP} - r_{100START} + 1 = N_x^f N_z^f - N_y^f + 1. \tag{36}$$

Время вычислений некоторой функции F параллельно-конвейерным методом запишем в виде:

$$T_{M^1} = \sum_{r=1}^{N_r} \max(\mathrm{T}_a), \tag{37}$$

где T_a — вектор значений затрат времени на обработку фрагментов в параллельном режиме.

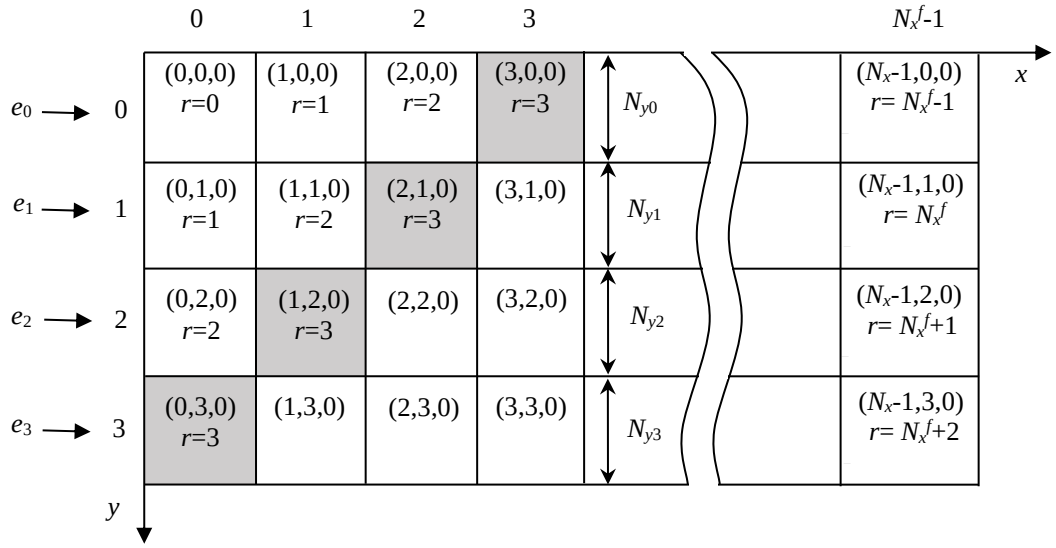

Рис. 1. Параллельно-конвейерный вычислительный процесс

Результаты исследования. Вычислительные эксперименты проводились на высокопроизводительной вычислительной системе К-60 Института прикладной математики им. М. В. Келдыша Российской академии наук. Использовалась секция с GPU, каждый узел которой оснащён двумя процессорами Intel Xeon Gold 6142 v4, четырьмя видеоадаптерами Nvidia Volta GV100GL и 768 Гб оперативной памяти.

Вычислительный эксперимент состоял из двух этапов — подготовительного и основного. На подготовительном этапе проверялась корректность декомпозиции расчетной области на подразделы, блоки и фрагменты путем поэлементного сравнения значений в узлах исходной сетки и в фрагментах, полученных в результате декомпозиции. Затем проверялась работа алгоритма управления потоками, в процессе которого фиксировалось время, затрачиваемое на расчет 1, 8, 16 и 32 фрагментов расчетной сетки размерностью 50 узлов по пространственным координатам x, y и z тем же количеством потоков CPU N_{S^1} итерационным попеременно-треугольным методом в параллельном режиме. Выполнялось 10 повторов с вычислением среднего арифметического T_a и стандартного отклонения σ. По полученным данным вычислялось время $T_a^1 = T_a / N_{S^1}$, затрачиваемое каждым потоком на обработку одного фрагмента расчетной сетки и ускорение $E = T_a^1(N_{S^1}) / T_a^1(1)$, равное отношению времени $T_a^1(N_{S^1})$ обработки одного фрагмента N_{S^1} потоками к соответствующему времени обработки одним потоком $T_a^1(1)$. Экспериментальные данные приведены в таблице 1. Проведённый эксперимент показал, что стандартное отклонение имеет наименьшее значение в случае использования 32 параллельных потоков CPU и составляет 0,026 мс, то есть использование 32 параллельных потоков CPU при расчёте 32 фрагментов расчётной сетки даёт более равномерную по времени загруженность программных потоков, что в целом повышает эффективность работы вычислительного узла. При этом среднее значение расчета одного фрагмента составило 4,14 мс. Зависимость ускорения E от числа потоков получилась линейная $E = 0,603 + 0,804 N_{S^1}$, с коэффициентом детерминации, равным 0,99. Получили, что с увеличением числа потоков ускорение разработанного алгоритма возрастает. Это свидетельствует об эффективном использовании подсистемы при работе с памятью.

Таблица 1

Результаты подготовительного этапа вычислительного эксперимента

N_{S^1}	max (T_a), мс	σ, мс	T_a, мс	E
1	3,38	0,141	3,38	1,00
8	3,66	0,042	0,46	7,39
16	3,94	0,028	0,25	13,73
32	4,14	0,026	0,13	26,13

На основном этапе вычислительного эксперимента трёхмерная расчетная область, имеющая размеры 1600, 1600, 200 по пространственным координатам x, y и z соответственно, разбивалась на 32 фрагмента по 50 узлов по каждой из координат x и y. Разбиение на фрагменты по координате z приведено в таблице 2. Для каждого варианта декомпозиции с десятикратной повторностью замерялось время обработки всей расчетной сетки предлагаемым параллельно-конвейерным методом и вычислялось его среднее значение T_{pm}. Ускорение E_{pm} вычислялось как отношение T_{pm} ко времени T_{sm} расчета последовательной версией алгоритма, равному 6963 мс. Получено уравнение регрессии $E_{pm} = 7,35 + 1,97 \ln(N_z^f)$ с коэффициентом детерминации, равным 0,94. Анализ результатов основного этапа вычислительного эксперимента показал существенное замедление роста E_{pm} при $N_z^f > 10$. Поэтому делаем вывод о том, что оптимальным является разбиение на фрагменты по координате z на величину, не превышающую 10.

Таблица 2

Результаты основного этапа вычислительного эксперимента

N_z^f	n_z^f	T_{pm}, мс	E_{pm}
1	200	1033,20	6,74
2	100	779,00	8,94
4	50	651,90	10,68
8	25	588,35	11,84
20	10	550,22	12,66

Обсуждение и заключение. В результате проведённых исследований разработана модель параллельно-конвейерного вычислительного процесса на примере одного из самых трудоёмких этапов решения системы сеточных уравнений модифицированным попеременно-треугольным итерационным методом. Её построение основано на моделях декомпозиции трёхмерной равномерной расчётной сетки, учитывающей технические характеристики используемого в расчетах оборудования.

Результаты, полученные в ходе вычислительных экспериментов, подтверждают эффективность разработанного метода. Подтверждена и корректность декомпозиции расчетной области на подразделы, блоки и фрагменты. Проверена работа алгоритма управления потоками, при этом выявлено, что стандартное отклонение имеет наименьшее значение в случае использования 32 параллельных потоков CPU и составляет 0,026 мс, то есть использование 32 параллельных потоков CPU при расчёте 32 фрагментов расчётной сетки даёт более равномерную по времени загруженность программных потоков. При этом среднее значение расчета одного фрагмента составило 4,14 мс.

Результаты обработки замеров времени расчетов предлагаемым параллельно-конвейерным методом показали существенное замедление роста ускорения при разбиении на фрагменты по координате z при $N_z^f > 10$. Получили, что оптимальным является разбиение на фрагменты по координате z на величину, не превышающую 10.

Список литературы

1. Shiganova T.A., Alekseenko E., Kazmin A.S. Predicting Range Expansion of Invasive Ctenophore *Mnemiopsis leidyi* A. Agassiz 1865 under Current Environmental Conditions and Future Climate Change Scenarios. *Estuarine, Coastal and Shelf Science*. 2019;227:106347. https://doi.org/10.1016/j.ecss.2019.106347

2. Sukhinov A.I., Chistyakov A.E., Nikitina A.V., Filina A.A., Lyashchenko T.V., Litvinov V.N. The Use of Supercomputer Technologies for Predictive Modeling of Pollutant Transport in Boundary Layers of the Atmosphere and Water Bodies. In book: L. Sokolinsky, M. Zymbler (eds). *Parallel Computational Technologies*. Cham: Springer; 2019. P. 225–241. https://doi.org/10.1007/978-3-030-28163-2_16

3. Rodriguez D., Gomez D., Alvarez D., Rivera S. A Review of Parallel Heterogeneous Computing Algorithms in Power Systems. *Algorithms*. 2021;14(10):275. https://doi.org/10.3390/a14100275

4. Osman A.M. Abdelrahman. GPU Computing Taxonomy. In ebook: Wen-Jyi Hwang (ed). *Recent Progress in Parallel and Distributed Computing*. London: InTech; 2017. https://doi.org/10.5772/intechopen.68179

5. Parker A. GPU Computing: The Future of Computing. In: *Proceedings of the West Virginia Academy of Science*. Morgantown, WV: WVAS; 2018. Vol. 90 (1). https://doi.org/10.55632/pwvas.v90i1.393

6. Nakano Koji. Theoretical Parallel Computing Models for GPU Computing. In book: Koç Ç. (ed). *Open Problems in Mathematics and Computational Science*. Cham: Springer; 2014. P. 341–359. https://doi.org/10.1007/978-3-319-10683-0_14

7. Bhargavi K., Sathish Babu B. GPU Computation and Platforms. In book: Ganesh Chandra Deka (ed). *Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing*. Hershey, PA: IGI Global; 2016. P. 136–174. 10.4018/978-1-4666-8853-7.ch007

8. Ebrahim Zarei Zefreh, Leili Mohammad Khanli, Shahriar Lotfi, Jaber Karimpour. 3-D Data Partitioning for 3-Level Perfectly Nested Loops on Heterogeneous Distributed System. *Concurrency and Computation: Practice and Experience*. 2017;29(5):e3976. 10.1002/cpe.3976

9. Fan Yang, Tongnian Shi, Han Chu, Kun Wang. The Design and Implementation of Parallel Algorithm Accelerator Based on CPU-GPU Collaborative Computing Environment. *Advanced Materials Research*. 2012;529:408–412. 10.4028/www.scientific.net/AMR.529.408

10. Varshini Subhash, Karran Pandey, Vijay Natarajan. A GPU Parallel Algorithm for Computing Morse-Smale Complexes. *IEEE Transactions on Visualization and Computer Graphics*. 2022. P. 1–15. 10.1109/TVCG.2022.3174769

11. Leiming Yu., Fanny Nina-Paravecino, David R. Kaeli, Qianqian Fang. Scalable and Massively Parallel Monte Carlo Photon Transport Simulations for Heterogeneous Computing Platforms. *Journal of Biomedical Optics*. 2018;23(1):010504. 10.1117/1.JBO.23.1.010504

12. Fujimoto R.M. Research Challenges in Parallel and Distributed Simulation. *ACM Transactions on Modeling and Computer Simulation*. 2016;26(4):1–29. 10.1145/2866577

13. Qiang Qin, ChangZhen Hu, TianBao Ma. Study on Complicated Solid Modeling and Cartesian Grid Generation Method. *Science China Technological Sciences*. 2014;57:630–636. 10.1007/s11431-014-5485-5

14. Seyong Lee, Jeffrey Vetter. Moving Heterogeneous GPU Computing into the Mainstream with Directive-Based, High-Level Programming Models. In: *Proc. DOE Exascale Research Conference*. Portland, Or; 2012.

15. Thoman P., Dichev K., Heller Th., Iakymchuk R., Aguilar X., Hasanov Kh., et al. A Taxonomy of Task-Based Parallel Programming Technologies for High-Performance Computing. *Journal of Supercomputing*. 2018;74(2):1422–1434. 10.1007/s11227-018-2238-4

References

1. Shiganova TA, Alekseenko E, Kazmin AS. Predicting Range Expansion of Invasive Ctenophore *Mnemiopsis leidyi* A. Agassiz 1865 under Current Environmental Conditions and Future Climate Change Scenarios. *Estuarine, Coastal and Shelf Science*. 2019;227:106347. https://doi.org/10.1016/j.ecss.2019.106347

2. Sukhinov AI, Chistyakov AE, Nikitina AV, Filina AA, Lyashchenko TV, Litvinov VN. The Use of Supercomputer Technologies for Predictive Modeling of Pollutant Transport in Boundary Layers of the Atmosphere and Water Bodies. In book: L Sokolinsky, M Zymbler (eds). *Parallel Computational Technologies*. Cham: Springer; 2019. P. 225–241. 10.1007/978-3-030-28163-2_16

3. Rodriguez D, Gomez D, Alvarez D, Rivera S. A Review of Parallel Heterogeneous Computing Algorithms in Power Systems. *Algorithms*. 2021;14(10):275. 10.3390/a14100275.

4. Abdelrahman AM Osman. GPU Computing Taxonomy. In ebook: Wen-Jyi Hwang (ed). *Recent Progress in Parallel and Distributed Computing*. London: InTech; 2017. http://dx.doi.org/10.5772/intechopen.68179

5. Parker A. GPU Computing: The Future of Computing. In: *Proceedings of the West Virginia Academy of Science*. Morgantown, WV: WVAS; 2018. Vol. 90 (1). 10.55632/pwvas.v90i1.393

6. Nakano Koji. Theoretical Parallel Computing Models for GPU Computing. In book: Koç Ç. (ed). *Open Problems in Mathematics and Computational Science*. Cham: Springer; 2014. P. 341–359. 10.1007/978-3-319-10683-0_14

7. Bhargavi K, Sathish Babu B. GPU Computation and Platforms. In book: Ganesh Chandra Deka (ed). *Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing*. Hershey, PA: IGI Global; 2016. P.136–174. 10.4018/978-1-4666-8853-7.ch007

8. Ebrahim Zarei Zefreh, Leili Mohammad Khanli, Shahriar Lotfi, Jaber Karimpour. 3-D Data Partitioning for 3-Level Perfectly Nested Loops on Heterogeneous Distributed System. *Concurrency and Computation: Practice and Experience*. 2017;29(5):e3976. 10.1002/cpe.3976

9. Fan Yang, Tongnian Shi, Han Chu, Kun Wang. The Design and Implementation of Parallel Algorithm Accelerator Based on CPU-GPU Collaborative Computing Environment. *Advanced Materials Research*. 2012;529:408–412. 10.4028/www.scientific.net/AMR.529.408

10. Varshini Subhash, Karran Pandey, Vijay Natarajan. A GPU Parallel Algorithm for Computing Morse-Smale Complexes. *IEEE Transactions on Visualization and Computer Graphics*. 2022. P. 1–15. 10.1109/TVCG.2022.3174769

11. Leiming Yu, Fanny Nina-Paravecino, David R Kaeli, Qianqian Fang. Scalable and Massively Parallel Monte Carlo Photon Transport Simulations for Heterogeneous Computing Platforms. *Journal of Biomedical Optics*. 2018;23(1):010504. 10.1117/1.JBO.23.1.010504

12. Fujimoto RM. Research Challenges in Parallel and Distributed Simulation. *ACM Transactions on Modeling and Computer Simulation*. 2016;26(4):1–29. 10.1145/2866577

13. Qiang Qin, ChangZhen Hu, TianBao Ma. Study on Complicated Solid Modeling and Cartesian Grid Generation Method. *Science China Technological Sciences*. 2014;57:630–636. 10.1007/s11431-014-5485-5

14. Seyong Lee, Jeffrey Vetter. Moving Heterogeneous GPU Computing into the Mainstream with Directive-Based, High-Level Programming Models. In: *Proc. DOE Exascale Research Conference*. Portland, Or; 2012.

15. Thoman P, Dichev K, Heller Th, Iakymchuk R, Aguilar X, Hasanov Kh, et al. A Taxonomy of Task-Based Parallel Programming Technologies for High-Performance Computing. *Journal of Supercomputing*. 2018;74(2):1422–1434. 10.1007/s11227-018-2238-4

Поступила в редакцию 17.07.2023
Поступила после рецензирования 14.08.2023
Принята к публикации 18.08.2023

Об авторах:

Владимир Николаевич Литвинов, кандидат технических наук, доцент кафедры математики и информатики Донского государственного технического университета (344003, РФ, г. Ростов-на-Дону, пл. Гагарина, 1), ResearcherID, ScopusID, ORCID, LitvinovVN@rambler.ru

Нелли Борисовна Руденко, кандидат технических наук, доцент, доцент кафедры математики и биоинформатики Азово-Черноморского инженерного института, ДГАУ (347740, РФ, г. Зерноград, ул. Ленина, 19), ScopusID, ORCID, nelli-rud@yandex.ru

Наталья Николаевна Грачева, кандидат технических наук, доцент кафедры математики и биоинформатики Азово-Черноморского инженерного института, ДГАУ (347740, Ростовская область, г. Зерноград, ул. Ленина, 19), ScopusID, ORCID, grann72@mail.ru

Заявленный вклад соавторов:

В.Н. Литвинов — формирование основной концепции, цели и задачи исследования, разработка алгоритмов для выполнения вычислительного эксперимента, проведение расчетов.

Н.Б. Руденко — анализ результатов исследований, формирование выводов, подготовка текста.

Н.Н. Грачева — написание программного кода для выполнения вычислительного эксперимента, подготовка иллюстраций, доработка текста, корректировка выводов.

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Все авторы прочитали и одобрили окончательный вариант рукописи.

Received 17.07.2023
Revised 14.08.2023
Accepted 18.08.2023

About the Authors:

Vladimir N. Litvinov, Cand.Sci. (Eng.), Associate Professor of the Mathematics and Informatics, Don State Technical University (1, Gagarin sq., Rostov-on-Don, 344003, RF), ResearcherID, ScopusID, ORCID, LitvinovVN@rambler.ru

Nelli B Rudenko, Cand.Sci. (Eng.), Associate Professor, Associate Professor of the Mathematics and Bioinformatics Department, Azov-Black Sea Engineering Institute, Don State Agrarian University (21, Lenina St., Zernograd, 347740, RF), ScopusID, ORCID, nelli-rud@yandex.ru

Natalya N Gracheva, Cand.Sci. (Eng.), Associate Professor of the Mathematics and Bioinformatics Department, Azov-Black Sea Engineering Institute, Don State Agrarian University (21, Lenina St., Zernograd, 347740, RF), ScopusID, ORCID, grann72@mail.ru

Claimed Contributorship:

VN Litvinova: basic concept formulation, research objectives and tasks, development of algorithms for performing a computational experiment, calculation analysis.

NB Rudenko: analysis of research results, drawing conclusions, text preparation.

NN Gracheva: writing program code for performing a computational experiment, preparing illustrations, finalizing the text, correction of the conclusions.

Conflict of interest statement: the authors do not have any conflict of interest.

All authors have read and approved the final manuscript.