УДК 004.273: 62-529

В.Н. Негода, А.А. Федотов

ПРИМЕНЕНИЕ ТЕХНОЛОГИИ GPGPU В АВТОМАТИЗИРОВАННОМ ПРОЕКТИРОВАНИИ СИСТЕМ УПРАВЛЕНИЯ

Негода Виктор Николаевич, *доктор технических наук, окончил радиотехнический факультет Ульяновского политехнического института, профессор кафедры «Вычислительная техника» Ульяновского государственного технического университета. Имеет статьи, монографии и авторские свидетельства в области проектирования встроенных систем контроля и управления. Область научных интересов – автоматизация проектирования логического управления техническими системами. [e-mail: nvn@ulstu.ru].*

Федотов Артем Анатольевич, *магистр техники и технологий, окончил факультет информационных систем и технологий УлГТУ. Имеет статьи в области высокопроизводительных вычислений. Область научных интересов – разработка и исследование эффективных реализаций алгоритмов в среде GPU. [e-mail: fedotov.artyom@gmail.com].*

Аннотация

Рассматриваются вопросы автоматизации проектирования программных реализаций алгоритмов управления технологическими процессами в условиях активного использования технологии GPGPU в рабочей станции САПР. Анализируются особенности проектного процесса, предлагается аналитико-экспериментальная процедура декомпозиции функций САПР с целью максимизации эффекта от применения GPGPU. Модифицированный проектный процесс базируется на известном процессе COMET, созданном специально для проектирования систем реального времени. Применение технологии GPGPU сосредоточено на этапе аналитического моделирования. Предлагается за основу создания программных моделей для выполнения динамического моделирования систем управления взять процесс формирования отображений множества связанных функций моделирования на массив параллельных процессорных ядер. Строится общая логико-алгебраическая модель такого отображения. В качестве примера применения предлагаемого подхода к аналитическому моделированию с активным использованием технологии GPGPU рассматривается одна из наиболее сложных задач проектирования системы управления электродуговых печей постоянного тока – динамическое моделирование процессов тепломассопереноса в шихте печи на основе сеточной модели. Представляются результаты экспериментов, выполненных с использованием технологии CUDA.

Ключевые слова: технология GPGPU, автоматизация проектирования систем управления технологическими процессами, распараллеливание вычислительных процедур САПР.

THE APPLICATION OF GPGPU TECHNOLOGY IN COMPUTER-AIDED CONTROL SYSTEMS DESIGN

Viktor Nikolaevich Negoda, *Doctor of Engineering; graduated from the Faculty of Radio-Engineering of Ulyanovsk Polytechnic Institute; Professor at the Department of Computer Science of Ulyanovsk State Technical University; an*

author of articles in the field of computer-aided design of embedded systems; area of expertise – computer-aided design of technical systems with logical control. e-mail: nvn@ulstu.ru.

Artem Anatolevich Fedotov, *Master of Computer Science; graduated from the Faculty of Information Systems and Technologies at Ulyanovsk State Technical University (UlSTU); an author of articles in the field of high performance computation; area of expertise – efficient computational procedure implementation on GPGPU. e-mail: fedotov.artyom@gmail.com.*

Abstract

The article examines the computer-aided design of control system implementation by using GPGPU technology. Features of design process are analyzed, analytic and experimental procedure of decomposition of the CAD functions for the purpose of maximizing utilization of GPGPU is offered. The modified design process is based on the well-known process named COMET, created specifically for designing real-time systems. Application of GPGPU technology focuses on analytical modeling stage. Creation of software models to perform dynamic simulation of control systems is proposed by mapping related modelling functions on an array of processor cores. We construct a general logic-algebraic model of such a map. As an example of the proposed approach to the analytical modeling of the active use of GPGPU technology we consider one of the most complex challenges of designing a control system of DC furnaces - dynamic modeling of heat and mass transfer in the batch furnace based on grid model. We present the results of the experiments performed using CUDA technology.

Key words: : GPGPU technology, computer-aided design of control system, parallelization of CAD procedures.

Введение

Технология использования графических процессоров для вычислений общего назначения GPGPU (General-purpose Graphics Processing Unit) в последние годы находит широкое применение в сфере высокопроизводительных вычислений и, в частности, при проектировании систем управления. Прежде всего, GPGPU используется для моделирования динамических систем средствами таких пакетов математического моделирования, которые имеют библиотеки функций, реализованных в среде GPU [1–4].

Известны также случаи использования GPGPU в САПР и системах трехмерного моделирования [1, 5–7]. Однако вопросы реализации функций САПР систем управления с активным использованием GPGPU в научно-технической литературе рассмотрены очень слабо. В частности, в работе [8] рассмотрены методы автоматизированного моделирования динамических режимов трансформаторов. В работе [9] можно найти описание практики использования GPU для проведения научных исследований на примере моделирования динамических систем.

При проектировании систем управления наиболее актуальным для применения технологии GPGPU являются задачи аналитического моделирования. Именно в этих задачах активно используются алгоритмы, обладающие большим потенциалом распараллеливания, что является необходимым условием получения положительного эффекта от применения GPU. В случаях, когда пространство проектных решений системы управления очень велико, а объекты управления описываются большими системами уравнений, процесс аналитического моделирования целесообразно строить на основе специально создаваемых программных средств. Вопросы создания таких средств рассматриваются ниже.

1 Аналитическое моделирование в проектном процессе с активным использованием технологии GPGPU

В качестве основы для базового проектного процесса целесообразнее всего выбрать метод COMET (Concurrent Object Modeling and architectural design mEThod) [10], специально разработанный для создания систем реального времени. Метод COMET предполагает выполнение шести этапов проектирования: моделирование требований, аналитическое моделирование, проектное моделирование, инкрементное конструирование программного обеспечения (ПО), инкрементная интеграция ПО, комплексное тестирование. Согласно методу COMET, основным содержанием этапа аналитического моделирования являются рассуждения с целью построения диаграмм средствами графического языка UML. Основным результатом этапа являются диаграммы классов, диаграммы деятельности, взаимодействия, состояний и последовательности [10, 11].

При создании систем управления этап аналитического моделирования дополняется процессами динамического моделирования с целью определения структурно-функциональной организации создаваемой системы, ее алгоритмов функционирования и параметров процесса управления. Динамическое моделирование предполагает выполнение следующих проектных процедур:

1. разработка вариантов структурно-функциональной организации системы управления;
2. разработка программных моделей агрегатов объекта управления и технологических процессов;
3. разработка программных моделей системы в целом для тех вариантов ее структурно-функциональной организации, которые вовлекаются в проектный процесс;
4. калибровка программных моделей;

5. аналитическое моделирование на основе созданных средств автоматизации и выбор наиболее рациональных проектных решений.

В общем случае имеет место многообразие вариантов структурно-функциональной организации системы управления даже в рамках одного проекта. Это многообразие порождается, прежде всего, возможностями варьировать набор функций системы управления и их распределение в сети контроллеров и массивах процессорных ядер, а также набор агрегатов объекта управления. Указанное обстоятельство повышает требования к быстродействию программ аналитического моделирования, что делает актуальными исследование их реализаций на основе технологии GPGPU.

При создании программ аналитического моделирования с ориентацией на мультипроцессорную среду возникает задача поиска такого отображения реализуемых функций на элементы этой среды, которое характеризуется достаточно малыми затратами времени исполнения. Поиск такого отображения является частным случаем классического двухциклового процесса, во внешнем цикле которого варьируются структурные решения, а во внутреннем – параметры отображения (рис. 1).

Варьирование структуры предполагает поиск наиболее рационального распределения функциональности между CPU и GPU. Параметрическое же варьирование

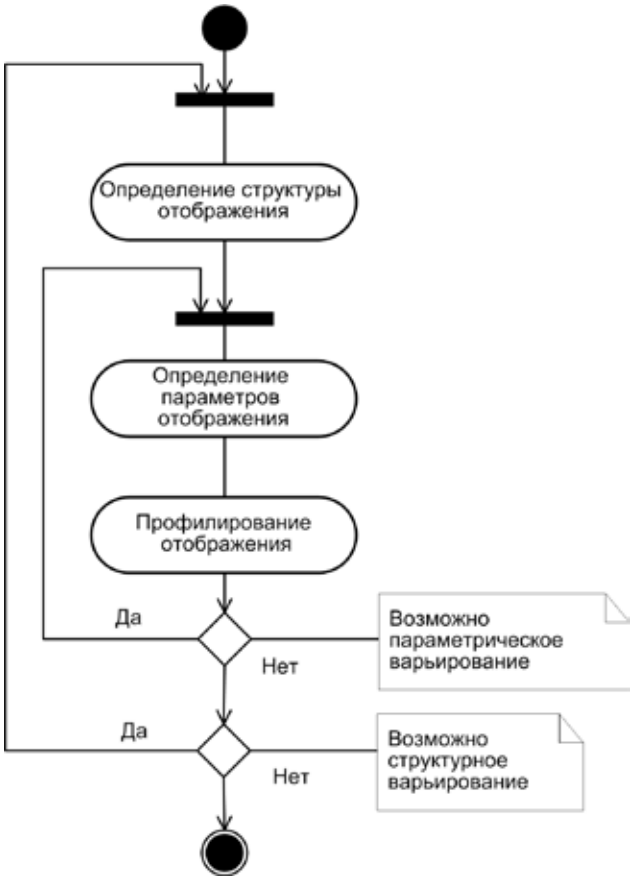

Рис. 1. Цикл разработки программы аналитического моделирования

подразумевает поиск лучшей конфигурации отображения как в среду CPU, так и в среду GPU. Поиск такой конфигурации сводится к подбору значений параметров реализации, специфичных для среды исполнения. В случае с GPU это конфигурация сетки блоков нитей, конфигурация потоков исполнения, конфигурация режима использования памяти и другое. Результатом данного этапа является сформированная конфигурация отображения модели системы управления на среду исполнения.

2 Детализация процессов отображения вычислительных задач на среду GPU

Практика использования технологии GPGPU [12–14] дает основания полагать, что многообразие проектных решений является естественным свойством процессов их программирования. Одни проектные решения отличаются от других прежде всего тем, что совокупность алгоритмов и структур данных вычислительной задачи отображается на пространство среды GPU по-разному. Кроме того, зачастую вполне допустимы алгоритмические преобразования вплоть до существенного изменения метода решения задачи. Таким образом, мы имеем дело с множеством потенциально рациональных отображений реализуемого процесса на элементы среды исполнения. Более того, порождение данного множества отображений связано с процессом последовательной детализации. Это значит, что изначально строится отображение для наиболее общих сущностей, специфицирующих задачу и среду реализации, а затем – для тех сущностей, которые детализируют процессы.

Отображение реализуемого процесса на элементы среды исполнения осуществляется в несколько шагов, среди которых доминируют два вида проектных процедур: декомпозиция и отображение. Содержание этих процедур, сформированное на основе личного практического опыта авторов данной статьи, представляется ниже.

Декомпозиция. Выполняется для исходной задачи с целью выделить множества элементов данных, которые могут обрабатываться независимо друг от друга. Предполагает выполнение декомпозиции исходной задачи, формирование групп подзадач и упорядочивание групп подзадач.

Декомпозиция исходной задачи. Выделение подзадач, которые необходимо решать в рамках отдельной нити. В результате формируется множество нитей решения $block=\{thread_i \mid i \in 1..nThread\}$.

Формирование групп подзадач. Упорядочивание подзадач в сеточную структуру. В результате каждому элементу вектора sKB сопоставляется описание структуры групп элементов подзадачи $grid=\{block_i \mid i \in 1..nBlock\}$. Это сопоставление фиксируется в соответствии $sKGB=sKB \times grid=(\langle grid,block\rangle \mid i \in 1..nKernel)$.

Упорядочивание групп подзадач. Формирование множества, состоящего из групп подзадач. В результате образуется множество функций ядра (kernel-функций),

представляющих из себя последовательность шагов, для решения исходной задачи:

$task = \{subTask_i \mid i \in nSubTask_i\}$,

где $subTask = (\langle kernel_i, sKGB_j \rangle \mid i,j \in 1..nSubTask)$.

Таким образом, в результате декомпозиции задача разбита на подзадачи и для каждой подзадачи задан вектор преобразующих функций, где для каждой функции определена структура элементов, входящих в нее:

$$task = \{kernel_i, \langle grid_j, block_j \rangle \mid i, j \in 1..nKernel\}.$$

Отображение. Выполняется для декомпозированной исходной задачи с целью планирования ее обработки на массиве процессорных ядер GPU. К этому виду проектных процедур относятся четыре отображения, описываемые ниже.

Отображение множества подзадач на множество потоков GPU. Отображение множества $sKGB$ на множество потоков GPU: $sKGB \rightarrow stream$. В результате данного отображения каждой функции преобразования сопоставляется поток исполнения:

$$sKGBS = \{\langle kernel_i, grid, block, stream_j \rangle \mid i \in 1..nKernel, j \in 1..nStream\}.$$

Отображение потока GPU на множество мультипроцессоров GPU. Отображение отдельного потока GPU на множество мультипроцессоров:

$stream \rightarrow multiProcs$,

где $multiProcs = \{MP_i \mid i \in 1..nMultiProc\}$,

где $MP = \{PRC_i \mid i \in 1..nProc\}$.

В результате элементам множества блоков нитей

$sBLK = \{BLK_i \mid i \in 1..nBlock\}$,

где $BLK = \{TH_i \mid i \in 1..nThread\}$ – множество нитей, сопоставляются мультипроцессоры, на которых осуществляется выполнение обработки элементов, входящих в данный блок:

$$multiProcs \times sBLK = \{\{BLK_i, MP_j\} \mid i \in 1..nBlock, j \in 1..nMultiProc\}.$$

Так же, каждой нити, входящей в блок, сопоставляется один из процессоров мультипроцессора: $BLK = \{TH_i, PRC_j \mid i \in 1..nThread, i \in 1..nProc\}$.

Отображение блока нитей на множество групп нитей. В результате формируется множество групп нитей, исполняемых в заданный момент времени:

$BLK \times sWRP = \{\{TH_i, WRP_j\} \mid i \in 1..nThread, j \in 1..nWarp\}$. Можно говорить о том, что в один момент времени в блоке исполняется $\left]\frac{T}{W_{size}}\right[$, где T – количество нитей в блоке, W_{size} – размер группы нитей [1].

Отображение последовательности операций нити на последовательности команд мультипроцессоров. В ходе процесса исполнения каждая нить имеет свой собственный указатель адреса инструкций и состояние регистров, поэтому они могут независимо друг от друга выполнять обработку входных данных, включая ветвления в алгоритме обработки. Таким образом, $TH = \{exSt, mpInstrGPU\}$, где $exexStState = \langle instrPtr, regState \rangle$ – вектор состояния исполнения, $mpInstrGPU = \langle mpInstrGPU_1, mpInstrGPU_2, ..., mpInstrGPU_N \rangle$ – вектор потока команд, где $mpInstrGPU_i \in instrGPU$.

В свою очередь,

$instrGPU = \{arithmInstr, jmpInstr, memInstr\}$ – множество команд GPU, включающее $arithmInstr$ – множество арифметических инструкций, $jmpInstr$ – множество инструкций условных и безусловных переходов, $memInstr = \{sharedMem, globalMem, regMem, localMem, constMem, texMem, PCIBus\}$ – множество инструкций обращения к памяти. Каждая команда имеет собственные характеристики скорости выполнения, а в случае с элементами множества $memInstr$ – также возможности параллельного обращения к одной ячейке нескольких вычислительных нитей.

Рассмотрим осуществление описанных выше проектных процедур на примере. Пусть исходная задача имеет пространственную сетку размером $taskSize = nRow \times nCol = 2048 \times 4096$ элементов, и при этом отображение выполняется для GPU GeForce 8800 GT.

Декомпозиция. Временная сетка задачи не позволяет производить параллельные вычисления. В свою очередь, пространственная сетка позволяет выполнять расчеты параллельно за счет использования метода прогонки, который подразумевает сначала параллельную прогонку по столбцам, а затем по строкам.

Декомпозиция исходной задачи. Множество элементов исходной задачи необходимо разбить на группы элементов, которые должны обрабатываться совместно. Каждой из подобных групп будет соответствовать отдельная нить вычислений. Как говорилось выше, алгоритм прогонки подразумевает 2 этапа обработки – обход по строкам и обход по столбцам. Таким образом, для первого шага группой элементов, обрабатываемых отдельной нитью являются элементы строки, а для второго – элементы столбца. Сформированы 2 множества нитей:

$$rowRunThs = \{RRT_i \mid i \in 1..2048\},$$
$$colRunThs = \{CRT_i \mid i \in 1..4096\}.$$

Для того, чтобы на GPU можно было запустить такое количество параллельных нитей их необходимо объединить в блоки.

Декомпозиция подзадачи. Объединенные в блоки нити должны запускаться на одном мультипроцессоре, и, как следствие, ограничиваться ресурсами мультипроцессора, а именно объемом разделяемой памяти, регистрами и т. п. На данном этапе необходимо определить «форму» блока нитей. Необходимо выбирать размеры блока, кратные размеру задачи, чтобы исключить из алгоритма обработки условные инструкции корректности индекса обрабатываемого элемента. В данном случае с целью

простоты возьмем $rowRunBlockDim = \langle 128,1,1 \rangle$, $colRunBlockDim = \langle 64,1,1 \rangle$. Таким образом, сформировано два множества конфигурации блоков нитей: $rowRunBks = \{RRB_i \mid i \in 1..16\}$, $colRunBks = \{CRB_i \mid i \in 1..64\}$.

Упорядочивание группы элементов подзадачи. Блоки нитей также должны быть объединены в структуру – сетку блоков. Это обусловлено наличием ограничений на требования к ресурсам отдельного блока, а также максимальным индексом нити и блока нитей – если размер задачи не позволяет уместить ее в рамках одного блока, то выполняется декомпозиция. Подобная трехуровневая декомпозиция позволяет учитывать специфику обработки на аппаратном уровне, но в данном случае она отсутствует, поэтому здесь конфигурация сетки блоков является простым следствием из конфигурации блока нитей. Это приводит к формированию двух множеств конфигурации сетки блоков нитей:

$$rowRunGridDim = \langle 16, 1, 1 \rangle,$$
$$colRunGridDim = \langle 64, 1, 1 \rangle.$$

Таким образом, в результате декомпозиции задача разбита на подзадачи и для каждой подзадачи задан вектор преобразующих функций, где для каждой функции определена структура элементов, входящих в нее:

$$timeStep = \{\langle task, rowRunKrn \rangle, \langle task, colRunKrn \rangle\},$$

где $rowRunKrn = \langle rowRunFn, rowRunBlockDim, rowRunGridDim \rangle = \langle rowRunFn, \langle 128, 1, 1\rangle, \langle 16, 1, 1\rangle\rangle$,
$colRunKrn = \langle rowRunFn, colRunBlockDim, colRunGridDim \rangle = \langle rowRunFn, \langle 64, 1, 1\rangle, \langle 64, 1, 1\rangle\rangle$.

Отображение.

Отображение множества подзадач на множество потоков GPU. В рассматриваемом алгоритме не требуется использование множества вычислительных потоков с разными ядрами, поэтому в данном случае множество потоков содержит лишь один элемент $streams = \{mainStream\}$. В результате данного отображения каждой функции преобразования сопоставляется поток исполнения:

$sKGBS = \langle\langle colRunKrn, mainStream\rangle, \langle rowRunKrn, mainStream\rangle\rangle$.

Отображение потока GPU на множество мультипроцессоров GPU. Отображение отдельного потока GPU на множество мультипроцессоров: $mainStream \rightarrow multiProcs$,
где $multiProcs = \{MP_i \mid i \in 1..14\}$,
где $MP = \{core_i \mid i \in 1..8\}$.

В результате каждому элементу множества $multiProcs$ сопоставляется подмножество блоков $kernel$-функции. В данном примере для построчного прохода нужно запустить обработку данных, структурированных в виде 16 блоков по 128 нитей; для прохода по столбцам – 64 блока по 64 нити. Исходя из правила, что блок целиком исполняется на отдельном мультипроцессоре, получаем для построчного прохода 12 мультипроцессоров, исполняющих один блок нитей, и 2 мультипроцессора с 2-мя блоками нитей; для прохода по столбцам – 6 мультипроцессоров с 4 блоками нитей и 8 мультипроцессоров, исполняющих 5 блоков нитей.

Отображение блока нитей на множество групп нитей. Данный этап отображения носит динамический характер и с трудом поддается планированию без детального анализа внутренних процессов GPU, протекающих в ходе выполнения обработки. Стоит лишь отметить, что в данном случае операция выполнения обхода строк и операция выполнения обхода столбцов имеют количество нитей в блоке, кратное размеру группы нитей (warp), равное 32.

Отображение последовательности операций нити на последовательности команд мультипроцессоров. Данное отображение является результатом трансляции C-кода в PTX-ассемблер и обычно осуществляется системой программирования.

3 Пример аналитического моделирования при проектировании систем управления дуговых печей постоянного тока

Одной из наиболее трудоемких задач аналитического моделирования в ходе проектирования систем управления электродуговых печей постоянного тока является моделирование процессов теплопереноса в объеме шихты [15]. Расчет температурного поля в объеме шихты сводится к решению системы линейных алгебраических уравнений (СЛАУ) для дискретной сетки, являющейся численным решением дифференциального уравнения, описывающего теплообмен в среде шихты.

Система уравнений первого этапа вычислений имеет вид:

$$\begin{cases} -f_y T'_{i-1,j} + \left(1+2f_y\right) T'_{i,j} - f_y T'_{i+1,j} = \\ \qquad = f_y T^k_{i,j}, i = 2,\ldots,n_y, \\ \left(1+2f_y\right) T'_{1,j} - 2f_y T'_{2,j} = T^k_{1,j}, \\ -2f_y T'_{n,j} + \left[1+2f_y\left(1+b_y\right)\right] T'_{n+1,j} = \\ \qquad = T^k_{n+1,j} + 2f_y b_y T_0 \left(n \equiv n_y\right). \end{cases}$$

Система уравнений второго этапа вычислений имеет вид:

$$\begin{cases} -f_z T^{k+1}_{i,j-1} + \left(1+2f_z\right) T^{k+1}_{i,j} - f_z T^{k+1}_{i,j+1} = \\ \qquad = T'_{i,j}, i = 2,\ldots,n_z, \\ \left(1+2f_z\right) T^{k+1}_{i,1} - 2f_z T^{k+1}_{i,2} = T'_{i,1}, \\ -2f_z T^{k+1}_{i,n} + \left[1+2f_z\left(1+b_z\right)\right] T^{k+1}_{1,n+1} = \\ \qquad = T'_{i,n+1} + 2f_z b_z T_0 \left(n \equiv n_z\right). \end{cases}$$

В этих уравнениях

$T^k_{i,j}$ – значение температуры в узле сетки (i, j) в момент времени k;

$T'_{i,j}$ – промежуточное значение температуры в узле сетки (i, j);

$$f_y = \frac{\alpha_y \Delta t}{\Delta y^2}; \quad f_z = \frac{\alpha_z \Delta t}{\Delta z^2};$$

$$a_i = \frac{f}{1 + f\left(2 - \alpha_{i-1}\right)}; \qquad \beta_j = \frac{T_j^k + fb_{i-1}}{1 + f\left(2 - \alpha_{i-1}\right)}.$$

В качестве метода решения сеточных уравнений в работе [15] используется неявная разностная схема, выбранная из-за свойства безусловной сходимости, что обеспечивает максимальную гибкость в варьировании значений шага сетки и шага по оси времени. В случае реализации метода в мультипроцессорной среде GPU/CPU алгоритмическое решение базируется на методе расщепления, который прост как в конвейерной, так и в параллельной реализациях. К тому же, одним из его достоинств является сокращенное количество вычислительных операций по сравнению с обычным методом прогонки.

В ходе описываемого эксперимента авторами построены различные отображения программных функций реализуемой модели в среду процессоров GPU, различающиеся количеством нитей параллельной обработки. Ввиду особенностей модели параллельной обработки в GPU, изменение количества узлов приводит к их иному распределению между нитями параллельной обработки, а именно изменяются количество блоков нитей и размер ячеек в сетке блоков. Алгоритм обработки строится таким образом, что для получения конечного результата необходимо выполнить два прохода по узлам сетки – по строкам и по столбцам.

При обходе по строкам строится отображение узлов сетки на множество процессорных элементов таким образом, что отдельная нить выполняет обработку всех узлов сетки в пределах отдельной строки. При распределении узлов между процессорными элементами количество нитей в блоке равно количеству узлов в столбцах сетки, но не превышает максимально возможного для конкретного GPU. Количество же блоков нитей равно наибольшему целому частному от деления количества узлов в столбцах сетки на размер блока нитей параллельной обработки. Таким образом,

$$blockSize = min(NZ, maxThreadsPerBlock),$$
$$gridSize = ceil(NZ/blockSize),$$

где NZ – количество узлов сетки в столбце (количество строк);

$maxThreadsPerBlock$ – максимальное количество нитей параллельной обработки в блоке;

min – функция, возвращающая минимальный из переданных аргументов;

$ceil$ – функция округления к наибольшему целому.

При обходе по столбцам отображение строится так, что отдельная нить выполняет обработку всех узлов сетки в пределах отдельного столбца. При распределении узлов между процессорными элементами, количество нитей в блоке равно количеству узлов в строках сетки, но не превышает максимально возможного для конкретного GPU. Количество же блоков нитей равно наибольшему целому частному от отношения количества узлов в строках сетки к размеру блока нитей параллельной обработки. Данному правилу соответствуют выражения:

$$blockSize = min(NY, maxThreadsPerBlock)),$$
$$gridSize = ceil(NY/blockSize),$$

где NY – количество столбцов.

При проведении экспериментов были использованы высокопроизводительный GPU Tesla C2075 и композиция из двух шестиядерных CPU Intel Xeon X5650. Причем, в программной модели для CPU было организовано 24 параллельных потока вычислений, что соответствует максимальному распараллеливанию для указанной композиции CPU (процессорные ядра поддерживают технологию Hyper-Threading). Начальными условиями эксперимента задан размер области нагрева шихты 0,12 м x 0,6 м при шаге по осям 0,0012 м и 0,0006 м соответственно, что дает сетку размером 102×102 узла. Моделируемое время на-

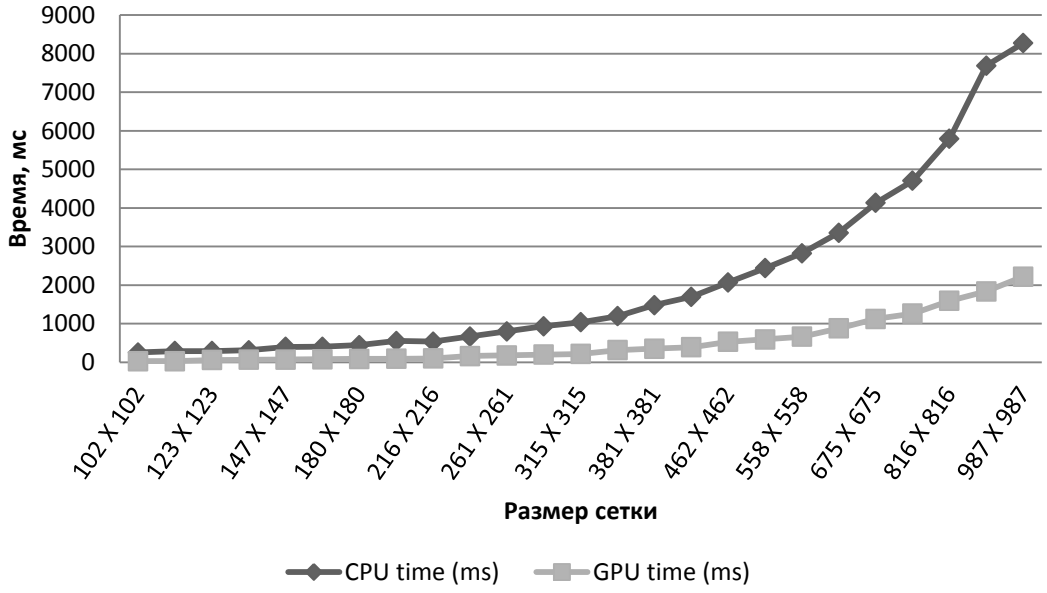

Рис. 2 Затраты времени на вычисления в среде CPU и GPU

грева составляет 1200 с с шагом 10 с. В ходе эксперимента в каждой следующей итерации размер нагреваемой области увеличивался на 10% относительно предыдущего размера, при этом остальные параметры эксперимента, такие как шаг по осям, моделируемое время и шаг по времени, оставались неизменными. Результаты эксперимента представлены на рисунке 2.

Результаты замеров времени вычисления показывают, что в большей части циклов расчета температурных полей время, затраченное GPU, примерно в 4 раза меньше времени, затрачиваемого CPU.

Заключение

Применение технологии GPGPU позволяет достигать более высоких показателей быстродействия программных моделей даже при моделировании небольшой части процессов. При увеличении количества программных моделей, вовлекаемых в аналитическое моделирование, следует ожидать более существенного выигрыша за счет более высокого уровня параллельности вычислений в среде GPU. В частности, в ходе автоматизированного проектирования систем управления электродуговых печей возникает потребность в моделировании динамических процессов гидропривода, регулятора тока дуги, систем охлаждения, энергосистемы, что порождает задачу распределения в пространстве массивов процессорных ядер программных функций нескольких моделей.

СПИСОК ЛИТЕРАТУРЫ

1. GPU Accelerated Applications. – URL : www.nvidia.ru/content/tesla/pdf/gpu-accelerated-applications-for-hpc.pdf.

2. Comparison of Some New Maple 15 Features with Mathematica 8. – URL : www.wolfram.com/mathematica/compare-mathematica/files/ Maple15NewFeatures.pdf.

3. White Paper: CUDA Programming within Mathematica. – URL : www.wolfram.com/products/CUDA_Programming_within_Mathematica.pdf.

4. Компьютерная модель гидроакустической телекоммуникационной системы в среде MatLab. / И.Н. Бурдинский [и др.] // Проектирование инженерных и научных приложений в среде MATLAB : тр. V-й Междунар. науч. Конф. – Харьков: ФЛП Е.В.Шейнина, 2011. – С. 447–452.

5. Get the most out of AutoCAD with NVIDIA GPUs. – URL : www.nvidia.pl/content/quadro/autodesk-suites/pdf/4723-SO-AutoCAD-LR.pdf.

6. Get the most out of Autodesk 3ds Max with NVIDIA. – URL : nvidia.com/content/quadro/odesk-suites/pdf/3dsMax-Solution-Overview-Nov13.pdf.

7. Get the most out of AutoCAD With NVIDIA GPUs. – URL : www.nvidia.pl/content/quadro/autodesk-suites/pdf/4723-SO-AutoCAD-LR.pdf.

8. Климов Д.А., Попов Г.В., Тихонов А.И. Методы автоматизированного моделирования динамических режимов трансформаторов / ГОУВПО «Ивановский государственный энергетический университет им. В.И. Ленина». – Иваново, 2006. – 100 с.

9. Бурый А.А., Поздняк П.Л. Практика использования GPU для проведения научных исследований на примере моделирования динамических систем // XXXV-я Дальневосточная математическая школа им. Золотова : тез. докл. – Владивосток, 2010. – С. 839–844.

10. Гома Х. UML. Проектирование систем реального времени, распределенных и параллельных приложений. – М. : ДМК Пресс, 2002. – 704 с.

11. Буч Г., Рамбо Дж., Якобсон А. Язык UML : Руководство пользователя. – М. : ДМК, 2000. – 432 с.

12. Федотов А.А. Эксперименты с реализациями алгоритма быстрого преобразования Фурье средствами NVIDIA CUDA // Современные информационные технологии: тр. межд. науч.-техн. конф. – Пенза: Пензенская государственная технологическая академия, 2012. – Вып. 16. – С. 132–135.

13. Федотов А.А. Исследование параллельных реализаций алгоритмов умножения матриц в среде GPU // Информатика и вычислительная техника : сб. науч. тр. / под ред. В.Н. Негоды. – Ульяновск : УлГТУ, 2010. – С. 506–510.

14. Федотов А.А., Негода В.Н. Средства поддержки профилирования программ в среде NVIDIA GPU // Информатика и вычислительная техника : сб. науч. тр. / под ред. Н.Н. Войта. – Ульяновск : УлГТУ, 2011. – С. 579–584.

15. Арутюнов В.А., Бухмиров В.В., Крупенников С.А. Математическое моделирование тепловой работы промышленных печей : учебник для вузов. – М. : Металлургия, 1990. – 239 с.

REFERENCES

1. GPU Accelerated Applications. Available at: http:// www.nvidia.ru/content/tesla/pdf/gpu-accelerated-applications-for-hpc.pdf.

2. Comparison of Some New Maple 15 Features with Mathematica 8. Available at: http://www.wolfram.com/mathematica/compare-mathematica/files/ Maple15NewFeatures.pdf.

3. White Paper: CUDA Programming within Mathematica. Available at: http:// www.wolfram.com/products/CUDA_Programming_within_Mathematica.pdf.

4. Burdinskiy I.N. and Others. Kompyuternaya model gidroakusticheskoy telekommunikatsionnoy sistemy v srede MatLab [Computer Model of the Hydroacoustic Telecommunication System MatLab Medium]. *Proyektirovaniye inzhenernykh i nauchnykh prilozheniy v srede MATLAB: Trudy 5 Mezhdunarodnoy nauchnoy Konferentsii* [Design of Engineering and Science Applications in MATLAB Medium. Proc. 5th Int. Conf.], Kharkiv: FLP E.V. Sheynina Publ., 2011, pp. 447–452.

5. Get the most out of AutoCAD with NVIDIA GPUs. Available at: http://www.nvidia.pl/content/quadro/autodesk-suites/pdf/4723-SO-AutoCAD-LR.pdf.

6. Get the most out of Autodesk 3ds Max with NVIDIA. Available at: http://nvidia.com/content/quadro/odesk-suites/pdf/3dsMax-Solution-Overview-Nov13.pdf.

7. Get the most out of AutoCAD With NVIDIA GPUs. Available at: http://www.nvidia.pl/content/quadro/autodesk-suites/pdf/4723-SO-AutoCAD-LR.pdf.

8. Klimov D.A., Popov G.V., Tikhonov A.I. *Metody avtomatizirovannogo modelirovaniya dinamicheskikh rezhimov transformatorov* [Methods of Computer-Aided Modeling of Transformer Dynamic Conditions]. Ivanovo, GOUVPO 'Ivanovskiy gosudarstvennyy energeticheskiy universitet im. V.I. Lenina' Publ., 2006. 100 p.

9. Buryy A.A., Pozdnyak P.L. Praktika ispolzovaniya GPU dlya provedeniya nauchnykh issledovaniy na primere modelirovaniya dinamicheskikh sistem [GPU Usage for Scientific Research by way of Dynamic Systems Modeling Example]. *XXXV-ya Dalnevostochnaya matematicheskaya shkola im. Zolotova: tez. dokl.* [Proc. XXXV Zolotov Far Eastern Mathematical School-Seminar]. Vladivostok, 2010, pp. 839–844. (In Russian)

10. Goma H. UML. *Proyektirovaniye sistem realnogo vremeni, raspredelennykh i parallelnykh prilozheniy* [Design Concurrent, Distributed, and Real-Time Applications with UML]. Moscow, DMK Press Publ., 2002. 704 p.

11. Booch G., Rumbaugh J., and Jacobson I. Yazyk UML: *Rukovodstvo polzovatelya* [The Unified Modeling Language: Users Guide]. Moscow, DMK Publ., 2000. 432 p.

12. Fedotov A.A. Eksperimenty s realizatsiyami algoritma bystrogo preobrazovaniya Furye sredstvami NVIDIA CUDA [Experiments of Fast-Fourier-Transform Algorithm with NVIDIA CUDA]. *Sovremennyye informatsionnyye tekhnologii: tr. mezhd. nauch.-tekhn. konf.* [Proc. of Scientific Conf.: Modern Information Technologies], Penza, Penzenskaya gosudarstvennaya tekhnologicheskaya akademiya Publ., 2012, iss. 16, pp. 132–135.

13. Fedotov A.A. Issledovaniye parallelnykh realizatsiy algoritmov umnozheniya matrits v srede GPU [Parallel Matrix-Multiplication-Algorithm Research on GPU]. *Informatika i vychislitelnaya tekhnika: sb. nauch. tr. pod red. V.N. Negody* [Computer Science: Proceedings under Editorship of V.N. Negoda], Ulyanovsk, UlSTU Publ., 2010, pp. 506–510.

14. Fedotov A.A., Negoda V.N. Sredstva podderzhki profilirovaniya programm v srede NVIDIA GPU [Program Profiling Support on NVIDIA GPU]. *Informatika i vychislitelnaya tekhnika : sb. nauch. tr. pod red. N.N. Voyta* [Computer Science: Proceedings under Editorship of N.N. Voyt], Ulyanovsk, UlSTU Publ., 2011, pp. 579–584.

15. Arutyunov V.A., Bukhmirov V.V., Krupennikov S.A. *Matematicheskoye modelirovaniye teplovoy raboty promyshlennykh pechey: uchebnik dlya vuzov* [Mathematical Modeling the Thermal Performance of Industrial Furnaces]. Moscow, Metallurgiya Publ., 1990. 239 p.