УДК 519.6

РЕШЕНИЕ УРАВНЕНИЯ ПЕРЕНОСА НЕЙТРОНОВ И ГАММА-КВАНТОВ МЕТОДОМ МОНТЕ-КАРЛО НА ЭВМ С ИСПОЛЬЗОВАНИЕМ АРИФМЕТИЧЕСКИХ УСКОРИТЕЛЕЙ

И. В. Семенов, А. Г. Малькин, А. С. Рыбкин
(РФЯЦ-ВНИИЭФ, г. Саров)

Представлены результаты разработки программного комплекса СМК-У, предназначенного для решения стационарного уравнения переноса нейтронов и гамма-квантов с использованием арифметических ускорителей. Описаны основные алгоритмические изменения по сравнению с программой СМК. Проведены численные исследования быстродействия на ряде задач из области атомной энергетики.

Ключевые слова: арифметический ускоритель, графический процессор, метод Монте-Карло, распараллеливание по событиям, тепловыделяющий элемент, тепловыделяющая сборка, ВВЭР-1 000.

Введение

Метод Монте-Карло — один из основных методов численного решения уравнения переноса, преимуществом которого является возможность получения решения с заданной точностью. Однако для этого может потребоваться большое количество времени и вычислительных ресурсов, причем основная нагрузка приходится на моделирование траекторий частиц. В программах решения уравнения переноса традиционная организация счета *по траекториям* хорошо масштабируется при распараллеливании на многопроцессорных вычислительных системах (МВС). Современные комплексы программ, такие как ПРИЗМА [1], MCU [2] и MCNP [3], используя тысячи процессоров, имеют весьма высокие коэффициенты эффективности (более 90 %).

На сегодняшний день наращивание вычислительных мощностей МВС идет по двум основным направлениям. Первое направление — увеличение количества универсальных процессоров с архитектурой x86-64, PowerPC и т. п. Ограничением при данном подходе является высокое потребление электроэнергии такими системами. Второе направление — использование так называемых сопроцессоров (арифметических ускорителей (АрУ) и т. д.). МВС, имеющие *гибридную* архитектуру, потребляют меньшее количество электроэнергии при той же вычислительной мощности. Однако архитектура ускорителей кардинально отличается от архитектуры x86.

АрУ представляет собой мощное *параллельное* устройство с общей памятью, способное одновременно обрабатывать тысячи потоков. Однако эти потоки не являются полностью независимыми. АрУ — результат технического развития графических видеокарт, основной задачей которых являлось выполнение одинаковых вычислений над различными данными. Поэтому все потоки разбиты на равные группы, которые выполняют векторные операции. Все потоки группы имеют общий кэш и быструю разделяемую память, явно контролируемую пользователем. Это дает программисту дополнительный рычаг для повышения эффективности выполнения программы. С другой стороны, сложность разработки больших программных комплексов существенно возрастает.

В публикациях [4—12], посвященных описанию программ для решения уравнения переноса гамма-квантов методом Монте-Карло с использованием АрУ, сообщается об ускорении программ более чем в 11 раз. Однако оно достигается за счет применения упрощенных моделей взаимодействия частиц с веществом с ограничениями на количество реакций, длину траекторий, спектр энергий, за счет использования одногрупповых сечений, упрощенной геометрии задач и т. д.

Если при решении уравнения переноса применять более реалистичные математические модели, то логическая сложность программы резко возрастает. При этом традиционный алгоритм распараллеливания по траекториям приведет к падению эффективности использования АрУ. Из-за больших различий между траекториями частиц все потоки АрУ будут выполняться последовательно.

Расчет одной траектории заключается в моделировании определенных событий (свободные пробеги, столкновения, пересечения границ и т. п.). Общее число и последовательность наступления этих событий для каждой траектории свои. Поэтому на АрУ имеет смысл параллельно моделировать не траектории, а одинаковые события для группы частиц.

Предлагаемый алгоритм был реализован в программном комплексе СМК-У [13, 14], который является развитием программы СМК [15].

Эффективность выполнения программы в значительной степени связана с организацией доступа к памяти АрУ. Из-за ограничений на размер памяти и скорость доступа к ней организация данных в программе существенно влияет на скорость вычислений. На АрУ также отсутствует возможность атомарного доступа к области памяти для чисел с двойной точностью. Поэтому для увеличения эффективности использования вычислительной мощности АрУ, помимо алгоритма моделирования частиц, была изменена структура исходного программного комплекса. Общая организация хранения данных и алгоритмы расчета различных функционалов от решения были разработаны заново, с учетом использования новой архитектуры.

Оценки коэффициентов ускорений программного комплекса СМК-У по сравнению с СМК проводились на ряде задач из области атомной энергетики. Результаты тестирования свидетельствуют о высокой эффективности использования АрУ для параллельных вычислений методом Монте-Карло.

Архитектура АрУ

АрУ хорошо подходят для вычислений с распределенными данными. Графические карты от компаний NVIDIA и AMD построены на базе нескольких десятков потоковых мультипроцессоров, которые являются устройствами параллельных вычислений. Мультипроцессор имеет архитектуру, называемую Single Instruction Multiple Threads (одна инструкция — множество потоков), предназначенную для управления тысячами потоков, исполняющих несколько различных подпрограмм.

Для программирования с учетом новой архитектуры компания NVIDIA в 2006 г. представила программную технологию CUDA, основанную на расширении языка программирования C/C++ [16]. В ней универсальный процессор определен как *хост* (host), т. е. он занимается управлением и загрузкой АрУ, в то время как АрУ работает как сопроцессор. Потоки на АрУ исполняются *группами* (warp), объединенными в *блоки* (block), а блоки, в свою очередь, собраны в *сеть* (grid). Общее число потоков равно числу блоков, умноженному на число потоков в блоке.

Вся память АрУ является внешней по отношению к универсальному процессору, ее можно разбить на два типа. Первый тип памяти имеет большой объем — от 3 до 6 Гбайт, но она относительно медленная (требуется около 400—600 тактов на чтение и запись данных). К ней относятся локальная, глобальная и текстурная память. Второй тип памяти имеет размер 64 Кбайта, и она очень быстрая (требуется менее 4 тактов на доступ к данным). Это регистровая, разделяемая память, константный кэш и текстурный кэш.

Алгоритм распараллеливания траекторий по событиям

При решении линейного уравнения переноса методом Монте-Карло все траектории частиц считаются независимыми и поэтому могут быть промоделированы по отдельности (алгоритм распараллеливания по траекториям). Благодаря этой особенности метод хорошо масштабируется на тысячи процессоров с высокой эффективностью.

Однако такой подход в МВС, имеющих гибридную архитектуру, не будет эффективным. Оптимальный режим работы АрУ достигается, когда все потоки в группе выполняют одинаковые арифметические операции над различными данными. Если в программе для части потоков группы выполнится некоторый условный переход, а для остальной части — нет, то данный участок программы будет выполняться последовательно: сначала арифметические операции при одном условии, а потом при другом.

Таким образом, из-за большого количества ветвлений в методе Монте-Карло считать всю

траекторию целиком на АрУ неэффективно. Все траектории сильно отличаются друг от друга по количеству и последовательности наступления событий (свободный пробег, столкновения, пересечение границы и т. п.). Однако, если рассмотреть все траектории в совокупности, то можно заметить, что все события, которые в них происходят, составляют определенный конечный набор. Моделирование события заключается в пересчете фазовых параметров частиц по определенным математическим формулам. Поэтому на АрУ имеет смысл параллельно считать одинаковые события для некоторого набора частиц, а не всю траекторию целиком. Такой способ счета авторы назвали *алгоритмом распараллеливания по событиям*. Опишем его более подробно.

Одновременно на каждом АрУ происходит моделирование пакета траекторий заданного количества частиц. Счет пакета продолжается до тех пор, пока в нем не останется ни одной частицы. В начале расчета фазовые параметры частиц, испущенных из источника, заносятся в массивы, размеры которых определяются числом частиц в пакете. Весь набор фазовых параметров делится на непересекающиеся множества в зависимости от события, которое должно быть рассчитано в данный момент. В первом множестве находятся все частицы, для которых должны быть разыграны фазовые параметры после столкновения (новая энергия и направление полета). Второе множество содержит частицы, для которых разыгрываются столкновения (изотоп и тип реакции). В третьем множестве — частицы, для которых нужно рассчитать свободный пробег (расчет полного сечения в области и расстояния до границы области). На АрУ запускается соответствующая процедура (моделирующая свободный пробег, столкновение или разыгрывающая параметры частицы) одновременно для всех частиц одного из этих множеств, имеющего максимальный размер. Таким образом, все потоки выполняют одну и ту же подпрограмму с фазовыми параметрами различных частиц. После вызова процедуры обработки события все частицы снова перераспределяются по множествам.

В начале расчета для всех частиц в точках рождения нужно рассчитать свободный пробег, поэтому первый шаг начинается с третьего множества. После того как один свободный пробег будет промоделирован, часть частиц столкнется в текущей области, другая часть перелетит через границу области, т. е. частицы уже будут разбиты на различные множества. На втором и последующих шагах будет обрабатываться множество максимального размера.

Схематическое изображение предложенного алгоритма представлено на рис. 1 (см. также цветную вкладку). На нем частицы (клетки) из одного множества помечены одинаковым цветом. Стрелками обозначено выполнение соответствующей процедуры для всех частиц данного множества, имеющего максимальный размер на текущем шаге.

При некоторых столкновениях нейтронов происходят реакции (n, 2n), (n, 3n) и деление, в результате чего рождаются новые частицы, которые необходимо промоделировать по отдельности. Для этого в массивах, где хранятся фазо-

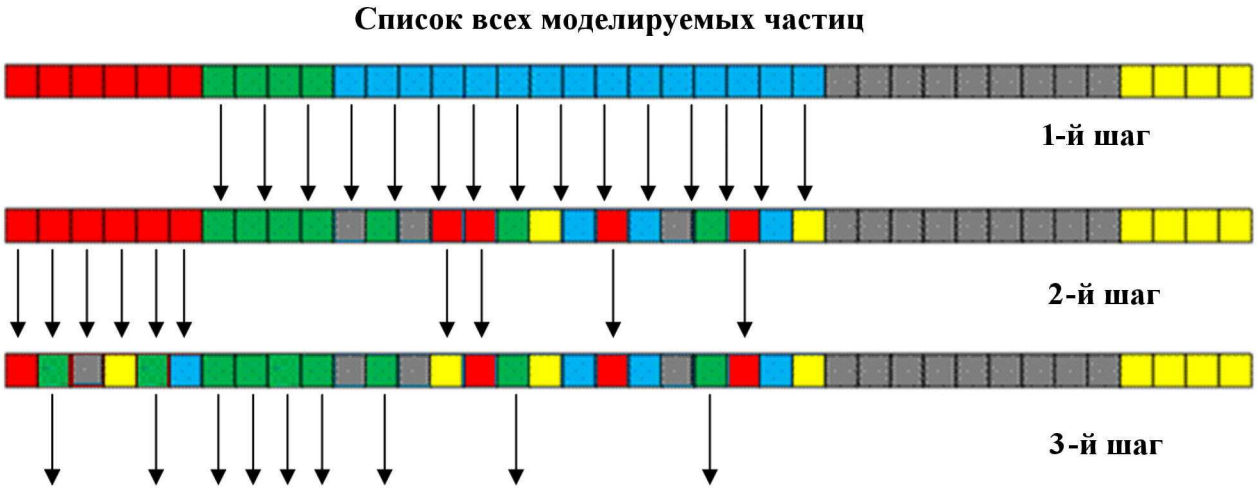

Рис. 1. Моделирование событий для всех частиц пакета, принадлежащих одному множеству

вые параметры частиц, имеются свободные ячейки, в которые записываются параметры частиц, рожденных в процессе моделирования. При гибели частицы (поглощение или вылет за внешнюю границу системы) соответствующие ячейки освобождаются и могут быть использованы для записей рожденных частиц.

Для более удобной работы с множествами частиц, свободными ячейками, точками ветвления каждой частице из пакета присваивается свой уникальный идентификатор, обозначающий ее текущее состояние (множество, которому принадлежит частица):

1 — розыгрыш параметров частицы (энергии и направления полета) после столкновения;

2 — розыгрыш изотопа, на котором произошло столкновение, и типа реакции;

3 — розыгрыш свободного пробега;

4 — частица после реакции (n, 2n);

5 — частица после реакции (n, 3n);

6 — частица после реакции деления;

7 — погибшая частица (пустая ячейка).

После расчета очередного события для всех частиц, имеющих идентификатор 4, 5 или 6, происходит копирование их фазовых параметров в пустые ячейки. Число копий равно числу образовавшихся частиц. Всем этим частицам присваивается идентификатор 1, далее определяется, которое из множеств частиц (с идентификаторами 1, 2 или 3) имеет наибольший размер, и для него запускается соответствующая процедура. На рис. 2 изображена блок-схема предложенного алгоритма.

В процессе моделирования число частиц постепенно уменьшается из-за поглощений или вылетов из системы. Когда их количество станет достаточно малым, данные о фазовых параметрах копируются на хост и оставшиеся траектории досчитываются на универсальном процессоре по тому же алгоритму. Это обусловлено тем, что при определенном количестве частиц в пакете время расчета с использованием АрУ больше по сравнению с расчетом на одном ядре универсального процессора. Дело в том, что при небольшом количестве частиц АрУ загружен не полностью и большая часть мультипроцессоров не участвует в вычислениях. Количество частиц, при котором АрУ более эффективен, чем одно ядро универсального процессора, определялось численно и для большинства задач составило 1 024.

Следующее усовершенствование алгоритма связано с обработкой геометрических данных. Одной из особенностей программы СМК-У является использование понятия геометрического блока для описания геометрии задачи. Геометрический блок — это совокупность поверхностей и областей определенного типа. В программе реализованы следующие типы блоков: сферический, осесимметричный и трехмерный. С помощью геометрических блоков описываются отдельные части геометрии задачи, затем блоки размещаются друг относительно друга так, чтобы составить всю геометрию системы. Такой подход позволяет описать геометрию практически любой сложности, но это приводит к увеличению вычислительной нагрузки на процедуру расчета расстояния до границы области.

Для каждого из типов блоков используется свой алгоритм расчета расстояния до поверхностей. Если в задании геометрии системы используются блоки различных типов, то эффективность АрУ уменьшается. Это происходит потому, что для частиц третьего множества расчет расстояния до границы запускается одновременно, но производится по разным алгоритмам, в зависимости от того, в блоке какого типа находится частица в данный момент, а значит, выполняется последовательно. Чтобы решить эту проблему, все частицы, для которых необходимо рассчитать расстояние до границы, разбиваются дополнительно на небольшие подмножества по типу геометрического блока, в котором они находятся. Расчет расстояния запускается отдельно для каждого подмножества. Благодаря этому потоки, вычисляющие расстояние для одного подмножества, выполняют одни и те же арифметические операции, что позволяет более эффективно использовать АрУ. Численные исследования показали, что дополнительное разбиение на подмножества дает выигрыш по времени выполнения процедуры расчета расстояния до границы блока до 20 % и более, в зависимости от сложности геометрии системы.

Организация хранения общих данных

Из-за ограничений на размер памяти и скорость доступа к ней на АрУ организация данных существенно влияет на эффективность вычислений. Идеальным является такое расположение данных, когда группа потоков одновременно обращается к памяти за одними и теми же или подряд расположенными данными. Такой доступ называется параллельным, и считывание данных происходит за один запрос. Ес-

Рис. 2. Блок-схема алгоритма распараллеливания по событиям

ли же данные в памяти не расположены друг за другом, то за один запрос происходит их считывание только для одного потока и доступ к памяти для всей группы будет последовательным. Это приведет к большим задержкам на ожидание необходимой информации. В силу того, что траектории частиц сильно различаются, добиться параллельного доступа к памяти непросто и не всегда возможно. Однако в большинстве случаев время на ожидание данных удается существенно сократить.

Вся информация о геометрии и изотопном составе областей хранится в глобальной памяти АрУ. Так как память этого типа имеет низкую скорость на чтение и запись, было уделено особое внимание способу хранения данных в ней.

В программе СМК вся информация о геометрии хранится в виде сложной древовидной структуры. Для частиц, находящихся в разных областях, одни и те же параметры физически расположены в разных частях памяти. Для более эффективного доступа к данным при использовании АрУ структуры были преобразованы в одномерные массивы: параметров блоков, параметров областей, матриц перехода между системами координат блоков и т. д.

Выбор способа размещения в памяти каждого геометрического массива определяется индивидуально, в зависимости от того, как он используется во время выполнения программы. Приведем два примера.

В качестве первого примера рассмотрим массив для хранения информации о параметрах геометрических объектов (блоков — A, областей — B). Обращение к их значениям происходит из разных частей программы, и при этом необходимо только одно: либо A, либо B. Для параллельного доступа к памяти оптимальным будет сначала расположить первый параметр всех объектов, затем второй и т. д. (рис. 3). При такой организации велика вероятность того, что потоки в группе будут запрашивать рядом расположенные данные, следовательно, такой запрос будет объединен в один.

В качестве второго примера рассмотрим массив для хранения центров локальных систем координат (3 координаты: X, Y и Z) и матриц поворота (9 значений M_{ij}). При переходе в локальную систему координат геометрического блока

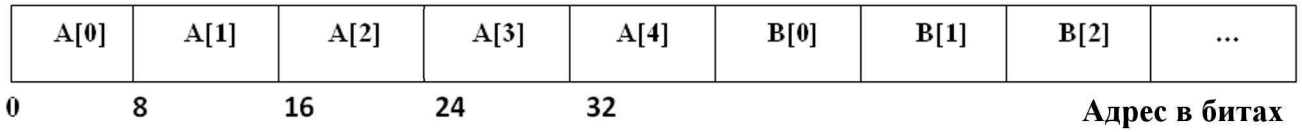

Рис. 3. Структура хранения информации о геометрии

программе необходимы все 12 значений. Поэтому, использование предыдущего подхода приведет к тому, что каждая группа потоков будет последовательно запрашивать из памяти по одному значению для всей группы. Ситуация усугубляется еще и тем, что траектории находятся в разных геометрических блоках. С другой стороны, можно заметить, что данные из рассматриваемого массива в математических формулах используются тройками. Следовательно, имеет смысл хранить все значения вместе, друг за другом, в одномерном массиве типа `double3` (рис. 4). Это позволит при обращении к элементам массива каждому потоку из группы считывать несколько значений за один запрос. Даже если каждый поток будет запрашивать данные из разных частей памяти, такой способ сократит время на их ожидание.

Организация накопления и хранения результатов

Алгоритмы расчета и накопления различных результатов в процессе решения уравнения переноса методом Монте-Карло являются важнейшей частью программного комплекса СМК-У. Все результаты можно разделить на три основные категории:

1) *стандартные интегральные* (параметры источника, интегральные значения количества реакций по всей системе, оценки для эффективного коэффициента размножения и т. д.) — их количество постоянно и не зависит от задачи;
2) *стандартные областные и поверхностные* (потоки через поверхности, количество столкновений по областям, блокам и т. д.) — их количество определяется задачей;
3) *заказные* — состав и количество определяются задачей и запросами пользователя.

Каждая категория имеет свои особенности по количеству накапливаемых результатов и реализации алгоритма накопления. По причине отсутствия на АрУ атомарного доступа для записи вещественных данных с двойной точностью возник вопрос об эффективной организации накопления интегральных результатов.

Стандартные интегральные результаты являются одними из самых важных, поэтому рассчитываются и выдаются всегда. При этом необходимо накапливать всего 26 результатов для интегральных характеристик системы и 9 значений для различных оценок эффективного коэффициента размножения нейтронов. В силу малого количества стандартных результатов было решено хранить их отдельные копии для каждого потока. Таким образом, поток может независимо от других рассчитывать только свои результаты, и проблема синхронизации в данном случае отпадает. В конце моделирования пакета частиц все рассчитанные результаты суммируются. При таком подходе для размера пакета в 1 млн частиц понадобится всего 272 Мбайт глобальной памяти, что не критично. Схема организации накопления стандартных интегральных результатов представлена на рис. 5.

Количество стандартных областных и поверхностных результатов может сильно меняться в зависимости от решаемой задачи. Например,

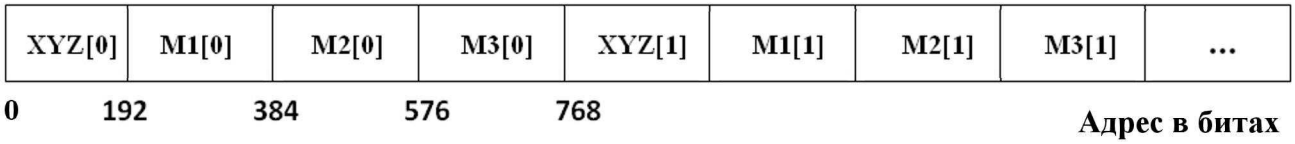

Рис. 4. Структура хранения информации о центрах локальных систем координат блоков и матрицах поворота

Рис. 5. Схема организации накопления стандартных интегральных результатов

если взять геометрию активной зоны (АЗ) реактора ВВЭР-1 000, то необходимо рассчитать 2 327 областных и 1 076 поверхностных результатов. Если же взять геометрию тепловыделяющей сборки (ТВС), количество результатов сократится на порядок. Простые оценки показывают, что при использовании предыдущего подхода для пакета, состоящего из 1 млн частиц, потребуется $\sim$ 26 Гбайт памяти. При этом максимальный объем памяти, доступный на современных АрУ, не превышает 6—8 Гбайт.

Уменьшить количество выделяемой памяти, необходимой для накопления стандартных областных и поверхностных результатов, можно, воспользовавшись особенностью организации потоков на АрУ. Все потоки разделяются на равные блоки с фиксированным числом потоков (это значение определяет пользователь) — в рассматриваемом случае размер блока равнялся 128; при этом имеются служебные функции для синхронизации потоков в рамках одного блока. В этих условиях можно хранить копию массива всех областных и поверхностных результатов не для каждого счетного потока, а для каждого блока потоков. После моделирования пакета частиц все рассчитанные результаты суммируются. Для такого способа понадобится всего 203 Мбайт глобальной памяти. Схема организации накопления стандартных областных и поверхностных результатов представлена на рис. 6.

Предложенный подход для накопления областных и поверхностных результатов имеет ряд положительных моментов. Во-первых, объем используемой памяти довольно мал, даже если необходимо накапливать и рассчитывать очень большое количество значений. Во-вторых, достаточно мало время исполнения, так как синхронизация потоков в блоке происходит только в конце моделирования события. Как будет показано ниже, время, затрачиваемое на расчет стандартных областных и поверхностных результатов, составляет около 5—10 % от общего времени выполнения программы.

Последнюю категорию составляют заказные результаты, их количество зависит не только от параметров решаемой задачи, но и от запросов пользователя. Количество накапливаемых значений ничем не ограничено. При определенных условиях использование предыдущих подходов приведет к нехватке памяти на АрУ. В качестве одного из решений этой проблемы можно использовать возможность АрУ писать данные напрямую в область памяти универсального процессора, объем которой намного больше, чем на АрУ. Однако недостатком такого подхода является то, что скорость передачи данных с центрального процессора на АрУ и обратно довольно медленная, она ограничена аппаратными возможностями PCI-E (Peripheral Component Interconnect Express).

Сравнение быстродействия

Сравнение быстродействия программ СМК-У и СМК проводилось на ряде задач атомной энергетики с использованием одного АрУ и одного ядра универсального процессора. Были взяты задачи с детальным описанием геометрии, так как на них можно более полно оценить эффективность предложенного алгоритма. На задачах с простой геометрией использование АрУ не эф-

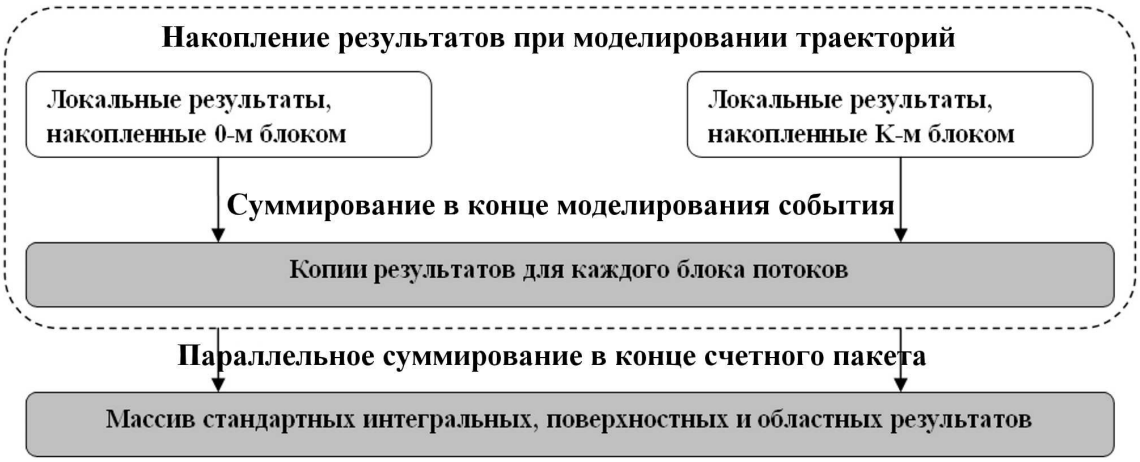

Рис. 6. Схема организации накопления стандартных областных и поверхностных результатов

фективно, так как в этом случае отношение времени подготовки данных ко времени их обработки возрастает.

Все результаты, полученные по двум программам, совпадают с точностью до статистической погрешности.

Тест 1. Расчет АЗ реактора. В качестве первого теста был выбран расчет эффективного коэффициента размножения нейтронов одного из состояний АЗ реактора ВВЭР-1 000 из работы [2]. Поперечное сечение АЗ представлено на рис. 7. Реактор состоит приблизительно из 100 ТВС, каждая из которых содержит 312 тепловыделяющих элементов (ТВЭЛов) (см. справа на рис. 7). ТВС могут отличаться друг от друга различным химическим составом ТВЭЛов.

Такое детальное описание задачи сильно затрудняет проведение расчетов, так как в целом в задаче задано около 10 000 областей. К тому же необходимо получить несколько десятков тысяч областных и поверхностных результатов.

Для упрощения расчета использовалось свойство симметрии АЗ, поэтому расчеты проводились на 1/6 части АЗ с применением тактики отражения на границах.

При выполнении расчетов варьировались следующие параметры:

- размер пакета частиц: $2^{11} \div 2^{19}$;
- использование или неиспользование при расчете модели теплового движения ядер среды;
- тип получаемых результатов: стандартные (порядка 100 значений) или стандартные плюс заказные (порядка 10 000 значений).

Зависимость коэффициентов ускорения от размера пакета частиц при выполнении программы СМК-У представлена на рис. 8. Из рисунка можно сделать следующие выводы:

- коэффициент ускорения растет при увеличении размера пакета частиц. Это объясняется тем, что с ростом размера пакета ускоритель большую часть времени работает при полной загрузке, т. е. практически все потоковые мультипроцессоры участвуют в вычислениях. Дальнейшее увеличение размера пакета слабо влияет на коэффициент ускорения;
- алгоритм накопления заказных результатов достаточно эффективен. Расчет порядка 10 000 дополнительных величин слабо сказывается на времени счета;
- при учете теплового движения ядер среды коэффициент ускорения ниже. Это объясняется более сложным и разветвленным алгоритмом моделирования.

Помимо расчета эффективного коэффициента размножения нейтронов АЗ ВВЭР-1 000, программа была протестирована на АЗ реакторов типа БН и СВБР. Максимальные значения коэффициентов ускорения в этих задачах получились равными 20 (при учете теплового движения ядер среды).

В большинстве случаев предприятия, занимающиеся расчетами реакторов, интересует не полная АЗ реактора, а отдельная ТВС. Для таких

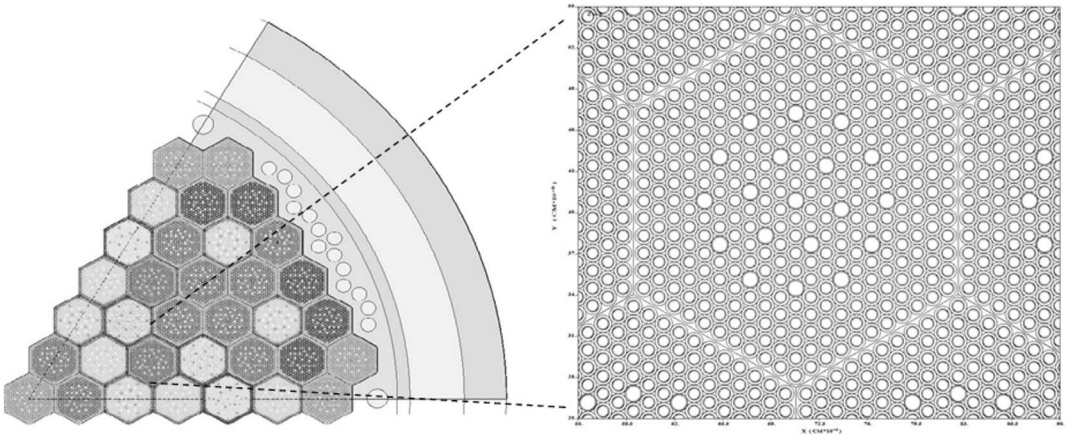

Рис. 7. Поперечное сечение 1/6 картограммы АЗ реактора ВВЭР-1 000

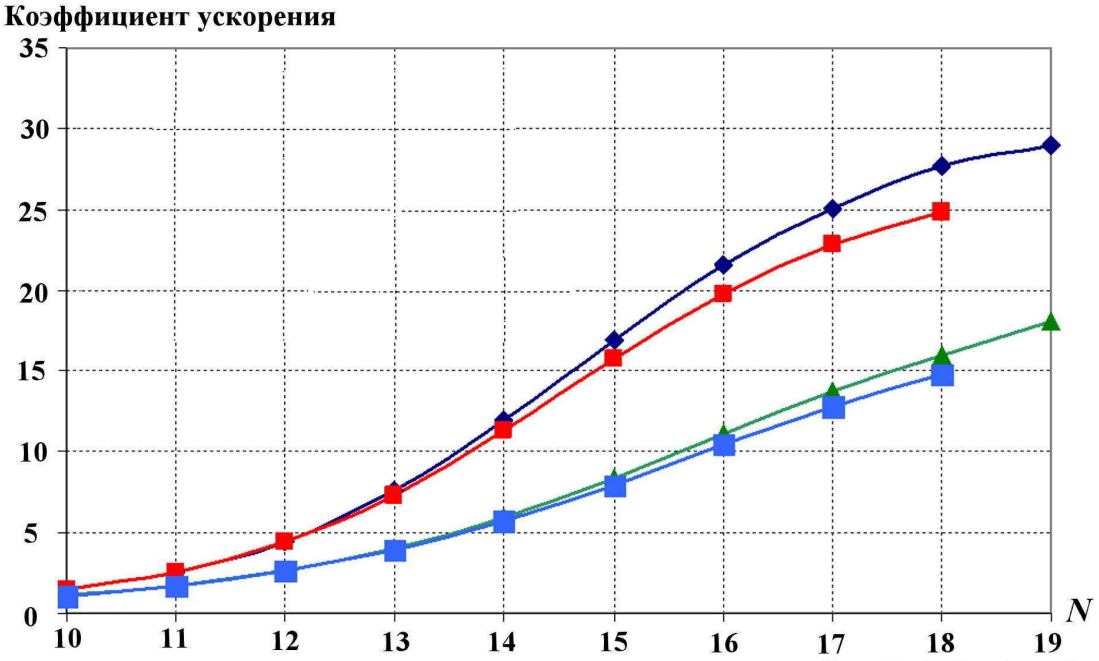

Рис. 8. Коэффициент ускорения при расчете АЗ реактора ВВЭР-1 000 по программе СМК-У: —♦—, —■— — без учета теплового движения ядер среды; —▲—, —■— — с учетом теплового движения ядер среды в приближении свободного максвелловского газа; —♦—, —▲— — только интегральные результаты; —■—, —■— — поверхностные и областные результаты (2^N — размер пакета)

задач коэффициент ускорения возрастает до 30 раз.

Тест 2. Расчет транспортно-упаковочного комплекта металлобетонного контейнера. В этом тесте бралась геометрия транспортно-упаковочного комплекта металлобетонного контейнера (ТУК-МБК, рис. 9) из работы [17], предназначенного для транспортировки отработанного или свежего ядерного топлива АЭС. Как и предыдущий, данный тест характеризуется очень подробным описанием геометрии. Однако здесь химический состав ТВС был гомогенно замешан, что немного упрощало расчет, хотя количество значений, которые необходимо было рассчитать, составляло несколько тысяч.

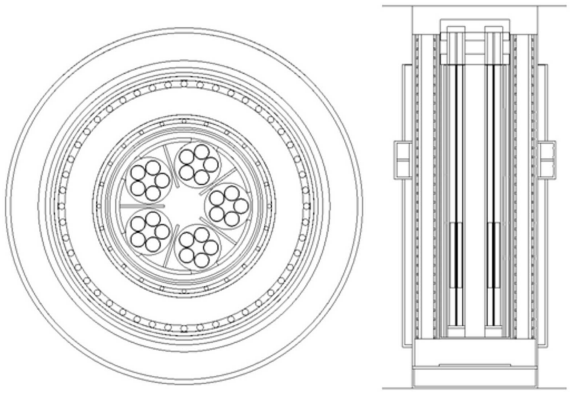

Рис. 9. Фронтальное (слева) и поперечное (справа) сечения ТУК-МБК

Расчет задач теста 2 проводился с такими же вариациями параметров, как и для теста 1.

В первой серии задач рассчитывался выход нейтронов из контейнера. Источником являлись кассеты контейнера. Зависимость ускорения от размера счетного пакета в этой серии расчетов представлена на рис. 10.

Коэффициенты ускорения в данном расчете ниже полученных в тесте 1. Это объясняется различиями в моделировании траекторий нейтронов в задачах на критические параметры и на линейный перенос. При расчете критических параметров нейтроны моделируются по поколениям — от точки деления до точки деления, а при расчете переноса — от источника и до гибели, в результате чего траектории получаются более длинными и разветвленными.

Во второй серии задач рассчитывался выход гамма-квантов из контейнера. Зависимость ускорения от размера счетного пакета для этих задач показана на рис. 11.

Видно, что расчет переноса гамма-квантов выполняется на АрУ примерно с такой же эффективностью, что и расчет переноса нейтронов.

Заключение

Описанный в работе новый алгоритм моделирования траекторий при использовании АрУ существенно сокращает время расчета: от 8 до 30 раз при решении задач с подробной геометрией, сложными математическими моделями взаимодействия частиц с веществом и т. п. Для повышения эффективности использования АрУ были проведены исследования по общей организации хранения данных и алгоритмам расчета различных функционалов от решения с учетом новой архитектуры. Предложенный алгоритм реализован в программном комплексе СМК-У, который предназначен для решения спектрального уравнения переноса нейтронов и гамма-квантов в произвольной трехмерной геометрии с возможностью получения различных результатов расчета и использования разных методов повышения эффективности счета (тактики).

Быстродействие программного комплекса СМК-У сравнивалось с быстродействием программы СМК, предназначенной для выполнения на универсальных процессорах. Оценки коэффициентов ускорений рассчитывались на ряде задач атомной энергетики с использованием одного АрУ и одного ядра универсального процессора. Результаты тестирования показали, что при расчете критических параметров АЗ реакторов было достигнуто максимальное ускорение в 20—25 раз в зависимости от типа реактора, а при решении задач линейного переноса нейтронов или гамма-квантов — до 8 раз. Разница результатов обусловлена различиями в моделировании пакета частиц.

Список литературы

1. *Кандиев Я. З., Серова Е. В.* Меченные частицы в расчетах переноса излучения методом Монте-Карло по программе ПРИЗМА // Атомная энергия. 2005. Т. 98. Вып. 3. С. 386—393.
2. *Gomin E., Kalugin M., Oleynik D.* VVER-1000 MOX Core Computational Benchmark. Specification and Results. France: Nuclear Energy Agency, 2006.
3. MCNP — A General Monte Carlo N-Particle Transport Code. Version 4a / Ed. by J. F. Briesmeister. Report LA-12625-M. Los Alamos National Laboratory, 1993.
4. Monte Carlo Simulation of X-ray Transport in GPU with CUDA. http://code.google.com/p/mcgpu/.
5. *Денисенко М. В., Сатанин А. М.* Применение гетерогенных вычислительных систем и технологии CUDA для моделирования физических процессов // Нижний Новгород: ННГУ, 2012.

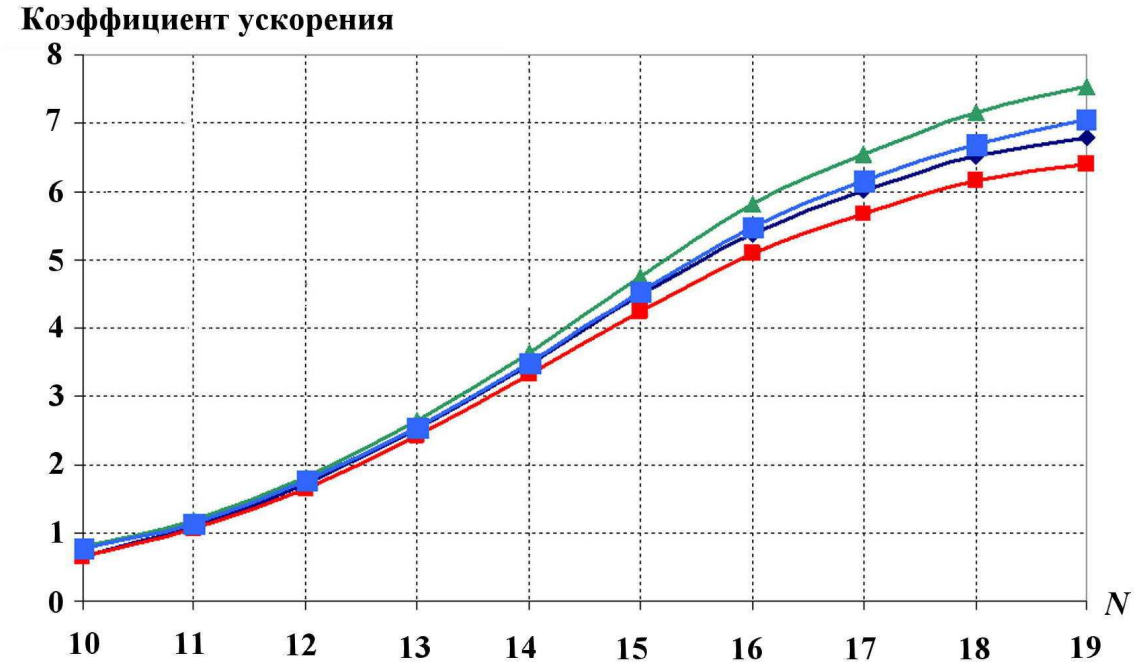

Рис. 10. Коэффициенты ускорения при расчетах ТУК-МБК с нейтронами по программе СМК-У: —♦—, —■— — без учета теплового движения ядер среды; —▲—, —■— — с учетом теплового движения ядер среды в приближении свободного максвелловского газа; —♦—, —▲— — только интегральные результаты; —■—, —■— — поверхностные и областные результаты (2^N — размер пакета)

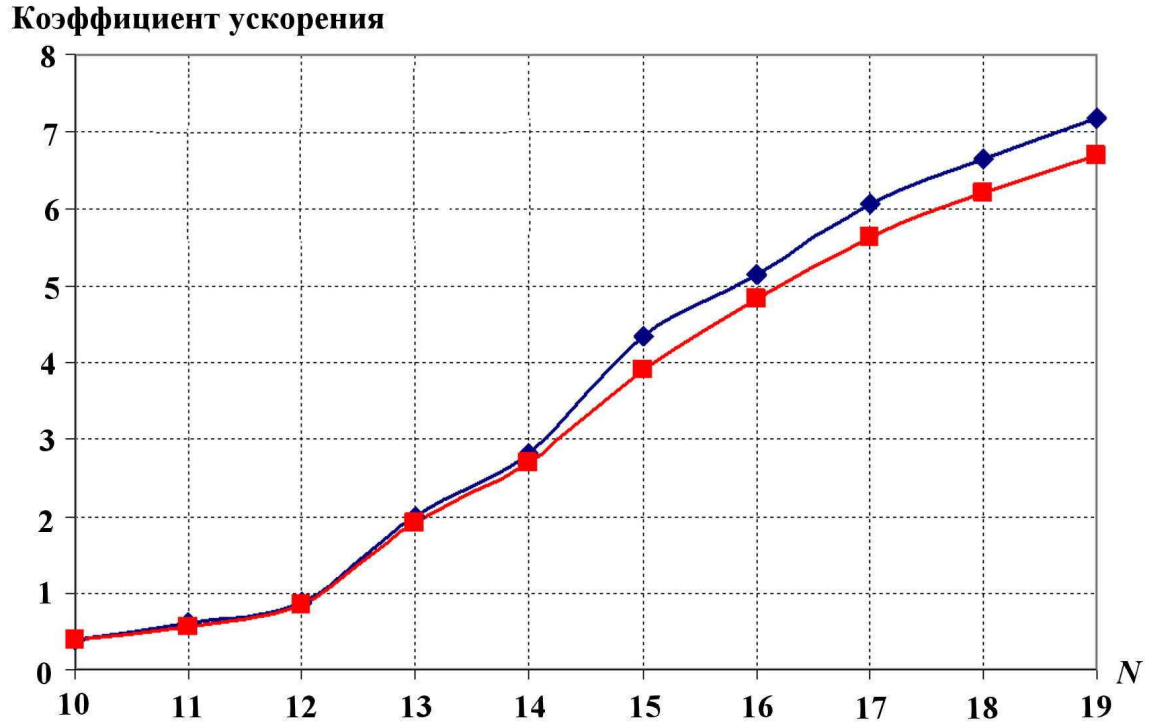

Рис. 11. Коэффициенты ускорения при расчетах ТУК-МБК с гамма-квантами по программе СМК-У: —♦— — только интегральные результаты; —■— — поверхностные и областные результаты (2^N — размер пакета)

6. *Старухин П. Ю., Клинаев Ю. В., Терин Д. В.* Эффективность использования параллельных вычислений по технологии CUDA в задачах моделирования методами Монте-Карло // Ежегод. межд. конф.-выставка "Информационные технологии в образовании (ИТО-2010)". Москва, 6—7 ноября 2010 г.
7. *Жуковский М. Е., Усков Р. В.* О применении графических процессоров видеоускорителей в прикладных задачах. Часть II. Моделирование поглощения гамма-излучения: Препринт № 20. М.: ИПМ им. М. В. Келдыша РАН, 2010.
8. *Жуковский М. Е., Подоляко С. В., Усков Р. В.* Моделирование переноса электронов в веществе на гибридных вычислительных системах // Вычислительные методы и программирование. 2011. Т. 12. С. 152—159.
9. *Жуковский М. Е., Скачков М. В.* О статистических методах моделирования переноса электронов в веществе // Вестник МГТУ им. Н. Э. Баумана. 2009. № 1. С. 31—46.
10. *Alerstam E., Svensson T., Andersson-Engels S.* Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration // J. Biomed. Optical. 2008. Vol. 13(6). P. 060504—060510.
11. *Fang Q., Boas D. A.* Monte Carlo simulation of photon migration in 3D turbit media accelerated by graphic processing units // Opt. Express. 2009. Vol. 17(22). P. 20178—20190.
12. *Nunu R., Jimin L., Xiaochao Q. et al.* GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues // Opt. Express. 2010. Vol 18(7). P. 6811—6823.
13. *Рыбкин А. С., Залялов А. Н., Малькин А. Г. и др.* Программный комплекс на базе гибридных вычислительных систем для расчета критических параметров методом Монте-Карло // XII Межд. семинар "Супервычисления и математическое моделирование". Саров, 11—15 октября 2010 г.
14. *Рыбкин А. С., Залялов А. Н., Малькин А. Г. и др.* Программный комплекс на базе гибридных вычислительных систем для расчета критических параметров методом Монте-Карло // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2012. Вып. 1. С. 52—58.
15. *Кочубей Ю. К., Житник А. К., Артемьева Е. В. и др.* Программа С-95. Моделирование совместного переноса нейтронов и гамма-квантов методом Монте-Карло // Там же. 2000. Вып. 2. С. 49—52.
16. CUDA Zone. http://nvidia.com/object/cuda_home.html
17. *Гуськов В. Д.* Опыт создания российских двухцелевых упаковочных комплектов на основе металлобетонных контейнеров // Тр. III Ежегод. межд. конф. "Радиационная безопасность". С.-Пб., 2008.

Статья поступила в редакцию 08.10.13.