УДК: 061.68 DOI: 10.53816/23061456_2022_9–10_151

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ ПОВЫШЕНИЯ ОПЕРАТИВНОСТИ ВЫПОЛНЕНИЯ ПОСЛЕДОВАТЕЛЬНО-ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ CUDA

STUDY OF INCREASING THE EFFICIENCY OF SERIAL-PARALLEL ALGORITHMS EXECUTION POSSIBILITIES BASED ON THE USE OF CUDA TECHNOLOGY

К.А. Неретина, д-р техн. наук В.А. Лохвицкий, канд. техн. наук А.И. Захаров, канд. техн. наук Г.А. Брякалов

K.A. Neretina, D.Sc. V.A. Lokhvickii, Ph.D. A.I. Zakharov, Ph.D. G.A. Bryakalov

Военно-космическая академия им. А.Ф. Можайского

Статья посвящена проблеме повышения оперативности обработки информации в многопроцессорных вычислительных системах. Необходимость такой работы определяется возросшей потребностью в свободном доступе к параллельной обработке информации, ввиду её больших объемов и ограниченности временных ресурсов, а также возможностью практического использования средств параллельного программирования. В последнее время получила распространение технология, позволяющая производить вычисления с использованием графических процессоров, поддерживающих произвольные вычисления на видеокартах, однако свободное их применение все еще затруднительно ввиду множества трудностей, возникающих в процессе распараллеливания сложных вычислительных задач.
Ключевые слова: обработка информации, многопроцессорные системы, параллельное программирование, графические процессоры.

The article is devoted to the problem of increasing the efficiency of information processing in multiprocessor computing systems. The need for such work is determined by the increased need for free access to parallel processing of information, due to its large volumes and limited time resources, as well as the possibility of practical use of parallel programming tools. Recently, a technology has become widespread that allows computing using graphics processors that support arbitrary computing on video cards, but their free use is still difficult, due to the many difficulties that arise in the parallelization process.
Keywords: information processing, multiprocessor systems, parallel programming, graphic processors.

Введение

До недавнего времени рост производительности обеспечивался повышением тактовой частоты основных вычислительных элементов компьютеров — процессоров. Однако дальнейшее увеличение тактовой частоты требует значительных технологических усилий, сопровождается существенным ростом энергопотребления и наталкивается на непреодолимые проблемы терморегуляции. В таких условиях потребовалось кардинальное изменение основного принципа производства компьютерной техники: вместо создания сложных высокочастотных «монолит-

ных» процессоров стали разрабатывать «составные» процессоры, состоящие из множества равноправных и сравнительно простых вычислительных элементов — ядер. Максимальная производительность процессоров в этом случае равна сумме производительности вычислительных ядер, входящих в процессоры. Таким образом, «собирая» в рамках процессоров большое количество ядер, можно добиться роста производительности без «проблематичного» повышения тактовой частоты [2].

В настоящее время всё чаще появляется необходимость перехода к использованию той или иной формы параллельных вычислений. В том числе переход к многоядерным системам также требует одновременного перехода к параллельным вычислениям, поскольку задействовать вычислительный потенциал многоядерных процессоров можно только в том случае, если разделить выполняемые вычисления на информационно-независимые блоки и организовать их выполнение параллельно на разных ядрах. Подобный подход позволяет выполнять необходимые вычисления с меньшими затратами времени, а также позволяет получать максимальное ускорение, которое ограничивается только числом имеющихся ядер и количеством «независимых» блоков в выполняемых вычислениях.

Однако при разработке параллельной программы для многопроцессорной системы мало разбить программу на параллельные ветви. Для эффективного использования ресурсов необходимо обеспечить равномерную загрузку всех процессоров, что в свою очередь означает, что все ветви программы должны выполнять примерно одинаковый объем вычислительной работы.

Дополнительной формой обеспечения параллелизма может служить конвейерная реализация обрабатывающих устройств, при которой выполнение операций в устройствах представляется в виде исполнения последовательности составляющих операцию подкоманд [2]. Кроме того, большую популярность набирает использование графических процессоров для распараллеливания на основе использования возможностей программно-аппаратной архитектуры технологий CUDA (англ. Compute Unified Device Architecture).

Особенности использования графических адаптеров для решения трудоемких вычислительных задач

В настоящее время функция преобразования видеокартой графической информации в форму, предназначенную для вывода на экран монитора, утратила своё первоначальное значение, и под графическим адаптером, в первую очередь, понимают устройство с графическим процессором, которое занимается формированием самого графического образа. Современные видеокарты не ограничиваются простым выводом изображения, а встроенный графический процессор может производить дополнительную обработку, снимая эту задачу с центрального процессора компьютера.

Следует отметить, что вычислительная мощность современных графических ускорителей, определяемая в первую очередь GPU, в сотни раз превышает аналогичные показатели центральных процессоров при примерно равной стоимости, однако долгое время эти мощности были доступны лишь для разработки графических приложений и не могли использоваться в иных целях, поэтому ведущими производителями видеокарт предпринимались многочисленные попытки приспособить их для решения трудоемких прикладных задач, в том числе неграфических [7]. К настоящему времени в корпорациях — лидерах по производству видеоускорителей (NVIDIA и ATI) — разработаны технологии, позволяющие использовать мощности видеоускорителей для проведения расчетов в прикладных задачах, одной из которых является CUDA [6].

CUDA (Compute Unified Device Architecture) — это технология от компании NVIDIA, предназначенная для разработки приложений для массивно-параллельных вычислительных устройств. Архитектура CUDA позволяет разработчику на свое усмотрение организовывать доступ к набору инструкций GPU и управлять его памятью. Платформа CUDA доступна разработчикам программного обеспечения через библиотеки с ускорением CUDA, директивы компилятора, такие как Open ACC, и расширения для стандартных языков программирования.

В отличие от обычных процессоров, имеющих одно устройство контроля исполнения, кэш и несколько арифметико-логических

устройств, графический процессор состоит из набора мультипроцессоров, каждый из которых имеет своё устройство контроля исполнения (Control), область разделяемой памяти (Cache), области регистров и значительно расширенный по сравнению с обычным процессором набор арифметико-логических устройств (ALU). Такая структура позволяет добиться не только параллельного выполнения алгоритма на разных мультипроцессорах, но и параллельной обработки данных в рамках одного мультипроцессора [1].

Показатели эффективности при решении задач с использованием видеокарт в первую очередь будут зависеть от её характеристик. Например, графическая карта NVIDIA TITAN RTX, которая предназначена для исследователей и разработчиков, обеспечивает сравнительно высокую скорость параллельной обработки данных с одновременной передачей данных между GPU и памятью. Однако, чтобы добиться такого успеха одной лишь аппаратной архитектуры GPU недостаточно, в данном случае необходимо ещё и программное обеспечение, которое позволит раскрыть все возможности GPU, а также грамотный подбор задач, решение которых можно представить в виде большого числа независимых информационных блоков. Таким образом, для реализации возможностей современных видеокарт необходимо соответствующее программное обеспечение, например, технологии CUDA и трудоемкие задачи, алгоритмы решения которых можно распараллелить.

Как было указано ранее [4], алгоритмы условно можно разделить на три группы: последовательные, последовательно-параллельные (ПП) и параллельные. Абсолютно последовательные алгоритмы не распараллеливаются, и их реализация не вызывает затруднений, так как для этого достаточно одного процессора. Для абсолютно параллельных алгоритмов также нет трудностей в реализации, если их число меньше или равно количеству работающих процессоров.

Даже если алгоритмов больше, то и этот вопрос решается положительно за счет масштабирования (подключение дополнительного числа процессоров).

Гораздо сложнее решается вопрос реализации ПП-алгоритмов:

– их необходимо предварительно распараллелить;

– определить число параллельных потоков (ветвей), так как именно они определяют загрузку рабочих процессоров;

– спланировать, запрограммировать и реализовать выполнение параллельной программы.

И даже в этом случае может не хватать количества рабочих процессоров или какая-то их часть будет простаивать. Отсюда и возникает актуальная роль правильного планирования параллельного вычислительного процесса, как максимальная декомпозиция алгоритма на отдельные равные независимые части, так и соответственно максимально полная загрузка рабочих процессоров.

Часть указанных трудностей в организации параллельного вычислительного процесса можно избежать, применяя аппаратно-программные средства распараллеливания многоядерных процессоров (CPU) с использованием вычисления на GPU (графический процессор) для вычисления технических, научных и бытовых задач. Вычисления на GPU совмещают в себе использование CPU и GPU с разнородной выборкой между ними, а именно: последовательную часть программы берет на себя CPU, в то время как трудоёмкие вычислительные задачи решаются GPU [1]. Благодаря особой архитектуре графического процессора происходит распараллеливание задач, которое приводит к ускорению обработки информации и уменьшает время, затрачиваемое на решение поставленных задач, в результате чего система становится более производительной с экономической точки зрения, поскольку может одновременно обрабатывать большое количество задач, используя при этом минимальные временные и технологические ресурсы.

В качестве наглядного примера совместного использования ресурсов центрального процессора (CPU) типового современного многопроцессорного компьютера с возможностями графического процессора (GPU) сформулируем математическую постановку одной из важнейших задач космического назначения — задачи прогноза параметров движения космического аппарата (КА) в пределах охраняемой зоны космического пространства Земли.

Математическая постановка задачи

Дано:

1. Дифференциальные уравнения движения КА в гринвичской системе координат (ГСК) —

система дифференциальных уравнений шестого порядка для неизвестных функций $X = x(t),\ Y = y(t),\ Z = z(t)$ в виде [5]:

$$\begin{cases} \ddot{X} = 2\omega\dot{Y} - \dfrac{\mu X}{r^3} + \omega^2 X; \\ \ddot{Y} = 2\omega\dot{X} - \dfrac{\mu Y}{r^3} + \omega^2 Y; \\ \ddot{Z} = -\dfrac{\mu Z}{r^3}, \end{cases}$$

где X, Y, Z — координаты радиус-вектора положения спутника в ГСК;

ω — угловая скорость вращения Земли вокруг своей оси, $\omega = \dfrac{2\pi}{T} \approx 7{,}29 \cdot 10^{-5}\ \text{с}^{-1}$;

μ — геоцентрическая гравитационная постоянная, численное значение которой принимается равным $\mu = 3{,}9860044 \cdot 10^{14}\ \text{м}^3 \cdot \text{с}^{-2}$;

r — расстояние от Земли до КА (высота орбиты).

2. Начальные условия на момент $t = 0$:

X_0, Y_0, Z_0 – координаты радиус-вектора положения КА в ГСК в момент времени $t = 0$;

$V_{x_0}, V_{y_0}, V_{z_0}$ — проекции вектора скорости спутника на оси ГСК в момент времени $t = 0$.

3. Расчетные формулы метода Рунге-Кутты 4-го порядка:

$$\vec{y}(t+h) = \vec{y}(t) + \frac{\left(\vec{k_1} + 2\vec{k_2} + 2\vec{k_3} + \vec{k_4}\right)}{6} + \vec{O}\left(h^5\right);$$

$$\vec{k_1} = hA\vec{y}(t) + h\vec{f}(t);$$

$$\vec{k_2} = hA\left(\vec{y}(t) + \frac{\vec{k_1}}{2}\right) + h\vec{f}\left(t + \frac{h}{2}\right);$$

$$\vec{k_3} = hA\left(\vec{y}(t) + \frac{\vec{k_2}}{2}\right) + h\vec{f}\left(t + \frac{h}{2}\right);$$

$$\vec{k_4} = hA\left(\vec{y}(t) + \vec{k_3}\right) + h\vec{f}(t+h).$$

4. Условия, ограничения и пояснения:

– в уравнениях движения не учитывается влияние Луны и Солнца на проекции составляющих ускорений;

– в уравнениях движения не учитывается влияние аномалий поля тяжести Земли на проекции составляющих ускорений;

– координаты и проекции скорости КА заданы в прямоугольной гринвичской системе координат.

Требуется:

– найти прогноз параметров движения КА по заданным начальным условиям;

– реализовать последовательную и параллельную версии программы для решения поставленной задачи, сопоставить времена выполнения обоих вариантов.

Результаты решения задачи

Задача была решена в несколько этапов.

Этап 1. Математическое решение задачи методом Рунге-Кутты 4-го порядка:

$$\begin{cases} \dot{X} = V_x; \\ \dot{Y} = V_y; \\ \dot{Z} = V_z; \\ \dot{V}_x = 2\omega V_y + \left(\omega^2 - \dfrac{\mu}{r^3}\right) X; \\ \dot{V}_y = 2\omega V_x + \left(\omega^2 - \dfrac{\mu}{r^3}\right) Y; \\ \dot{V}_z = -\dfrac{\mu Z}{r^3}. \end{cases}$$

Шаг 1:

$$\begin{cases} a_{11} = V_{x_0}; \\ b_{11} = V_{y_0}; \\ c_{11} = V_{z_0}; \\ d_{11} = 2\omega V_{y_0} + \left(\omega^2 - \dfrac{\mu}{r^3}\right) X_0; \\ e_{11} = 2\omega V_{x_0} + \left(\omega^2 - \dfrac{\mu}{r^3}\right) Y_0; \\ f_{11} = -\dfrac{\mu Z_0}{r^3}. \end{cases}$$

$$\begin{cases} a_{12} = V_{x_0} + \dfrac{d_{11}}{2}h; \\ b_{12} = V_{y_0} + \dfrac{e_{11}}{2}h; \\ c_{12} = V_{z_0} + \dfrac{f_{11}}{2}h; \\ d_{12} = 2\omega\left(V_{y_0} + \dfrac{e_{11}}{2}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(X_0 + \dfrac{a_{11}}{2}h\right); \\ e_{12} = 2\omega\left(V_{x_0} + \dfrac{d_{11}}{2}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(Y_0 + \dfrac{b_{11}}{2}h\right); \\ f_{12} = -\dfrac{\mu}{r^3}\left(Z_0 + \dfrac{c_{11}}{2}h\right). \end{cases}$$

$$\begin{cases} a_{13} = V_{x_0} + \dfrac{d_{12}}{2}h; \\ b_{13} = V_{y_0} + \dfrac{e_{12}}{2}h; \\ c_{13} = V_{z_0} + \dfrac{f_{12}}{2}h; \\ d_{13} = 2\omega\left(V_{y_0} + \dfrac{e_{12}}{2}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(X_0 + \dfrac{a_{12}}{2}h\right); \\ e_{13} = 2\omega\left(V_{x_0} + \dfrac{d_{12}}{2}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(Y_0 + \dfrac{b_{12}}{2}h\right); \\ f_{13} = -\dfrac{\mu}{r^3}\left(Z_0 + \dfrac{c_{12}}{2}h\right). \end{cases}$$

$$\begin{cases} a_{14} = V_{x_0} + d_{13}h; \\ b_{14} = V_{y_0} + e_{13}h; \\ c_{14} = V_{z_0} + f_{13}h; \\ d_{14} = 2\omega\left(V_{y_0} + e_{13}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(X_0 + a_{13}h\right); \\ e_{14} = 2\omega\left(V_{x_0} + d_{13}h\right) + \left(\omega^2 - \dfrac{\mu}{r^3}\right)\left(Y_0 + b_{13}h\right); \\ f_{14} = -\dfrac{\mu}{r^3}\left(Z_0 + c_{13}h\right). \end{cases}$$

$$\begin{cases} X_1 = X_0 + \dfrac{h}{6}\left(a_{11} + 2a_{12} + 2a_{13} + a_{14}\right); \\ Y_1 = Y_0 + \dfrac{h}{6}\left(b_{11} + 2b_{12} + 2b_{13} + b_{14}\right); \\ Z_1 = Z_0 + \dfrac{h}{6}\left(c_{11} + 2c_{12} + 2c_{13} + c_{14}\right); \\ V_{x_1} = V_{x_0} + \dfrac{h}{6}\left(d_{11} + 2d_{12} + 2d_{13} + d_{14}\right); \\ V_{y_1} = V_{y_0} + \dfrac{h}{6}\left(e_{11} + 2e_{12} + 2e_{13} + e_{14}\right); \\ V_{z_1} = V_{z_0} + \dfrac{h}{6}\left(f_{11} + 2f_{12} + 2f_{13} + f_{14}\right). \end{cases}$$

Каждый следующий шаг выполняется аналогично Шагу 1.

Этап 2. Введение параллелизма в алгоритм решения.

Если каждый коэффициент $a_{ij}, b_{ij}, c_{ij}, d_{ij}, e_{ij}, f_{ij}$, где $i = \overline{1,n}$, $j = \overline{1,4}$ расписать не через предыдущий, а через имеющиеся начальные условия и константы, то на каждом шаге возможно находить 24 коэффициента параллельно друг другу, после чего производить сборку для нахождения $X_i, Y_i, Z_i, V_{x_i}, V_{y_i}, V_{z_i}$, где $i = \overline{1,n}$.

Этап 3. Программная реализация.

Последовательное решение задачи реализовано на языке Python в среде JetBrains PyCharm 2018. Параллельный код выполнен с использованием технологий CUDA для графического процессора от NVIDIA.

Этап 4. Сравнение результатов выполнения программ.

Результатами решения поставленной задачи выступали параметры движения КА: проекции координат и вектора скорости на оси ГСК, скорость движения КА на момент времени $T = h \cdot n$, где h — шаг по времени, а n — номер текущего шага (итерации), а также время нахождения заданных параметров в пределах текущего шага (итерации).

В рамках статьи особое внимание стоит обратить на время, затрачиваемое на вычисление необходимых параметров последовательной и параллельной версиями программы. Результаты исследования приведены на рисунке.

Каждая итерация параллельной программы выполняется в среднем в два раза быстрее последовательной. Скачки графика последовательной реализации обусловлены разной загрузкой центрального процессора другими процессами. Стоит отметить, что немало ресурсов уходит на переключение

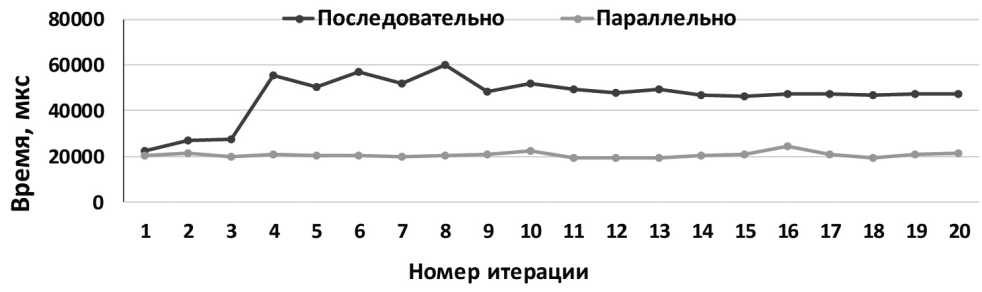

Рис. Зависимость времени нахождения параметров движения КА от номера итерации

с центрального процессора на графический, что может значительно уменьшить общий выигрыш во времени относительно последовательной реализации программы, поэтому выгоднее всего передавать управление вычислениями графическому процессору при выполнении одной и той же команды ко множеству данных (SIMD по классификации Флинна) [3].

Заключение

Таким образом, в статье рассмотрен способ повышения оперативности обработки информации, заключающийся в разбиении алгоритма решения задачи на информационно-независимые блоки, а также в применении аппаратно-программной технологии CUDA к их параллельной обработке, что подтверждает перспективные возможности вычислений с использованием графического процессора.

В частности, в рассмотренном примере совместное использование графического и центрального процессоров позволило повысить оперативность определения параметров движения космического аппарата в среднем в два раза. Стоит отметить, что наибольший выигрыш в оперативности достижим при решении задач, относящихся к классу SIMD по классификации Флинна, поскольку это поможет минимизировать временные затраты на переключение от центрального процессора к графическому и обращение к ядрам графического процессора.

Литература

1. Боресков А.В., Харламов А.А. Основы работы с технологией CUDA. — М.: ДМК-Пресс, 2010. 232 с.

2. Гергель В.П. Высокопроизводительные вычисления для многоядерных многопроцессорных систем: учебное пособие. — Нижний Новгород: ННГУ, 2010. 421 с.

3. Гергель В.П. Теория и практика параллельных вычислений. — М.: Интернет-Университет, БИНОМ. Лаборатория знаний, 2007. 464 с.

4. Захаров А.И., Брякалов Г.А., Неретина К.А. Влияние параллельных вычислений и структуры алгоритмов решаемых задач на оперативность обработки информации в многопроцессорных вычислительных системах // Вестник Российского нового университета. Сложные системы: модели, анализ и управление. — М.: РосНОУ, 2021. С. 143–149.

5. Крылов В.И. Основы теории движения ИСЗ (часть первая: невозмущённое движение): учебное пособие. — М.: МИИГАиК, 2015. 52 с.

6. Malik M.Z. CUDA 4.1 Programming Cookbook. 1st edition, Packt Publishing, 2016.

7. Сандерс Д., Кэндрот Э. Технология CUDA в примерах. Введение в программирование графических процессов. — М.: ДМКПресс, 2015. 232 с.

References

1. Boreskov A.V., Kharlamov A.A. Bases of work with CUDA technology. — M.: DMK-Press, 2010. 232 p.

2. Gergel V.P. High-performance calculations for multicore multiprocessor systems. The manual is Nizhny Novgorod: UNN, 2010. 421 p.

3. Gergel V.P. Theory and practice of parallel calculations. — M.: Internet University, BING. Laboratory of knowledge, 2007. 464 p.

4. Zakharov A.I., BryakalovG.A., Neretin K.A. Influence of parallel calculations and structure of algorithms of solvable tasks on efficiency of information processing in multiprocessor computing systems // The Messenger of the Russian new university. Complex systems: models, analysis and management. — M.: RosNOU, 2021. Pp. 143–149.

5. Krylov V.I. Bases of the theory of the movement artificial satellite (part one: not indignant movement). Manual. — M.: MIIGAiK, 2015. 52 p.

6. Malik M.Z. CUDA 4.1 Programming Cookbook. 1st edition, Packt Publishing, 2016.

7. Sanders D., Kendrot E. CUDA technology in examples. Introduction to programming of graphic processes. — M.: DMKPress, 2015. 232 p.