ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 4(18), 2013, с.17—31

УДК 519.682.3

В. А. Роганов, А. А. Кузнецов, Г. А. Матвеев, В. И. Осипов

Методы адаптации системы параллельного программирования OpenTS для поддержки работы T-приложений на гибридных вычислительных кластерах

Аннотация. В статье описаны методы адаптации системы параллельного программирования OpenTS для обеспечения работы параллельных T++-приложений на гибридных кластерах, узлы которых содержат как классические процессоры, так и графические ускорители (например, NVIDIA GPGPU).

Ключевые слова и фразы: динамическое распараллеливание, T-система с открытой архитектурой, OpenTS, язык программирования T++, графические ускорители, гибридные кластерные системы.

Введение

T-система [1] является оригинальной российской разработкой, объединяющей в себе наиболее удачные черты функционального программирования, dataflow-систем и традиционных языков и методов программирования. T-система базируется на функциональной парадигме и предполагает определенные ограничения на стиль программирования в случае ее использования. Взамен она предоставляет бесконфликтную модель динамического распараллеливания, в которой невозможны взаимные блокировки и некорректный доступ к разделяемым переменным.

T-система с открытой архитектурой (OpenTS) [2] была разработана в ИПС им. А.К. Айламазяна РАН в рамках суперкомпьютерной программы «СКИФ» Союзного государства. Она представляет собой современную реализацию идей T-системы и обеспечивает лучшую, чем предыдущие версии системы, интеграцию ба-

зовых возможностей функционального подхода с возможностями языка программирования C++. OpenTS [3] обладает открытой и масштабируемой архитектурой, легко адаптируемой к стремительно меняющимся аппаратным платформам современных суперкомпьютеров. Поддерживаемый системой OpenTS входной язык программирования T++ является синтаксически и семантически гладким расширением языка программирования C++, а среда исполнения T-приложений представляет собой ортогональную надстройку (T-суперструктуру) над стандартной последовательной средой программирования.

Подход к автоматическому динамическому распараллеливанию программ, предложенный в T-системе, позволяет получить хорошие результаты по утилизации вычислительной мощности современных кластерных установок, что связано с природой используемой модели вычислений. Система OpenTS [4],[5] ассимилирует многие другие технологии параллельного программирования: специальную модель общей памяти, модель распределенных потоков и объектов, распределенную сборку мусора, и, наконец, высокоуровневую языковую надстройку, являющуюся уникальной по своим характеристикам технологией для поддержки максимальной совместимости с традиционными языками по синтаксису и семантике, но при этом эффективно распараллеливаемой моделью вычислений.

CUDA — вычислительная архитектура, разработанная компанией NVIDIA и предназначенная для разработки параллельных программ, выполнимых непосредственно на графических ускорителях. В сочетании с развитой программной платформой архитектура CUDA позволяет программисту задействовать возможности современных графических процессоров для создания высокопроизводительных приложений — производительность нескольких современных графических ускорителей в составе одного вычислительного узла может составлять несколько терафлопс.

В настоящее время архитектура CUDA стремительно эволюционирует как в части повышения вычислительной мощности (ко-

личество одновременно работающих вычислительных ядер), так и в части повышения уровня программного кода, который поддерживается базовым компилятором «nvcc». В связи с этим, становится возможной тесная интеграция системы параллельного программирования OpenTS и программно-аппаратной платформы CUDA.

В данной статье описываются подходы к реализации принципа автономного динамического распараллеливания функциональных программ на GPU-устройствах. Реализация данных подходов позволит эффективно задействовать мощь современных графических ускорителей и использовать гибридные кластеры с архитектурой CPU+GPU для эффективного исполнения хорошо масштабируемых T-программ. Это позволит эффективно сочетать стандартные возможности CUDA с динамическим распараллеливанием на большое количество вычислительных ядер, что крайне важно на современном этапе эволюции суперкомпьютерной индустрии.

1. Особенности архитектуры NVIDIA CUDA

Задача поддержки автономного динамического распараллеливания программ внутри GPU принципиально отличается от простого вызова вычислительных ядер CUDA с уровня языка T++. Под автономностью здесь и далее понимается режим, когда часть программы, выполняемая потоковыми мультипроцессорами CUDA, работает на ускорителе автономно, то есть без какого либо участия центрального процессора вычислительного узла, инициирующего решение задачи на GPU.

Трудности реализации автономного динамического распараллеливания программ внутри GPU состоят в том, что графические ускорители изначально не были ориентированы на динамическое распараллеливание, то есть на режимы, когда вычислительная работа появляется непосредственно в процессе вычислений. Это связано со спецификой области применения графических устройств, и еще пару лет назад реализация подобной возможности выглядела бы малоперспективной.

С недавних пор, благодаря проникновению на рынок высокопроизводительных вычислений, а также конкуренции с ведущими

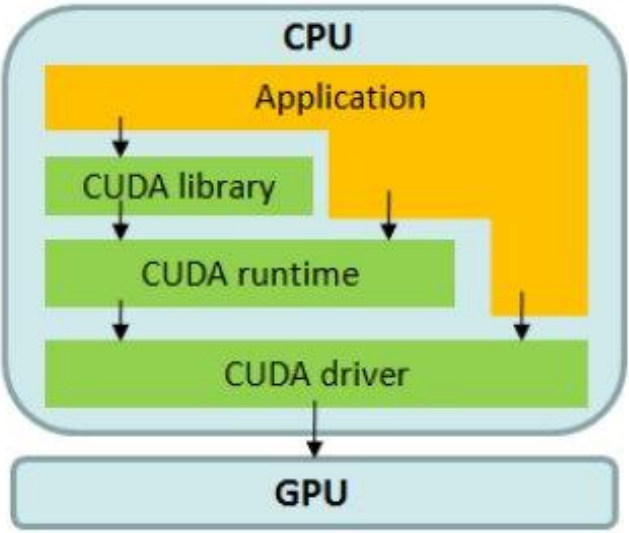

Рис. 1. Архитектура NVIDIA CUDA

производителями оборудования для супер-ЭВМ, фирма NVIDIA начала активно расширять функциональные возможности своих графических ускорителей (каждое поколение аппаратуры имеет свой индекс поддержки возможностей — Compute Capability).

Реализованные в программно-аппаратной архитектуре CUDA способы работы с динамической памятью и поддержка виртуальных функций в коде вычислительных ядер в сочетании с атомарными операциями уже позволяют реализовать многопоточный механизм обработки, способный автономно порождать и распределять вычислительную работу внутри графического ускорителя. Эти и некоторые другие возможности доступны для устройств начиная с CUDA Compute Capability 2.1 и CUDA SDK 5.0.

1.1. Особенности аппаратной архитектуры NVIDIA Kepler

В отличие от технологий, используемых в архитектуре Fermi, новая архитектура Kepler от NVIDIA разрабатывалась с нуля, с

целью обеспечить конечного пользователя максимальным быстродействием. Видеоадаптеры, построенные на основе этой технологии, созданы для работы с параллельными вычислениями в высокопроизводительных системах.

Характеристики новых CUDA-устройств:

- 2880 вычислительных ядер CUDA;
- 15 блоков SMX;
- 384-разрядный контроллер памяти;
- до 24 гигабайт видео памяти DDR5;
- второе поколение памяти ECC;
- динамический параллелизм (возможность потоков GPU генерировать новые потоки для быстрой адаптации к новым данным);
- функция Hyper-Q (функция позволяет нескольким ядрам CPU одновременно использовать ядра CUDA на одном GPU Kepler);
- общая производительность ожидается на уровне 1.5 TFLOPS DP FP64;
- пропускная способность 250 Gb/s.

Наиболее важными в контексте данного исследования представляются следующие возможности:

1.1.1. Динамический параллелизм

Эта технология позволяет потокам GPU динамически порождать новые потоки, что существенно упрощает параллельное программирование за счет применения GPU-ускорения к широкому спектру распространенных алгоритмов, таких как адаптивное уточнение сеток, быстрые мультипольные и мультисеточные методы.

1.1.2. Hyper-Q

Эта функция позволяет нескольким ядрам CPU одновременно использовать ядра CUDA на одном GPU-устройстве архитектуры «Kepler». Нагрузка на GPU значительно возрастает, уменьшается время простоя CPU и улучшается программируемость. Hyper-Q —

это подходящее решение для кластерных задач, использующих MPI.

1.1.3. SMX (мультипроцессор нового поколения)

Будучи основным строительным материалом каждого GPU, потоковый мультипроцессор SMX был создан с нуля для высокой производительности и экономичности. Он обеспечивает производительность на Ватт до 3-х раз выше по сравнению с потоковым мультипроцессором Fermi.

Экономичность SMX была достигнута за счет увеличения вчетверо числа CUDA-ядер при сокращении частоты каждого ядра, отключении питания вычислительных ядер GPU, находящихся в простое, и увеличении площади GPU, предназначенной для параллельных расчетов, за счет уменьшения модулей управляющей логики.

2. Методы адаптации системы параллельного программирования OpenTS для поддержки работы T-приложений на вычислительных кластерах с гибридными узлами на базе ускорителей NVIDIA GPGPU

2.1. Основные трудности поддержки автономного динамического распараллеливания на текущих версиях GPU

Основные трудности, которые стоят на пути поддержки динамического распараллеливания в стиле T-системы внутри CUDA-устройств, состоят в следующем:

(1) Слабая поддержка в платформе CUDA механизмов синхронизации и обмена данными между отдельными потоковыми мультипроцессорами GPU-устройства. Для повышения производительности рекомендуется разбивать задачу на совершенно независимые блоки, что, безусловно, очень выгодно во многих случаях при статическом распараллеливании. При динамическом же распараллеливании вычислительные подзадачи появляются непосредственно во время счета, и между ними дина-

мически возникают зависимости по данным. Кроме того, их необходимо максимально равномерно распределять по всем доступным вычислительным ядрам для выравнивания нагрузки. Формально есть рекомендованный способ корректной синхронизации между разными потоковыми процессорами — завершение вычислительного ядра, то есть с привлечением центрального процессора. Очевидно, такой подход идет вразрез с целями автономного динамического распараллеливания и подходит только для случая выравнивания нагрузки между несколькими вычислительными узлами. Имеется вариант прямой синхронизации через глобальную память, который выбран в качестве основного в данном исследовании. Важно отметить, что он не нарушает стандарта CUDA, то есть должен работать при любом количестве и любой комбинации активных потоковых процессоров.

(2) Возможность сохранять и восстанавливать вычислительный контекст счетных функций. В случае исполнения программ на центральном процессоре есть специальные примитивы, которые позволяют сохранять/возобновлять исполнение отдельных потоков, что необходимо делать при динамическом распараллеливании в стиле T-системы, когда происходит обращение к неготовым переменным. Примерами таких примитивов являются makecontext/swapcontext, а также различные аналоги. Эти возможности хорошо задокументированы и поддерживаются системными библиотеками и компиляторами. В случае GPU планирование и управление потоком вычислений берет на себя сама программно-аппаратная среда CUDA, которая скрывает явные механизмы управления потоками от программиста. Для использования переключения контекста приходится задействовать низкоуровневые возможности среды CUDA, включая уровень ассемблера «PTX» и аппаратуры, которые задокументированы и стандартизованы хуже, чем входной язык компилятора nvcc. В последнее время компания NVIDIA активно развивает направление Dynamic Parallelism для CUDA-устройств с индексами Compute Capability 3.5 и выше, что может повлечь появление новых возможностей, так или иначе связанных с переключением контекста. Еще один способ переключения контекста состоит в преобразовании исходного кода T-функции таким образом, чтобы она в приостановленном состоянии могла превращаться в объект, который можно обратно превратить в выполняющуюся функцию. Несмотря на некоторые накладные

расходы, связанные с переработкой T++-компилятора, такой способ весьма универсален, и годится практически для любых аппаратных ускорителей. Этот способ выбран в качестве основного для реализации первой версии системы автономного динамического распараллеливания.

2.2. Поддержка динамического параллелизма в ускорителях архитектуры NVIDIA Kepler

Как уже отмечалось ранее, в настоящее время фирма NVIDIA активно развивает направление Dynamic Parallelism, которое добавляет в модель программирования CUDA возможность запуска новых вычислительных ядер внутри устройства, то есть без участия центрального процессора вычислительного узла:

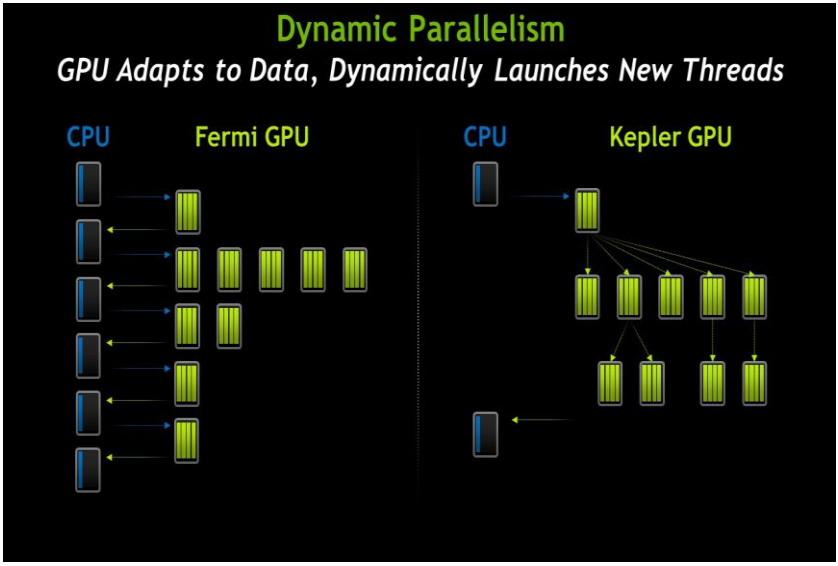

Рис. 2. Динамический параллелизм в NVIDIA GPU

В настоящее время на рынке доступна видеокарта, которая поддерживает эти новые возможности (Compute Capability 3.5),

что дает возможность продолжить исследование и разработку подсистемы динамического распараллеливания при наличии принципиально новых возможностей. Характеристики устройства:

- модель видеокарты: GTX Titan;
- серия видеокарты: GeForce;
- техпроцесс: 28 нм;
- частота видеопроцессора: 837 МГц;
- частота видеопамяти: 6.08 ГГц;
- видеопамять: 6.14 Гб;
- тип видеопамяти: GDDR5;
- разрядность шины: 384 bit;
- размеры: 267x111 мм;
- количество занимаемых слотов: 2.

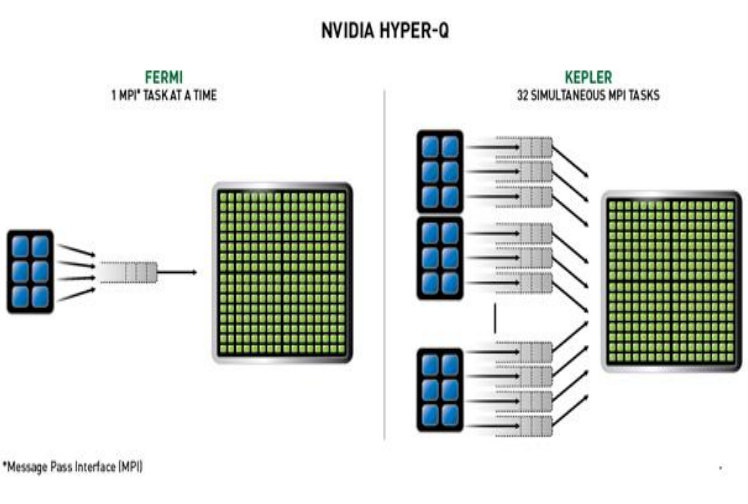

Рис. 3 — Технология NVIDIA Hyper-Q

Среди особенностей CUDA-устройств архитектуры Kepler — асинхронность работы и параллельное взаимодействие с центральным процессором.

Эти возможности могут оказаться полезными при взаимодействии нескольких CUDA-устройств между собой для балансировки

нагрузки во время динамического распараллеливания ресурсоемких T-приложений.

2.3. Технические решения, позволяющие реализовать автономный динамический параллелизм

Технические решения, перечисленные ниже, позиционируются не как окончательные, но как рабочие варианты преодоления тех трудностей, которые стоят на пути реализации автономного динамического распараллеливания программ внутри CUDA-устройств.

(1) Динамическое порождение объектов (неготовых значений и T-функций). Данная возможность доступна в современных CUDA-устройствах при использовании операторов new/delete на стороне GPU-устройства, то есть в CUDA-коде.

(2) Превращение T-функций в объект (для постановки в очередь на счет, остановки/возобновления счета). Данная возможность доступна в современных CUDA-устройствах при использовании классов входного языка CUDA, при этом для методов классов доступны виртуальные функции (допускается переопределение методов).

(3) Использование всех вычислительных ядер CUDA-устройства (то есть всех потоковых мультипроцессоров). Данная возможность реализуется запуском произвольного количества CUDA-потоков (CUDA Streams).

(4) Синхронизация данных между отдельными блоками потоков. Стандарт CUDA не предусматривает (и, более того, не рекомендует) согласованный обмен данными за пределами одного вычислительного блока. Хотя такая возможность имеется, ее рекомендуют избегать, так как обмен данными через глобальную память уменьшает быстродействие. Тем не менее, поскольку в современных графических ускорителях используется быстрая память DDR5, использование атомарных операций совместно с дополнительными опциями ассемблера PTX позволяет организовать корректное взаимодействие между несколькими параллельно работающими блоками потоков.

(5) Взаимное исключение. Для корректной реализации общей очереди подзадач используются атомарные операции над ячейками глобальной памяти.

(6) Переключение контекста Т-функции. На настоящий момент опробована методика преобразования тела Т-функции в метод соответствующего объекта, при этом при некоторых ограничениях на структуру кода Т-функции ее исполнение можно разбить на стадии, выполнение каждой из которых зависит от готовности определенных Т-переменных. Таким образом, представляется возможным после проведения соответствующего DataFlow-анализа Т-конвертером преобразовывать Т-функции, удовлетворяющие некоторым ограничениям на стиль их кода, к виду, когда переключение контекста может быть реализовано путем возврата из них с переключением состояния на следующую стадию при готовности всех Т-переменных, значения которых потребуются в новой стадии.

Заключение

В результате проведенных исследований получены следующие результаты:

- Выработаны методы адаптации системы параллельного программирования OpenTS для поддержки работы Т-приложений на вычислительных кластерах с гибридными узлами на базе ускорителей NVIDIA GPGPU.
- Проведен анализ особенностей архитектуры CUDA с точки зрения возможности динамического распараллеливания в стиле Т-системы. Выявлен ряд узких мест, для которых выработаны универсальные технические решения, которые можно совершенствовать по мере появления новых возможностей динамично развивающейся технологии GPGPU.
- Сформулированы принципы интеграции подсистемы автономного динамического распараллеливания в ядро OpenTS, которые позволят вызывать определенные Т-функции на устройстве с поддержкой CUDA.
- Проведен анализ тенденций в развитии программно-аппаратной платформы NVIDIA CUDA, по результатам которого представляется весьма вероятным, что в ближайшее время будут

доступны графические ускорители со значительно более подходящим набором возможностей для поддержки динамического распараллеливания в стиле Т-системы.

Благодарности. *Работы, положенные в основу данной статьи, были выполнены в рамках проекта «Адаптация системы параллельного программирования OpenTS для поддержки работы T++-приложений на вычислительных кластерах с гибридными узлами на базе FPGA и GPGPU» по Программе фундаментальных научных исследований ОНИТ РАН «Архитектурно-программные решения и обеспечение безопасности суперкомпьютерных информационно-вычислительных комплексов новых поколений», а также в рамках НИР «Методы и программные средства разработки параллельных приложений и обеспечения функционирования вычислительных комплексов и сетей нового поколения».*

Список литературы

[1] Абрамов С.М., Васенин В.А., Мамчиц Е.Е., Роганов В.А., Слепухин А.Ф. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Научная сессия МИФИ-2001. Т.2. Информатика и процессы управления. Информационные технологии. Сетевые технологии. Параллельные вычислительные технологии. Сборник научных трудов. – М., 2001. – С. 34–235.

[2] Абрамов С.М., Кузнецов А.А., Роганов В.А. Кроссплатформенная версия Т-системы с открытой архитектурой // Труды Международной научной конференции «Параллельные вычислительные технологии (ПаВТ'2007)». Т. 1. Челябинск, 29 января–2 февраля 2007 г. – Челябинск: изд. ЮУрГУ. – С. 115 –121

[3] Абрамов С.М., Кузнецов А.А., Роганов В.А.. «Кроссплатформенная версия Т-системы с открытой архитектурой» // Вычислительные методы и программирование, 2007, Т. 8, № 1, Раздел 2. – С. 175– 180, URL: http://num-meth.srcc.msu.su/

[4] Абрамов С. М., Есин Г. И., Загоровский И. М., Матвеев Г. А., Роганов В. А. Принципы организации отказоустойчивых параллельных вычислений для решения вычислительных задач и задач управления в Т-Системе с открытой архитектурой (OpenTS) // Международная конференция «Программные системы: теория и приложения (PSTA-2006)», 23–28 октября 2006 г., г. Переславль-Залесский, Институт Программных Систем РАН. –С. 257–264

[5] Кузнецов А.А., Роганов В.А. Экспериментальная реализация отказоустойчивой версии системы OpenTS для платформы Windows CCS // Труды Второй Международной научной конференции «Суперкомпьютерные системы и их применение (SSA'2008)», 27–29 октября 2008 г., Минск. – Минск: ОИПИ НАН Беларуси, 2008. –С. 65–70, ISBN 978-985-6744-46-7

Рекомендовал к публикации *чл.-корр. РАН С. М. Абрамов*

Об авторах:

Владимир Александрович Роганов
научный сотрудник ИЦМС ИПС им. А.К. Айламазяна РАН. Разработчик современных версий Т-системы, ведущий разработчик системы OpenTS. Принимал активное участие в суперкомпьютерных проектах Союзного государства России и Беларуси, в том числе в проектах «СКИФ» и «СКИФ-ГРИД».
e-mail: mailto:var@pereslavl.ru

Антон Александрович Кузнецов

научный сотрудник ИЦМС ИПС им. А.К.Айламазяна РАН. Один из разработчиков системы OpenTS, ведущий разработчик системы распределенных вычислений SkyTS. Область научных интересов: системы параллельного программирования, распределенные вычисления в гетерогенных средах, геоинформационные системы.

e-mail: mailto:tonic@pereslavl.ru

Герман Анатольевич Матвеев

ведущий инженер-исследователь ИЦМС ИПС им. А.К. Айламазяна РАН. Один из разработчиков системы OpenTS. Принимал участие в суперкомпьютерных проектах Союзного государства России и Беларуси.

*e-mail:*mailto:gera@prime.botik.ru

Валерий Иванович Осипов

к.ф.-м.н., инженер-исследователь ИЦМС ИПС им. А.К. Айламазяна РАН. Один из разработчиков системы OpenTS. Принимал участие в суперкомпьютерных проектах Союзного государства России и Беларуси.

e-mail: mailto:val@pereslavl.ru

Образец ссылки на публикацию:

В. А. Роганов, А. А. Кузнецов, Г. А. Матвеев, В. И. Осипов. Методы адаптации системы параллельного программирования OpenTS для поддержки работы Т-приложений на гибридных вы-

числительных кластерах // Программные системы: теория и приложения: электрон. научн. журн. 2013. Т. 4, № 4(18), с. 17–31.
URL: http://psta.psiras.ru/read/psta2013_4_17-31.pdf

V. A. Roganov, A. A. Kuznetsov, G. A. Matveev, V. I. Osipov. Methods of adaptation of the OpenTS parallel programming system runtime for the hybrid computing clusters.

Abstract. The article describes methods to be used to adapt the OpenTS parallel programming system and runtime for the optimal performance on hybrid clusters with computational nodes having accelerator hardware based on NVIDIA GPGPU.

Key Words and Phrases: dynamic parallelization, T-system with an open architecture, OpenTS, T++ programming language, GPU accelerator, hybrid cluster systems.