УДК 621

Эффективное использование гетерогенных вычислительных систем

Л.П. Чернышева, Д.П. Харитонов
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
Иваново, Российская Федерация
E-mail: chernlu@vvs.ispu.ru

Авторское резюме

Состояние вопроса: В гетерогенных системах используется сложный способ организации параллельной вычислительной системы, в них объединены многоядерные процессоры и графические платы. Вопрос эффективного использования вычислительных систем с такой сложной архитектурой в настоящее время изучен недостаточно.
Материалы и методы: Решаются системы обыкновенных дифференциальных уравнений большой размерности на различных технологиях параллельного программирования. Ускорение вычислений определяется с использованием закона Амдала.
Результаты: Приведены численные эксперименты по нахождению путей эффективного использования гетерогенной системы.
Выводы: Результаты численных экспериментов показали, что комбинация различных технологий параллельного программирования, учитывающая особенности архитектуры системы, является средством повышения эффективности использования гетерогенной системы.

Ключевые слова: гетерогенные системы, эффективность вычислений, методы решения обыкновенных дифференциальных уравнений больших размерностей.

Effective Usage of Heterogeneous Computing Systems

L.P. Chernysheva, D.P. Haritonov
Ivanovo State Power Engineering University, Ivanovo, Russian Federation
E-mail: chernlu@vvs.ispu.ru

Abstract

Background: In heterogeneous systems a sophisticated way of organizing a parallel computing system is used, they combined multi-core processors and graphics cards. The important issue of efficient use of computing systems with a complex architecture is currently poorly researched.
Materials and methods: The systems of ordinary differential equations in various large-scale parallel programming technologies are being solved. Acceleration of calculations is determined by using the law of Amdal.
Results: Numerical experiments are carried out to find the ways of efficient use of a heterogeneous system.
Conclusion: The results of numeric experiments show that the combination of different parallel programming, taking into account the characteristics of the system architecture, is a means of improving efficiency of the heterogeneous system.

Key words: heterogeneous system, calculations efficiency, methods for the solution of ordinary differential equations of large dimension.

Введение. Современные высокопроизводительные вычислительные системы (ВВС) являются гетерогенными (гибридными). Они состоят из компонентов двух основных типов: многоядерного центрального процессора и массивно-параллельных ускорителей, например графических процессоров (GPU) фирмы NVIDIA. Гибридные технологии используют самые мощные суперкомпьютеры в мире.

На данных системах стало возможным создавать высокопроизводительные приложения для такой требовательной к вычислительным ресурсам области, как математическое моделирование физических процессов, чрезвычайных ситуаций.

Важной характеристикой ВВС является ее производительность. Производительность любой вычислительной системы определяется следующими факторами: составом, размером, временными характеристиками вычислительной компоненты и памяти данной системы, структурой и пропускной способностью коммуникационной среды, используемыми компилятором и операционной системой.

Была поставлена задача – исследовать производительность гетерогенной вычислительной системы*.

Анализ производительности проводился на основе результатов численных экспериментов при решении системы обыкновенных дифференциальных уравнений и с использованием специальной библиотеки программ для решения задач линейной алгебры BLAS. Данная библиотека используется в популярном пакете оценки производительности LINPACK, на ос-

* Работа выполнена при поддержке Министерства образования и науки РФ ГК№13 G25.31.0077.

© ФГБОУВПО «Ивановский государственный энергетический университет имени В. И. Ленина»

нове которого формируется список самых мощных суперкомпьютеров в мире (Top500) и в СНГ (Top50).

Исследования проводились на гетерогенной вычислительной системе со следующими техническими характеристиками. Система состоит из двух стоек. Каждая стойка содержит четырехядерный центральный процессор и четыре графических процессора NVIDIA. Модель графического процессора – GTX 295. Тактовая частота графической подсистемы – 576 МГц. Тактовая частота процессора – 1,242 ГГц. Тактовая частота памяти – 1008 МГц. Интерфейс памяти – 446 бит. Доступная графическая память – 3711 МБ. Выделенная видеопамять – 896 МБ. Разделяемая системная память – 2,815 ГБ. Используемая шина – PCI Express x16 Gen2. Количество ядер исполнения в системе – 240. Это вторая версия аппаратной реализации CUDA (Compute Unified Device Architecture).

Оценка производительности. Пусть дана система обыкновенных дифференциальных уравнений с линейными коэффициентами:

$$\frac{dx_i}{dt} = A_{i1}x_1 + A_{i2}x_2 + A_{13}x_3 + ... + A_{iN}x_N,$$

$$i = 1,2,3,...,N,$$

где x_1, x_2, x_3,…, x_N – параметры, определяющие состояние системы; t – время; A_{i1}, A_{i2}, A_{i3},…, A_{iN}, i = 1, 2, 3,…, N, $A_{ij} \in R$ – линейные коэффициенты, входящие в правые части.

Дана задача Коши, то есть в начальный момент времени известны значения параметров: $x_i = 1,0,\ i = 1,2,3,...,N,\ t^0 = 0,0.$

Необходимо найти значения параметров $x_1, x_2, x_3, ..., x_N$ для $\forall t > t^0$.

Для оценки производительности воспользуемся методом Эйлера первого порядка точности.

Пусть начальные значения параметров известны всем нитям. Вычисления могут быть распараллелены за счет вычисления на разных нитях новых значений параметров. Но для перехода к следующему моменту времени необходима синхронизация и видимость новых значений параметров всеми нитями. В данном случае синхронизация не будет приводить к потерям времени, так как вычисление правых частей потребует одинаковых затрат времени на каждой нити. Поэтому и была выбрана система обыкновенных дифференциальных уравнений с линейными коэффициентами, чтобы оценить производительность вычислительной системы, не теряя времени на несбалансированный алгоритм.

Значения коэффициентов системы A_{ij}, i = 1, 2, 3,…, N, j = 1, 2, 3,…, N, хранятся в файле и получены с помощью нахождения матрицы, у которой известны ее собственные числа. Количество параметров в системе обыкновенных уравнений N выбирается произвольно и может быть достаточно большим числом.

Решим данную задачу с помощью технологии OpenMP. Гетерогенная вычислительная система имеет четыре ядра с общей памятью. Будем использовать четыре нити в параллельной секции. Организуем вычисления таким образом, чтобы равномерно загрузить все нити. Для этого разделим уравнения между нитями на приблизительно одинаковые части. Каждая нить будет вычислять новые значения для своей части параметров, и все нити работают одновременно, что должно дать ускорение вычислений.

При использовании технологии CUDA все вычисления будем проводить на устройстве (Device). Определим функцию, которая вычисляет правые части, причем каждая нить, обратившись к данной функции, будет вычислять только одну правую часть.

Функция запуска ядра выполняет векторные операции, в которых каждая нить вычисляет только одну координату вектора, и все нити работают одновременно. Это дает заметное ускорение вычислений. Для перехода к следующему шагу вычислений обязательно необходима синхронизация, для того чтобы вновь вычисленные значения были доступны всем нитям.

Основная программа, которой первой передается управление при запуске программы, определяет переменные, используемые в программе, считывает из файла массив линейных коэффициентов. Для более удобной работы на Device этот массив из двумерного переписывается в одномерный массив. Выделяется память под массивы на Device. На ядре определяется основной цикл вычислений. Для определения ускорения вычисляется время выполнения расчетов. Для определения ускорения вычислений по закону Амдала вычислим время, потраченное на выполнение последовательной части параллельной программы.

Воспользуемся более точным методом второго порядка точности – методом Рунге-Кутты 2 (улучшенный метод Эйлера). Этот метод состоит из двух этапов. Пусть начальные значения параметров известны всем нитям. Вычисления могут быть распараллелены за счет вычисления на разных нитях значений параметров в средней точке шага по времени, значений правых частей в средней точке шага по времени и вычисления новых значений параметров. Но на первом и втором этапах вычислений необходима синхронизация и видимость новых значений параметров всеми нитями. В данном случае синхронизация не будет приводить к потерям времени, так как вычисление правых частей потребует одинаковых затрат времени на каждой нити за счет специально подобранной системы обыкновенных дифференциальных уравнений. По срав-

© ФГБОУВПО «Ивановский государственный энергетический университет имени В. И. Ленина»

нению с выше приведенным методом Эйлера, в данном методе три раза необходимо проводить синхронизацию.

Для решения данной задачи воспользуемся технологией OpenMP. Будем использовать четыре нити в параллельной секции. Организуем вычисления таким образом, чтобы равномерно загрузить все нити. Для этого разделим уравнения между нитями на приблизительно одинаковые части. Каждая нить будет вычислять значения для своей части параметров, и все нити работают одновременно. Затем необходима синхронизация. После этого все нити одновременно вычисляют значения правых частей для своей части параметров. Опять проводим синхронизацию и вычисляем новые значения параметров на четырех нитях, что даст ускорение вычислений.

При использовании технологии CUDA все вычисления будем проводить на устройстве Device.

Функция запуска ядра Kernel() выполняет векторные операции, в которых каждая нить вычисляет только одну координату вектора значений параметров в средней точке шага по времени и все нити работают одновременно. Для перехода к вычислению правых частей в средней точке шага по времени обязательно необходима синхронизация, для того чтобы вновь вычисленные значения были доступны всем нитям. После этого вновь проводим синхронизацию и вычисляем на Device новые значения параметров. На ядре определяется основной цикл вычислений. Для определения ускорения вычисляется время выполнения расчетов и время выполнения последовательной части программы.

Анализ результатов численных экспериментов. При использовании метода Эйлера были проведены расчеты при различном числе параметров в системе: N = 10, 20, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500. При этом замерялось время вычислений в однопроцессорном варианте, при использовании системы параллельного программирования OpenMP и с помощью технологии CUDA. Результаты экспериментов представлены в табл. 1 (время вычислений указано в секундах).

Анализ полученных результатов показывает, что по мере увеличения числа параметров N увеличивается время расчетов. При небольших размерностях лучшие результаты получены в однопроцессорном варианте расчета. При числе параметров N<450 программа на OpenMP показывает результат лучше, чем программа на CUDA. Как только число параметров N больше или равно 450 происходит качественный скачок: время вычислений в однопроцессорном варианте и с использованием технологии OpenMP резко увеличивается, в то время как расчеты на CUDA стабильны.

Отсутствие скачка по времени для CUDA при увеличении числа параметров N объясняется самой архитектурой графических плат, которая основана на концепции SIMD (Single Instruction stream – Multiple Data stream). На гетерогенной системе одна инструкция позволяет одновременно обрабатывать множество данных. Так как количество ядер исполнения на плате равно 240, то при выполнении программы на OpenMP на одной нити вычисляется 125 параметров при N = 500, а на CUDA на одной нити вычисляются 2–3 параметра. На CUDA запускается одна инструкция для всех нитей, она одновременно выполняется на всех блоках системы.

Таблица 1. **Время вычислений методом Эйлера**

Количество параметров системы, N	Однопроцессорный вариант	Программа на OpenMP	Программа на CUDA
10	0,047094	8,327547	8,784894
20	0,159673	8,196842	11,474345
50	0,833504	8,310476	27,116039
100	3,287302	8,861372	27,618175
150	7,304334	9,852252	24,870775
200	12,765929	11,479267	30,059693
250	20,468849	13,725429	26,028151
300	28,414301	15,936759	28,130957
350	38,636586	18,907467	28,130957
400	50,303642	22,365452	26,320322
450	685,751731	116,339088	25,283171
500	1657,96427	258,010311	31,155687

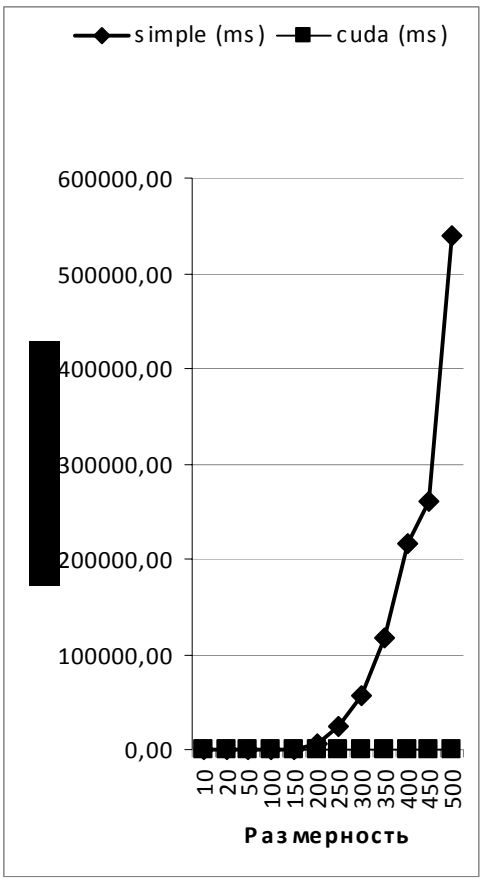
При реализации на OpenMP каждая нить содержит как инструкцию, так и данные, с которыми работает. После выполнения инструкции каждая нить обращается к общей памяти, куда записывает вычисленные значения. Очевидно, что при небольшом числе параметров можно проводить вычисления как на OpenMP, так и на CUDA, причем CUDA будет выполняться дольше. Но как только переходим к большим размерностям, преимущества расчетов на CUDA очевидны.

Если учитывать результаты расчетов и в однопроцессорном варианте (см. рисунок), то преимущества CUDA становятся еще более очевидными. При выполнении вычислений на больших расчетных областях использование однопроцессорного варианта совершенно неэффективно. В случае проведения расчетов в режиме реального времени, когда необходимо быстро реагировать на изменение реальных условий, целесообразным является использование технологий OpenMP и CUDA при небольших размерностях и CUDA при больших размерностях расчетной области.

Для вычисления ускорения воспользуемся отношением времени вычислений при использовании одной нити к времени вычислений на нескольких нитях и законом Амдала.

Значения ускорения вычислений на OpenMP и CUDA в зависимости от размерности системы представлены в табл. 2.

© ФГБОУВПО «Ивановский государственный энергетический университет имени В. И. Ленина»



Зависимость времени вычислений от размерности системы в однопроцессорном варианте и на CUDA (время указано в миллисекундах)

Таблица 2. **Ускорение вычислений**

Технологии	Количество параметров системы, *N*			
	360	400	450	500
OpenMP	2,12358	2,24917	5,90494	6,44152
CUDA	1,42549	1,91099	27,12285	51,76969

При использовании метода Рунге-Кутты 2 были проведены расчеты при тех же значениях параметров в системе: *N* = 10, 20, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500. При этом замерялось время вычислений в однопроцессорном варианте, при использовании системы параллельного программирования OpenMP и с помощью технологии CUDA. Результаты экспериментов представлены в табл. 3 (время вычислений указано в секундах).

Расчеты в однопроцессорном варианте при *N* = 500 не проводились, так как расчеты при *N* = 450 уже заняли приблизительно 23 минуты. Как только число параметров *N* становится больше 400, время выполнения программы на OpenMP резко возрастает, в то время как программа на CUDA выполняется на 10 с дольше по сравнению с предыдущим опытом.

Сравнительный анализ времени выполнения программ на CUDA методами Эйлера и Рунге-Кутты показывает, что временные затраты незначительно различаются, несмотря на то, что в методе Рунге-Кутты 2 необходимо выполнять синхронизацию данных три раза. Так как время вычислений приблизительно одинаковое, то предпочтительно применять метод Рунге-Кутты 2, как метод с более высоким порядком точности.

Таблица 3. **Время вычислений методом Рунге-Кутты 2**

№ опыта	Количество параметров системы, *N*	Однопроцессорный вариант	Программа на OpenMP	Программа на CUDA
1	10	0,0869	10,7992	9,1374
2	20	0,2974	10,8086	14,0926
3	50	1,6095	11,0511	33,1983
4	100	6,5553	12,5667	32,8061
5	150	4,3347	14,5489	32,7685
6	200	25,2397	17,5807	27,3600
7	250	39,2823	21,6133	28,3385
8	300	56,4879	26,2712	26,3089
9	350	76,9093	31,8031	26,5860
10	400	99,9336	38,8574	26,3846
11	450	1373,073	227,1335	25,3642
12	500	–	511,0161	35,3607

Использование теста LINPACK для оценки производительности гетерогенной вычислительной системы. Для оценки производительности суперкомпьютеров существует набор тестов производительности HPC Challenge Benchmark. Входящие в него тесты предназначены для оценки нескольких атрибутов суперкомпьютеров, которые значительно влияют на производительность реальных высокопроизводительных задач. В этот набор входит тест HPL (Highly Parallel Linpack). Тест LINPACK оценивает производительность вычислений с плавающей запятой и оценивает коммуникацию между частями суперкомпьютера. В тесте LINPACK выполняется решение плотной СЛАУ методом LU-разложения. Он используется для составления списка Top500 суперкомпьютеров мира.

Тест состоит в решении системы линейных уравнений с помощью LU-факторизации. Основное время затрачивается на векторные операции типа FMA (умножение и сложение). Производительность определяется как количество «полезных» вычислительных операций над числами с плавающей точкой в расчете на 1 секунду.

Для выполнения вычислений используется библиотека BLAS (Basic Linear Algebra Subprograms) – набор базовых операций линейной алгебры. Вычислительные ядра в библиотеке BLAS написаны на автокоде.

Для оценки производительности необходимо, чтобы все устройства гетерогенной системы работали с максимальной нагрузкой. В этом случае получим пиковую производительность. Конечно, при решении реальных задач трудно получить такую производительность гетерогенной вычислительной системы.

© ФГБОУВПО «Ивановский государственный энергетический университет имени В. И. Ленина»

Для оценки производительности гетерогенной вычислительной системы воспользуемся функцией, выполняющей перемножение матриц и умножение полученной матрицы на скаляр. Размер матриц N будем изменять от 300 до 6000 элементов.

Проведены вычисления для однопроцессорного варианта и с использованием технологии CUDA.

Анализ результатов. Пусть N – размер матриц – принимает значения 300, 350, 400, 450, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 5000, 6000. Расчеты для N = 5000 и N = 6000 в однопроцессорном варианте не проведены, так как уже при N = 4000 расчеты занимают 9 минут. Полученные результаты представлены в табл. 3.

Таблица 4. **Время выполнения теста LINPACK**

Количество параметров системы, N	Однопроцессорный вариант (время в секундах)	Программа на CUDA (время в миллисекундах)
300	1,0	0,255168
350	1,1	0,356832
400	1,5	0,471136
450	1,6	0,679136
500	1,7	0,976736
1000	6,0	6,312384
1500	25,0	20,524288
2000	56,0	47,921024
2500	118,0	93,937347
3000	216,0	157,646179
3500	360,0	251,305191
4000	540,0	374,056915
5000	–	735,291565
6000	–	1254,519531

При $N \leq 1000$ элементов полученное время приблизительно одинаково как в однопроцессорном, так и в многопроцессорном вариантах. По мере увеличения размерности N различие становится все более очевидным. При $N \geq 5000$ элементов вычисления в однопроцессорном варианте занимают более 10 мин, в то время как вычисления на CUDA выполнялись приблизительно за одну минуту.

Результаты вычислительных экспериментов показали, что матрицы размера 100×100 или 1000×1000 слишком малы для гетерогенной вычислительной системы. И лишь при размере 6000×6000 можно подойти к важным и интересным значениям оценки производительности. Только при решении задач такой и выше размерностей можно получить пиковую производительность.

Заключение

Исследование производительности гетерогенной системы на системе обыкновенных дифференциальных уравнений с линейными правыми частями с помощью методов Эйлера и Рунге-Кутты 2 показало, что результаты, полученные при использовании метода Рунге-Кутты 2, аналогичны результатам для метода Эйлера, но сам метод имеет второй порядок точности.

Исследование, проведенное с помощью программы cuBlas, входящей в общепринятый тест производительности суперкомпьютеров LINPACK, показало, что использование CUDA на гетерогенной системе дает наилучшие по сравнению с OpenMP результаты при больших размерностях системы.

Список литературы

1. **Воеводин В.В., Воеводин Вл.В.** Параллельные вычисления. – СПб.: БХВ- Петербург, 2002. – 608 с.

2. **Эндрюс Г.Р.** Основы многопоточного, параллельного и распределенного программирования: пер. с англ. – М.: Изд. дом «Вильямс», 2003. – 512 с.

3. **Эхтер Ш., Робертс Дж.** Многоядерное программирование. – СПб.: Питер, 2010. – 316 с.

4. **Боресков А.В., Харламов А.А.** Основы работы с технологией CUDA. – М.: ДМК Пресс, 2011. – 232 с.

5. **Сандерс Дж., Кэндрот Э.** Технологии CUDA в примерах: введение в программирование графических процессоров: пер. с англ. – М.: ДМК Пресс, 2011. – 232 с.

6. **Миллер Р., Боксер Л.** Последовательные и параллельные алгоритмы: общий подход: пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2012. – 406 с.

References

1. Voevodin, V.V., Voevodin, Vl.V. *Parallel'nye vychisleniya* [Parallel calculations]. Saint-Petersburg, BKhV-Peterburg, 2002. 608 p.

2. Endryus, G.R. *Osnovy mnogopotochnogo, parallel'nogo i raspredelennogo programmirovaniya* [Basics of multithreaded, parallel and distributed programming]. Moscow, Izdatel'skiy dom «Vil'yams», 2003. 512 p.

3. Ekhter, Sh., Roberts, Dzh. *Mnogoyadernoe programmirovanie* [Multicore programming]. Saint-Petersburg, Piter, 2010. 316 p.

4. Boreskov, A.V., Kharlamov, A.A. *Osnovy raboty s tekhnologiey CUDA* [Basics of work with CUDA technology]. Moscow, DMK Press, 2011. 232 p.

5. Sanders, Dzh., Kendrot, E. *Tekhnologii CUDA v primerakh: vvedenie v programmirovanie graficheskikh protsessorov* [CUDA technologies in examples: introduction to programming with graphic cards]. Moscow, DMK Press, 2011. 232 p.

6. Miller R., Bokser L. *Posledovatel'nye i parallel'nye algoritmy: obshchiy podkhod* [Serial and parallel algorithms: common approach]. Moscow, BINOM, Laboratoriya znaniy, 2012. 406 p.

Чернышева Людмила Павловна,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
старший преподаватель кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29.

Харитонов Денис Петрович,
ФГБОУВПО «Ивановский государственный энергетический университет имени В.И. Ленина»,
аспирант кафедры высокопроизводительных вычислительных систем,
телефон (4932) 26-98-29.

© ФГБОУВПО «Ивановский государственный энергетический университет имени В. И. Ленина»