Расширение возможностей DVM-системы для решения задач, использующих нерегулярные сетки*

В.А. Бахтин[1,2], А.С. Колганов[1,2], В.А. Крюков[1,2], Н.В. Поддерюгина[1], М.Н. Притула[1]

ИПМ им. М.В. Келдыша РАН[1], МГУ имени М.В. Ломоносова[2]

DVM-система предназначена для разработки параллельных программ научно-технических расчетов на языках C-DVMH и Fortran-DVMH. Эти языки используют единую модель параллельного программирования (DVMH-модель) и являются расширением стандартных языков Си и Фортран спецификациями параллелизма, оформленными в виде директив компилятору. DVMH-модель позволяет создавать эффективные параллельные программы для гетерогенных вычислительных кластеров, в узлах которых в качестве вычислительных устройств наряду с универсальными многоядерными процессорами могут использоваться ускорители (графические процессоры или сопроцессоры Intel Xeon Phi). В статье будут рассмотрены новые возможности работы с нерегулярными сетками, которые были реализованы в компиляторе CDVMH в последнее время. Использование разработанного расширения позволяет существенно упростить распараллеливание на кластер имеющихся приложений на нерегулярных сетках.

Ключевые слова: Си, нерегулярная сетка, неструктурированная сетка, параллельное программирование, директивное расширение языка, распараллеливание на кластер.

1. Введение

Для удовлетворения желания увеличения точности вычислений, исследователям-вычислителям приходится значительно измельчать расчетную сетку. Это приводит к пропорциональному росту потребления памяти ЭВМ и увеличению времени расчетов. Частично с этой проблемой позволяет справиться переход с использования структурированных сеток на неструктурированные. В этом случае появляется возможность варьировать подробность сетки по расчетной области, тем самым сократив и время на излишне точный обсчет некоторых областей, и оперативную память, освобожденную от хранения невостребованно подробных полей величин. Также привлекательной чертой является абстрагирование численных методов от геометрии расчетной области и практическое снятие требований к ней.

Все больше программ пишется в более общем виде: для нерегулярных сеток с прицелом на широкое применение и переиспользование программных кодов [2]. Однако, такие программы значительно сложнее по своей структуре. При работе с регулярными сетками отношение соседства, равно как и пространственные координаты, не приходилось хранить явно, так как эти свойства и величины напрямую связывались с многомерными индексными пространствами массивов величин. Такой подход наряду с очевидным преимуществом в виде экономии памяти, также задавал понятные правила для распараллеливания вычислений как на уровне векторизации, так и на уровне вычислительных кластеров и сетей. С одной стороны, оптимизирующие компиляторы наблюдают обращения к элементам массивов с константными смещениями, что позволяет организовать одновременное выполнение сразу нескольких витков цикла. С другой стороны, развился подход параллелизма по данным, в котором массивы данных разрезаются на блоки, каждый блок обрабатывается отдельным процессором с помощью той же (исходной) программы, время от времени обмениваясь граничными элементами.

К решению данной проблемы практикуется несколько подходов:

- предлагаются способы написания параллельных программ на нерегулярных сетках [2, 4, 8];
- разрабатываются подходы к распараллеливанию последовательных программ на неструктурированных сетках вручную [1];

* Работа поддержана программой Президиума РАН № II.4П и грантом РФФИ 16-07-01067.

- разрабатываются наборы инструментальных средств (библиотеки функций), скрывающие сложности, возникающие при распараллеливании для распределенной памяти [7];
- разрабатываются автоматические механизмы распараллеливания, основанные на модели инспектор/исполнитель [3, 9];
- разрабатываются специализированные языки [6].

Модель DVMH [5] построена на парадигме параллелизма по данным. В основе этой модели лежит понятие распределенного многомерного массива. При этом у каждого процессора имеется не только локальная часть распределенного массива, но и так называемые теневые грани – копии элементов из локальных частей соседних процессоров, через которые осуществляется основное взаимодействие процессоров. Распределение вычислений производится посредством их отображения на распределенные массивы, и, вследствие заранее известных смещений по индексам используемых массивов величин, обращения происходят либо в свою локальную часть, либо в теневые грани, определяемые как продолжение локальной части по конкретному измерению распределенного массива на заранее известную ширину. Например, для шаблона типа “крест” с 4 соседями, элемент с индексами (i,j) рассчитывается по элементам с индексами (i‑1,j), (i,j-1), (i+1,j), (i,j+1), что приводит к необходимости иметь теневые грани ширины 1 по обоим измерениям.

DVMH-компиляторы преобразуют обращения к распределенным многомерным массивам к форме, независимой от размеров и положения локальной части на каждом процессоре, при этом исходные индексные выражения остаются нетронутыми. В результате каждое обращение к распределенным данным ведется в глобальных (исходных) индексах, а при доступе к памяти применяются вычисляемые во время выполнения коэффициенты и смещения для каждого измерения. Такой подход (в отличие от изменения индексных выражений) позволяет абстрагироваться от содержания распараллеливаемых циклов, но и вводит серьезное ограничение на форму адресуемой каждым процессором части распределенного массива, называемой расширенной локальной частью, что является объединением локальной части и теневых граней. Так, в модели DVMH имеются только блочные распределения с теневыми гранями, т.е. расширенная локальная часть представляет собой подмассив исходного массива вида (A_1:B_1, A_2:B_2, A_3:B_3, ..., A_n:B_n). В качестве замечания можно отметить, что модель DVMH не дает средств описания даже циклических распределений (которые, наряду с блочными, используются при распараллеливании программ на регулярных сетках), так как их поддержка потребовала бы выполнение операций деления при каждом обращении к массивам и была отвергнута в целях оптимизации.

В данной статье обсуждается два пути расширения возможностей DVM-системы, нацеленные на борьбу с этим ограничением. Первый из них позволяет распределять данные пользователю вручную средствами MPI или с помощью другой технологии параллельного программирования, оставляя возможность использования DVM-языков внутри узла (многоядерные ЦПУ, ГПУ). Второй путь предполагает значительное расширение DVM-языков и изменение DVMH-компиляторов с целью введения новых типов распределенных массивов, параллельных циклов и иных вспомогательных конструкций, позволяющих существенно упростить распараллеливание на кластер имеющихся приложений на нерегулярных сетках.

2. Использование средств DVMH в MPI-программах

В настоящее время, когда параллельные машины уже не одно десятилетие эксплуатируются для проведения расчетов, имеется множество программ, которые уже распараллелены на кластер, однако не имеют распараллеливания по ядрам ЦПУ, а также не используют ГПУ.

Традиционно в DVM-подходе весь процесс программирования (или распараллеливания имеющихся последовательных программ) начинается с распределения массивов, а затем отображения на них параллельных вычислений. Это означает, что для использования средств DVM-системы распараллеленные, например, на MPI, программы приходится превращать обратно в последовательные и заменять распределенные вручную данные и вычисления на описанные на DVM-языке распределенные массивы и параллельные циклы.

Однако, во-первых, автору не всегда хочется отказываться от своей параллельной программы, а во-вторых, не всегда удастся перевести исходную схему распределения данных и

вычислений на DVM-язык. В частности, перевод задач на нерегулярных сетках в модель DVMH потребует применения нетривиальных решений и трюков и не всегда возможно.

Одним из способов избавиться от обеих проблем является новый режим работы DVM-системы, в котором она не вовлечена в межпроцессорное взаимодействие, а работает локально в каждом процессе.

Данный режим включается заданием специально созданной MPI-библиотеки при сборке DVM-системы. Эта библиотека не производит никаких коммуникаций и не конфликтует с реальными MPI-реализациями. В результате для системы поддержки выполнения DVMH-программ создается иллюзия запуска программы на 1 процессоре.

Кроме такого режима, в компилятор C-DVMH введено понятие нераспределенного параллельного цикла, для которого нет необходимости задавать отображение на распределенный массив. Например, трехмерный параллельный цикл может выглядеть так:

```
#pragma dvm parallel(3)
for (int i = L1; i <= H1; i++)
    for (int j = L2; j <= H2; j++)
        for (int k = L3; k <= H3; k++)
...
```

По определению такой цикл выполняется всеми процессорами текущей многопроцессорной системы, но т.к. DVM-система в описанном новом режиме считает многопроцессорной системой ровно один процесс, то такая конструкция не приводит к размножению вычислений, а только лишь позволяет использовать параллелизм внутри одного процесса (ЦПУ или ГПУ). Как следствие, появляется возможность не задавать ни одного распределенного в терминах модели DVMH массива и в то же время пользоваться возможностями DVM-системы:

- добавление параллелизма в общей памяти (ядра ЦПУ): с использованием OpenMP или без, возможность задания привязки нитей;
- использование ГПУ: не только «наивное» портирование параллельного цикла на ускоритель, но и выполнение автоматической реорганизации данных, упрощенное управление перемещениями данных;
- подбор оптимизационных параметров;
- удобные средства отладки производительности.

Такой режим может быть использован в том числе для получения промежуточных результатов в процессе проведения полноценного распараллеливания программы в модели DVMH. Он позволяет быстро и заметно проще (распределенные массивы имеют набор ограничений при работе с ними, а при таком подходе их заводить необязательно) получить программу для многоядерного ЦПУ и ГПУ, а также оценить перспективы ускорения целевой программы на кластере с многоядерными ЦПУ и ускорителями.

В качестве примера приведем программу, являющуюся частью большого развитого комплекса вычислительных программ. Будучи ориентированным на решение по явной схеме систем гиперболических уравнений (в основном, газовой динамики) в двумерных областях сложной формы с использованием неструктурированных сеток, этот код был написан на C++ с очень широким использованием объектно-ориентированного подхода для обеспечения максимальной универсальности и простоты дальнейшего развития.

Так как. эта программа является частью целого комплекса, код ее основан на богатой платформе базовых понятий и структур данных. Это приводит к значительным размерам (39 тыс. строк) и сложности всей программы, если ее рассматривать целиком. Полноценное распараллеливание на кластер вряд ли возможно без рассмотрения и модификации всей программы, однако новые возможности DVM-системы позволили выделить относительно нетрудоемкий первый этап распараллеливания по ядрам ЦПУ и на графический ускоритель.

Такое распараллеливание вполне может проводиться «локально», т.е. модификации требуются только в вычислительно-емких частях программы, которые занимают около 3 тыс. строк.

На 12 ядрах ЦПУ было получено ускорение в 9,83 раза относительно последовательной версии, а на ГПУ NVIDIA GTX Titan – в 18 раз относительно последовательной версии. Данные результаты подтверждают эффективность отображения рассматриваемой программы DVM-системой на ускорители и многоядерные ЦПУ и дают основания продолжить распараллеливание программы уже с использованием распределенных массивов в модели DVMH.

3. Новые возможности работы с нерегулярными сетками в компиляторе CDVMH

Для работы с нерегулярными сетками введен новый вид распределения массивов и шаблонов – поэлементное распределение. Этот вид распределения не накладывает никаких ограничений на то, какие элементы массива должны располагаться на одном и том же процессоре, или какие элементы массива должны располагаться на соседних процессорах. Напротив, он позволяет задать произвольную принадлежность каждого элемента массива независимо.

Добавлены два новых правила поэлементного распределения: косвенное (indirect) и производное (derived). Косвенное распределение задается массивом целых чисел, размер которого равен размеру косвенно распределяемого измерения, а значения задают номер домена. При этом доменов может быть как больше числа процессоров, так и меньше. DVM-система гарантирует принадлежность всех элементов одного домена одному и тому же процессору.

Производное распределение задается правилом, по форме похожим на правило выравнивания (ALIGN) модели DVMH. Однако, у него появляется значительно большая гибкость. Синтаксис можно описать так, как показано на рис. 1.

```
      indirect-rule :: indirect ( var-name )
      derived-rule ::= derived ( derived-elem-list with derived-templ )
      derived-elem ::= int-range-expr
      int-range-expr ::= прозвольное целочисленное выражение + в индексных
выражениях допустимы диапазоны, использование align-dummy переменных.
      derived-templ ::= var-name [ derived-templ-axis-spec ]...
      derived-templ-axis-spec ::= [ ] | [ @ align-dummy [ + shadow-name
]... ] | [ int-expr ]
```

Рис. 1. Формула БНФ для новых правил распределения

Все ссылки на распределенные массивы в int-range-expr обязаны быть доступны (элемент входит в расширенную локальную часть) для соответствующего элемента шаблона (перебор элементов шаблона осуществляется по его локальной части и указанным теневым граням). Если производным правилом один и тот же элемент подлежит распределению сразу на несколько процессоров, то DVM-система решает на какой из них фактически будет распределен такой элемент, а на остальных процессорах добавляет его в теневую грань с названием «overlay». Элементов, не распределенных ни на один процессор, быть не должно. Такие случаи являются ошибкой времени выполнения и приводят к останову. Вычисленные несуществующие индексы распределяемого массива игнорируются, не приводя к ошибке.

Наложение (overlay) вводится для возможности согласованного распределения сеточных элементов. Например, ячейки, ребра, вершины. В таком случае появляется возможность построить одно распределение на основе другого, причем в любой последовательности.

В результате такого распределения у массива появляется два вида нумерации элементов: глобальная (она же исходная в последовательной программе) и локальная. Локальная нумерация непрерывна в рамках одного процессора, т.е. существует такой порядок локальных элементов, что их локальные индексы полностью заполнят некоторый целочисленный отрезок [Li, Hi].

Также вводятся поэлементые теневые грани. Теневая грань — это набор элементов, не принадлежащих текущему процессу (требование принадлежности соседнему процессу снимается), для которых, во-первых, возможен доступ без специальных указаний из любой точки программы, и, во-вторых, введены специальные средства работы с ними: обновление указанием shadow_renew, расширение параллельного цикла указанием shadow_compute и т. п.

В отличие от традиционных, поэлементные теневые грани добавляются к шаблонам во время работы программы и имеют имя для ссылки на них. Задаются они практически так же, как и производное распределение, см. рис. 2.

```
      shadow-add ::= shadow_add ( templ-name [ shadow-axis ]... = shadow-
name ) [ include_to ( var-name-list ) ]
      shadow-axis ::= [ ] | [ derived-elem-list with derived-templ ]
```

Рис. 2. Формула БНФ для задания поэлементных теневых граней

Ровно один из shadow-axis должен быть непустыми скобочками. Все массивы из списка, указанного в include_to должны быть выравнены на шаблон, к измерению которого добавляется теневая грань. В результате выполнения такой директивы, к шаблону добавляется теневая грань и включается в указанные распределенные массивы. После этой операции теневые элементы массивов доступны на чтение из программы, а также могут обновляться с помощью директивы shadow_renew.

Для реализации поэлементных теневых граней и производного распределения компилятор на основе указанных выражений генерирует специального вида функцию, в которую передается системой поддержки параметры для обхода локальной части шаблона. Эта функция, обходя шаблон, заполняет буфер индексов элементов согласно выражениям в левой части правила отображения, а затем возвращает обратно в систему поддержки. Затем он анализируется средствами системы поддержки.

Для экспериментальной эксплуатации этих возможностей была введена вспомогательная директива локализации значений индексного массива, которая изменяет значения целочисленного массива, заменяя глобальные индексы указанного целевого массива на локальные (рис. 3).

```
      localize-spec ::= localize ( ref-var-name => target-var-name [ axis-
specifier ]...
      axis-specifier ::= [ ] | [ : ]
```

Рис. 3. Формула БНФ для директивы локализации значений индексного массива

После проведения такой операции становится возможным использовать имеющийся способ компиляции параллельных циклов: они будут выполняться полностью в локальных индексах.

Вместе с модификацией директивы теневых обменов и реализацией обменов для поэлементных теневых граней (которые теперь не обязательно происходят с соседними процессорами, а с произвольным подмножеством процессоров) этот набор расширений позволяет распараллелить и запустить приложения на нерегулярных сетках на кластер с ускорителями.

Были распараллелены два приложения: двумерная задача теплопроводности в шестиграннике (явная и неявная схема) и задача численного моделирования газодинамических течений.

Данные приложения производят расчет как в вершинах сетки, так и в ее ячейках, что потребовало согласованного распределения массивов, локализации индексных массивов, поэлементных теневых граней.

В таблицах 1 и 2 показаны результаты прогонов первого приложения на различном числе ЦПУ и ГПУ кластера К-100.

Таблица 1. Параллельная эффективность DVMH-программ на ЦПУ

Схема	**Посл.**	**Параллельное выполнение, на разном количестве ядер ЦПУ**											
		2		**4**		**8**		**12**		**24**		**96**	
	Время	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.
Явная	174	87,2	2	50	3,49	32	5,45	21,7	8,02	11,1	15,7	2,75	63,3
Неявная	928	728	1,27	611	1,52	449	2,07	309	3	155	6	42,8	21,7

Таблица 2. Параллельная эффективность DVMH-программ на ГПУ

Схема	**Посл.**	**Параллельное выполнение, на разном количестве ГПУ**											
		1		**2**		**3**		**6**		**12**		**24**	
	Время	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.	Время	Уск.
Явная	174	7	24,8	3,81	46	2,76	63	1,5	116	0,87	200	0,55	316
Неявная	928	77	12	42,3	22	29,8	31	16,3	57	9,64	96	6,55	142

4. Заключение

Сделаны шаги по расширению возможностей DVM-системы для решения задач, использующих нерегулярные сетки. Это возможность перехода на ручное распределение данных, а также новые экспериментальные конструкции языка CDVMH и их реализация в компиляторе и системе поддержки, позволяющие распараллелить приложения на нерегулярных сетках на кластер с многоядерными процессорами и ГПУ.

В дальнейшем предполагается обогатить новые средства и улучшить их интеграцию с DVMH-программами с блочно-распределенными данными.

Литература

1. Андрианов А.Н., Ефимкин К.Н. Подход к параллельной реализации численных методов на неструктурированных сетках // Вычислительные методы и программирование. 2007. Т. 8. С 6-17.
2. Козубская Т.К. Разработка моделей и методов повышенной точности для численного исследования задач прикладной аэроакустики. Диссертация на соискание ученой степени доктора физико-математических наук, Москва, 2010.
3. Mahesh Ravishankar, John Eisenlohr, Louis-Noel Pouchet, J. Ramanujam, Atanas Rountev, P. Sadayappan. Code Generation for Parallel Execution of a Class of Irregular Loops on Distributed Memory Systems. SC12, November 10-16, 2012.
4. Lizandro Solano-Quinde, Zhi Jian Wang, Brett Bode, Arun K. Soman. Unstructured grid applications on GPU: performance analysis and improvement. Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, 2011.
5. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Сазанов Ю.Л. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. // Вестник Южно-Уральского университета, серия «Математическое моделирование и программирование», №18 (277), выпуск 12, 2012 г., С. 82-92.
6. Zachary DeVito, Niels Joubert, Francisco Palacios, Stephen Oakley, Montserrat Medina, Mike Barrientos, Erich Elsen, Frank Ham, Alex Aiken, Karthik Duraisamy, Eric Darve, Juan Alonso, Pat Hanrahan. Liszt: A Domain Specific Language for Building Portable Mesh-based PDE Solvers. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), 2011.
7. Василевский Ю.В., Коньшин И.Н., Копытов Г.В., Терехов К.М. INMOST – программная платформа и графическая среда для разработки параллельных численных моделей на сетках общего вида: Учебное пособие // М.: Издательство Московского университета, 2013. – 144 с.
8. Горобец А.В., Суков С.А., Железняков А.О., Богданов П.Б., Четверушкин Б.Н. Применение GPU в рамках гибридного двухуровневого распараллеливания MPI+OpenMP на гетерогенных вычислительных системах // Вестник ЮУрГУ, т. 242, No 25, 2011, С. 76–86.
9. J. Saltz, K. Crowley, R. Mirchandaney, and H. Berryman. Run-time scheduling and execution of loops on message passing machines. J. Parallel Distrib. Comput., vol. 8, pp. 303–312, 1990.

Extension of DVM-system capabilities to solve problems which use irregular grids

V.A. Bakhtin[1,2], A.S. Kolganov[1,2], V.A. Krukov[1,2], N.V. Podderyugina[1], M.N. Pritula[1]

Keldysh Institute of Applied Mathematics Russian Academy of Sciences[1],
Lomonosov Moscow State University[2]

DVM-system was designed to create parallel programs of scientific-technical calculations in C-DVMH and Fortran-DVMH languages. These languages use the same model of parallel programming (DVMH-model) and are the extensions of standard C and Fortran languages with parallelism specifications, implemented as compiler directives. DVMH-model allows to create efficient parallel programs for heterogeneous computational clusters, which nodes use as computing devices not only universal multi-core processors but also can use attached accelerators (GPUs or Intel Xeon Phi coprocessors). This article discusses new means to work with irregular grids, which were implemented in the C-DVMH compiler recently. Using the developed extension can significantly ease irregular grid applications parallelization on cluster.

Keywords: irregular grid, unstructured grid, parallel programming, directive-based language, cluster parallelization.

References

1. Andrianov A.N., Efimkin K.N. Podhod k parallel'noy realizacii chislennyh metodov na nestrukturirovannyh setkah [Approach for Parallel Implementation of Numerical Methods on Unstructured Grids]. Vychislitel'nye metody i programmirovanie [Numerical Methods and Programming]. 2007, vol. 8, P. 6-17.
2. Kozubskaya T.K. Razrabotka modeley i metodov povyshennoy tochnosti dlya chislennogo issledovania zadach prikladnoy aeroakustiki [Development of Models and Methods of Increased Accuracy for Numerical Analysis of Applied Aeroacoustic Problems]. PhD thesis. Moscow, 2010.
3. Mahesh Ravishankar, John Eisenlohr, Louis-Noel Pouchet, J. Ramanujam, Atanas Rountev, P. Sadayappan. Code Generation for Parallel Execution of a Class of Irregular Loops on Distributed Memory Systems. SC12, November 10-16, 2012.
4. Lizandro Solano-Quinde, Zhi Jian Wang, Brett Bode, Arun K. Soman. Unstructured grid applications on GPU: performance analysis and improvement. Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, 2011.
5. Bakhtin V.A., Klinov M.S., Krukov V.A., Podderyugina N.V., Pritula M.N., Sazanov Y.L. Rasshirenie DVM-modeli parallel'nogo programmirovania dlya klasterov s geterogennymi uzlami [Extension of DVM Model of Parallel Programming for Clusters with Heterogeneous Nodes]. Vestnik Yuzhno-Ural'skogo universiteta, seriya "Matematicheskoe modelirovanie i programmirovanie" [Bulletin of the South Ural State University, Series "Mathematical Modelling, Programming & Computer Software"]. No 18 (277), issue 12, 2012, P. 82-92.
6. Zachary DeVito, Niels Joubert, Francisco Palacios, Stephen Oakley, Montserrat Medina, Mike Barrientos, Erich Elsen, Frank Ham, Alex Aiken, Karthik Duraisamy, Eric Darve, Juan Alonso, Pat Hanrahan. Liszt: A Domain Specific Language for Building Portable Mesh-based PDE Solvers. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), 2011.
7. Vasilevskiy Y.V., Konshin I.N., Kopytov G.V., Terekhov K.M. INMOST – programmnaya platform i graficheskaya sreda dlya razrabotki parallel'nyh chislennyh modeley na setkah obschego vida [INMOST – Programming Platform and Graphical Environment for Development of Parallel

Numerical Models on Various Grids]. Textbook. Moscow, Publishing of the Moscow State University, 2013. 144 p.

8. Gorobets A.V., Sukov S.A., Zheleznyakov A.O., Bogdanov P.B., Chetverushkin B.N. Primenenie GPU v ramkah gibridnogo dvuhurovnevogo rasparallelivania MPI+OpenMP na geterogennyh vychislitel'nyh sistemah [Usage of GPU in Hybrid Two-Level MPI+OpenMP Parallelization on Heterogeneous Systems]. Vestnik YUUrGU [Bulletin of the South Ural State University]. Vol. 242, No 25, P. 76-86.
9. J. Saltz, K. Crowley, R. Mirchandaney, and H. Berryman. Run-time scheduling and execution of loops on message passing machines. J. Parallel Distrib. Comput., vol. 8, pp. 303–312, 1990.