УДК 004,432 **DOI: 10.14529/cmse190106**

РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ DVM-СИСТЕМЫ*

© 2019 В.А. Бахтин, Д.А. Захаров, А.С. Колганов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула

ИПМ им. М.В. Келдыша РАН

(125047 Москва, Миусская пл., д. 4)

E-mail: bakhtin@keldysh.ru, s123-93@mail.ru, alexander.k.s@mail.ru, krukov@keldysh.ru, konov@keldysh.ru, pritula@keldysh.ru

Поступила в редакцию: 21.06.2018

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 описывается опыт использования DVM-системы для распараллеливания различных прикладных программ. Рассматривается метод инкрементального или частичного распараллеливания, возможности системы для работы с неструктурированными сетками, новые средства для отображения MPI-программ на многоядерные процессоры и ускорители. Исследуется эффективность выполнения параллельных DVMH-программ на гетерогенных вычислительных кластерах K-10, K-100, Ломоносов и MVS-10P. Описаны основные преимущества DVM-подхода при разработке параллельных программ. Представлены основные возможности инструментов DVM-системы для анализа производительности и функциональной отладки параллельных программ. Определяются направления для дальнейшего развития DVM-системы.

Ключевые слова: автоматизация разработки параллельных программ, DVM-система, спецификации параллелизма, ускоритель, графический процессор, сопроцессор, Фортран, Си.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Бахтин В.А., Захаров Д.А., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н. Решение прикладных задач с использованием DVM-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 1. С. 89–106. DOI: 10.14529/cmse190106.

Введение

В последнее время все большее распространение получают вычислительные кластеры, в узлах которых установлены ускорители различной архитектуры. Внедрение таких вычислительных систем серьезно усложнило процесс разработки прикладных программ. Для высокопроизводительных вычислений на современных кластерах широко используются три модели программирования — MPI, OpenMP и CUDA. При этом пока вполне можно обходиться комбинацией MPI/OpenMP или MPI/CUDA, поскольку графические ускорители (далее ГПУ) используют для тех программ, для которых они значительно эффективнее, чем многоядерные универсальные процессоры (далее ЦПУ), и, следовательно, неполной загрузкой многоядерных процессоров можно пренебречь. Если будут

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России — 2017».

появляться программы, в которых только часть вычислений выгодно производить на ГПУ, а оставшиеся вычисления лучше оставить на ЦПУ, то придется использовать и комбинацию из трех перечисленных моделей.

Технически объединять низкоуровневые модели программирования, реализованные через библиотеки, проще, чем высокоуровневые модели, реализуемые посредством языков и соответствующих компиляторов. Но программировать, отлаживать, сопровождать, переносить на другие ЭВМ такие программы гораздо сложнее. Например, при переходе от ГПУ фирмы NVIDIA к ГПУ фирмы AMD придется заменить CUDA на OpenCL. Поэтому важно использовать высокоуровневые модели и языки программирования.

Среди высокоуровневых моделей программирования особое место занимают модели, реализуемые посредством добавления в программы на стандартных последовательных языках спецификаций, управляющих отображением этих программ на параллельные машины. Эти спецификации, оформляемые в виде комментариев в Фортран-программах или директив компилятору (прагм) в программах на языках Си и Си++, не видны для обычных компиляторов, что значительно упрощает внедрение новых моделей параллельного программирования. Такими моделями для кластеров являются HPF и XcalableMP, для мультипроцессоров — OpenMP, для ускорителей — OpenACC, для гибридных вычислительных кластеров с ускорителями — DVMH.

Система автоматизации разработки параллельных программ (DVM-система) существенно упрощает процесс разработки параллельных DVMH-программ. Получаемые программы без каких-либо изменений могут выполняться на кластерах, использующих многоядерные процессоры, графические ускорители и сопроцессоры Intel Xeon Phi. Это достигается за счет различных оптимизаций, которые выполняются как статически, при компиляции DVMH-программ, так и динамически. Получаемые параллельные программы могут настраиваться при запуске на выделенные для их выполнения ресурсы — количество узлов кластера, ядер, ускорителей и их производительность [1].

Данные оптимизации позволяют добиться высокой эффективности выполнения различных тестовых программ (например, программ из пакета NAS NPB [2, 3]) и реальных приложений (получены результаты при использовании более 1000 ГПУ [4]).

DVM-система активно развивается, появляются новые возможности, которые расширяют область применимости системы. Например, позволяют распараллеливать не только задачи на структурированных сетках (для которых DVM-система была предназначена изначально), но и задачи на неструктурированных сетках, а также использовать средства DVM для добавления новых уровней параллелизма в уже существующие MPI-программы.

В статье представлен опыт разработки параллельных программ с использованием DVM-системы. В разделе 1 представлены результаты инкрементального распараллеливания программы моделирования многокомпонентной фильтрации при разработке месторождений нефти и газа. В разделе 2 описан опыт распараллеливания программы, решающей двумерную задачу теплопроводности в шестиграннике. В разделе 3 демонстрируется использование средств DVM для дополнительного распараллеливания MPI-программы, решающей по явной схеме систему гиперболических уравнений в двумерной области сложной формы. В разделе 4 перечислены основные возможности DVM-системы, сформулированы преимущества DVM-подхода при разработке параллельных программ. В заключении обобщаются основные результаты исследования и определяются направления для дальнейшего развития DVM-системы.

1. Распараллеливание прикладных задач, использующих структурированные сетки

Распараллеливание программы в модели DVMH можно разделить на следующие этапы:

1. Распределение данных (массивов) и вычислений (параллельных циклов).
2. Определение и спецификация удаленных данных (данные, которые вычисляются на одном процессоре, а используются на других).
3. Определение вычислительных регионов (последовательности операторов, циклов) для выполнения на ускорителях.
4. Управление перемещением данных между памятью ЦПУ и памятью ускорителей.

Основная сложность разработки параллельной программы для кластера — необходимость принятия глобальных решений по распределению данных и вычислений с учетом свойств всей программы, а затем выполнения кропотливой работы по модификации программы и ее отладке. Большой объем программного кода, многомодульность, многофункциональность затрудняет принятие решений по согласованному распределению данных и вычислений.

Для решения данной проблемы может использоваться метод инкрементального, или частичного распараллеливания. Идея этого метода заключается в том, что распараллеливанию подвергается не вся программа целиком, а ее части (области распараллеливания) — в них заводятся дополнительные экземпляры требуемых данных, производится распределение этих данных и соответствующих вычислений. Данные области могут быть построены на основе времени, полученного с помощью профилирования последовательной программы.

Для взаимодействия с теми частями программы, которые не подвергались распараллеливанию, используются операции копирования исходных (нераспределенных) данных в дополнительные (распределенные) данные и обратно. Конечно, операции копирования могут снизить или вообще ликвидировать эффект от распараллеливания. Кроме того, до полного распараллеливания (пока не все массивы программы будут распределены) программе будет требоваться память и для распределенных, и для нераспределенных массивов, что может ограничивать размер решаемой задачи.

К достоинствам инкрементального распараллеливания можно отнести:

1. Возможность распараллелить не всю программу, а ее времяемкие фрагменты, упрощает работу программиста, так как существенно сокращается объем кода программы для анализа и распараллеливания.
2. Отказ от распараллеливания сложных фрагментов программы позволяет с большей вероятностью найти хорошие решения для выделенных областей распараллеливания.
3. Найденные решения могут быть использованы в качестве подсказки при распараллеливании других частей программы на следующих этапах.

Данный метод был успешно применен для распараллеливания программы моделирования многокомпонентной фильтрации при разработке месторождений нефти и газа.

1.1. Многокомпонентная многофазная изотермическая фильтрация

Композиционные модели фильтрации используются при подробном моделировании залежей, содержащих легкие углеводороды (конденсат и газ) в том случае, когда необходимо тщательно описывать массообмен между фазами, либо когда пластовые флюиды содержат ценные неуглеводородные компоненты. Эти модели также часто используются для изучения методов увеличения нефтеотдачи при закачке газов высокого давления, азота, углекислого газа и других агентов.

Последовательная версия программы «Композит» для решения задач многокомпонентной многофазной изотермической фильтрации была разработана в НИИСИ РАН [5]. Параллельная версия программы была разработана на языке Fortran-DVMH специалистами ИПМ РАН.

Для определения времяемких фрагментов программы, требующих распараллеливания, использовалось профилирование. В табл. 1 показано время выполнения различных процедур программы «Композит».

Таблица 1

Время выполнения различных процедур программы «Композит»

Процедура	**Время, в секундах**
INDA	0,0872
INCON	0,0204
GRID	0,3002
DECL	0,3088
CONDIN	0,3032
INSATZ	3,0720
TRANSM	0,0331
TRANSMOD	0,0035
REMAPI	9,2835
REMAPR	25,9266
RESERVS	0,8419
WELLIN	2,4177
COEF	2,5344
SXYN	476,3081
CFPR	2,9180
PWELL	0,0661
LSOR	12,4683
CWELL	4,2935
COMPOZ	45,4745
SXYDEF	313,9402
Вся программа	**910,2972**

Данные были получены на суперкомпьютере К-100 [6] с использованием анализатора производительности, который входит в состав DVM-системы. Механизм интервалов, реализованный в анализаторе, позволяет получить временные характеристики выполнения

программы с различной степенью подробности (например, для каждого цикла, для каждой процедуры). Если не учитывать процедуры, в которых выполняется ввод/вывод — REMAPI и REMAPR, то основное время занимают процедуры: COMPOZ, SXYN, SXYDEF и LSOR. Эти процедуры и были распараллелены на первом этапе.

В табл. 2 показано время выполнения (в секундах) частично распараллеленной программы «Композит» на суперкомпьютере К-100 при использовании различного числа вычислительных узлов.

Таблица 2

Время выполнения частично распараллеленной программы на К-100

Процедура	**1 ядро**	**1 узел**	**2 узла**	**3 узла**	**4 узла**	**8 узлов**
INDA	0,0498	0,0372	0,0588	0,1095	0,1157	0,0879
INCON	0,0185	0,0243	0,0233	0,0191	0,0190	0,0249
GRID	0,3114	0,3201	0,3197	0,3197	0,3204	0,3164
DECL	0,3229	0,3225	0,3208	0,3200	0,3205	0,3367
CONDIN	0,3083	0,9399	0,9389	0,8835	0,8824	0,9561
INSATZ	3,1876	0,4715	0,4179	0,3903	0,3925	0,4238
TRANSM	0,0314	0,0307	0,0309	0,0315	0,0314	0,0316
TRANSMOD	0,0042	0,0026	0,0055	0,0027	0,0023	0,0060
REMAPI	9,0168	8,9658	8,9520	8,9647	8,9968	9,1267
REMAPR	26,7028	26,3997	26,3024	26,0393	25,9841	26,0864
RESERVS	0,8317	0,8338	0,8359	0,8340	0,8343	0,8663
WELLIN	2,4394	37,5341	37,5114	37,3192	37,2989	37,4203
COEF	2,7969	3,0714	3,0623	3,0590	3,0536	3,0149
SXYN	436,1784	39,6501	20,4180	14,1194	10,8128	10,9114
CFPR	2,786	2,9011	2,8950	2,8890	2,8966	2,8845
PWELL	0,0654	0,0676	0,0659	0,0656	0,0680	0,0656
LSOR	12,3597	2,2833	2,1432	2,3122	2,3465	2,7084
CWELL	4,2860	4,4267	4,4218	4,4199	4,4250	4,4209
COMPOZ	67,2481	18,0482	15,1819	14,3580	14,0307	14,3647
SXYDEF	329,4486	41,5750	35,3501	33,2744	32,4631	34,5857
Вся программа	**907,8578**	**161,9112**	**133,4614**	**123,7524**	**119,7692**	**122,4489**

При использовании 4-х вычислительных узлов К-100 программа ускоряется в 7,5 раз по сравнению с выполнением программы на одном ядре (время счета сокращается с 908 до 120 секунд). Дальнейшее увеличение числа используемых узлов приводит к замедлению программы.

Основная причина — рост накладных расходов на копирование данных из распределенных массивов в исходные нераспределенные массивы, которое выполняется после завершения выполнения параллельной версии процедуры (табл. 3). При увеличении числа используемых узлов, время выполнения операций копирования массивов начинает существенно превышать время выполнения самой процедуры (например, процедура SXYDEF на 1 узле выполняется 29,35 секунд, время копирований составляет 12 секунд; на 4-х узлах — время счета составляет 7,57 секунд, а время копирования уже 24,74 секунды). Избавиться от операций копирования позволяет лишь полное распараллеливание программы.

Таблица 3

Накладные расходы на копирование данных между исходными данными и их копиями

Интервал	1 ядро	1 узел	2 узла	3 узла	4 узла	8 узлов
Копирование до выполнения SXYN	5,7900	1,1122	0,5898	0,4751	0,3694	0,2567
Процедура SXYN	429,7672	38,1114	19,2957	12,8864	9,6858	9,6749
Копирование после выполнения SXYN	0,0700	0,2091	0,3135	0,4028	0,4483	0,4869
Копирование до выполнения LSOR	0,8583	0,3204	0,1805	0,1302	0,1062	0,0870
Процедура LSOR	11,3605	1,5575	1,2807	1,2538	1,3051	1,5619
Копирование после выполнения LSOR	0,1229	0,3321	0,6065	0,7633	0,8358	0,8358
Копирование до выполнения COMPOZ	18,9040	3,3755	1,8916	1,3577	1,3577	0,8558
Процедура COMPOZ	44,4443	8,7812	4,8451	3,4420	2,4817	2,2543
Копирование после выполнения COMPOZ	2,1775	5,2959	8,2695	9,2633	10,3155	11,2422
Копирование до выполнения SXYDEF	10,6708	1,8021	1,0436	0,7518	0,6409	0,4999
Процедура SXYDEF	312,9793	29,3578	15,0164	10,0026	7,5699	7,5578
Копирование после выполнения SXYDEF	4,9856	10,2023	19,2134	22,0208	24,0996	26,3115

Если распараллеливание основных процедур (2700 из 13844 строк) программы фактически не требовало изменения текста последовательной программы и было выполнено достаточно быстро, то полное распараллеливание потребовало серьезного изменения структуры программы: инлайн подстановки процедур (для фрагментов программы, выполняемых на ускорителях), преобразования операторов ввода/вывода (DVMH-модель накладывает некоторые ограничения на использование распределенных массивов в операторах ввода/вывода), перехода на динамические массивы (вместо их моделирования) и др. Найденные при распараллеливании основных процедур решения по распределению данных были использованы при распараллеливании других частей программы.

Полное распараллеливание программы позволяет запускать ее в различных режимах. Режим работы DVMH-программы, количество используемых нитей, графических процессоров задается при помощи переменных окружения и не требует перекомпиляции программы. Переменная DVMH_PPN задает количество MPI-процессов, которые будут запускаться на каждом узле суперкомпьютера. Переменная DVMH_NUM_THREADS определяет количество рабочих нитей, которые будут созданы для выполнения программы на ядрах ЦПУ, для каждого из процессов. Переменная DVMH_NUM_CUDAS определяет количество CUDA-ускорителей для каждого из процессов. Переменные DVMH_CPU_PERF и DVMH_CUDAS_PERF позволяют задать условную производительность (вес) ЦПУ- и CUDA-устройств. С учетом этой производительности будут распределяться данные между различными вычислительными устройствами узла. Если ка-

кая-то из перечисленных переменных не будет задана, то ее оптимальное значение определяет система поддержки выполнения DVMH-программ, которая обеспечивает эффективное использование всех ресурсов узла.

В табл. 4 и 5 показано время выполнения (в секундах) 100 итераций программы «Композит» в режиме MPI/OpenMP на суперкомпьютерах MVS-10P [7], K-100 и Ломоносов [8] при использовании от 1 до 8 вычислительных узлов для вариантов расчета при закачке в пласт сухого газа и газа, обогащенного промежуточными фракциями.

Таблица 4

Время выполнения 100 итераций параллельной программы для варианта расчета с закачкой сухого газа на разном числе узлов

Суперкомпьютер	**1 MPI**	**2 MPI**	**4 MPI**	**8 MPI**	**16 MPI**
MVS-10P	140,69	55,57	30,58	17,26	10,83
K-100	165,57	81,53	48,78	27,05	16,85
Ломоносов	290,06	142,00	75,63	42,04	25,42

Таблица 5

Время выполнения 100 итераций параллельной программы для варианта расчета с закачкой жирного газа на разном числе узлов

Суперкомпьютер	**1 MPI**	**2 MPI**	**4 MPI**	**8 MPI**	**16 MPI**
MVS-10P	143,27	56,30	31,22	17,94	11,50
K-100	165,96	82,28	44,49	25,21	17,14
Ломоносов	288,56	140,77	74,96	40,88	24,32

На каждом узле суперкомпьютера запускался 1 или 2 MPI-процесса, каждый из которых создавал 8 (для MVS-10P), 6 (для K-100) или 4 OpenMP-нити. Всего для расчетов использовалось до 128 ядер ЦПУ. При увеличении в 16 раз количества используемых ядер расчет ускоряется в 10–13 раз.

В табл. 6 показано время выполнения полного расчета программы «Композит» для сетки 261x261x6 на суперкомпьютерах K-100 и K-10 [6] при использовании графических ускорителей.

Таблица 6

Время выполнения программы «Композит» в различных режимах на одном узле суперкомпьютера K-100 и K-10

Система	**Последовательная программа**	**1 MPI**			**2 MPI**		
K-100	11698	12 нитей	1 GPU	12 нитей + 1 GPU	6 нитей	1 GPU	6 нитей + 1 GPU
		1152	2518	979	1179	1258	845
K-10	9961,8	16 нитей	1 GPU	16 нитей + 1 GPU	8 нитей	1 GPU	8 нитей + 1 GPU
		912	1745	816,2	945	969,8	742

Использование 1-го графического ускорителя позволяет ускорить выполнение программы в 4,6–5,7 раз по сравнению с выполнением на 1-ом ядре. Но при этом программа выполняется медленнее чем, при использовании всех ядер ЦПУ узла. Основная причина — не удается сбалансировать нагрузку нитей (вычисления в различных точках сетки, в которых находятся добывающие, нагнетательные скважины, существенно отличаются). Получить ускорение за счет использования графических ускорителей удается за счет совмещения вычислений на ЦПУ и ГПУ. В таком режиме программа ускоряется с 11698 секунд до 845 секунд на К-100; с 9961,8 секунд до 742 секунд на К-10. Таким образом, при использовании всех вычислительных устройств 1 узла суперкомпьютера ускорение выполнения программы составляет 13,84 раза для К-100 и 13,42 раза для К-10 по сравнению с выполнением программы на 1-м ядре.

2. Распараллеливание прикладных задач, использующих неструктурированные сетки

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.

В 2016 году были сформулированы основные предложения по расширению DVMH-модели для задач, использующих неструктурированные сетки [9]:

- введены новые правила для распределения данных (поэлементное — косвенное и производное);
- новые возможности для построения согласованных распределений (блочных или поэлементных);
- новый способ для задания произвольных по содержанию буферов удаленных элементов с эффективным однородным доступом к ним и обновлением;
- возможность реорганизации данных — оптимизации шаблона доступа к памяти путем изменения порядка хранения локальных элементов;
- сохранение быстрого доступа к распределенным массивам с помощью механизмов перехода на локальную индексацию.

На данный момент не все возможности, описанные выше, были реализованы в DVM-системе. Однако некоторые виды программ на нерегулярных сетках удается распараллелить имеющимися средствами уже сейчас.

2.1. Двумерная задача теплопроводности в шестиграннике

Рассмотрим двумерную задачу теплопроводности с постоянным, но разрывным коэффициентом в шестиграннике (рис. 1). Область состоит из двух материалов с различными коэффициентами температуропроводности.

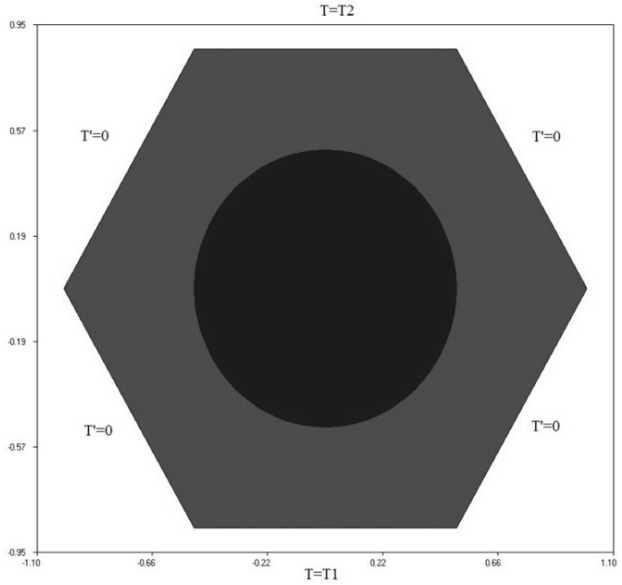

Рис. 1. Расчетная область и граничные условия

Рассмотрим фрагмент программы на языке Фортран, реализующий одну итерацию по времени для расчета по явной схеме (рис. 2):

```
      do i = 1, np2
        nn = ii(i)
        nb = npa(i)
        if (nb.ge.0) then
          s1 = FS(xp2(i),yp2(i),tv)
          s2 = 0d0
          do j = 1, nn
            j1 = jj(j,i)
            s2 = s2 + aa(j,i) * tt1(j1)
          enddo
          s0 = s1 + s2
          tt2(i) = tt1(i) + tau * s0
        else if (nb.eq.-1) then
          tt2(i) = vtemp1
        else if (nb.eq.-2) then
          tt2(i) = vtemp2
        endif
        s0 = (tt2(i) - tt1(i)) / tau
        gt = DMAX1(gt,DABS(s0))
      enddo
      do i = 1, np2
        tt1(i) = tt2(i)
      enddo
```

Рис. 2. Фрагмент Фортран-программы на нерегулярной сетке

Как видно из этого фрагмента, массивы величин `tt1` и `tt2` являются одномерными несмотря на то, что расчетная область двумерная. Также видно, что количество соседей у каждой ячейки не является константой, а читается из массива `ii`, а сами номера соседних ячеек читаются из массива `jj`, с которым связан также массив `aa`, описывающий каждую такую связь с соседом (в данном примере — просто коэффициент при суммировании). Существующая версия DVM-системы позволяет распараллелить такую программу при следующем ограничении: существует такое небольшое M, при котором все

соседи ячейки с номером n имеют номера, отличающиеся от n не более, чем на M. Ключевое слово здесь «небольшое», т.к. очевидно, что иначе в качестве такого M подойдет общее количество элементов сетки. Допустимая величина M — вещь субъективная, однако, учитывая описанный ниже способ распараллеливания, предлагается считать допустимым такое M, при котором M/N стремится к нулю при измельчении сетки (N — общее количество ячеек в сетке).

При распараллеливании в модели DVMH данной программы, были распределены:

- массивы искомых величин `tt1` и `tt2`;
- массивы описания элементов сетки `npa`, `xp2`, `yp2`;
- вспомогательные топологические массивы `ii` и `jj`;
- массив описания связей `aa`.

В результате, все вычисления производились только над распределенными данными с использованием только распределенных данных. Наиболее существенным при таком распараллеливании было вычисление необходимой ширины теневых граней для распределенного массива `tt1`.

Размеры теневых граней у `tt1` должны обеспечить присутствие (в локальной части или в теневой грани) на процессоре, владеющим элементом с индексом `i` также и всех элементов с индексами `jj(1:ii(i),i)`. Очевидно, что любое число M из приведенного выше ограничения является подходящей шириной теневой грани для массива `tt1`. Так как эта характеристика становится известной только во время выполнения, то был применен следующий подход:

1. Сначала делалось блочное распределение одномерных массивов без теневых граней.
2. Затем, анализируя результаты этого распределения и данные о связях, каждый процессор вычислял минимальную ширину теневой грани, покрывающую все его нужды (ссылки).
3. Затем производилось объединение этих требований со взятием максимального из них.

Учитывая, что теневые грани — это то, что будет пересылаться каждую итерацию по времени между процессорами, видно, что в этом подходе имеется сразу 2 места, ведущие к завышению пересылаемых объемов за счет ненужных данных.

Во-первых, ширина теневых граней в модели DVMH в настоящее время является универсальной величиной, не зависящей от номера процессора, т.е. нельзя одному процессору иметь ширину теневых граней 5, а другому — 10. Поэтому, чтобы удовлетворить требования всех процессоров, приходится задавать максимальную среди минимально необходимых отдельным процессорам ширину теневых граней.

Во-вторых, в теневые грани возможно включать только непрерывные участки распределенного массива, непосредственно примыкающие к локальной части процессора и невозможно включать отдельные элементы распределенных массивов. Это ограничение заставляет иметь теневую грань достаточно широкую для того, чтобы объять все элементы массива, на которые имеются ссылки при вычислении собственных элементов.
Однако в рассматриваемой задаче размеры ячеек сетки были примерно одинаковыми, а также отсутствовали «вытянутые» элементы (с диаметром, значительно отличающимся от среднего), что позволило удачным для распараллеливания в модели DVMH способом упорядочить сеточные элементы с целью минимизации включения лишних элементов в

теневые грани распределенных массивов. А именно, было применено геометрическое упорядочивание снизу–вверх, слева–направо.

Такой порядок нумерации сеточных элементов в сочетании с блочным распределением одномерных массивов величин (индексируемых номерами сеточных элементов) приводит к распределению расчетной области по процессорам слоями (горизонтальными полосами). Одну из расчетных сеток можно видеть на рис. 3, на котором изображен результат расчета (стационарное распределение температуры).

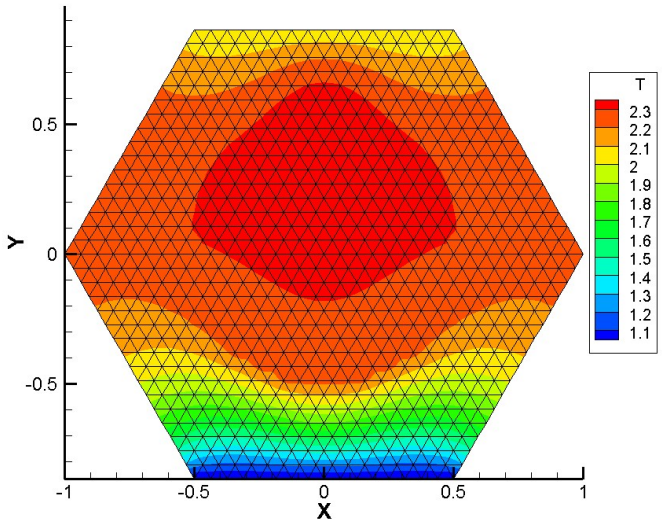

Рис. 3. Стационарное распределение температуры

В табл. 7 и 8 показаны время (в секундах) и ускорения параллельных версий программы для решения задачи теплопроводности (явная и неявная схемы) при использовании различного числа ЦПУ и ГПУ кластера К-100 для сетки 8 млн. узлов.

Таблица 7

Параллельное выполнение программы 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 8

Параллельное выполнение программы на ЦПУ

Схема	Посл	Параллельное выполнение, на разном количестве ядер ЦПУ											
		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

Полная реализация всех предложений по расширению DVMH-модели для задач, использующих неструктурированные сетки, должна повысить эффективность выполнения данных программ и улучшить полученные на данном этапе ускорения.

3. Дополнительное распараллеливание MPI-программ

В настоящее время, когда параллельные машины уже не одно десятилетие эксплуатируются для проведения расчетов, имеется множество программ, которые уже распараллелены на кластер, однако не позволяют эффективно использовать многоядерные процессоры и ускорители.

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 удается перевести исходную схему распределения данных и вычислений на DVM-язык. Одним из способов избавиться от обеих проблем стал новый режим работы DVM-системы, в котором ее возможности используются локально в каждом MPI-процессе для организации работы на многоядерных процессорах и ускорителях. Данный режим включается заданием специально созданной MPI-библиотеки при сборке DVM-системы.

Кроме такого режима, в компиляторы C-DVMH [10] и Fortran-DVMH [11] введено понятие нераспределенного параллельного цикла, для которого нет необходимости задавать отображение на распределенный массив. Например, трехмерный параллельный цикл может выглядеть так (рис. 4).

```
#pragma dvm parallel(3) reduction (max(eps))
for (int i = L1; i <= H1; i++)
    for (int j = L2; j <= H2; j++)
        for (int k = L3; k <= H3; k++)
...
!DVM$ PARALLEL(I,J,K) REDUCTION (MAX(EPS))
      DO I = L1,H1
         DO J = L2, H2
            DO K = L3, H3
...
```

Рис. 4. Новый режим распределения вычислений

По определению такой цикл выполняется всеми процессорами текущей многопроцессорной системы, но в описанном новом режиме 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 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 (В.А. Гасилов, А.С. Болдарев, ИПМ им. М.В. Келдыша РАН). 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 программы в модели DVMH вряд ли возможно без рассмотрения и модификации всей программы. Новые возможности позволили выполнить «локальное» распараллеливание вычислительноемких частей программы. Модификации подверглись лишь 3 из 39 тыс. строк программы.

В результате такого распараллеливания на 12 ядрах ЦПУ с использованием OpenMP-нитей 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.

4. Преимущества DVM-подхода для разработки параллельных программ

DVM-система автоматизирует процесс разработки параллельных программ.

Анализатор производительности позволяет определить времяемкие фрагменты или циклы программы, требующие распараллеливания. Механизм интервалов, реализованный в анализаторе, позволяет получить временные характеристики выполнения программы с различной степенью подробности. Пользователь DVM-системы может управлять разбиением программы на интервалы при ее компиляции. Для каждого такого фрагмента собирается информация, которая позволяет оценить потери, возникающие при распараллеливании программы, например, потери из-за выполнения межпроцессорных обменов, потери из-за перемещений данных из памяти ЦПУ в ГПУ, потери из-за простоев процессоров, на которых выполнение программы завершилось раньше, чем на остальных и.т.п. Данная

информация очень востребована при распараллеливании программы и важна при ее дальнейшей отладке эффективности.

DVM-отладчики автоматизируют процесс обнаружения ошибок, возникающих в процессе распараллеливания программы. Например, сравнительная отладка позволяет быстро обнаружить ошибки в распараллеливаемой программе, возникающие в результате переноса программы из ОС Windows в ОС Linux, или различия при выполнении программы на ЦПУ и ГПУ.

DVMH-модель параллельного программирования рассчитана на работу со структурированными и неструктурированными сетками, базируется на модели параллелизма по данным и модели параллелизма по управлению, что позволяет ее использовать для дополнительного распараллеливания в MPI-программах.

DVMH-компиляторы преобразуют входную программу в параллельную программу использующую стандартные технологии программирования MPI, OpenMP и CUDA, выполняя при этом различные анализы (например, режим автоматического определения приватных переменных для параллельных циклов) и оптимизации (например, реорганизацию данных для повышения эффективности доступа к глобальной памяти ГПУ).

Таким образом, использование DVM-системы позволяет существенно упростить процесс разработки параллельных программ для гетерогенных вычислительных кластеров.

Заключение

Новые возможности, которые появились в DVM-системе в последнее время, существенно расширили класс программ, которые могут быть распараллелены с помощью DVM-подхода.

Новые правила для распределения данных (косвенное, производное, согласованное) позволили использовать DVMH-модель для распараллеливания задач, использующих неструктурированные сетки.

Новый режим работы DVM-системы, в котором ее возможности используются локально в каждом MPI-процессе, позволяет отображать существующие MPI-программы на многоядерные процессоры и графические ускорители с использованием высокоуровневых DVM-спецификаций параллелизма.

Новые конструкции для распределения вычислений (например, нераспределенный параллельный цикл) существенно упрощают процесс инкрементального распараллеливания программ в DVM-модели.

В новой версии компилятора C-DVMH на базе Clang и инфраструктуры LLVM реализована начальная поддержка подмножества конструкций языка C++.

С использованием данных возможностей были разработаны параллельные версии программ:

- для моделирования многокомпонентной фильтрации при разработке месторождений нефти и газа,
- решения двумерной задачи теплопроводности в шестиграннике,
- решения по явной схеме системы гиперболических уравнений в двумерной области сложной формы.

Полученные DVMH-программы эффективно выполняются на гетерогенных кластерах, использующих многоядерные процессоры и графические ускорители.

В качестве следующего шага в развитии DVM-системы планируется расширить язык C-DVMH и компилятор с него для полноценной поддержки наиболее востребованных конструкций языка C++ (язык CDVMH++), а также исследовать подходы для интеграции модулей, разработанных с помощью DVM, в MPI-программы пользователей.

Работа подготовлена при поддержке программы президиума РАН №26 «Фундаментальные основы создания алгоритмов и программного обеспечения для перспективных сверхвысокопроизводительных вычислений».

Литература

1. Бахтин В.А., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н. Методы динамической настройки DVMH-программ на кластеры с ускорителями // Суперкомпьютерные дни в России: Труды международной конференции (Москва, 28–29 сентября 2015 г.). М.: Изд-во МГУ, 2015. С. 257–268.
2. Алексахин В.Ф., Бахтин В.А., Жукова О.Ф., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Савицкая О.А., Шуберт А.В. Распараллеливание на графические процессоры тестов NAS NPB 3.3.1 на языке Fortran DVMH // Вестник Уфимского государственного авиационного технического университета. 2015. Т. 19, №1(67). С. 240–250.
3. Алексахин В.Ф., Бахтин В.А., Жукова О.Ф., Колганов А.С., Крюков В.А., Островская И.П., Поддерюгина Н.В., Притула М.Н., Савицкая О.А. Распараллеливание на языке Fortran-DVMH для сопроцессора Intel Xeon Phi тестов NAS NPB3.3.1 // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (Екатеринбург, 31 марта–2 апреля 2015 г.). Челябинск: Издательский центр ЮУрГУ, 2015. С. 19–30.
4. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н., Смирнов А.А. Использование языка Fortran DVMH для решения задач гидродинамики на высокопроизводительных гибридных вычислительных системах // Вестник Южно-Уральского государственного университета, серия «Вычислительная математика и информатика». Челябинск: Издательский центр ЮУрГУ, 2013. Т. 2, № 3. С. 106–120. DOI: 10.14529/cmse130308.
5. Бахтин В.А., Королев А.В., Поддерюгина Н.В. Использование параллельных вычислений для моделирования многокомпонентной фильтрации при разработке месторождений нефти и газа // Международная конференция «Математика и информационные технологии в нефтегазовом комплексе»: тезисы докладов (Сургут, 16–20 мая 2016 г.). Сургут: ИЦ СурГУ, 2016. С. 164–166.
6. ИПМ им. М.В. Келдыша РАН. Вычислительные ресурсы. URL: http://www.kiam.ru/MVS/resourses/ (дата обращения: 18.05.2018).
7. Межведомственный Суперкомпьютерный Центр. Вычислительные системы. URL: http://www.jscc.ru/scomputers.html (дата обращения: 18.05.2018).
8. Суперкомпьютерный комплекс МГУ имени М.В. Ломоносова. URL: http://parallel.ru/cluster/ (дата обращения: 18.05.2018).
9. Бахтин В.А., Колганов А.С., Крюков В.А., Поддерюгина Н.В., Поляков С.В., Притула М.Н. Расширение DVMH-модели для работы с нерегулярными сетками // Па-

раллельные вычислительные технологии (ПаВТ'2016): труды международной научной конференции (Архангельск, 28 марта–1 апреля 2016 г.). Челябинск: Издательский центр ЮУрГУ, 2016. С. 757.

10. Язык C-DVMH. C-DVMH компилятор. Компиляция, выполнение и отладка CDVMH-программ. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-ru.pdf (дата обращения: 18.05.2018).
11. Язык Fortran-DVMH. Fortran-DVMH компилятор. Компиляция, выполнение и отладка DVMH-программ. URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-ru.pdf (дата обращения: 18.05.2018).

Бахтин Владимир Александрович, к.ф.-м.н., ведущий научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Захаров Дмитрий Александрович, младший научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Колганов Александр Сергеевич, младший научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Крюков Виктор Алексеевич, д.ф.-м.н., профессор, главный научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Поддерюгина Наталия Викторовна, к.ф.-м.н., старший научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

Притула Михаил Николаевич, старший научный сотрудник, Институт прикладной математики им. М.В. Келдыша РАН (Москва, Российская Федерация)

DOI: 10.14529/cmse190106

DEVELOPMENT OF PARALLEL APPLICATIONS USING DVM-SYSTEM

© 2019 V.A. Bakhtin, D.A. Zaharov, A.S. Kolganov, V.A. Krukov, N.V. Podderyugina, M.N. Pritula

Keldysh Institute of Applied Mathematics
(Miusskaya sq., 4, Moscow, 125047 Russia)
E-mail: bakhtin@keldysh.ru, s123-93@mail.ru, alexander.k.s@mail.ru, krukov@keldysh.ru, konov@keldysh.ru, pritula@keldysh.ru

Received: 21.06.2018

DVM-system was designed to create parallel programs of scientific-technical computations 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 by parallelism specifications, implemented as compiler directives. DVMH-model allows creating efficient parallel programs for heterogeneous computational clusters, the nodes of which use as computing devices not only universal multi-core processors but also can use attached accelerators (GPUs or Intel Xeon Phi coprocessors). This article describes the experience of parallelizing various application programs using DVM-system. The method of incremental or partial parallelization, the system's capabilities for working with unstructured grids, new tools for mapping MPI-programs to multi-core processors and accelerators are considered. The efficiency of parallel DVMH-programs on heterogeneous computing clusters K-10, K-100, Lomonosov and MVS-10P is investigated. The main advantages of DVM-approach for the development of parallel programs are described. The main features of DVM-system tools for performance analysis and functional debugging of parallel programs are presented. The directions for further development of DVM-system are determined.

Keywords: automation the development of parallel programs, DVM-system, parallelism specification directives, accelerator, GPU, coprocessor, Fortran, C.

FOR CITATION

Bakhtin V.A., Zaharov D.A., Kolganov A.S., Krukov V.A., Podderyugina N.V., Pritula M.N. Development of Parallel Applications Using DVM-system. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2019. vol. 8, no. 1. pp. 89–106. (in Russian) DOI: 10.14529/cmse190106.

References

1. Bakhtin V.A., Kolganov A.S., Krukov V.A., Podderugina N.V., Pritula M.N. Methods of Dynamic Tuning of DVMH Programs on Clusters with Accelerators. *Superkompjuternye dni v Rossii: Trudy mezhdunarodnoj konferencii (Moskva, 28–29 sentjabrja 2015)* [Russian Supercomputing Days: Proceedings of the International Scientific Conference (Moscow, Russia, September 28–29, 2015)]. Moscow, Publishing of the Moscow State University, 2015. pp. 257–268. (in Russian)
2. Aleksahin V.F., Bakhtin V.A., Kolganov A.S., Krukov V.A., Podderugina N.V., Pritula M.N., Savitskaya O.A., Shubert A.V., Zhukova O.F. GPU Parallelization of NAS NPB3.3.1 Benchmarks Using Fortran DVMH Programming Language. *Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta* [Bulletin of the Ufa State Aviation Technical University]. Ufa, 2015. vol. 19, no. 67. pp. 240–250. (in Russian)
3. Aleksahin V.F., Bakhtin V.A., Kolganov A.S., Krukov V.A., Ostrovskaya I.P., Podderugina N.V., Pritula M.N., Savitskaya O.A., Zhukova O.F. Parallelization of NAS NPB3.3.1 Tests on Fortran-DVMH for Intel Xeon Phi Coproccessor. *Parallelnye vychislitelnye tehnologii (PaVT'2015): trudy mezhdunarodnoj nauchnoj konferencii (Ekaterinburg, 31 marta–2 aprelja 2015)* [Parallel computational technologies 2015: Proceedings of the International Scientific Conference (Ekaterinburg, Russia, March, 31–April, 2, 2015)]. Chelyabinsk, Publishing of the South Ural State University, 2015. pp. 19–30. (in Russian)
4. Bakhtin V.A., Klinov M.S., Krukov V.A., Podderugina N.V., Pritula M.N., Smirnov A.A. Usage of Fortran DVMH Language for Solving Hydrodynamics Problems on Hybrid Computing Systems. *Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya "Vychislitelnaja matematika i informatika"* [Bulletin of the South Ural State University. Computation Mathematics and Software Engineering]. Chelyabinsk, Publishing of the South Ural State University, 2013. vol. 2, no. 3. pp. 106–120. (in Russian). DOI: 10.14529/cmse130308.
5. Bakhtin V.A., Korolev A.V., Podderugina N.V. Parallel Computations for Compositional Flow Simulation During Oil and Gas Fields Development. *Mezhdunarodnaja konferencija "Matematika i informacionnye tehnologii v neftegazovom komplekse": tezisy dokladov (Surgut, 16–20 maja 2016)* [Mathematics and Informational Technologies for Oil and Gas Industry: Abstracts of the International Scientific Conference (Surgut, Russia, May, 16–20, 2016)]. Surgut: Publishing center SurSU, 2016. pp. 164–166. (in Russian)
6. Keldysh Institute of Applied Mathematics. Computing Resources. Available at: http://www.kiam.ru/MVS/resourses/ (accessed: 18.05.2018).

7. Joint Supercomputer Center of the Russian Academy of Sciences. Available at: http://www.jscc.ru/scomputers.html (accessed: 18.05.2018).
8. Supercomputing Center of Lomonosov Moscow State University. Available at: http://parallel.ru/cluster/ (accessed: 18.05.2018).
9. Bakhtin V.A., Kolganov A.S., Krukov V.A., Podderugina N.V., Polyakov S.V., Pritula M.N. DVMH Model Extension for Operation with Irregular Grids. *Parallelnye vychislitelnye tehnologii (PaVT'2016): trudy mezhdunarodnoj nauchnoj konferencii (Arhangelsk, 28 marta–1 aprelja 2016)* [Parallel computational technologies 2015: Proceedings of the International Scientific Conference (Archangelsk, Russia, March, 28–April, 1, 2016)]. Chelyabinsk, Publishing of the South Ural State University, 2016. pp. 757. (in Russian)
10. C-DVMH Language, C-DVMH Compiler, Compilation, Execution and Debugging of DVMH Programs. Available at: http://dvm-system.org/static_data/docs/CDVMH-reference-en.pdf (accessed: 18.05.2018).
11. Fortran-DVMH Language. Fortran-DVMH Compiler. The Compilation, Execution and Debugging of DVMH-programs. Available at: http://dvm-system.org/static_data/docs/FDVMH-user-guide-en.pdf (accessed: 18.05.2018)