Опыт решения прикладных задач с использованием DVM-системы*

В.Ф. Алексахин[1], В.А. Бахтин[1,2], Д.А. Захаров[1], А.С. Колганов[1,2], А.В. Королев[3], В.А. Крюков[1,2], Н.В. Поддерюгина[1], М.Н. Притула[1]

ИПМ им. М.В. Келдыша РАН[1], МГУ им. М.В. Ломоносова[2], НИИСИ РАН[3]

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.

Ключевые слова: автоматизация разработки параллельных программ, DVM-система, ускоритель, ГПУ, Фортран, Си.

1. Введение

Система автоматизации разработки параллельных программ (DVM-система) существенно упрощает процесс разработки параллельных программ для гибридных вычислительных кластеров. Получаемые DVMH-программы без каких-либо изменений могут эффективно выполняться на кластерах различной архитектуры, использующих многоядерные универсальные процессоры (далее ЦПУ), графические ускорители (далее ГПУ) 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 рапараллеливать 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 описывается опыт разработки параллельных программ с использованием DVM-системы.

2. Распараллеливание прикладных задач, использующих структурированные сетки

Распараллеливание программы в модели DVMH можно разделить на следующие этапы:

а) Распределение данных (массивов) и вычислений (параллельных циклов).

б) Определение и спецификация удаленных данных (данные, которые вычисляются на одном процессоре, а используются на других).

* Работа поддержана грантами РФФИ № 16-07-01014, 16-07-01067, 16-37-00266 и 17-01-00820.

в) Определение вычислительных регионов (последовательности операторов, циклов) для выполнения на ускорителях.

г) 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 времен, полученных 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:

а) Возможность распараллелить не всю программу, а ее времяемкие фрагменты, упрощает работу программиста, так как существенно сокращается объем кода программы для анализа и распараллеливания.

б) Отказ от распараллеливания сложных фрагментов программы позволяет с большей вероятностью найти хорошие решения для выделенных областей распараллеливания.

в) 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.

2.1 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 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. На рис.1 показаны времена выполнения (в секундах) различных процедур программы.

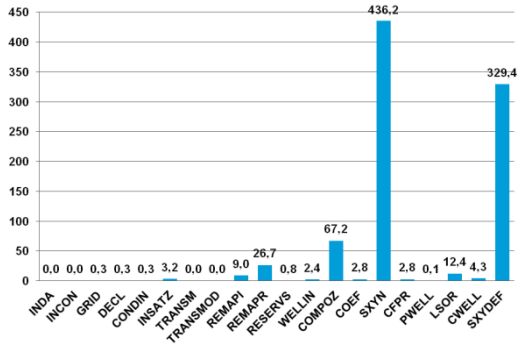

Рис. 1. Времена выполнения различных процедур программы

Данные времена были получены на суперкомпьютере K-100 [6] с использованием анализатора производительности, который входит в состав DVM-системы. Если не учитывать процедуры, в которых выполняется ввод/вывод (REMAPI и REMAPR), то основное время занимают процедуры: COMPOZ, SXYN, SXYDEF, LSOR. Эти процедуры и были распараллелены на первом этапе.

На рис. 2 показаны времена выполнения (в секундах) частично распараллеленной программы на суперкомпьютере K-100 при использовании различного числа вычислительных узлов.

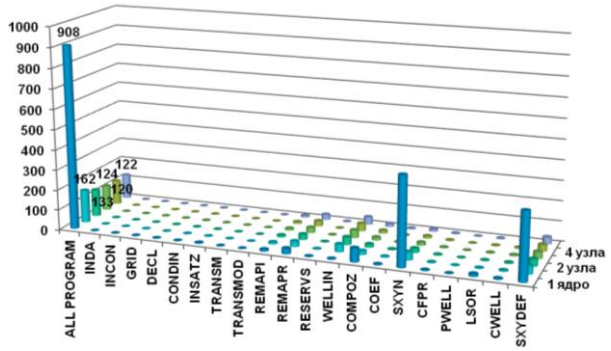

Рис. 2. Времена выполнения частично распараллеленной программы на K-100.

При использовании 4-х вычислительных узлов K-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 выполяется после завершения выполнения параллельной версии процедуры (рис. 3).

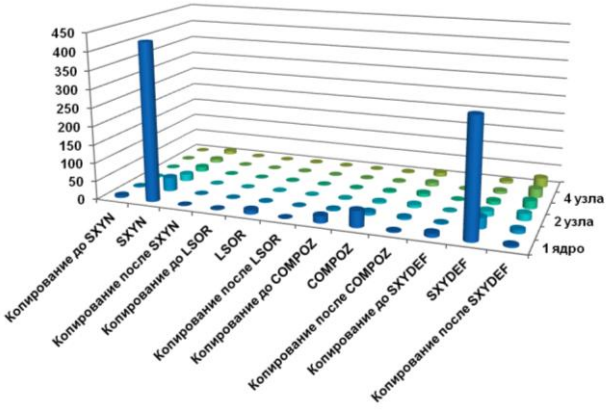

Рис. 3. Накладные расходы на копирование данных между исходными данными и их копиями.

При увеличении числа используемых узлов, время выполнения операций копирования массивов по завершении выполнения процедур начинает существенно превышать время выполне-

ния 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 24,64 секунды). Избавиться от операций копирования позволяет лишь полное распараллеливание программы.

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 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. На рис. 4-5 показаны времена выполнения 100 итераций программы 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.

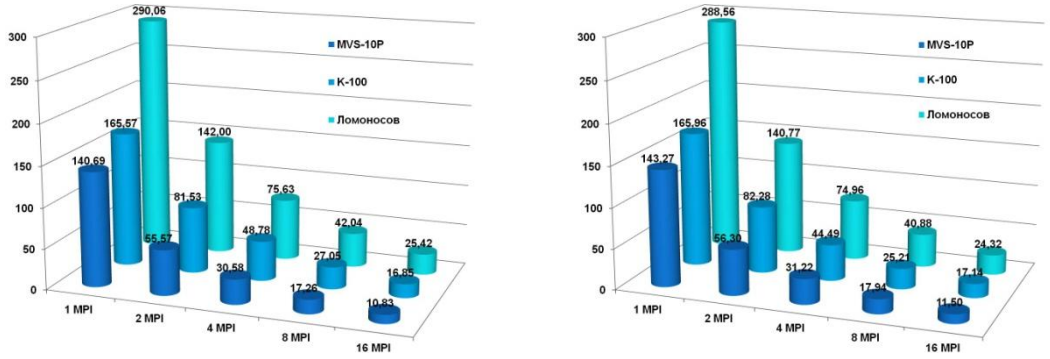

Рис. 4-5. Времена выполнения 100 итераций параллельной программы на разном числе узлов для вариантов расчета с закачкой сухого и жирного газа.

На каждом узле 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.

На рис. 6-7 показаны времена выполнения полного рассчета программы на суперкомпьютерах K-100 и K-10 [6] при использовании графических ускорителей. Синим цветом отмечены времена, полученные при использовании только ядер ЦПУ, желтым – графических ускорителей, зеленым – при совместном использовании ядер ЦПУ и ускорителей.

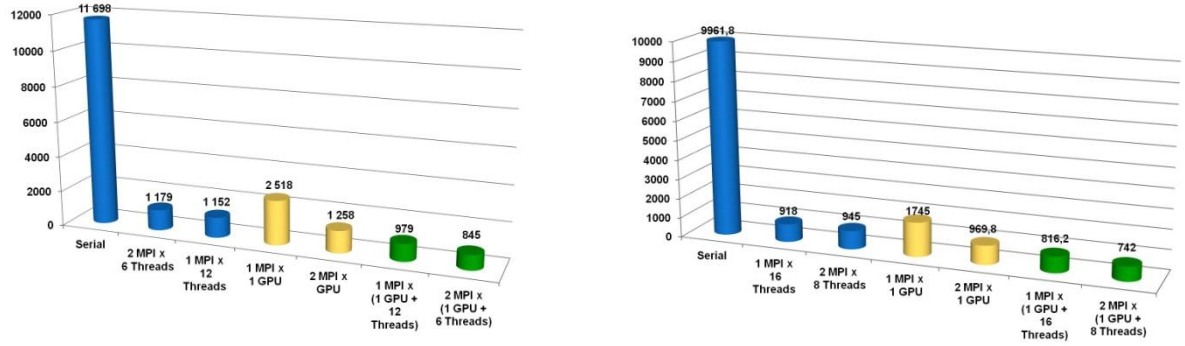

Рис. 6-7. Времена выполнения программы в различных режимах на суперкомпьютерах K-100 и K-10.

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 вы-

полняется медленнее чем, при использовании всех ядер ЦПУ узла. Основная причина – не удается сбаланисровать 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 K-100; с 9961,8 секунд до 742 секунд на K-10. Таким образом, при использовании всех вычислительных устройств 1 узла ускорение выполнения программы составляет 13,84 раза для K-100 и 13,42 раза для K-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. 8). Область состоит из двух материалов с различными коэффициентами температуропроводности.

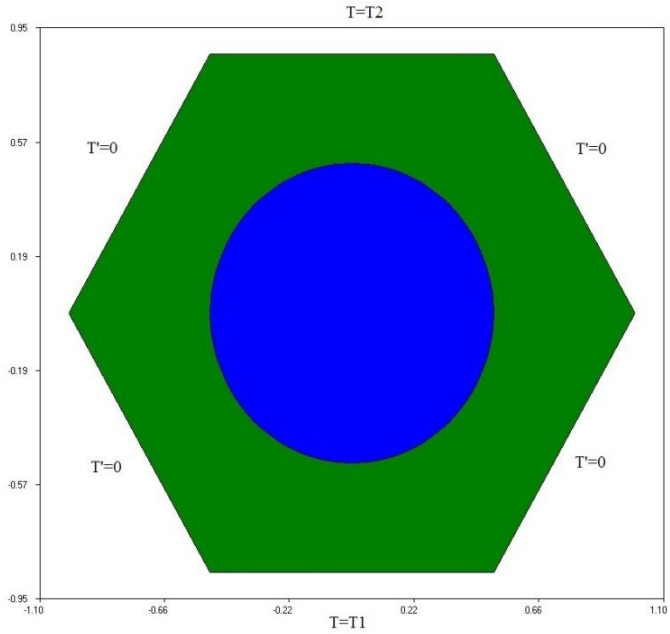

Рис. 8. Расчетная область и граничные условия.

Рассмотрим фрагмент программы на языке Фортран, реализующий одну итерацию по времени для расчета по явной схеме (рис. 9):

```
      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. 9. 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 ячеки 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. 10, на котором изображен результат расчета (стационарное распределение температуры).

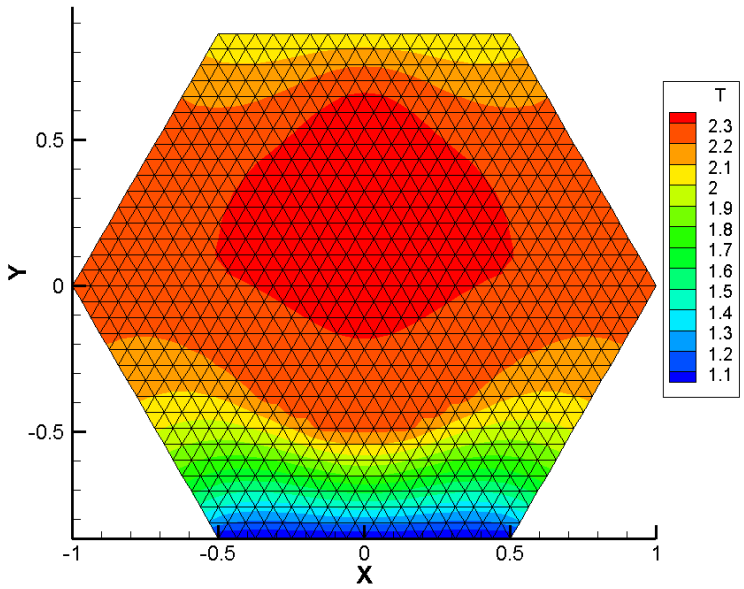

Рис. 10. Стационарное распределение температуры.

В таблицах 1 и 2 показаны времена (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 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

Полная реализация всех предложений по расширению DVMH-модели для задач, использующих неструктурированные сетки, должна повысить эффективность выполнения этих программ.

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. 11):

```
#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. 11. 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). 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 вычислительно-емких 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. Заключение

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 отстальных 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, позволяет осуществлять дополнительное распараллеливание MPI-программ.

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.

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 (28-29 сентября 2015 г., г. Москва), 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. Сборник трудов Международной научной конференции «Параллельные вычислительные технологии 2015», Челябинск: Издательский центр ЮУрГУ, 2015, С. 19-30.

4. Бахтин В.А., Клинов М.С., Крюков В.А., Поддерюгина Н.В., Притула М.Н. 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, Том 2, Выпуск 3, С. 106-120.
5. Бахтин В.А., Королев А.В., Поддерюгина Н.В. Использование параллельных вычислений для моделирования многокомпонентной фильтрации при разработке месторождений нефти и газа. Тезисы международной конференции «Математика и информационные технологии в нефтегазовом комплексе». Сургут: Издательский центр СурГУ, 2016, С. 164-166.
6. ИПМ им. М.В. Келдыша РАН. Вычислительные ресурсы. URL: http://www.kiam.ru/MVS/resourses/ (дата обращения: 14.04.2017).
7. Межведомственный Суперкомпьютерный Центр. Вычислительные системы. URL: http://www.jscc.ru/scomputers.html (дата обращения: 14.04.2017).
8. Суперкомпьютерный комплекс МГУ имени М.В. Ломоносова. URL: http://parallel.ru/cluster/ (дата обращения: 14.04.2017).
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. Сборник трудов Международной научной конференции «Параллельные вычислительные технологии 2016», 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: 14.04.2017).
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: 14.04.2017).

HPC Applications Experience using DVM-system

V.F. Aleksahin[1],V.A. Bakhtin[1,2], D.A. Zaharov[1], A.S. Kolganov[1,2], A.V. Korolev[3], V.A. Krukov[1,2], N.V. Podderyugina[1], M.N. Pritula[1]

Keldysh Institute of Applied Mathematics[1], Lomonosov Moscow State University[2], Scientific Research Institute for System Studies[3]

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 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 describes the experience of parallelizing various application programs using DVM-system.

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

References

1. Bakhtin V.A., Kolganov A.S., Krukov V.A., Podderugina N.V., Pritula M.N. Metody dinamicheskoj nastrojki DVMH-programm na klastery s uskoriteljami [Methods of dynamic tuning of DVMH programs on clusters with accelerators]. Superkomp'juternye dni v Rossii: Trudy mezhdunarodnoj konferencii (28-29 sentjabrja 2015, Moskva) [Russian Supercomputing Days: Proceedings of the International Scientific Conference (Moscow, Russia, September 28-29)]. Moscow, Publishing of the Moscow State University, 2015, P. 257–268.
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. Rasparallelivanie na graficheskie processory testov NAS NPB 3.3.1 na jazyke Fortran DVMH [GPU parallelization of NAS NPB3.3.1 benchmarks using Fortran DVMH programming language]. Bulletin of the Ufa State Aviation Technical University, Ufa, 2015, Vol. 19, No 67, P. 240-250.
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. Rasparallelivanie na jazyke Fortran-DVMH dlja soprocessora Intel Xeon Phi testov NAS NPB3.3.1 [Parallelization of NAS NPB3.3.1 tests on Fortran-DVMH for Intel Xeon Phi coproccessor]. Sbornik trudov Mezhdunarodnoj nauchnoj konferencii "Parallel'nye vychislitel'nye tehnologii 2015" [Parallel computational technologies 2015: Proceedings of the International Scientific Conference]. Chelyabinsk, Publishing of the South Ural State University, 2015, P. 19-30.
4. Bakhtin V.A., Klinov M.S., Krukov V.A., Podderugina N.V., Pritula M.N. Ispol'zovanie jazyka Fortran DVMH dlja reshenija zadach gidrodinamiki na vysokoproizvoditel'nyh gibridnyh vychislitel'nyh sistemah [Usage of Fortran DVMH language for solving hydrodynamics problems on hybrid computing systems]. 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, P. 106-120.
5. Bakhtin V.A., Korolev A.V., Podderugina N.V. Parallel computations for compositional flow simulation during oil and gas fields development // Abstracts of International conference "Mathematics and Informational Technologies for Oil and Gas Industry", Surgut: Publishing center SurSU, 2016, P. 166-168.
6. Keldysh Institute of Applied Mathematics. Computing resources. URL: http://www.kiam.ru/MVS/resourses/ (accessed:14.04.2017).
7. Joint Supercomputer Center of the Russian Academy of Sciences. URL: http://www.jscc.ru/scomputers.html (accessed:14.04.2017).
8. Supercomputing Center of Lomonosov Moscow State University. URL: http://parallel.ru/cluster/ (accessed:14.04.2017).
9. Bakhtin V.A., Kolganov A.S., Krukov V.A., Podderugina N.V., Polyakov S.V., Pritula M.N. Rasshirenie DVMH-modeli dlja raboty s nereguljarnymi setkami [DVMH model extension for operation with irregular grids]. Sbornik trudov Mezhdunarodnoj nauchnoj konferencii "Parallel'nye vychislitel'nye tehnologii 2016" [Parallel computational technologies 2016: Proceedings of the International Scientific Conference]. Chelyabinsk, Publishing of the South Ural State University, 2016, P. 757.
10. C-DVMH language, C-DVMH compiler, compilation, execution and debugging of DVMH programs. URL: http://dvm-system.org/static_data/docs/CDVMH-reference-en.pdf (accessed:14.04.2017).
11. Fortran-DVMH language. Fortran-DVMH compiler. The compilation, execution and debugging of DVMH-programs. URL: http://dvm-system.org/static_data/docs/FDVMH-user-guide-en.pdf (accessed:14.04.2017).