АВТОМАТИЧЕСКОЕ ОТОБРАЖЕНИЕ ФОРТРАН-ПРОГРАММ НА КЛАСТЕРЫ С УСКОРИТЕЛЯМИ

В.А. Бахтин, М.С. Клинов, А.С. Колганов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула

Федеральное государственное бюджетное учреждение науки Институт прикладной математики им. М.В. Келдыша Российской академии наук

В статье рассматриваются результаты использования системы САПФОР для распараллеливания последовательных программ на кластеры с графическими ускорителями, в том числе программ с регулярными зависимостями по данным. Система переводит программу на языке Fortran в программу на языке Fortran DVMH. Полученная программа запускается на кластере. Язык Fortran DVMH и компиляторы для него входят в состав DVM-системы. Приведены экспериментальные данные об эффективности выполнения программ на графических и универсальных процессорах кластера К-100.

Сложность программирования для кластеров с графическими ускорителями делает актуальными исследования по автоматизации программирования.

Изначально система САПФОР [1] была реализована для кластеров с универсальными процессорами, при этом в качестве промежуточного языка программирования использовался язык Fortran DVM. Для мультипроцессора система САПФОР позволяет получить программы на языке Fortran OpenMP. При реализации отображения на графические ускорители в системе САПФОР используется язык Fortran DVMH [2], который представляет собой расширение языка Fortran DVM.

В настоящее время исследуются возможности расширения языка Fortran DVMH для поддержки ускорителей Intel Xeon Phi. Такая преемственность промежуточных языков существенно упрощает разработку систем для отображения последовательных программ на новые архитектуры ЭВМ, при этом многие блоки системы находят новое применение и используются повторно.

В 2012 году были получены результаты распараллеливания при помощи системы САПФОР двух приложений из области гидродинамики на кластер с графическими ускорителями: двумерная и трехмерная задачи о течении несжимаемой жидкости или слабосжимаемого газа около прямоугольной выемки. В этих задачах в качестве математической модели используется гиперболический вариант квазигазодинамической системы [3].

В ходе данного исследования система САПФОР была опробована на двух приложениях из области лазерных технологий обработки материалов: двумерная и трехмерная задачи моделирования процессов плавления многокомпонентных порошков при селективном лазерном спекании на основе многокомпонентной и многофазной гидродинамической модели [4]. В этих задачах в порошке имеются поры, заполненные газом. В процессе плавления легкоплавкой компоненты порошка образуется жидкость. Будем называть в статье эти задачи Спекание2D и Спекание3D соответственно.

Помимо этого система САПФОР была развита для поддержки распараллеливания программ с регулярными зависимостями по данным между витками циклов и протестирована на примере алгоритма численного решения краевой задачи для уравнения Лапласа методом последовательной верхней релаксации. В этой программе на витках цикла есть зависимость от данных, вычисляемых на соседних витках. Выполнение таких циклов может быть реализовано в виде конвейера. Организация конвейера для графических ускорителей отличается от организации конвейера для универсальных процессоров, потому что ключевым моментом для ускорителей является обеспечение согласованного доступа к памяти устройства. Соответствующая поддержка таких циклов была сделана на уровне языка Fortran DVMH, в компиляторе с этого языка и в библиотеке функций системы поддержки выполнения DVMH-программ. Такому развитию системы была посвящена статья [5]. Без этого развития было бы невозможно осуществить отображение последовательных программ на кластер с графическими ускорителями в системе САПФОР. В этом плане отображение параллельных программ (на языке Fortran DVMH) на кластер с графическими ускорителями является частью отображения последовательных программ на этот кластер.

Для получения результатов распараллеливания этих программ потребовалось развитие системы САПФОР, которое было выполнено. Внесены изменения в блоки анализа, в блок построения вариантов распараллеливания на кластер с графическими ускорителями. Также были выполнены некоторые преобразования распараллеливаемых программ. Имеются в виду такие преобразования, после которых программа остается последовательной (без каких-либо указаний о ее распараллеливании), но которая уже может быть преобразована системой распараллеливания в эффективную параллельную программу. Более подробно преобразования будут рассмотрены далее.

На данный момент такие преобразования делает пользователь системы САПФОР, а не сама система. При этом подразумевается, что пользователь системы обладает навыками программирования на языке Fortran:

либо является автором распараллеливаемой программы, либо специалистом по ее преобразованию. В дальнейшем планируется, что подобные преобразования будут частично или полностью автоматизированы.

В состав системы САПФОР входит диалоговая оболочка и автоматически распараллеливающий компилятор, сокращенно АРК. Для данного исследования использовался как раз АРК. Показ причин, по которым не удавалось эффективно распараллелить исходную программу (без преобразований), не был развит для поддержки данных входных программ и не тестировался на них. В рамках данного исследования исследовались возможности системы распараллеливания по эффективному распараллеливанию этих программ как таковые. Улучшение диалоговой оболочки для этих и других программ из этого класса программ (похожих на этих) будет следующим этапом.

1. Распараллеливание приложений Спекание2D и Спекание3D при помощи автоматически распараллеливающего компилятора

Рассмотрим процесс распараллеливания.

Для приложения Спекание2D у программиста имеется последовательная Fortran-программа из 872 строк, которая подается на вход АРК. В процессе работы компилятора создается файл с информацией о ходе распараллеливания. Эта информация является достаточно подробной, но и в силу этого объемной. Она не представляется удобной для конечного пользователя. Данное исследование проводилось разработчиками системы САПФОР, как следствие они были способны разобраться в объемной диагностике процесса распараллеливания программы. По диагностике были определены причины, по которым циклы программы остались не распараллелены или распараллелены неэффективно. По этим причинам были определены фрагменты программы, которые требовалось изменить в последовательной программе, она была изменена и снова подана на вход АРК.

После изменения программа для приложения Спекание2D стала занимать 1007 строк.

После преобразования текст программы для Спекания3D возрос с 901 строки до 1555 строк.

В целом преобразования хоть и объемные, но не сложные. Они не требуют знания о распараллеливании программы, в отличие от распараллеливания программистом программы. Во многих местах преобразования аналогичные, за счет чего выполняются быстро, хоть и требуют определенного навыка. Полученная программа является последовательной, написана на том же языке, что и исходная, на Fortran. Корректность преобразований проверяется по мере их накопления путем сравнения результатов выполнения исходной и преобразованной программы. Это позволяет не ошибиться программисту на данном этапе.

Рассмотрим преобразования одного фрагмента из этих программ. Эти преобразования оказались самыми сложными из тех, с которыми сталкивались авторы статьи (не только на примере этих программ), при подготовке входных программ для АРК.

Речь идет о следующем фрагменте программы:

```
do j=1,Nz
do i=1,Nx
  do jj=0,1
  do ii=0,1
    iloc(1)=1-2*ii
    jloc(1)=0

    iloc(2)=0
    jloc(2)=1-2*jj

    i0=i-1+ii
    j0=j-1+jj
    do na=1,2
      i1=i0+iloc(na)
      j1=j0+jloc(na)

      bs(na,1)=x(i1,j1)-x(i0,j0)
      bs(na,2)=z(i1,j1)-z(i0,j0)
    end do
    …
    flx(i,j-1+jj)=flx(i,j-1+jj)-iloc(1)*fl(1)
    flz(i-1+ii,j)=flz(i-1+ii,j)-jloc(2)*fl(2)
  end do
  end do

end do
end do
```

Некоторые детали фрагмента были опущены (заменены на “…”).

С точки зрения распараллеливания во внешнем цикле (по i и по j) есть OUTPUT-зависимость (запись с разных витков в одну ячейку памяти). Ее можно устранить путем заведения дополнительного массива, который будет использоваться для переноса значений, на которые в данном случае надо изменить элемент массива. Дополнительный массив будет привязан своими индексами к значениям переменных цикла на каждом витке. Адресуя его элементы можно брать значения с конкретных витков.

В таком случае в данном цикле не будут записываться элементы целевого массива, а будет заполняться дополнительный массив. Запись в целевой будет сделана в последующих циклах для того, чтобы обеспечить такой же порядок суммирования в один элемент массива, который был в исходной (не преобразованной) программе. Например, заполнение дополнительного массива:

```
      tmp_arr(i,j,ii,jj,1)=iloc(1)*fl(1)
      tmp_arr(i,j,ii,jj,2)=jloc(2)*fl(2)
A запись в целевой массив:
do j=1,Nz
do i=1,Nx
  flx(i,j)=flx(i,j)-tmp_arr(i,j,0,1,1)
  flx(i,j)=flx(i,j)-tmp_arr(i,j,1,1,1)
end do
end do
```

Вторая особенность рассматриваемого фрагмента связана с косвенной индексацией элементов массивов на чтение. На самом деле выполняются операции чтения соседних элементов, но для системы распараллеливания это должно быть извлекаемым из текста программы. В данном случае это возможно через замену косвенной индексации на явную. Для этого заведены вспомогательные скалярные переменные, которые будут присвоены в начале каждого витка и использованы на нем. В скалярные переменные будут присвоены значения массивов с косвенной индексацией в зависимости от значений итерационных переменных циклов на витке. Например, для обращений x(i1,j1) будет использована переменная x_i1_j1:

```
if(na.eq.1) then
  if(ii.eq.0) then
    if(jj.eq.0) then
      x_i1_j1=x(i,j-1)
      …
```

Характеристики исходных и преобразованных программ приведены в таблице 1. Под программой с обозначением Fortran АРК подразумевается программа на языке Fortran, подаваемая на вход АРК. Процентное соотношение приведено относительно исходной Fortran-программы.

Таблица 1. Характеристики программ Спекание2D и Спекание3D

Программы	Характеристики	Fortran	Fortran АРК
Спекание2D	общее число строк	872	1007 (115%)
	добавлено строк	–	135 (15%)
	изменено строк	–	23 (2,6%)
Спекание3D	общее число строк	901	1555 (173%)
	добавлено строк	–	654 (73%)
	изменено строк	–	91 (10%)

После распараллеливания приложений с помощью АРК были получены тексты программ на языке Fortran OpenMP, Fortran DVM и Fortran DVMH. Характеристики полученных программ приведены в таблице 2.

В таблицах 3 и 4 приведены времена выполнения полученных версий программ на одном узле кластера К-100 [6] при использовании различного числа ядер и графических ускорителей(GPU). Для сравнения приведено время работы автоматически распараллеленной программы при помощи компилятора Intel (опция -parallel).

Таблица 2. Размер различных версий программ Спекание2D и Спекание3D, в строках

Программы	Fortran АРК	Fortran OpenMP	Fortran DVM	Fortran DVMH
Спекание2D	1007	1035	1068	1206
Спекание3D	1555	1617	1648	1845

Таблица 3. Времена выполнения программы Спекание2D (1000x1000, ITMAX=100) на кластере К-100, в секундах

	Последовательная	Intel -parallel	АРК OpenMP	АРК DVM	АРК DVMH
1 ядро	19.23	19.72	20.53	25.96	20.16
6 ядер	–	20.41	4.79	6.52	5.48
12 ядер	–	20.81	3.41	4.49	3.72
1 GPU	–	–	–	–	12.03
2 GPU	–	–	–	–	10.01
3 GPU	–	–	–	–	7.17

Таблица 4. Времена выполнения программы Спекание3D (100x100x100, ITMAX=100) на кластере К-100, в секундах

	Последовательная	Intel -parallel	АРК OpenMP	АРК DVM	АРК DVMH
1 ядро	53.01	58.54	50.03	128.75	56.56
6 ядер	–	51.05	24.11	25.59	13.61
12 ядер	–	50.46	23.80	13.41	8.46
1 GPU	–	–	–	–	6.43
2 GPU	–	–	–	–	4.94
3 GPU	–	–	–	–	4.21

Компилятор Intel не справляется с автоматическим распараллеливанием данных программ. Получаемые параллельные программы практически не ускоряются при увеличении числа используемых ядер. Основная проблема — приватные массивы в циклах (рабочие массивы, зависимость по которым не мешает выполнять цикл параллельно, если для каждого процесса/нити использовать свой экземпляр таких переменных). Установить факт, что тот или иной массив можно сделать приватным, достаточно сложная задача для статического анализатора (как правило, он указывает на наличие зависимости между витками цикла, что приводит к его последовательному выполнению). В системе САПФОР реализован механизм спецкомментариев для передачи информации о свойствах программы, извлечь которые на этапе анализа невозможно или очень трудоемко. Все приватные массивы были указаны в соответствующих спецкомментариях, что позволило получить более эффективные варианты параллельных программ.

На 1 ядре универсального процессора DVM-варианты программ выполняются медленнее остальных. Например, DVM-вариант программы Спекание3D выполняется в 2.4 раза дольше последовательного варианта. Причина — линеаризация распределенных между процессами массивов, выполняемая компилятором Fortran DVM. Память для элементов таких массивов выделяется системой поддержки выполнения DVM-программ. Для локальной секции массива на каждом процессоре память отводится в соответствии с форматом распределения данных и с учетом теневых граней. Все ссылки к элементам распределенных массивов вида ARRAY(I,J,K) заменяются компилятором на ссылки вида BASE(ARRAY_OFFSET+I+COEFF_ARRAY1*J+COEFF2*K), где BASE – это база, относительно которой адресуются все распределяемые массивы; ARRAY_OFFSET – смещение начала массива относительно базы; COEFF_ARRAY1, COEFF_ARRAY2 – коэффициенты адресации распределенного массива. Такая программа хуже распознается и оптимизируется стандартными Fortran компиляторами.

DVMH-версия программы лишена данного недостатка. Для каждого параллельного цикла компилятор с языка Fortran DVMH гененерирует CUDA-обработчик для вычислений на GPU, а также хост-обработчик для выполнения на универсальных ядрах процессора. Обработчик – подпрограмма, осуществляющая выполнение части параллельного цикла на конкретном устройстве. Распределенные массивы передаются в хост-обработчики как параметры — массивы, перенимающие размер (в формальном параметре вместо верхней границы последней размерности указывается звездочка “*”). Такой подход позволяет не линеаризовать обращения к массивам внутри хост-обработчиков. В результате время работы DVMH-программы на 1 ядре практически совпадает с временем работы последовательной программы.

OpenMP-вариант для приложения Спекание2D выполняется на 1 узле кластера быстрее других вариантов. DVMH-вариант на 1 GPU выполняется быстрее, чем на 1 ядре универсального процессора (например, последовательная программа), но на 3 GPU (1 узел кластера К-100) уступает 12 ядрам универсального процессора (также 1 узел кластера К-100).

Для Спекания3D OpenMP-вариант не самый лучший для ядер универсального процессора, а DVMH-вариант обгоняет на 3-х GPU 12 ядер универсального процессора, тем самым, является самым быстрым вариантом для случая использования 1 узла кластера К-100.

При увеличении размера задачи можно ожидать большего выигрыша от использования графических ускорителей вплоть до получения выигрыша на нескольких узлах К-100.

Для всех параллельных программ (полученных автоматически через АРК или через распараллеливающий компилятор от Intel) была проверена корректность работы путем сравнения результатов работы программ с последовательным вариантом. Относительная погрешность менее $1*10^{-9}$.

2. Распараллеливание программы, содержащей цикл с регулярными зависимостями по данным между витками при помощи автоматически распараллеливающего компилятора

В качестве такой программы взята программа, реализующая метод последовательной верхней релаксации (Successive over-relaxation — программа SOR).

Текст последовательной программы на языке Fortran занимает 48 строк. Без каких-либо изменений он подавался на вход АРК и распараллеливающему компилятору Intel (с опцией -parallel).

При помощи АРК были получены тексты программ на языках Fortran OpenMP, Fortran DVM, Fortran DVMH.

Времена выполнения программ на кластере К-100 приведены в таблице 5. В программе использовались массивы размера 800x800x800 и было выполнено 50 итераций.

Таблица 5. Времена выполнения программы SOR на кластере К-100, в секундах

	Последовательная	Intel -parallel	АРК OpenMP	АРК DVM	АРК DVMH
1 ядро	88.96	89.00	88.98	90.12	89.02
6 ядер	–	88.99	15.58	32.59	30.67
12 ядер	–	88.99	7.91	22.87	22.47
1 GPU	–	–	–	–	48.34
2 GPU	–	–	–	–	26.82
3 GPU	–	–	–	–	20.33

Intel-компилятор не ускорил выполнение программы SOR.

OpenMP-вариант программы выполняется на 1 узле кластера К-100 быстрее других вариантов.

DVMH-вариант программы на 1 GPU выполняется почти в 2 раза быстрее, чем лучший вариант на 1 ядре универсального процессора. Лучшее время для DVMH-версии программы дает использование 3-х GPU, по сравнению с использованием 12 ядер универсального процессора.

Заключение

Использование системы автоматизации распараллеливания упрощает процесс разработки программ для современных ЭВМ, многие из которых имеют различного вида вычислительные устройства в своем составе. Особенно в случае автоматизации преобразований. В настоящий момент над последовательными программами требуется выполнить определенные преобразования, которые препятствуют их распараллеливанию системой САПФОР. Эти преобразования не содержат вставок никаких указаний о распараллеливании. В этом плане программа остается последовательной, написанной на языке Fortran.

Получены результаты распараллеливания на кластер с графическими ускорителями двух приложений из области лазерных технологий обработки материалов (Спекание2D и 3D) и программы SOR, которая содержит цикл с регулярными зависимостями по данным между витками. Использование 1 GPU на приложении Спекание2D и SOR дает ускорение на кластере К-100 относительно использования 1 ядра универсального процессора. На приложении Спекание3D использование полученной с помощью системы САПФОР программы на языке Fortran DVMH на графических ускорителях позволяет получить наилучшее время выполнения на 1 узле кластера К-100 относительно вариантов программ, полученных с помощью системы САПФОР на языках Fortran OpenMP, Fortran DVM или полученных компилятором Intel программ в модели OpenMP.

В настоящий момент производится улучшение системы САПФОР для распараллеливания реальных приложений, содержащих циклы с регулярными зависимостями по данным: тесты NAS Parallel Benchmarks LU, BT, SP.

Исследование выполнено при финансовой поддержке грантов РФФИ №13-07-00580, 14-01-00109 и Программ фундаментальных исследований президиума РАН №15, №16 и №18.

ЛИТЕРАТУРА:

1. В.А. Бахтин, И.Г. Бородич, Н.А. Катаев, М.С. Клинов, Н.В. Ковалева, В.А. Крюков, Н.В. Поддерюгина. “Диалог с программистом в системе автоматизации распараллеливания САПФОР” // Вестник Нижегородского университета им. Н.И. Лобачевского, Н. Новгород: Изд-во ННГУ, 2012, №5 (2), С. 242-245
2. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. “Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами” // Вестник Южно-Уральского государственного университета, 2012, №18, С. 82-92
3. В.А. Бахтин, И.Г. Бородич, Н.А. Катаев, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. “Распараллеливание с помощью DVM-системы некоторых приложений гидродинамики для кластеров с графическими процессорами” // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск), М.: Изд-во МГУ, 2012, С. 444-450
4. В.Г. Низьев, А.В. Колдоба, Ф.Х. Мирзаде, В.Я. Панченко, Ю.А. Повещенко, М.В. Попов, “Численное моделирование плавления двухкомпонентных порошков при лазерном спекании” // Математическое моделирование, 2011, Т. 23, №4, С. 90-102
5. В.Ф. Алексахин, В.А. Бахтин, О.Ф. Жукова, А.С. Колганов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, О.А. Савицкая, А.В. Шуберт. “Распараллеливание на графические процессоры тестов NAS NPB 3.3.1 на языке Fortran DVMH” // Параллельные вычислительные технологии (ПаВТ’2014): труды международной научной конференции (1-3 апреля 2014 г., г. Ростов-на-Дону), Челябинск: Издательский центр ЮУрГУ, 2014, С. 30-41
6. Гибридный вычислительный кластер К-100. URL: http://www.kiam.ru/MVS/resourses/k100.html (дата обращения 31.05.2014)