Средства поддержки параллельных вычислений в стандартах языка Фортран

А.М. Горелик

Аннотация. Анализируются средства поддержки параллельных вычислений в стандартах языка Фортран (в Фортране 90/95/2003/2008).
Действующий стандарт Фортрана (Фортран 2008) содержит новые средства поддержки параллельных вычислений – coarrays (комассивы). Coarray черты были известны ранее как Co-array Fortran – расширение Фортрана 95. Приводится описание этих средств параллельности в Фортране 2008.

Ключевые слова: параллельные вычисления, Фортран, комассивы.

Введение

Современный Фортран – это язык программирования, ориентированный на решение научно-технических задач, которые требуют большого объема вычислений. Особенно актуальным является применение Фортрана при решении крупномасштабных вычислительных задач с использованием современных супер-ЭВМ с параллельной архитектурой. Решение таких задач требуется в различных сферах фундаментальных научных исследований и во многих прикладных областях.

Фортран постоянно развивается и совершенствуется в соответствии с развитием вычислительной техники, языков и технологий программирования.

Язык Фортран несколько раз подвергался стандартизации в рамках ANSI и ISO (Фортран 66, Фортран 77, Фортран 90, Фортран 95, Фортран 2003 и Фортран 2008 [1–13]). Стандартизация языков программирования создает предпосылки для повышения мобильности программного обеспечения. Действующий стандарт Фортран 2008 был принят в 2010 году.

Средства параллельного программирования впервые были введены в стандарт языка в Фортране 90. Необходимость их введения была вызвана появлением в компьютерах аппаратных средств векторной обработки. Эти свойства аппаратуры, в свою очередь, формировались на основе требований решаемых задач. Для таких компьютеров разрабатывались различные средства, включая и расширения Фортрана, однако использование нестандартных средств снижало мобильность программ; это привело к необходимости унифицировать подобные средства в стандарте языка Фортран.

В Фортран 90 были введены средства явной спецификации векторных операций. В Фортране 95 средства поддержки параллельности получили дальнейшее развитие.

Хотя Фортран 90/95/2003 содержит некоторые средства поддержки параллельности, для параллельного программирования используются специальные библиотеки или языки – расширения Фортрана (MPI, HPF, OpenMP, Co-Array Fortran и др.)

В настоящее время средства, поддерживающие параллельную обработку для многопроцессорной и многоядерной архитектуры (coarrays), введены непосредственно в стандарт языка – Фортран 2008 [1, 7, 13].

Coarray черты были известны ранее как Co-array Fortran – расширение Фортрана 95.

1. Средства поддержки параллельности в Фортране 90/95/2003

1.1. Перечень средств

В Фортране 90/95/2003 имеются следующие средства полезные для параллельного программирования.

- Операции над массивами и секциями массивов как над целыми объектами на поэлементной основе и присваивания массивам.
- Условное присваивание массивам под управлением логической маски (оператор where и конструкция where – end where).
- Оператор и конструкция forall (начиная с Фортрана 95).
- Спецификация процедур без побочного эффекта (начиная с Фортрана 95).
- Поэлементные процедуры, аргументами которых могут быть как скаляры, так и массивы.
- Большой набор встроенных процедур для работы с массивами.

Ниже рассматривается, как перечисленные черты влияют на параллельность.

1.2. Операции над массивами и секциями массивов и присваивания массивам

Возможность работы с массивами и секциями массивов как с целыми объектами на поэлементной основе фактически специфицирует параллелизм действий над элементами массива или нескольких массивов.

Пример

```
real :: x(10, 10), y(10, 10), x1(10,10), y1(40)
. . .
x = y + x1 / 2.5
y1(1:10) = y1(10:1:-1)
```

Поскольку операции и присваивания для разных элементов массивов могут быть выполнены в произвольном порядке, они могут выполняться параллельно (если реализация поддерживает параллельные вычисления), т.е. это позволяет компилятору сгенерировать эффективный код с учетом особенностей аппаратуры.

1.3. Условное присваивание массиву под управлением логической маски

Оператор и конструкция присваивания под управлением логической маски (where, where – end where) позволяют выполнить вычисление выражений и присваивания значений не для всех элементов массива, а только для тех, которые удовлетворяют некоторому условию.

Пример

```
real :: a(100), w(100), y(100)
. . .
! Оператор where
where (a < 0.0) a = 0.0
! Пример конструкции where
where (w /= y)
    a = (w + y) / (w – y)
elsewhere
    a = 0.0
end where
```

В этом случае также вычисления и присваивания для различных элементов массивов могут выполняться в произвольном порядке, и потому выполнение может быть параллельным.

1.4. Оператор и конструкция forall

Оператор и конструкция forall обеспечивают множественное присваивание массивам, но в несколько иной форме.

Конструкция forall функционально похожа на do-цикл, но выполняется иначе. Итерации do-цикла выполняются последовательно, а в forall – сначала вычисляются правые части операторов присваивания для всех значений индексов, а затем производятся присваивания.

Пример

```
. . .
! Присваивание диагонали матрицы matr
      forall (i = 1:n) matr (i, i) = b( i )
!  Пример конструкции forall с маской
! Маска позволяет избежать ошибки
      forall (i = 1:n, y( i ) .ne. 0.0)
            x( i ) = 1.0 / y( i )
      end forall
```

Выполнение (для различных значений индексов) может быть параллельным, хотя результат не зависит от того, поддерживает ли компилятор параллельное выполнение forall. (Последнее замечание относится и к другим средствам). Таким образом, оператор и конструкция forall позволяют специфицировать параллельный цикл.

1.5. Спецификация процедур без побочного эффекта

При написании параллельных программ серьезной проблемой является такое свойство программы как детерминизм. Возможны ситуации, при которых для одной и той же вычислительной среды, одна и та же программа может вырабатывать разные результаты.

Одной из причин недетерминизма является использование в выражениях функций с побочным эффектом. Полезной чертой для использования в параллельных программах является спецификация pure (введенная в Фортране 95), которая используется для процедур без побочного эффекта.

Использование pure-процедур является одной из мер предотвращения недетерминизма в параллельных программах. Вызов такой процедуры можно использовать в тех случаях, где возможна параллельная обработка; в частности, можно вызвать такую процедуру в операторе присваивания, входящем в конструкцию forall, без таких нежелательных последствий как недетерминизм.

1.6. Поэлементные процедуры

Многие процедуры, называемые поэлементными (elemental), можно применять не только к скалярам, но и к массивам; при этом такие процедуры выполняются поэлементно для соответствующих элементов.

Поскольку результат выполнения поэлементных процедур не зависит от порядка выбора элементов массивов, они могут выполняться параллельно.

Пример

```
real  :: x(10, 10),  y(10, 10), z(10, 10)
. . .
x = y + sqrt (z)
```

Начиная с Фортрана 95, разрешено использовать не только встроенные поэлементные процедуры (как в Фортране 90), но также поэлементные процедуры, определяемые в программе.

1.7. Встроенные процедуры для работы с массивами

В языке имеется большой набор встроенных процедур для работы с массивами. Это часто используемые математические функции, которые входят в состав самого языка, а не программируются каждым пользователем заново, что позволяет эффективно использовать особенности архитектуры, включая и параллельное выполнение.

Имеются следующие группы функций: функции редукции, функции умножения векторов и матриц, функция транспонирования матрицы, функция слияния массивов под управлением маски, функции упаковки и распаковки массивов, функция конструирования массива добавлением копий из элементов исходного массива, функция изменения конфигурации массива, функции сдвига, функции определения положения в массиве.

Поскольку встроенные процедуры реализуются компилятором, реализация ориентируется на конкретную архитектуру, что позволяет сгенерировать эффективный код с учетом особенностей аппаратуры, в том числе обеспечивают параллельное выполнение.

1.8. Другие черты, влияющие на параллельность

Помимо указанных выше средств поддержки параллельности, которые имеются непосредственно в современных стандартах Фортрана, имеются и другие черты, косвенно влияющие на параллельность. Современный Фортран (в отличие от предшественников) позволяет разрабатывать программы лучше структурированные, более наглядные; такие программы легче распараллелить.

Если программа плохо структурирована, не наглядная, – сложно разобраться в общей логической схеме программы; такую программу трудно, а часто невозможно распараллелить, даже если используемый алгоритм допускает распараллеливание. Это не зависит от того, применяется ли автоматическое распараллеливание или используются специальные средства реализации параллельности.

Кроме того, многие архаизмы могут серьезно затруднить или сделать невозможной распараллеливание программы. Много проблем для распараллеливания возникает при использовании различных средств связи по памяти (операторов common, equivalence и других средств).

Таким образом, программа, написанная на современном Фортране и не использующая устаревшие черты, лучше поддается распараллеливанию, чем программа, написанная на “старом” Фортране.

Для многопроцессорных параллельных компьютеров разработаны специальные системы параллельного программирования (OpenMP, HPF, MPI, Co-Array Fortran и др.), которые являются фактически расширениями современного Фортрана, они существенно используют современные средства языка.

2. Новые средства поддержки параллельности в Фортране 2008

2.1. Общая информация

Новый стандарт языка Фортран (Фортран 2008) сохранил преемственность с предыдущими стандартами; таким образом, описанные выше средства сохранены и в Фортране 2008.

В Фортран 2008 добавлена конструкция цикла (с заголовком DO CONCURRENT), в котором нет зависимостей по данным между итерациями, что позволяет выполнять итерации цикла в произвольном порядке, или потенциально параллельно.

Основным же новшеством этого стандарта являются средства поддержки параллельных вычислений – coarrays (комассивы). Coarray черты были известны ранее как Co-array Fortran – расширение Фортрана 95, а еще раньше как F—.

В Фортран 2008 включены только базовые черты (ядро). В настоящее время разрабатываются расширения, которые будут окончательно определены в Техническом отчете, имеющем статус стандарта.

Ниже приводится краткое описание этих средств.

Полное описание – в [1, 7].

2.2. Модель параллелизма

Средства, введенные в Фортран 2008, реализуют модель SPMD (Single Program/ Multiple Data – одна программа множественный поток данных).

Программа, содержащая новые средства, интерпретируется так, как если бы она была сдублирована фиксированное число раз, и все копии выполнялись асинхронно. Каждая копия имеет свой собственный набор локальных объектов данных и называется экземпляром или образом (image).

В язык введены средства доступа к глобальным данным, средства синхронизации экземпляров и встроенные процедуры, среди них функции, возвращающие количество экземпляров, номер текущего экземпляра и др.

Число экземпляров может быть равно числу физических процессоров, но может быть больше или меньше. Экземпляры нумеруются от 1 до num_images ( ).

Новые средства могут быть реализованы на компьютерах с различной параллельной архитектурой.

2.3. Данные

Для доступа к данным из разных копий расширен синтаксис массивов Фортрана. Добавлен новый объект (распределенный массив), называемый комассив (coarray). В объявлении такого объекта добавляются квадратные скобки.

Например, объявление

real, dimension (1000), codimension [*] :: x, y

означает, что каждый (об этом говорит символ *) экземпляр программы имеет два вещественных комассива x и y размером 1000.

Ссылка на массив без квадратных скобок относится к локальным массивам. Квадратные скобки используются для доступа к массивам из другого экземпляра; там, где есть квадратные скобки, включается коммуникация между экземплярами.

Например, если для объявленных выше массивов x и y экземпляр выполняет оператор

x(:) = y (:) [m]

комассив y из экземпляра m копируется в локальный комассив x выполняемого экземпляра. Термины для комассивов снабжаются префиксом “ко” (коиндексы, кограницы и др.).

Число индексов плюс коиндексов одного массива не должно быть больше 15. Комассивы не допускаются в операторах common и equivalence.

2.4. Синхронизация

Каждый экземпляр выполняется независимо от других. Для синхронизации в языке введены следующие операторы:

sinc all
sinc images
sinc memory
lock
unlock
и конструкция
critical
. . .
end critical

Ниже эти операторы рассматриваются более подробно.

2.4.1. Оператор синхронизации всех экземпляров

sync all – барьерная синхронизация всех экземпляров.

Выполнение экземпляра приостанавливается до тех пор, пока на барьере не окажутся все экземпляры, т.е. пока все не выполнят оператор sync all.

Пример

```
! Чтение в экземпляре 1 и рассылка всем! другим экземплярам
real :: x [*]
  . . .
sync all
if (this_image ( ) ==1) then
read (*,*) x
  do j = 2, num_images ( )
     x [ j ] = x
  end do
end if
sync all
. . .
```

В приведенном фрагменте первый sync all гарантирует, что экземпляр с номером 1 не мешает другим экземплярам использовать до этого момента переменную x. Второй sync all гарантирует, что другой экземпляр не получит новое значение переменной x до его установки в экземпляре с номером 1.

2.4.2. Оператор синхронизации группы экземпляров

sync images (< *список номеров экземпляров* >) – синхронизация группы экземпляров.

Синхронизация экземпляра с другими экземплярами, номера которых перечислены в списке. Для указания всех экземпляров используется символ *.

Пример

```
! Синхронизация  экземпляра с номером 1 ! со всеми другими
if (this_image ( ) ==1) then
! Вычисление coarray данных, необходимых ! другим экземплярам
  sync images (*)
else
  sync images (1)
!  Использование данных, вычисленных в ! экземпляре 1
end if
```

2.4.3. Конструкция критические секции

Эта конструкция используется в тех ситуациях, когда необходимо чтобы фрагмент программы выполнялся в каждый момент времени только одним экземпляром.

Конструкция имеет вид:

```
critical
  ! Операторы, выполняемые в каждый
  ! момент времени ! только одним экземпляром
end critical
```

Пример

```
critical
  global_counter[1] = global_counter[1] + 1
  print *, global_counter
. . .
end critical
```

В этом фрагменте глобальный счетчик (комассив) поочередно (последовательно) увеличивается на первом экземпляре всеми другими экземплярами.

2.4.4. Операторы блокировки и разблокировки (замки)

lock (< *переменная* >)
unlock (< *переменная* >)

Эти операторы обеспечивают механизм управления доступом к данным, которые определяются или на которые ссылаются более чем один экземпляр.

Поскольку различные экземпляры могут оперировать с одними и теми же данными, для обеспечения корректности доступа к данным и их корректной модификации используются

операторы блокировки, обеспечивающие решение проблем корректности доступа, записи и обновления данных.

Переменная в этих операторах должна быть производного типа lock_type.

Этот тип определяется во встроенном модуле iso_fortran_env и имеет приватные компоненты. Переменная может иметь значение locked или unlocked. Начальное значение – unlocked. Изменить значение переменной можно только при выполнении оператора lock или unlock.

2.4.5. Дополнительные спецификаторы

Операторы синхронизации sync all, sync images, sync memory, lock и unlock имеют необязательные спецификаторы stat= и errmsg=. Эти спецификаторы играют ту же роль для этих операторов, как и для операторов allocate и deallocate.

2.5. Встроенные процедуры

В Фортран 2008 добавлены новые встроенные процедуры, среди них функции, возвращающие количество экземпляров, номер текущего экземпляра, верхние и нижние кограницы комассива и др.

Ниже приведен перечень встроенных функций и подпрограмм для реализации новых средств параллельных вычислений. Квадратные скобки используются для необязательных аргументов.

num_images ( ) – функция возвращает число экземпляров

this image ( ) – функция возвращает индекс вызывающего экземпляра

this_image (coarray [, dim]) – функция возвращает набор коиндексов комассива для вызывающего экземпляра

image_index (coarray, sub) – функция выполняет конвертирование коиндексов в индекс экземпляра

co_lbound (coarray [, dim] [, kind]) – функция возвращает значение нижней кограницы комассива

co_ubound (coarray [, dim] [, kind] – функция возвращает значение верхней кограницы комассива

atomic_define (atom, value) – подпрограмма, определяет значение переменной атомарно

atomic_ref (value, atom) – подпрограмма, ссылка на переменную атомарно.

Примечание. Необязательный аргумент dim – это номер измерения комассива (коранг); необязательный аргумент kind – это параметр типа.

Пример:

```
program example
  implicit none
  integer :: k
  character (len=20) :: name [*]
  if (this_image ( ) ==1) then
    print *, ‘enter your name : ‘
    read (*, ‘(a)’ ) name
    do k = 2, num_images ( )
      name [k]  = name
    end do
  end if
  sync all
  print *, name, ‘from image’, this_image ( )
end program example
```

Заключение

В статье рассмотрены средства поддержки параллельных вычислений, которые определены во всех стандартах языка Фортран, начиная с Фортрана 90.

Средства явной спецификации векторных операций и некоторые другие черты параллельности, специфицированные до Фортрана 2008, реализованы практически на всех современных компьютерах для различных платформ.

В стандарте Фортран 2008, который принят в 2010 году, введены средства параллельности, называемые coarrays. Модель параллельности – разделенное глобальное адресное пространство. Программа интерпретируется так, как если бы она была сдублирована фиксированное число раз и все копии (images) выполнялись асинхронно. В языке определены средства доступа к данным другого экземпляра программы, средства синхронизации для управления взаимодействием экземпляров. В некоторых компиляторах эти средства уже реализованы (например, Intel, Cray).

В настоящее время разрабатывается проект дальнейшего развития параллельных средств в Фортране [14]. Этот проект включает такие возможности как группы экземпляров (team of

images), коллективные операции передачи данных, возможность определения топологий и др. Этот проект войдет в следующую ревизию стандарта.

Литература

1. ISO/IEC 1539-1: 2010. Information technology – Programming languages – Fortran – Part1: Base Language.
2. ISO/IEC 1539-1: 2004. Information technology – Programming languages – Fortran – Part1: Base Language.
3. ISO/IEC 1539-1: 1997. Information technology – Programming languages – Fortran – Part1: Base Language.
4. ISO/IEC 1539: 1991(E). Information technology – Programming languages – Fortran.
5. ANSI X3.9-1966. USA Standard FORTRAN.
6. ANSI X3.9-1978. American National Standard – Programming Language FORTRAN. (ISO 1539-1980).
7. J.Reid. Coarrays in the next Fortran Standard. ISO/IEC JTC1/SC22/WG5 №1762-2.
8. Фортран 90. Международный стандарт. / Перевод с англ. С.Г.Дробышевич, редактор перевода А.М. Горелик. - М.: Финансы и статистика, 1998. – 416с.
9. Горелик А.М Программирование на современном Фортране. – М.: Финансы и статистика, 2006.- 352с.
10. Горелик А.М. Современный Фортран для компьютеров традиционной архитектуры и для параллельных вычислительных систем. // Вычислительные методы и программирование. 2004, т.5., стр. 137-149 (http://num-meth.srcc.msu.ru).
11. Горелик А.М. Эволюция языка программирования Фортран (1957–2007) и перспективы его развития. // Вычислительные методы и программирование, 2008, т.9, N2, стр. 223-241.
12. Gorelik A.M. Statements and Intrinsic Procedures in Consecutive Fortran Standards. // ACM SIGPLAN Fortran Forum, 2008, N3, pp. 16-28.
13. Горелик А.М. Новый стандарт языка Фортран (Фортран 2008). // Препринты ИПМ им. М.В.Келдыша РАН. 2011. №16. 29с.
14. TS 18508 Additional Parallel Features in Fortran. ISO/IEC JTC1/SC22/WG5/N2007 Draft.

Горелик Алла Моисеевна. Старший научный сотрудник ИПМ им. М.В. Келдыша РАН. Окончила Московский областной педагогический институт (физико-математический факультет). Кандидат физико-математических наук. Член ISO/IEC JTC1/SC22/WG5. Автор более 100 печатных работ и пяти книг. Область научных интересов: языки программирования и компиляторы, технологии программирования больших вычислительных задач, параллельные вычисления.
E-mail: gorelik@keldysh.ru