НОРМА – СПЕЦИАЛИЗИРОВАННАЯ СИСТЕМА ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ[1]

А.Н. Андрианов, А.Б. Бугеря, К.Н. Ефимкин

Институт прикладной математики им. М.В. Келдыша РАН

Введение

Распределенные вычислительные системы, состоящие из объединенных высокоскоростными каналами связи ЭВМ различной архитектуры (включая многопроцессорные ЭВМ с общей и распределенной памятью), в настоящее время следует рассматривать как наиболее доступные и перспективные средства для решения задач, требующих большого объема вычислений.

При создании прикладных программ для подобных систем прикладной специалист скорее всего хотел бы знать как можно меньше об устройстве конкретной системы (но при этом полностью использовать ее возможности); не иметь проблем при переходе с одной системы на другую; избежать детального изучения при программировании моделей параллельности, особенностей компиляторов, организации и использования системных библиотек и т.п.

[1] Работа выполнена при частичной финансовой поддержке РФФИ, грант № 01-01-00411.

Реализация такой комфортной модели разработки требует решения двух основных проблем:

- упрощения программирования и отладки;
- обеспечения мобильности параллельных программ.

В данном проекте предпринимается попытка решить эти проблемы на базе подхода к разработке параллельных программ, основанного на применении специализированного декларативного языка Норма [1,2] (разработка ИПМ РАН).

Специализированный декларативный язык нового поколения Норма позволяет описывать решение широкого класса задач математической физики и при этом использовать привычные для прикладного специалиста понятия. Описание решения задачи в математических терминах имеет очень высокий уровень абстракции и отражает метод решения, а не его реализацию при конкретных условиях. Такое описание не ориентировано на конкретную архитектуру компьютера, поэтому оно предоставляет большие возможности для выявления естественного параллелизма и организации вычислений. Программа на языке Норма фактически является представлением математических расчетных формул, полученных прикладным специалистом. Таким образом, программирование на языке Норма не требует написания программы в смысле традиционных универсальных языков программирования, а выходные программы для параллельных ЭВМ с различной архитектурой строятся автоматически при помощи транслятора-синтезатора.

Язык Норма является языком сверхвысокого уровня и обеспечивает дружественный интерфейс с пользователем: фактически программирование в этом случае не требует написания программ в смысле традиционных универсальных языков программирования [3]. Такой подход к программированию основан на следующем тезисе: «универсальные языки программирования могут быть дружественными только для системного программиста».

Конечно, создание специализированных языков не является новостью в программировании; нельзя также сказать, что исследований в этом направлении нет. Однако авторы считают, что данное направление является весьма перспективным и интенсификация исследований в этой области позволит приблизиться к решению весьма актуальной в настоящий момент, но трудной проблемы разработки прикладных параллельных программ.

1. О подходе к созданию языков нового поколения

Практика показывает, что язык математики позволяет достаточно строго определять предметную область, используя математические понятия. Фак-

тически каждый раз, когда оформлялась какая-либо строгая дисциплина, которая позволяла производить количественные оценки (в широком смысле, например, выбирать элементы, удовлетворяющие определенным условиям), она описывалась в математических обозначениях. В большинстве случаев такие формулировки не носили характер процесса, то есть не предписывали четкой последовательности действий, а лишь задавали те условия, которые должны выполняться в процессе получения результата.

Второе обстоятельство, которое следует подчеркнуть, – в математических описаниях такого рода отсутствовало понятие памяти. В следствие этого значения одного семейства различались значениями индексов и вопрос об использовании одной и той же ячейки памяти для экономии просто не возникал. Теперь очевидно, что такие описания обладают естественным параллелизмом, который обусловлен только функциональными зависимостями по данным. Более того, эти описания упрощают процесс выявления параллельности (распараллеливания), так как в них не используется понятие состояния памяти, зависящего от времени, и нет необходимости решать проблемы, связанные с анализом состояния памяти (поэтому часто говорят о необходимости языков с однократным присваиванием).

Таким образом, в качестве исходного языка описания решения задачи целесообразно взять сложившуюся в данной предметной области математическую нотацию и использовать ее для автоматического построения рабочей программы. Дружественный интерфейс обеспечивается автоматически, поскольку пользователь формулирует задачу в привычных для него терминах.

При таком подходе исчезает проблема доказательства правильности программ, в том смысле, что не требуется доказывать эквивалентность программы, например на языке Фортран, и той математической записи, которая специфицирует правила получения решения. Оказывается, что из этих правил можно, при определенных условиях, автоматически получить конкретный алгоритм, учитывая функцию оптимизации, которая может задаваться по-разному: это может быть экономия памяти, параллельность вычислений и т.п.

В большинстве случаев описание метода решения задачи в математических терминах, применяемых в конкретных областях, без сомнения имеет высокий уровень абстракции и допускает большие возможности выявления естественного параллелизма. Более того, это описание не содержит никакой ориентации на конкретную архитектуру вычислительной системы. Следовательно, этот уровень описания имеет суперустойчивый характер и отражает лишь метод решения, а не его реализацию в конкретных условиях. Подчеркнем, что до появления вычислительных систем математическое описание служило в качестве описания вычислительного решения сложных задач.

Таким образом, применение рассматриваемого подхода связано с решением двух главных задач:

- формализации языка описания решения задач, сложившегося в данной предметной области, до уровня, непосредственно воспринимаемого вычислительной системой;
- разработки методов, позволяющих определять по формальному описанию такой порядок действий, который наиболее соответствует реальному окружению (решение задачи синтеза).

Мы ограничимся в основном обсуждением вопросов, связанных с параллельностью, хотя возникают и другие проблемы, связанные с оптимизацией решения задачи: экономия памяти, удобство отладки и т.п.

2. Конкретизация высказанных соображений на примере

В данном разделе кратко остановимся лишь на некоторых свойствах специализированного декларативного языка для описания решения задач сеточными методами – языка Норма, только в связи с рассматриваемым ниже примером. Более подробные сведения о языке и состоянии разработки даны в следующем разделе.

Приведенный ниже известный пример программы на Фортране взят из статьи [4], в которой автор, Лесли Лэмпорт, использует этот пример для пояснения метода гиперплоскостей, применяемого при автоматическом распараллеливании циклов:

```
DO 99 I=1,L
DO 99 J=2,M
DO 99 K=2,N                                                    (1)
   U(J,K) = (U(J+1,K) + U(J,K+1) + U(J-1,K) + U(J,K-1)) * 0.25
99 CONTINUE
```

Этот фрагмент представляет собой упрощенный вариант разностной схемы для решения задачи Лапласа на прямоугольной сетке (J,K) путем фиксированного числа итераций по I. Ограничения, накладываемые этой конструкцией на порядок вычислений, требуют анализа порядка вычислений (который задается вложенностью циклов), анализа значений переменных (который необходим из-за возникающего при выполнении программы переприсваивания значений переменным), а также анализа индексных зависимостей. Все это необходимо учитывать для того, чтобы параллельный вариант давал тот же результат, что и последовательный.

В результате анализа можно установить, что при вычислении значений в некоторой точке трехмерного индексного пространства с координатами

(I,J,K) (хотя явно указана только плоскость (J,K), переменная I вводит неявно еще одну координату), значения переменных, которые имеют индексные смещения J-1 и K-1, берутся с итерации I, остальные – с итерации I-1. При этом считается, что на всех границах значения переменной U известны и не зависят от I. Доказывается, что можно найти семейство плоскостей таких, что на любом шаге можно подсчитать значения переменной U во всех точках, принадлежащих очередной плоскости, параллельно и независимо (подробности читатель может найти в статье Лэмпорта). Заметим теперь, что этот фрагмент может быть представлен в следующей форме:

$$U_{j,k,i} = \frac{U_{j+1,k,i-1} + U_{j,k+1,i-1} + U_{j-1,k,i} + U_{j,k-1,i}}{4}, \quad (2)$$
$$j = 2,...,M; k = 2,...,N; i = 1,...,L.$$

С точностью до обозначений этот вариант является обычной математической записью, в которой естественным образом учитывается итерационный индекс. Кроме этого, запись не содержит переприсваивания значений переменным (повторного использования памяти). Поэтому при выявлении параллельных действий по записи (2) не нужно учитывать ни это обстоятельство (мы имеем дело с языком с однократным присваиванием), ни порядок вычисления витков цикла. Остается только проблема анализа индексных зависимостей.

Поскольку при таком описании перед математиком возникает геометрическая фигура – параллелепипед, то он вряд ли забудет задать необходимые для счета известные значения на границах (гранях параллелепипеда) или позаботиться о том, чтобы они были вычислены. А если и забудет, то сообщение об ошибке может быть выдано в содержательных терминах, со ссылкой на соответствующую грань.

Теперь рассмотрим простейший вариант схемы решения предлагаемой задачи:

$$U_{j,k,i} = \frac{U_{j+1,k,i-1} + U_{j,k+1,i-1} + U_{j-1,k,i-1} + U_{j,k-1,i-1}}{4}, \quad (3)$$
$$j = 2,...,M; k = 2,...,N; i = 1,...,L.$$

Этот вариант получается естественным образом, когда значение на новой итерации находят только по значениям на предыдущей итерации, и допускает широкое и очевидное распараллеливание (можно одновременно считать все точки на новом итерационном слое).

В варианте (1) как сильно усложнено выявление параллельности, так и значительно снижен достижимый уровень распараллеливания. Наличие этого варианта записи может объясняться либо желанием оптимизировать алгоритм за счет экономии памяти, либо желанием уменьшить число итераций, либо и тем и другим. Для последовательной реализации такой подход вполне оправдан. Однако потеря возможности широкого распараллеливания алгоритма может обернуться фатальным обстоятельством в условиях параллельной среды выполнения. Отсюда следует, что идея систематического перехода от последовательной программы к параллельной (как, например, в HPF [5]) не всегда будет приводить к приемлемым результатам. Наоборот, систематический переход от описания, допускающего максимальное распараллеливание, к вариантам, который за счет снижения уровня распараллеливания оптимизируют требуемые параметры, по-видимому, является более перспективным подходом. Отметим также, что алгоритм распараллеливания для формы (2) может дать (в случае оптимизации памяти) тот же уровень распараллеливания и объем используемой памяти, как и метод гиперплоскостей Лэмпорта для фрагмента (1).

В настоящее время типичной является ситуация, когда прикладной специалист разрабатывает (или уже разработал ранее) последовательную программу для решения своей задачи, например на Фортране, и ищет способ перехода от этой программы к параллельной. По мнению авторов, в настоящий момент такой переход требует существенных усилий и рассчитывать на автоматическое эффективное решение этой проблемы (например, при помощи распараллеливающих компиляторов) не приходится.

По нашему мнению, вместо перехода от последовательной версии программы к параллельной либо непосредственной разработки параллельной программы существует более перспективный третий путь, когда первичная формулировка проблемы может реализовываться как в параллельном, так и в последовательном варианте. Этот подход основывается на главном принципе: при формулировке решения задачи не накладывай лишних ограничений – можно столкнуться в дальнейшем с такой средой выполнения, в которой они вызовут неудобства и неэффективность. К счастью, математические описания почти всегда исповедуют этот принцип.

Вернемся теперь к языку Норма, который является практической иллюстрацией рассматриваемого подхода.

3. Язык Норма и содержательные обозначения

Язык Норма изначально был ориентирован на описание методов решения задач математической физики разностными методами. В этой предметной области сложилась определенная система обозначений,

используемая прикладным специалистом при записи расчетных формул, получаемых в процессе разработки метода решения.

Эта система обозначений включает как общие математические понятия (например, целые, вещественные числа, векторы, матрицы, функции, ряды), так и понятия, типичные для конкретной предметной области (например, сетка, индексное пространство, сеточная функция, итерации по времени или пространству).

Типичные понятия используются в конкретной предметной области достаточно часто и отражают ее специфику, однако разделение на общие математические понятия и типичные математические понятия условно. Существенно, чтобы язык, предназначенный для использования в некоторой предметной области, позволял применять как общие, так и типичные обозначения, и этих обозначений было достаточно для описания большинства задач данной предметной области. С другой стороны, возможность применения общих математических понятий определяет возможность использования специализированного языка и в других областях – все зависит от "мощности" набора этих понятий, допустимых в языке.

Проблему определения набора содержательных обозначений (общих и типичных) для конкретной предметной области надо решать, видимо, с учетом двух факторов:

- обеспечения дружественного интерфейса для пользователя (терминов должно быть достаточно, работать с ними просто и удобно, ничего лишнего знать не нужно);

- разрешимости задачи синтеза выходной программы по непроцедурному описанию ("мощные" наборы понятий, близкие к универсальным, часто приводят к трудноразрешимым или неразрешимым проблемам при решении задачи синтеза).

Ниже демонстрируются некоторые возможности языка Норма. Для представления понятия индексного пространства в языке Норма введено понятие области. Область – это совокупность целочисленных наборов $i_1,...,i_n; n > 0$, каждый из которых задает координаты точки n-мерного индексного пространства. С каждым направлением (осью координат) n-мерного пространства задачи связывается уникальное имя – имя индекса (имя оси координат индексного пространства). Частным случаем понятия области является сетка – логически-прямоугольная область. Одномерная область служит для задания диапазона точек на некоторой оси координат индексного пространства. В простейшем случае при описании одномерной области указывается имя одномерной области, имя индекса и диапазон изменения значений индекса:

AxisK : (k = 1..15).

Многомерная область строится при помощи операции ";" произведения областей. Пример описания двумерной области, полученной с помощью операции произведения одномерных областей **AxisK** и **AxisL**:

Square : (AxisK : (k = 1..15); AxisL : (l = 1..5)).

Для представления простых переменных в языке Норма используются скалярные величины (скаляры), а для представления векторов, матриц – величины, определенные на области. Описание величины задает тип величины – ***REAL, INTEGER*** или ***DOUBLE,*** а величина на области, кроме этого, связывается с указанной в описании областью, то есть значения этой величины могут быть вычислены в каждой точке этой области. Описания

VARIABLE First, Last ***DEFINED ON*** Square

определяют величины First, Last на области Square; это означает, что эти величины могут принимать значения в каждой точке области Square для $k = 1,...,15; l = 1,...,5$.

Расчетные формулы, получаемые прикладным специалистом, обычно записываются в виде соотношений. Например, расчетные формулы для решения системы линейных уравнений методом Гаусса - Жордана имеют вид

$$m0_{i,j} = a_{i,j}, i = 1,...,N; j = 1,...,N;$$

$$r0_i = b_i, i = 1,...,N;$$

$$m_{t,t,j} = \frac{m_{t-1,t,j}}{m_{t-1,t,t}}, j = 1,...,N; t = 1,...,N;$$

$$r_{t,t} = \frac{r_{t-1,t}}{m_{t-1,t,t}}, t = 1,...,N;$$

$$m_{t,i,j} = m_{t-1,i,j} - m_{t-1,i,t} \times m_{t,t,j}, j = 1,...,N; i = 1,...,t-1,t+1,..,N, t = 1,...,N;$$

$$r_{t,i} = r_{t-1,i} - m_{t-1,i,t} \times r_{t,t}, i = 1,...,t-1,t+1,...,N; t = 1,...,N;$$

$$x_i = r_{N,i}, i = 1,...,N.$$

Фрагмент программы на Норме, соответствующий этим расчетным формулам, приведен ниже:

```
Ot:(t=0..n). Oi:(i=1..n). Oj:(j=1..n).
Oij:(Oi;Oj). Otij:(Ot;Oij).
```

Oti:(Ot;Oi). Otij1:Otij/t=1..n.
Oti1:Oti/t=1..n.
DOMAIN PARAMETERS n = 10.
VARIABLE a ***ON*** Oij. ***VARIABLE*** m ***ON*** Otij.
VARIABLE b,x ***ON*** Oi. ***VARIABLE*** r ***ON*** Oti.
INPUT a ***ON*** Oij,b ***ON*** Oi.
OUTPUT x ***ON*** Oi.
FOR Otij/t=0 ***ASSUME*** m = a.
FOR Oti/t=0 ***ASSUME*** r=b.
OtiEQtj1,OtiNEtj1:Otij1/i=t.
OiEQti1,OiNEti1:Oti1/i=t.
FOR OtiEQtj1 ***ASSUME*** m = m[t-1,i=t]/m[t-1,i=t,j=t].
FOR OiEQti1 ***ASSUME*** r = r[t-1,i=t]/m[t-1,i=t,j=t].
FOR OtiNEtj1 ***ASSUME*** m = m[t-1]-m[t-1,j=t]*m[i=t].
FOR OiNEti1 ***ASSUME*** r = r[t-1]-m[t-1,j=t]*r[i=t].
FOR Oi ***ASSUME*** x = r[t=n].

Необходимые вычисления описаны при помощи конструкции ***FOR*** <область> ***ASSUME*** <соотношение>.

Эта конструкция – аналог форм (2) и (3) – является ключевым понятием языка Норма. Соотношение задает правило вычисления значений величины из левой части по значениям величин из правой части и индексные зависимости между переменными. Величина из левой части должна быть вычислена во всех точках области; правило для этого вычисления определено однозначно, однако при этом не требуется немедленное выполнение вычисления в данном месте программы и не задается порядок или способ вычисления. Некоторое внешнее сходство с оператором присваивания не должно вводить в заблуждение.

Индексы без смещений в записи формул могут быть опущены – при анализе программы они автоматически восстанавливаются транслятором.

В рассматриваемом фрагменте использованы также условные области. Так, описание OtiEQtj1,OtiNEtj1:Otij1/i=t определяет две непересекающиеся подобласти OtiEQtj1 и OtiNEtj1. Первая подобласть состоит из точек модифицируемой области Otij1, в которых заданное условие $i = t$ истинно, вторая – из точек, в которых это условие ложно (то есть $i \neq t$). В общем случае условие при описании условных областей представляется логическим выражением.

Отличия в приведенных выше двух способах записи расчетных формул связаны только с формой записи – представлением индексов, линейностью записи, содержательно эти способы эквивалентны.

При решении задач математической физики разностными методами процесс вычислений часто носит итеративный характер. Такие процессы можно задавать с помощью уже описанных средств языка Норма. Для этого необходимо изменить (расширить) описание областей, определенных в задаче, добавив дополнительное направление, соответствующее индексу итерации. Однако такое расширение неоправданно, так как это направление, по существу, является фиктивным – оно отражает способ вычисления, а не пространственно-временную сетку. В языке Норма определена специальная конструкция – итерация, позволяющая описывать итеративный вычислительный процесс. Неформально итерация задает итеративные вычисления с индексом итерации, меняющимся от 1 до некоторого целого положительного значения, которое определяется условием завершения итерации.

Ниже приведена схема некоторого итерационного процесса (номера строк не являются элементами языка и нужны только для пояснений, даваемых ниже). Строка 1 является заголовком итерации. В этой строке указываются имена итерируемых переменных (**r,p,u,e**) и идентификатор итерационного индекса (**nt**).

1. ***ITERATION*** r,p,u,e ***ON*** nt
2. ***BOUNDARY***
3. ***FOR*** gran1 ***ASSUME*** p=0;r=0;u=0;e=1.0.
4. ***END BOUNDARY***
5. ***INITIAL*** nt=0:
6. ***FOR*** gridsol ***ASSUME*** p=p0;r=r0;u=u0.
7. ***FOR*** gridsol ***ASSUME*** e=p/(r*(gamma-1.0))+u*u*0.5.
8. ***END INITIAL***
9. **<тело итерации>**
10. ***EXIT WHEN*** nt=ntend.
11. ***END ITERATION*** nt.

При описании итерации можно использовать возможность задания граничных значений (строки 2-4). Граничные значения итерируемой величины задаются при помощи операторов ***ASSUME***. Эти значения считаются неизменными на протяжении всей итерации и заданными на каждом шаге итерации.

Для задания начальных значений итерируемым переменным используются операторы, приведенные в строках 5-8. Фактически начальные значения – это такие значения, которые присваиваются итерируемым переменным при индексе итерации **nt=0**.

Тело итерации представляет собой часть программы, написанной на языке Норма. В частности, внутри итерации можно определить новую итерацию по другому направлению.

Завершение итерационного процесса происходит в случае, если логическое выражение, заданное в качестве условия выхода (строка 10), принимает значение «истина». Последняя строка (11) является завершением записи итерационного процесса с индексом итерации **nt**.

Знакомство с некоторыми возможностями языка Норма закончим функциями редукции, которые являются аналогами традиционных математических обозначений *sum*, *prod*, *max*, *min*. Обращение к этим функциям имеет следующий вид:

<имя-функции> ((имя-области) арифметическое-выражение).

Область определяет множество точек, в которых необходимо вычислить функцию; арифметическое выражение – множество значений, к которым применяется функция.

Пусть необходимо вычислить $V_i = W_i + \sum_{j=1}^{m} A_{i,j} \times X_j, i = 1,...,n.$

Описание этого вычисления на языке Норма

Grid:(Oi:(I=1..N); Oj:(J=1..M))
VARIABLE A ***DEFINED ON*** Grid.
VARIABLE V,W ***DEFINED ON*** Oi.
VARIABLE X ***DEFINED ON*** Oj.
FOR Oi ***ASSUME*** V = W + ***SUM***((OJ) A*X).

совпадает (с точностью до обозначений) с исходной расчетной формулой. Реализация редукционных функций для различных вычислительных окружений является нетривиальной задачей, однако эта проблема решается транслятором, а не пользователем.

Информация о языке и системе Норма размещена на сайте [15].

4. О результатах исследований по языку Норма

Несколько слов о проблеме разрешимости задачи синтеза выходной программы для языка Норма и результатах практической реализации и применения этого языка.

Язык Норма обладает несколькими характерными свойствами – это язык с однократным присваиванием; такие понятия, как память, побочный эффект, управляющие операторы, отсутствуют по определению. Эти свойства, и некоторые другие ограничения, в первую очередь на вид

индексных выражений и способы описания индексных пространств, позволяют конструктивно обосновать разрешимость задачи синтеза выходной программы [6-8]. Решение задачи синтеза основано на выявлении и анализе естественного параллелизма, который можно извлечь непосредственно из заданного описания, без применения символьных преобразований.

Полученные теоретические результаты использованы при создании транслятора-синтезатора, который позволяет получать по описанию на Норме выходные программы для распределенных параллельных вычислительных систем (параллельные вычисления с передачей сообщений), параллельных вычислительных систем с общей памятью (параллельные вычисления над общей памятью) и последовательных ЭВМ. Выходные программы могут быть построены на языках Фортран MPI, Фортран PVM, Фортран GNS, Фортран Convex, Фортран-77. Определение выходного языка осуществляется заданием соответствующей опции транслятора. Система Норма реализована для ОС UNIX, Windows, DOS. В настоящее время завершается разработка интерфейса, упрощающего работу пользователя с системой Норма в сетевом окружении.

Следует отметить, что ограничения на индексные выражения в Норме аналогичны тем ограничениям, которые накладываются в большинстве известных алгоритмов автоматического распараллеливания циклов. Это в известном смысле объясняет тот факт, что Норма может использоваться не только для описания решения задач разностными методами на статических регулярных сетках, как предполагалось сначала, но и для более широкого класса задач вычислительного характера. Например, на Норме было описано решение задачи [9], которое связано с решением интегральных уравнений, а также решены задачи на нерегулярных и адаптивных сетках.

Отметим интересный факт – в большинстве случаев исходной информацией для разработки Норма-программы была последовательная Фортран-программа, а не расчетные формулы, описывающие метод решения задачи. Это, конечно, затрудняло написание Норма-программы, поскольку требовалось еще понять, а что же делается в Фортран-программе и зачем это делается. С другой стороны, представление метода решения задачи в виде последовательной программы, по-видимому, является весьма распространенной исторически сложившейся формой для прикладного специалиста, который не видит проблем, связанных с извлечением параллелизма из последовательного представления.

Система Норма была использована при разработке параллельных программ для решения ряда задач математической физики на статических структурных сетках: моделирование дозвукового течения вязкого газа на примере обтекания воздухом пластины конечной длины (алгоритм расчета

основан на использовании кинетически-согласованных разностных схем КСРС[10], моделирование ударноволнового запуска сверхзвуковой струи газа из канала прямоугольного сечения (алгоритм расчета основан на трехмерной нестационарной схеме С.К.Годунова [11]), исследование задачи о течении газовой смеси с конденсацией в криовакуумном модуле на основе прямого численного решения кинетического уравнения Больцмана [12] и т.п. Кроме того, авторы выполнили ряд работ, связанных с программированием решения задач математической физики на статических неструктурированных и адаптивных вложенных сетках [13,14].

Список литературы

1. *Андрианов А.Н., Ефимкин К.Н., Задыхайло И.Б.* Непроцедурный язык для решения задач математической физики //Программирование. – 1991. – № 2. – С. 80-94.
2. *Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Задыхайло И.Б.* Норма. Описание языка. Рабочий стандарт: Препринт ИПМ им. М.В.Келдыша РАН. – 1995. – № 120. – 50 с.
3. *Задыхайло И.Б., Ефимкин К.Н.* Содержательные обозначения и языки нового поколения //Информационные технологии и вычислительные системы. – 1996. – № 2. – С. 46-58.
4. *Lamport L.* The Parallel execution of DO loops //Communications ACM. – 1974. – Vol.17, N 2. – P. 83-93.
5. *Steele G.L., High Jr.* Performance Fortran: status report. Workshop on Languages, Compilers and Run-Time Environments for Distributed Memory Multiprocessors, Boulder, CO, 30 Sept.-2 Oct. 1992: Reprint in SIGPLAN NOTICES. – 1993. – Vol.28, N 1. – P. 1-4.
6. *Андрианов А.Н.* Организация циклических вычислений в языке Норма: Препринт ИПМ им. М.В. Келдыша АН СССР. – 1986. – № 171. – 28 с.
7. *Андрианов А.Н., Андрианова Е.А.* Организация циклического процесса по непроцедурной записи //Программирование. – 1996. – № 4. – С.62-72.
8. *Андрианов А.Н.* Синтез параллельных и векторных программ по непроцедурной записи в языке Норма: Дис. … докт. физ.-мат. наук. – М., 2001. – 158 с.
9. *Васильев М.М., Ефимкин К.Н., Иванова В.Н.* О применении метода гидродинамических потенциалов к задаче обтекания тела //Математическое моделирование. – 1994. – Т.6, № 10. – С. 58-65.
10. *Андрианов А.Н., Гусева Г.Н., Задыхайло И.Б.* Применение языка Норма для расчета дозвукового течения вязкого газа //Математическое моделирование. – 1999. – Т.11, № 9. – С. 45-53.

11. *Andrianov A.N., Bazarov S.B., Bugerya A.B., Efimkin K.N.* Solution of three-dimensional problems of gas dynamics on multiprocessors computers //Computational Mathematics and Modeling. – 1999. – Vol.10, N 2. – P. 140-150.
12. *Andrianov A.N., Efimkin K.N., Levashov V.Y., Shishkova I.N.* The Norma Language Application to Solution of Strong Nonequilibrium Transfer Processes Problem with Condensation of Mixtures on the Multiprocessor System //Computational Science - ICCS 2001, Proceedings of International Conference, San Francisco, May 2001, Lecture Notes in Computer Science. – 2001. – Vol.2073. – P. 502-510.
13. *Андрианов А.Н.* Использование языка Норма для решения вычислительных задач на нерегулярных сетках //Тез. Всерос. науч. конф. "Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов". – Новороссийск, 1998. – С. 120-123.
14. *Андрианов А.Н., Жохова А.В., Четверушкин Б.Н.* Использование параллельных алгоритмов для расчетов газодинамических течений на нерегулярных сетках //Прикладная математика и информатика. – М.: Диалог-МГУ, 2000. – С. 68-76.
15. http: //www.keldysh.ru/norma