РАЗРАБОТКА И ПРИМЕНЕНИЕ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ В ИНСТРУМЕНТАЛЬНОМ КОМПЛЕКСЕ ORLANDO TOOLS

С.А. Горский

Введение. Активное внедрение в практику расчетных работ параллельной вычислительной техники (особенно кластерных архитектур) способствует возвращению интереса к пакетной проблематике, но вместе с тем требует реконструировать старые и определить новые подходы к программированию вычислений в пакетах программ как системах модульного программирования.

Существует широкий класс сложных предметных областей, в которых процесс компьютерного решения задач представляется в виде последовательно-параллельного выполнения подпрограмм, работающих над полем структурированных данных, являющихся фактическими параметрами этих подпрограмм. Типичным представителем систем с такой организацией процесса решения задач является пакет прикладных программ с функциональным наполнением в виде библиотек автономно компилируемых вычислительных модулей, написанных на базовых языках программирования C/C++, Фортран и др.

Анализ существующих подходов к автоматизации параллельного программирования показывает, что основное внимание в подобных системах уделяется описанию параллельной программы для решения одной конкретной задачи. В системе модульного параллельного программирования требуется удобная и гибкая среда, обеспечивающая решение на вычислительной модели предметной области целого класса задач, объекты которых связаны между собой определенными отношениями. Таким образом, создание инструментальных средств автоматического конструирования параллельных программ в системах модульного программирования остается актуальной проблемой.

Модели предметных областей, создаваемые в рамках подобных пакетов программ, относятся к классу вычислительных моделей [1] и позволяют описывать вычислительные возможности пакета программ при решении определенного класса задач. Такую вычислительную модель можно определить как совокупность значимых величин (параметров) предметной области и информационно-логических связей между ними, реализуемых модулями пакета программ. Таким образом, вычислительная модель, по сути, определяет правила применения и сочетания модулей в процессе решения задачи и позволяет автоматически осуществлять планирование вычислений по непроцедурной постановке задачи вида «по заданным значениям параметров $x_1, x_2, \ldots, x_v$ вычислить значения параметров $y_1, y_2, \ldots, y_u$».

Высшей формой развития пакетов программ можно считать интеллектуальные пакеты, в том числе пакеты знаний [2]. Технология построения пакетов знаний включает следующие этапы: определение множества отношений между объектами исследуемой предметной области и формирование ее концептуальной схемы; формирование библиотеки вычислительных модулей, реализующих абстрактные операции над объектами предметной области; проведение анализа концептуальной схемы с целью выявления ее внутреннего параллелизма; формирование постановок исследовательских задач и синтез параллельных планов их решения; построение параллельных управляющих программ, реализующих эти планы. Заключительный этап предполагает программную реализацию в общем случае параллельного плана решения задачи средствами языка программирования высокого уровня. При конструировании параллельных программ на основе библиотек вычислительных модулей, важными факторами являются эффективность и переносимость создаваемых программ, скорость их проектирования, разработки и отладки. Данная процедура, многократно выполняемая при решении задач, может оказаться достаточно сложной для пользователей пакета прикладных программ и обоснованно требует автоматизации.

В рамках этой технологии в простейшем случае концептуальная схема предметной области пакета знаний определяется [3] структурой $S=<Z, W, F>$, где: $Z=\{z_1, z_2,\ldots,z_m\}$ – множество параметров предметной области; $W=\{w_1, w_2, \ldots, w_k\}$ – множество допустимых типов данных; $F=\{f_1, f_2, \ldots, f_n\}$ – множество вычислительных модулей предметной области. С каждым модулем f_i связано два множества параметров $IN^i, OUT^i \subset Z$, называемых соответственно его входом и выходом. Вход модуля IN^i определяет параметры, значения которых необходимо задать, чтобы получить значения параметров, представленных выходом OUT^i. Множества входных и выходных параметров всех модулей из F определяются булевыми матрицами A и B размерности $n \times m$, элементы которых $a_{ij}=1$ ($b_{ij}=1$), если параметр $z_j \in IN^i$ ($z_j \in OUT^i$). Далее в работе модуль рассматривается как объект концептуальной схемы предметной области. План решения задачи строится автоматически по непроцедурной постановке задачи, формулируемой на S, является в общем случае параллельным и задается в виде булевой матрицы X размерности $k \times n$, где k – число уровней плана решения задачи, $n= |F|$, $x_{ji}=1$ означает, что модуль f_i размещен на j-м уровне плана.

Эффективная реализация перечисленных выше возможностей пакета знаний применительно к выполнению параллельных программ на вычислительных кластерах требует разработки новых методов, моделей и языков в рамках технологии параллельного программирования.

Модель. При построении модели вычислений пакета знаний важно определить способы декомпозиции общей задачи на отдельные, в достаточной мере независимые, подзадачи, допускающие возможность их параллельного выполнения (задания параллелизма по управлению), а также распределения структур данных с целью обеспечения параллельной обработки различных их частей (описания параллелизма по данным).

Множество допустимых типов значений параметров концептуальной схемы предметной области S включает стандартные простые типы данных (целый, вещественный, логический и символьный), используемые для определения скалярных величин (параметров-скаляров). А также структурированные типы данных – массивы, создаваемые на основе простых типов. Параметры-массивы представляются в виде векторов и матриц. Их границы изменения индексов задаются параметрами-скалярами целого типа. При построении плана решения задачи по концептуальной схеме предметной области S, возможно задание параллелизма по управлению, но не обеспечивается возможность описания параллелизма по данным.

С целью решения этого вопроса, для определения параллельных структур данных, дополним множество допустимых типов данных W концептуальной схемы предметной области S новым типом данных параметр-список. Параметр-список создается на основе одного параметра любого типа (скалярного, векторного или матричного) и включает множество вариантов значений этого параметра. Число элементов параметра-списка задается параметром-скаляром целого типа. Основное отличие параметра-списка от параметра-массива заключается в способе обработки элементов параметров таких типов. Параметр-массив целиком передается в вычислительный модуль, в котором далее осуществляется его обработка, в то время как параметр-список может обрабатываться поэлементно, в общем случае, параллельно, множеством экземпляров вычислительного модуля на разных процессорах. Наличие в концептуальной схеме предметной области параметров-списков приводит к появлению в процессе вычислений новых объектов (элементов параметров-списков и обрабатывающих их экземпляров модулей) и требует построения модели с более высоким уровнем детализации объектов предметной области и отношений между ними.

Определим модель вычислений в виде структуры $CM=<S, PE, FE>$, где S – это рассмотренная выше концептуальная схема предметной области, $PE=\{pe_1, pe_2, ..., pe_v\}$ – множество элементов параметров-списков, $FE=\{fe_1, fe_2, ..., fe_u\}$ – множество экземпляров модулей предметной области, $v,u \in N$. Для формального описания модели вычислений применим следующую нотацию: $R(O_1(n_1,n_2):O_2(n_3,n_4)/c)$, где O_1, O_2 – множества объектов предметной области, связанные бинарным отношением R; n_1, n_2 – числа, которые означают соответственно минимальное и максимальное число элементов O_1, связанных с элементом O_2; n_3, n_4 – числа, которые означают соответственно минимальное и максимальное число элементов O_2, связанных с элементом O_1; c – ограничение целостности, накладываемое на элементы O_2, участвующие в отношении R. Тогда связи между множествами основных объектов модели вычислений зададим следующими отношениями (рис. 1):

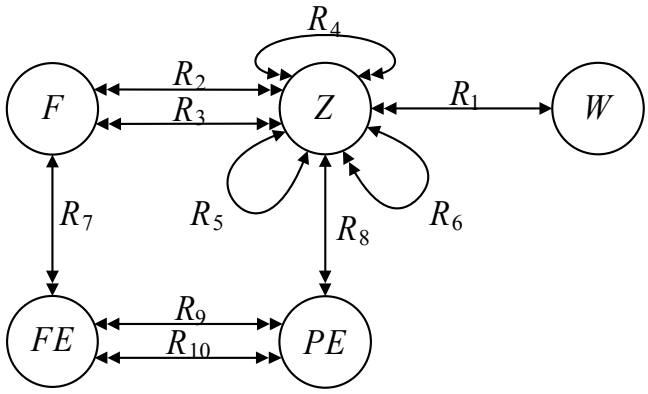

Рис. 1. Модель вычислений

- $R_1(Z(1, m): W(1, 1))$ – типизированные параметры;
- $R_2(Z(1, m):F(1, n)/c_1)$ – входные параметры из $ZI \subset Z$ для модулей из F; c_1 – ограничение на множество входных параметров модуля ($IN^i \cap OUT^i = \varnothing$);
- $R_3(Z(1, m):F(1, n)/c_1)$ – выходные параметры из $ZO \subset Z$ для модулей из F; c_1 – ограничение на множество выходных параметров модуля ($OUT^i \cap IN^i = \varnothing$);
- $R_4(ZB(1, 2):ZA(1, ma)/c_2)$ – параметры из $ZB \subset Z$, задающие границы изменения индексов параметров-массивов из $ZA \subset Z$, $1 \le ma < m$; c_2 – ограничение, требующее, чтобы эти параметры были скалярами целого типа;
- $R_5(ZL(1, 1):ZU(1, 1)/c_3)$ – параметры-списки из $ZL \subset Z$, создаваемые на основе параметров из $ZU \subset Z$; c_3 – ограничение, накладываемое на тип параметра, на основе которого создается список (он может быть параметром любого типа, кроме списка);
- $R_6(ZB(1, 1):ZL(1, ml)/c_2)$ – параметры из $ZB \subset Z$, задающие число элементов параметров-списков из $ZL \subset Z$, $1 \le ml < m$; c_2 – ограничение целостности, требующее, чтобы эти параметры были скалярами целого типа;

- $R_7(F(1, 1):FE(1, r))$ – экземпляры модуля, $r \in N$;
- $R_8(ZL(1, 1):PE(1, s))$ – элементы параметров-списков, $s \in N$;
- $R_9(FE(1, r):PE(1, s)/c_4)$ – входные элементы параметров-списков экземпляра модуля, $r, s \in N$; c_4 – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали;
- $R_{10}(FE(1, r):PE(1, s)/c_5)$ – выходные элементы параметров-списков экземпляра модуля, $r, s \in N$; c_5 – ограничение, требующее, чтобы индексы этих элементов и обрабатывающего их экземпляра модуля совпадали.

В процессе вычислений параметры-списки могут обрабатываться как неделимые структуры данных (рис. 2, а), так и поэлементно (рис. 2, б). В первом случае обработка параметра-списка осуществляется аналогично обработке параметра-массива. Во втором случае под поэлементной обработкой параметров-списков подразумевается следующее: пусть x', y' – параметры-списки, созданные на основе параметров x и y, модуль f_i: $IN^i \rightarrow OUT^i$ предназначен для их поэлементной обработки, $x \in IN^i$, $y \in OUT^i$. Интерпретация модуля f_i выполняется следующим образом: 1) происходит параллельный запуск r экземпляров (r – размерность списков x', y') модуля f_i, j-му экземпляру модуля f_i^j передается j-й элемент списка x'; 2) результат присваивается j-му элементу списка y'.

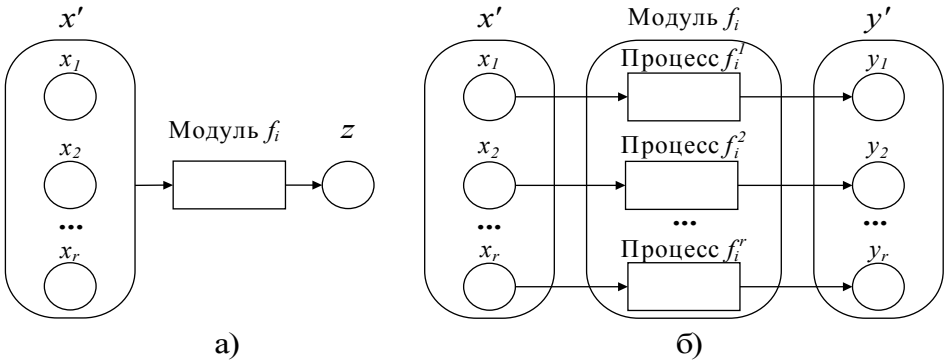

Рис.2. Обработка параметра-списка как неделимой структуры данных (а); поэлементная обработка параметра-списка (б)

Инструментальный комплекс. Архитектура инструментального комплекса для создания пакетов параллельных прикладных программ (рис. 3) включает многооконный текстовый редактор, транслятор, подсистему компиляции и подсистему запуска.

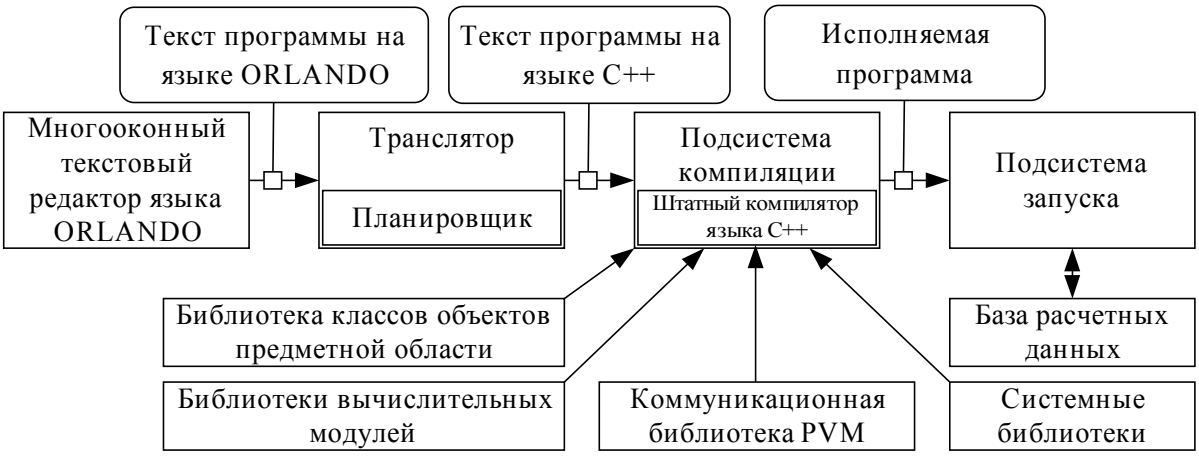

Рис. 3. Архитектура инструментального комплекса

Многооконный текстовый редактор предназначен для ввода описания вычислительной модели предметной области на специализированном входном языке ORLANDO (Objects Relations in Language of Descriptions). Данный язык относится к классу декларативных языков. ORLANDO включает набор языковых конструкций для описания объектов вычислительной модели предметной области (параметров, операций, модулей и постановок задач) и инструкций по вводу/выводу данных, инициализации значений входных параметров задачи и запуску вычислительного процесса. Вычислительная модель предметной области детально представлена в работе [4]. Текстовый редактор обеспечивает набор функций, необходимых для быстрого и удобного формирования языковых конструкций ORLANDO, а также средства для выполнения динамической проверки синтаксиса вводимых конструкций и их семантики с точки зрения корректности и целостности вычислительной модели предметной области.

Транслятор производит разбор описания предметной области на языке ORLANDO и его перевод на язык C++. При обработке конструкции постановки задачи выполняется запуск планировщика, построение параллельного плана решения задачи и включение в текст программы вызовов вычислительных модулей, реализующих план вычислений.

Подсистема компиляции параллельных программ осуществляет подключение к вычислительному кластеру, копирует исходные файлы программы на кластер, производит компиляцию основной программы и прикладных модулей с использованием штатного компилятора языка C++. При компиляции используются: библиотека классов, предназначенных для реализации объектов вычислительной модели предметной области; системные библиотеки для работы с внутренними структурами данных, управления запуском вычислительных модулей и ввода/вывода данных; библиотеки вычислительных модулей, реализующих операции предметной области; коммуникационная библиотека PVM, обеспечивающая поддержку параллельного программирования.

Подсистема запуска параллельных программ копирует из базы расчетных данных на вычислительный кластер файлы, содержащие значения входных параметров, выполняет запуск программы, осуществляет мониторинг работы этой программы и по завершении вычислений копирует в базу расчетных данных файл, содержащий значения целевых параметров.

Пример. В качестве примера рассмотрим пакет программ Градиент, предназначенный для поиска глобального минимума некоторой функции *f(x)* методом мультистарта.

Мультистарт представляет собой семейство методов, и в зависимости от способа выбора начальных точек (случайный или детерминированный), а также алгоритма поиска локального минимума выделяют ту или иную его разновидность.

Описание предметной области пакета Градиент включает следующие параметры:

- PointCount – количество начальных точек для метода мультистарта;
- SPxy – количество координат начальной точки;
- StartPoint [SPxy] – вектор, содержащий координаты начальной точки;
- ParStartPoint[PointCount] – параметр-список, созданный на основе параметра StartPoint;
- EPxyv – число элементов вектора EndPoint;
- EndPoint [EPxyv] – вектор, содержащий координаты точки и значение локального минимума;
- ParEndPoint[PointCount] – параметр-список, созданный на основе параметра EndPoint;
- Flen – длина текстовой строки, которая задает функцию;
- Function [Flen] – минимизируемая функция в текстовом виде;
- BCount – количество границ на координату (минимальное и максимальное значение координат);
- Bounds [SPxy, BCount] – матрица, задающая границы изменения координат;
- Val – значение глобального минимума;
- MPoint – координаты точки глобального минимума;
- GrInc – приращение, с помощью которого вычисляется значение градиента в точке;
- InitialShift – начальное значение шага смещения;
- MaxCall – максимальное число вызовов функции;
- Accuracy – точность вычисления локального минимума.

Параметры Accuracy, MaxCall, InitialShift, GrInc, PointCount влияют на точность, скорость и надежность нахождения глобального экстремума функции. Их выбор в общем случае зависит от исследуемой функции и требований, предъявляемых к точности нахождения точки глобального экстремума.

Схемы операций предметной области определенны следующим образом:

- Gen(BCount, PointCount, SPxy, Bounds → ParStartPoint) осуществляет генерацию начальных точек для запуска метода градиента;
- Grad(SPxy, StartPoint, EPxyv, GrInc, InitialShift, MaxCall, Accuracy, Flen, Function → EndPoint) осуществляет спуск методом градиента;
- Res(PointCount, ParEndPoint, EPxyv, SPxy → MPoint, Val) находит минимальное значение функции.

После имени операции слева и справа от стрелки располагаются соответственно списки входных и выходных параметров операций. Для операции Grad выполняется параллельный запуск множества экземпляров её модуля. Элементы параметров-списков StartPoint и EndPoint, являющихся соответственно входным и выходным параметрами операции Grad, обрабатываются независимо друг от друга в отдельных процессах – экземплярах модуля этой операции.

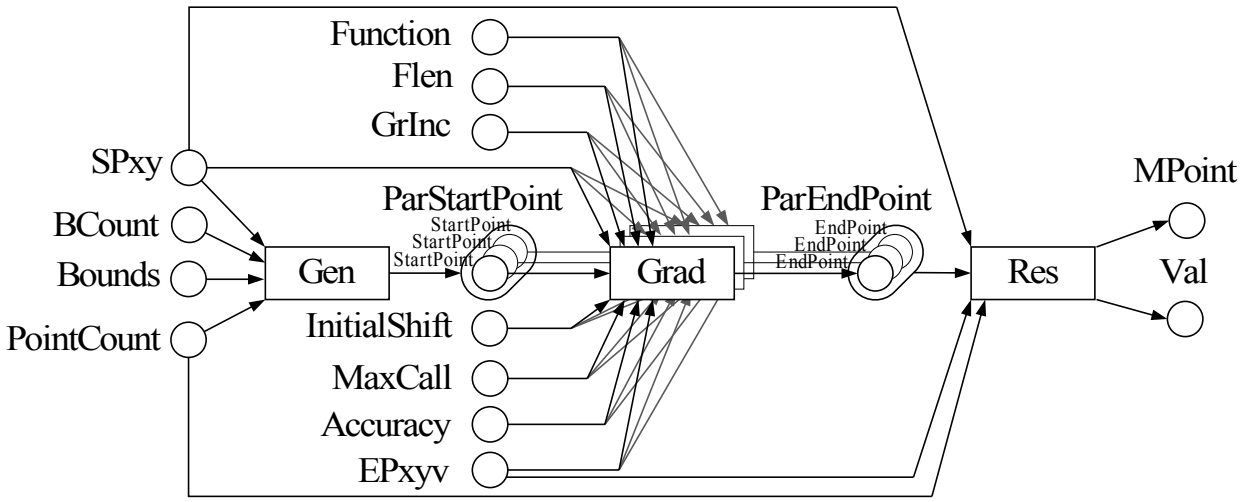

Рис. 4. Информационный двудольный ориентированный граф

Информационный двудольный ориентированный граф, представляющий схему решения задачи, изображён на рис. 4.

В качестве примера использования пакета ГРАДИЕНТ рассмотрим минимизацию функции В. Я. Катковника:

$f(x,y)=0.5(x^2+y^2)(1+0.5cos(1.5x)cos(\pi y)+0.5cos(\sqrt{5}\,x)cos(3.5y))$.

с ограничением:

$(x-2)^2+y^2-4\le 0$.

Данное ограничение определяет область внутри круга радиуса 2 с центром в точке с координатами (2; 0) и с глобальным минимумом, расположенным на ее границе.

Данная функция является сильно многоэкстремальной и относится к классу сложных задач оптимизации. Трехмерное изображение этой функции представлено на рис. 5. Она имеет глобальный экстремум в начале координат. В окрестности глобального минимума оптимизируемая функция имеет локальные минимумы, мало отличающиеся от глобального и друг от друга.

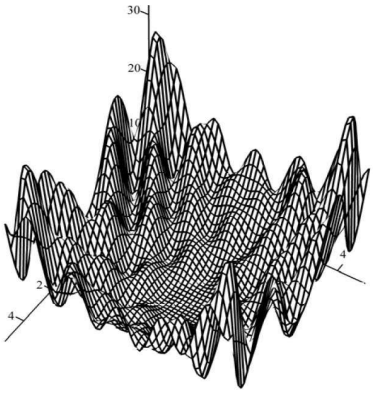

Рис. 5. Функция Катковника в области [-5;5]x[-5;5]

При проведении расчетов на вычислительном кластере создавались виртуальные машины, включающие от 1 до 8 двухпроцессорных узлов кластера. Расчеты проводились при значении параметра PointCount равном 1000. В таблице приведены время счета с использованием различного числа узлов кластера и соответствующий коэффициент ускорения счета.

Расчетное время на различном числе процессоров

Число процессоров	1	2	3	4	8
Время счета (с)	245	123	85	64	31
Коэффициент ускорения	1	1,992	2,882	3,828	7,903

Заключение. Суть и новизна результатов, представленных в данной работе, заключаются в разработке инструментальных и языковых средств представления пакетных знаний о предметной области, средств

формирования постановок исследовательских задач и исполнения параллельных программ решения задач на многопроцессорных вычислительных комплексах. Представленный подход характеризуется применением методов крупноблочного распараллеливания в рамках общей схемы решения задачи, программированием в терминах объектов предметной области, обеспечением простоты и гибкости модификации описаний предметной области и постановок задач, автоматизацией конструирования параллельной программы. Разработанные инструментальные средства ориентированы на кластерные вычислительные системы, работающие под управлением ОС Linux.

Работа выполнена при частичной поддержке РФФИ (грант №10-07-00146).

ЛИТЕРАТУРА:

1. Э.Х. Тыугу Концептуальное программирование – М.: Наука, 1984. – 256 с.
2. Г.А. Опарин Основанная на знаниях технология решения вычислительных задач // Информационные технологии контроля и управления транспортными системами. Вып. 6. – Иркутск: ИрИИТ, 2000. – С. 3-15.
3. Г.А. Опарин, А.П. Новопашин Булево моделирование планирования действий в распределенных вычислительных системах // Теория и системы управления. – 2004. – №5. – С.105-108.
4. Г.А. Опарин, А.Г. Феоктистов Инструментальная распределенная вычислительная САТУРН-среда // Программные продукты и системы. 2002. № 2. С. 27–30.