ТЕХНОЛОГИЯ АВТОМАТИЧЕСКОГО КОНСТРУИРОВАНИЯ АДАПТИВНЫХ ВЫЧИСЛЕНИЙ НА ОСНОВЕ ФОРМАЛИЗМА КОРТЕЖА И МАСКИ

В.А. Подчукаев, Д.Ю. Петров, М.Ю. Пономарёв, Д.А. Фёдоров Д.А., Д.С. Шевченко Д.С.

Предложена технология автоматического конструирования адаптивных вычислений, базирующаяся на не имеющем аналогов в отечественной и зарубежной литературе формализме кортежа и маски, развиваемом авторами статьи.

Введение

Переход в ближайшем будущем от суперкомпьютерных систем терафлопного уровня к системам петафлопного уровня, предусматриваемых проектами: Chapel компании Cray, Fortress компании Sun Microsystems и «X-10» компании IBM, требует реализации адаптивных вычислений, при которых аппаратная архитектура суперкомпьютера подстраивается под структуру (топологию) вычислительной задачи.

Вне зависимости от того, на какой аппаратной платформе строится вычислительный кластер (HPChigh-perfomance claster), будь то:

- классические многоядерные процессоры,
- векторные процессоры,
- ПЛИС-процессоры,
- ПАИС-процессоры (или процессоры, реализованные на Программируемых Аналоговых Интегральных Схемах),
- аналого-цифровые процессоры, реализуемые на системах на кристалле (SOC-System-on-Chip),

собственно теория адаптивных вычислений складывается из решения двух основных задач:

1) синтеза топологии многопроцессорного вычислительного ядра (вычислителя), в котором осуществлено распараллеливание вычислений под ту или иную вычислительную задачу;

2) создания программного кода, доставляющего решение вычислительной задаче на синтезированном вычислителе.

Настоящая статья преследует цель предложить формализм теории адаптивных вычислений, исходя из условия конструирования топологии вычислителя, обусловленной следующим математическим описанием предметной области.

1. Математическая модель предметной области

В [1] для описания объектов и/или явлений естественной и/или искусственной природы предложена абстрактная математическая модель, распространяющаяся как на системы обыкновенных конечномерных дифференциальных уравнений, так и на уравнения в частных производных

$$\dot{x} = [P(v,t)]_{(*)}x + [B(v,t)]_{(*)}u, \qquad (1)$$

$$x \in R^n,\ v \in R^n,\ u \in R^m,\ t \in [0, \infty),$$

где x – вектор состояний; u – вектор внешних воздействий; $[P(v,t)]_{(*)}$, $[B(v,t)]_{(*)}$– функциональные матрицы соответствующих размеров, индекс $_{(*)}$ в обозначении которых задаёт область изменения их аргументов. В роли этих аргументов выступают время t и некоторый n-мерный вектор v (условно, вектор пространственных переменных), подчинённый некоторому заданному

уравнению (*) вплоть до того, что такими уравнениями может задаваться область изменения аргументов каждого элемента функциональных матриц.

Предметом исследований в статье является порождённая математической моделью (1) следующая математическая модель любого элемента функциональных матриц и любой компоненты вектора внешних воздействий u_ρ $(\rho = \overline{1,m})$, представляющая собой алгебраическую сумму произведений конечного числа символов $a_1, \ldots, a_n$ (назовём их упорядоченную совокупность **кортежем**) с заданным числом математических операций над ними [2, С.198-199]

$$u_\rho(a_1, ..., a_n) = \sum_{i=1}^{k} \pm c_i \left(\prod_{j=1}^{n} \varphi_{i,j,\alpha}(a_j^{m_j}) \right)_i, \qquad (2)$$

где c_i $(i = \overline{1,n})$ - числовой коэффициент.

Будем полагать, что от всех или части элементов кортежа берутся элементарные трансцендентные функции (показательные, логарифмические, тригонометрические, обратные тригонометрические функции), гиперболические и обратные гиперболические функции, эллиптические и бесселевы функции и т.д. Обозначим эту операцию в виде $\varphi_{i,j,\alpha}$, где i – номер слагаемого в (2), j - номер элемента кортежа, от которого берётся функция $\varphi_{i,j,\alpha}$, α - идентификатор класса функции (например, $\alpha = 1$ - для тригонометрических функций; $\alpha = 2$ - для обратных тригонометрических функций и т.д.). Условимся для алгебраических функций использовать признак $\alpha = 0$, при котором собственно обозначение φ можно опустить, допустив для показательных и логарифмических функций использование в качестве m_i соответствующего элемента кортежа.

Отметим, что математическая модель вычислителя (2) содержит все 8 важнейших математических операций: сложение $\left(обозначено \sum или +\right)$, вычитание (-), умножение $(\prod)$, включая следующие 5 операций, идентифицируемые показателем степени m_j:

- $m_j = -1$ - деление $\left(a_j^{-1} = \frac{1}{a_j} \right)$;

- $m_j = \eta$, где η-натуральное число, - возведение в степень $\left(a_j^{\eta}\right)$;

- $m_j = (\eta)$, где η-натуральное число, - дифференцирование $\left(a_j^{(\eta)}\right)$;

- $m_j = \{\eta\}$, где η-натуральное число, - интегрирование $\left(\eta = 1, 2, 3\right)$- кратность интеграла $\left(a_j^{\{\eta\}}\right)$;

- $m_j = \frac{1}{\eta}$, где η-натуральное число, - извлечение арифметического корня степени η $\left(\sqrt[\eta]{a_j}\right)$.

2. Вербальная постановка задачи адаптивных вычислений

Дано: (2). Требуется автоматически спроектировать алгоритм распараллеливания вычислений и программно реализовать его на той или иной аппаратной платформе вычислителя.

Нетрудно видеть, что поставленная задача сопряжена с решением 2-х проблем:

1) геометрической интерпретацией (2) с точки зрения топологии вычислительного ядра;

2) формализацией автоматического конструирования вычислений в условиях либо графического представления технической реализации вычислений, либо программной технической

реализации, связанной с обращением к соответствующим ядрам вычислителя, например, имеющим IP-адрес.

Изложим идею автоматического конструирования вычислений применительно к математической модели (2).

3. Геометрическая интерпретация топологии вычислительного ядра

Введём в рассмотрение образ математической модели (2), назвав его **маской**. Будем понимать под маской некоторую матрицу, каждая строка которой будет соответствовать определённому слагаемому алгебраической суммы (2), являясь умножителем всех элементов кортежа с выполняемыми математическими операциями над ними. При этом сама маска (матрица) будет выполнять роль сумматора.

Тем самым маска будет задавать двумерную топологию вычислительного ядра, организуемого следующим образом: в первом столбце маски будем записывать числовые коэффициенты c_i со знаком (по умолчанию «+» опустим, а «» условимся записывать обязательно). В остальных столбцах маски (со 2-го по *(n+1)-й*) посредством показателей степени будем записывать математические операции над соответствующим элементом кортежа с a_1 по a_n включительно (в случае нескольких операций, отделяя соответствующие показатели степени m_j и обозначения функций разделителем «;»).

Если тот или иной элемент кортежа не присутствует в том или ином слагаемом алгебраической суммы, представляющем собой произведение всех элементов кортежа, в соответствующей строке маски в ячейке, соответствующей этому элементу, будем записывать «0», что даёт $a_j^0 = 1$.

3.1. Пример

Продемонстрируем двумерную топологию вычислительного ядра, синтезированную в результате использования формализма кортежа и маски, на примере следующей математической модели

$$u(x,a,b) = x^4 + a \cdot x^3 + b \cdot x^3 + a \cdot b \cdot x^2 . \qquad (3)$$

Модели (3) соответствуют следующий кортеж

$$\begin{bmatrix} x & a & b \end{bmatrix}$$

и маска

$$\begin{bmatrix} 1 & 4 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 1 & 3 & 0 & 1 \\ 1 & 2 & 1 & 1 \end{bmatrix}.$$

Её анализ показывает, что для технической реализации маски необходимы: сумматор на 4 входа (суммирующий строки маски), 4 умножителя (по числу строк маски) и 4 элемента возведения в степень первого элемента кортежа.

Если представить маску в виде

$$\begin{bmatrix} 1 & 2;2 & 0 & 0 \\ 1 & 2;1 & 1 & 0 \\ 1 & 2;1 & 0 & 1 \\ 1 & 2 & 1 & 1 \end{bmatrix},$$

то количество элементов возведения в степень первого элемента кортежа можно сократить до одного, который возводит x в квадрат $\left(x^2\right)$, что видно из приведённой маски.

Если привести в (3) подобные члены, то получим выражение

$$u(x,a,b) = (x+a)(x+b)x^2,$$

которое можно рассматривать как распознанные в (3) алгебраические формулы бинома Ньютона, полученные «выдавливанием» их из исходного выражения (3). При этом исходная двумерная топология вычислительного ядра, реализующего (3), трансформируется в многослойную топологию, иллюстрируемую следующим рисунком

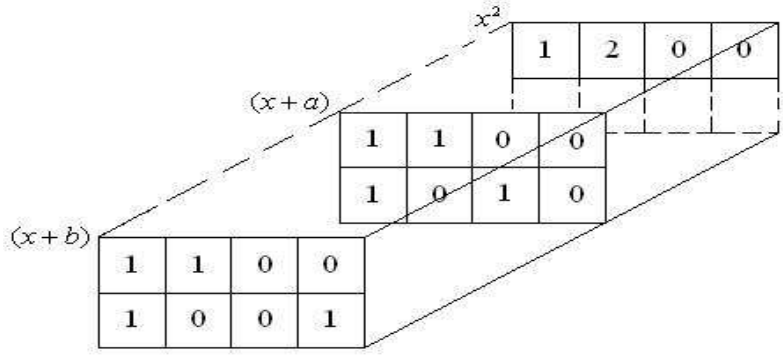

Рис. 1. Геометрическая интерпретация многослойной топологии вычислительного ядра

Технической реализацией топологии, изображённой на рис.1, может быть:

1) многослойная печатная плата с интерпретацией слоёв рисунка как сумматоров, а строк на каждом слое как умножителей,

2) топология заказной ПЛИС, ПАИС или SOC;

3) и т.д.

Таким образом предлагаемый формализм кортежа и маски позволяет дать наглядную графическую интерпретацию топологии вычислительного ядра, в которой математическим операциям, выполняемым ядром, ставятся во взаимно-однозначное соответствие показатели степени элементов кортежа, являющиеся элементами маски, чьи строки представляют собой умножители, а сумма всех строк является сумматором и при этом знак учитывается знаком элементов первого столбца маски.

4. Задача синтеза топологии вычислительного ядра

Приведенный пример и иллюстрирующий его рисунок показывают, что ключом к получению многослойной топологии вычислительного ядра является решение задачи распознавания образов математических формул в произвольно заданной алгебраической сумме произведений конечного числа символов с заданным числом математических операций над ними. Применительно к классу алгебраически-дифференциальных формул (формулы Виета, бином Ньютона, формула Лейбница) с использованием математического аппарата кортежа и маски эта задача поставлена и решена в статье [3], где на языке кортежа и маски сконструированы образы указанных формул и описан алгоритм их автоматического распознавания. Предложенный алгоритм программно реализован (программой viet) на

вычислительном ресурсе класса WEB 2.0 в среде аналитических вычислений «АНАЛИТИК-С» [4], доступной пользователям Интернет по адресу http://www.sgau.ru/analitik_c/.

Отметим, что после ввода (2) в командном окне «АНАЛИТИК-С» кортеж и маска генерируются программой viet автоматически, выполняя роль внутренних переменных указанной программы.

Проблема распознавания математических формул других классов функций остаётся открытой, впрочем, не вызывая принципиальных затруднений. Поскольку предложенный алгоритм применим и к этим классам формул в условиях представления маски (обозначим её через M) в виде $M[1:k,1:n+1,1:s]$, где k – число слагаемых в (2); n – число элементов кортежа; s – число формул, «вложенных» в (2). Тем самым размерность матрицы, обозначающей маску, дополнена третьим измерением, в которое «выдавливаются» найденные в (2) формулы. Так первая из найденных в (2) формул в маске будет иметь адрес M[…,…,2], вторая – M[…,…,3] и так далее. При этом по адресу M[…,…,1] постоянно будет прописан остаток исходного выражения (2), который является предметом дальнейшего поиска возможных содержащихся в нём формул.

Отметим, что в основу механизма распознавания образов алгебраически-дифференциальных формул положена следующая функциональная схема:

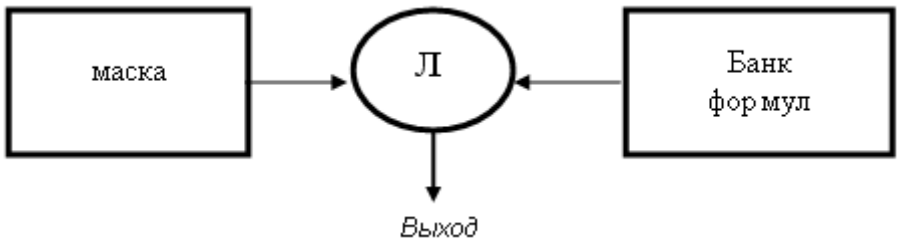

Рис. 2. Функциональная схема механизма распознавания образов формул

где Л – обозначение логического блока для работы с банком формул, представляющим собой набор масок распознаваемых формул.

Поскольку от перемены мест слагаемых сумма не меняется, строки маски можно переставлять, равно как и её столбцы, начиная со второго (от перемены мест сомножителей произведение не меняется). Этими обстоятельствами можно воспользоваться, выделяя в маске компактные слои (по горизонтали), соответствующие признакам класса распознаваемых формул, идентифицируемых индексом $\alpha = 0, 1, 2, \ldots$, присутствующим в обозначении взятия функции $\varphi_{i,j,\alpha}$, и используя вышеприведённую функциональную схему для распознавания.

5. Задача технической реализации топологии вычислительного ядра, задаваемой маской

Описанный формализм представления вычислительного кластера с помощью кортежа и маски есть не что иное, как аналог его функциональной схемы. Так, кортеж описывает входы вычислителя, а маска (или сумматор) его выход.

Из примера видно, что строки маски в явном виде задают алгоритм распараллеливания вычислений, то есть задают параллельные потоки выполнения математических операций над элементами кортежа (сигналами в аналоговой и/или цифровой форме), которые после выполнения всех вычислений в каждом из потоков и их перемножения должны быть просуммированы.

Поставив во взаимно-однозначное соответствие ячейкам маски соответствующие технические устройства аналогового и/или цифрового типов, реализующие записанные в ячейку математические операции над соответствующим элементом кортежа, можно получить функциональную схему вычислителя.

В статье [5] подобная задача эквивалентных преобразований формализована как задача рефлексивной семантики, которая состоит в интерпретации одного формализованного языка (языка математических операций) в категории другого языка (языка технической реализации математических операций в аналоговой или цифровой форме). Причём, оба языка являются формализацией одной и той же системы понятий, в роли которых выступают математические операции. Поскольку оба языка фактически совпадают, исследованию подлежит выполнимость в модели технической реализации (МТР) свойства модели математической операции (ММО), задаваемого в маске целым числом, возможно со знаком и в скобках, обозначающим соответствующую математическую операцию.

В статье [5] задача рефлексивной семантики в указанном выше смысле решена для случая графического представления МТР в обозначениях пакета Anadigm Designer 2, предназначенного для программирования ПАИС. Результатом решения стала программа-конвертер автоматического проектирования для заданной маски связей между её элементами, результатом выполнения которой является электрическая принципиальная схема вычислителя в обозначениях указанного выше пакета на аппаратной платформе ПАИС производства компании Anadigm, позволяющая синтезировать программный код для прошивки ПАИС.

Предложенную в [5] технологию синтеза программного кода для реализации вычислителя на выбранной аппаратной платформе продемонстрируем на примере программируемых логических контроллеров (ПЛК) производства компании Siemens (400 серии) [6].

5.1. Пример

1-й этап предлагаемой технологии – решение задачи рефлексивной семантики, что означает составление таблицы соответствии ММО $\Rightarrow$ МТР. Применительно к 400-й серии ПЛК Siemens эта таблица имеет вид:

Таблица 5.1. Таблица соответствия ММО фрагментам кода на языке STL

№ п/п	ММО	МТР на языке STL (Statement List) компании Siemens	Технические ограничения ПЛК
1	сложение	+R VD0, VD4 **{ ID1(in1, in2/out) }**	Максимальное число входов сумматора – 2. Техническая реализация сложения 3-х входов (VD0, VD4, VD8): **ID1 (VD0, VD4)** **ID1 (VD4, VD8)**
2	вычитание	-R VD0, VD4 **{ ID2 (in1, in2/out) }**	Максимальное число входов вычитателя – 2. Техническая реализация вычитания 2-х значений (VD4, VD8): **ID2 (VD0, VD4)** **ID2 (VD4, VD8)**
3	умножение	*R VD0, VD4 **{ ID3 (in1, in2/out) }**	Максимальное число входов умножителя – 2. Техническая реализация 3-х умножений (VD4, VD8, VD12): **ID3 (VD0, VD4)** **ID3 (VD4, VD8)**
4	деление	/R VD0, VD4 **{ ID4 (in1, in2/out) }**	Число входов делителя – 2. Техническая реализация последовательно выполняемых 2-х делений VD0 на (VD4, VD8, VD12): **ID4 (VD0, VD4)**

			ID4 (VD4, VD8)
5	взятие арифметического корня	SQRT VD0,VD4 **{ ID5 (in, out)}**	Пример технической реализации вычисления арифметического корня 4-ой степени (VD0, VD4, VD8): **ID5 (VD0, VD4)** **ID5 (VD4, VD8)**
6	возведение в степень	LN VD0, AC0 *R VD4, AC0 EXP AC0, AC0 **{ ID6 (in1, in2, out)}**	Округление максимального значения числа с плавающей запятой. Пример: возвести число 5 в степень 3/2 (VD0 = 5, VD4=1,5): **ID6 (VD0, VD4, AC0)**
7	дифференцирование	CALL "LEAD_LAG",DB80 IN :=MD10 OUT :=MD20 **{ ID7 (in, out) }**	В переменной DBD 12 блока данных DB80 устанавливается длительность опережения, а переменная задержки DBD 16 обнуляется. Пример: продифференцировать входную переменную MD10: **ID7 (MD10, MD20)**
8	интегрирование	CALL "LEAD_LAG",DB80 IN :=MD10 OUT :=MD20 **{ ID8 (in, out) }**	В переменной DBD 16 блока данных DB80 устанавливается длительность задержки, а переменная опережения DBD 12 обнуляется. Пример: проинтегрировать входную переменную MD10: **ID8 (MD10, MD20)**
9	реализация коэффициента усиления	*R VD10, VD14 **{ ID9 (in1, in2/out) }**	Вход in1 должен содержать коэффициент усиления, а in2 произведение всех элементов строки маски начиная со второго **ID9 (VD10, VD14)**

2-й этап предлагаемой технологии – автоматический синтез программного кода, реализующего вычислитель с использованием фрагментов МТР, приведенных во втором столбце таблицы 5.1. Написание такого кода на любом из языков высокого уровня не встречает принципиальных затруднений.

Заключение

Резюмируя изложенное можно заключить, что предлагаемая идеологигия реализации адаптивных вычислений может использоваться как на готовых аппаратных платформах, предназначенных для реализации вычислительных кластеров, так и для разработки заказных конфигураций этих платформ под конкретную вычислительную задачу. Последнее представляется предпочтительным, поскольку готовые аппаратные платформы заведомо избыточны.

Следующими приложениями описанной технологии являются её адаптация к:

1) Grid-технологиям распределённых вычислений [7]. Собственно формирование Grid-модели вычислений состоит в разработке таблицы соответствия, аналогичной таблице 1, где столбец МТР представляет собой фрагменты кода обращения к узлам Grid, осуществляющим техническую реализацию соответствующих математических операций;

2) синтезу атомарно-молекулярной структуры наноструктурных элементов технической реализации АСУ ТП и АСУ РВ [8]. Собственно формирование атомарно-молекулярной структуры наноструктурных элементов состоит в разработке таблицы соответствия, аналогичной таб-

лице 1, где столбец МТР представляет собой фрагменты атомарно-молекулярной структуры, осуществляющие техническую реализацию соответствующих математических операций.

Литература

1. Подчукаев В.А. Аналитические методы теории автоматического управления. – М.: Физматлит, 2002. – 256 с.
2. Подчукаев В.А. Теория информационных процессов и систем: учебное пособие для вузов. – М.: Гардарики, 2007. – 207 с.
3. Подчукаев В.А., Кулаков К.А. Аналитическое проектирование технической реализации законов управления//Мехатроника, автоматизация, управление. 2007. № 7. С.33-39.
4. Подчукаев В.А. (правообладатель). Свидетельство Федеральной службы по интеллектуальной собственности, патентам и товарным знакам об официальной регистрации программы для ЭВМ № 2007612671 «Среда аналитических вычислений «АНАЛИТИК-С» от 21.06.2007 г.
5. Подчукаев В.А., Шевченко Д.С. Автоматическое проектирование технической реализации законов управления на платформе программируемых аналоговых интегральных схем производства компании Anadigm//Цифровые системы управления и обработки информации: приложение к журналу «Мехатроника, автоматизация, управление». 2008. № 7. С. 7-11.
6. Подчукаев В.А., Петров Д.Ю. Автоматическое проектирование технической реализации законов управления на платформе программируемых логических контроллеров производства компании Siemens//Материалы 2-ой Российской мультиконференции по проблемам управления. Мехатроника, автоматизация, управления. – Санкт-Петербург: ГНЦ РФ ЦНИИ «Электроприбор, 2008. С. 93- 96.
7. Подчукаев В.А., Фёдоров Д.А. Grid-технологии создания вычислительного кластера на базе распределённой компьютерной сети Саратовского государственного аграрного университета имени Н.И. Вавилова// Приложение к журналу «Открытое образование»: труды XXXV юбилейной международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе «IT+S&E’08». - Украина, Крым, Ялта-Гурзуф, 2008. - С. 57-58.
8. Подчукаев В.А., Шевченко Д.С. Автоматическое проектирование атомарно-молекулярной структуры наноэлектронных компонентов микросистемной техники//Приложение к журналу «Открытое образование»: труды XXXV юбилейной международной конференции «Информационные технологии в науке, образовании, телекоммуникации и бизнесе «IT+S&E’08». - Украина, Крым, Ялта-Гурзуф, 2008. - С. 47-49.