РАСШИРЕННАЯ МОДЕЛЬ КОНЦЕПЦИИ Q-ДЕТЕРМИНАНТА И ЕЕ ПРИМЕНЕНИЕ ДЛЯ РЕАЛИЗАЦИИ РЕСУРСА ПАРАЛЛЕЛИЗМА ЧИСЛЕННЫХ АЛГОРИТМОВ*

В.Н. Алеева, Н.В. Валькевич, Ю.С. Лаптева, Д.Е. Тарасов
Южно-Уральский государственный университет (НИУ),
г. Челябинск, Россия

Аннотация. *Описана расширенная модель концепции Q-детерминанта. Показано ее применение для реализации ресурса параллелизма численных алгоритмов на параллельных вычислительных системах с общей и распределенной памятью. Результаты исследования могут использоваться для повышения эффективности реализации численных алгоритмов на параллельных вычислительных системах.*

Ключевые слова: *Q-детерминант алгоритма, представление алгоритма в форме Q-детерминанта, Q-эффективная реализация алгоритма, ресурс параллелизма алгоритма, параллельная вычислительная система, параллельная программа,*
Q-эффективная программа.

Введение. Использование ресурса параллелизма численных алгоритмов может решить проблему повышения эффективности их выполнения на параллельных вычислительных системах (ВС). В связи с этим очень важным и развитым направлением является исследование параллельной структуры алгоритмов и программ [1]. Для исследования алгоритмов используется их представление с помощью графов. Для описания свойств, особенностей, статических и динамических характеристик алгоритмов с целью их эффективной реализации на параллельных ВС создается Интернет-энциклопедия AlgoWiki [2]. Современное состояние исследований в области распараллеливания алгоритмов и их реализации на параллельных ВС представлено в докладе [3]. В нем поставлен ряд вопросов, которые на сегодня остаются открытыми, в частности, вопросы, связанные со сложностью описания алгоритмов: «Как изобразить потенциально бесконечный граф? Как изобразить потенциально многомерный граф? Как показать зависимость структуры графа от размера задачи? Как выразить имеющийся параллелизм и показать возможный способ параллельного исполнения?». Еще один подход к созданию параллельных программ - синтез параллельных программ. Метод синтеза был впервые описан в [4]. Основанный на нем синтез параллельных программ заключается в том,

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00865 а, при поддержке Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011).

что из базы знаний параллельных алгоритмов конструируются новые параллельные алгоритмы для решения более крупных задач. На основе метода синтеза разработана технология фрагментированного программирования и реализующие ее язык и система программирования LuNA. В настоящее время это направление исследований развивается [5]. Подход является универсальным, но проблему исследования и использования ресурса параллелизма алгоритмов не решает. Исследование ресурса параллелизма алгоритмов проводится с помощью их программной реализации [6]. Если решать задачу выявления ресурса параллелизма алгоритма, то использование программы, реализующей алгоритм, может оказаться ошибочным, поскольку программа может не содержать всех реализаций алгоритма, в частности, при ее создании могла быть потеряна реализация, полностью использующая ресурс параллелизма алгоритма.

По-видимому, для решения проблемы исследования ресурса параллелизма алгоритмов и его реализации на параллельных ВС более перспективным является подход, основанный на универсальном описании алгоритма, показывающем ресурс параллелизма в полной мере. Один из таких подходов – концепция Q-детерминанта [7].

Целью данного исследования является описание расширенной модели Q-детерминанта, а также основанного на ней метода, с помощью которого для численного алгоритма можно разработать программу, использующую ресурс параллелизма алгоритма. Использование концепции Q-детерминанта для разработки параллельных программ предлагается впервые.

Концепция Q-детерминанта. Для лучшего понимания данной работы опишем кратко основные понятия концепции Q-детерминанта.

Пусть α – алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$, где $N = \{n_1, ..., n_k\}$ – множество параметров размерности проблемы или пустое множество, B – множество входных данных, $\bar{y} = (y_1, ..., y_m)$ – множество выходных данных. $\overline{N}$ – вектор $(\bar{n}_1, ..., \bar{n}_k)$, где $\bar{n}_i$ – некоторое заданное значение параметра n_i. $\{\overline{N}\}$ – множество всевозможных векторов $\overline{N}$. Q – множество операций, используемых алгоритмом α.

Любое однозначное отображение $w: \{\overline{N}\} \to V$, где V – множество всех выражений над B и Q, называется безусловным Q-термом. Если при любом $\overline{N} \in \{\overline{N}\}$ и любой интерпретации переменных B $w(\overline{N})$ принимает значение логического типа, то w называется безусловным

логическим Q-термом. Пусть $u_1,...,u_l$ – безусловные логические Q-термы, $w_1,...,w_l$ – безусловные Q-термы. Множество пар (u_i,w_i), где $i=1,...,l$, обозначается $(\overline{u},\overline{w})=\{(u_i,w_i)\}_{i=1,...,l}$ и называется условным Q-термом длины l. Счетное множество пар безусловных Q-термов $(\overline{u},\overline{w})=\{(u_i,w_i)\}_{i=1,2,...}$ называется условным бесконечным Q-термом, если $\{(u_i,w_i)\}_{i=1,...,l}$ является условным Q-термом для любого $l<\infty$. Если не имеет значения, является ли Q-терм безусловным, условным или условным бесконечным, то его можно называть Q-термом.

Под вычислением безусловного Q-терма w при интерпретации B следует понимать вычисление выражения $w(\overline{N})$ при некотором $\overline{N}\in\{\overline{N}\}$. Для вычисления при заданной интерпретации B и некотором $\overline{N}\in\{\overline{N}\}$ условного Q-терма $(\overline{u},\overline{w})=\{(u_i,w_i)\}_{i=1,...,l}$ необходимо найти такие $u_{i_0}(\overline{N}),w_{i_0}(\overline{N})$, что $u_{i_0}(\overline{N})$ принимает значение *true*, а значение $w_{i_0}(\overline{N})$ определено. В качестве значения $(\overline{u},\overline{w})$ нужно взять $w_{i_0}(\overline{N})$. Если установлено, что выражений $u_{i_0}(\overline{N}),w_{i_0}(\overline{N})$ не существует, то значение $(\overline{u},\overline{w})$ для данной интерпретации B и данного $\overline{N}$ не определено. Вычисление условного бесконечного Q-терма определяется аналогично.

Предположим, что I_1,I_2,I_3 – подмножества множества $I=(1,...,m)$ такие, что: одно или два из множеств $I_i(i=1,2,3)$ могут быть пустыми, $I_1\cup I_2\cup I3=I$, $I_i\cap I_j=\varnothing\,(i\neq j;i,j=1,2,3)$. Множество Q-термов $\{f_i\}_{i\in I}$ удовлетворяет условиям: $f_{i_1}(i_1\in I_1)$ – безусловный Q-терм, $f_{i_1}=w^{j_1}$; $f_{i_2}(i_2\in I_2)$ – условный Q-терм, $f_{i_2}=\{(u_j^{i_2},w_j^{j_2})\}_{j=1,...,l_{i_2}}$, l_{i_2} является вычислимой функцией параметров N; $f_{i_3}(i_3\in I_3)$ – условный бесконечный Q-терм, $f_{i_3}=\{(u_j^{i_3},w_j^{j_3})\}_{j=1,2,...}$. Если алгоритм α состоит в том, что для определения $y_i(i\in I)$ требуется вычислить Q-терм f_i, то множество Q-термов $y_i(i\in I)$ называется Q-детерминантом алгоритма α, а представление алгоритма в виде $y_i=f_i(i\in I)$ представлением в форме Q-детерминанта.

Реализацией алгоритма α, представленного в форме Q-детерминанта $y_i = f_i(i \in I)$, называется вычисление Q-термов $f_i(i \in I)$ при заданной интерпретации B и некотором $\overline{N} \in \{\overline{N}\}$. Если реализация такова, что выражения $W(\overline{N}) = \{w^{j_1}(\overline{N})\ (i_1 \in I_1);$

$u_j^{i_2}(\overline{N}), w_j^{j_2}(\overline{N})\ (i_2 \in I_2, j = 1,...,l_{i_2});\ u_j^{i_3}(\overline{N}), w_j^{j_3}(\overline{N})\ (i_3 \in I_3, j = 1,2,...)\}$

вычисляются одновременно и при их вычислении операции выполняются по мере готовности, то реализация называется Q-эффективной. Если алгоритм допускает распараллеливание, то его Q-эффективная реализация полностью использует ресурс параллелизма алгоритма, поэтому является максимально параллельной реализаций алгоритма. Реализация алгоритма α называется выполнимой, если одновременно необходимо выполнять конечное число операций.

Расширенная модель концепции Q-детерминанта. Модель концепции Q-детерминанта позволяет исследовать машинно-независимые свойства численных алгоритмов, но не учитывает особенности их выполнения на параллельных ВС, например, зависимость реализаций алгоритмов от потерь, возникающих при работе с памятью. В связи с этим расширим модель концепции Q-детерминанта путем добавления двух подмоделей, которые представляют собой модели параллельных вычислений, отражающие в абстрактной форме архитектуру целевых ВС: ВС с общей и ВС с распределенной памятью. Исходную модель Q-детерминанта будем называть базовой.

В качестве модели параллельных вычислений, ориентированной на ВС с общей памятью, будем использовать PRAM (Parallel Random Access Machine) [8], которая характеризуется следующими особенностями: существует набор однотипных процессоров; все процессоры имеют общий блок памяти и могут взаимодействовать только через общую память; блок доступа к памяти соединяет процессоры с общей памятью. Издержки на синхронизацию процессоров не учитываются. Все процессоры могут выполнять самостоятельные операции из множества Q над частью данных в одну единицу времени. Это может приводить к одновременному обращению разных процессоров к одной и той же ячейке памяти. Такая ситуация делает модель нереалистичной, так как на физическом уровне доступ к одной и той же ячейке упорядочивается модулями памяти. Для решения данной проблемы были введены варианты модели, зависящие от допустимых комбинаций одновременного выполнения операций чтения и записи одной ячейки памяти. Для организации параллельных вычислений на многопроцессорных системах с общей памятью в настоящее время

наиболее широко применяется технология OpenMP. Эту технологию будем применять при разработке параллельных программ на основе расширенной модели концепции Q-детерминанта для ВС с общей памятью.

В качестве модели параллельных вычислений, ориентированной на ВС с распределенной памятью, будем использовать модель BSP (Bulk-Synchronous Parallel) [9]. В ней каждый процессор имеет локальную память. Процессоры реализуют взаимодействия с помощью односторонних обменов сообщениями. Параллельный алгоритм в рамках модели BSP представляется как последовательность *супершагов* (superstep), на каждом из которых процессоры выполняют локальные вычисления, обмены сообщениями и барьерную синхронизацию. Модель позволяет выразить суммарное время выполнения всех обменов процессорами на фиксированном супершаге.

Одним из часто используемых в параллельном и распределенном программировании фреймворков является парадигма «мастер-рабочие» [10]. В рамках расширенной модели Q-детерминанта для подмодели, ориентированной на распределенную память, в данном исследовании ограничимся парадигмой «мастер-рабочие» с конфигурацией, включающей одного мастера и множество рабочих, которая является наиболее популярной. В этом случае рассматриваемая модель характеризуется следующими свойствами: имеется множество однородных процессорных узлов с общей памятью; процессорные узлы соединены сетью, по которой можно передавать данные от одного узла другому; среди процессорных узлов выделяется, по крайней мере, один, называемый узлом-мастером, будем обозначать его M, остальные узлы, по крайней мере один, называются узлами-рабочими, будем обозначать их P. В соответствии с парадигмой «мастер-рабочие» вычислительный процесс разбивается на несколько супершагов: 1) инициализация, супершаг завершается барьерной синхронизацией; 2) посылка задания от узла M всем узлам P; 3) вычисление на каждом узле P без обменов с другими узлами; 4) посылка результатов от всех узлов P узлу M, супершаг завершается барьерной синхронизацией; 5) слияние на узле M полученных результатов; 6) завершение, вывод результатов. Супершаги 2 – 5 могут образовывать цикл.

Технология MPI (Message Passing Interface) [11] де-факто является стандартом для параллельного программирования на распределенной памяти. Эту технологию будем применять при разработке параллельных программ на основе расширенной модели концепции Q-детерминанта для ВС с распределенной памятью.

Реализация ресурса параллелизма численных алгоритмов. Метод проектирования параллельной программы для Q-эффективной реализации алгоритма, использующий расширенную модель концепции Q-детерминанта, состоит из трех этапов:

Этап 1. Построение Q-детерминанта алгоритма.

Этап 2. Описание плана выполнения Q-эффективной реализации алгоритма.

Этап 3. Разработка программы для выполнения Q-эффективной реализации алгоритма в случае, если Q-эффективная реализация выполнима.

Первые два этапа метода используют базовую модель концепции Q-детерминанта, третий этап осуществляется в рамках подмоделей расширенной модели, ориентированных на ВС с общей и распределенной памятью. Полученную с помощью описанного метода программу будем называть Q-эффективной.

Метод апробирован на алгоритмах, Q-детерминанты которых содержат Q-термы различного типа: алгоритм умножения плотных и разреженных матриц, метод Гаусса-Жордана и метод Якоби для решения СЛАУ. Q-эффективные программы разработаны для общей и распределенной памяти. Они используют ресурс параллелизма алгоритмов полностью. Для разработки Q-эффективных программ использовался язык программирования C++. Для общей памяти применялась технология OpenMP, для распределенной MPI и OpenMP. Анализировались динамические характеристики программ. Исследование проводилось на суперкомпьютере «Торнадо ЮУрГУ». Программы для общей памяти выполнялись на одном процессорном узле.

Рассмотрим процесс разработки Q-эффективной программы на примере метода Гаусса-Жордана.

Этап 1. Предположим, что система линейных уравнений $A\bar{x}=\bar{b}$ такова, что $A=\left[a_{ij}\right]_{i,\,j=1,\ldots,n}$ – матрица размерности $n\times n$ с ненулевым определителем, $\bar{x}=(x_1,\ldots,x_n)^T$, $\bar{b}=(a_{1,n+1},\ldots,a_{n,n+1})^T$. Пусть $\bar{A}=\left[a_{ij}\right]_{i,=1,\ldots,n;\,j=1,\ldots,n+1}$ – расширенная матрица системы. Метод Гаусса-Жордана состоит из n шагов.

*Шаг 1.*В качестве ведущего элемента выберем первый ненулевой элемент первой строки $a_{1j_1}(1\le j_1\le n)$. Получим расширенную матрицу $\bar{A}^{j_1}=\left[a_i^{j_1}{}_j\right]_{i=1,\ldots,n;\,j=1,\ldots,n+1}$.

Шаг $k\ (2 \le k \le n)$. После шага $(k-1)$ будет получена расширенная матрица $\overline{A}^{j_1 \dots j_{k-1}}$. В качестве ведущего элемента выберем первый ненулевой элемент строки k $a_{kj_k}^{j_1 \dots j_{k-1}} (1 \le j_k \le n)$. Получим расширенную матрицу $\overline{A}^{j_1 \dots j_k} = \left[a_{ij}^{j_1 \dots j_k} \right]_{i=1,\dots,n;\ j=1,\dots,n+1}$.

После шага n имеем систему уравнений $A^{j_1 \dots j_n} \overline{x} = \overline{b}^{j_1 \dots j_n}$, где $A^{j_1 \dots j_n} = \left[a_{ij}^{j_1 \dots j_n} \right]_{i,\ j=1,\dots,n}$, $\overline{b}^{j_1 \dots j_n} = (a_{1,n+1}^{j_1 \dots j_n}, \dots, a_{n,n+1}^{j_1 \dots j_n})^T$. Введем обозначения: $L_{j_1} = \underset{j=1}{\overset{j_1-1}{\wedge}} (a_{1j} = 0)$, если $j_1 \ne 1$; $L_{j_1} = true$, если $j_1 = 1$; $L_{jl} = \underset{j=1}{\overset{j_l-1}{\wedge}} (a_{lj}^{j_1 \dots j_{l-1}} = 0)$, если $j_l \ne 1$; $L_{j1} = true$, если $j_l = 1$ $(l = 2, \dots, n)$. Перестановки элементов $(1, \dots, n)$ могут быть пронумерованы. Пусть i – номер перестановки $(j_1, \dots, j_n)$. Тогда $u_i = L_{j_1} \wedge (a_{1j_1} \ne 0) \wedge (\underset{l=2}{\overset{n}{\wedge}} (L_{j_l} \wedge (a_{lj_l}^{j_1 \dots j_{l-1}} \ne 0)))$ и $w_i^{j_l} = a_{l,n+1}^{j_1 \dots j_n}$ $(l = 1, \dots, n)$ являются безусловными Q-термами. Q-детерминант метода Гаусса-Жордана состоит из n условных Q-термов, а представление в форме Q-детерминанта имеет вид $x_j = \{(u_1, w_1^j), \dots, (u_{n!}, w_{n!}^j)\}$ $(j = 1, \dots, n)$.

Этап 2. Опишем план выполнения Q-эффективной реализации метода Гаусса-Жордана. В соответствии с определением Q-эффективной реализации все безусловные Q-термы $u_i, w_i^j (i = 1, \dots, n!; j = 1, \dots, n)$ должны вычисляться одновременно. Это требование приводит к тому, что должно выполняться параллельное вычисление Q-термов $u_i (i = 1, \dots, n!)$ и параллельное вычисление матриц $\overline{A}^{j_1}, \overline{A}^{j_1 j_2}, \dots, \overline{A}^{j_1 j_2 \dots j_n}$ для любых возможных значений $j_1, j_2, \dots, j_n$. Первый цикл вычислений состоит в том, что одновременно начинают вычисляться Q-термы $u_i (i = 1, \dots, n!)$ и матрицы $\overline{A}^{j_1}$. Вычисление u_i начинается с подвыражений $L_{j_1} \wedge (a_{1j_1} \ne 0) (j_1 = 1, \dots, n)$, так как только их операции готовы к выполнению. Только одно из этих подвыражений будет иметь значение *true*. Соответствующее ему значение j_1 обозначим r_1. Далее прекращается вычисление u_i и $\overline{A}^{j_1}$, для которых $j_1 \ne r_1$. Во втором цикле одновременно вычисляются Q-термы u_i, для которых $j_1 = r_1$ и матрицы

$\overline{A}^{r_1 j_2}(j_2 = 1,...,n; j_2 \neq r_1)$. При вычислении u_i нужно вычислять только подвыражения $L_{j_2} \wedge (a_{2 j_2}^{r_1} \neq 0)(j_2 = 1,...,n; j_2 \neq r_1)$, так как только их операции готовы к выполнению. Только одно из этих подвыражений будет иметь значение *true*. Соответствующее ему значение j_2 обозначим r_2. Далее прекращается вычисление $\overline{A}^{r_1 j_2}(j_2 = 1,...,n; j_2 \neq r_1)$ и $u_i (i = 1,...,n!)$, для которых $j_2 \neq r_2$. Следующие $(n-3)$ цикла вычислений выполняются аналогично. В заключение нужно вычислить единственную матрицу $\overline{A}^{r_1 ... r_{n-1} j_n}(j_n \neq r_1, r_2, ..., r_{n-1})$, так как j_n имеет единственное значение, обозначим его r_n. В результате получим решение системы линейных уравнений $x_{r_j} = a_{j,n+1}^{r_1 ... r_n}(j = 1,...,n)$. Q-эффективная реализация метода Гаусса-Жордана выполнима.

Этап 3. Полученный на этапе 2 план выполнения Q-эффективной реализации используется для разработки параллельной программы, предназначенной для ВС с общей памятью. При использовании распределенной памяти план выполнения необходимо дополнить планом распределения вычислений по узлам P. Для метода Гаусса-Жордана он заключается в том, что каждая матрица $\overline{A}^{j_1}(j_1 = 1,...,n)$ и соответствующий ей Q-терм u_i должны вычисляться на своем узле P. Если количество узлов P меньше n, то узел P может выполнять вычисления для нескольких значений j_1. Узлы P получают от узла M информацию для вычисления матриц $\overline{A}^{j_1}$ и соответствующих Q-термов u_i. Результаты вычисления r_1 и $\overline{A}^{r_1}$ посылаются на узел M. Каждая матрица $\overline{A}^{r_1 j_2}(j_2 = 1,...,n; j_2 \neq r_1)$ и соответствующий ей Q-терм u_i должны вычисляться на своем узле P. Узлы P получают от узла M информацию для вычисления матриц $\overline{A}^{r_1 j_2}(j_2 = 1,...,n; j_2 \neq r_1)$ и соответствующих Q-термов u_i. Результаты вычисления r_2 и $\overline{A}^{r_1 r_2}$ посылаются на узел M. Следующие $(n-3)$ цикла вычислений выполняются аналогично. Последний цикл состоит в вычислении единственной матрицы $\overline{A}^{r_1 r_2 ... r_n}$, поэтому может выполняться на узле M.

Заключение. Q-детерминант делает численный алгоритм прозрачным с точки зрения структуры и реализации. В частности, он позволяет выразить имеющий параллелизм алгоритма и показать возможный способ его параллельного исполнения. В результате становится возможным получать эффективные реализации алгоритма для реальных параллельных ВС.

Библиографический список

1. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб.: БХВ-Петербург, 2002. – 608 с.

2. Антонов, А.С. Первая версия Открытой энциклопедии свойств алгоритмов / А.С. Антонов, Вад. В. Воеводин, Вл.В. Воеводин, А.М. Теплов, А.В. Фролов // Сборник «Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции» (31 марта - 2 апреля 2015 г., г. Екатеринбург). – Челябинск: Издательский центр ЮУрГУ, 2015. – С. 31–42.

3. Воеводин, Вл.В. Параллельные алгоритмы под микроскопом [Электронный ресурс] / Вл.В. Воеводин. – Режим доступа: http://omega.sp.susu.ru/books/conference/PaVT2016/talks/Voevodin.pdf [Дата обращения: 3 июля 2017 года].

4. Вальковский, В.А. Синтез параллельных программ и систем на вычислительных моделях / В.А. Вальковский, В.Э. Малышкин. – Новосибирск: Наука, 1988. – 128 с.

5. Malyshkin, Victor E. Distributed Algorithm of Data Allocation in the Fragmented Programming / Victor E. Malyshkin, Vladislav A. Perepelkin, Georgy F. Schukin // Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V.Malyshkin (Ed.). – Springer International Publishing Switzerland, 2015. – LNCS 9251. – P. 80–85.

6. Легалов, А. И. Функциональный язык для создания архитектурно-независимых параллельных программ / А. И. Легалов // Вычислительные технологии, – 2005. – № 1 (10). – С. 71–89.

7. Алеева, В.Н. Анализ параллельных численных алгоритмов: Препринт № 590 / В.Н. Алеева. – Новосибирск: ВЦ СО АН СССР, 1985. – 23 с.

8. McColl, W.F. General Purpose Parallel Computing / W.F. McColl // Lectures on Parallel Computation, Cambridge International Series on Parallel Computation. – USA: Cambridge University Press, 1993. – P. 337–391.

9. Valiant, L.G. A bridging model for parallel computation / L.G. Valiant // Communications of the ACM, – 1990. – Vol. 33, № 8. – P. 103–111.

10. Leung, J.Y.-T. Scheduling problems in master-slave model / H. Zhao // Annals of OperationsResearch, – 2008. – Vol. 159, № 1. – P. 215–231.

11. Gropp, W. Using MPI: portable parallel programming with the messagepassing interface / W. Gropp, E. Lusk, A. Skjellum. – Second Edi. – MIT Press, 1999. – 371 p.

METADATA

Title: Extended model of the concept of Q-determinant and its application for realization the parallelism resource of numerical algorithms.

Authors: V.A. Aleeva, N.V. Valkevich, Yu.S. Lapteva, D.E. Tarasov.

Abstract: We describe an extended model of the Q-determinant concept. We show how to use that model for realization the parallelism resource of numerical algorithms on parallel computing systems with shared and distributed memory. Results of research can be used to increase the efficiency of implementing numerical algorithms on parallel computing systems.

Key words: Q-determinant of algorithm; algorithm representation as Q-determinant; Q-effective algorithm implementation; parallelism resource of algorithm; parallel computing system; parallel program; Q-effective program.