УДК 004.021, 004.032.24, 004.051, 004.272 DOI: 10.14529/cmse210304

ПРИМЕНЕНИЕ КОНЦЕПЦИИ Q-ДЕТЕРМИНАНТА ДЛЯ ЭФФЕКТИВНОЙ РЕАЛИЗАЦИИ ЧИСЛЕННЫХ АЛГОРИТМОВ НА ПРИМЕРЕ МЕТОДА СОПРЯЖЕННЫХ ГРАДИЕНТОВ ДЛЯ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ

© 2021 В.Н. Алеева, М.Б. Шатов

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: aleevavn@susu.ru, charming.flurry@yandex.ru

Поступила в редакцию: 24.05.2021

Проблема повышения эффективности параллельных вычислений чрезвычайно актуальна. В статье продемонстрировано применение концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений. Концепция Q-детерминанта основана на унифицированном представлении численных алгоритмов в форме Q-детерминанта. Любой численный алгоритм имеет Q-детерминант. Q-детерминант состоит из Q-термов. Их число равно числу выходных данных алгоритма. Каждый Q-терм описывает все возможные способы вычисления одного из выходных данных на основе входных данных. Q-детерминант позволяет выразить и оценить внутренний параллелизм алгоритма, а также показать способ его параллельного исполнения. В работе приведены основные понятия концепции Q-детерминанта, необходимые для понимания приведенного исследования. Также описан основанный на концепции Q-детерминанта метод проектирования эффективных программ для численных алгоритмов. Результатом применения метода является программа, полностью использующая ресурс параллелизма алгоритма. Такая программа называется Q-эффективной. В качестве применения метода проектирования Q-эффективных программ описано проектирование программ для реализации метода сопряженных градиентов на параллельных вычислительных системах с общей и распределенной памятью. Приведены также результаты экспериментального исследования разработанных программ, проведенного с помощью суперкомпьютера «Торнадо ЮУрГУ».

Ключевые слова: повышение эффективности параллельных вычислений, Q-детерминант алгоритма, представление алгоритма в форме Q-детерминанта, Q-эффективная реализация алгоритма, ресурс параллелизма алгоритма, Q-эффективная программа.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алеева В.Н., Шатов М.Б. Применение концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 56–71. DOI: 10.14529/cmse210304.

Введение

Основной целью параллельных вычислений является ускорение решения вычислительных задач. Для достижения этой цели используется распараллеливание алгоритмов, применяемых для решения задач. Чем больше реализация алгоритма использует ресурс параллелизма алгоритма, тем большее ускорение решения задачи она обеспечивает и тем она эффективнее. Самая эффективная реализация алгоритма использует ресурс параллелизма алгоритма полностью. Для ее выполнения требуется одновременно применять больше вычислителей (процессоров, ядер) параллельной вычислительной системы (ПВС), чем для выполнения любой другой реализации алгоритма, так как одновременно нужно выполнять

большее количество операций. Вычислительные мощности ПВС растут, что способствует эффективной реализации алгоритмов. Однако, на практике, как правило, параллельные программы выполняют не самые эффективные реализации алгоритмов, следовательно, не применяют вычислительные ресурсы ПВС настолько, насколько допускают алгоритмы, а это приводит к потере ускорения решения вычислительных задач. Таким образом, проблема эффективной реализации алгоритмов на ПВС является актуальной.

Целью исследования является демонстрация применения концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений. Для достижения цели решаются следующие задачи:

1) построение Q-детерминанта алгоритма, выполняющего метод сопряженных градиентов;
2) описание эффективной реализации алгоритма, выполняющего метод сопряженных градиентов;
3) разработка эффективных программ для метода сопряженных градиентов, предназначенных для ПВС с общей и распределенной памятью.

Статья относится к направлению исследований, представленному работами [1, 2, 8–12], и вносит вклад в развитие данного направления.

Статья организована следующим образом. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 приведены некоторые основные понятия концепции Q-детерминанта, используемые в статье. В разделе 3 изложен метод проектирования Q-эффективных программ. В разделе 4 описано применение метода проектирования Q-эффективных программ для эффективной реализации алгоритма, выполняющего метод сопряженных градиентов. В разделе 5 представлены результаты экспериментального исследования Q-эффективных программ, реализующих метод сопряженных градиентов. которые были разработаны в последнее время, а также приведен обзор Q-эффективных программ, разработанных ранее. Заключение содержит краткое изложение полученных результатов, выводы об их применении и описание одного из перспективных направлений дальнейшего исследования.

1. Обзор работ по теме исследования

Приведем обзор работ по теме исследования ресурса параллелизма численных алгоритмов и его реализации. Таких работ, использующих универсальные подходы, очень мало.

Во-первых, отметим работы [3, 23], где есть очень важные и развитые исследования параллельной структуры алгоритмов и программ для их реализации на ПВС. Для исследования используются графы алгоритмов. Эти исследования адаптированы в открытой энциклопедии свойств алгоритмов AlgoWiki [5, 13]. При определении и реализации ресурса параллелизма алгоритмов используется индивидуальный подход к каждому алгоритму. Программное обеспечение для анализа ресурса параллелизма алгоритмов не рассматривается. Не рассматривается также единый для алгоритмов метод проектирования параллельных программ, использующих весь ресурс параллелизма алгоритмов.

Во-вторых, предлагаются подходы к разработке параллельных программ. Это привело к созданию различных языков параллельного программирования и инструментов. T-система [20] — одна из таких разработок. Она является инструментом для программирования, обеспечивающим автоматическое динамическое распараллеливание программ. Однако в работах по данному направлению исследований не показано, что параллельные програм-

мы, созданные с использованием Т-системы, полностью используют ресурс параллелизма алгоритмов. Синтез параллельных программ — это еще один подход к созданию параллельных программ. Он заключается в разработке новых параллельных алгоритмов с использованием базы знаний параллельных алгоритмов для решения более сложных задач. Технология фрагментированного программирования, язык ее реализации и система программирования LuNA разработаны на основе метода синтеза параллельных программ [7]. Такой подход не решает проблему исследования ресурса параллелизма алгоритмов и использования его в полной мере. Для снятия ограничений на ресурсы ПВС используется проектирование параллельных программ с помощью функционального языка программирования, который не зависит от архитектуры ПВС. Примером такого направления исследований может служить работа [15]. Однако здесь также нет обоснования, что созданные программы используют весь ресурс параллелизма реализуемых алгоритмов.

В-третьих, существует множество исследований, заключающихся в разработке параллельных программ для конкретных алгоритмов или для конкретной архитектуры ПВС. Например, к таким исследованиям относятся [17, 18, 21, 24]. Подобные исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС определенной архитектуры, но они не обеспечивают общего универсального подхода.

Приведенный обзор показывает, что существующие подходы при решении проблемы исследования и использования ресурса параллелизма алгоритмов либо малоэффективны, либо неприменимы, либо не являются универсальными. Возможно, данный обзор не является полным, так как он основан на доступных авторам источниках. Вместе с тем, как мы отмечали ранее, ресурс параллелизма алгоритмов при реализации на ПВС используется не полностью. Этот факт дает основание сделать вывод, что либо приемлемое решение проблемы исследования и использования ресурса параллелизма алгоритмов разработано, но не достаточно широко известно, поэтому не применяется, либо его нет.

2. Некоторые основные понятия концепции Q-детерминанта

Рассмотрим алгоритмическую проблему $\bar{y} = F(N, B)$, где $N = \{n_1, \ldots, n_k\}$ — множество параметров размерности проблемы или N — пустое множество, B — множество входных данных, $\bar{y} = \{y_1, \ldots, y_m\}$ — множество выходных данных, при этом целое число m является либо константой, либо значением вычислимой функции параметров N при условии, что $N \neq \varnothing$. Здесь n_i $(i \in \{1, \ldots, k\})$ равно любому положительному целому числу. Если $N = \{n_1, \ldots, n_k\}$, то через $\bar{N} = \{\bar{n}_1, \ldots, \bar{n}_k\}$ обозначим набор из k положительных целых чисел, где $\bar{n}_i$ — некоторое заданное значение параметра n_i для каждого $i \in \{1, \ldots, k\}$. Через $\{\bar{N}\}$ обозначим множество всех возможных k-наборов $\bar{N}$. Пусть $\mathfrak{A}$ — алгоритм для решения алгоритмической проблемы, Q — набор операций, используемых алгоритмом $\mathfrak{A}$.

Определение 1. Определим выражение над B и Q, как терм в стандартном смысле математической логики [4].

Определение 2. Мы называем выражение цепочкой длины n, если оно является результатом применения некоторой ассоциативной операции из Q к n выражениям.

Определение 3. Если $N = \varnothing$, то любое выражение w над B и Q мы называем безусловным Q-термом. Пусть $N \neq \varnothing$ и V — множество всех выражений над B и Q. Тогда любое отображение $w : \{\bar{N}\} \to V \cup \varnothing$ также называется безусловным Q-термом.

Определение 4. Пусть $N = \varnothing$ и w — безусловный Q-терм. Предположим, что выражение w над B и Q имеет значение логического типа при любой интерпретации переменных B. Тогда безусловный Q-терм w называется безусловным логическим Q-термом. Пусть $N \neq \varnothing$ и w — безусловный Q-терм. Если выражение $w(\bar{N})$ для каждого $\bar{N} \in \{\bar{N}\}$ имеет значение логического типа при любой интерпретации переменных B, то безусловный Q-терм w называется безусловным логическим Q-термом.

Определение 5. Пусть $u_1, \ldots, u_l$ — безусловные логические Q-термы, $w_1, \ldots, w_l$ — безусловные Q-термы. Тогда множество l пар $(\widehat{u}, \widehat{w}) = \{(u_i, w_i)\}_{i \in \{1,\ldots,l\}}$ называется условным Q-термом длины l.

Определение 6. Пусть $(\widehat{u}, \widehat{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ — счетное множество пар безусловных Q-термов. Предположим, что $\{(u_i, w_i)\}_{i \in \{1,\ldots,l\}}$ — условный Q-терм для любого $l < \infty$. Тогда мы называем $(\widehat{u}, \widehat{w})$ условным бесконечным Q-термом.

Опишем нахождение значения безусловного Q-терма w при интерпретации переменных B. Если $N = \varnothing$, то нахождение значения выражения w означает нахождение значения безусловного Q-терма w при любой интерпретации переменных B. Если $N \neq \varnothing$ и $w(\bar{N}) \neq \varnothing$, то $w(\bar{N})$ является выражением над B и Q. Можно найти значение выражения $w(\bar{N})$ при любой интерпретации переменных B. Конечно, мы опускаем значение $w(\bar{N}) = \varnothing$. Следовательно, мы находим значение безусловного Q-терма w при любой интерпретации переменных B. Теперь опишем нахождение значения условного Q-терма $(\widehat{u}, \widehat{w})$ при интерпретации переменных B. Пусть $N = \varnothing$. Находим значения выражений u_i, w_i для $i \in \{i = 1, \ldots, l\}$. При нахождении значений мы можем найти пару u_{i_0}, w_{i_0} такую, что u_{i_0} имеет значение **true**. Следовательно, мы можем найти значение w_{i_0}. Тогда считаем, что $(\widehat{u}, \widehat{w})$ имеет значение w_{i_0}. В противном случае считаем, что значение $(\widehat{u}, \widehat{w})$ при интерпретации переменных B не определено. Пусть $N \neq \varnothing$ и $\bar{N} \in \{\bar{N}\}$. Находим выражения $u_i(\bar{N}), w_i(\bar{N})$ для $i \in \{i = 1, \ldots, l\}$. При нахождении значений мы можем найти пару $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ такую, что $u_{i_0}(\bar{N})$ имеет значение **true**. Следовательно, мы можем найти значение $w_{i_0}(\bar{N})$. Тогда мы считаем, что $(\widehat{u}, \widehat{w})$ имеет значение $w_{i_0}(\bar{N})$. В противном случае считаем, что значение $(\widehat{u}, \widehat{w})$ для $\bar{N}$ и при такой интерпретации переменных B не определено. Аналогично можно определить значение условного бесконечного Q-терма.

Определение 7. Пусть $M = \{1, \ldots, m\}$. Предположим, что алгоритм $\mathfrak{A}$ состоит в нахождении для каждого $i \in M$ значения y_i путем вычисления значения Q-терма f_i. Тогда набор Q-термов $\{f_i \mid i \in M\}$ называется Q-детерминантом алгоритма $\mathfrak{A}$. Система уравнений $\{y_i = f_i \mid i \in M\}$ называется представлением алгоритма $\mathfrak{A}$ в форме Q-детерминанта.

Определение 8. Процесс вычисления Q-термов $\{f_i \mid i \in M\}$ алгоритма $\mathfrak{A}$ называется реализацией алгоритма $\mathfrak{A}$. Реализация алгоритма $\mathfrak{A}$ называется параллельной, если существуют операции, которые выполняются одновременно.

Определение 9. Реализация алгоритма $\mathfrak{A}$ называется Q-эффективной, если Q-термы $\{f_i \mid i \in M\}$ вычисляются одновременно, операции при их вычислении выполняются по мере готовности, при этом, если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания.

Замечание 1. Определение Q-эффективной реализации показывает, что она полностью использует ресурс параллелизма алгоритма.

Ресурс параллелизма алгоритма характеризуют его высота и ширина. Эти понятия рассматриваются в работах [2, 9–11]. В [2, 11] описана программная Q-система, разработанная для автоматизированного исследования ресурса параллелизма алгоритмов, а также сравнения ресурсов параллелизма алгоритмов, решающих одну и ту же алгоритмическую проблему.

Определение 10. Реализация алгоритма $\mathfrak{A}$ называется выполнимой, если одновременно должно выполняться конечное (непустое) множество операций.

Замечание 2. Существуют алгоритмы, Q-эффективная реализация которых невыполнима. Пример такого алгоритма приведен в работе [2].

3. Метод проектирования Q-эффективных программ

Блок-схема численного алгоритма позволяет разработать последовательную программу, реализующую алгоритм. Аналогично, используя представление численного алгоритма в форме Q-детерминанта, можно разработать параллельную программу, реализующую представленный алгоритм. Эта идея лежит в основе метода проектирования Q-эффективных программ. Модель концепции Q-детерминанта, которую мы называем базовой, позволяет исследовать только машинно-независимые свойства алгоритмов. Поэтому базовая модель была расширена, чтобы учесть особенности реализации алгоритмов на реальных ПВС. Расширенная модель концепции Q-детерминанта получена путем добавления моделей параллельных вычислений: PRAM [19] для общей памяти и BSP [22] для распределенной памяти. Метод проектирования Q-эффективных программ использует расширенную модель концепции Q-детерминанта и состоит из этапов:

1) построение Q-детерминанта алгоритма;
2) описание Q-эффективной реализации алгоритма;
3) разработка параллельной программы для выполнимой Q-эффективной реализации алгоритма.

На первых двух этапах метода используется базовая модель концепции Q-детерминанта, а на третьем этапе — расширенная модель. Программа, полученная с помощью данного метода, была названа Q-эффективной, а процесс ее разработки Q-эффективным программированием. Так как Q-эффективная программа выполняет Q-эффективную реализацию алгоритма, то она полностью использует ресурс параллелизма алгоритма. Таким образом, Q-эффективная программа имеет самый высокий параллелизм среди программ, реализующих алгоритм.

Для разработки Q-эффективной программы для общей памяти достаточно описания Q-эффективной реализации алгоритма. При разработке Q-эффективной программы для распределенной памяти следует учитывать, что данные должны обладать свойством локальности, иначе могут возникнуть многочисленные промахи кэша при их считывании из памяти, что приведет к снижению быстродействия. Обеспечить локальность данных позволяет распределение вычислений между вычислительными узлами ПВС, основанное на описании Q-эффективной реализации алгоритма. В нашем исследовании для распределения вычислений между вычислительными узлами применяется принцип «master-slave» [16]. При этом мы используем один вычислительный узел «master» и несколько вычислительных узлов «slave». Узел «master» обозначается буквой M, а множество узлов «slave» — буквой S. В этом исследовании при разработке Q-эффективных программ используется язык про-

граммирования $C++$, технология OpenMP для ПВС с общей памятью, технологии MPI и OpenMP для ПВС с распределенной памятью.

Дальнейшим развитием модели концепции Q-детерминанта может быть добавление моделей параллельных вычислений для вычислительных систем с другими архитектурными особенностями. Также для создания Q-эффективных программ можно использовать различные языки программирования и технологии параллельного программирования. Таким образом, для одного численного алгоритма существует потенциально бесконечное множество Q-эффективных программ, каждая из которых создается и используется в рамках определенной вычислительной инфраструктуры. По-видимому, из всех Q-эффективных программ для данного алгоритма не существует лучшей по производительности, но каждая из этих программ является наиболее эффективной для той вычислительной инфраструктуры, для которой она создавалась.

Более подробно метод проектирования Q-эффективных программ описан в работе [8].

4. Применение метода проектирования Q-эффективных программ для эффективной реализации метода сопряженных градиентов

Метод сопряженных градиентов для решения систем линейных уравнений [14] широко применяется на практике, поэтому его исследование с целью эффективной реализации является актуальным. Опишем метод сопряженных градиентов. Предположим, нужно решить систему линейных уравнений $A\vec{x} = \vec{b}$, где $A = [a_{ij}]_{i,j=1,\dots,n}$ — симметричная положительно определенная матрица, $\vec{x} = (x_1, \dots, x_n)^T$ и $\vec{b} = (b_1, \dots, b_n)^T$. Пусть $\vec{x}^0 = (x_1^0, \dots, x_n^0)^T$ — начальное приближение решения системы. Процесс решения системы можно представить как минимизацию функционала $(A\vec{x}, \vec{x}) - 2(\vec{b}, \vec{x})$. Минимизируя данный функционал с использованием подпространств Крылова, получаем алгоритм $\mathfrak{B}$, реализующий метод сопряженных градиентов.

Итерационный процесс алгоритма $\mathfrak{B}$ можно записать так:

$$\vec{r}^0 = \vec{b} - A\vec{x}^0, \tag{1}$$

$$\vec{z}^0 = \vec{r}^0, \tag{2}$$

$$\vec{x}^k = \vec{x}^{k-1} + \alpha_k \vec{z}^{k-1}, \text{ где } k \in \{1, 2, \dots\}, \tag{3}$$

$$\alpha_k = (\vec{r}^{k-1}, \vec{r}^{k-1})/(A\vec{z}^{k-1}, \vec{z}^{k-1}), \tag{4}$$

$$\vec{r}^k = \vec{r}^{k-1} - \alpha_k A\vec{z}^{k-1}, \tag{5}$$

$$\beta_k = (\vec{r}^k, \vec{r}^k)/(\vec{r}^{k-1}, \vec{r}^{k-1}), \tag{6}$$

$$\vec{z}^k = \vec{r}^k + \beta_k \vec{z}^{k-1}. \tag{7}$$

В результате применения метода решение системы линейных уравнений может быть достигнуто на итерации n, если отсутствует погрешность вычислений. Если реализовывать метод на ПВС, то погрешность вычислений существует, поэтому для получения решения количество итераций будет больше n. В качестве критерия остановки итерационного процесса будем использовать выполнение условия

$$\|\vec{r}^k\|/\|\vec{b}\| < \epsilon, \tag{8}$$

где $\|\vec{r}^k\|/\|\vec{b}\|$ — относительная невязка на k-м шаге итерации, ϵ — точность вычислений.

Продемонстрируем применение метода проектирования Q-эффективных программ для эффективной реализации алгоритма $\mathfrak{B}$.

Этап 1. Q-детерминант алгоритма $\mathfrak{B}$ состоит из n условных бесконечных Q-термов, а представление алгоритма $\mathfrak{B}$ в форме Q-детерминанта имеет вид

$$x_i = \{(u^0, x_i^0), (u^1, x_i^1), \dots, (u^k, x_i^k), \dots\}, \text{ где } i \in \{1, \dots, n\},$$

$u^l = \|\vec{r}^l\|/\|\vec{b}\| < \epsilon$ для любого $l \in \{0, 1, \dots\}$.

Этап 2. Опишем Q-эффективную реализацию алгоритма $\mathfrak{B}$. В соответствии с определением Q-эффективной реализации все Q-термы, составляющие Q-детерминант, должны вычисляться одновременно, при этом их операции должны выполняться по мере готовности. Во-первых, нужно вычислить Q-терм u^0. Если значение u^0 **true**, то вычисление заканчивается, так как получено решение с заданной точностью, при этом n-кортеж $\{x_i = x_i^0\}_{i\in\{1,\dots,n\}}$ является решением. Если значение u^0 **false**, то должно продолжаться вычисление Q-термов u^1 и x_i^1 для всех $i \in \{1, \dots, n\}$ одновременно. Предположим, что вычисление продолжается на шаге итерации $k \geq 1$, при этом вычислены Q-термы u^k и x_i^k для всех $i \in \{1, \dots, n\}$ и значением u^k является **true**. Тогда вычисление должно быть завершено, так как получено решение с заданной точностью, при этом решением является n-кортеж $\{x_i = x_i^k\}_{i\in\{1,\dots,n\}}$. Если значение u^k **false**, то должно продолжаться вычисление Q-термов u^{k+1} и x_i^{k+1} для всех $i \in \{1, \dots, n\}$ одновременно.

Приведем описание процесса вычисления Q-термов u^l и x_i^l для всех $l \in \{0, 1, \dots\}$ и $i \in \{1, \dots, n\}$. Вычисление u^0 состоит в вычислении по формуле (1) одновременно с вычислением $\|\vec{b}\|$, а затем по формуле (8) при $k = 0$. Мы видим, что вычисление $\vec{x}^1$ должно состоять из вычислений по формулам (4) при $k = 1$, а затем (3) при $k = 1$ с учетом того, что $\vec{z}^0 = \vec{r}^0$. Вычисление u^1 состоит в последовательном вычислении по формулам (5) при $k = 1$, а затем (8) при $k = 1$. Для подготовки второй итерации должно быть выполнено вычисление по формуле (6) при $k = 1$, а затем (7) при $k = 1$. Вычисление $\vec{x}^l$ для всех $l \in \{2, 3, \dots\}$ должно состоять из вычислений по формулам (4) при $k = l$, (3) при $k = l$, применяемым последовательно. Вычисление u^l для всех $l \in \{2, 3, \dots\}$ состоит в вычислении по формулам (5) при $k = l$, а затем (8) при $k = l$. Для подготовки следующей $(l + 1)$-й итерации должно быть выполнено вычисление по формуле (6) при $k = l$, а затем (7) при $k = l$. При вычислении по формулам операции должны выполняться по мере готовности к выполнению. При этом операции цепочки должны выполняться по схеме сдваивания.

Q-эффективная реализация алгоритма $\mathfrak{B}$ выполнима, так как при реализации одновременно необходимо выполнять конечное число операций.

Этап 3. Описание Q-эффективной реализации алгоритма $\mathfrak{B}$, полученное на втором этапе метода, можно использовать для разработки Q-эффективной программы для общей памяти. Опишем процесс реализации алгоритма $\mathfrak{B}$ на ПВС с распределенной памятью, применяя принцип «master-slave».

Каждая компонента вектора $\vec{r}^0$ (см. (1)) вычисляется на отдельном узле S. Если количество узлов S меньше n, то каждый из узлов S должен выполнять вычисления для нескольких компонент вектора $\vec{r}^0$. Перед вычислением r_i^0, где $i \in \{1, \dots, n\}$ узел S получает от узла M компоненту b_i вектора $\vec{b}$, строку матрицы A с номером i и вектор $\vec{x}^0$. Одновременно с вычислением вектора $\vec{r}^0$ вычисляется значение $\|\vec{b}\|$ на узле M . Вычислен-

ный вектор $\vec{r}^0$ передается на узел M. На узле M выполняется вычисление Q-терма u^0. Если значение u^0 **true**, то вычисление заканчивается.

Для определения коэффициента α_1 (см. (4)) вычисление $A\vec{z}^0$, учитывая, что $\vec{z}^0 = \vec{r}^0$, выполняется на узлах S по аналогии с вычислением $A\vec{x}^0$. Для этого узел M передает на узлы S вектор $\vec{r}^0$. Одновременно с операцией $A\vec{z}^0$ выполняется вычисление скалярного произведения $(\vec{r}^0, \vec{r}^0)$ на узле M. Полученные на узлах S результаты, которые являются компонентами вектора $A\vec{z}^0$, передаются на узел M для вычисления скалярного произведения векторов $A\vec{z}^0$ и $\vec{z}^0$. Завершает вычисление α_1 операция деления на узле M. После этого выполняется вычисление приближения решения $\vec{x}^1$ (см. (3)) на узле M.

Одновременно с вычислением $\vec{x}^1$ на узле M на некотором одном узле S вычисляется $\vec{r}^1$ (см. (5)). Использовать несколько узлов S в этом случае нецелесообразно. Перед вычислением $\vec{r}^1$ узел M передает на узел S вектор $\vec{r}^0$, коэффициент α_1, вектор $A\vec{z}^0$ и результат скалярного произведения $(\vec{r}^0, \vec{r}^0)$. На этом же узле S вычисляется коэффициент β_1 (см. (6)). Для этого находится скалярное произведение $(\vec{r}^1, \vec{r}^1)$ и выполняется операция деления. После этого на узле S вычисляется также вектор $\vec{z}^1$ (см. (7)). Вычисленные на узле S значения $\vec{r}^1$, $(\vec{r}^1, \vec{r}^1)$, β_1 и $\vec{z}^1$ передаются на узел M. На узле M вычисляется Q-терм u^1. Если значение u^1 **true**, то вычисление заканчивается.

Для определения коэффициента α_2 (см. (4)) вычисление $A\vec{z}^1$ выполняется аналогично вычислению $A\vec{z}^0$. Для этого узел M передает на узлы S вектор $\vec{z}^1$. Полученные на узлах S компоненты вектора $A\vec{z}^1$ передаются на узел M для вычисления скалярного произведения векторов $A\vec{z}^1$ и $\vec{z}^1$. Вычислять скалярное произведения $(\vec{r}^1, \vec{r}^1)$ не нужно, так как оно было вычислено в процессе вычисления коэффициента β_1. Завершает вычисление α_2 операция деления на узле M. Вычисление $\vec{x}^2$ (см. (3)) выполняется на узле M.

Одновременно с вычислением $\vec{x}^2$ вычисляется $\vec{r}^2$ (см. (5)) на одном из узлов S. Перед вычислением $\vec{r}^2$ узел M передает на узел S вектор $\vec{r}^1$, коэффициент α_2, вектор $A\vec{z}^1$, результат скалярного произведения $(\vec{r}^1, \vec{r}^1)$ и вектор $\vec{z}^1$. Коэффициент β_2 (см. (6)) вычисляется на этом же узле S. Для этого выполняется скалярное произведение $(\vec{r}^2, \vec{r}^2)$ и операция деления. Затем на узле S вычисляется вектор $\vec{z}^2$ (см. (7)). Вычисленные на узле S значения $\vec{r}^2$, $(\vec{r}^2, \vec{r}^2)$, β_2 и $\vec{z}^2$ передаются на узел M. На узле M выполняется вычисление Q-терма u^2. Если значение u^2 **true**, то вычисление заканчивается.

Все последующие итерации алгоритма $\mathfrak{B}$ выполняются по аналогии со второй итерацией.

5. Разработка и экспериментальное исследование Q-эффективных программ

Для выполнения Q-эффективной реализации алгоритма $\mathfrak{B}$ были разработаны Q-эффективные программы для общей памяти и для распределенной памяти с применением принципа «master-slave», проектирование которых описано в разделе 4. Также была разработана Q-эффективная программа для распределенной памяти, не использующая принцип «master-slave», и две программы, которые не являются Q-эффективными, при этом первая из них предназначена для общей памяти, а вторая для распределенной памяти с применением принципа «master-slave». Программы, не являющиеся Q-эффективными, выполняют реализацию алгоритма $\mathfrak{B}$, отличающуюся от Q-эффективной реализации тем, что вычисления по формулам (5), (6) и (7) выполняются последовательно.

Для всех разработанных программ было проведено экспериментальное исследование динамических характеристик — времени выполнения и ускорения. Оно проводилось на суперкомпьютере «Торнадо» Южно-Уральского государственного университета. Для общей памяти был использован один вычислительный узел, для распределенной памяти — несколько вычислительных узлов. В экспериментах были задействованы два центральных процессора вычислительных узлов с многоядерными ускорителями Intel Xeon X5680 с частотой 3.33 GHz, каждый из которых имеет 6 ядер и поддерживает 12 потоков, оперативная память узла 24 Гб ECC DDR3 Full buffered [6]. При экспериментальном исследовании находилось решение системы линейных уравнений с плотной матрицей A, имеющей размер $n = 1000$, которая была сгенерирована с помощью специально разработанного программного обеспечения. Компоненты векторов $\vec{b}$ и $\vec{x}^0$ формировались путем генерации случайных чисел.

На рис. 1 показаны графики времени выполнения и ускорения программ для общей памяти. Рисунок 2 содержит графики времени выполнения и ускорения программ для распределенной памяти.

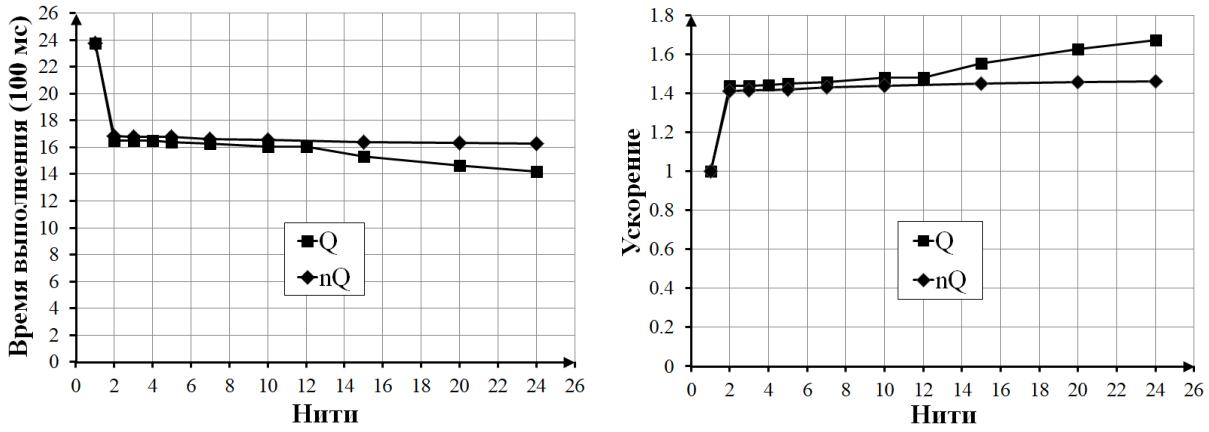

Рис. 1. Время выполнения и ускорение программ для общей памяти (Q — Q-эффективная программа, nQ — не Q-эффективная программа)

Поясним результаты экспериментального исследования.

На динамические характеристики Q-эффективной программы влияет множество факторов. Однако главное влияние оказывает ресурс параллелизма алгоритма, который программа выполняет. Ресурсы параллелизма у алгоритмов разные. Существуют алгоритмы, все операции которых могут быть выполнены параллельно. Примером такого алгоритма является алгоритм сложения двух векторов. Но есть также алгоритмы, которые не имеют параллельных реализаций, например, алгоритм, Q-детерминант которого состоит из одного безусловного Q-терма, имеющего вид

$$(\dots((((a_1 + a_2) \times a_3 + a_4) \times a_5 + a_6) \times a_7 + a_8)\dots) \times a_{2n-1} + a_{2n},$$

где n — параметр размерности. Подавляющее большинство алгоритмов имеют ресурс параллелизма, находящийся между этими двумя крайностями. Любой подход к проектированию параллельных программ основывается на распараллеливании выполняемых программой алгоритмов в рамках их ресурса параллелизма. Следовательно, значения динамических характеристик проектируемых программ из-за разных ресурсов параллелизма выполняемых алгоритмов будут разными.

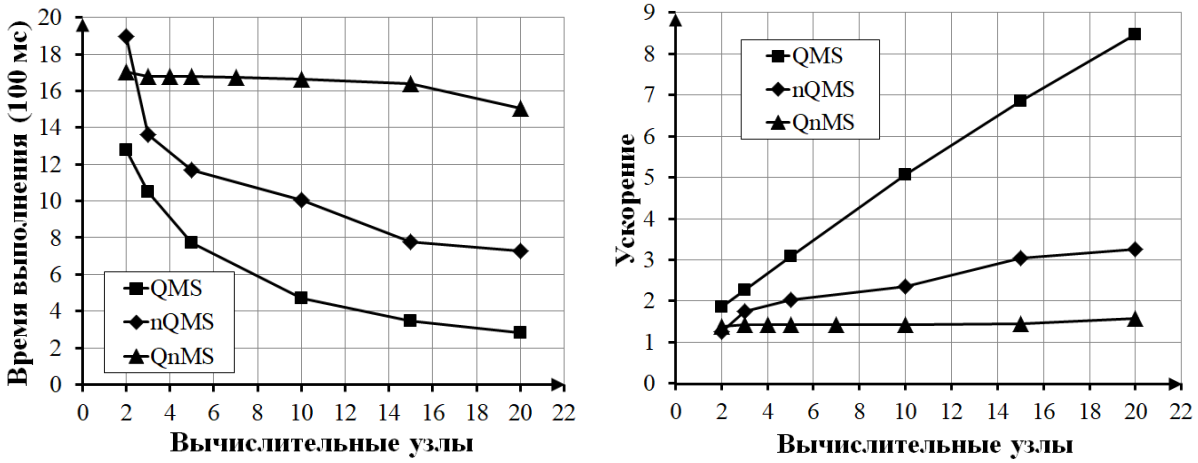

Рис. 2. Времея выполнения и ускорение программ для распределенной памяти (QMS — Q-эффективная программа, использующая принцип «master-slave», nQMS — не Q-эффективная программа, использующая принцип «master-slave», QnMS — Q-эффективная программа, не использующая принцип «master-slave»)

Проводя экспериментальное исследование Q-эффективных программ, можно лишь констатировать, каковы значения их динамических характеристик, но улучшить эти значения, не меняя используемую вычислительную инфраструктуру, то есть условия разработки и исполнения программ, не удастся. С помощью данного исследования получены значения динамических характеристик Q-эффективных программ, выполняющих алгоритм $\mathfrak{B}$, который реализует метод сопряженных градиентов. Если эти значения не устраивают пользователя, то для решаемой алгоритмической проблемы возможно с помощью программной Q-системы [2, 11] подобрать алгоритм с лучшим ресурсом параллелизма, чем алгоритм $\mathfrak{B}$.

Кроме ресурса параллелизма выполняемого параллельной программой алгоритма отметим и другие факторы, которые могут существенно влиять на динамические характеристики программы.

Следует подчеркнуть, что, как известно, время выполнения параллельных программ увеличивают пересылки информации между вычислительными узлами ПВС, а также синхронизация вычислений.

Если при реализации на ПВС Q-эффективной программе выделено не достаточное количество вычислительных ресурсов, в нашем случае вычислительных ядер, узлов, чтобы могла быть выполнена Q-эффективная реализация алгоритма, то значения динамических характеристик ухудшаются. Разработанные в данном исследовании Q-эффективные программы используют ресурс параллелизма алгоритма $\mathfrak{B}$ полностью, но выделенных ресурсов ПВС не достаточно для выполнения Q-эффективной реализации, поэтому полученные значения динамических характеристик всех трех Q-эффективных программ Q (рис. 1), QMS и QnMS (рис. 2) хуже, чем могли бы быть, если бы ресурсов было достаточно. Особенно нехватка ресурсов ПВС повлияла на время выполнения и ускорение Q-эффективной программы Q для общей памяти. Отметим, что на динамические характеристики программ nQ (рис. 1) и nQMS (рис. 2), не являющихся Q-эффективными, нехватка ресурсов ПВС также оказала влияние.

На рис. 2 можно видеть, что Q-эффективная программа QMS для распределенной памяти с применением принципа «master-slave» демонстрирует наивысшее среди соперников ускорение. Объясним этот факт.

Лучшие динамические характеристики программы QMS по сравнению с программой QnMS, по нашему мнению, можно объяснить, в частности, тем, что при разработке программы QMS обеспечивается свойство локальности данных за счет распределения вычислений по узлам, а при разработке программы QnMS нет, в результате чего возникают многочисленные промахи кэша при считывании данных из памяти. Обеспечение локальности данных при параллельных вычислениях имеет важное значение, так как повышает быстродействие программ.

Если программа, не являющаяся Q-эффективной, и Q-эффективная программа имеют одинаковые условия разработки и исполнения, то динамические характеристики программы, не являющейся Q-эффективной, хуже, чем Q-эффективной, так как она использует не весь ресурс параллелизма алгоритма. Результаты экспериментального исследования программ QMS и nQMS (рис. 2), а также программ Q и nQ (рис. 1) подтверждают это.

Все три Q-эффективные программы Q (рис. 1), QMS и QnMS (рис. 2) соответствуют разным вычислительным инфраструктурам. С помощью экспериментального исследования мы установили, что эти Q-эффективные программы имеют разное быстродействие. Но даже при влиянии на динамические характеристики перечисленных выше факторов каждая из Q-эффективных программ является наиболее эффективной для своей вычислительной инфраструктуры.

Применение метода проектирования параллельных программ для Q-эффективной реализации алгоритмов с Q-детерминантами различной структуры рассмотрено ранее в ряде выпускных квалификационных работ студентов Южно-Уральского государственного университета. Их обзор приведен в [9]. В этих работах были исследованы алгоритмы умножения плотных и разреженных матриц, метод Гаусса—Жордана, метод Якоби для решения систем линейных уравнений. Рассмотрен также метод прогонки для решения систем трехточечных уравнений, имеющий малый ресурс параллелизма. В этом случае не следует использовать распределенную память, поскольку это может привести к снижению производительности. Также рассмотрен метод Фурье для решения системы разностных уравнений, обладающий большим ресурсом параллелизма. Его можно эффективно реализовать на распределенной памяти, используя принцип «master-slave». Следует отметить, что исследования, приведенные в работах, показали, что для некоторых алгоритмов использовать принцип «master-slave» не целесообразно, так как это может привести к увеличению обменов между вычислительными узлами. Примером в этом случае является алгоритм для реализации метода Якоби для решения систем пятиточечных разностных уравнений. Исследования показали также, что некоторые алгоритмы имеют сложные, но хорошо структурированные Q-детерминанты. Например, такие алгоритмы реализуют метод Гаусса—Жордана и метод Гаусса—Зейделя для решения систем линейных уравнений. Аналогичными свойствами обладают и многие другие известные алгоритмы. Хорошо структурированные Q-детерминанты алгоритмов упрощают создание Q-эффективных программ для этих алгоритмов.

Заключение

В статье показано применение метода проектирования Q-эффективных программ, основанного на концепции Q-детерминанта, для эффективной реализации численных алгорит-

мов на примере метода сопряженных градиентов для решения систем линейных уравнений. Для этого были решены следующие задачи:

1) построение Q-детерминанта алгоритма, выполняющего метод сопряженных градиентов;
2) описание Q-эффективной реализации алгоритма, выполняющего метод сопряженных градиентов;
3) разработка Q-эффективных программ для метода сопряженных градиентов, предназначенных для ПВС с общей и распределенной памятью, и их экспериментальное исследование.

Данное исследование пополнило коллекцию алгоритмов, для которых разработаны Q-эффективные программы и оценены их динамические характеристики.

Применение метода проектирования Q-эффективных программ решает проблему наиболее полного использования ресурса параллелизма численных алгоритмов и тем самым делает возможной эффективную реализацию численных алгоритмов на ПВС. Программная Q-система для исследования ресурса параллелизма численных алгоритмов [2, 11], метод проектирования Q-эффективных программ и технология Q-эффективного программирования [9] в комплексе являются одним из решений проблемы повышения эффективности параллельных вычислений, использующих численные алгоритмы. Их могут применять разработчики программного обеспечения для проектирования эффективных программ, предназначенных для любых ПВС — от персональных компьютеров с многоядерными процессорами до суперкомпьютеров. Это приведет к уменьшению времени выполнения программного обеспечения и повышению быстродействия вычислительных систем.

Одним из перспективных направлений развития описанных в статье исследований является автоматизированное проектирование и исполнение Q-эффективных программ. Исследования по данному направлению уже ведутся.

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

Литература

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

2. Алеева В.Н., Зотова П.С., Склезнев Д.С. Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной Q-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205.

3. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

4. Ершов Ю.Л., Палютин Е.А. Математическая логика. М.: Наука, 1987. 336 с.

5. Открытая энциклопедия свойств алгоритмов. URL: `https://algowiki-project.org/ru` (дата обращения: 16.05.2021).

6. Суперкомпьютер «Торнадо ЮУрГУ». URL: `http://supercomputer.susu.ru/computers/tornado/` (дата обращения: 16.05.2021).

7. Akhmed-Zaki D., Lebedev D., Malyshkin V., et al. Automated Construction of High Performance Distributed Programs in LuNA System // Parallel Computing Technologies (PaCT 2019). Lecture Notes in Computer Science. Vol. 11657. 2019. P. 3–9. DOI: 10.1007/978-3-030-25636-4_1.

8. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant // Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. Vol. 965. 2019. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48.

9. Aleeva V.N. Improving Parallel Computing Efficiency // Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

10. Aleeva V.N., Aleev R.Zh. High-Performance Computing Using Application of Q-determinant of Numerical Algorithms // Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160.

11. Aleeva V., Bogatyreva E., Skleznev A., et al. Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. Vol. 1129. 2019. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.

12. Aleeva V.N., Sharabura I.S., Suleymanov D.E. Software System for Maximal Parallelization of Algorithms on the Base of the Conception of Q-determinant // Parallel Computing Technologies (PaCT 2015). Lecture Notes in Computer Science. Vol. 9251. 2015. P. 3–9. DOI: 10.1007/978-3-319-21909-7_1.

13. Antonov A.S., Dongarra J., Voevodin V.V. AlgoWiki Project as an Extension of the Top500 Methodology // Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.

14. Henk A. van der Vorst. Iterative Krylov Methods for Large Linear System. USA: Cambridge University Press, 2003. 221 p. DOI: 10.1017/CBO9780511615115.

15. Legalov A.I., Vasilyev V.S., Matkovskii I.V., et al. A Toolkit for the Development of Data-Driven Functional Parallel Programmes // Parallel Computational Technologies (PCT'2018). Communications in Computer and Information Science. Vol. 910. 2018. P. 16–30. DOI: 10.1007/978-3-319-99673-8_2.

16. Leung J.Y.-T., Zhao H. Scheduling problems in master-slave model // Annals of Operations Research. 2008. Vol. 159. P. 215–231. DOI: 10.1007/s10479-007-0271-4.

17. Li Y., Dou W., Yang K., et al. Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis // 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

18. Matveev S.A., Zagidullin R.R., Smirnov A.P., et al. Parallel Numerical Algorithm for Solving Advection Equation for Coagulating Particles // Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 2. P. 43–54. DOI: 10.14529/jsfi180204.

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

20. Moskovsky A., Roganov V., Abramov S. Parallelism Granules Aggregation with the T-System // Parallel Computing Technologies (PaCT 2007). Lecture Notes in Computer Science. Vol. 4671. 2007. P. 293–302. DOI: 10.1007/978-3-540-73940-1_30.

21. Prifti V., Bala R., Tafa I., et al. The time profit obtained by parallelization of quicksort algorithm used for numerical sorting // Science and Information Conference (SAI). 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

22. Valiant L.G. A bridging model for parallel computation // Communications of the ACM. 1990. Vol. 33, no. 8. P. 103–111. DOI: 10.1145/79173.79181.

23. Voevodin V.V., Voevodin Vl.V. The V-Ray technology of optimizing programs to parallel computers // Numerical Analysis and Its Applications (WNAA 1996). Lecture Notes in Computer Science. Vol. 1196. 1997. P. 546–556. DOI: 10.1007/3-540-62598-4_136.

24. You J., Kezhang H., Liang H., et al. Research on parallel algorithms for calculating static characteristics of electromagnetic relay // IEEE 11th Conference on Industrial Electronics and Applications (ICIEA). 2016. P. 1421–1425. DOI: 10.1109/ICIEA.2016.7603808.

Алеева Валентина Николаевна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Шатов Михаил Борисович, студент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse210304

APPLICATION OF THE Q-DETERMINANT CONCEPT FOR EFFICIENT IMPLEMENTATION OF NUMERICAL ALGORITHMS BY THE EXAMPLE OF THE CONJUGATE GRADIENT METHOD FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)
E-mail: aleevavn@susu.ru, charming.flurry@yandex.ru

Received: 24.05.2021

The problem of improving the efficiency of parallel computing is very topical. The article demonstrates the application of the concept of Q-determinant for the effective implementation of numerical algorithms by the example of the conjugate gradient method for solving systems of linear equations. The concept of the Q-determinant is based on a unified representation of numerical algorithms in the form of the Q-determinant. Any numerical algorithm has a Q-determinant. The Q-determinant consists of Q-terms. Their number is equal to the number of output data items. Each Q-term describes all possible ways to compute one of the output data items based on the input data. The Q-determinant allows you to express and evaluate the internal parallelism of the algorithm, as well as to show the method of its parallel execution. The article gives the main notions of the Q-determinant concept necessary for better understanding of our research. Also, we describe a method of designing effective programs for numerical algorithms on the base of the concept of the Q-determinant. As a result, we obtain the program which uses the parallelism resource of the algorithm completely, and this program is called Q-effective. As application of the method for design of Q-effective programs, we describe the designing programs for conjugate gradient method for implementation on parallel computing systems with shared and distributed memory. Finally, for developed programs we present the results of experiments on a supercomputer "Tornado SUSU".

Keywords: improving parallel computing efficiency, Q-determinant of algorithm, representation of algorithm in the form of Q-determinant, Q-effective implementation of algorithm, parallelism resource of algorithm, Q-effective program.

FOR CITATION

Aleeva V.N., Shatov M.B. Application of the Q-determinant Concept for Efficient Implementation of Numerical Algorithms by the Example of the Conjugate Gradient Method for Solving Systems of Linear Equations. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2021. Vol. 10, no. 3. P. 56–71. (in Russian) DOI: 10.14529/cmse210304.

References

1. Aleeva V.N. Analysis of Parallel Numerical Algorithms. Preprint no. 590. Novosibirsk, Computing Center of the Siberian Branch of the Academy of Sciences of the USSR, 1985. 23 p. (in Russian)
2. Aleeva V.N., Zotova P.S., Skleznev D.S. Advancement of research for the parallelism resource of numerical algorithms with help of software Q-system. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 2. P. 66–81. (in Russian) DOI: 10.14529/cmse210205.
3. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St. Petersburg, BHV-Petersburg, 2002. 608 p. (in Russian)
4. Ershov Yu.L., Palyutin E.A. Mathematical Logic. Moscow, Mir, 1984. 303 p.
5. Open Encyclopedia of Parallel Algorithmic Features. URL: `https://algowiki-project.org/en` (accessed: 16.05.2021).
6. "Tornado SUSU" Supercomputer. URL: `http://supercomputer.susu.ru/en/computers/tornado/` (accessed: 16.05.2021).
7. Akhmed-Zaki D., Lebedev D., Malyshkin V., et al. Automated Construction of High Performance Distributed Programs in LuNA System. Parallel Computing Technologies (PaCT 2019). Lecture Notes in Computer Science. Vol. 11657. 2019. P. 3–9. DOI: 10.1007/978-3-030-25636-4_1.
8. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant. Supercomputing. RuSCDays 2018. Communications in Computer and Information Science. Vol. 965. 2019. P. 565–577. DOI: 10.1007/978-3-030-05807-4_48.
9. Aleeva V.N. Improving Parallel Computing Efficiency. Proceedings – 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.
10. Aleeva V.N., Aleev R.Zh. High-Performance Computing Using Application of Q-determinant of Numerical Algorithms. Proceedings – 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. 8 p. Article number 8570160. DOI: 10.1109/GloSIC.2018.8570160.
11. Aleeva V., Bogatyreva E., Skleznev A., et al. Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. Vol. 1129. 2019. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.

12. Aleeva V.N., Sharabura I.S., Suleymanov D.E. Software System for Maximal Parallelization of Algorithms on the Base of the Conception of Q-determinant. Parallel Computing Technologies (PaCT 2015). Lecture Notes in Computer Science. Vol. 9251. 2015. P. 3–9. DOI: 10.1007/978-3-319-21909-7_1.

13. Antonov A.S., Dongarra J., Voevodin V.V. AlgoWiki Project as an Extension of the Top500 Methodology. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 1. P. 4–10. DOI: 10.14529/jsfi180101.

14. Henk A. van der Vorst. Iterative Krylov Methods for Large Linear System. USA, Cambridge University Press, 2003. 221 p. DOI: 10.1017/CBO9780511615115.

15. Legalov A.I., Vasilyev V.S., Matkovskii I.V., et al. A Toolkit for the Development of Data-Driven Functional Parallel Programmes. Parallel Computational Technologies (PCT'2018). Communications in Computer and Information Science. Vol. 910. 2018. P. 16–30. DOI: 10.1007/978-3-319-99673-8_2.

16. Leung J.Y.-T., Zhao H. Scheduling problems in master-slave mode. Annals of Operations Research. 2008. Vol. 159. P. 215–231. DOI: 10.1007/s10479-007-0271-4.

17. Li Y., Dou W., Yang K., et al. Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis. 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

18. Matveev S.A., Zagidullin R.R., Smirnov A.P., et al. Parallel Numerical Algorithm for Solving Advection Equation for Coagulating Particles. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 2. P. 43–54. DOI: 10.14529/jsfi180204.

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

20. Moskovsky A., Roganov V., Abramov S. Parallelism Granules Aggregation with the T-System. Parallel Computing Technologies (PaCT 2007). Lecture Notes in Computer Science. Vol. 4671. 2007. P. 293–302. DOI: 10.1007/978-3-540-73940-1_30.

21. Prifti V., Bala R., Tafa I., et al. The time profit obtained by parallelization of quicksort algorithm used for numerical sorting. Science and Information Conference (SAI). 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

22. Valiant L.G. A bridging model for parallel computation. Communications of the ACM. 1990. Vol. 33, no. 8. P. 103–111. DOI: 10.1145/79173.79181.

23. Voevodin V.V., Voevodin Vl.V. The V-Ray technology of optimizing programs to parallel computers. Numerical Analysis and Its Applications (WNAA 1996). Lecture Notes in Computer Science. Vol. 1196. 1997. P. 546–556. DOI: 10.1007/3-540-62598-4_136.

24. You J., Kezhang H., Liang H., et al. Research on parallel algorithms for calculating static characteristics of electromagnetic relay. IEEE 11th Conference on Industrial Electronics and Applications (ICIEA). 2016. P. 1421–1425. DOI: 10.1109/ICIEA.2016.7603808.