УДК 004.021, 004.032.24, 004.051, 004.272 **DOI: 10.14529/cmse230303**

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЕ И ИСПОЛНЕНИЕ ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ ЧИСЛЕННЫХ АЛГОРИТМОВ

© 2023 В.Н. Алеева

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

Поступила в редакцию: 29.06.2023

Проектировать эффективные параллельные программы для многопроцессорных архитектур сложно, так как нет четких формальных правил, которых необходимо придерживаться. Для решения этой проблемы при реализации численных алгоритмов может применяться концепция Q-детерминанта. Данная теория позволяет проводить автоматизированный анализ ресурса параллелизма алгоритма, автоматизированное сравнение ресурсов параллелизма алгоритмов, решающих одну и ту же алгоритмическую проблему, проектировать эффективные программы для реализации алгоритмов с помощью специально разработанного метода проектирования, повысить эффективность реализации численных методов и алгоритмических проблем. Результаты, полученные на основе концепции Q-детерминанта, представляют собой один из вариантов решения проблемы эффективной реализации численных алгоритмов, методов и алгоритмических проблем на параллельных вычислительных системах. Однако пока остается не решенной фундаментальная проблема автоматизированного проектирования и исполнения для любого численного алгоритма программы, реализующей алгоритм эффективно. В статье описана разработка единой для численных алгоритмов программной системы проектирования и исполнения Q-эффективных программ — эффективных программ, спроектированных с помощью концепции Q-детерминанта. Система предназначена для использования на параллельных вычислительных системах с общей памятью. Она состоит из компилятора и виртуальной машины. Компилятор преобразует представление алгоритма в форме Q-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью. Виртуальная машина исполняет программу, полученную с помощью компилятора. В статье также приведено экспериментальное исследование созданной программной системы с применением суперкомпьютера «Торнадо ЮУрГУ».

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

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алеева В.Н. Автоматизированное проектирование и исполнение эффективных программ для численных алгоритмов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 3. С. 31–49. DOI: 10.14529/cmse230303.

Введение

Концепция Q-детерминанта является одним из подходов к распараллеливанию численных алгоритмов. Впервые она была изложена в работе [1], ее развитие и применение описаны в работах [2, 3, 13–18]. Результаты, полученные на основе концепции Q-детерминанта, представляют собой один из вариантов решения проблемы эффективной реализации численных алгоритмов, численных методов и алгоритмических проблем на параллельных вычислительных системах (ПВС). Кратко их можно сформулировать так.

1) Разработана программная Q-система для исследования ресурса параллелизма алгоритмов [2, 14–17].

2) Предложен метод проектирования Q-эффективных программ для выполнения Q-эффективных реализаций алгоритмов, использующих ресурс параллелизма алгоритмов полностью [3, 13–16].
3) Описана технология Q-эффективного программирования для повышения эффективности реализации численных методов решения алгоритмических проблем и самих алгоритмических проблем [14, 15].

Пока остается не решенной проблема автоматизированного проектирования и исполнения для любого численного алгоритма программы, реализующей алгоритм эффективно.

Цель данного исследования заключается в том, чтобы показать возможность создания программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Для достижения цели решаются следующие задачи.

1) Проектирование и реализация компилятора, преобразующего представление алгоритма в форме Q-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью.
2) Проектирование и реализация виртуальной машины, исполняющей программу, полученную с помощью компилятора.
3) Проведение экспериментального исследования функционирования программной системы, состоящей из компилятора и виртуальной машины, с применением суперкомпьютера «Торнадо ЮУрГУ».

В решение данных задач внесли вклад студенты кафедры системного программирования ЮУрГУ Юферов А.В. (задачи 1 и 2) и Иванов А.Д. (задача 3).

Статья организована следующим образом. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 приведены используемые в статье понятия концепции Q-детерминанта. Раздел 3 содержит описание проектирования, а раздел 4 описание реализации программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. В разделе 5 представлены результаты функционального тестирования на персональном компьютере созданной программной системы и приведено ее экспериментальное исследование с применением суперкомпьютера «Торнадо ЮУрГУ». Заключение содержит краткое изложение полученных результатов, выводы об их значении и применении, описание дальнейших исследований.

1. Обзор работ по теме исследования

В сфере высокопроизводительных вычислений одним из ведущих ученых мира является Джек Донгарра (Jack Dongarra). На протяжении десятков лет производительность вычислительной техники росла в соответствии с законом Мура [25]. И если большинство программных средств не поспевало за аппаратными достижениями, то программное обеспечение высокопроизводительных вычислений справлялось с этим успешно. Во многом это объясняется использованием алгоритмов Дж. Донгарры. В университете Теннесси для поддержки библиотеки матрично-векторных операций DPLASMA [22] научным коллективом с участием Дж. Донгарры развивается подход к конструированию и исполнению параллельных программ на основе машинно-независимого представления прикладного алгоритма в виде бесконтурного ориентированного графа. Это позволяет обеспечивать высокопроизводительную реализацию прикладных алгоритмов благодаря планированию вычислений и динамической поддержке исполнительной системы PaRSEC [23]. Дж. Донгарра занимается

также вопросами автоматического распараллеливания алгоритмов [20]. Ассоциация вычислительной техники (ACM) присудила Джеку Донгарре премию Тьюринга за 2021 год.

В Российской Федерации одним из наиболее развитых направлений исследований в области параллельных вычислений является направление, созданное В.В. Воеводиным. Именно оно наиболее близко к направлению, использующему концепцию Q-детерминанта. Из работ, связанных с направлением исследований В.В. Воеводина, отметим монографию [4], где проводится очень важное и развитое исследование параллельной структуры алгоритмов и программ для их реализации на ПВС. Исследование параллельной структуры алгоритмов основано на описании и изучении информационных графов алгоритмов. Оно применяется в открытой энциклопедии AlgoWiki [9, 19]. Этот проект сформировался под руководством Вл. В. Воеводина и Дж. Донгарры. В энциклопедии описываются и изучаются графы конкретных алгоритмов. Это позволяет реализовывать алгоритмы эффективно. Однако в работах данного направления исследований программное обеспечение для автоматизированного исследования и использования ресурса параллелизма алгоритмов не рассматривается.

Еще один подход к эффективной реализации алгоритмов на ПВС разрабатывается в Стэндфордском университете под руководством А. Айкена (A. Aiken). Подход заключается в том, что отдельно описывается прикладной алгоритм и способ его отображения на вычислительные ресурсы ПВС. Научным коллективом, разрабатывающим подход, создана система Legion [21], реализующая этот подход. Недостатком системы является отсутствие автоматизации в конструировании способа отображения прикладного алгоритма на вычислительные ресурсы.

Существуют направления исследований, в которых предлагаются универсальные подходы к созданию параллельных программ. Одним из результатов таких исследований является Т-система [26]. Она обеспечивает среду программирования с поддержкой автоматического динамического распараллеливания программ. Вместе с тем нельзя утверждать, что параллельные программы, созданные с помощью Т-системы, полностью используют ресурс параллелизма реализуемого ими алгоритма, поскольку распараллеливаемые программы могут не содержать всех реализаций алгоритма, в частности, при их создании могла быть не учтена самая параллельная реализация. Метод синтеза параллельных программ — еще один подход к созданию параллельных программ. Он состоит в том, чтобы с помощью базы знаний параллельных алгоритмов конструировать параллельные алгоритмы для решения более сложных задач. На основе метода разработаны технология фрагментарного программирования, язык его реализации и система программирования LuNA [12]. Такой подход к созданию параллельных программ является универсальным, но не решает проблему автоматизированного исследования и использования ресурса параллелизма алгоритмов. Отметим еще одно направление исследований. Для преодоления ресурсных ограничений ПВС предлагаются методы построения параллельных архитектурно-независимых программ с использованием функционального языка программирования [24]. Однако не показано, что созданные программы используют весь ресурс параллелизма алгоритмов.

Многочисленными являются исследования, заключающиеся в разработке параллельных программ, учитывающих специфику алгоритмов, а также архитектуру ПВС. Их примерами являются [8, 11]. Такие исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС определенной архитектуры, однако они не обеспечивают универсального подхода к разработке эффективных параллельных программ.

Приведенный обзор, а также факт, что ресурс параллелизма алгоритмов при реализации на ПВС часто используется не полностью, по-видимому, доказывают, что в настоящее время нет признанных в научном сообществе и широко применяемых на практике решений задач исследования и использования ресурса параллелизма численных алгоритмов, тем более в автоматизированном режиме. Другими словами, проблема автоматизированного распараллеливания алгоритмов, проектирования и исполнения реализующих их эффективных программ не решена. По мнению автора перспективным для ее решения является подход, основанный на унифицированном представлении алгоритма, показывающем ресурс параллелизма в полной мере. Например, подход, основанный на концепции Q-детерминанта.

2. Теоретические основы разработки программной системы

Приведем понятия концепции Q-детерминанта, используемые в данном исследовании.

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 [5]. Каждое выражение w имеет уровень вложенности, обозначим его через T^w.

Пример 1. Выражения и их уровни вложенности:
1) $w_1 = b_1 \times (b_2 + b_3)/b_4$, $T^{w_1} = 3$:
2) $w_2 = (b_1 \geq b_2) \vee ((b_3 \times b_4) \leq b_5 \times (b_6 + b_7))$, $T^{w_2} = 4$.

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.

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.

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.

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.

Q-термы можно вычислять. Процесс нахождения значений Q-термов описан в [2, 15].

Определение 4. Пусть $M = \{1, \dots, m\}$. 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 5. 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.

Предположим, что U, C и I образуют разбиение множества $M = \{1, \dots, m\}$ с пустыми членами, то есть: $U \cup C \cup I = M$; $U \cap C = U \cap I = C \cap I = \varnothing$; кроме того, одно или два подмножества U, C и I могут быть пустыми. Предположим также, что подмножества U, C и I можно связать с подмножествами множества Q-термов $\{f_i \mid i \in M\}$ таким образом:

1) для каждого $i \in U$ существует Q-терм f_i, который является безусловным, и $f_i = w^i$;
2) для каждого $i \in C$ существует Q-терм f_i, который является условным, и $f_i = \{(u^i_j, w^i_j)\}_{j \in \{1,\dots,l(i)\}}$, где $l(i)$ — либо константа, либо значение вычислимой функции от N, если $N \neq \varnothing$;
3) для каждого $i \in I$ существует Q-терм f_i, который является условным бесконечным, и $f_i = \{(u^i_j, w^i_j)\}_{j \in \{1,2,\dots\}}$.

Определение 6. Опишем реализацию алгоритма $\mathfrak{A}$, называемую Q-эффективной.

Пусть $N = \varnothing$. Зададим интерпретацию переменных B. Будем вычислять выражения

$$W = \left\{w^i(i \in U); u^i_j, w^i_j(i \in C, j \in \{1, \dots, l(i)\}); u^i_j, w^i_j(i \in I, j \in \{1, 2, \dots\})\right\} \tag{1}$$

одновременно, параллельно. Мы говорим, что операция готова к выполнению, если вычислены значения всех ее операндов. При вычислении каждого из выражений W (см. (1)) мы выполняем операции, как только они готовы к выполнению. Если несколько операций цепочки готовы к выполнению, то они выполняются по схеме сдваивания. Например, вычисление цепочки $a_1 + a_2 + a_3 + a_4$ по схеме сдваивания выполняется так: сначала вычисляем $b_1 = a_1 + a_2$ и $b_2 = a_3 + a_4$ одновременно, затем $c = b_1 + b_2$. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ u^i_j имеет значение **false**, то вычисление соответствующего w^i_j прекращается. Если для $i \in C \cup I$ и $j \in \{1, 2, \dots\}$ вычисление пары (u^i_j, w^i_j) приводит к тому, что значение одного выражения не определено, то вычисление второго выражения прекращается. Если для $i \in C \cup I$ вычисление некоторой пары $(u^i_{j_0}, w^i_{j_0})$ приводит к определению их значений и значение $u^i_{j_0}$ **true**, то вычисление выражений u^i_j, w^i_j прекращается для любого $j \neq j_0$.

Теперь пусть $N \neq \varnothing$. Зададим интерпретацию переменных B и $\bar{N} \in \{\bar{N}\}$. Получаем множество выражений

$$W(\bar{N}) = \left\{w^i(\bar{N})(i \in U); u^i_j(\bar{N}), w^i_j(\bar{N})(i \in C, j \in \{1, \dots, l(i)\}); \right. \\ \left. u^i_j(\bar{N}), w^i_j(\bar{N})(i \in I, j \in \{1, 2, \dots\})\right\}. \tag{2}$$

Выражения $W(\bar{N})$ (см. (2)) вычисляются по аналогии с выражениями W (см. (1)).

Цепочки должны вычисляться по схеме сдваивания, так как она обеспечивает минимальное время вычисления цепочки. Обоснуем это утверждение. Высота каждого из алгоритмов, вычисляющих цепочку длины n, составляет не менее $\lceil \log n \rceil$ и не более $n - 1$. При

этом высота алгоритма, использующего схему сдваивания, равна $\lceil \log n \rceil$, т.е. является минимальной. Предположим, что программы, реализующие алгоритмы, имеют одинаковую вычислительную инфраструктуру, т.е. условия разработки и выполнения [3, 15]. Тогда меньше всего времени для выполнения последовательных инструкций понадобится программе для реализации алгоритма, использующего схему сдваивания. Следовательно, в соответствии с законом Амдала программа, реализующая алгоритм с использованием схемы сдваивания, будет иметь время выполнения не больше, чем каждая из остальных программ.

Определение 7. Реализация алгоритма $\mathfrak{A}$ называется выполнимой, если одновременно должно выполняться конечное (непустое) множество операций.

Q-эффективная реализация полностью использует ресурс параллелизма алгоритма, который опишем далее. Для этого будем использовать следующие условия и обозначения.

1. Алгоритм $\mathfrak{A}$ представлен в форме Q-детерминанта $y_i = f_i$ для всех $i \in M$.
2. Значения Q-термов f_i для всех $i \in M$ определяются при любой интерпретации переменных B и для любого $\bar{N} \in \{\bar{N}\}$, если $N \neq \varnothing$.
3. Пусть $N = \varnothing$ и $I \neq \varnothing$. Тогда при заданной интерпретации переменных B для любого $i \in I$ существует пара выражений $u^i_{j_i}, w^i_{j_i}$ такая, что значение $u^i_{j_i}$ равно **true** и значение $w^i_{j_i}$ определено. Введем обозначение

$$\widetilde{W} = \left\{w^i (i \in U); u^i_j, w^i_j (i \in C, j \in \{1, \ldots, l(i)\}); u^i_{j_i}, w^i_{j_i} (i \in I)\right\}.$$

4. Пусть $N \neq \varnothing$ и $I \neq \varnothing$. Тогда при заданной интерпретации переменных B для любого $\bar{N} \in \{\bar{N}\}$ и $i \in I$ существует пара выражений $u^i_{j_i}(\bar{N}), w^i_{j_i}(\bar{N})$ такая, что значение $u^i_{j_i}(\bar{N})$ равно **true**, а значение $w^i_{j_i}(\bar{N})$ определено. Введем обозначение

$$\widetilde{W}(\bar{N}) = \left\{w^i(\bar{N}) (i \in U); u^i_j(\bar{N}), w^i_j(\bar{N}) (i \in C, j \in \{1, \ldots, l(i)\}); u^i_{j_i}(\bar{N}), w^i_{j_i}(\bar{N}) (i \in I)\right\}.$$

5. Q-эффективная реализация алгоритма $\mathfrak{A}$ выполнима.

Определение 8. Обозначим через $D_{\mathfrak{A}}$ высоту и через $P_{\mathfrak{A}}$ ширину алгоритма $\mathfrak{A}$, характеризующие его ресурс параллелизма, и определим их следующим образом.

Если $N = \varnothing$, то

$$D_{\mathfrak{A}} = \begin{cases} \max\limits_{w \in W} T^w, & \text{если } I = \varnothing, \\ \max\limits_{w \in \widetilde{W}} T^w, & \text{если } I \neq \varnothing; \end{cases} \qquad P_{\mathfrak{A}} = \max_{1 \leq r \leq D_{\mathfrak{A}}} \sum_{w \in W} O^w_r,$$

где O^w_r — количество операций уровня вложенности r выражения w.

Если $N \neq \varnothing$, то

$$D_{\mathfrak{A}}(\bar{N}) = \begin{cases} \max\limits_{w(\bar{N}) \in W(\bar{N})} T^{w(\bar{N})}, & \text{если } I = \varnothing, \\ \max\limits_{w(\bar{N}) \in \widetilde{W}(\bar{N})} T^{w(\bar{N})}, & \text{если } I \neq \varnothing; \end{cases} \qquad P_{\mathfrak{A}}(\bar{N}) = \max_{1 \leq r \leq D_{\mathfrak{A}}(\bar{N})} \sum_{w(\bar{N}) \in W(\bar{N})} O^{w(\bar{N})}_r,$$

где $O^{w(\bar{N})}_r$ — количество операций уровня вложенности r выражения $w(\bar{N})$.

$D_{\mathfrak{A}}$ характеризует время выполнения Q-эффективной реализации алгоритма, а $P_{\mathfrak{A}}$ — количество вычислителей вычислительной системы (вычислительных ядер, процессоров), необходимое для выполнения Q-эффективной реализации алгоритма, а также масштабируемость алгоритма.

Определение 9. Программа называется Q-эффективной, если она выполняет Q-эффективную реализацию алгоритма.

В исследованиях на основе концепции Q-детерминанта для оценки эффективности параллельных программ используются их динамические характеристики: время выполнения, ускорение и эффективность. Ускорение программы вычисляется по формуле $S = T_1/T_p$, а эффективность по формуле $E = S/p$, где T_1 — время выполнения программы на одном вычислительном ядре процессора ПВС, T_p — время выполнения программы на p вычислительных ядрах одного или нескольких процессоров ПВС, p — количество используемых программой вычислительных ядер. Мы считаем, что программа эффективнее другой, если она имеет динамические характеристики лучше, чем другая программа.

Q-эффективная программа использует весь ресурс параллелизма алгоритма. Но когда она выполняется на ПВС, то возможно, что из-за нехватки вычислительных ресурсов ПВС не все операции будут выполняться по мере их готовности к выполнению. В результате будет выполняться реализация алгоритма, не использующая весь ресурс параллелизма алгоритма. Вместе с тем в статье [15, раздел 5.4] доказана следующая теорема. Предположим, что для численного алгоритма имеются Q-эффективная и не Q-эффективная программы с одинаковой вычислительной инфраструктурой. Тогда динамические характеристики Q-эффективной программы не уступают динамическим характеристикам не Q-эффективной программы. Следовательно, каждая из Q-эффективных программ эффективна для своей вычислительной инфраструктуры. Q-эффективная реализация получила свое название из-за этого факта. Этот факт подтвержден также экспериментально [3, 15]. Он убеждает в том, что разрабатывать и использовать Q-эффективные программы целесообразно.

3. Проектирование программной системы

Разработанный ранее метод проектирования параллельной программы для выполнения Q-эффективной реализации численного алгоритма [3, 13–16] может применять любой разработчик при проектировании для любого численного алгоритма программы, использующей ресурс параллелизма алгоритма полностью. Однако для этого нужно знать, что такое Q-детерминант алгоритма, и уметь его строить для конкретных алгоритмов. Но представим, что разработчику достаточно уметь описывать классическим способом численные алгоритмы, которые он применяет, например, с помощью блок-схемы, и иметь доступ к программному обеспечению, которое по описанию алгоритма получает Q-эффективную реализацию, а затем ее выполняет. По-видимому, такая возможность важна. Но можно ли ее реализовать? Подсистема программной Q-системы [2, 14–17] преобразует блок-схему алгоритма в его представление в форме Q-детерминанта, которое является унифицированным представлением численных алгоритмов. Для лучшего понимания приведем простой пример представления алгоритма в форме Q-детерминанта, полученного с помощью Q-системы.

Пример 2. Рассмотрим алгоритм скалярного произведения векторов

$$\vec{A} = (A(1), A(2), \ldots, A(8)) \text{ и } \vec{B} = (B(1), B(2), \ldots, B(8)),$$

использующий схему сдваивания. Его представление в форме Q-детерминанта имеет вид

$$S = ((A(1)\cdot B(1) + A(2)\cdot B(2)) + (A(3)\cdot B(3) + A(4)\cdot B(4)))+ \\ + ((A(5)\cdot B(5) + A(6)\cdot B(6)) + (A(7)\cdot B(7) + A(8)\cdot B(8))). \quad (3)$$

На рис. 1 показано представление в форме Q-детерминанта алгоритма скалярного произведения векторов $\vec{A}$ и $\vec{B}$, полученное с помощью Q-системы. Представление использует обозначения: «op» — операция, «f0» — первый операнд операции, «s0» — второй операнд операции. Уравнение (3) и рис. 1 описывают одно и то же вычисление.


```
S= ;{
     "op":"+","f0":{
                    "op":"+","f0":{
                                   "op":"+","f0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(1)","s0":"B(1)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(2)","s0":"B(2)"
                                                                 }
                                                  },
                                   "s0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(3)","s0":"B(3)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(4)","s0":"B(4)"
                                                                 }
                                                  }
                                   },
                    "s0":{
                                   "op":"+","f0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(5)","s0":"B(5)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(6)","s0":"B(6)"
                                                                 }
                                                  },
                                   "s0":{
                                                  "op":"+","f0":{
                                                                 "op":"*","f0":"A(7)","s0":"B(7)"
                                                                 },
                                                  "s0":{
                                                                 "op":"*","f0":"A(8)","s0":"B(8)"
                                                                 }
                                                  }
                                   }
                    }
}
```


Рис. 1. Представление в форме Q-детерминанта алгоритма скалярного произведения векторов длины 8

Используя представление алгоритмов в форме Q-детерминантов, можно разрабатывать единое для всех численных алгоритмов программное обеспечение. Примером является программное обеспечение Q-системы для получения Q-эффективной реализации любого численного алгоритма и вычисления на ее основе характеристик ресурса параллелизма алгоритма. Аналогично возможно разработать единое для всех численных алгоритмов программное обеспечение для получения Q-эффективной реализации алгоритма и проектирования на ее основе исполняемой программы для выполнения Q-эффективной реализации. Эта идея лежит в основе создания единой программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов.

Для создания программной системы мы использовали следующую постановку задачи. В состав программной системы должны входить два программных продукта:

1) компилятор — преобразователь представления алгоритма в форме Q-детерминанта в код программы;
2) виртуальная машина — исполнитель создаваемых компилятором программ.

На вход компилятор должен принимать два файла:

1) файл с представлением алгоритма в форме Q-детерминанта, полученный Q-системой;
2) текстовый файл с необходимыми для построения программы метаданными алгоритма, создаваемый пользователем системы.

В качестве выходных данных компилятор должен создавать бинарный файл с исполняемой виртуальной машиной программой. В заголовочной секции файл должен содержать информацию о входных и выходных данных алгоритма, получаемых компилятором из файла метаданных алгоритма, а также дополнительных параметрах, позволяющих виртуальной машине исполнять программу, например, объем необходимой алгоритму памяти.

Виртуальная машина в качестве входных параметров должна принимать:

1) исполняемую программу, созданную с помощью компилятора;
2) пути файлов входных и выходных данных исполняемого алгоритма.

Файл входных данных алгоритма готовит пользователь. Результатом работы виртуальной машины является файл со значениями выходных данных. Пользователь на своем рабочем компьютере может с помощью компилятора готовить программу для исполнения, а исполнение может осуществляться на персональном компьютере или ПВС. Виртуальная машина в данном исследовании должна быть ориентирована на ПВС с общей памятью.

При проектировании программной системы была разработана диаграмма размещения ее компонентов, показанная на рис. 2. Компонент «Q-система» представляет собой подси-

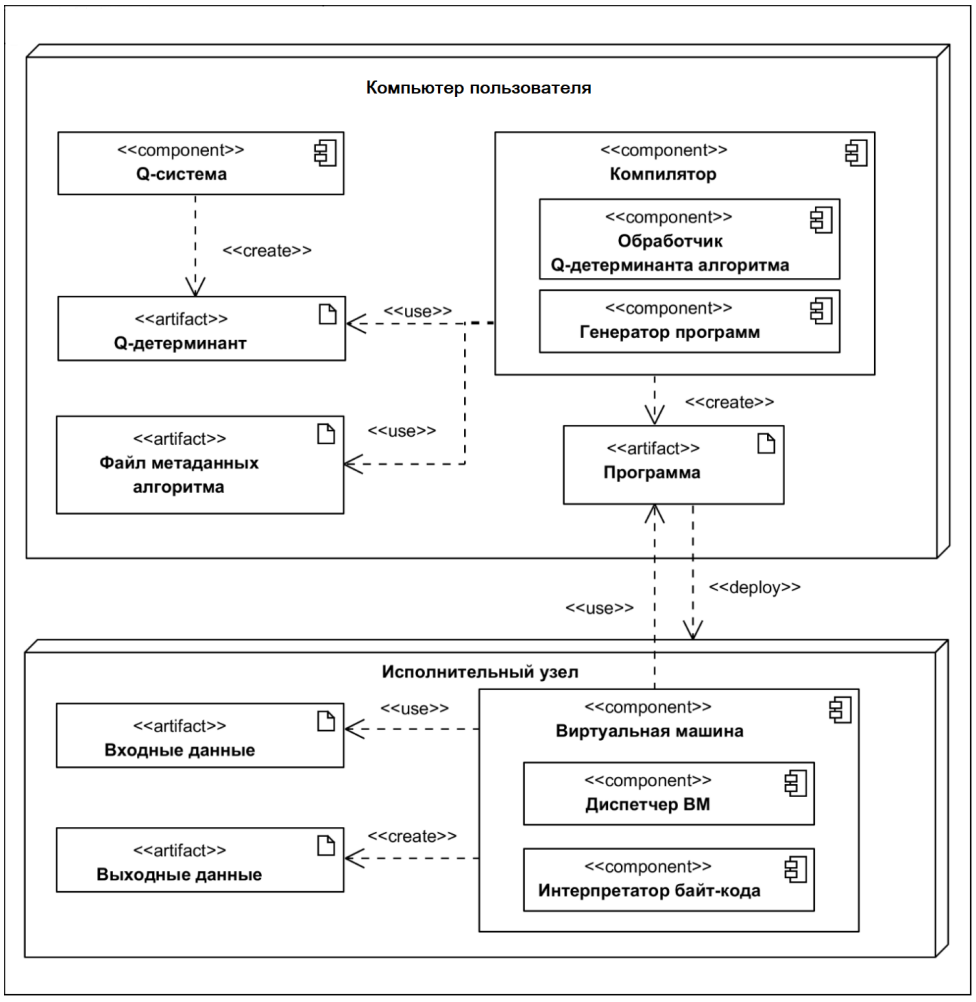


Рис. 2. Диаграмма размещения компонентов программной системы

стему Q-системы для формирования по блок-схеме алгоритма его представления в форме Q-детерминанта, которое обозначено на диаграмме артефактом «Q-детерминант». Компонент «Компилятор» преобразует представление алгоритма в форме Q-детерминанта в код программы. Артефакт «Файл метаданных алгоритма» — текстовый файл, создаваемый пользователем. Он используется компилятором для информации о входных данных алгоритма, их описания, а также текстового описания самого алгоритма. Подкомпонент «Обработчик Q-детерминанта алгоритма», используя Q-детерминант, строит в памяти структуру, содержащую операции Q-детерминанта и связи между ними. Назовем эту структуру деревом алгоритма. Подкомпонент «Генератор программ» записывает в бинарном виде метаданные алгоритма и сформированную с помощью дерева алгоритма очередь команд для выполнения Q-эффективной реализации алгоритма в файл программы, обозначенный артефактом «Программа». Компонент «Виртуальная машина» исполняет программу, созданную компонентом «Компилятор». Подкомпонент «Диспетчер ВМ» обеспечивает распределение задач по потокам исполнения, в которых работают объекты компонента «Интерпретатор байт-кода». Виртуальная машина получает поток команд из файла программы, а также читает входные данные алгоритма из файла, обозначенного артефактом «Входные данные». После окончания работы виртуальной машины полученные выходные данные записываются в файл, обозначенный артефактом «Выходные данные».

4. Реализация программной системы

Основной упор при выборе инструментов для реализации был сделан на кроссплатформенность, открытость исходного кода и производительность конечного решения, чтобы не потерять преимущество, которое предоставляет Q-эффективная реализация алгоритма. В качестве основного языка разработки был выбран язык $C{+}{+}$ из-за скорости его работы, гибкости и кроссплатформенности.

Рассмотрим некоторые основные решения по реализации программной системы.

Файл метаданных алгоритма, который использует компилятор, имеет формат YAML. Он содержит три корневых атрибута: description — описывающая строка, которая будет показана пользователю при выводе информации о программе виртуальной машиной; input parameters — список входных данных алгоритма; output parameters — список выходных данных алгоритма. Во время компиляции компилятор проверяет, что все индексы в пределах заданных в этом файле значений и при обнаружении выхода за пределы прерывает процесс с ошибкой.

В качестве формата файла программы используется разработанный бинарный формат, описанный далее. Таким образом, программы, создаваемые компилятором, имеют специфику. Поэтому, чтобы исполнять их, потребовалось разработать программное обеспечение, которое было названо виртуальной машиной. Файл программы включает заголовок, секции с дополнительными данными и секцию с командами. Заголовок схематично показан на рис. 3. В первой строке размещаются сигнатура данных и версия формата файла. Поле «Сигнатура данных» содержит константу, позволяющую однозначно идентифицировать тип данных. Для того чтобы убедиться, что файл является программой, которую виртуальная машина способна выполнить, реализация класса, читающего данные программы, проверяет значение сигнатуры данных. Пользователь, открыв программу в текстовом редакторе, по значению сигнатуры данных сможет понять, является ли файл программой. Поле «Версия формата файла» позволяет устанавливать версии файлов программ. Это

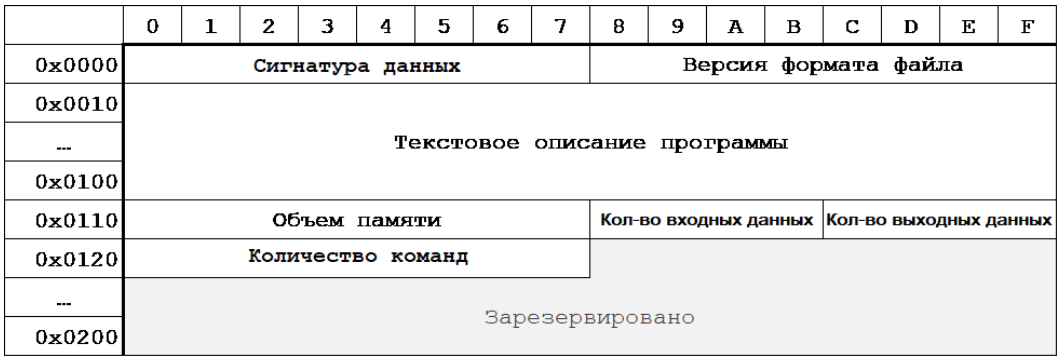


Рис. 3. Схема заголовка файла программы

необходимо для того, чтобы добавлять новые возможности в связку компилятор — виртуальная машина. Поле «Текстовое описание программы» содержит описание, показываемое пользователю в справке о программе. Компилятор получает значение этого поля из файла метаданных алгоритма из корневого параметра description. Поле «Объем памяти» содержит число, характеризующее необходимый для выделения объем памяти, измеряемый в ячейках памяти виртуальной машины. Значения полей «Кол-во входных данных» и «Кол-во выходных данных» компилятор рассчитывает автоматически на основе списков входных и выходных данных алгоритма из файла метаданных алгоритма. Поле «Количество команд» содержит общее число команд в программе.

После заголовка начинается раздел с метаинформацией о входных и выходных данных алгоритма. Каждое данное описывается с помощью структуры, схематически представленной на рис. 4. В файле сначала размещаются все метаданные о входных данных алгоритма, а затем все метаданные о выходных данных. Их количество известно из параметров «Кол-во входных данных» и «Кол-во выходных данных» заголовка. Поле «Обозначение (наимено-

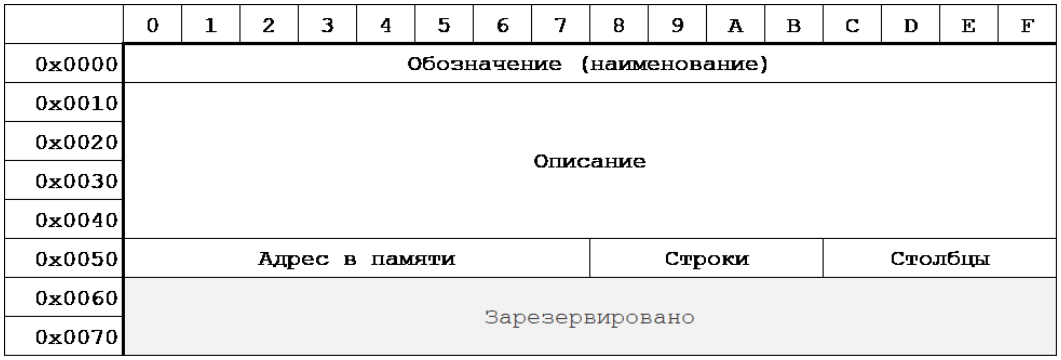


Рис. 4. Схема структуры метаданных о данных алгоритма в файле программы

вание)» содержит имя данного, использованное в представлении алгоритма в форме Q-детерминанта в виде строки. Содержимое поля «Описание» представляет текстовое описание данного, показываемое пользователю при выводе справочной информации о программе. Поле «Адрес в памяти» содержит адрес размещения данного. Если данное является матрицей, то это адрес первого элемента первой строки матрицы. Содержимое полей «Строки» и «Столбцы» является количеством строк и столбцов матрицы соответственно. Параметр «Адрес в памяти» компилятор рассчитывает автоматически, а значения остальных параметров получает из структур, описывающих эти данные в файле метаданных адгоритма.

Последним в файле программы идет раздел с очередью команд. Схема структуры команды показана на рис. 5. Число команд определяется параметром «Количество команд» из заголовка. Интерпретатору байт-кода отдаются две первые строки схемы.

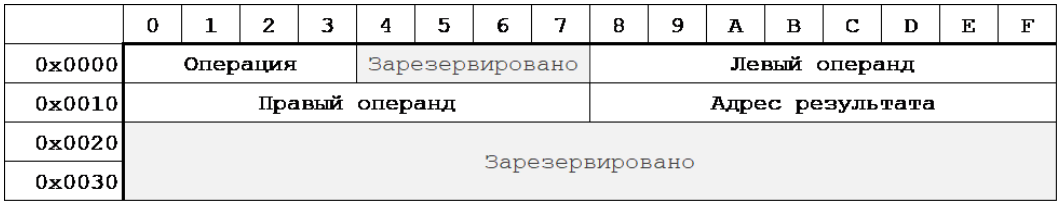


Рис. 5. Схема структуры команды в файле программы

В качестве формата файлов для входных и выходных данных алгоритма был выбран формат CSV с разделителем точка с запятой. Распределение памяти для данных происходит на этапе компиляции. Скалярные данные рассматриваются как квадратные матрицы порядка 1. Объем памяти, необходимый входному/выходному данному, рассчитывается как произведение количества строк на количество столбцов этого данного. В представлении алгоритма в форме Q-детерминанта, получаемом от Q-системы, индексация массивов и матриц имеет начальный индекс 1. Получение значения по индексу рассчитывается по формуле $address + columns \cdot (i-1) + j - 1$, где $address$ — адрес ячейки с индексом (1,1), $columns$ – количество столбцов матрицы, i и j — индексы, по которым происходит обращение. Если использован один индекс, то j будет иметь значение 1. Если оба индекса упущены, то i и j будут иметь значение 1. Вначале распределяется память для входных данных, начиная с ячейки 1. Все матрицы «укладываются» в память построчно по порядку появления в файле метаданных программы. Затем резервируется память для выходных данных точно также, как и для входных. Далее распределяется память под все операции, кроме корневых. Корневыми считаются операции, значение которых будет являться значением Q-термов. Значения Q-термов — выходные данные программы, а память для них уже была выделена.

Опишем работу виртуальной машины. Ядром виртуальной машины являются три ее подсистемы: диспетчер задач, интерпретатор байт-кода программы, подсистема для реализация памяти виртуальной машины. Перед началом работы производятся подготовительные действия: в интерпретаторе байт-кода программы регистрируются все обработчики команд, выделяется память, создается и конфигурируется диспетчер задач. Обработка Q-эффективной программы начинается с создания рабочих потоков. Этим процессом занимается диспетчер задач. В каждом потоке создается объект интерпретатора байт-кода (далее интерпретатор). Количество потоков определяется, как количество вычислителей, доступных на ПВС, минус один или задается пользователем вручную. Один поток (основной) выделяется на работу самого диспетчера. После создания потоков начинается процесс создания для них заданий. Этим процессом также занимается диспетчер. Диспетчер получает команды с помощью передаваемого в него объекта, который представляет из себя реализацию шаблона проектирования итератор (Iterator). За счет этой абстракции команды могут быть получены из любого источника: из файла, из заранее заданного массива (актуально для тестов) или по сети. Максимальный размер задания для потока определяется как константа, заданная в коде виртуальной машины, или заданная пользователем. Как только задание создано, оно ставится в очередь на выполнение. Первый свободный поток берет задание из очереди и начинает его выполнение.

Задание — это очередь команд. Для обработки очередной команды интерпретатор выбирает необходимый обработчик команды и делегирует ему ответственность по вычислению. Если команда готова к выполнению, то она исполняется, а результат записывается в память по адресу, указанному в самой команде. В противном случае команда помещается в конец задания. Процесс обработки задания интерпретатором продолжается пока исходное задание не опустеет.

Репозиторий компилятора доступен по ссылке [6], а виртуальной машины по ссылке [7].

5. Экспериментальное исследование программной системы

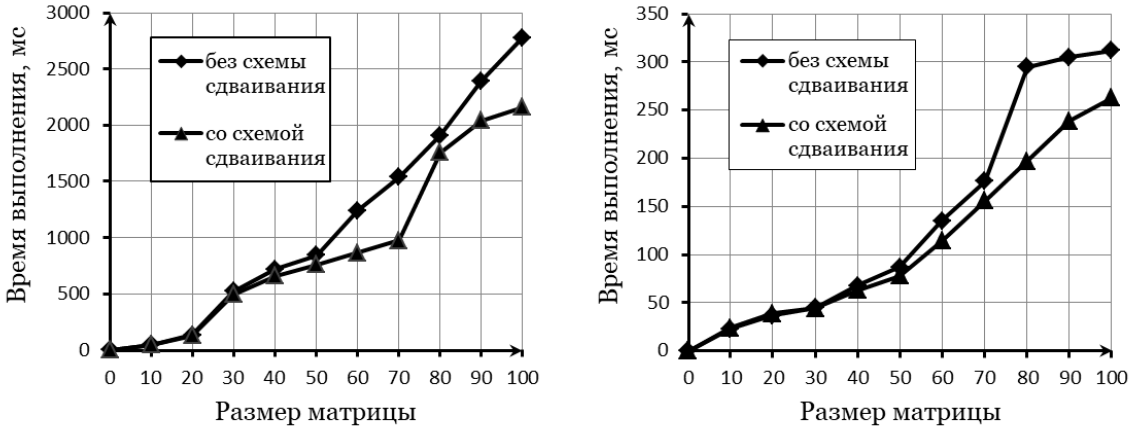
При тестировании программной системы на персональном компьютере было проведено функциональное тестирование с использованием алгоритмов с различными структурами Q-детерминантов, для которых с помощью Q-системы были получены представления в форме Q-детерминантов. Для функционального тестирования применялись следующие алгоритмы: скалярное произведение векторов без использования схемы сдваивания, скалярное произведение векторов с использованием схемы сдваивания, умножение матриц без использования схемы сдваивания, умножение матриц с использованием схемы сдваивания, поиск максимального элемента в массиве чисел, алгоритмы, реализующие методы Гаусса—Жордана, Гаусса—Зейделя и Якоби для решения СЛАУ.

При тестировании входные данные алгоритмов генерировались случайным образом, а выходные сравнивались с эталонными данными, полученными с помощью реализаций в среде MatLAB. Для всех алгоритмов все тесты успешно прошли этапы компиляции и исполнения виртуальной машиной, полученные значения выходных данных совпали с эталонными, при этом для итерационных алгоритмов они совпали с заданной точностью.

Экспериментальное исследование разработанной программной системы проводилось на суперкомпьютере «Торнадо ЮУрГУ». В экспериментах были задействованы два центральных процессора Intel Xeon X5680 с частотой 3.33 GHz одного вычислительного узла, каждый из которых имеет 6 ядер и поддерживает 12 потоков, оперативная память узла 24 Гб ECC DDR3 Full buffered [10].

Для экспериментального исследования использовались два алгоритма умножения двух квадратных матриц: алгоритм $\mathfrak{B}$ без использования схемы сдваивания и алгоритм $\mathfrak{C}$ с использованием схемы сдваивания. Результаты экспериментального исследования приведены на рис. 6. Графики слева показывают время выполнения Q-эффективных программ, выполняющих Q-эффективные реализации алгритмов $\mathfrak{B}$ и $\mathfrak{C}$, а графики справа время выполнения операций этих Q-эффективных реализаций. Таким образом, программная система выполняет Q-эффективную программу алгоритма $\mathfrak{C}$ быстрее, чем алгоритма $\mathfrak{B}$. Также для алгоритма $\mathfrak{C}$ на выполнение операций Q-эффективной реализации требуется меньше времени, чем для алгоритма $\mathfrak{B}$. Полученные результаты объясняются тем, что высота алгоритма $\mathfrak{C}$ меньше, чем алгоритма $\mathfrak{B}$. Исследование времени выполнения операций Q-эффективных реализаций проводилось с целью оценки его вклада в общее время выполнения Q-эффективных программ.

Нам не известны аналоги разработанной в данном исследовании программной системы, поэтому сравнение с аналогами не проводилось. Приведенные результаты экспериментов дают возможность читателю оценить быстродействие Q-эффективных программ. Следует иметь ввиду, что оно зависит от ресурса параллелизма алгоритмов и от условий разработки и выполнения программ, т.е. вычислительной инфраструктуры программ [3]. В статье [15]



а) Q-эффективная реализация

б) Операции Q-эффективной реализации

Рис. 6. Результаты экспериментального исследования

доказано, что при одной и той же вычислительной инфраструктуре Q-эффективная программа имеет динамические характеристики не хуже, чем любая программа, выполняющая другую реализацию того же алгоритма, т.е. Q-эффективная программа эффективна для своей вычислительной инфраструктуры. По нашему мнению, хотя быстродействие важно, ценность программных систем, подобных разработанной в данном исследовании, состоит в том, что они дают возможность не тратить время разработчика на программирование.

Заключение

В статье представлено исследование по созданию на основе концепции Q-детерминанта программной системы автоматизированного проектирования и исполнения программ для эффективной реализации численных алгоритмов. Оно включает решение следующих задач.

1) Проектирование и реализация компилятора, преобразующего представление алгоритма в форме Q-детерминанта в исполняемую программу, использующую ресурс параллелизма алгоритма полностью.
2) Проектирование и реализация виртуальной машины, исполняющей программу, полученную с помощью компилятора.
3) Проведение экспериментального исследования функционирования программной системы, состоящей из компилятора и виртуальной машины, с применением суперкомпьютера «Торнадо ЮУрГУ».

Результаты тестирования разработанной программной системы на персональном компьютере и ее экспериментального исследования на суперкомпьютере «Торнадо ЮУрГУ» подтвердили адекватность и эффективность использованных решений.

Главным результатом исследования является то, что показана возможность создания единой для численных алгоритмов программной системы, которая по блок-схеме алгоритма проектирует программу, использующую ресурс параллелизма алгоритма полностью, и исполняет ее. Разработанная программная система предназначена для использования на ПВС с общей памятью. Целью дальнейших исследований является возможность использования распределенной памяти ПВС. Кроме того, планируется использование вариантов решений на основе концепции Q-детерминанта, отличных от описанного в данном исследовании.

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. 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.
4. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
5. Ершов Ю.Л., Палютин Е.А. Математическая логика. М.: Наука, 1987. 336 с.
6. Репозиторий компилятора программной системы. URL: `https://github.com/yuferovalex/qc` (дата обращения: 29.06.2023).
7. Репозиторий виртуальной машины программной системы. URL: `https://github.com/yuferovalex/qvm` (дата обращения: 29.06.2023).
8. Недожогин Н.С., Копысов С.П., Новиков А.К. Параллельное решение систем линейных уравнений на гибридной архитектуре CPU+GPU // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 2. С. 40–54. DOI: 10.14529/cmse200203.
9. Открытая энциклопедия свойств алгоритмов. URL: `https://algowiki-project.org/ru` (дата обращения: 30.05.2023).
10. Суперкомпьютер «Торнадо ЮУрГУ». URL: `http://supercomputer.susu.ru/computers/tornado/` (дата обращения: 30.05.2023).
11. Afanasyev I.V., Voevodin V.V., Komatsu K., *et al.* Distributed Graph Algorithms for Multiple Vector Engines of NEC SX-Aurora TSUBASA Systems // Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 2. P. 95–113. DOI: 10.14529/jsfi210206.
12. Akhmed-Zaki D., Lebedev D., Malyshkin V., *et al.* Automated Construction of High Performance Distributed Programs in LuNA System // Parallel Computing Technologies (PaCT 2019). Vol. 11657. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.
13. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant // Supercomputing. RuSCDays 2018. Vol. 965. Springer, 2019. P. 565–577. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-05807-4_48.
14. 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.
15. Aleeva V., Aleev R. Investigation and Implementation of Parallelism Resources of Numerical Algorithms // ACM Transactions on Parallel Computing. 2023. Vol. 10. no. 2. Article number 8. P. 1–64. DOI: 10.1145/3583755.

16. 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.

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

18. 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). Vol. 9251. Springer, 2015. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-21909-7_1.

19. 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.

20. Balaprakash P., Dongarra J., Gamblin T., *et al.* Autotuning in High-Performance Computing Applications // Proceedings of the IEEE. 2018. Vol. 106, no. 11. P. 2068–2083. DOI: 10.1109/JPROC.2018.2841200.

21. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions // SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

22. Bosilca G., Bouteiller A., Danalis A., *et al.* Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA // 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. 2011. P. 1432–1441. DOI: 10.1109/IPDPS.2011.299.

23. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability // Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

24. 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). Vol. 910. Springer, 2018. P. 16–30. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_2.

25. Moore G. Cramming More Components onto Integrated Circuits // Electronics Magazine. 1965. Vol. 38, no. 8. P. 114–117.

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

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

DOI: 10.14529/cmse230303

COMPUTER-AIDED DESIGN AND EXECUTION OF EFFECTIVE PROGRAMS FOR NUMERICAL ALGORITHMS

© 2023 V.N. Aleeva

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: aleevavn@susu.ru

Received: 29.06.2023

Designing effective parallel programs for multiprocessor architectures is difficult because there are no clear formal rules to follow. The concept of the Q-determinant can be applied to solve this problem when implementing numerical algorithms. This theory allows for automated analysis of the algorithm parallelism resource, automated comparison of the parallelism resources of algorithms solving the same algorithmic problem. In addition, it makes it possible to design effective programs for the implementation of numerical algorithms using a specially developed design method, improve the efficiency of the implementation of numerical methods and algorithmic problems. The results obtained on the basis of the Q-determinant concept are one of the options for solving the problem of effective implementation of numerical algorithms, methods and algorithmic problems on parallel computing systems. However, the fundamental problem of computer-aided design and execution for any numerical algorithm of a program that implements the algorithm effectively remains unresolved. The paper describes the development of a software system for designing and executing Q-effective programs that is unified for numerical algorithms. A Q-effective program is an effective program designed using the concept of a Q-determinant. The system is intended for use on parallel computing systems with shared memory. It consists of a compiler and a virtual machine. The compiler converts the representation of the algorithm in the form of a Q-determinant into an executable program that uses the algorithm's parallelism resource completely. The virtual machine executes the program generated by the compiler. The paper also provides an experimental study of the created software system using the SUSU Tornado supercomputer.

Keywords: Q-determinant of algorithm, representation of algorithm in form of Q-determinant, Q-effective implementation of algorithm, parallelism resource of algorithm, software Q-system, parallel computing system, parallel program, Q-effective program.

FOR CITATION

Aleeva V.N. Computer-aided Design and Execution of Effective Programs for Numerical Algorithms. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 3. P. 31–49. (in Russian) DOI: 10.14529/cmse230303.

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 the 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. 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.

4. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St.Petersburg, BHV-Petersburg, 2002. 608 p. (in Russian)
5. Ershov Yu.L., Palyutin E.A. Mathematical Logic. Moscow, Mir, 1984. 303 p.
6. Software system compiler repository. URL: `https://github.com/yuferovalex/qc` (accessed: 29.06.2023).
7. Software system virtual machine repository. URL: `https://github.com/yuferovalex/qvm` (accessed: 29.06.2023).
8. Nedozhogin N.S., Kopysov S.P., Novikov A.K. Parallel Solving of Linear Equations Systems on Hybrid Architecture CPU+GPU. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 2. P. 40–54. (in Russian) DOI: 10.14529/cmse200203.
9. Open Encyclopedia of Parallel Algorithmic Features. URL: `https://algowiki-project.org/en` (accessed: 30.05.2023).
10. "Tornado SUSU" Supercomputer. URL: `http://supercomputer.susu.ru/en/computers/tornado/` (accessed: 30.05.2023).
11. Afanasyev I.V., Voevodin V.V., Komatsu K., *et al.* Distributed Graph Algorithms for Multiple Vector Engines of NEC SX-Aurora TSUBASA Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 2. P. 95–113. DOI: 10.14529/jsfi210206.
12. Akhmed-Zaki D., Lebedev D., Malyshkin V., *et al.* Automated Construction of High Performance Distributed Programs in LuNA System. Parallel Computing Technologies (PaCT 2019). Vol. 11657. Springer, 2019. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-25636-4_1.
13. Aleeva V. Designing a Parallel Programs on the Base of the Conception of Q-Determinant. Supercomputing. RuSCDays 2018. Vol. 965. Springer, 2019. P. 565–577. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-05807-4_48.
14. 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.
15. Aleeva V., Aleev R. Investigation and Implementation of Parallelism Resources of Numerical Algorithms. ACM Transactions on Parallel Computing. 2023. Vol. 10. no. 2. Article number 8. P. 1–64. DOI: 10.1145/3583755.
16. 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.
17. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. Vol. 1129. Springer, 2019. P. 641–652. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_52.

18. 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). Vol. 9251. Springer, 2015. P. 3–9. Lecture Notes in Computer Science. DOI: 10.1007/978-3-319-21909-7_1.

19. 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.

20. Balaprakash P., Dongarra J., Gamblin T., *et al.* Autotuning in High-Performance Computing Applications. Proceedings of the IEEE. 2018. Vol. 106, no. 11. P. 2068–2083. DOI: 10.1109/JPROC.2018.2841200.

21. Bauer M., Treichler S., Slaughter E., Aiken A. Legion: Expressing locality and independence with logical regions. SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. 2012. P. 1–11. DOI: 10.1109/SC.2012.71.

22. Bosilca G., Bouteiller A., Danalis A., *et al.* Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA. 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum. 2011. P. 1432–1441. DOI: 10.1109/IPDPS.2011.299.

23. Bosilca G., Bouteiller A., Danalis A., *et al.* PaRSEC: Exploiting Heterogeneity to Enhance Scalability. Computing in Science & Engineering. 2013. Vol. 15, no. 6. P. 36–45. DOI: 10.1109/MCSE.2013.98.

24. 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). Vol. 910. Springer, 2018. P. 16–30. Communications in Computer and Information Science. DOI: 10.1007/978-3-319-99673-8_2.

25. Moore G. Cramming More Components onto Integrated Circuits. Electronics Magazine. 1965. Vol. 38, no. 8. P. 114–117.

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