Применение Q-детерминанта численных алгоритмов для параллельных вычислений*

В.Н. Алеева, Р.Ж. Алеев

Южно-Уральский государственный университет (НИУ)

Концепция Q-детерминанта является одним из подходов к распараллеливанию численных алгоритмов. В основе концепции лежит понятие Q-детерминанта алгоритма, где Q – множество операций, используемых алгоритмом. Q-детерминант состоит из Q-термов. Количество Q-термов, составляющих Q-детерминант алгоритма, равно числу выходных данных алгоритма. Каждый Q-терм описывает все возможные способы вычисления в соответствии с алгоритмом одного из выходных данных в зависимости от входных данных. Любой численный алгоритм имеет Q-детерминант и может быть представлен в форме Q-детерминанта. Такое представление является универсальным описанием численных алгоритмов. Оно делает алгоритм прозрачным с точки зрения структуры и реализации. Q-детерминант содержит только машинно-независимые свойства алгоритма, однако он может быть использован для реализации алгоритма на параллельных вычислительных системах. В работе описаны решения, основанные на применении Q-детерминанта, которые позволяют для любого численного алгоритма определить ресурс параллелизма и разработать Q-эффективную программу, использующую ресурс параллелизма полностью. Показано также практическое применение предлагаемых решений на примере численных алгоритмов с различными структурами Q-детерминантов: алгоритмов умножения плотных и разреженных матриц, методов решения систем линейных уравнений Гаусса–Жордана, Якоби, Гаусса–Зейделя и других. Работа продолжает исследования, начатые в предыдущих работах авторов. Полученные результаты могут быть использованы для повышения эффективности реализации численных алгоритмов на параллельных вычислительных системах.

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

1. Введение

Высокопроизводительные вычисления с использованием параллельных вычислительных систем (ПВС) все более необходимы в различных областях. При этом наблюдается значительный разрыв между вычислительным потенциалом ПВС и его использованием. Одной из причин данной проблемы является то, что реализация алгоритмов на ПВС не является эффективной, так как не используется весь ресурс параллелизма алгоритмов. В результате алгоритмы не применяют вычислительные ресурсы настолько, насколько они способны это делать. Таким образом, проблемы исследования ресурса параллелизма алгоритмов и его использования при реализации алгоритмов на ПВС являются актуальными, а их решение имеет научное и практическое значение.

Решению проблемы эффективной реализации алгоритмов на ПВС посвящено много научных исследований. Приведем краткий обзор некоторых из них. Очень важным и развитым направлением по исследованию параллельной структуры алгоритмов и программ с целью их реализации на ПВС является направление, основы которого изложены в [1,2]. Результаты исследований данного направления используются при создании Интернет-энциклопедии AlgoWiki [3]. В энциклопедии описываются свойства, особенности, статические и динамические характеристики конкретных алгоритмов. Это помогает реализовывать описанные алгоритмы эффективно. Однако в рамках данного направления программное исследование ресурса парал-

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

лелизма алгоритмов не рассматривается. Также не предлагается технология создания параллельных программ, использующих весь ресурс параллелизма алгоритмов. За время развития параллельных вычислений предложены различные подходы к разработке параллельных программ, созданы десятки языков параллельного программирования и множество различных инструментальных средств. Среди таких разработок Т-система [4,5]. Она представляет среду программирования с поддержкой автоматического динамического распараллеливания программ. Однако нельзя утверждать, что создаваемые с помощью Т-системы параллельные программы используют ресурс параллелизма алгоритмов полностью. Еще одним подходом к созданию параллельных программ является синтез параллельных программ. Метод синтеза параллельных программ заключается в том, чтобы из базы знаний параллельных алгоритмов конструировать новые параллельные алгоритмы для решения более крупных задач. На основе метода синтеза параллельных программ разработана технология фрагментированного программирования и реализующие ее язык и система программирования LuNA. В настоящее время это направление исследований развивается [6,7]. Подход является универсальным, но проблему исследования и использования ресурса параллелизма алгоритмов он не решает. Для преодоления ресурсных ограничений в статье [8] предлагаются методы построения параллельных архитектурно-независимых программ с использованием функционального языка программирования. Однако не показано, используют ли создаваемые программы весь ресурс параллелизма алгоритмов. Существует много исследований, в которых при разработке параллельных программ учитывается специфика алгоритмов и архитектуры ПВС. В качестве примеров таких исследований можно привести [9–16]. Эти исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС конкретной архитектуры. Однако они не предлагают универсального подхода. Хотелось бы отметить, что в таких исследованиях нет информации о степени использования ресурса параллелизма алгоритма. В результате может появиться новое исследование, которое использует ресурс параллелизма лучше.

Цель данной работы – описание решений для исследования ресурса параллелизма любого численного алгоритма и для разработки параллельных программ, использующих ресурс параллелизма численных алгоритмов полностью, а также их практическая реализация. Для достижения цели используется концепция Q-детерминанта [17], при этом возникают следующие задачи.

1. Разработка методов исследования ресурса параллелизма любого численного алгоритма.
2. Разработка метода проектирования параллельных программ, использующих ресурс параллелизма численных алгоритмов полностью.
3. Практическое применение методов исследования ресурса параллелизма численных алгоритмов и проведение вычислительных экспериментов.
4. Практическое применение метода проектирования параллельных программ и проведение вычислительных экспериментов.

Данная работа продолжает исследования, начатые в предыдущих работах авторов.

Статья состоит из трех разделов и заключения. В первом разделе приведены методы исследования ресурса параллелизма численных алгоритмов и проектирования параллельных программ. Второй раздел содержит описание практического применения предлагаемых методов. В третьем разделе приводятся результаты экспериментов, подтверждающих адекватность и эффективность предлагаемых методов. Заключение содержит краткое описание полученных результатов, выводы, которые можно сделать на их основе, описание перспектив использования и направления дальнейшего исследования.

2. Методы исследования ресурса параллелизма численных алгоритмов и проектирования параллельных программ

Теоретической основой проводимых исследований является концепция Q-детерминанта. Дадим основные определения концепции для лучшего понимания данной работы.

Пусть α – численный алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$, где $N = \{n_1, \dots, n_k\}$ – множество параметров размерности проблемы или пустое множество, B – множество входных данных, $\bar{y} = (y_1, \dots, y_m)$ – множество выходных данных. Пусть $\bar{N}$ – вектор $(\bar{n}_1, \dots, \bar{n}_k)$, где $\bar{n}_i$ – некоторое заданное значение параметра n_i $(i = 1, \dots, k)$, $\{\bar{N}\}$ – множество всевозможных векторов $\bar{N}$, Q – множество операций, используемых алгоритмом α.

Далее под выражением над B и Q следует понимать терм сигнатуры поля действительных чисел, пополненной операциями из множества Q, в стандартном смысле математической логики [18]. Безусловным Q-термом называется любое однозначное отображение $w: \{\bar{N}\} \to V$, где V – множество всех выражений над B и Q. Безусловный Q-терм w называется безусловным логическим Q-термом, если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B выражение $w(\bar{N})$ принимает значение логического типа. Пусть $u_1, \dots, u_l$ – безусловные логические Q-термы, $w_1, \dots, w_l$ – безусловные Q-термы. Множество пар (u_i, w_i) $(i = 1, \dots, l)$ обозначается $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\dots,l}$ и называется условным Q-термом длины l. Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\dots}$ называется условным бесконечным Q-термом, если $\{(u_i, w_i)\}_{i=1,\dots,l}$ является условным Q-термом для любого $l < \infty$. Если не имеет значения, является ли Q-терм безусловным, условным или условным бесконечным, то его можно называть Q-термом.

Под вычислением безусловного Q-терма w при интерпретации B следует понимать вычисление выражения $w(\bar{N})$ при некотором $\bar{N} \in \{\bar{N}\}$. Для вычисления при заданной интерпретации B и некотором $\bar{N} \in \{\bar{N}\}$ условного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\dots,l}$ необходимо найти такую пару $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$, что $u_{i_0}(\bar{N})$ принимает значение $true$, а значение $w_{i_0}(\bar{N})$ определено. В этом случае значение $(\bar{u}, \bar{w})$ равно $w_{i_0}(\bar{N})$. Если установлено, что пары выражений $u_{i_0}(\bar{N}), w_{i_0}(\bar{N})$ не существует, то значение $(\bar{u}. \bar{w})$ для данной интерпретации B и данного $\bar{N}$ не определено. Аналогично определяется вычисление условного бесконечного Q-терма.

Предположим, что I_1, I_2, I_3 – подмножества множества $I = (1, \dots, m)$ такие, что: одно или два из подмножеств I_i $(i = 1,2,3)$ могут быть пустыми, $I_1 \cup I_2 \cup I_3 = I$, $I_i \cap I_j = \emptyset$ $(i \neq j; i, j = 1,2,3)$. Предположим также, что имеется следующее множество Q-термов $\{f_i\}_{i \in I}$: f_{i_1} – безусловный Q-терм для $i_1 \in I_1$, $f_{i_1} = w^{i_1}$; f_{i_2} – условный Q-терм для $i_2 \in I_2$, $f_{i_2} = \{\left(u_j^{i_2}, w_j^{i_2}\right)\}_{j=1,\dots,l_{i_2}}$, l_{i_2} является вычислимой функцией параметров N; f_{i_3} – условный бесконечный Q-терм для $i_3 \in I_3$, $f_{i_3} = \{\left(u_j^{i_3}, w_j^{i_3}\right)\}_{j=1,2,\dots}$. Предположим, что алгоритм α заключается в том, что для $i \in I$ находит y_i путем вычисления Q-терма f_i. В этом случае множество Q-термов f_i $(i \in I)$ называется Q-детерминантом алгоритма α, а представление алгоритма α в виде $y_i = f_i$ $(i \in I)$ представлением в форме Q-детерминанта.

Приведем примеры представления алгоритмов в форме Q-детерминанта. Рассмотрим алгоритм умножения матриц $A = [a_{ij}]_{i=1,\dots,n; j=1,\dots,k}$ и $B = [b_{ij}]_{i=1,\dots,k; j=1,\dots,m}$. Результатом является матрица $C = [c_{ij}]_{i=1,\dots,n; j=1,\dots,m}$, где $c_{ij} = \sum_{s=1}^{k} a_{is} b_{sj}$. Мы видим, что алгоритм умножения матриц сразу представлен в форме Q-детерминанта, который состоит из nm безусловных Q-термов.

Однако, как правило, алгоритмы не представлены в форме Q-детерминанта, то есть их Q-детерминант скрыт. Рассмотрим метод Гаусса–Жордана решения системы линейных уравнений $A\bar{x} = \bar{b}$. Предположим, что $A = [a_{ij}]_{i,j=1,\dots,n}$ – матрица с ненулевым определителем. Пусть $\bar{A}$ – расширенная матрица системы. Метод Гаусса–Жордана состоит из n шагов. На шаге k $(k = 1, \dots, n)$ в строке k матрицы A выбираем первый ненулевой элемент в качестве ведущего. Обозначим номер столбца ведущего элемента через j_k. Обозначим расширенную матрицу системы линейных уравнений, полученной после шага k $(k = 1, \dots, n)$, $\bar{A}^{j_1 \dots j_k} = [a_{ij}^{j_1 \dots j_k}]_{i=1,\dots,n; j=1,\dots,n+1}$. После шага n получим систему $\bar{A}^{j_1 \dots j_n} \bar{x} = \bar{b}^{j_1 \dots j_n}$, где $\bar{A}^{j_1 \dots j_n} = [a_{ij}^{j_1 \dots j_n}]_{i=1,\dots,n; j=1,\dots,n+1}$, $\bar{b}^{j_1 \dots j_n} = (a_{1,n+1}^{j_1 \dots j_n}, \dots, a_{n,n+1}^{j_1 \dots j_n})^T$. Введем обозначения: $L_{j_1} = \bigwedge_{j=1}^{j_1 - 1} (a_{1j} = 0)$, если $j_1 \neq 1$, $L_{j_1} = true$, если $j_1 = 1$, $L_{j_l} = \bigwedge_{j=1}^{j_l - 1} (a_{lj}^{j_1 \dots j_{l-1}} = 0)$, если $j_l \neq 1$, $L_{j_l} = true$, если $j_l = 1$ $(l = 2, \dots, n)$. Перестановки элементов $(1, \dots, n)$ можно занумеровать. Пусть i номер перестановки $(j_1, \dots, j_n)$. Тогда $w_i^{j_l} = a_{i,n+1}^{j_1 \dots j_n}$ $(l = 1, \dots, n)$ и $u_i = L_{j_1} \wedge (a_{1 j_1} \neq 0) \wedge (\bigwedge_{l=2}^{n} (L_{j_l} \wedge (a_{l j_l}^{j_1 \dots j_{l-1}} \neq 0)))$ являются безусловными Q-термами. Q-детерминант метода Гаусса–Жордана состоит из n условных Q-термов длины $n!$, а представление в форме Q-детерминанта имеет вид $x_j = \left\{\left(u_1, w_1^j\right), \dots, \left(u_{n!}, w_{n!}^j\right)\right\}$ $(j = 1, \dots, n)$.

Рассмотрим метод Гаусса–Зейделя решения системы линейных уравнений в качестве примера алгоритма, Q-детерминант которого содержит условные бесконечные Q-термы. Пусть $A = [a_{ij}]_{i,j=1,\dots,n}$, $a_{ii} \neq 0$ $(i = 1, \dots, n)$, $\bar{x} = (x_1, \dots, x_n)^T$, $\bar{b} = (a_{1,n+1}, \dots, a_{n,n+1})^T$. Введем обозначения $c_{ij} = -a_{ij}/a_{ii}$ и $d_i = b_i/a_{ii}$ $(i,j = 1, \dots, n)$. Пусть $\bar{x}^0 = (x_1^0, \dots, x_n^0)$ – начальное приближение. Тогда итерационный процесс можно записать в виде $x_i^{k+1} = \sum_{j=1}^{i-1} c_{ij} x_j^{k+1} + \sum_{j=i+1}^{n} c_{ij} x_j^k + d_i$ $(i = 1, \dots, n; k = 1,2, \dots)$. Критерием окончания итерационного процесса является условие $\|x^{k+1} - x^k\| < \varepsilon$, где ε – точность вычисления. Q-детерминант метода Гаусса–Зейделя состоит из n условных бесконечных Q-термов, а представление в форме Q-детерминанта имеет вид $x_i = \{(\|\bar{x}^1 - \bar{x}^0\| < \varepsilon, x_i^1), \dots, (\|\bar{x}^k - \bar{x}^{k-1}\| < \varepsilon, x_i^k), \dots\}$ $(i = 1, \dots, n)$.

Если алгоритм α представлен в форме Q-детерминанта $y_i = f_i$ $(i \in I)$, то процесс вычисления Q-термов f_i при заданной интерпретации B и некотором $\bar{N} \in \{\bar{N}\}$ называется реализацией алгоритма α. Если реализация алгоритма α характеризуется тем, что выражения $W(\bar{N}) = \left\{w^{i_1}(\bar{N})(i_1 \in I_1);\ u_j^{i_2}(\bar{N}), w_j^{i_2}(\bar{N})\left(i_2 \in I_2, j = 1, \dots, l_{i_2}\right);\ u_j^{i_3}(\bar{N}), w_j^{i_3}(\bar{N})\left(i_3 \in I_3 j = 1,2, \dots\right)\right\}$ вычисляются одновременно и при этом операции выполняются по мере вычисления их операндов, то реализация называется Q-эффективной. Q-эффективная реализация использует ресурс параллелизма алгоритма полностью, поэтому является максимально параллельной реализаций алгоритма. Реализация алгоритма α называется выполнимой, если она такова, что одновременно необходимо выполнять конечное число операций. Существуют алгоритмы, для которых Q-эффективная реализация не является выполнимой. Их процент незначителен.

Выражение и входящие в него операции имеют уровни вложенности. Для обозначения числа уровней вложенности выражения $w(\bar{N})$ будем использовать $T^{w(\bar{N})}$. Цепочкой выражений длины n будем называть выражение, полученное из n выражений путем применения $(n-1)$ раз одной из ассоциативных операций множества Q без указания порядка их выполнения с помощью скобок. При определении уровня вложенности выражения и его подвыражений порядок выполнения операций цепочки задается по схеме сдваивания. Например, для вычисления цепочки $a_1 + a_2 + a_3 + a_4$ по схеме сдваивания сначала нужно вычислить $b_1 = a_1 + a_2$ и $b_2 = a_3 + a_4$, а затем $c = b_1 + b_2$.

Для выполнимой Q-эффективной реализации алгоритма α введем характеристики параллельной сложности: $D_\alpha(\bar{N})$ – максимальное число уровней вложенности выражений $W(\bar{N})$, $D_\alpha(\bar{N}) = \max\limits_{w(\bar{N}) \in W(\bar{N})} T^{w(\bar{N})}$; $P_\alpha(\bar{N})$ – максимальное количество операций всех уровней вложенности всех выражений $W(\bar{N})$, $P_\alpha(\bar{N}) = \max\limits_{1 \le r \le D_\alpha(\bar{N})} \sum_{w(\bar{N}) \in W(\bar{N})} O_r^{w(\bar{N})}$, где $O_r^{w(\bar{N})}$– количество операций уровня вложенности r выражения $w(\bar{N})$. $D_\alpha(\bar{N})$ характеризует время выполнения Q-эффективной реализации алгоритма, а $P_\alpha(\bar{N})$ количество процессоров, необходимое для выполнения Q-эффективной реализации алгоритма. $D_\alpha(\bar{N})$ будем называть высотой алгоритма, а $P_\alpha(\bar{N})$ его шириной.

Рассмотрим следующие методы исследования ресурса параллелизма численных алгоритмов с использованием концепции Q-детерминанта.

1. Метод построения Q-детерминанта алгоритма на основе его блок-схемы.
2. Метод получения Q-эффективной реализации алгоритма по его Q-детерминанту.
3. Метод вычисления характеристик параллельной сложности выполнимой Q-эффективной реализации алгоритма.
4. Метод сравнения характеристик параллельной сложности Q-эффективных реализаций двух алгоритмов, решающих одну и ту же алгоритмическую проблему.

Как правило, алгоритм не представлен в форме Q-детерминанта, поэтому для построения Q-детерминанта необходим метод, использующий общепринятое представление алгоритма, например, с помощью блок-схемы. Анализ блок-схемы алгоритма будем проводить при фиксированных параметрах размерности алгоритмической проблемы. Блок-схема обрабатывается сверху вниз. Для разработки метода построения Q-детерминанта алгоритма на основе его блок-схемы были исследованы особенности различных алгоритмов. Особенность алгоритмов, Q-детерминант которых состоит из безусловных Q-термов, заключается в том, что проход по блок-схеме алгоритма должен осуществляться последовательно, как если бы алгоритм выполнялся. После прохода по блок-схеме в качестве значений Q-термов $f_{i_1}(i_1 \in I_1)$ будут получены выражения $w^{i_1}(\overline{N})$ $(i_1 \in I_1)$. Эти выражения формируются с помощью содержимого блоков, участвующих в вычислении $y_{i_1}(i_1 \in I_1)$. В случае алгоритмов, Q-детерминант которых содержит условные Q-термы, при проходе через блок, в котором записано условие, содержащее входные переменные, происходит разветвление, а затем каждая ветвь обрабатывается отдельно. В результате обработки одной ветви формируется список пар $(u_j^{i_2}(\overline{N}), w_j^{i_2}(\overline{N}))(i_2 \in I_2)$ для некоторого $j = 1, \dots, l_{i_2}$. По завершению обработки одной ветви обработчик блок-схемы возвращается на ближайший блок, где произошло разветвление, и продолжает обработку уже с противоположным условием, записанным в данном блоке. После обработки всех ветвей при проходе по блок-схеме будут сгенерированы Q-термы $(u_j^{i_2}(\overline{N}), w_j^{i_2}(\overline{N}))(i_2 \in I_2, j = 1, \dots, l_{i_2})$. Q-детерминанты итерационных алгоритмов содержат условные бесконечные Q-термы. Путем ограничения количества итераций случай, когда Q-детерминант содержит условные бесконечные Q-термы, сводится к случаю, когда Q-детерминант содержит условные Q-термы конечной длины. Результатом применения к численному алгоритму описанного метода является множество Q-термов f_i $(i \in I)$ для некоторого значения $\overline{N} \in \{\overline{N}\}$. Следовательно, формируется множество выражений $W(\overline{N})$.

Уровни вложенности операций, входящих в выражения $W(\overline{N})$, можно вычислить с помощью определения уровня вложенности операции. Метод получения Q-эффективной реализации алгоритма по его Q-детерминанту предназначен для выполнения этих вычислений. Множество выражений $W(\overline{N})$ с вычисленными уровнями вложенности операций называется планом выполнения Q-эффективной реализации алгоритма.

Метод вычисления характеристик параллельной сложности Q-эффективной реализации алгоритма использует план выполнения Q-эффективной реализации. Для вычисления $D_\alpha(\overline{N})$ метод определяет максимальное значение уровня вложенности операций, входящих в выражения $W(\overline{N})$. А для вычисления $P_\alpha(\overline{N})$ метод находит количество операций $\sum_{w(\overline{N}) \in W(\overline{N})} O_r^{w(\overline{N})}$ каждого из уровней вложенности r $(1 \le r \le D_\alpha(\overline{N}))$, а затем определяет максимальное из найденных значений. Отметим, что полученная с помощью данного метода оценка ширины алгоритма $P_\alpha(\overline{N})$ зачастую может превышать количество процессоров, используемых одновременно при реализации алгоритма на ПВС. Это связано с тем, что при реализации алгоритма необходимо стремиться вычислять без дублирования одинаковые выражения и подвыражения, из которых состоят Q-термы. Однако при этом также необходимо помнить, что исключение дублирования не всегда целесообразно, так как может привести к снижению быстродействия из-за дополнительных пересылок между процессорными узлами при использовании распределенной памяти.

Результаты вычислений, полученные с помощью описанных методов, мы предлагаем сохранять в базе данных. База данных должна содержать уникальный идентификатор, название, текстовое описание исследуемого алгоритма, его Q-детерминанты для различных значений параметров размерности $\overline{N} \in \{\overline{N}\}$ вместе с характеристиками параллельной сложности Q-эффективной реализации алгоритма $D_\alpha(\overline{N})$ и $P_\alpha(\overline{N})$.

Метод сравнения характеристик параллельной сложности Q-эффективных реализаций двух алгоритмов, решающих одну и ту же алгоритмическую проблему, использует содержащуюся в базе данных информацию. Он сравнивает характеристики параллельной сложности Q-эффективных реализаций для одинаковых наборов значений параметров размерности N, что дает возможность определить алгоритм с лучшей характеристикой $D_\alpha(\overline{N})$ или $P_\alpha(\overline{N})$.

Модель концепции Q-детерминанта позволяет исследовать машинно-независимые свойства численных алгоритмов, но не учитывает особенности их выполнения на ПВС. Например, модель не учитывает зависимость реализаций алгоритмов от потерь, возникающих при обращении к памяти. По этой причине расширим модель концепции Q-детерминанта, добавив в нее две подмодели, которые отражают в абстрактной форме архитектуру целевых ПВС с общей и с распределенной памятью. Исходную модель концепции Q-детерминанта будем называть базовой.

В качестве модели параллельных вычислений, ориентированной на ПВС с общей памятью, будем использовать PRAM [19]. Для организации параллельных вычислений на многопроцессорных системах с общей памятью в настоящее время наиболее широко применяется технология OpenMP. Эту технологию будем применять при разработке параллельных программ на основе расширенной модели концепции Q-детерминанта для ПВС с общей памятью. В качестве модели параллельных вычислений, ориентированной на ПВС с распределенной памятью, будем использовать модель BSP [20]. Одним из часто используемых в параллельном и распределенном программировании фреймворков является парадигма «мастер-рабочие» [21]. В рамках расширенной модели концепции Q-детерминанта для подмодели, ориентированной на распределенную память, в данном исследовании ограничимся парадигмой «мастер-рабочие» с конфигурацией, включающей одного мастера и множество рабочих, так как такая конфигурация является наиболее популярной. Технология MPI [22] де-факто является стандартом для параллельного программирования на распределенной памяти. Эту технологию будем применять при разработке параллельных программ на основе расширенной модели концепции Q-детерминанта для ПВС с распределенной памятью.

Опишем метод проектирования параллельных программ, использующих ресурс параллелизма численных алгоритмов полностью. Он основан на расширенной модели концепции Q-детерминанта и состоит из трех этапов.

1. Построение Q-детерминанта алгоритма.
2. Описание плана выполнения Q-эффективной реализации алгоритма.
3. Разработка программы для выполнения Q-эффективной реализации алгоритма, если она выполнима.

При построении Q-детерминанта алгоритма значения параметров размерности N не фиксируются. Первые два этапа метода используют базовую модель концепции Q-детерминанта. Третий этап осуществляется в рамках подмоделей расширенной модели, ориентированных на вычислительные системы с общей и распределенной памятью. При использовании распределенной памяти план выполнения Q-эффективной реализации дополняется планом для распределения вычислений по узлам. План выполнения Q-эффективной реализации алгоритма и план распределения вычислений по узлам дают исследователю возможность при необходимости оценить теоретически параметры, которые могут быть оценены в рамках моделей параллельных вычислений PRAM и BSP.

Программа называется Q-эффективной, если она создается с использованием описанного метода проектирования параллельных программ. Процесс ее создания называется Q-эффективным программированием. Q-эффективная программа полностью использует ресурс параллелизма алгоритма, так как выполняет его Q-эффективную реализацию.

3. Практическое применение предложенных методов

В настоящее время разработана первая версия программной системы, реализующей методы исследования ресурса параллелизма численных алгоритмов. Самый важный и самый сложный модуль программной системы реализует метод построения Q-детерминанта алгоритма на основе его блок-схемы. Программная реализация этого модуля впервые была описана в [23]. В качестве формата представления блок-схемы алгоритма был использован формат JSON, так как он позволяет работать с описанием блок-схем алгоритмов в текстовом формате. Q-детерминанты могут формироваться как в формате JSON, так и в строковом формате. Примеры описания блок-схем и Q-детерминантов некоторых алгоритмов приведены в [23]. Программа разработана с помощью языка программирования C# на базе платформы .NET. Разработка выполнялась в соответствии с парадигмой объектно-ориентированного программирования. Программа тестировалась с помощью алгоритмов, Q-детерминанты которых содержат различные типы Q-термов. К ним относятся алгоритм умножения матриц, алгоритм нахождения максимального числа в последовательности чисел, алгоритм Эвклида, метод Якоби решения системы линейных уравнений.

Вторая версия программной реализации метода построения Q-детерминанта алгоритма на основе его блок-схемы разрабатывается в настоящее время. Она формирует файл, содержащий более детальное описание структуры Q-детерминанта, чем первая, что предоставляет более широкие возможности для исследования машинно-независимых свойств алгоритмов. Опишем содержание формируемого файла. Каждому условному Q-терму соответствует столько строк файла, какова длина l Q-терма. Каждая из строк содержит идентификатор выходной переменной, вычисляемой с помощью данного Q-терма, знак равенства и одну пару (u_i, w_i) $(i = 1, \dots, l)$. Q-термы u_i и w_i описаны в формате JSON и разделены с помощью точки с запятой. Безусловному Q-терму соответствует одна строка файла. В этом случае логического Q-терма нет, поэтому вместо него используется пробел. С помощью второй версии программы фактически формируется представление алгоритма в форме Q-детерминанта для фиксированных параметров размерности N алгоритмической проблемы. В настоящее время разработано программное обеспечение для конвертации формата выходного файла, полученного с помощью второй версии программы, в формат выходного файла первой версии. Это позволяет использовать уже разработанную базу данных и ее приложения. Параллельно с этим проектируется база данных, ориентированная на описание структуры Q-детерминанта, поученное в результате использования второй версии программы.

База данных программной системы была создана с помощью системы управления базами данных Microsoft SQL Server. Для взаимодействия с базой данных разработано серверное приложение. Задачами серверного приложения являются чтение, добавление, редактирование и удаление информации об алгоритмах, а также чтение, добавление и удаление Q-детерминантов алгоритмов. Также разработаны и реализованы алгоритмы для получения плана выполнения Q-эффективной реализации алгоритма, для вычисления характеристик параллельной сложности Q-эффективной реализации алгоритма, для сравнения характеристик параллельной сложности Q-эффективных реализаций двух алгоритмов. Для пользователей программной системы было создано клиентское приложение. Более подробно реализация базы данных и ее приложений описаны в [24]. В последнее время база данных переведена на платформу СУБД PostgreSQL.

Программная система, реализующая методы исследования ресурса параллелизма численных алгоритмов, получила название «Q-система (Q-system)». Она предназначена для исследования ресурса внутреннего параллелизма любого численного алгоритма. Кроме того она дает возможность выбрать алгоритм с лучшим ресурсом внутреннего параллелизма из нескольких алгоритмов, решающих одну и ту же алгоритмическую проблему. К выбранному с помощью Q-системы алгоритму может быть применен метод проектирования Q-эффективных программ. Учитывая, что Q-детерминант содержит все машинно-независимые свойства алгоритма, с помощью Q-системы впоследствии можно автоматизировать исследование различных свойств алгоритмов путем разработки новых функций системы. В настоящее время Q-система находится в опытной эксплуатации.

Необходимо отметить, что рассчитывать на то, что в базу данных могут быть записаны Q-детерминанты для любых сколь угодно больших значений параметров размерности, нельзя, так как средства разработки накладывают ограничения на размер Q-детерминантов при их формировании и использовании. В связи с этим важно решить задачу не только интерполяции функций D_α и P_α, которые хранятся в базе данных в табличном виде, но и их экстраполяции. Для решения этих задач разрабатывается программное обеспечение.

Для пользователей Q-системы было бы удобно иметь графическое представление характеристик параллельной сложности D_α и P_α. Эту функциональность планируется реализовать для случая, когда число параметров (параметры размерности и параметр для количества итераций), от которых зависят характеристики D_α и P_α, не превышает двух.

Возможность практического применения метода проектирования Q-эффективных программ и экспериментальное исследование разработанных Q-эффективных программ показаны в нескольких работах студентов Южно-Уральского госуниверситета на примере алгоритмов, имеющих различные структуры Q-детерминантов. Исследования проводились на суперкомпьютере «Торнадо ЮУрГУ». Q-эффективные программы для общей памяти выполнялись на одном процессорном узле, а для распределенной памяти на нескольких процессорных узлах.

Алгоритмы умножения плотных и разреженных матриц исследованы в [25]. Метод Гаусса–Жордана решения системы линейных уравнений рассматривается в [26]. В [27] приводятся результаты исследования метода Якоби решения системы линейных уравнений. Исследование метода Гаусса–Зейделя решения системы линейных уравнений проведено в [28]. Для этих алгоритмов разработаны Q-эффективные программы для общей и распределенной памяти и получены оценки их динамических характеристик.

Метод проектирования Q-эффективных программ был апробирован на алгоритмах с малым и большим ресурсом параллелизма. Например, в [29] рассматривается метод прогонки решения системы линейных трехточечных уравнений. План выполнения Q-эффективной реализации показывает, что данный метод обладает малым ресурсом параллелизма, поэтому при выполнении его Q-эффективной реализации, например, на кластерных системах, использование распределенной памяти не целесообразно, так как может привести к увеличению времени выполнения по сравнению с использованием общей памяти. В [29] также исследуется метод Фурье решения системы разностных уравнений. Он обладает большим ресурсом параллелизма и может быть реализован эффективно на распределенной памяти с использованием принципа «мастер-рабочие».

Следует отметить, что исследования по практическому применению метода проектирования Q-эффективных программ показывают, что существуют алгоритмы, для которых применять принцип «мастер-рабочие» не целесообразно, поскольку увеличивается количество передач между вычислительными узлами. Примером такого алгоритма может быть метод Якоби для решения пятиточечных разностных уравнений.

Интерес представляет также апробация метода проектирования Q-эффективных программ с помощью алгоритмов, у которых Q-детерминант имеет сложную структуру. К таким алгоритмам относятся, например, метод Гаусса–Жордана [26] и метод Гаусса–Зейделя решения системы линейных уравнений [28]. Заслуживает внимания факт, что эти алгоритмы, как и другие известные широко используемые алгоритмы, к которым было применено Q-эффективное программирование, имеют хотя и сложные, но хорошо структурированные Q-детерминанты и Q-эффективные реализации. Этот факт облегчает создание для них Q-эффективных программ.

4. Результаты вычислительных экспериментов

Приведем некоторые результаты проведенных вычислительных экспериментов. Сначала покажем примеры описаний структур Q-детерминантов, сформированных с помощью второй версии программной реализации метода построения Q-детерминанта алгоритма на основе его блок-схемы. Для наглядности значения параметров размерности N в примерах использованы минимальные. Описание структуры Q-детерминанта алгоритма умножения плотных матриц $C = A \times B$, где A, B и C имеют размер 2×2, показано на рис. 1; алгоритма нахождения максимального числа max в последовательности чисел $A(i)$ $(i = 1,2,3)$ на рис. 2; метода Гаусса–Жордана решения системы линейных уравнений $AX = B$, где A имеет размер 2×2, $B = (A(1,3), A(2,3))^T$, на рис. 3; двух итераций метода Якоби решения системы линейных уравнений $AX = B$, где A имеет размер 2×2, $X0$ – начальное приближение, e – точность вычислений, на рис. 4. Для описания безусловных Q-термов в формате JSON используются обозначения: op – операция (operation), fO – первый операнд (firstOperand), sO – второй операнд (secondOperand). Длина используемых обозначений была уменьшена по сравнению с предыдущей реализацией программы с целью сокращения размеров формируемых файлов.

С помощью второй версии программной реализации метода построения Q-детерминанта алгоритма на основе его блок-схемы были получены и записаны в базу данных Q-системы Q-детерминанты алгоритмов вычисления квадратного уравнения, вычисления скалярного произведения векторов, умножения матриц, нахождения максимального элемента в последовательности чисел, методов решения систем линейных уравнений Гаусса–Жордана, Якоби и Гаусса–Зейделя. Для этих Q-детерминантов программно были рассчитаны и записаны в базу данных характеристики параллельной сложности $D_\alpha(\bar{N})$ и $P_\alpha(\bar{N})$. В режиме просмотра информации Q-система доступна по адресу https://qclient.herokuapp.com.

Далее проиллюстрируем динамические характеристики некоторых разработанных Q-эффективных программ. Ускорение Q-эффективной программы определялось по формуле $S = T_1/T_p$, где T_1 – время выполнения последовательной программы на одном вычислительном ядре, T_p – время выполнения Q-эффективной программы на p вычислительных ядрах. Для вычисления эффективности Q-эффективной программы применялась формула $E = S/p$, где p – количество используемых вычислительных ядер.

На рис. 5 и рис. 6 показаны графики, демонстрирующие ускорение и эффективность Q-эффективных программ для алгоритмов умножения плотных и разреженных матриц для общей и распределенной памяти соответственно. На графиках представлены результаты экспериментов для матриц размера 30000×30000 [25]. Аналогично для метода Гаусса–Зейделя на рис. 7 и рис. 8 приведены графики, демонстрирующие ускорение и эффективность Q-эффективных программ для общей и распределенной памяти соответственно. Эксперименты проводились для систем линейных уравнений $AX = B$, где A имеет размер 45000×45000 [28].

```
C(1,1)= ;{"op": "+", "fO": {"op": "*", "fO": "A(1,1)", "sO": "B(1,1)"}, "sO": {"op": "*", "fO": "A(1,2)", "sO": "B(2,1)"}}
C(1,2)= ;{"op": "+", "fO": {"op": "*", "fO": "A(1,1)", "sO": "B(1,2)"}, "sO": {"op": "*", "fO": "A(1,2)", "sO": "B(2,2)"}}
C(2,1)= ;{"op": "+", "fO": {"op": "*", "fO": "A(2,1)", "sO": "B(1,1)"}, "sO": {"op": "*", "fO": "A(2,2)", "sO": "B(2,1)"}}
C(2,2)= ;{"op": "+", "fO": {"op": "*", "fO": "A(2,1)", "sO": "B(1,2)"}, "sO": {"op": "*", "fO": "A(2,2)", "sO": "B(2,2)"}}
```

Рис. 1. Описание структуры Q-детерминанта алгоритма умножения матриц

```
max={"op": "&", "fO": {"op": "<", "fO": "A(1)", "sO": "A(2)"}, "sO": {"op": "<", "fO": "A(2)", "sO": "A(3)"}};A(3)
max={"op": "&", "fO": {"op": "<", "fO": "A(1)", "sO": "A(2)"}, "sO": {"op": ">=", "fO": "A(2)", "sO": "A(3)"}};A(2)
max={"op": "&", "fO": {"op": ">=", "fO": "A(1)", "sO": "A(2)"}, "sO": {"op": "<", "fO": "A(1)", "sO": "A(3)"}};A(3)
max={"op": "&", "fO": {"op": ">=", "fO": "A(1)", "sO": "A(2)"}, "sO": {"op": ">=", "fO": "A(1)", "sO": "A(3)"}};A(1)
```

Рис. 2. Описание структуры Q-детерминанта алгоритма нахождения максимального числа в последовательности чисел

```
X(1)={"op": "!=", "fO": "A(1,1)", "sO": "0"};{"op": "-", "fO": {"op": "/", "fO": "A(1,3)", "sO": "A(1,1)"}, "sO": {"op": "*", "fO":
{"op": "/", "fO": {"op": "-", "fO": "A(2,3)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,3)", "sO": "A(1,1)"}, "sO": "A(2,1)"}},
"sO": {"op": "-", "fO": "A(2,2)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,2)", "sO": "A(1,1)"}, "sO": "A(2,1)"}}}, "sO": {"op":
"/", "fO": "A(1,2)", "sO": "A(1,1)"}}}
X(2)={"op": "!=", "fO": "A(1,1)", "sO": "0"};{"op": "/", "fO": {"op": "-", "fO": "A(2,3)", "sO": {"op": "*", "fO": {"op": "/", "fO":
"A(1,3)", "sO": "A(1,1)"}, "sO": "A(2,1)"}}, "sO": {"op": "-", "fO": "A(2,2)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,2)",
"sO": "A(1,1)"}, "sO": "A(2,1)"}}}
X(1)={"op": "==", "fO": "A(1,1)", "sO": "0"};{"op": "/", "fO": {"op": "-", "fO": "A(2,3)", "sO": {"op": "*", "fO": {"op": "/", "fO":
"A(1,3)", "sO": "A(1,2)"}, "sO": "A(2,2)"}}, "sO": {"op": "-", "fO": "A(2,1)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,1)",
"sO": "A(1,2)"}, "sO": "A(2,2)"}}}
X(2)={"op": "==", "fO": "A(1,1)", "sO": "0"};{"op": "-", "fO": {"op": "/", "fO": "A(1,3)", "sO": "A(1,2)"}, "sO": {"op": "*", "fO":
{"op": "/", "fO": {"op": "-", "fO": "A(2,3)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,3)", "sO": "A(1,2)"}, "sO": "A(2,2)"}},
"sO": {"op": "-", "fO": "A(2,1)", "sO": {"op": "*", "fO": {"op": "/", "fO": "A(1,1)", "sO": "A(1,2)"}, "sO": "A(2,2)"}}}, "sO": {"op":
"/", "fO": "A(1,1)", "sO": "A(1,2)"}}}
```

Рис. 3. Описание структуры Q-детерминанта метода Гаусса–Жордана

```
X(1)={"op": "<", "fO": {"op": "+", "fO": {"op": "abs", "od": {"op": "-", "fO": "X0(1)", "sO": {"op": "/", "fO": {"op": "-",
"fO": "B(1)", "sO": {"op": "*", "fO": "A(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}}}, "sO": {"op": "abs", "od": {"op": "-",
"fO": "X0(2)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0(1)"}}, "sO":
"A(2,2)"}}}}, "sO": "e"};{"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO": {"op": "*", "fO": "A(1,2)", "sO": "X0(2)"}},
"sO": "A(1,1)"}
X(2)={"op": "<", "fO": {"op": "+", "fO": {"op": "abs", "od": {"op": "-", "fO": "X0(1)", "sO": {"op": "/", "fO": {"op": "-",
"fO": "B(1)", "sO": {"op": "*", "fO": "A(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}}}, "sO": {"op": "abs", "od": {"op": "-",
"fO": "X0(2)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0(1)"}}, "sO":
"A(2,2)"}}}}, "sO": "e"};{"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0(1)"}},
"sO": "A(2,2)"}
X(1)={"op": "<", "fO": {"op": "+", "fO": {"op": "abs", "od": {"op": "-", "fO": {"op": "/", "fO": {"op": "-", "fO": "B(1)",
"sO": {"op": "*", "fO": "A(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}, "sO": {"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO":
{"op": "*", "fO": "A(1,2)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0
(1)"}}, "sO": "A(2,2)"}}}, "sO": "A(1,1)"}}}, "sO": {"op": "abs", "od": {"op": "-", "fO": {"op": "/", "fO": {"op": "-",
"fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0(1)"}}, "sO": "A(2,2)"}, "sO": {"op": "/", "fO": {"op": "-", "fO":
"B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO": {"op": "*", "fO": "A
(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}}}, "sO": "A(2,2)"}}}}, "sO": "e"};{"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO":
{"op": "*", "fO": "A(1,2)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0
(1)"}}, "sO": "A(2,2)"}}}, "sO": "A(1,1)"}
X(2)={"op": "<", "fO": {"op": "+", "fO": {"op": "abs", "od": {"op": "-", "fO": {"op": "/", "fO": {"op": "-", "fO": "B(1)",
"sO": {"op": "*", "fO": "A(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}, "sO": {"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO":
{"op": "*", "fO": "A(1,2)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0
(1)"}}, "sO": "A(2,2)"}}}, "sO": "A(1,1)"}}}, "sO": {"op": "abs", "od": {"op": "-", "fO": {"op": "/", "fO": {"op": "-",
"fO": "B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": "X0(1)"}}, "sO": "A(2,2)"}, "sO": {"op": "/", "fO": {"op": "-", "fO":
"B(2)", "sO": {"op": "*", "fO": "A(2,1)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO": {"op": "*", "fO": "A
(1,2)", "sO": "X0(2)"}}, "sO": "A(1,1)"}}}, "sO": "A(2,2)"}}}}, "sO": "e"};{"op": "/", "fO": {"op": "-", "fO": "B(2)", "sO":
{"op": "*", "fO": "A(2,1)", "sO": {"op": "/", "fO": {"op": "-", "fO": "B(1)", "sO": {"op": "*", "fO": "A(1,2)", "sO": "X0
(2)"}}, "sO": "A(1,1)"}}}, "sO": "A(2,2)"}
```

Рис. 4. Описание структуры Q-детерминанта метода Якоби (две итерации)

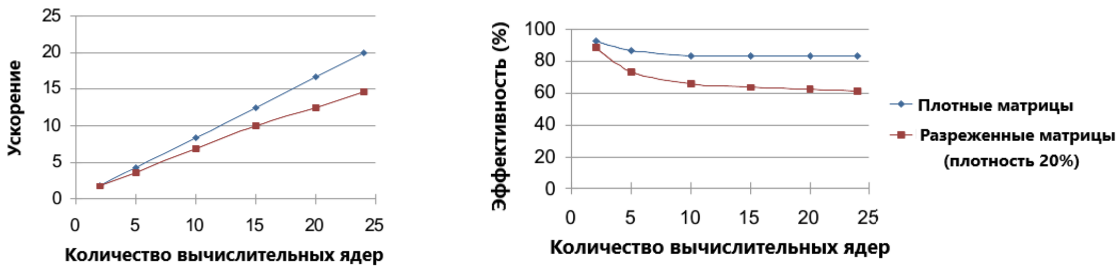

Рис. 5. Графики, демонстрирующие ускорение (слева) и эффективность (справа) Q-эффективных программ для алгоритмов умножения плотных и разреженных матриц для общей памяти

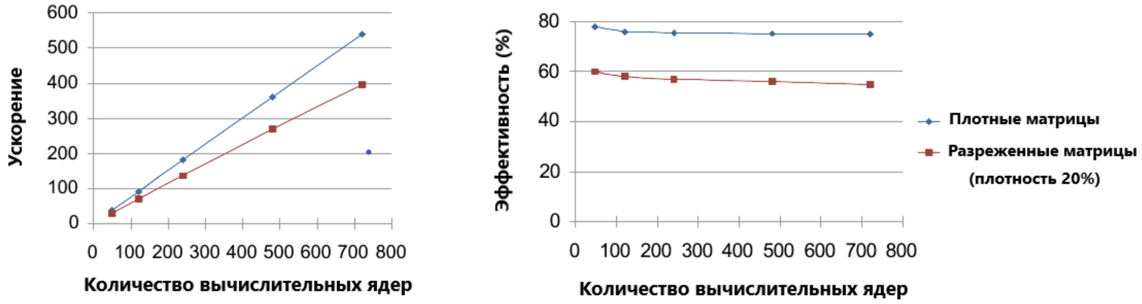

Рис. 6. Графики, демонстрирующие ускорение (слева) и эффективность (справа) Q-эффективных программ для алгоритмов умножения плотных и разреженных матриц для распределенной памяти

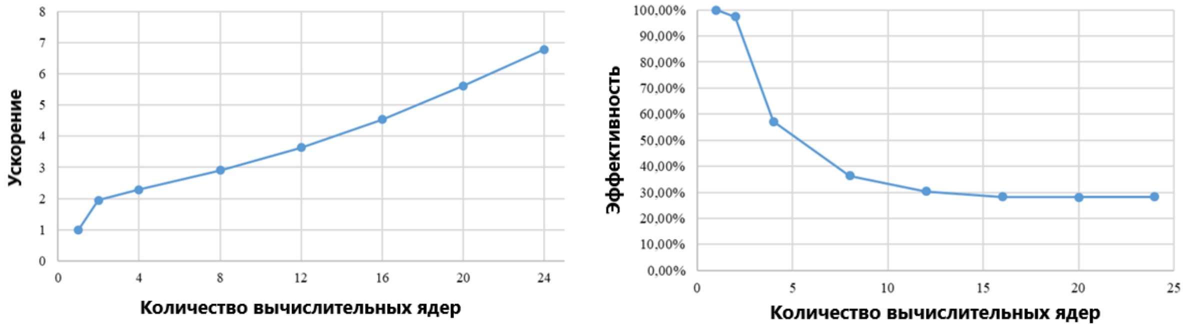

Рис. 7. Графики, демонстрирующие ускорение (слева) и эффективность (справа) Q-эффективной программы для метода Гаусса–Зейделя для общей памяти

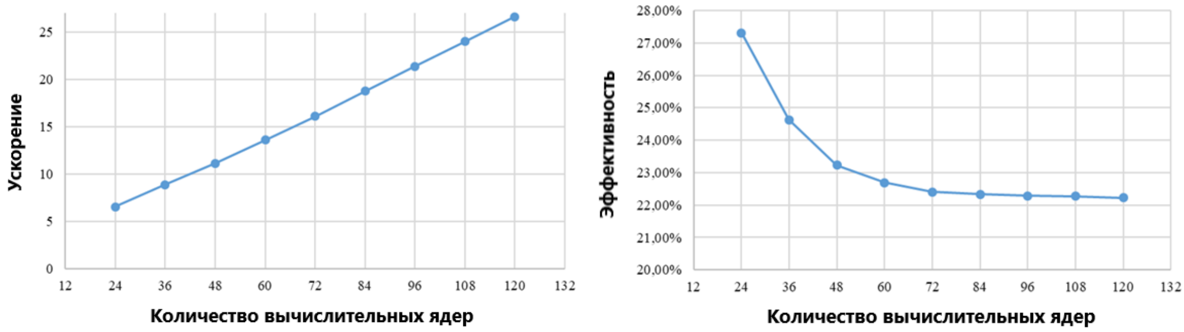

Рис. 8. Графики, демонстрирующие ускорение (слева) и эффективность (справа) Q-эффективной программы для метода Гаусса–Зейделя для распределенной памяти

5. Заключение

В работе описаны решения для исследования ресурса параллелизма любого численного алгоритма и для разработки параллельных программ, использующих ресурс параллелизма численных алгоритмов полностью, а также показано практическое применение предложенных решений. В том числе описано, какое развитие за последнее время получила Q-система для выявления внутреннего параллелизма численных алгоритмов на основе концепции Q-детерминанта, а также приведены результаты дальнейшей апробации метода проектирования Q-эффективных программ.

Полученные результаты исследований позволяют сделать следующие важные выводы.

1. Q-детерминант дает возможность выразить и оценить внутренний параллелизм любого численного алгоритма, а также показать возможный способ его параллельного исполнения.
2. Как блок-схема алгоритма облегчает труд разработчика при создании последовательной программы, так аналогично Q-детерминант численного алгоритма помогает разработчику создать программу для выполнения Q-эффективной реализации алгоритма. В результате становится возможной эффективная реализация численных алгоритмов на реальных ПВС.

Q-детерминант содержит все машинно-независимые свойства алгоритма, поэтому впоследствии функциональность Q-системы может быть дополнена с целью исследования различных машинно-независимых свойств численных алгоритмов.

Расширенная модель концепции Q-детерминанта может быть расширена и далее путем добавления моделей параллельных вычислений, ориентированных на вычислительные системы с другими архитектурными особенностями. Также для разработки Q-эффективных программ могут использоваться различные языки программирования и различные технологии параллельного программирования. Таким образом, для одного численного алгоритма может быть разработано потенциально бесконечное множество Q-эффективных программ. Каждая из этих программ будет выполнять Q-эффективную реализацию алгоритма, которая, с формальной точки зрения, является максимально быстрой реализацией алгоритма. По-видимому, для каждого алгоритма среди всех Q-эффективных программ не существует лучшей по быстродействию, но каждая из Q-эффективных программ наиболее эффективна для той вычислительной инфраструктуры, для которой она создавалась. На наш взгляд особую значимость в перспективе имеет решение проблемы автоматизированного проектирования Q-эффективных программ.

Литература

1. Voevodin V.V., Voevodin V.V. The V-Ray technology of optimizing programs to parallel computers. In: Vulkov L.G., Yalamov P., Wasniewski J. (eds.) WNAA 1996. LNCS, vol. 1196, pp. 546–556. Springer, Heidelberg (1997).

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

3. Открытая энциклопедия параллельных алгоритмических функций. URL: http://algowiki-project.org/en/Open_Encyclopedia_of_Parallel_Algorithmic_Features (дата обращения: 22.01.2019).

4. Абрамов С.М., Адамович А.И., Коваленко М. Р. Т-система – среда программирования с поддержкой автоматического динамического распараллеливания программ. Пример реализации алгоритма построения изображений методом трассировки лучей // Программирование. 1999. 25 (2). С. 100–107.

5. Абрамов С.М., Васенин В.А., Мамчиц Е.Е., Роганов В.А., Слепухин А.Ф. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Научная сессия МИФИ–2001, 22–26 января 2001 г.: Сборник научных трудов. Т. 2. 2001. С. 234.

6. Malyshkin V.E., Perepelkin V.A., Schukin G.F. Distributed Algorithm of Data Allocation in the Fragmented Programming // Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS 9251. P. 80–85. DOI: 10.1007/978-3-319-21909-7_8.

7. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. Control Flow Usage to Improse Performanse of Fragmented // Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS 9251. P. 86–90. DOI: 10.1007/978-3-319-21909-7_9.

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

9. Gurieva Y.L., Il'in V.P. On Parallel Computational Technologies of Augmented Domain Decomposition Methods// Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland, 2015. LNCS 9251. P. 35–46. DOI: 10.1007/978-3-319-21909-7_4.

10. Suplatov D.A., Voevodin V.V., Svedas V.K. Robust enzyme design: Bioinformatic tools for improved protein stability// Biotechnology journal. – Wiley - VCH Verlag GmbH & CO. KGaA (Germany). 2015. V. 10, № 3. С. 344–355. DOI: 10.1002/biot.201400150.

11. Schlueter M., Munetomo M.. Parallelization strategies for evolutionary algorithms for MINLP. In: IEEE Congress on Evolutionary Computation, 2013, P. 635–641.

12. Wang Q., Liu J., Tang X., Wang F., Fu G., Xing Z. Accelerating embarrassingly parallel algorithm on Intel MIC // IEEE International Conference on Progress in Informatics and Computing, 2014. P. 213–218.

13. Li Y., Dou W., Yang K., Miao S. 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.

14. You J., Kezhang H., Liang H., Xiao B. Research on parallel algorithms for calculating static characteristics of electromagnetic relay // IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), 2016. P. 1421–1425.

15. Prifti V., Bala R., Tafa I., Saatciu D., Fejzaj J. The time profit obtained by parallelization of quicksort algorithm used for numerical sorting // Science and Information Conference (SAI), 2015. P. 897–901.

16. Rajashri A. Parallelization of shortest path algorithm using OpenMP and MPI. In: International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017. P. 304–309.

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

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

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

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

22. Gropp, W., Lusk E., Skjellum A. Using MPI: portable parallel programming with the message-passing interface. – Second Edi. – MIT Press, 1999. 371 p.

23. Багаутдинов А.Р. Разработка методов исследования параллелизма алгоритмов на основе концепции Q-детерминанта и их программная реализация: Вып. квалиф. работа магистра по направлению «Фундаментальная информатика и информационные технологии»: 02.04.02 / Южно-Уральский государственный университет. Челябинск, 2017. 32 л. URL: http://omega.sp.susu.ru/publications/masterthesis/17-Bagautdinov.pdf (дата обращения: 23.01.2019).

24. Алеева В.Н., Иванов Н.А. Исследование внутреннего параллелизма численных алгоритмов. Параллельные вычислительные технологии – XII международная конференция, ПаВТ'2018, г. Ростов-на-Дону, 2–6 апреля 2018 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2018. С. 224–234.

25. Валькевич Н.В. Q-эффективная реализация алгоритма умножения матриц на суперкомпьютере «Торнадо ЮУрГУ»: Вып. квалиф. работа бакалавра по направлению «Фундаментальная информатика и информационные технологии»: 02.03.02/ Южно-Уральский государственный университет. Челябинск, 2017. 33 л. URL: http://omega.sp.susu.ru/publications/bachelorthesis/17-Valkevich.pdf (дата обращения: 21.01.2019).

26. Тарасов Д.Е. Q-эффективный кодизайн реализации метода Гаусса–Жордана на суперкомпьютере «Торнадо ЮУрГУ»: Вып. квалиф. работа магистра по направлению «Фундаменталь-

ная информатика и информационные технологии»: 02.04.02 / Южно-Уральский государственный университет. Челябинск, 2017. 41 л. URL: http://omega.sp.susu.ru/publications/masterthesis/17-Tarasov.pdf (дата обращения: 22.01.2019).

27. Лаптева Ю.С. Q-эффективная реализация метода Якоби для решения СЛАУ на суперкомпьютере «Торнадо ЮУрГУ»: Вып. квалиф. работа бакалавра по направлению «Фундаментальная информатика и информационные технологии»: 02.03.02 / Южно-Уральский государственный университет. Челябинск, 2017. 30 л. URL: http://omega.sp.susu.ru/publications/bachelorthesis/17-Lapteva.pdf (дата обращения: 21.01.2019).

28. Нечепоренко А.Д. Разработка Q-эффективной программы для решения СЛАУ методом Гаусса–Зейделя: Вып. квалиф. работа бакалавра по направлению «Фундаментальная информатика и информационные технологии»: 02.03.02 / Южно-Уральский государственный университет. Челябинск, 2018. 32 л. URL: http://omega.sp.susu.ru/publications/bachelorthesis/18-Necheporenko.pdf (дата обращения: 21.01.2019).

29. Баженова Л.А. Применение метода проектирования Q-эффективной программы для решения системы сеточных уравнений: Вып. квалиф. работа бакалавра по направлению «Фундаментальная информатика и информационные технологии»: 02.03.02 / Южно-Уральский государственный университет. Челябинск, 2018. 30 л. URL: http://omega.sp.susu.ru/publications/bachelorthesis/18-Bazhenova.pdf (дата обращения: 24.01.2019).