Исследование внутреннего параллелизма численных алгоритмов*

В.Н. Алеева, Н.А. Иванов

Южно-Уральский государственный университет (НИУ)

Исследование внутреннего параллелизма численных алгоритмов с целью его использования для эффективной реализации алгоритмов на параллельных вычислительных системах является актуальной задачей. Особый интерес представляет проведение такого исследования в автоматизированном режиме. В работе описан прототип программной системы для исследования внутреннего параллелизма численных алгоритмов. Для разработки программной системы использована концепция Q-детерминанта, которая основана на универсальном описании численных алгоритмов в форме Q-детерминанта, показывающем ресурс параллелизма в полной мере. Программная система дает возможность для любого численного алгоритма оценить такие характеристики параллельной сложности, как высота и ширина алгоритма, а также из множества численных алгоритмов, решающих одну и ту же алгоритмическую проблему, выбрать алгоритм с лучшим ресурсом внутреннего параллелизма.

Ключевые слова: Q-детерминант алгоритма, представление алгоритма в форме Q-детерминанта, Q-эффективная реализация алгоритма, ресурс параллелизма алгоритма, внутренний параллелизм алгоритма, высота алгоритма, ширина алгоритма.

1. Введение

Эффективность выполнения алгоритмов на параллельных вычислительных системах (ВС) остается низкой. Повышение эффективности приведет к увеличению быстродействия параллельных ВС. Одним из способов повышения эффективности выполнения алгоритмов является использование ресурса внутреннего параллелизма. В связи с этим проблема разработки инструментальных средств для исследования внутреннего параллелизма алгоритмов в автоматизированном режиме является актуальной, а ее решение имеет научное и практическое значение.

Цель данной работы – показать возможность создания программной системы для выявления внутреннего параллелизма численных алгоритмов на основе концепции Q-детерминанта. Для достижения цели разработаны методы исследования ресурса параллелизма численных алгоритмов, выполнена их программная реализация, проведено экспериментальное исследование разработанного программного обеспечения. Предлагаемые методы и основанные на них алгоритмы являются новыми и имеют научную значимость.

Статья состоит из четырех разделов и заключения. В первом разделе приведены методы исследования ресурса параллелизма численных алгоритмов на основе концепции Q-детерминанта [1]. Второй раздел содержит описание программной реализации предлагаемых методов. В третьем разделе приводятся результаты экспериментов, подтверждающих адекватность и эффективность предлагаемых методов. В четвертом разделе дан обзор основных научных направлений, близких к тематике данного исследования. Заключение содержит краткое описание полученных результатов, описание перспектив их использования и направления дальнейшего исследования.

2. Методы исследования ресурса параллелизма численных алгоритмов

Методы исследования ресурса параллелизма численных алгоритмов основаны на концепции Q-детерминанта, поэтому кратко изложим используемые понятия концепции.

*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).

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

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

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

Предположим, что I_1, I_2, I_3 – подмножества множества $I = (1,...,m)$ такие, что: одно или два из множеств $I_i\ (i = 1,2,3)$ могут быть пустыми, $I_1 \cup I_2 \cup I3 = I$, $I_i \cap I_j = \varnothing\ (i \neq j; i, j = 1,2,3)$. Предположим также, что множество Q-термов $\{f_i\}_{i \in I}$ удовлетворяет условиям:

1. $f_{i_1}\ (i_1 \in I_1)$ – безусловный Q-терм, $f_{i_1} = w^{i_1}$;
2. $f_{i_2}\ (i_2 \in I_2)$ – условный Q-терм, $f_{i_2} = \{(u_j^{i_2}, w_j^{i_2})\}_{j=1,...,l_{i_2}}$, l_{i_2} является вычислимой функцией параметров N;
3. $f_{i_3}\ (i_3 \in I_3)$ – условный бесконечный Q-терм, $f_{i_3} = \{(u_j^{i_3}, w_j^{i_3})\}_{j=1,2,...}$.

Если алгоритм α состоит в том, что для определения $y_i\ (i \in I)$ требуется вычислить Q-терм f_i, то множество Q-термов $y_i\ (i \in I)$ называется Q-детерминантом алгоритма α, а представление алгоритма в виде $y_i = f_i\ (i \in I)$ представлением в форме Q-детерминанта.

Реализацией алгоритма α, представленного в форме Q-детерминанта $y_i = f_i\ (i \in I)$, называется вычисление Q-термов $f_i\ (i \in I)$ при заданной интерпретации B и некотором $\overline{N} \in \{\overline{N}\}$. Если реализация алгоритма такова, что выражения, составляющие множество $W(\overline{N}) = \{w^{i_1}(\overline{N})\ (i_1 \in I_1); u_j^{i_2}(\overline{N}), w_j^{i_2}(\overline{N})\ (i_2 \in I_2, j = 1,...,l_{i_2}); u_j^{i_3}(\overline{N}), w_j^{i_3}(\overline{N})\ (i_3 \in I_3, j = 1,2,...)\}$,

вычисляются одновременно и при их вычислении операции выполняются по мере готовности, то реализация называется Q-эффективной. Если алгоритм допускает распараллеливание, то его Q-эффективная реализация полностью использует ресурс внутреннего параллелизма алгоритма, поэтому является максимально параллельной реализаций алгоритма.

Выражение имеет уровни вложенности. Будем обозначать число уровней вложенности выражения w через T^w.

Уровни вложенности выражения, его подвыражений и операций определяются индуктивно:

1. Константы и элементы B имеют нулевой уровень вложенности.
2. Если w выражение, то $T^w = T^{(w)}$.
3. Если w – выражение, $T^w = i-1$ $(i \geq 1)$ и $f \in Q$ – одноместная операция, то $T^{f(w)} = i$, а w называется подвыражением $(i-1)$-го уровня вложенности выражения $f(w)$. В этом случае $f \in Q$ называется операцией i-го уровня вложенности.
4. Если w и v – выражения, $T^w = i$, $T^v = j$ и $g \in Q$ – двуместная операция, то $T^{g(w,v)} = k$, где $k = \max(i, j)+1$. Выражения w и v называются подвыражениями $g(w,v)$ соответственно i и j уровней вложенности, а операция g называется операцией k-го уровня вложенности.

Выражение, образованное из n выражений путем применения без использования скобок $n-1$ раз одной из ассоциативных операций множества Q, будем называть цепочкой выражений длины n. При определении уровня вложенности выражения и его подвыражений порядок выполнения операций цепочки задается по схеме сдваивания.

Реализация алгоритма α называется выполнимой, если одновременно необходимо выполнять конечное число операций. Существуют алгоритмы, для которых Q-эффективная реализация не является выполнимой.

Введем характеристики параллельной сложности выполнимой Q-эффективной реализации алгоритма α:

1. $D_\alpha(\overline{N})$ – максимальное число уровней вложенности выражений множества $W(\overline{N})$, т.е. $D_\alpha(\overline{N}) = \max\limits_{w(\overline{N}) \in W(\overline{N})} T^{w(\overline{N})}$;
2. $P_\alpha(\overline{N})$ – максимальное из значений, выражающих количество операций на каждом из уровней вложенности всех выражений множества $W(\overline{N})$, т.е. $P_\alpha(\overline{N}) = \max\limits_{1 \leq r \leq D_\alpha(\overline{N})} \sum\limits_{w(\overline{N}) \in W(\overline{N})} O_r^{w(\overline{N})}$, где $O_r^{w(\overline{N})}$ – количество операций уровня вложенности r выражения $w(\overline{N})$.

$D_\alpha(\overline{N})$ является верхней оценкой высоты алгоритма, $P_\alpha(\overline{N})$ – верхней оценкой его ширины. $D_\alpha(\overline{N})$ характеризует быстродействие алгоритма, $P_\alpha(\overline{N})$ – количество процессоров, необходимое для реализации алгоритма.

Для исследования ресурса параллелизма численных алгоритмов были разработаны следующие методы:

1. метод построения Q-детерминанта алгоритма на основе его блок-схемы;
2. метод получения Q-эффективной реализации алгоритма на основе Q-детерминанта;
3. метод вычисления характеристик параллельной сложности Q-эффективной реализации алгоритма;
4. метод сравнения характеристик параллельной сложности Q-эффективных реализаций алгоритмов.

Численный алгоритм редко представлен в форме Q-детерминанта, поэтому необходим метод построения Q-детерминанта на основе общепринятого представления алгоритма, например, блок-схемы. Для разработки метода построения Q-детерминанта алгоритма на основе блок-схемы было рассмотрено множество различных алгоритмов и проанализированы особенности, связанные с реализацией метода.

Особенность алгоритмов, Q-детерминант которых состоит из безусловных Q-термов, заключается в том, что прохождение по блок-схеме можно осуществлять последовательно, как если бы выполнялся рассматриваемый алгоритм. В формируемый Q-терм будет записано конечное выражение.

Для алгоритмов, Q-детерминант которых состоит из условных Q-термов, прохождение по блок-схеме разветвляется и каждая ветвь обрабатывается отдельно. По завершению обработки одной ветви обработчик блок-схемы возвращается на прежний блок и продолжает обработку уже с противоположным условием, записанным в данном блоке. Формируемый Q-терм в этом случае представляет собой список пар, состоящих из логического выражения и выражения.

То же самое касается и алгоритмов, Q-детерминант которых состоит из условных бесконечных Q-термов. Проблема хранения условных бесконечных Q-термов решается ограничением количества итераций. Предполагается, что для таких алгоритмов можно определить необходимое количество итераций.

Блок-схема обрабатывается сверху вниз. Такое направление обусловлено тем, что оно позволяет последовательно проходить по блок-схеме и следить за тем, как строится Q-детерминант. Анализ блок-схемы проводится при фиксированных параметрах размерности алгоритмической проблемы. Результатом применения к численному алгоритму описанного метода является множество Q-термов $f_i\,(i \in I)$ для некоторого фиксированного значения $\overline{N} \in \{\overline{N}\}$. Следовательно, будет сформировано множество выражений $W(\overline{N})$. Подробно метод построения Q-детерминанта алгоритма на основе его блок-схемы описан в работе [3].

Метод получения Q-эффективной реализации алгоритма на основе Q-детерминанта состоит в том, что, используя определение уровня вложенности операции, можно вычислить уровни вложенности операций, входящих в выражения множества $W(\overline{N})$. Множество выражений $W(\overline{N})$ с вычисленными уровнями вложенности операций представляет собой план выполнения Q-эффективной реализации алгоритма.

Метод вычисления характеристик параллельной сложности Q-эффективной реализации алгоритма заключается в том, что для вычисления $D_\alpha(\overline{N})$ находится максимальное значение уровня вложенности операций, входящих в выражения множества $W(\overline{N})$, а для определения $P_\alpha(\overline{N})$ нужно найти 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 вычислить максимальное из них.

Результаты вычислений, полученные с помощью описанных методов, предлагается хранить в базе данных. Таким образом, база данных должна содержать информацию об исследуемых алгоритмах и их Q-детерминанты для различных значений параметров размерности $\overline{N} \in \{\overline{N}\}$ вместе с характеристиками параллельной сложности Q-эффективной реализации $D_\alpha(\overline{N})$ и $P_\alpha(\overline{N})$.

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

3. Программная реализация методов исследования ресурса параллелизма численных алгоритмов

Была выполнена программная реализация метода построения Q-детерминанта алгоритма на основе блок-схемы. Для описания входных и выходных данных используется декларативный язык JSON. В соответствии с форматом JSON блок-схема представляется в виде описания блоков Vertices и соединений Edges. Блоки Vertices определяются номером Id, типом Type и тек-

стовым содержимым Content. Значения Type для блоков Vertices: 0 – блок «Начало»; 1 – блок «Конец»; 2 – блок присваивания; 3 – блок условия; 4 – блок ввода; 5 – блок вывода. Соединения Edges определяются номерами начальных и конечных блоков From и To и типом соединения Type. Значения Type для соединений Edges: 0 – проход по условию «нет»; 1 – проход по условию «да»; 2 – обычное соединение. Описание блок-схемы алгоритма умножения матриц в формате JSON представлено на рис. 1.

Программа реализована с использованием языка программирования C# на базе платформы .NET. Она разрабатывалась в соответствии с парадигмой объектно-ориентированного программирования. В качестве интегрированной среды разработки использовалась Microsoft Visual Studio. Подробно программная реализация метода построения Q-детерминанта алгоритма на основе его блок-схемы описана в работе [3].

Для разработки базы данных (БД) была выбрана система управления базами данных Microsoft SQL Server. БД содержит следующие сущности: algorithms – исследуемый алгоритм; determinants – Q-детерминант, составленный для определенных размерностей. Спецификации сущностей описаны в таблицах 1 и 2.

Таблица 1. Спецификация сущности algorithms

№	Атрибут	Семантика	Примечание	Тип данных
1	id	Уникальный идентификатор	Первичный ключ, является автоинкрементальным	int
2	name	Название алгоритма	Название алгоритма, которое выбирает пользователь	varchar(50)
3	description	Описание алгоритма	Краткое описание алгоритма	varchar(200)

Таблица 2. Спецификация сущности determinants

№	Атрибут	Семантика	Примечание	Тип данных
1	id	Уникальный идентификатор	Первичный ключ, является автоинкрементальным	int
2	algorithm	Уникальный идентификатор алгоритма	Внешний ключ к Algorithms.id; алгоритм, к которому относится данный Q-детерминант	int
3	expression	Q-детерминант	Q-детерминант, хранимый в формате JSON	text
4	ticks	Верхняя оценка высоты алгоритма	Значение $D_\alpha(\overline{N})$	int
5	processors	Верхняя оценка ширины алгоритма	Значение $P_\alpha(\overline{N})$	int
6	dimensions	Вектор значений параметров размерности алгоритмической проблемы, соответствующих Q-детерминанту	Описывается в формате JSON	text

```
{"Vertices":[
   {"Id":1,"Type":0,"Content":"Start"},
   {"Id":2,"Type":4,"Content":"l"},
   {"Id":3,"Type":4,"Content":"m"},
   {"Id":4,"Type":4,"Content":"n"},
   {"Id":5,"Type":4,"Content":"A[l,m]"},
   {"Id":6,"Type":4,"Content":"B[m,n]"},
   {"Id":7,"Type":2,"Content":"C[l,n]=0"},
   {"Id":8,"Type":2,"Content":"i=1"},
   {"Id":9,"Type":3,"Content":"i<=l"},
   {"Id":10,"Type":2,"Content":"j=1"},
   {"Id":11,"Type":3,"Content":"j<=n"},
   {"Id":12,"Type":2,"Content":"k=1"},
   {"Id":13,"Type":3,"Content":"k<=m"},
   {"Id":14,"Type":2,"Content":"C(i,j)=C(i,j)+A(i,k)*B(k,j)"},
   {"Id":15,"Type":2,"Content":"k=k+1"},
   {"Id":16,"Type":2,"Content":"j=j+1"},
   {"Id":17,"Type":2,"Content":"i=i+1"},
   {"Id":18,"Type":5,"Content":"C[l,n]"},
   {"Id":19,"Type":1,"Content":"End"}],
"Edges":[
   {"From":1,"To":2,"Type":2},   {"From":2,"To":3,"Type":2},
   {"From":3,"To":4,"Type":2},   {"From":4,"To":5,"Type":2},
   {"From":5,"To":6,"Type":2},   {"From":6,"To":7,"Type":2},
   {"From":7,"To":8,"Type":2},   {"From":8,"To":9,"Type":2},
   {"From":9,"To":10,"Type":1},  {"From":9,"To":18,"Type":0},
   {"From":10,"To":11,"Type":2}, {"From":11,"To":12,"Type":1},
   {"From":11,"To":17,"Type":0}, {"From":12,"To":13,"Type":2},
   {"From":13,"To":14,"Type":1}, {"From":13,"To":16,"Type":0},
   {"From":14,"To":15,"Type":2}, {"From":15,"To":13,"Type":2},
   {"From":16,"To":11,"Type":2}, {"From":17,"To":9,"Type":2},
   {"From":18,"To":19,"Type":2}]}
```


Рис. 1. Блок-схема алгоритма умножения матриц в формате JSON

Для взаимодействия с базой данных разработано серверное приложение – ASP.NET-приложение, реализующее RESTful-интерфейс. Задачи серверного приложения:

1. возможность получения, добавления, редактирования и удаления информации об алгоритмах;
2. возможность получения, добавления и удаления Q-детерминантов алгоритмов.

Данное приложение предоставляет интерфейс взаимодействия с сущностями algorithms и determinants.

Для взаимодействия с сущностью algorithms реализованы следующие методы:

1. GET algorithms/ – получение списка алгоритмов из базы данных с полной информацией о них;
2. POST algorithms/ – добавление нового алгоритма в базу данных, на вход подается JSON-документ с полями name и description;
3. DELETE algorithms/<идентификатор алгоритма> – удаление алгоритма с данным идентификатором из базы данных;
4. PUT algorithms/<идентификатор алгоритма> – обновление информации об алгоритме с данным идентификатором;
5. POST algorithms/<идентификатор алгоритма> – добавление в базу данных информации о Q-детерминанте, составленном для алгоритма с данным идентификатором, а также рассчитанных характеристик параллельной сложности Q-эффективной реализации алгоритма $D_\alpha(\overline{N})$ и $P_\alpha(\overline{N})$; на вход подается JSON – документ с полями Determinant и Dimensions; далее алгоритм расчета характеристик параллельной сложности Q-эффективной реализации алгоритма будет рассмотрен подробнее;
6. GET algorithms/<идентификатор алгоритма> – получение списка идентификаторов Q-детерминантов, составленных для алгоритма с данным идентификатором;
7. POST algorithms/<название атрибута>/commoncompare – сравнение характеристик параллельной сложности Q-эффективной реализации алгоритма с одинаковыми размерностями для двух алгоритмов с данными идентификаторами с указанием сравниваемого атрибута – ticks или processors; на вход подается JSON-документ с полями id1 и id2, которые содержат

идентификаторы сравниваемых алгоритмов; далее алгоритм сравнения будет рассмотрен подробнее.

Для взаимодействия с сущностью determinants реализованы следующие методы:

1. GET determinants/<идентификатор Q-детерминанта> – получение Q-детерминанта с данным идентификатором;
2. DELETE determinants/<идентификатор Q-детерминанта> – удаление Q-детерминанта с данным идентификатором;
3. GET determinants/<идентификатор Q-детерминанта>/ticks – получение характеристики $D_\alpha(\overline{N})$ Q-детерминанта с данным идентификатором;
4. GET determinants/<идентификатор Q-детерминанта>/processors – получение характеристики $P_\alpha(\overline{N})$ Q-детерминанта с данными идентификатором;
5. GET determinants/<идентификатор Q-детерминанта>/dimensions – получение вектора значений параметров размерности алгоритмической проблемы, соответствующих Q-детерминанту с данным идентификатором.

Алгоритм получения Q-эффективной реализации и вычисления характеристик параллельной сложности Q-эффективной реализации алгоритма состоит в следующем. На вход подается Q-детерминант в виде JSON-документа. Производится десериализация данного документа в специальную структуру, основанную на системе классов, соответствующих различным типам операций и исходных данных. При создании специальной структуры производится расчет такого массива списков, что каждый из списков содержит операции соответствующего уровня вложенности. Числа и переменные возвращают пустой массив. Унарные операции берут за основу массив операнда и добавляют новый список, содержащий данную операцию. Бинарные операции берут за основу массивы операндов, соединяют их и добавляют новый список, содержащий данную операцию. Пара из безусловного логического Q-терма и безусловного Q-терма берет за основу массивы операндов и соединяет их. Список Q-термов берет за основу массивы элементов, из которых состоит, и соединяет их. Полученный массив дает представление о плане выполнения Q-эффективной реализации алгоритма, то есть о том, на каком этапе какая операция должна выполняться. Значение $D_\alpha(\overline{N})$ равно длине массива, а $P_\alpha(\overline{N})$ – размеру самого большого списка в массиве.

Алгоритм сравнения характеристик параллельной сложности Q-эффективных реализаций двух алгоритмов заключается в следующем. На вход подаются идентификаторы сравниваемых алгоритмов. Выполнение алгоритма сравнения производится в три этапа. На первом этапе определяются идентификаторы Q-детерминантов, характеристики параллельной сложности которых будут сравниваться. Для этого производится поиск Q-детерминантов, составленных для сравниваемых алгоритмов, затем получение размерностей данных Q-детерминантов. Среди Q-детерминантов, составленных для первого алгоритма, производится поиск тех, размерности которых встречаются у Q-детерминантов, составленных для второго алгоритма. Идентификаторы пар соответствующих Q-детерминантов записываются в двумерный массив, строки в котором соответствуют отдельному алгоритму. На втором этапе определяются данные, на основе которых будет проводиться сравнение. Выполняется это так. Производится проход по элементам двумерного массива с идентификаторами Q-детерминантов. В другой массив того же размера записываются соответствующие значения атрибута, для которого выполняется сравнение ($D_\alpha(\overline{N})$ или $P_\alpha(\overline{N})$). На третьем этапе сравнение завершается. Если двумерный массив с данными, полученный на втором этапе, пустой, то возвращается ошибка. В противном случае производится проход по столбцам двумерного массива с данными, из значения первой строки вычитается значение второй строки, результат суммируется и в конце возвращается.

Также было разработано клиентское Windows-приложение, которое управляет БД с помощью вызова методов серверного приложения. Окно приложения содержит три вкладки: «Алгоритмы», «Q-детерминанты» и «Сравнение». Во вкладке «Алгоритмы» расположена таблица, содержащая результат запроса всех алгоритмов, а также расположены элементы интерфейса, позволяющие получать, добавлять, обновлять и удалять алгоритмы. Во вкладке «Q-детерминанты» расположена таблица, содержащая результат запроса всех Q-детерминантов, составленных для выбранного алгоритма, а также расположены элементы интерфейса, позво-

ляющие выбирать алгоритм и получать, добавлять и удалять составленные для него Q-детерминанты. Во вкладке «Сравнение» расположены элементы интерфейса, позволяющие выбрать алгоритмы для сравнения и получить результат их сравнения.

4. Результаты экспериментов

Программное обеспечение, реализующее метод построения Q-детерминанта алгоритма на основе блок-схемы, протестировано на алгоритмах, Q-детерминанты которых содержат различные типы Q-термов. Подробно проведенные вычислительные эксперименты описаны в работе [3]. Работу программного обеспечения, реализующего другие описанные методы, покажем на примере двух простых алгоритмов для вычисления скалярного произведения векторов: с последовательным сложением произведений компонент векторов и со сложением по схеме сдваивания. Для каждого из алгоритмов было построено по три Q-детерминанта, соответствующих значениям параметра размерности 4, 7 и 11. Для записи информации об алгоритмах в базу данных была использована система элементов управления во вкладке «Алгоритмы» (рис. 2). Чтобы записать построенные для алгоритмов Q-детерминанты в базу данных, использовалась система элементов управления во вкладке «Q-детерминанты» (рис. 3). Сравнение характеристик параллельной сложности Q-эффективных реализаций алгоритмов осуществлялось с помощью системы элементов управления во вкладке «Сравнение алгоритмов» (рис. 4). Данный эксперимент показывает, что из двух сравниваемых алгоритмов скалярного произведения векторов алгоритм со сложением произведений компонент векторов по схеме сдваивания имеет высоту меньше, а ширина алгоритмов одинакова.

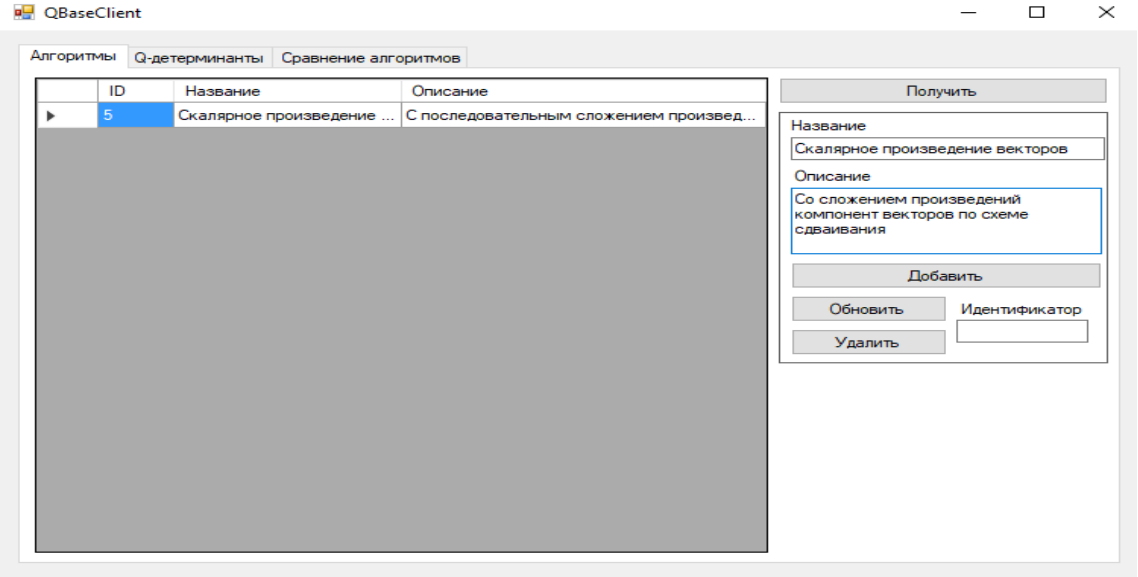

Рис. 2. Запись информации об алгоритмах в базу данных

5. Обзор близких по тематике научных исследований

Обозначим основные научные направления, близкие к тематике данного исследования. Очень важным и развитым направлением исследований является исследование параллельной структуры алгоритмов и программ с целью их реализации на параллельных ВС. В частности, благодаря этому направлению стало возможным использовать для параллельных ВС накопленный фонд последовательных программ. Основа направления описана в публикациях [4, 5]. Для исследования параллельных алгоритмов используется их представление с помощью графов. Направление активно развивается. В настоящее время для описания свойств, особенностей, статических и динамических характеристик алгоритмов с целью эффективной реализации алгоритмов на параллельных вычислительных системах создается Интернет-энциклопедия AlgoWiki [6]. Современное состояние исследований в области распараллеливания алгоритмов и

их реализации на параллельных вычислительных системах представлено в докладе Вл.В. Воеводина [7].

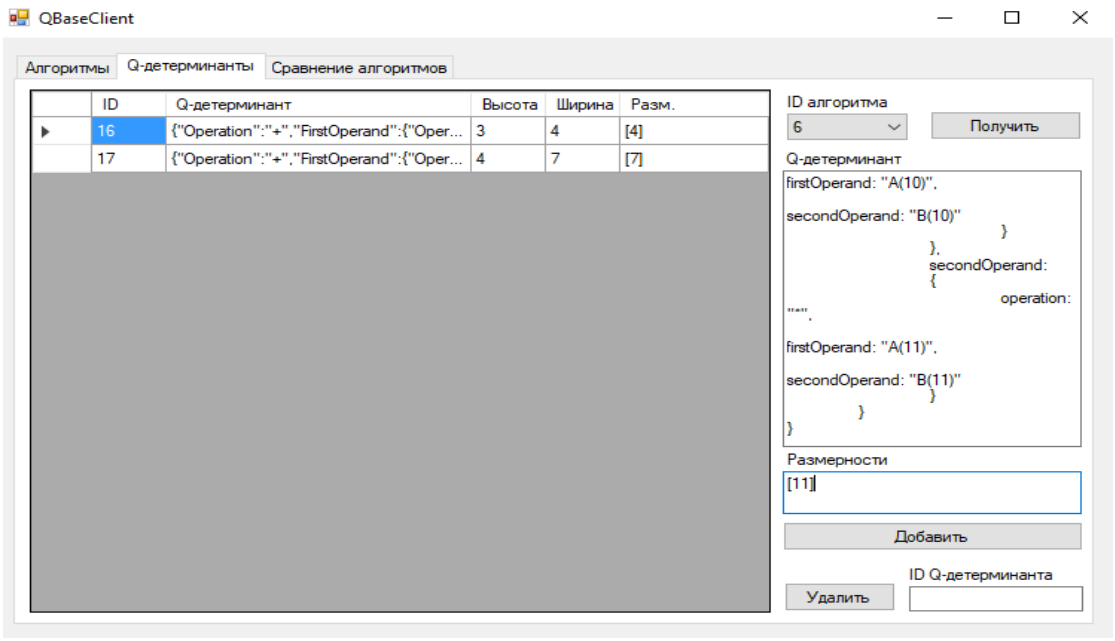

Рис. 3. Запись Q-детерминантов алгоритмов в базу данных

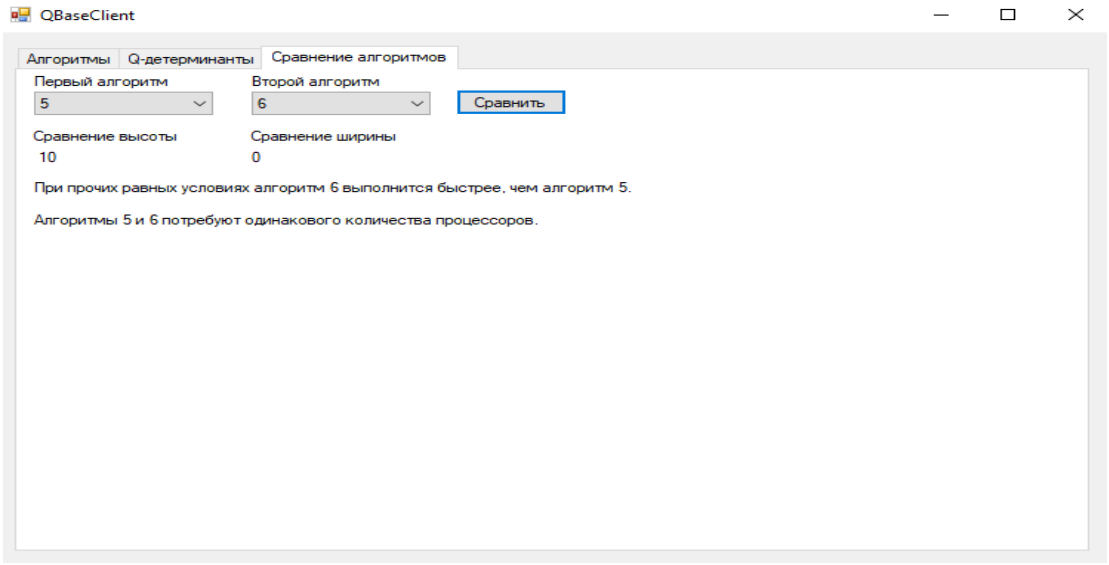

Рис. 4. Сравнение характеристик параллельной сложности Q-эффективных реализаций алгоритмов

Многочисленными являются исследования, в которых при разработке параллельных программ учитывается специфика алгоритмов и архитектуры параллельных ВС. В качестве примеров таких исследований можно привести [8–13]. Подобные исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на параллельных ВС конкретной архитектуры, однако не предлагают универсального подхода. Еще один подход к созданию параллельных программ - синтез параллельных программ. Метод синтеза параллельных программ был впервые описан в публикации [14]. Основанный на нем синтез параллельных программ заключается в том, чтобы из базы знаний существующих реализованных 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 [15, 16]. Подход является универ-

сальным, но проблему исследования ресурса параллелизма алгоритмов не решает. С помощью программной реализации алгоритмов проводятся исследования их ресурса параллелизма, например, данному направлению посвящены работы [17, 18]. Если решать задачу выявления максимального ресурса параллелизма алгоритма, то использование программы, реализующей алгоритм, может оказаться не выгодным и даже ошибочным, поскольку программа может не содержать всех реализаций алгоритма, в частности, при ее создании могла быть потеряна реализация, которая полностью использует ресурс параллелизма алгоритма.

Анализ показывает, что существующие подходы при решении проблемы исследования ресурса параллелизма алгоритмов и его реализации на параллельных вычислительных системах либо неприменимы, либо малоэффективны, либо не являются универсальными. Перспективным является подход, основанный на универсальном описании алгоритма, показывающем ресурс параллелизма в полной мере. В данной работе предлагается такой подход для решения проблемы исследования ресурса параллелизма численных алгоритмов. Подход использует концепцию Q-детерминанта.

6. Заключение

В работе показана возможность создания программной системы для выявления внутреннего параллелизма численных алгоритмов на основе концепции Q-детерминанта. Приведено описание методов исследования ресурса параллелизма численных алгоритмов, их программной реализации и вычислительных экспериментов, подтверждающих адекватность и эффективность предлагаемых методов и программной реализации. В качестве теоретической основы при исследовании используется концепция Q-детерминанта. Разработка программной системы для исследования внутреннего параллелизма численных алгоритмов на основе концепции Q-детерминанта выполняется впервые.

В настоящее время разработан прототип программной системы. Для его дальнейшего тестирования и доведения до рабочей версии планируется использовать алгоритмы для решения задач линейной алгебры, имеющие сложную структуру Q-детерминанта. В результате разработчик прикладного программного обеспечения для параллельных ВС получит инструментальное средство, которое по блок-схеме численного алгоритма оценивает его высоту и ширину, а также из множества численных алгоритмов, решающих одну и ту же алгоритмическую проблему, находит алгоритм с лучшей характеристикой параллельной сложности.

Q-детерминант алгоритма содержит все машинно-независимые свойства алгоритма. Так как разрабатываемая программная система строит Q-детерминант любого численного алгоритма, то с ее помощью можно автоматизировать исследование различных свойств численных алгоритмов путем разработки соответствующих функций системы.

Литература

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

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

3. 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: 23.01.2018).

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

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

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

7. Воеводин Вл.В. Параллельные алгоритмы под микроскопом // Доклад на международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2016». URL: http://omega.sp.susu.ru/books/conference/PaVT2016/talks/Voevodin.pdf (дата обращения: 25.01.2018).

8. Воеводин Вад.В., Овчинников С.Л., Романов С.Ю. Разработка высокоэффективных масштабируемых программ в задаче ультразвуковой томографии // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). М.: Изд-во МГУ, 2012. Том 13. С. 307–315.

9. Гервич Л.Р., Кравченко Е.Н., Штейнберг Б.Я., Юрушкин М.В. Автоматизация распараллеливания программ с блочным размещением данных // Сибирский журнал вычислительной математики. 2015. Т. 18, № 1. С. 41–53.

10. Климов Ю.А., Орлов А.Ю., Шворин А.Б. Перспективные подходы к созданию масштабируемых приложений для суперкомпьютеров гибридной архитектуры // Информационные технологии и вычислительные системы. 2012. №2. С. 3–10.

11. 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.

12. 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. Vol. 10, No. 3. C. 344–355. DOI: 10.1002/biot.201400150.

13. Venkata M.G., Shamis P., Sampath R., Graham R.L., Ladd J.S. Optimizing blocking and nonblocking reduction operations for multicore systems: hierarchical design and implementation // Proceedings of IEEE Cluster. 2013. P. 1–8. DOI: 10.1109/cluster.2013.6702676.

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

15. 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.

16. Malyshkin V.E., Perepelkin V.A., and 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.

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

18. Легалов А. И., Ледяев Д. Н., Анкудинов А. В. Инструментальная поддержка многокритериального анализа при разработке сложных технических систем // Вестник Сибирского государственного аэрокосмического университета. 2009. № 2 (23). С. 50–55.